Claudio Neroni e Riccardo Battistutta

Corso di Programmazione applicativa su AS/400 in Rpg Ile

Ultima revisione 10 novembre 2008

Altre modifiche in corso!

Postfazione

Abbiamo scritto il presente manuale per fare da traccia ad un corso introduttivo alla programmazione RpgIle su AS/400. Esso non sostituisce nessuno dei manuali in dotazione al software di base di tale macchina. È piuttosto la sintesi delle esperienze di addestramento di numerosi programmatori a partire dall'ormai lontano 1980, anno dell'apparizione della prima versione di questo software sul mitico S/38.

Gli argomenti trattati sono un sottoinsieme di tutti quelli possibili poiché il primo scopo del testo è quello di abilitare allievi completamente digiuni nel giro di 60 o, al massimo, 90 giorni, come detta il mercato. Abbiamo perciò privilegiato arbitrariamente un gruppo di argomenti che ci sembra sufficiente a inventare e a rendere autonomo un programmatore sbarbato.

Il testo è concepito per l'uso contemporaneo di un AS/400 sul quale esercitarsi, ma può essere utile anche senza *corpore vili*.

Il corso porterà all'autonomia sufficiente a capire l'analista per cui si lavorerà e alla capacità di consultare e capire i manuali DDS, RPG e CL sul web, quando necessario.

Dal sito www.neroni.it è sempre scaricabile gratuitamente l'ultima edizione del corrente manuale e sarà bene accetto un commento alla mail claudio@neroni.it per segnalarci errori, argomenti secondo voi necessari ma non sviluppati o, infine, apprezzamenti e giudizi.

Claudio Neroni e Riccardo Battistutta

2003

Mi son trovato, nel rivedere l'opera per una migliore edizione dopo il primo utilizzo, a scavare una discreta quantità di idee tanto metabolizzate nella pratica dell'insegnamento da considerarle in qualche modo ovvie e sottintese. La durata breve del corso ha reso indispensabile l'evidenziazione scritta di certi atteggiamenti logici che il tempo breve non permette di comunicare per semplice osmosi agli allievi. È nata così una particolare attenzione alla descrizione del modo in cui nasce la programmazione, sperando che serva a far scattare la scintilla della comprensione e dell'autonomia negli allievi.

Claudio Neroni

23-09-2008

Fino ad oggi il pregio principale dell'opera, alla faccia delle pretese culturali degli autori, sembrano essere la gratuità e l'italianità. Penserò per ripicca se trasformarla in comense ☺.

Lavorando per la gloria, devo dire che il centinaio di richieste di ultima edizione mi hanno effettivamente indotto a riprendere l'impresa. Ho riletto il manuale con scrupolo e credo di aver ritrovato l'intenzione di finirlo.

Sperèm!

Claudio Neroni

10-10-2008

Sto concludendo questo modo vecchio stile di usare l' RPG Ile che comunque è indispensabile alla formazione di un manutentore. Poi vedrò se merita allevare sviluppatori alla moda. Non vorrei convertire gente che pensa moderno il PC in ambito gestionale.

Claudio Neroni

Note sulla terminologia

Nel testo si menzionano frequentemente Rpg400 ed RpgIle. Il termine RpgIV è da considerare sinonimo di RpgIle.

Per Rpg400 si intende la versione più vecchia del linguaggio, ancora molto presente nei pacchetti praticamente in uso, ma ferma per sempre al rilascio 3.2.0.

Per RpgIle si intende l'ultima versione del linguaggio, nata nel rilascio 3.1.0 e in forte evoluzione fino ad oggi, rilascio 5.2.0.

Per RPG si intende di solito sia Rpg400 che RpgIle. Nella maggior parte dei casi, l'uso del termine RPG copre entrambi i linguaggi, salva diversa indicazione del contesto.

Il testo insegna un RpgIle moderno solo dove necessario, cavalcando una via di mezzo tra vecchio e nuovo per creare familiarità con entrambi.

L'IBM ha anche tentato di soppiantare il termine AS/400 con il termine iSeries. L'operazione è goffamente commerciale e affligge forse l'hardware. Un cambio di nome poteva essere giustificato nel passaggio di tecnologia del processore AS/400 da CISC a RISC, ma non fu attuato.

Le parti in rosso sono da completare, rivedere o verificare.

SOMMARIO

Ritentare.
Il sommario deve essere ottenuto automaticamente da Word ma cambiare la tipologia dei paragrafi da indicizzare non è così indenne da inconvenienti come vorrei per farlo subito.

1 Uso dell'AS/400 da parte del programmatore. (01C)

Per accedere al sistema AS/400 è necessario possedere alcune nozioni di base.

In primis, come un guidatore non deve essere anche meccanico per guidare, così un programmatore applicativo non deve necessariamente avere nozioni oltre a quelle tipiche richieste per usare l'interfaccia di programmazione verso il sistema. Può averne, ma solo per cultura e non per necessità. Il programmatore deve invece possedere tutte le nozioni per costruire e variare l'interfaccia che egli medesimo, programmando, mette a disposizione dell'utente finale.

1.1 Comandi.

I comandi del sistema AS/400 hanno nomi fino a 10 caratteri, composti riunendo abbreviazioni di parole o brevi frasi standard precostituite. Ad esempio.

CRT	=CReaTe
DLT	=DeLeTe
DSP	=DiSPlay
LIB	=LIBrary
F	=File
OBJ	=OBJect
MOV	=MOVe
PDM	=Program Developement Manager
PF	=Physical File
SPL	=SPooL
TAP	=TAPe
WRK	=WoRK

Sono in maiuscolo le lettere del testo che generano l'abbreviazione.

Tali parole, riunite opportunamente, formano i nomi dei comandi. Ad esempio.

CRTPF	=Create Physical File
DLTF	=Delete File
DSPLIB	=Display Library
DSPTAP	=Display Tape
MOVOBJ	=Move Object
WRKOBJPDM	=Work Object PDM
WRKSPLF	=Work Spool File

Naturalmente, non basta che l'acronimo sia significativo, ma occorre che il programmatore del sistema abbia veramente concepito la funzione e le abbia dato proprio quel nome.

1.2 Parametri dei comandi.

Ogni comando è poi dotato di parole chiave (keyword) e, dentro le parole chiave, di parametri, secondo la forma seguente.

```
CRTPF        FILE(library/file)  RCDLEN(100)  TEXT(*SRCMBRTXT)
```

I nomi delle keyword non rispettano regole altrettanto precise che i nomi dei comandi, e meno ancora hanno regole i parametri, anche per quei valori precostituiti che, di solito, cominciano con un asterisco.

1.3 Immissione comandi.

Il video fondamentale per interagire con il sistema è quello fornito dal programma **immissione Comandi** (CALL QCMD) che permette la digitazione dei comandi, vera e completa interfaccia di chiamata dei programmi di sistema.

Battuto il solo nome di un comando o il comando e qualche parametro, è sempre possibile, tramite F4, accedere ad una forma particolarmente estesa ed assistita del comando stesso, il

prompter. Invio, sia sul video comandi che sul prompter, provoca l'esecuzione del comando stesso.

I comandi digitati vengono accodati al programma esecutore (QCMD) che li smaltisce immediatamente e li fotografa in una coda messaggi del programma, costantemente visibile tramite il video comandi stesso. La coda contiene anche i messaggi di risposta del sistema e dei programmi eseguiti. Tramite i tasti di scorrimento, detta coda è rivedibile per il ripescaggio di comandi già eseguiti ma che possono servire da scheletro ad una nuova richiesta. Se il cursore è posizionato su di un comando già eseguito, F9 lo copia nell'area di digitazione, F4 lo copia e lo prompterizza. Se il cursore non è posizionato, F9 copia l'ultimo o, di seguito, il precedente all'ultimo copiato. F10 commuta una visualizzazione più estesa o meno estesa della coda messaggi.

1.4 Riga comandi.

Su molti video di sistema è presente la **riga comandi**, che visualizza poche informazioni circa i comandi già eseguiti e si presta solo con difficoltà alla digitazione di comandi più lunghi della riga medesima. Da questa si accede comunque al prompter. F9 copia l'ultimo comando eseguito o, di seguito, il precedente all'ultimo copiato, attingendo comunque alla stessa coda detta per l'**immissione comandi**.

1.5 Prompter.

Il **prompter** (suggeritore) è un'interfaccia che permette la digitazione assistita dei comandi, sia quelli di sistema che, eventualmente, quelli predisposti dal programmatore. Esso fornisce un breve testo per ogni parametro e permette l'accesso ai video di aiuto che spiegano e commentano con dettaglio i particolari dei campi del comando e dei valori che vi si possono inserire. L'intervento, a richiesta, del prompter acconcia i dati digitati nella forma più estesa con keyword e parametri. Molti comandi sopportano, peraltro, una digitazione semplificata fino al terzo o quarto parametro senza keyword. Oltre tale limite, le restanti keyword vanno comunque digitate per esteso.

1.6 Menù.

Mentre l'accesso dal video comandi richiede conoscenza preventiva dei comandi da usare, i **menù**, permettono di navigare a vista alla ricerca del comando adatto. Ai menù si accede tramite il comando GO.

GO MAIN	Accede al menù principale, dal quale sono raggiungibili tutti gli altri.
GO PROGRAM	Presenta il menù più adatto al programmatore, ma è comunque raggiungibile dal MAIN.
GO CMDxxx	Dove xxx è una parola cercata (ad es: CRT). Presenta l'elenco di tutti i comandi che contengono nel loro nome la parola cercata (ad es: CRTLIB, CRTPF, …).

Il menù MAIN può essere chiamato anche da F16 dell'immissione comandi.

Se il profilo lo prevede, finita l'esecuzione del programma iniziale, il collegamento si ferma sul **menù iniziale**. Il menù iniziale di un profilo si cambia dinamicamente battendo F23 sul menù desiderato. Alla chiusura dei menù, il collegamento termina.

1.7 Profilo utente del programmatore e sue opzioni.

Per accedere al sistema AS/400 come programmatore è necessario usare un profilo utente adatto già definito sull'elaboratore da un responsabile della sicurezza.

Nome e parola d'ordine del profilo utente, digitate sul video di collegamento, permettono di ottenere un video di immissione comandi o un video menù dai quali richiamare, con l'opportuna autorità, le funzioni di programmazione.

Al programmatore si consiglia l'uso abituale non dei menù ma dell'immissione comandi, che fornisce una visione più immediata delle azioni precedentemente svolte. I menù possono essere chiamati per il tempo necessario a trovare i comandi non ben conosciuti ma è buona cosa memorizzarne il nome per successive chiamate e per ottenere così un'esperienza immediatamente travasabile nella scrittura di programmi in control language.

Il profilo da usare deve avere una serie di caratteristiche, alcune necessarie, altre opportune.

Tali caratteristiche possono essere assegnate tramite il comando di creazione CRTUSRPRF (Create User Profile) o quello di modifica CHGUSRPRF (Change User Profile), comunque esercitati dal responsabile della sicurezza. Alcune caratteristiche possono essere gestite anche dal profilo medesimo tramite il comando CHGPRF (Change Profile).

```
CRTUSRPRF                          Create User Profile
      USRPRF(CORSORPG)                   User Profile
      USRCLS(*PGMR)                      User Class
      INLPGM(CORSORPGIP)                 Initial Program
      INLMNU(*SIGNOFF)                   Initial Menu
      TEXT('Corso RPG.')                 Text
      PTYLMT(0)                          Priority Limit
      JOBD(CORSORPGJD)                   Job Description
      GRPPRF(QPGMR)                      Group Profile
      OWNER(*GRPPRF)                     Owner
      ATNPGM(QCMD)                       Attention Program
      USROPT(*CLKWD)                     User Option
```

Segue una breve illustrazione delle singole parole chiave usate nella creazione o nella modifica del profilo utente.

User Profile — Il nome del profilo utente (Mario Rossi) può essere il cognome del programmatore (ROSSI) o il cognome seguito dall'iniziale del nome (ROSSIM). Essendo utile spesso per capire chi ha fatto che cosa, sono sconsigliabili:

- i nomi di fantasia (PIPPO, VENDETTA2);
- i nomi di funzione (CORSO, PGMR);
- il nome di una persona che svolgeva la funzione prima di quella attuale (GATES).

User Class — La classe del programmatore (*PGMR) autorizza a funzioni necessarie per programmare; la classe dell'utente (*USER) è troppo scarsa; la classe del responsabile della sicurezza (*SECOFR) è eccessiva per un programmatore junior.

Initial Program — Il programma iniziale serve a preparare l'ambiente in cui agire per quanto attiene di solito alla lista librerie utente e di sistema, alla modalità di presentazione dei messaggi, al primo comando da eseguire, alla possibilità di disconnettere il job interattivo quando cade (se diversa dal valore di sistema), al trasferimento all'immissione comandi, etc. Come ultima istruzione del programma iniziale, si consiglia di eseguire il TFRCTL (Transfer Control) al QCMD per poi lavorare interattivamente con l'Immissione comandi. Si noti che in tal caso il programma iniziale non sarà un CLLE ma un CLP, il solo a permettere il transfer control.

Initial Menu — Se si è scelto di lavorare sull'immissione comandi, si consiglia di utilizzare l'opzione *SIGNOFF che, terminato il programma iniziale, evita di permanere sui menù. Altrimenti si indicherà il nome di un menù (MAIN, PROGRAM, …) che comunque potrà essere reimpostato a piacere digitando F23 sul menù desiderato come menù iniziale.

Text	Preferibilmente funzione e nome e cognome ("Programmatore Mario Rossi").
Priority Limit	Se nulla osta da parte del responsabile del sistema, si assuma la migliore: "0" (zero).
Job Description	Non abbia mai il medesimo nome del profilo utente: tale scelta è stata già compiuta dal modulo base IBM (la cosiddetta *Architettura*) nella libreria ACGGAA. L'eventuale omonimia renderebbe ineseguibili i lavori batch lanciati tramite il modulo base da parte dell'utente corrente. Comunque, tramite la descrizione lavoro si possono scegliere lista librerie iniziale, coda di emissione, etc.
Group Profile	Capo gruppo del profilo, dal quale ricavare le autorità per agire sugli oggetti. Sia normalmente QPGMR, profilo generico fornito dal sistema come prototipo di programmatore e universalmente usato come proprietario d'ogni cosa.
Owner	Sia *GRPPRF per attribuire al Capo gruppo la proprietà degli oggetti creati dal programmatore corrente.
Attention Program	E' il programma chiamato quando, con immissione non impedita, si preme il tasto Attn (Attention, sul PC Esc = Escape). La scelta più semplice è QCMD (Immissione comandi) che permette di intervenire facilmente a valle di un programma già in esecuzione ma fermo su video.
User Option	Sono consigliabili i parametri seguenti.

*CLKWD	Mostra le parole chiave (invece delle scelte possibili) durante il prompt.
*HELPFULL	Mostra l'aiuto in pagina piena invece che in finestra. Le finestre di testo in ambiente carattere sono miserelle in quanto a quantità di dati mostrati e costringono ad una manipolazione frenetica dei roll. Meglio quindi una buona videata piena da vedere senza fatica aggiuntiva.
*PRTMSG	Avvisa dell'avvenuta stampa fisica di uno spool file del profilo, ovunque avvenga. Serve ad allertarsi per il ritiro della stampa e, più spesso, visto che di solito il programmatore stampa poco, ad essere avvisati che si sta sprecando carta per sbaglio.
*STSMSG	Evidenzia i messaggi di stato che il sistema emette quando in interattivo sta eseguendo qualcosa di lungo (ad es: copia di file, attese) con l'immissione impedita.

1.8 I lavori del programmatore, interattivi e batch.

Il programmatore esegue di solito due tipi di lavori.

- L'interattivo, con il quale modifica i sorgenti e manipola gli oggetti.
- I batch, con i quali i compilatori trasformano i sorgenti in oggetti.

Atteggiandosi per prova come utente, eseguirà anche interattivi e batch di tipo applicativo da lui stesso concepiti. Trascuriamo per ora questi, anche se costituiscono l'oggetto ultimo del presente corso, e occupiamoci dei due tipi detti.

Solitamente, l'uso dell'utility PDM (Program Developement Manager) incanala e risolve in maniera soddisfacente e semplice il problema dell'attività di programmazione, senza precludere strade più raffinate, interne ed esterne al PDM.

Un lavoro interattivo nasce dal collegamento al sistema tramite un video terminale o tramite la sua emulazione su un personal. Al termine delle operazioni di collegamento, compare un video di immissione comandi oppure un video di menù dotato di riga comandi. Su tali video, digitando comandi od operando scelte successive, il programmatore chiama in esecuzione i programmi del PDM per manipolare i sorgenti o gli oggetti. Il PDM visualizza elenchi di sorgenti sui quali si interviene mediante opzioni interattive di creazione, modifica e copia ed opzioni batch che sottomettono in esecuzione la compilazione dei sorgenti per la loro trasformazione in programmi oggetto eseguibili.

Le compilazioni possono anche avvenire anche in interattivo (regolate le opzioni PDM), ma tale uso assassina la prestazioni interattive dei quattrocentini, con i quali a volte si deve combattere in competizione con un gruppo di utenti esasperati e feroci.

1.9 Gestione dei lavori.

Una volta sottomessi, i lavori batch vivono di vita propria.

Il comando di Sottomissione Lavoro (SBMJOB = Submit Job) porta nella keyword CMD il comando da eseguire nel lavoro batch ed inserisce il nuovo lavoro in una Coda Lavori (JOBQ = Job Queue) facendo uso delle caratteristiche di una Descrizione Lavori (JOBD = Job Description).

```
SBMJOB   CMD(CRTPF FILE(library/file) RCDLEN(100) TEXT(*SRCMBRTXT)
         JOBD(jobdescription) JOBQ(jobqueue)
```

Il lavoro viene allora automaticamente preso in carico dal sottosistema che si alimenta dei lavori giacenti sulla coda. Non appena gli è possibile, il sottosistema esegue il lavoro pendente fino al suo esito finale; allora il lavoro muore e di lui restano solo un messaggio conclusivo e le stampe generate. Tipicamente il messaggio finale indirizzato al video lanciatore indicherà **Fine normale** o **Fine anomala**. In quest'ultimo caso, specialmente se il lavoro è una compilazione, sarà necessario esaminare le stampe, correggere il problema e rieseguire.

Se in coda giacciono molti lavori, può succedere che uno specifico lavoro tardi ad avviarsi e, se il sistema è molto carico, può anche avvenire che il lavoro duri un tempo eccessivo. In questi casi, fatte le dovute considerazioni sul carico della macchina e sull'opportunità di non superare certe soglie, può essere giudicato necessario intervenire sull'ordine dei lavori in coda o sulla loro priorità in esecuzione.

Si usa allora il comando WRKSBMJOB (Work Submitted Job) che elenca i lavori sottomessi dall'utente, anche se svolti con un profilo diverso, e permette di modificare i valori richiesti. In altri casi può essere utile il comando WRKUSRJOB (Work User Job) che elenca tutti i lavori dell'utente. In altri WRKJOBQ (Work Job Queue) per elencare i lavori in una coda. In altri ancora WRKACTJOB (Work Active Job) per elencare tutti i lavori attivi sul sistema. Tutti i comandi elencati permettono di scegliere un singolo lavoro e di modificargli le caratteristiche elencate nel CHGJOB (Change Job).

1.10 Comandi abbreviati.

Se, come si consiglia, l'accesso alla macchina avviene tramite l'immissione comandi, è utile costruirsi una batteria di comandi abbreviati duplicando gli originali di sistema con nomi più corti. Il comando da eseguire, con l'autorità necessaria, è il seguente.

```
CRTDUPOBJ  FROMOBJ(comando) FROMLIB(QSYS) OBJTYPE(*CMD) TOOBJ(breve)
           TOLIB(libreria)
```

Ad esempio, si possono duplicare i seguenti comandi con i corrispondenti nomi abbreviati.

WRKACTJOB	=A	WRKSBMJOB	=M
WRKSPLF	=WS	DSPOBJD	=J
WRKOBJ	=JJ	WRKOBJPDM	=JJJ
WRKLIBPDM	=JJJJ	WRKSYSSTS	=S

1.11 Lista librerie.

Al momento di eseguire un programma, i file sui quali operare ed i programmi chiamati successivamente, se non già indirizzati ad una specifica libreria, vengono cercati lungo la **lista librerie**.

La lista è un elenco di librerie che vengono ordinatamente esplorate una dopo l'altra alla ricerca degli oggetti da usare. Tra gli oggetti di egual tipo e nome presenti in lista, viene usato il primo trovato e trascurati tutti quelli presenti nelle librerie successive.

Un **sistema informativo**, ad esempio, è definito da una lista librerie nella quale si trovano solitamente le seguenti tipologie.

- Libreria menù.
- Libreria dati aziendali.
- Libreria programmi di pacchetto.

Un sistema informativo più complesso può contenere più tipologie di quelle elencate.

- Libreria menù.
- Libreria dati comuni a più sistemi informativi simili.
- Libreria dati aziendali.
- Libreria programmi personalizzati.
- Libreria programmi di pacchetto.

Normalmente i nomi di librerie non vanno inclusi nei programmi per favorire la possibilità di usare le liste. Un programma contenente l'indirizzamento ad una specifica libreria non può, infatti, essere usato sic et simpliciter in un altro sistema informativo ma ne andrà generata una versione particolare. Si afferra la difficoltà di gestire con questo criterio sulla stessa macchina sia un sistema informativo duplicato per prove, sia un sistema informativo in cui solo l'azienda ed i dati aziendali cambiano. E' infatti frequente che un unico ente gestisca più aziende della medesima tipologia con i medesimi programmi sulla stessa macchina. Difficoltà simili si trovano anche nell'installare un pacchetto programmi su macchine diverse quando, come di solito avviene, la libreria dati cambia nome per adeguarsi all'azienda.

La prima impostazione della lista librerie in un lavoro interattivo avviene per diversi canali.

- Dal profilo utente al lavoro.
- Dalla descrizione lavori propria del profilo.
- Da un'esplicita modifica compiuta nel programma iniziale del profilo.

La prima impostazione della lista librerie in un lavoro batch avviene invece nei seguenti modi.

- Dal comando di sottomissione del lavoro.
- Dalla descrizione lavori usata nella sottomissione.
- Dalla lista librerie in vigore nell'interattivo lanciatore al momento del lancio.

Nel corso del lavoro, si può sempre modificare la lista librerie mediante l'uso del comando CHGLIBL (Change Library List) nel quale si scrive l'elenco delle librerie desiderato.

La modifica ha effetto solo sulle aperture di file e programmi successivi e non coinvolge file e programmi già aperti.

La lista librerie è composta da quattro sezioni relativamente indipendenti.

- La lista librerie di sistema.
- La libreria corrente.

- La libreria di prodotto.
- La lista librerie utente.

La lista librerie di sistema è la sezione comune a tutti i lavori che nascono sul sistema. Il suo valore iniziale è contenuto nel valore di sistema QSYSLIBL la cui impostazione avviene una tantum in fase di installazione del sistema o di un nuovo rilascio del sistema operativo. Contiene solitamente solo librerie del software di base come QSYS. La modifica estemporanea della lista librerie di sistema avviene con un comando apposito CHGSYSLIBL (Change System Library List) rimuovendo od aggiungendo una sola libreria alla volta. L'aggiunta inoltre avviene solo in testa alla lista.

La libreria corrente viene maneggiata insieme alla lista librerie utente tramite il comando CHGLIBL (Change Library List), che oltre a manipolare la sezione utente della lista, può sostituire la libreria corrente. La libreria corrente viene usata come default in molti comandi di creazione. Anche l'esecuzione dei comandi può manipolare la libreria corrente se nella definizione del comando è personalizzata la libreria corrente.

La libreria di prodotto viene inserita estemporaneamente durante l'esecuzione di un comando, ma viene tolta quando il comando termina. Viene inoltre sostituita alla chiamata di un comando, anche se il secondo comando è chiamato dal primo.

La lista librerie utente è la sezione specifica di ogni lavoro. Il suo valore iniziale è contenuto nel valore di sistema QUSRLIBL la cui impostazione avviene una tantum in fase di installazione del sistema o di un nuovo rilascio del sistema operativo. Contiene solitamente librerie del software applicativo e librerie di dati. La modifica estemporanea della lista librerie utente avviene con un comando apposito CHGLIBL (Change Library List), che sostituisce la lista intera, o con alcuni comandi correttivi della lista già in essere come ADDLIBLE (Add Library List Entry) o come RMVLIBLE (Remove Library List Entry), che aggiungono o rimuovono una sola voce lungo la lista.

1.12 Descrizione lavoro.

Molte delle caratteristiche che si conviene di assegnare ad un lavoro per la sua migliore esecuzione, si annotano in un oggetto particolare, la descrizione lavoro, creabile tramite il comando CRTJOBD (Create Job Description) e modificabile tramite il comando CHGJOBD (Change Job Description) per i quali si elencano alcune parole chiavi significative.

INLLIBL	Lista iniziale delle librerie dell'utente.
JOBQ	Coda di immissione del lavoro.
OUTQ	Coda di emissione per le stampe.
LOG	Dettaglio della joblog (insieme dei messaggi relativi al lavoro).
ENDSEV	Gravità di cancellazione del lavoro.

La descrizione lavoro viene usata in più punti già menzionati ai quali si rimanda.

2 Organizzazione dei dati. (O2C)

L'AS/400 è essenzialmente una macchina da database, ampiamente attrezzata per gestire dati attraverso un database nativo definito tramite linguaggi specifici ed univoci e dotata di linguaggi applicativi fortemente orientati alle operazioni su database.

Il concetto di dato va quindi sviluppato in primis et ante omnia.

2.1 Definizione dei dati.

Per descrivere un gruppo di soggetti omogenei, si definisce un insieme di tipi di informazione da raccogliere su tutti i soggetti, assegnando ad ogni tipo un nome (**campo**), come pure all'insieme (**tracciato** o **formato**). Ogni soggetto dispone di uno spazio proprio in cui disporre ordinatamente le informazioni raccolte (**record**). Il gruppo di informazioni su più soggetti risiede a sua volta in una spazio contiguo (**file**).

All'interno della struttura descritta si dispone ogni singola informazione. Ogni **dato** giace quindi in un **campo**, che fa parte di un **record**, che fa parte di un **file**.

Preme sottolineare che:

- Ogni record contiene gli stessi campi con dati però diversi.
- Ciascun campo figura in tutti i record di un file, anche quando è vuoto.
- Il campo e il tracciato sono definizioni qualitative.
- Il record ed il file sono contenitori di dati specifici.

In altre parole, il termine **dato** identifica il contenuto di una singola informazione (**campo**) su di un soggetto. Tutte le informazioni su di un soggetto sono raccolte in una descrizione (**record**) rispettando uno schema fisso (**tracciato** o **formato**). Le descrizioni di più soggetti radunate insieme formano a loro volta un archivio (**file**).

L'esempio nativo della meccanografia riporta il caso di un'azienda che per conoscere i propri clienti tiene un gruppo (**file**) di schede descrittive (**record**) con una griglia (**tracciato** o **formato**) di spazi predefiniti (**campi**) per riportare informazioni standardizzate (**dati nei campi**) come la ragione sociale, l'indirizzo, il codice.

Nella pratica, i termini introdotti sono spesso usati in maniera ambivalente, per definire sia enti astratti che enti concreti. Ad esempio il termine *campo XYZ*, che è la definizione qualitativa di un tipo di informazione, viene contestualmente spesso usato per significare il campo XYZ di quello specifico record. Più raramente si confonde addirittura il campo col suo valore, il dato.

Le confusioni non dipendono tanto da un cattivo uso personale ma, piuttosto, dalla lacunosità ed ambiguità dell'uso comune. L'elaborazione dati che si pratica nelle aziende è nata lì e stenta a risentire delle teorizzazioni esterne. Con tali imprecisioni occorre convivere.

2.2 Dati sull'AS/400.

Quanto detto nel paragrafo precedente è valido su qualunque **database**. Di qui in avanti si parlerà invece più specificamente dell'**AS/400**.

Il file, come sopra definito, si chiama **membro** ed il nome di "**file**" viene assegnato all'insieme dei membri. Il file risulta cioè composto da più tronconi che contengono gruppi di record diversi. Nella gestione normale, comunque, prevalgono i file dati monomembro.

I file contengono anche la definizione del loro tracciato e, quando vengono menzionati in un programma, il tracciato viene automaticamente conosciuto anche dal programma, che così può accedere al file rispettandone il tracciato nativo.

I file risiedono all'interno di contenitori chiamati **librerie**, simili agli indirizzari di un pc ma che, a differenza di quanto succede a questi ultimi, non possono contenere altre librerie.

I file possono essere di **tipo dati** o di **tipo sorgente**. I file dati hanno un tracciato completamente arbitrario deciso in un apposito sorgente di creazione. I file di tipo sorgente

sono invece sempre dotati di un tracciato di tre campi: campo data, campo sequenza e campo dati.

2.3 File dati.

Per i file contenenti dati è sufficiente quanto detto più sopra.

Si aggiunge soltanto che i file dati relativi ad una singola azienda sono solitamente raggruppati in un unico contenitore di file sincronizzati tra loro, detto libreria dati.

Il file dati è solitamente monomembro. Il multimembro è usato in certe organizzazioni come area di transito dei dati estratti, ordinati e stampati. Sono invece rarissimi i multimembro contenenti dati stabili.

2.4 Accesso ai file dati.

L'accesso ai dati contenuti nei file dati avviene soprattutto attraverso i programmi applicativi scritti in RPG e negli altri linguaggi (Cobol,...) ma anche attraverso programmi di utilità come Dfu, Query e Twf o Wrkdbf.

La costruzione e l'uso di database di file dati è lo scopo del presente corso.

2.5 File sorgenti.

Un testo sorgente è un testo scritto secondo le regole di un certo linguaggio, che, esaminato dal corrispondente compilatore, provoca la generazione di un programma eseguibile sul medesimo elaboratore. Il contenitore di un testo sorgente è il membro di un file di tipo sorgente. Ogni record, nel campo dati, recepisce una riga del testo che costituisce il sorgente.

Il file sorgente è sempre multimembro, non essendoci alcuna necessità di differenziare il tracciato dei vari membri. La distinzione dei vari tracciati necessari nei vari tipi di linguaggio è operata comunque dal SEU (programma di utilità per la gestione dei sorgenti), interpretando il campo dati di ogni record in base al contenuto del campo stesso.

Per libreria sorgente si intende il contenitore che contiene in prevalenza o, meglio, esclusivamente file sorgenti, spesso specializzati per tipo di linguaggio. Nei file sorgenti risiedono i membri sorgenti, ciascuno dei quali contiene un solo testo sorgente di un solo tipo. Nel medesimo file possono coesistere membri con tipi di sorgente diversi.

Nel seguito, per sorgente si intenderà tanto il testo sorgente contenuto quanto il membro sorgente contenente.

Quando si accede all'area dati AS/400 come se fosse un disco di personal computer, libreria e file sono identificabili come indirizzari (il secondo dentro il primo) mentre il membro corrisponde al file come inteso sul personal computer.

2.6 Accesso ai file sorgenti.

Dell'accesso ai sorgenti e delle opzioni da praticare su di essi si parlerà diffusamente nel capitolo del PDM (Program Developement Manager).

3 Source Entry Utility (SEU). (03R)

E' lo strumento mediante il quale si effettua la codifica di un qualunque tipo di sorgente. Viene messo a disposizione dal sistema per il programmatore. A seconda del tipo di sorgente che si sta editando lo strumento stesso effettua il controllo che il tipo di specifica che si desidera codificare sia compatibile con il tipo di sorgente in utilizzo. Attenzione, poiché lo strumento abilita o no la codifica delle specifiche ma non effettua nessun altro tipo di controllo di validità su ciò che si sta editando; solo il comando di creazione dell'oggetto specifico (compilazione) effettua tutti i controlli di validità producendo la lista di compilazione.

3.1 Comandi di riga (inserimento, copia, cancellazione, spostamento, scheletro,…)

I comandi di riga/specifica sono piuttosto semplici ed intuitivi.

Inserimento di una riga.

Il comando per eseguire l'inserimento di una nuova riga è I = Insert. Questo comando si limita ad inserire una nuova riga generica a tracciato libero; è a carico del programmatore l'attribuzione del tipo di specifica e degli opportuni codici operativi. Per agevolare il compito vi è la possibilità di richiedere l'inserimento di una nuova riga ma che presenti il prompt di un tipo ben definito di specifica. Il comando è IP = Insert Prompt. Oltre alle due sillabe si deve specificare il tipo di specifica per la quale si sta richiedendo il prompt. Se si deve inserire una nuova riga C(calcolo) il comando è IPC, se invece la nuova riga è di tipo FX(definizione file esterni) il comando è IPFX e così per tutti i tipi di specifica. Qualunque tipo di specifica, anche per i file fisici e logici, video stampe etc.

Copia di una o più specifiche.

Il comando per eseguire la copia di una specifica esistente è C = Copy. Semplicemente specificando C sulla specifica che si intende copiare e dando la destinazione di dove collocare la specifica copiata generandone una nuova. La destinazione finale deve essere specificata o prima o dopo un'altra specifica esistente. Se si desidera copiare una specifica prima di un'altra specifica si deve specificare B = Before la specifica di destinazione finale. S Se si desidera copiare una specifica dopo di un'altra specifica si deve specificare A = After la specifica di destinazione finale.

Il comando per eseguire la copia di più specifiche esistenti e contigue è CC. Questo comando funziona in tutto e per tutto come il comando C = Copy per la singola riga. L'unica differenza è che si dovrà indicare qual è la prima e l'ultima specifica che si intende copiare. Quindi, per delimitare le specifiche da copiare si dovranno specificare due gruppi di istruzioni CC; CC (da questa specifica), CC (a quest'altra specifica). Tutte le specifiche comprese tra i due comandi CC/CC (incluse le specifiche contrassegnate con CC) saranno copiate ed inserite prima o dopo la specifica indicata come destinazione finale. Solo per il comando di copia è prevista la possibilità di richiedere la ripetibilità del comando. Questa funzione è richiedibile sia per il singolo comando che per quello multiplo. Per ottenere la ripetibilità della richiesta è sufficiente specificare R dopo il comando C o CC che divengono quindi CR e CCR/CCR.

Cancellazione di una o più specifiche.

Il comando per eseguire la cancellazione di una specifica esistente è D = Delete. Questo comando funziona in maniera simile al comando C = Copy, con la differenza che per effettuare la cancellazione di una singola riga si deve specificare il solo comando D = Delete. Per effettuare la cancellazione di più righe si devono specificare due gruppi di istruzioni DD; DD (da questa specifica), DD (a quest'altra specifica). Tutte le specifiche comprese tra i due comandi DD/DD (incluse le specifiche contrassegnate con DD) saranno cancellate. In questo caso ovviamente non si deve specificare alcuna destinazione finale.

Spostamento di una o più specifiche.

Il comando per eseguire lo spostamento di una specifica esistente è M = Move. Questo comando funziona in maniera simile al comando C = Copy, con la differenza che per effettuare lo spostamento di una singola riga si deve specificare il solo comando M = Move. Per effettuare lo spostamento di più righe si devono specificare due gruppi di istruzioni MM; MM (da questa specifica), MM (a quest'altra specifica). Tutte le specifiche comprese tra i due comandi MM/MM (incluse le specifiche contrassegnate con MM) saranno spostate ed inserite prima o dopo la specifica indicata come destinazione finale.

Definizione di una specifica scheletro e suo utilizzo.

Il SEU da la possibilità di costituirsi una specifica personalizzata nell'ambito di una sessione di editazione di un file sorgente. Se per esempio necessita avere a disposizione una riga di commento standard predefinita e facilmente utilizzabile senza doverla ogni volta riscrivere o copiare, si può efficacemente utilizzare il comando S = Skeleton. Innanzitutto si procede alla definizione della specifica che si intende eleggere come S = Skeleton tramite le comuni attività di codifica. Quindi per eleggere la specifica a S = Skeleton è sufficiente digitare S sul campo sequenza a sinistra della specifica stessa (numeratore progressivo specifiche), posizionare il cursore lungo la specifica e premere Enter. D'ora in avanti per ottenere l'inserimento automatico della specifica da noi designata come S = Skeleton con il cursore già posizionato è sufficiente richiedere l'inserimento di una nuova riga di tipo S, ovvero IPS.

3.2 Scansione e sostituzione.

Il SEU offre la possibilità di agevolare il lavoro del programmatore mettendo a sua disposizione una riga comandi specifica per la sessione di editazione. In questa riga comandi è possibile specificare diversi comandi; ad esempio se si deve ritornare alla prima specifica del sorgente in editazione sarà sufficiente specificare il comando T = Top e premere Enter. Se al contrario si deve andare all'ultima specifica sarà sufficiente specificare il comando B = Bottom e premere Enter.

Scansione e ricerca.

E' frequente la necessità di ricercare una istruzione, un nome di file od un indicatore in tutto il sorgente. Per evitare di dovere scorrere l'intero sorgente ed effettuare la ricerca a vista, si può scrivere la stringa da ricercare nel sorgente sulla riga comandi e premere il tasto di funzione F16. Il SEU eseguirà la scansione sull'intero file sorgente e si fermerà, visualizzando la specifica che contiene la stringa ricercata, tante volte quante sono le volte che la stringa ricercata è stata trovata. Se non diversamente specificato, la scansione inizia dalla specifica sulla quale si è posizionati fino all'ultima specifica del sorgente. Si può peraltro specificare, come parametro aggiuntivo alla stringa di ricerca, se la ricerca deve partire dalla prima specifica F = First Statement oppure dall'ultima specifica L = Last Statement.

Scansione e sostituzione.

Un'altra frequente necessità è quella di effettuare una scansione (ricerca) unitamente a quella di dovere sostituire ciò che si sta ricercando con un altro valore. Un esempio calzante potrebbe essere il caso in cui si è deciso di sostituire un campo in un file video con un altro di diverso significato ma con lo stesso formato. La modifica del sorgente del file video si effettuerà utilizzando l' SDA, mentre la modifica dell' RPG si farà utilizzando il SEU. All'interno del SEU, premere F14. Compare un video di richiesta parametri sul quale si deve specificare la stringa da ricercare nel sorgente, la stringa in sostituzione della precedente ed un altro parametro per indicare se le ricerche e sostituzioni debbono essere effettuate tutte insieme con una unica richiesta da parte del programmatore nel momento in cui viene premuto il tasto di funzione F17 oppure se le scansioni e sostituzioni debbono essere effettuate una alla volta. Tornando alla necessità iniziale di dovere sostituire il nome di un campo con un altro ovunque esso sia stato

menzionato all'interno del programma RPG, ecco che questa funzionalità offerta dal SEU assolve in modo efficace e sicuro tale necessità.

3.3 Esame lista di compilazione e contemporanea correzione errori.

Ogni volta che per un sorgente si lancia il comando di creazione dell'oggetto compilato, se non diversamente specificato, il sistema produce una lista di compilazione. Nella lista di compilazione si trovano tutte le informazioni relative a tutte le risorse esterne ed interne a cui si fa riferimento nel sorgente stesso; tutti i campi di tutti i file di database, video o stampa che si sono menzionati nelle specifiche di definizione dei file, dove sono stati definiti, utilizzati e modificati. Insomma la lista di compilazione è molto esauriente e ricca di informazioni. Ma ormai non si effettua più la stampa su carta della lista di compilazione per successiva consultazione e ricerca degli eventuali errori che il compilatore ha rilevato. Si noti che se il compilatore ha rilevato degli errori l'oggetto non è stato creato né tanto meno sostituita l'eventuale precedente edizione. Eseguire la ricerca degli errori ed apportare le necessarie correzioni al file sorgente lo si può effettuare direttamente da una sessione di editazione del sorgente utilizzando il SEU. Infatti entrando in editazione di un sorgente è possibile richiamare un video di richiesta parametri premendo il tasto funzionale F15. Specificando l'opzione 2 e premendo Enter, si accede alla visualizzazione della lista di compilazione per il sorgente corrente. Il video viene suddiviso automaticamente in due parti: nella metà superiore del video si continua ad essere in editazione del file sorgente, nella metà inferiore del video si è in visualizzazione della lista di compilazione. Come per tutti i file di spool, è possibile effettuare una ricerca di una stringa all'interno degli stessi. Nel caso specifico di esame della lista di compilazione dal SEU, la ricerca non solo è possibile, ma addirittura ne sono state aumentate le potenzialità. Infatti nel caso di ricerca degli errori rilevati dal compilatore in fase di creazione dell'oggetto, è sufficiente specificare come stringa di ricerca il valore speciale *ERR e premere F16 per avviare la ricerca. Al primo errore rilevato sulla lista di compilazione questa funzione si interromperà consentendo al programmatore di comprendere l'errore. Una volta compreso l'errore, semplicemente passando alla parte superiore del video ovvero all'editazione, è possibile ricercare il punto nel quale si devono effettuare le modifiche per la risoluzione degli errori. Si noti che se la ricerca degli errori con il valore speciale *ERR non si è mai fermata sulle specifiche di calcolo C, la prima istruzione sulla quale si fermerà è una istruzione priva di errori veri; da qui in avanti è quindi meglio sostituire il valore speciale di ricerca *ERR con la stringa "* 7030" che è il codice di errore generico che il compilatore RPG utilizza per segnalare gli errori di definizione campi all'interno del programma stesso.

4 Gestione e comparazione sorgenti. (04C)

4.1 Livelli di gestione di un pacchetto programmi.

Lo sviluppo di un pacchetto programmi, a prescindere dal tipo di distribuzione agli utenti finali, deve essere gestito a più livelli.

- Livello analisi.
  È costituito dalla descrizione di obiettivi, database e algoritmi compiuta dall'analista con l'aiuto del committente e dell'utente destinatario del pacchetto.
  Tipicamente, il livello è contenuto nel documento iniziale di analisi.
- Livello sorgente.
  È costituito dall'insieme delle istruzioni direttamente scritte dal programmatore.
  Tipicamente, il livello è contenuto nella libreria sorgente.
- Livello oggetto.
  È costituito dall'insieme dei file dati, dei programmi applicativi e di ogni altra tipologia di oggetti necessaria al funzionamento del pacchetto.
  Tipicamente, il livello è contenuto nella libreria dati e nella libreria programmi.
- Livello personalizzazione.
  È costituito da alcuni dei dati di personalizzazione del pacchetto che influiscono sul modo di funzionare dei programmi. Essi possono essere considerati di importanza pari ai sorgenti ma, diversamente dai sorgenti, non necessitano di compilazione e sono usati direttamente anche come oggetti.
  Tipicamente, il livello è contenuto nella libreria menù (ad esempio, di modulo base IBM o simile) da usare così com'è. Altre personalizzazioni possono essere presenti nella stessa libreria sorgente, sotto forma di file dati il cui contenuto verrà poi copiato nella libreria dati.

Il livello sorgente del pacchetto deve permettere sempre tramite le operazioni di compilazione (i vari comandi di creazione) la ricreazione integrale del livello oggetto, naturalmente fatta eccezione per i dati imputabili dall'utente finale.

In questa sede si esaminerà il livello sorgente, padre di ogni altro.

4.2 Livello sorgente.

Il problema fondamentale di uno sviluppo è la sua dipendenza dall'analisi.

Nel cosmo AS/400 capita piuttosto spesso che la figura dell'analista e quella del programmatore siano poco distinte, comportando questo vantaggi e svantaggi. Non esiste comunque il "minutatore" che, senza responsabilità, traduce in linguaggio di programmazione una analisi di dettaglio. Il programmatore è quindi sempre anche analista di dettaglio.

La più parte del travaglio creativo dell'analista dovrebbe esaurirsi prima dell'avvio della programmazione mediando le necessità dell'utente, le competenze applicative dell'analista e le capacità tecniche del programmatore.

Ma nessuna analisi è così precisa, autonoma e fortunata da bastare per tutto il corso di uno sviluppo. Essa va rivista e aggiustata in corso d'opera più volte per fare fronte alle ulteriori precisazioni dell'obiettivo da parte dell'utente e alle difficoltà tecniche nella realizzazione degli algoritmi previsti.

Esistono perciò sempre modifiche significative sia nell'analisi applicativa sia nell'analisi di dettaglio che intervengono a sorgenti già scritti. Ne segue che un sorgente già funzionante deve essere spesso modificato alcune volte prima di raggiungere l'edizione definitiva.

In questo caso capita abbastanza spesso di assumere soluzioni diverse dalla prima intrapresa. Nasce la necessità di salvaguardarsi dai cambi di parere che inducono a sconvolgere un sorgente già scritto per poi dolorosamente tornare sui propri passi ridigitando istruzioni perdute.

4.3 Annotazione delle modifiche nei sorgenti.

Se le modifiche da farsi in un sorgente sono di modesta entità ed il pacchetto non è in corso di primo sviluppo, è parere corrente che la seguente tecnica possa essere applicata con soddisfazione.

a. Col termine "asteriscare" si intende l'apposizione sulla specifica di un contrassegno (in RPG e sulle DDS è un asterisco alla posizione 7) che la trasforma in un commento.
b. Nessuna riga deve essere cancellata o modificata. È possibile soltanto aggiungere righe nuove o asteriscare righe preesistenti.
c. L'aggiunta di una specifica si risolve inserendo una riga nuova.
d. La cancellazione di una specifica si risolve asteriscando la riga vecchia.
e. La modifica di una specifica si risolve asteriscando la riga vecchia e aggiungendone una nuova.
f. Tutte le righe aggiunte o modificate vengono contrassegnate con un identificativo del tipo.

XXYYY

Dove: XX = Sigla personale del programmatore che esegue la modifica.
YYY = Progressivo della modifica sul pacchetto, magari anagraficizzata.

La tecnica illustrata è particolarmente utile quando il programmatore esegue modifiche ad un pacchetto consolidato di cui in seguito riceverà nuove versioni. In tal caso egli dovrà riportare modifiche indigene sulle nuove versioni periodicamente ricevute dal laboratorio di sviluppo della casa di software autrice del pacchetto.

Essa è altresì usata all'interno delle case di software per riconoscere le modifiche apportate nel tempo, da chi e per che ragione. Perde però di efficacia se sui nuovi rilasci non si provvede ad eliminare sistematicamente le righe asteriscate prima di metterli in circolazione.

4.4 Conservazione dei sorgenti prima della modifica.

Il software di base dell'AS/400 non è dotato di una gestione di annullamento e ripristino automatica. Si rimedia di solito copiando i sorgenti nei momenti cruciali in membri di servizio dello stesso file sorgente annotando nel testo esterno diciture del tipo "ante modifica tal dei tali".

È preferibile, almeno provvisoriamente, copiare il membro allo stato attuale con un nome suffissato, ad esempio, con "_1" oppure "_2" e così via. Conviene copiare anche i sorgenti coinvolti con il sorgente in questione (il display menzionato nell' RPG) usando addirittura lo stesso suffisso.

Altra soluzione: copiare il detto gruppo di membri sorgenti con lo stesso nome ma in file sorgenti estemporanei con un nome sistematico (SV001, SV002,…) e nella stessa libreria sorgente degli originali.

Si nota che la sistematicità del comportamento permette di ritrovare con facilità le copie fatte a cuor leggero ma necessarie e ricercate in momenti faticosi per ritrovare istruzioni ormai perdute nel sorgente corrente.

Anche prima di recuperare istruzioni, si consiglia la copiatura preliminare dello stato attuale dell'arte: gli andirivieni sono spesso ripetuti più volte.

4.5 Comparazioni di sorgenti

Per valutare quali modifiche siano state fatte tra due edizioni del medesimo sorgente, si può usare naturalmente il confronto a vista, commutando rapidamente sullo stesso video la visione dei due sorgenti. Il metodo era perfetto sui terminali ma è faticoso sui pc in emulazione di terminale.

Il sistema fornisce fortunatamente un comando specifico e abbastanza valido per tale scopo.

Si consiglia di caricare nelle proprie la seguente opzione Pdm, piuttosto che usare la precostituita 54.

CX = ?CMPPFM ?*NEWFILE(&L/&F) ?*NEWMBR(&N) OLDFILE(&L/&F) OLDMBR(&N)

Per l'uso occorre digitare l'opzione sulla riga di elenco membri del Pdm, premere invio e sul prompt digitare il nome del membro di confronto.

Si raccomanda attenzione per evitare di compiere la falsa comparazione di un sorgente con se stesso. Il comando di comparazione non se ne accorge e la trappola è grave.

4.6 Visualizzazione ed analisi dei risultati.

Siano dati due membri sorgenti, un originale ed un modificato e si confrontino con il comando CMPPFM (Compare Physical File Member).

Per semplicità ci si limita a una quindicina di righe con un solo caso per tipo di avvenimento.

- Riformattazione della riga allargando o stringendo zone in bianco.
- Eliminazione.
- Raddoppiamento.
- Sostituzione.
- Spostamento.
- Aggiunta.

Originale

```
Riga 01
Riga 02 riformattata
Riga 03
Riga 04
Riga 05 spostata
Riga 06 da eliminare
Riga 07
Riga 08 raddoppiata
Riga 09
Riga 10
Riga 11
Riga 12
Riga 13 da sostituire
Riga 14 da sostituire
Riga 15
```

Modificato

```
Riga 01
Riga 02     riformattata
Riga 03
Riga 04
Riga 07
Riga 08 raddoppiata
Riga 08 raddoppiata
Riga 09
Riga 10
Riga 11 modificata
Riga 12 cambiata
Riga 13 sostituita
Riga 14 sostituita
Riga a1 aggiunta
Riga a2 aggiunta
Riga 05 spostata
Riga 15
```

Per illustrare la stampa di confronto, ci si limita alla parte dati.

Si nota che il confronto risulterà più faticoso in presenza di istruzioni asteriscate come nel metodo indicato per la manutenzione di programmi altrui perché la visualizzazione della comparazione risulterà inquinata dalla presenza di righe superflue che provocano un intercalare anomalo di righe originali e di righe modificate.

Il confronto in verticale contrassegna le righe riformattate, inserite e cancellate all'interno di un unico flusso nel quale il comparatore intercala arbitrariamente le righe provenienti dai due sorgenti confrontati.

Confronto verticale

```
   ----+----1----+----2----+----3

RN-Riga 02     riformattata
RO-Riga 02 riformattata

D -Riga 05 spostata
D -Riga 06 da eliminare

I -Riga 08 raddoppiata

I -Riga 11 modificata
I -Riga 12 cambiata
I -Riga 13 sostituita
I -Riga 14 sostituita
I -Riga a1 aggiunta
I -Riga a2 aggiunta
I -Riga 05 spostata
D -Riga 11
D -Riga 12
D -Riga 13 da sostituire
D -Riga 14 da sostituire
```

Il confronto in orizzontale OPTION(*WIDE) contrassegna le righe riformattate, inserite e cancellate ma in due sezioni affiancate, diminuendo l'arbitrarietà dell'intercalazione.

Confronto orizzontale

```
   ----+----1----+----2----+----3----+--     ----+----1----+----2----+----3

RN-Riga 02     riformattata             ç RO-Riga 02 riformattata

                                        ç D -Riga 05 spostata
                                        ç D -Riga 06 da eliminare

I -Riga 08 raddoppiata                  ç

I -Riga 11 modificata                   ç D -Riga 11
I -Riga 12 cambiata                     ç D -Riga 12
I -Riga 13 sostituita                   ç D -Riga 13 da sostituire
I -Riga 14 sostituita                   ç D -Riga 14 da sostituire
I -Riga a1 aggiunta                     ç
I -Riga a2 aggiunta                     ç
I -Riga 05 spostata                     ç
```

5 Program Developement Manager (PDM). (05C)

Disponendo della sola interfaccia comandi e volendo eseguire un comando che coinvolge un oggetto, occorre.

- Cercare l'oggetto sul quale agire. Non sempre il suo nome è conosciuto esattamente.
- Conoscere ed indagare preliminarmente l'oggetto stesso e l'ambiente che lo contiene o lo conterrà.
- Eseguire il comando desiderato.
- Verificare l'esito del comando reindagando l'oggetto modificato.

Tutto questo comporta la digitazione ripetuta più e più volte del nome dell'oggetto e, contestualmente, dei comandi necessari.

Se poi la stessa attività va eseguita su molti oggetti simili, la ridondanza della digitazione raggiunge il parossismo.

Il Pdm è un programma di utilità che permette l'elencazione e la scelta di oggetti e quindi l'esecuzione di comandi precostituiti sugli oggetti scelti. Il comando da eseguire è associato all'opzione digitata nel campo scelta dell'elenco.

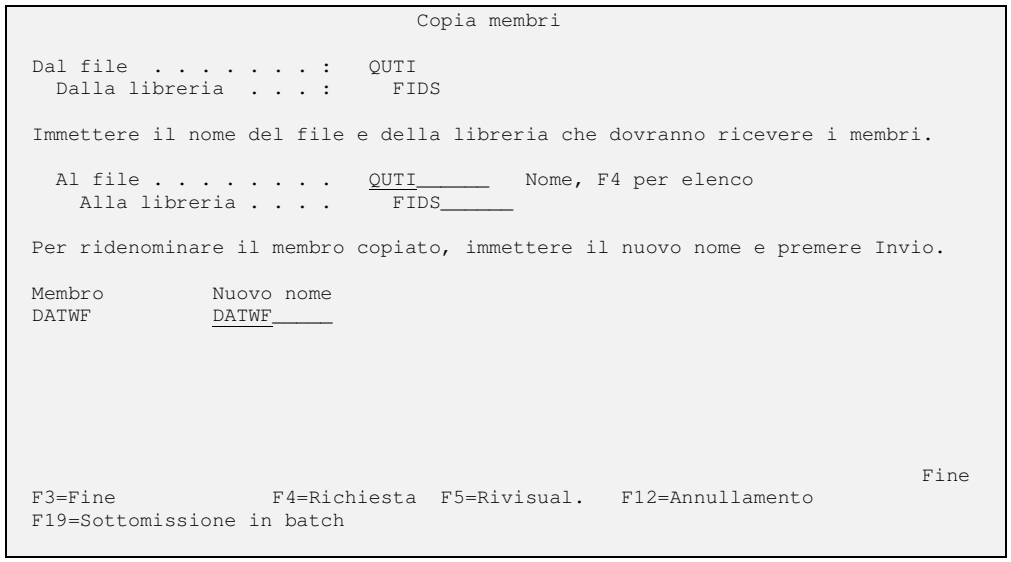
Gli scopi del programma sono molteplici e si intersecano.

- Gestire i sorgenti degli oggetti.
- Gestire la creazione degli oggetti a partire dai sorgenti, quando previsto.
- Gestire gli oggetti, creati o no attraverso il Pdm.

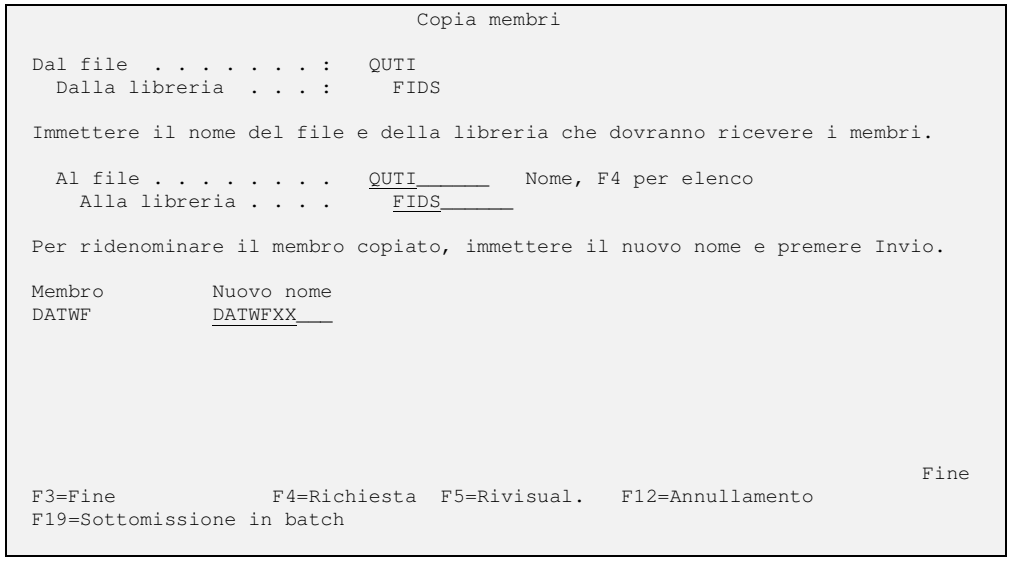
Si esemplifica un elenco di sorgenti e l'opzione 3 di copiatura su di un sorgente.

```
                          Gestione dei membri con il PDM              S44524BA

File . . . . . .    QUTI______
  Libreria. . . .     FID_______           Inizio elenco da . . .   __________

Immettere le opzioni e premere Invio.
 2=Editazione   3=Copia  4=Cancel.   5=Visualiz.   6=Stampa    7=Ridenom.
 8=Visual. descriz.  9=Salvat. 13=Modif. testo 14=Compilaz. 15=Creaz. modulo...

Opz  Membro      Tipo        Testo
__   CRSLDS      PF          Cursor location.
__   CRSLOC      RPGLE       Cursor location.
__   DATFW       RPGLE       Trasforma data da ccyymmdd a ddmmccyy e ddmmyy.
__   DATP        RPGLE       Prova Conversioni date.
__   DATPW       DSPF        Prova Conversioni date.
3_   DATWF       RPGLE       Trasforma data da ddmmccyy (o ddmmyy) a ccyymmdd.
__   FIDS        PF          File Information Data Structure.
__   HELPC       CLLE        Manage Help. Cpp.
                                                                      Segue...
Parametri o comando
===> ____________________________________________________________________________
F3=Fine          F4=Richiesta          F5=Rivisualizzaz.     F6=Creazione
F9=Duplicazione  F10=Immiss. comandi   F23=Altre opzioni     F24=Altri tasti
```

All'invio esce un video di richiesta di ulteriori informazioni.



Si completano le informazioni richieste e si preme invio.



Se la copiatura provoca ricopertura di un sorgente già esistente, esce un video di conferma.

```
                          Conferma copia membro

Per questa operazione di copia già esiste il seguente membro:

  Membro esistente . . . . . . . . . . :   DATWFXX
  File . . . . . . . . . . . . . . . . :   QUTI
    Libreria . . . . . . . . . . . . . :     NERONIFIDS

  Membro da copiare. . . . . . . . . . :   DATWF
  File . . . . . . . . . . . . . . . . :   QUTI
    Libreria . . . . . . . . . . . . . :     NERONIFIDS

Immettere la scelta e premere Invio.
Premere F12=Annull. per terminare e non eseguire l'operazione di compilazione.

  Cancellazione membro esistente . . . .   N   Y=Sì, N=No









F12=Annullamento
```

Si conferma digitando Y e premendo invio.

Viene evidenziato un messaggio di completamento al piede del video.

```
                          Gestione dei membri con il PDM             S44524BA

File . . . . . .    QUTI______
  Libreria. . . .     FID_______           Inizio elenco da . . .   __________

Immettere le opzioni e premere Invio.
 2=Editazione   3=Copia  4=Cancel.   5=Visualiz.   6=Stampa    7=Ridenom.
 8=Visual. descriz.  9=Salvat. 13=Modif. testo 14=Compilaz. 15=Creaz. modulo...

Opz  Membro      Tipo        Testo
__   CRSLDS      PF          Cursor location.
__   CRSLOC      RPGLE       Cursor location.
__   DATFW       RPGLE       Trasforma data da ccyymmdd a ddmmccyy e ddmmyy.
__   DATP        RPGLE       Prova Conversioni date.
__   DATPW       DSPF        Prova Conversioni date.
__   DATWF       RPGLE       Trasforma data da ddmmccyy (o ddmmyy) a ccyymmdd.
__   FIDS        PF          File Information Data Structure.
__   HELPC       CLLE        Manage Help. Cpp.
                                                                      Segue...
Parametri o comando
===> ____________________________________________________________________________
F3=Fine          F4=Richiesta          F5=Rivisualizzaz.     F6=Creazione
F9=Duplicazione  F10=Immiss. comandi   F23=Altre opzioni     F24=Altri tasti
Il membro DATWFXX è stato aggiunto al file QUTI in NERONIFIDS.              +
```

Portando il cursore sul messaggio e rollando in avanti, si vede anche un secondo messaggio.

```
 138 record copiati dal membro DATWF.
```

5.1 Elenchi di Librerie, Oggetti, Membri.

Il Pdm viene richiamato tramite alcuni comandi che in esecuzione presentano i tre elenchi fondamentali del programma.

- **WRKLIBPDM** Lavora con Elenco Pdm delle Librerie.
- **WRKOBJPDM** Lavora con Elenco Pdm degli Oggetti di una Libreria.
- **WRKMBRPDM** Lavora con Elenco Pdm dei Membri di un File Fisico.

Esiste anche un menù riassuntivo che, chiamando comunque uno a scelta dei tre elenchi menzionati, conserva la visibilità delle parzializzazioni eseguite nel run precedente.

- **STRPDM** Avvia Pdm.

Dall'elenco delle librerie l'opzione 12 chiama il Pdm sugli oggetti della libreria.

Dall'elenco degli oggetti l'opzione 12 su un file fisico chiama il Pdm sui membri del file.

5.2 Parzializzazione degli elenchi.

Noto che le complicazioni qui sotto illustrate mirano alla costruzione di elenchi ristretti, e perciò più utili, sui quali esercitare successivamente le opzioni desiderate.

Da un qualsiasi Elenco Pdm è possibile ottenere, tramite F17, una videata di parzializzazione. Il suo riempimento e l'invio provocano il ritorno al video di partenza che di lì in avanti presenta un elenco rispettoso delle parzializzazioni richieste.

Il ritorno a monte dell'elenco visualizzato neutralizza tutte le parzializzazioni che lo affliggono.

Un elenco a valle non risente delle parzializzazioni a monte. Cioè, ad esempio, l'elenco oggetti chiamato da un elenco librerie non risente delle parzializzazioni richieste a partire dall'elenco librerie.

WRKLIBPDM permette nel comando solo la parzializzazione su.

- Nome libreria generico esteso.

Ma, una volta presentato l'elenco, F17 permette una scelta più complessa.

- Nome libreria generico esteso, riportando la scelta già digitata sul comando per una diversa manipolazione.
- Tipo libreria.
- Testo libreria.

WRKOBJPDM permette nel comando la parzializzazione su.

- Nome libreria. Una sola per volta.
- Nome oggetto generico esteso.

Ma, una volta presentato l'elenco, F17 permette una scelta più complessa.

- Nome oggetto generico esteso, riportando la scelta già digitata sul comando per una diversa manipolazione.
- Tipo oggetto.
- Attributo oggetto generico esteso.
- Range per dimensione oggetto.
- Testo oggetto.

WRKMBRPDM permette nel comando la parzializzazione su.

- Nome qualificato file fisico. Uno solo per volta.

Ma, una volta presentato l'elenco, F17 permette una scelta più complessa.

- Nome membro generico esteso.
- Tipo membro generico esteso.
- Range per data modifica membro.
- Testo membro.

La locuzione "nome generico esteso" (ovvero nome generico secondo Pdm) significa che il nome designato come tale può contenere il carattere jolly * (asterisco) per provocare i seguenti comportamenti.

- Un asterisco in coda indica che si considerano solo i nomi che iniziano con i caratteri prima dell'asterisco (il concetto di nome generico secondo l'AS/400).
- Un asterisco in testa (ma solo se non manifesta la presenza di una parola riservata) indica che si considerano solo i nomi che finiscono con i caratteri dopo l'asterisco.
- Un asterisco in testa e uno in coda indicano che si considerano solo i nomi che contengono i caratteri tra i due asterischi.
- Un asterisco intermedio indica che si considerano solo i nomi che iniziano con i caratteri prima dell'asterisco e che finiscono con i caratteri dopo l'asterisco.

- Un doppio asterisco si usa in casi molto particolari per neutralizzare l'interpretazione di una stringa come parola riservata. Ad esempio, **ALL chiede l'elenco di tutti i valori che terminano con ALL perché lo spontaneo *ALL è già parola riservata che significa tutt'altro.

Gli stessi concetti si applicano anche al tipo oggetto generico esteso.

La scelta di una espressione alfanumerica come testo provoca la considerazione di quelle voci di elenco che portano un testo che contiene esattamente la stringa designata, senza riguardo al maiuscolo/minuscolo.

Tutte le parzializzazioni sono contemporaneamente considerate.

Si esemplifica un elenco di oggetti

```
                          Gestione degli oggetti con il PDM            S44524BA

Libreria. . . . .   NERONI____       Inizio elenco da. . . . . .   __________
                                     Inizio elenco da tipo . . .   __________

Immettere le opzioni e premere Invio.
  2=Modifica     3=Copia       4=Cancellaz.  5=Visualiz.     7=Ridemoninaz.
  8=Visualiz. descrizione      9=Salvatag.  10=Ripristino   11=Spostamento...

Opz  Oggetto     Tipo        Attributo   Testo
__   BNT01       *PGM        RPGLE       Bontà.
__   BTT01W      *FILE       DSPF        Bottoni.
__   BTTSRC      *FILE       PF-SRC      Bottoni. Source.
__   EVENT00F    *FILE       PF-DTA      Eventi.
__   RIGAD00F    *FILE       PF-DTA      Righe documenti.
__   TESTN01L    *FILE       LF          Testate dolenti.
__   TRAMV00P    *FILE       PRTF        Stampa del tram.

                                                                        Fine
Parametri o comando
===> ____________________________________________________________________________
F3=Fine           F4=Richiesta          F5=Rivisualizzaz.       F6=Creazione
F9=Duplicazione   F10=Immissione comandi    F23=Altre opzioni   F24=Altri tasti
```

Si batte F17 ed esce un video di parzializzazione.

```
                           Sottogruppo elenco oggetti

Immettere le scelte e premere Invio.

  Oggetto. . . . . . . . . . .   *ALL_____   *ALL, nome, *generico*

  Tipo oggetto . . . . . . . .   *ALL_____   *ALL, *tipo

  Attributo oggetto. . . . . .   *ALL_____   *ALL, attributo, *generico*,
                                             *BLANK
  Dimensione oggetto
    Da . . . . . . . . . . . .   _________0   0 - 9999999999
    A. . . . . . . . . . . . .   9999999999   0 - 9999999999



  Testo. . . . . . . . . . . .   *ALL_____________________________________



F3=Fine        F5=Rivisualizzazione        F12=Annullamento
```

Si completano le informazioni richieste e si preme invio.

```
                          Sottogruppo elenco oggetti

Immettere le scelte e premere Invio.

  Oggetto. . . . . . . . . . . .   *NT*______   *ALL, nome, *generico*

  Tipo oggetto . . . . . . . . .   *ALL______   *ALL, *tipo

  Attributo oggetto. . . . . . .   *ALL______   *ALL, attributo, *generico*,
                                                *BLANK
  Dimensione oggetto
    Da . . . . . . . . . . . . .   _________0   0 - 9999999999
    A. . . . . . . . . . . . . .   9999999999   0 - 9999999999



  Testo. . . . . . . . . . . . .   *ALL_____________________________________



F3=Fine        F5=Rivisualizzazione        F12=Annullamento
```

Viene mostrato l'elenco parzializzato.

```
                        Gestione degli oggetti con il PDM           S44524BA

Libreria. . . . .   NERONI____       Inizio elenco da. . . . . .   __________
                                      Inizio elenco da tipo . . .   __________

Immettere le opzioni e premere Invio.
  2=Modifica     3=Copia       4=Cancellaz.  5=Visualiz.     7=Ridemoninaz.
  8=Visualiz. descrizione      9=Salvatag.  10=Ripristino   11=Spostamento...

Opz  Oggetto     Tipo        Attributo   Testo
__   BNT01       *PGM        RPGLE       Bontà.
__   EVENT00F    *FILE       PF-DTA      Eventi.
__   TESTN01L    *FILE       LF          Testate dolenti.









                                                                        Fine
Parametri o comando
===> ____________________________________________________________________________
F3=Fine           F4=Richiesta          F5=Rivisualizzaz.       F6=Creazione
F9=Duplicazione   F10=Immissione comandi    F23=Altre opzioni   F24=Altri tasti
```

5.3 Scelte operabili sulle testate degli elenchi.

Sulla testata degli elenchi è possibile reimpostare il parametro principale, già scelto prima di entrare.

- Il tipo elenco sull'elenco librerie.
- La libreria sull'elenco oggetti.
- Il file qualificato sull'elenco membri.

Inoltre su tutti gli elenchi è possibile scegliere il punto di inizio dell'elenco stesso che, comunque, può essere scorso ancora anche all'indietro rispetto al nuovo punto di inizio.

- Il nome libreria sull'elenco librerie.
- Il nome ed il tipo oggetto sull'elenco oggetti.
- Il nome membro sull'elenco membri.

5.4 Modifica dei valori assunti.

Il Pdm utilizza per ogni specifico utente una serie di parametri ai quali l'utente stesso può accedere tramite F18 dai video del pacchetto. Vengono qui illustrati solo i parametri che usualmente si modificano, fidando nei default che il sistema assume per gli utenti nuovi.

Tali parametri entrano nella composizione dei comandi generati dall'esecuzione delle opzioni insieme ai dati prelevati dalla riga di elenco su cui si esercita l'opzione.

Libreria oggetti.

Quando richiesta la creazione di un oggetto tramite, ad esempio l'opzione 14, il Pdm compone un comando di creazione che genera l'oggetto nella libreria qui designata.

Sostituzione oggetto.

Ad evitare continue conferme al lancio delle creazioni, conviene richiedere la sostituzione senza conferma degli oggetti creati.

Descrizione lavoro.

Quando il Pdm necessita di eseguire la sottomissione di un lavoro batch, utilizza la descrizione lavoro qui designata.

Tramite la descrizione lavoro si scelgono ad esempio la coda di immissione e la coda di emissione del batch.

Noto che se si usa il Submit Job senza modifiche (lista iniziale librerie INLLIBL(*CURRENT)), la lista librerie del batch viene presa dalla lista librerie interattiva valida al momento della sottomissione.

Se invece si usa un Submit Job modificato con INLLIBL(*JOBD), la lista librerie del batch viene presa dalla descrizione lavoro qui designata.

File opzioni, Libreria, Membro.

È il contenitore delle opzioni personalizzate. Normalmente è meglio lasciare intatto il prototipo (file QAUOOPT, libreria QGPL, membro QAUOOPT) e duplicarlo in una libreria personale, magari omonima del profilo, e solo allora inserire le opzioni personalizzate spiegate poco oltre nel duplicato qui designato.

Registrazione comandi opzione.

Chiede al Pdm di inserire nella joblog interattiva i comandi eseguiti tramite opzione in modo che siano duplicabili con F9 come se fossero digitati.

5.5 Opzioni standard richiamabili da elenco.

Composto l'elenco nella maniera migliore e portate in vista le righe utili, il Pdm prevede un sostenuto numero di scelte possibili precostituite, tutte con codice numerico con una o due cifre. La stessa scelta su elenchi diversi esegue comandi diversi ma di significato simile. Così, anche sull'elenco oggetti, la stessa scelta esegue comandi diversi ma di tipo simile su tipi oggetti diversi. Il Pdm sceglie il comando in base al tipo elenco e alla scelta digitata e ne riempie i bastanti parametri facendo uso del contenuto della riga dell'elenco su cui la scelta è esercitata.

Ad esempio l'opzione 2 (Modifica) provoca, secondo il tipo elenco e il tipo oggetto, comandi diversi.

Tipo elenco	Tipo oggetto	Comando	Parametri
Lib	Libreria	CHGLIB	LIB(NomeLib) TYPE(TipoLib) TEXT(TestoLib)
Obj	File Fisico	CHGPF	FILE(NomeLib/NomeObj) TEXT(TestoObj)
Obj	Programma	CHGPGM	PGM(NomeLib/NomeObj) TEXT(TestoObj)
Mbr	Membro Dati	***	***
Mbr	Membro Sorgente	STRSEU	SRCFILE(NomeLib/NomeFile) SRCMBR(NomeMembro) OPTION(2)

Ad esempio l'opzione 14 (Crea), applicabile solo all'elenco membri, provoca, secondo il tipo seu, comandi diversi.

Tipo seu	Comando	Parametri
PF	CRTPF	FILE(NomeLib/NomeMembro) SRCFILE(NomeLib/NomeFile) SRCMBR(NomeMembro)
LF	CRTLF	FILE(NomeLib/NomeMembro) SRCFILE(NomeLib/NomeFile) SRCMBR(NomeMembro)
DSPF	CRTDSPF	FILE(NomeLib/NomeMembro) SRCFILE(NomeLib/NomeFile) SRCMBR(NomeMembro) REPLACE(*YES)
PRTF	CRTPRTF	FILE(NomeLib/NomeMembro) SRCFILE(NomeLib/NomeFile) SRCMBR(NomeMembro) REPLACE(*YES)
RPG	CRTRPGPGM	PGM(NomeLib/NomeMembro) SRCFILE(NomeLib/NomeFile) SRCMBR(NomeMembro) REPLACE(*YES)
RPGLE	CRTBNDRPG	PGM(NomeLib/NomeMembro) SRCFILE(NomeLib/NomeFile) SRCMBR(NomeMembro) REPLACE(*YES)
SQLRPG	CRTSQLRPG	PGM(NomeLib/NomeMembro) SRCFILE(NomeLib/NomeFile) SRCMBR(NomeMembro) REPLACE(*YES)
SQLRPGLE	CRTSQLRPGI	OBJ(NomeLib/NomeMembro) SRCFILE(NomeLib/NomeFile) SRCMBR(NomeMembro) OBJTYPE(*PGM) REPLACE(*YES)
CMD	CRTCMD	CMD(NomeLib/NomeMembro) PGM(*LIBL/NomeMembro) SRCFILE(NomeLib/NomeFile) SRCMBR(NomeMembro) REPLACE(*YES)
PNLGRP	CRTPNLGRP	PNLGRP(NomeLib/NomeMembro) SRCFILE(NomeLib/NomeFile) SRCMBR(NomeMembro) REPLACE(*YES)
CLP	CRTCLPGM	PGM(NomeLib/NomeMembro) SRCFILE(NomeLib/NomeFile) SRCMBR(NomeMembro) REPLACE(*YES)
CLLE	CRTBNDCL	PGM(NomeLib/NomeMembro) SRCFILE(NomeLib/NomeFile) SRCMBR(NomeMembro) REPLACE(*YES)
TBL	CRTTBL	TBL(NomeLib/NomeMembro) SRCFILE(NERONI1/NomeFile) SRCMBR(NomeMembro)

5.6 Personalizzazione delle opzioni.

Dagli elenchi Pdm è possibile accedere tramite F16 alla Gestione delle opzioni definite dall'utente.

Si voglia, ad esempio, inserire una opzione di richiamo di una utility di gestione dati nei file a partire dall'elenco oggetti e sul primo membro.

```
TWF FILE(NomeLib/NomeFile) MBR(*FIRST)
```

Tramite F6 si accede al video Creazione di una opzione definita dall'utente sul quale si codifica la nuova opzione e le si attribuisce un comando.

```
Opzione     UF
Comando     TWF FILE(&L/&N) MBR(*FIRST)
Dove:       &N  Viene sostituito con il nome file presente sulla riga.
            &L  Viene sostituito con il nome della libreria ove giace il file.
```

Si voglia, invece, inserire una opzione di richiamo della stessa utility di gestione dati nei file a partire però dall'elenco membri e su un membro specificamente scelto.

```
TWF FILE(NomeLib/NomeFile) MBR(NomeMembro)
Opzione     UM
Comando     TWF FILE(&L/&F) MBR(&N)
Dove:       &N  Viene sostituito con il nome membro presente sulla riga.
            &F  Viene sostituito con il nome del file ove giace il membro.
            &L  Viene sostituito con il nome della libreria ove giace il file.
```

Noto che le opzioni personalizzate non sono sensibili al tipo o sottotipo della riga dell'elenco Pdm.

L'elenco completo delle variabili &x è visionabile nell'aiuto del video di aggiunta e modifica dell'opzione personalizzata.

5.7 Esempio di Opzione personalizzata.

Da un elenco Pdm batti F16 ed ottieni l'elenco delle opzioni personalizzate già presenti.

```
                      Work with User-Defined Options                M170PUB1

 File . . . . . :   QAUOOPT          Member . . . . . :   QAUOOPT
   Library  . . . . :     NERONI1        Position to  . . . :   ____

 Type options, press Enter.
   2=Change         3=Copy         4=Delete         5=Display

 Opt  Option  Command
  =     NR    JNSTRPG SRCFILE(&L/&F) SRCMBR(&N) UPDTYPE(*BOTH)
  =     QF    RUNQRY QRYFILE((&L/&N)) RCDSLT(*YES)
  =     QM    RUNQRY QRYFILE((&L/&F &N)) RCDSLT(*YES)
  =     SJ    ?SBMDBJOB FILE(&L/&F) MBR(&N)
  =     UA    DSPOBJAUT OBJ(&L/&N) OBJTYPE(&T)



                                                                    Bottom
 Command
 ===> ____________________________________________________________________________
 F3=Exit             F4=Prompt           F5=Refresh        F6=Create
 F9=Retrieve         F10=Command entry                     F24=More keys
```

Batti F6. Compare un video di immissione della nuova opzione personalizzata.

```
                       Create User-Defined Option

 Type option and command, press Enter.

   Option  . . . . . .   __   Option to create

   Command . . . . . .   ____________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________

F3=Exit       F4=Prompt       F12=Cancel
```

Digita nuova opzione e comando corrispondente.

```
                       Create User-Defined Option

 Type option and command, press Enter.

   Option  . . . . . .   CO   Option to create

   Command . . . . . .   CHGOBJOWN__________________________________________
=============================================================================
=============================================================================
=====================================

F3=Exit       F4=Prompt       F12=Cancel
```

Batti F4: compare il prompter del comando scelto.

```
                        Change Object Owner (CHGOBJOWN)

 Type choices, press Enter.

 Object . . . . . . . . . . . . . OBJ           __________
   Library  . . . . . . . . . . .                  *LIBL______
 Object type  . . . . . . . . . . OBJTYPE        __________
 ASP device . . . . . . . . . . . ASPDEV         *_________
 New owner  . . . . . . . . . . . NEWOWN         __________
 Current owner authority  . . . . CUROWNAUT      *REVOKE____

                                                                        Bottom
 F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
 F24=More keys
```

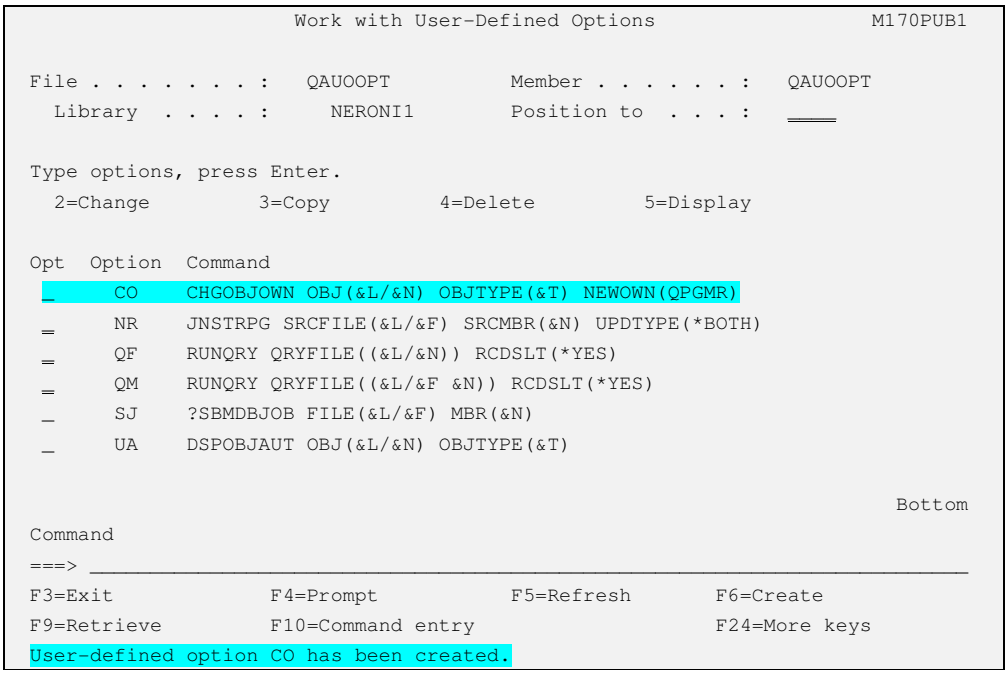
Completa il comando con le variabili e premi invio.

```
                        Change Object Owner (CHGOBJOWN)

 Type choices, press Enter.

 Object . . . . . . . . . . . . . OBJ           &N________
   Library  . . . . . . . . . . .                  &L_________
 Object type  . . . . . . . . . . OBJTYPE        &T_________
 ASP device . . . . . . . . . . . ASPDEV         *_________
 New owner  . . . . . . . . . . . NEWOWN         QPGMR______
 Current owner authority  . . . . CUROWNAUT      *REVOKE____

                                                                        Bottom
 F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
 F24=More keys
```

Sul video successivo premi ancora invio.

```
                        Create User-Defined Option

 Type option and command, press Enter.

   Option  . . . . . . . . .   CO   Option to create

   Command . . . . . . . . .   CHGOBJOWN OBJ(&L/&N) OBJTYPE(&T) NEWOWN(QPGMR) __
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________

F3=Exit       F4=Prompt       F12=Cancel
```

Visiona l'elenco arricchito con la nuova opzione.



5.8 Ripetizione di una opzione.

Spesso una opzione va ripetuta su tutto l'elenco o su una parte consistente e consecutiva di righe. I tal caso si digita l'opzione sulla prima riga e si duplica sul resto dell'elenco facendo uso di F13 e del cursore posizionato sull'opzione da ripetere. La duplicazione dell'opzione va fino al fondo dell'elenco. Per annullare la parte finale della ripetizione, basta posizionarsi su una riga, sbiancare manualmente l'opzione, mantenere il cursore sul campo sbiancato e ripetere due volte F13. Noto che la ripetizione delle opzioni numeriche non avviene sulle righe per le quali l'opzione non avrebbe significato.

Noto che F5, oltre a rinfrescare il contenuto dell'elenco riattingendo ai dati, spazza tutte le opzioni dall'elenco.

5.9 Scansione ed esecuzione di un'opzione a seguito della scansione.

L'opzione 25 permette la scansione e può affliggere solo i file fisici, normalmente sorgenti, eccezionalmente dati, sia attraverso l'elenco oggetti che attraverso l'elenco membri.

Se ne consiglia però l'uso solo sull'elenco membri poiché, in caso di interruzione a mezza via della funzione, la stessa viene più facilmente ripresa da questo elenco che non dall'altro, un po' cieco in questa azione.

Una opzione 25 non sopporta altre opzioni contemporanee diverse da 25.

L'invio sulle opzioni 25 provoca la comparsa di un video Ricerca di stringa per il quale si segnalano solo alcuni parametri.

Ricerca.

Si digita una stringa alfanumerica da cercare nei membri scelti.

Opzione.

Si sceglie una opzione Pdm da esercitare sui membri che contengono la stringa cercata.
La scelta 2 = Editazione permette l'innesco della funzione di sostituzione, che viene ricordata in ogni editazione successiva fino alla fine del run di ricerca.
La scelta 14 = Compilazione permette la ricompilazione di tutti gli oggetti che contengono una stringa.

5.10 Utilizzo sulle librerie sorgenti.

L'uso più notevole del Pdm è quello che permette il governo di un gruppo di sorgenti isolati in una libreria detta allora a sua volta "sorgente". Il Pdm permette la chiamata del SEU su ogni membro sorgente e facilita le compilazioni dei sorgenti per crearne gli oggetti da inserire nelle librerie dei programmi (dette a volte americanescamente oggetto) e nelle librerie dei dati. Le opzioni più frequenti sono.

- 2 = Editazione tramite SEU.
- 3 = Copia. Soprattutto per salvataggio.
- 4 = Cancellazione.
- 5 = Visualizzazione tramite SEU.
- 14 = Compilazione.
- 17 = Sda sui sorgenti video.

5.11 Utilizzo sulle librerie dati.

Sulle librerie dati il Pdm semplifica la maggior parte delle operazioni, soprattutto se multiple. Si esemplifica qualche caso senza nessuna pretesa di completezza.

- Indagine sul contenuto del sistema e delle librerie.
- Installazione mediante duplicazione.
- Copiatura e ridenominazione.
- Modifiche di massa alle caratteristiche dei file, ad esempio SIZE(*NOMAX) dimenticati in compilazione.
- Applicazione del giornale ai file fisici.

5.12 Utilizzo sulle librerie programmi.

Sulle librerie programmi il Pdm semplifica qualche operazione di massa sempre del tipo attribuzione caratteristiche dimenticate in compilazione.

6 File Fisici. (06C)

Il file fisico è il contenitore vero dei dati e consiste dei seguenti elementi.

- I record contenitori dei dati.
- I membri contenitori dei record.
- La definizione del tracciato secondo il quale sono scritti i record.
- La definizione delle chiavi di accesso al file.

Le **DDS** (Data Description Specification, *Specifiche di Descrizione dei Dati*) servono a definire i vari tipi di file, costituiscono un linguaggio a sé stante e, unitamente agli appositi comandi di creazione, permettono di inserire i file di database nel sistema.

Un file fisico può contenere un solo tracciato record.

Un programma RPG può conoscere il file dichiarandolo nelle specifiche File come file descritto esternamente. Il tracciato del file verrà allora incorporato nel programma che potrà compiere operazioni di lettura, scrittura e ricalco sui dati con pieno rispetto del tracciato stesso.

Se invece nell' RPG si dichiara il file come descritto internamente, si devono inserire le apposite specifiche di immissione (0) e di emissione (20) e non si ha alcun rispetto automatico del tracciato ma solo la coincidenza intenzionale di definizioni continuamente ritrascritte a mano.

Nei fisici, pur essendo possibile, è meglio non inserire chiavi di accesso. Un fisico senza chiavi si copia più rapidamente e occupa meno spazio. Delle chiavi si parlerà quindi nei file logici (7) sia per la definizione che per l'uso.

6.1 Schema di minutazione del sorgente del file fisico.

Per esemplificare, si definisce un file fisico composto da campi di vari tipi. Le specifiche di definizione dei dati, qui riportate, vengono scritte in un sorgente che, a sua volta, è il membro di un apposito file fisico sorgente.

```
PF  A.........T.Nome++++++RLun++TPdB......Funzioni++++++++++++++++++++++++++++++++
    A           R FMTRCD
    A             FLDALF         12
    A             FLDPAK          7P 2
    A             FLDSIG          5S 0
    A             FLDBIN         10B 0
```

Dove:	A	=Specifica DDS di definizione file.			
	R	=Specifica di definizione del record			
	FMTRCD	=Nome del tracciato.			
	FLDALF	=Nome di campo alfanumerico	lungo 12	da 1	a 12
	FLDPAK	=Nome di campo numerico impaccato	lungo 7 con 2 decimali	da 13	a 16
	FLDSIG	=Nome di campo numerico segnato	lungo 5 con 0 decimali	da 17	a 21
	FLDBIN	=Nome di campo numerico binario	lungo 10 con 0 decimali	da 22	a 25

Le dimensioni indicate per i campi sono secondo necessità e i campi sono giustapposti nel tracciato. I vari tipi di campo occupano un adeguato numero di byte.

- Alfanumerico: tanti byte quanta la lunghezza.
- Numerico impaccato: si aumenta la lunghezza di uno, se dispari si aumenta ancora di uno, si divide a metà.
- Numerico segnato: tanti byte quanta la lunghezza.
- Numerico binario: due fino a 9999, quattro fino a 9.999.999.999.

I nomi usati nel sorgente non possono superare la lunghezza di dieci caratteri, ma è ancora uso non superare 8 per i nomi di tracciato e 6 per i nomi di campo per poterli usare senza rinominazioni dentro il vecchio RPG.

Le posizioni iniziali e finali dei campi lungo il tracciato sono indicate ma non sono necessarie durante il trattamento dei dati presenti nei campi e perciò non verranno più indicate.

6.2 Convenzioni di denominazione.

Il database è dotato di un numero notevole di contenitori di svariati tipi che occorre denominare con qualche regola. Si deve adottare un metodo facile, ripetibile e univoco di assegnazione di nomi appropriati ad ogni contenitore che si fa nascere.

Ma non basta stabilire una regola, occorre che essa sia ragionevole, semplice, autoesplicativa ed omnicomprensiva. Solo così può essere condivisa e diventare una convenzione a cui attenersi per essere capiti al volo da altri sviluppatori.

Più che una certa convenzione, è indispensabile avere una convenzione omogenea nell'ambito più largo possibile: in un'applicazione, in un pacchetto, in tutti i pacchetti di una casa di software o di una installazione.

Gli esempi sotto riportati sono veramente in uso e meritano perciò una attenta considerazione nell'ambito dello sviluppo applicativo.

Si nota che le convenzioni nascono per il database ma poi si estendono spesso anche alla denominazione dei campi interni ai programmi. E' però preferibile non dettare al programmatore una regola ferrea per tali nomi perché così facendo ci si impiccia dei procedimenti logici molto personali che non possono fare a meno di una certa libertà dentro i programmi.

- Nel Cobol, che permette nomi di 30 caratteri, viene spontaneo usare nomi di campo che somigliano più ad un testo che ad un nome. Il database AS/400 recepisce il nome Cobol nella parola chiave a livello di campo ALIAS che contiene una stringa alfanumerica usabile come nome di campo.
- Nel Control Language la denominazione dei campi interni di un programma si adegua spesso all'uso della variabile che così adotta come nome quello della parola chiave in cui viene usata. Ad esempio, se nel programma serve una variabile da fornire al comando seguente.

  DLYJOB DLY(variabile)

  La variabile numerica di 6 cifre con zero decimali, che indica i secondi di attesa si chiamerà facilmente &DLY ed il comando si scriverà come segue.

  DLYJOB DLY(&DLY)
- Nelle versioni RpgIle dopo il rilascio 430, i nomi interni ai programmi si sono ulteriormente allungati a 14 caratteri. Capita così di prefissare o suffissare nomi già stabiliti nel database per scopi di salvataggio, copia, transito di singoli dati o di record interi. Tali nomi, evidentemente, non rispettano e non devono rispettare le regole del database ma regolette interne al singolo programma, con buona pace delle convenzioni.

Nei paragrafi successivi si illustra una sola convenzione che, per la parte file e record, rispetta le ACG (Applicazioni Contabili Gestionali) dell'IBM Italia mentre per i campi, essendo il database ACG di mente Cobolistica, cioè senza univocità, riporta l'esperienza personale di chi scrive, molto prossima all'uso italiano più esperto.

6.3 Criteri di denominazione dei file fisici di database.

La convenzione più in uso per la denominazione dei file fisici consiste nell'attribuzione di una radice di 5 caratteri più il suffisso 00F. La radice simboleggia il tipo di soggetti archiviati nel file mediante un acronimo. **ANPEROOF**, ad esempio, potrebbe significare **AN**agrafico **PER**sone; oppure **ANPRO00F AN**agrafico **PRO**fessioni.

6.4 Criteri di denominazione dei record.

Riferendoci sempre alla convenzione più comune e continuando l'esempio intrapreso, il nome del tracciato è solitamente costituito dalla sola radice, **ANPER** e **ANPRO**, nei casi visti.

6.5 Criteri di denominazione dei campi.

L' RPG, al momento della lettura da database, copia tutti i campi del tracciato, letto dal buffer di input, direttamente nei campi in memoria. Per buffer si intende la zona di memoria in cui il programma mette i dati così come li legge da database. Il buffer non è comunque accessibile nei programmi RPG mentre è accessibile nei programmi Cobol.

E' un uso Cobolistico dare a campi simili di file diversi lo stesso nome, perché il Cobol non trascrive nulla automaticamente dal buffer ai campi in memoria.

E' invece particolarmente utile in RPG che il nome di un campo sia del tutto unico nel database. Dichiarando più file nello stesso RPG, non si correrà mai il rischio, leggendo un record dal database, di riempire di dati un campo che figura anche in un altro tracciato.

Contro le eventuali omonimie, può essere utile la ridenominazione dei campi, singola (vedi Ridenominazione su Specifiche Immissione) o, meglio, a livello di file (vedi Prefix su Specifiche File).

Ad evitare voli di fantasia più tardi incomprensibili, occorre sempre stabilire delle regole che conducano senza fatica alla denominazione dei campi.

Ad esempio, si dia la **regola** che il nome di un campo **ffppss** risulti composto da 3 parti.

- **ff** 2 caratteri Unico sul database a significare il file di appartenenza.
- **pp** 2 caratteri Abbreviazione della prima parola del testo del campo.
- **ss** 2 caratteri Abbreviazione della seconda parola del testo del campo.

Dati i testi descrittivi relativi ai campi contenuti in due file esemplificativi.

File Persone.

- Annullamento
- Codice Persona.
- Nome Persona.
- Data Nascita.
- Codice Professione.
- Titolo Studio.
- Sesso.

File Professioni.

- Annullamento
- Codice Professione.
- Descrizione Professione.
- Tariffa Minima.
- Tariffa Massima.

Si estrae un dizionario delle parole usate e a ciascuna si attribuisce un'abbreviazione.

Dizionario delle parole.

- AN = Annullamento
- CD = Codice
- DT = Data
- DS = Descrizione
- NN = (Nullo)
- MN = Minima
- MX = Massima
- NM = Nome
- NS = Nascita
- PE = Persona
- PR = Professione
- SE = Sesso

- ST = Studio
- TI = Titolo
- TF = Tariffa

Si compongono infine i nomi dei campi secondo la regola.

File Persone (ANPER00F).

- **P1ANNN** P1 = File Persone AN = Annullamento
- **P1CDPE** P1 = File Persone CD = Codice PE = Persona
- **P1NMPE** P1 = File Persone NM = Nome PE = Persona
- **P1DTNS** P1 = File Persone DT = Data NS = Nascita
- **P1CDPR** P1 = File Persone CD = Codice PR = Professione
- **P1TIST** P1 = File Persone TI = Titolo ST = Studio
- **P1SENN** P1 = File Persone SE = Sesso

File Professioni (ANPRO00F).

- **P2ANNN** P2 = File Professioni AN = Annullamento
- **P2CDPR** P2 = File Professioni CD = Codice PR = Professione
- **P2DSPR** P2 = File Professioni DS = Descrizione PR = Professione
- **P2TFMN** P2 = File Professioni TF = Tariffa MN = Minima
- **P2TFMX** P2 = File Professioni TF = Tariffa MX = Massima

E' chiaro, quindi, che stabiliti testi semplici di due parole ed un dizionario di parole abbreviate senza ripetizioni, è possibile assegnare nomi mnemonici, cioè facili da ricordare. Durante la minutazione dei programmi non si è quindi costretti a rileggere continuamente i tracciati per ritrovare i nomi assegnati.

Attribuiti ai campi tipi e dimensioni opportune, i sorgenti dei due fisici potrebbero essere i seguenti.

```
PF  A.........T.Nome++++++RLun++TPdB......Funzioni++++++++++++++++++++++++++++++
     * File ANPER00F
A           R ANPER
A             P1ANNN          1
A             P1CDPE          3
A             P1NMPE         25
A             P1DTNS          8P 0
A             P1CDPR          5
A             P1TIST         20
A             P1SENN         20
```

```
PF  A.........T.Nome++++++RLun++TPdB......Funzioni++++++++++++++++++++++++++++++
     * File ANPRO00F
A           R ANPRO
A             P2ANNN          1
A             P2CDPR          5
A             P2DEPR         30
A             P2TFMN          9P 2
A             P2TFMX          9P 2
```

Si noti che le descrizioni composte da una sola parola generano un nome di campo con un suffisso **NN** e che da eventuali descrizioni di tre o più parole si ricavano solo le due più significative per mantenere tutti i nomi della stessa lunghezza.

6.6 Uso del Dizionario.

Nel seguito, per **database** si intenderà non l'universo dei file presenti su di un sistema ma il gruppo grande o piccolo di file necessari ad una applicazione.

Come si è visto sopra, nella costruzione di un database si possono definire i campi dei file fisici direttamente all'interno dei loro sorgenti, dichiarando in tale sede tipo, lunghezza, decimali, intestazioni, testo, valori, escursioni, edit.

Oppure, la definizione può avvenire in un file fisico prototipo, destinato non a contenere dati ma soltanto definizioni. Il prototipo viene chiamato **dizionario** e contiene tutti i tipi di campi menzionati nel database ma privi del prefisso identificativo del file. I singoli campi dei file fisici sono allora dichiarati nel sorgente di competenza con riferimento al dizionario.

L'esempio precedente potrebbe essere minutato come segue. Si osservi l'ordinamento alfabetico dei nomi di campo del dizionario a contrasto con l'ordine logico—applicativo dei file fisici.

```
PF  A.........T.Nome++++++RLun++TPdB......Funzioni++++++++++++++++++++++++++++++
     * File DIZIO
A           R DIZIOR
A             ANNN            1
A             CDPE            3
A             CDPR            5
A             DEPR           30
A             DTNS            8P 0
A             NMPE           25
A             SENN            1
A             TFMN            9P 2
A             TFMX            9P 2
A             TIST           20
PF  A.........T.Nome++++++RLun++TPdB......Funzioni++++++++++++++++++++++++++++++
     * File ANPER00F
A                                       REF(DIZIO)
A           R ANPER
A             P1ANNN    R               REFFLD(ANNN)
A             P1CDPE    R               REFFLD(CDPE)
A             P1NMPE    R               REFFLD(NMPE)
A             P1DTNS    R               REFFLD(DTNS)
A             P1CDPR    R               REFFLD(CDPR)
A             P1TIST    R               REFFLD(TIST)
PF  A.........T.Nome++++++RLun++TPdB......Funzioni++++++++++++++++++++++++++++++
     * File ANPRO00F
A                                       REF(DIZIO)
A           R ANPRO
A             P2ANNN    R               REFFLD(ANNN)
A             P2CDPR    R               REFFLD(CDPR)
A             P2DEPR    R               REFFLD(DEPR)
A             P2TFMN    R               REFFLD(TFMN)
A             P2TFMX    R               REFFLD(TFMX)
```

6.7 Caratteristiche aggiuntive attribuibili ai campi di un file fisico.

Le caratteristiche illustrate nei prossimi paragrafi vengono incorporate nel file fisico o nel dizionario a volte senza alcun effetto immediato. Vi si inseriscono, tuttavia, perché vengano ereditate tramite la funzione di riferimento (REF e REFFLD implicito o esplicito).

Esse riaffioreranno nei file video e nei file stampa che conservano il riferimento, nonché in alcune utility come il Query e il Dfu.

6.8 Intestazioni di colonna e Testo del campo.

Per migliorare la leggibilità dei campi utilizzati nei file fisici e, a maggior ragione, nei dizionari, è possibile attribuire ad ogni campo una intestazione di colonna ed un testo.

```
PF  A.........T.Nome++++++RLun++TPdB......Funzioni++++++++++++++++++++++++++++++
A             CDPR            5
A                                       COLHDG('Codice' +
A                                              'professione')
A                                       TEXT('Codice della professione.')
```

Il testo del campo è una costante descrittiva lunga fino a 50 caratteri.

L'intestazione di colonna è costituita da una, due o tre stringhe, ciascuna lunga fino a 20 caratteri.

Se il testo non viene attribuito, viene assunta come testo la giustapposizione delle tre stringhe della intestazioni di colonna, naturalmente fino al totale di 50 caratteri.

Nei programmi di servizio e di utilità (Query, Dfu, Sda) si fa largo uso dell'intestazione non solo sopra (corta e breve come il campo) ma anche a sinistra del campo (lunga e di lunghezza omogenea con le altre). E' perciò normalmente preferibile la soluzione lunga, più comprensibile e riutilizzabile. Le intestazioni sono comunque sempre modificabili al momento dell'uso.

Circa l'intestazione e il testo dei campi, conviene quindi.

- Usare parole intere e non abbreviazioni, sempre poco chiare.
- Limitarsi a due o tre parole, una per ciascuna delle tre stringhe possibili, preferibilmente verbi e sostantivi senza preposizioni ed articoli.
- Evitare la parola chiave TEXT e lasciare che venga composta automaticamente a partire dal COLHDG, evitando così l'onere di battere due volte le stesse parole.
- Nel Query abbreviare eventualmente solo al momento dell'uso, per ottenere una intestazione sopra il campo non più larga del campo stesso.
- Nell'Sda abbreviare eventualmente solo al momento dell'uso, per ottenere una intestazione a sinistra del campo della stessa larghezza delle intestazioni dei campi soprastanti.

6.9 Codici di edizione.

Se il campo è numerico, è sempre fortemente consigliabile attribuirgli un codice di edizione (Edit Code) o una parola di edizione (Edit Word).

```
PF  A.........T.Nome++++++RLun++TPdB......Funzioni++++++++++++++++++++++++++++++
    A            TFMN           9P 2
    A                                      COLHDG('Tariffa' +
    A                                             'minima')
    A                                      EDTCDE(K)
    A            DTNS           8P 0
    A                                      COLHDG('Data' +
    A                                             'nascita')
    A                                      EDTWRD('    /  /  ')
```

Se il campo Tariffa minima contiene "1234567,89", l'edit code K ne provoca la visualizzazione nella forma "1.234.567,89"o "1.234.567,89-", se negativo.

Se il campo Data nascita contiene "20020227", l'edit word indicata ne provoca la visualizzazione nella forma "2002/02/27".

Per il significato degli edit code e delle edit word, vedi il capitolo Edit (34).

6.10 Valori permessi.

Se per il campo sono previsti un numero limitato di valori, è moderatamente consigliabile attribuirgli un elenco di valori (VALUES) o un'escursione (RANGE), utili non tanto per il controllo che provocano quando affliggono il video, ma per il promemoria che costituiscono per lo sviluppatore che può interrogare il dizionario o il file fisico con il comando DSPFFD (Display File Field Description) per sapere come comportarsi nei controlli a programma.

```
PF  A.........T.Nome++++++RLun++TPdB......Funzioni++++++++++++++++++++++++++++++
    A            SENN           1
    A                                      COLHDG('Sesso')
    A                                      VALUES('M' 'F' 'C')
    A            DTNS           8P 0
    A                                      COLHDG('Data' +
    A                                             'nascita')
    A                                      EDTWRD('    /  /  ')
    A                                      RANGE(18800101 20501231)
```

Se nel campo Sesso a video viene digitato un valore diverso dai tre previsti o se nel campo Data nascita a video viene digitato un valore non compreso tra i limiti indicati, il sistema rifiuta la modifica senza chiedere l'intervento del programma applicativo.

Tuttavia i metodi di controllo illustrati tolgono ai programmi la piena potestà di eseguire o non eseguire il controllo che, a volte viene innescato involontariamente digitando o ricalcando un valore nel campo. Negli schemi di programmazione appena complessi tale tecnica di controllo provoca inconvenienti più grossi dei benefici ed è perciò opportuno evitarla. Nelle specifiche del video, a livello di campo, si fa allora uso della parola chiave DLTCHK (Delete Check) che elimina l'eredità di tali controlli dal database al video.

6.11 Schema finale di un Dizionario.

Riunendo nel dizionario tutte le caratteristiche desiderate, si ottiene un sorgente simile al seguente.

```
PF  A.........T.Nome++++++RLun++TPdB......Funzioni++++++++++++++++++++++++++++++
     * Dizionario delle parole.
     *·  CD = Codice
     *·  DT = Data
     *·  DS = Descrizione
     *·  NN = (Nullo)
     *·  MN = Minima
     *·  MX = Massima
     *·  NM = Nome
     *·  NS = Nascita
     *·  PE = Persona
     *·  PR = Professione
     *·  SE = Sesso
     *·  ST = Studio
     *·  TI = Titolo
     *·  TF = Tariffa
     *
    * Dizionario dei campi.
    A          R DIZIOR
    A                                      TEXT('Dizionario.')
    A            ANNN           1
    A                                      COLHDG('Annullamento')
    A                                      VALUES(' ' 'A')
    A            CDPE           3
    A                                      COLHDG('Codice' +
    A                                             'persona')
    A            CDPR           5
    A                                      COLHDG('Codice' +
    A                                             'professione')
    A            DEPR          30
    A                                      COLHDG('Descrizione' +
    A                                             'professione')
    A            DTNS           8P 0
    A                                      COLHDG('Data' +
    A                                             'nascita')
    A                                      EDTWRD('    /  /  ')
    A                                      RANGE(18800101 20501231)
    A            NMPE          25
    A                                      COLHDG('Nome' +
    A                                             'persona')
    A            SENN           1
    A                                      COLHDG('Sesso')
    A                                      VALUES('M' 'F' 'C')
    A            TFMN           9P 2
    A                                      COLHDG('Tariffa' +
    A                                             'minima')
    A                                      EDTCDE(K)
    A            TFMX           9P 2
    A                                      COLHDG('Tariffa' +
    A                                             'massima')
    A                                      EDTCDE(K)
    A            TIST          20
    A                                      COLHDG('Titolo' +
    A                                             'studio')
```

Si riportano i sorgenti dei due file perché si vuole evidenziare che anche ai record è possibile ed opportuno attribuire un testo.

Naturalmente la creazione dei file deve avvenire dopo la creazione dell'ultima edizione del dizionario, allo scopo di incorporare le nuove caratteristiche dei campi.

```
PF   A.........T.Nome++++++RLun++TPdB......Funzioni++++++++++++++++++++++++++++++
     A* File ANPER00F
     A                                      REF(DIZIO)
     A          R ANPER
     A                                      TEXT('Persone.')
     A            P1ANNN    R             REFFLD(ANNN)
     A            P1CDPE    R             REFFLD(CDPE)
     A            P1NMPE    R             REFFLD(NMPE)
     A            P1DTNS    R             REFFLD(DTNS)
     A            P1CDPR    R             REFFLD(CDPR)
     A            P1TIST    R             REFFLD(TIST)
```

```
PF   A.........T.Nome++++++RLun++TPdB......Funzioni++++++++++++++++++++++++++++++
     A* File ANPRO00F
     A                                      REF(DIZIO)
     A          R ANPRO
     A                                      TEXT('Professioni.')
     A            P2ANNN    R             REFFLD(ANNN)
     A            P2CDPR    R             REFFLD(CDPR)
     A            P2DEPR    R             REFFLD(DEPR)
     A            P2TFMN    R             REFFLD(TFMN)
     A            P2TFMX    R             REFFLD(TFMX)
```

6.12 Create Physical File.

Il passaggio dal sorgente all'oggetto File Fisico avviene tramite il comando di creazione CRTPF (**Create Physical File**). Esso costruisce un oggetto conforme al sorgente, per quanto riguarda tracciato e campi, e conforme ai parametri per quanto riguarda il file. Sorgente e comando di creazione del file fisico, come per molti altri oggetti, cooperano nella definizione dell'oggetto finale.

```
CRTPF                                  Create Physical File
      FILE(PHYLIB/PHYFILE)                   Physical Library/Physical File
      SRCFILE(SRCLIB/QDDSSRC)                Source Library/Source File
      SRCMBR(*FILE)                          Source Member
      MBR(*FILE)                             Physical Member
      TEXT(*SRCMBRTXT)                       Physical File Text
      MAXMBRS(1)                             Max Members
      SIZE(*NOMAX)                           Size
      REUSEDLT(*NO)                          Reuse Deleted
```

Segue una breve illustrazione delle singole parole chiave usate nella creazione del file fisico.

Physical File	Il nome del file fisico da creare.
Physical Library	Il nome della libreria destinata a contenere il file fisico da creare.
Source File	Il nome del file sorgente.
Source Library	Il nome della libreria contenente il file sorgente.
Source Member	Il nome del membro contenente il sorgente, oppure. *FILE Lo stesso nome presente nel parametro Physical File.
Physical Member	Il nome del membro da aggiungere nel file fisico da creare, oppure. *FILE Lo stesso nome presente nel parametro Physical File.
Physical File Text	Il testo da attribuire al file fisico da creare, oppure. *SRCMBRTXT Lo stesso testo del membro sorgente.
Max Members	Il numero massimo di membri che saranno tollerati nel file fisico da creare, da 1 a 32767, oppure. *NOMAX Il massimo permesso dal sistema (32767).
Size	Il numero massimo di record che saranno tollerati nei membri del file fisico da creare. Il parametro permette tre valori numerici. Il primo indica il

numero iniziale di record del membro. Il secondo indica il numero di record in un incremento. Il terzo indica il numero di incrementi permessi. I tre valori assunti per difetto sono 10000, 1000, 3. Tale balletto di numerini ha poco scopo pratico ed è perciò preferibile assegnare al parametro il valore singolo seguente.

*NOMAX Il massimo numero di record permesso dal sistema.

Reuse Deleted Riutilizzazione del posto occupato dai record cancellati.

*NO Non riutilizzare i buchi.

*YES Riutilizzare i buchi.

6.13 Create Physical File senza sorgente.

E' possibile anche creare un file senza sorgente, e quindi senza tracciato o, perlomeno, con un tracciato semplificato di un campo solo omonimo del tracciato e del file.

```
CRTPF                                  Create Physical File
      FILE(PHYLIB/PHYFILE)                    Physical Library/Physical File
      RCDLEN(nnn)                             Record Length
```

Dove valgono le stesse parole chiave del soprastante Create Physical File con sorgente, ad eccezione di SRCFILE e SRCMBR, sostituiti da RCDLEN.

Record Length Lunghezza del tracciato record.

Questo tipo di file fisico è destinato a far da contenitore a dati da leggere nei programmi RPG tramite descrizione interna.

6.14 Level check.

Dal tracciato di un file fisico viene sempre ricavata una espressione alfanumerica di 13 caratteri esadecimali che sintetizza il tipo, il numero, le caratteristiche e i nomi dei campi e del tracciato stesso. Tale espressione, il **level check**, viene incorporata dal compilatore nel programma che, al momento dell'esecuzione, controlla che i file in corso di apertura siano dotati ancora di record con lo stesso level check. In caso di errore, avviene una segnalazione terminale. E' possibile, sotto certe regole, rinunciare al controllo indicando LVLCKK(*NO) nella creazione, nella modifica o nel reindirizzamento del file, ma questo equivale a pretendere di vincere alla roulette russa sempre. Per igiene, si sconsiglia la pratica e non se ne danno particolari.

Il level check di un tracciato non cambia al cambiare di parole chiavi come COLHDG, TEXT, EDTCDE, EDTWRD, VALUES, RANGE perché esse non toccano le sue caratteristiche costitutive. Tanto meno sul level check influisce la macchina di compilazione.

Il sistema calcola il level check per tutti i record di tutti i tipi di file e ne fa nei programmi l'uso illustrato. Tanto vale per fisici, logici, display e printer file, per i quali non si ripeterà più quanto detto qui.

7 File Logici. (07C)

Il file logico non è un contenitore di dati ma soltanto una vista sui dati contenuti nel file fisico. Esso permette l'accesso ai dati stessi in modo simile a quanto si fa con il file fisico e consiste dei seguenti elementi.

- I record logici che indirizzano ai record fisici.
- I membri logici contenitori dei record logici.
- La definizione del tracciato del logico secondo il quale sono visti i record fisici.
- La definizione delle chiavi di accesso al logico e quindi dell'ordinamento dei record logici.
- Le inclusioni e le esclusioni di gruppi di record fisici dalla vista costituita dal logico.

Come per i fisici, le **DDS** (Data Description Specification, *Specifiche di Descrizione dei Dati*) servono a definire le caratteristiche dei file logici e, unitamente agli appositi comandi di creazione, permettono di inserirli nel sistema.

Per quanto riguarda la dichiarazione dei file logici nell' RPG, valgono le stesse considerazione fatte per i file fisici. Se il logico è dotato di chiavi e l' RPG le richiede, anche la definizione delle chiavi viene incorporata automaticamente nel programma.

Un file logico contiene normalmente uno o, eccezionalmente, più tracciati record.

Si definiscono qui di seguito le **chiavi** attraverso l'uso che se ne fa.

- Per chiave di un file si intende il campo o l'elenco di campi conoscendo il contenuto dei quali un programma è in grado di recuperare tutto il record mediante apposite operazioni.
- Se inoltre il file è dotato di chiavi e un programma legge il file in sequenza di chiavi, la lettura avverrà nell'ordine richiesto dal contenuto dei campi chiave.
- Le chiavi permettono anche il posizionamento preliminare lungo il file logico prima di una lettura sequenziale.

Si può immaginare un file logico con chiavi come un duplicato del file fisico ordinato secondo le chiavi, con la differenza che al posto di ogni record fisico si trova l'indirizzo al record fisico.

Particolari informazioni sono richieste circa la chiave sulle specifiche File dei file dichiarati internamente. Tali informazioni, di uso ormai improbabile, sono completamente omesse nel presente corso.

7.1 Schema di minutazione del sorgente del file logico.

Per esemplificare, si definisce un file logico basato sull'ultimo esempio riportato nei file fisici (6.11). Le specifiche di definizione dei dati, qui riportate, vengono scritte in un sorgente che, a sua volta, è il membro di un apposito file fisico sorgente.

```
LF  A.........T.Nome++++++.Lun++TPdB......Funzioni++++++++++++++++++++++++++++++
    A* File ANPER02L
    A          R ANPER
    A                                      PFILE(ANPER00F)
    A          K P1CDPE
    A          O P1ANNU                    CMP(EQ 'A')
```

Dove:
- A = Specifica DDS di definizione file.
- R = Specifica di definizione del record
- ANPER = Nome del tracciato.
- PFILE = Il file logico si basa sul file fisico ANPER00F
- K = Identifica il campo P1CDPE sul quale viene costruita la chiave.
- O = Omette quei record che nel campo P1ANNU portano il valore "A".

Si nota che i campi chiave possono essere molteplici e così pure le regole di omissione.

7.2 Puntamento dei logici ai fisici.

Si nota che il file fisico di base del logico viene cercato in lista librerie al momento della compilazione e quindi stabilmente agganciato dal logico, anche se l'uno o l'altro vengono mossi

altrove successivamente. E' permesso ma sconsigliabile usare un nome qualificato per il file fisico di base, come è sconsigliabile ogni puntamento ai nomi di librerie da parte di ogni tipo di parola chiave dentro le tutte le specie di DDS.

Conviene osservare che in RPG sia la compilazione che l'esecuzione, in assenza di forzature (i reindirizzamenti) comunque non eseguibili tramite le sole istruzioni RPG, cercano sempre in lista librerie il file dichiarato nelle specifiche File.

Si annota, di passaggio, che qualunque nome qualificato è normalmente sconsigliabile in qualunque tipo di sorgente.

7.3 Residenza dei logici rispetto ai fisici.

Come ogni oggetto, il file fisico risiede in una libreria e, di primo acchito, non si pongono vincoli sulla libreria di residenza dei logici che lo sovrastano.

Tuttavia, in presenza della divaricazione delle librerie, si presentano una serie di fenomeni indesiderati.

- Durante la duplicazione per prova di un sistema informativo o per l'avviamento di uno nuovo, il ripristino delle due librerie, l'una contenente i fisici e l'altra contenente i logici, fallisce gravemente perché il risultato finale è una libreria duplicata di logici che puntano alla libreria originale dei fisici. Stesso inconveniente anche praticando la duplicazione oggetti in linea.
- Il ripristino della libreria contenente logici fuori posto diventa suddito del ripristino della libreria dei fisici anche nel caso di ricaricamento dopo un crash. Prima occorre ricaricare la libreria dei fisici e poi quella dei logici, pena la non rigenerazione dei logici. Si nota che il crash è un evento traumatico in vista del quale è sciocco apparecchiare gratuite difficoltà aggiuntive.
- Uno degli errori più faticosi da diagnosticare è l'utilizzo nello stesso programma di un fisico e di un suo logico quando, per accidente, il logico non punta al medesimo fisico. Se si separano logici e fisici, tale errore diventa ancora più machiavellico.

Il consiglio fondamentale è quindi usare una sola libreria per fisici e logici.

Si coglie l'occasione per raccomandare di valutare le argomentazioni favorevoli alla separazione di fisici e logici con occhio particolarmente critico. Facilmente discendono da strateghi da tavolino.

7.4 Alterazioni di formato rispetto ai fisici.

Se il sorgente del logico non menziona alcun campo dati, è come se li menzionasse tutti ed il logico ha allora lo stesso tracciato del fisico e, quindi, lo stesso Level Check.

Se nel logico, per ragioni di riservatezza, si vuole includere solo un sottoinsieme dei campi del fisico, a fronte del solito fisico dell'esempio, si possono elencare esplicitamente i campi da includere.

```
LF  A.........T.Nome++++++.Lun++TPdB......Funzioni++++++++++++++++++++++++++++++
    A* File ANPER12L
    A          R ANPER
    A                                      PFILE(ANPER00F)
    A            P1ANNN
    A            P1CDPE
    A            P1NMPE
    A            P1DTNS
    A            P1SENN
    A          K P1CDPE
    A          O P1ANNU                    CMP(EQ 'A')
```

Se un logico usufruisce dello stesso tracciato già definito in un altro logico sullo stesso fisico, si può usare la parola chiave FORMAT.

```
LF  A.........T.Nome++++++.Lun++TPdB......Funzioni++++++++++++++++++++++++++++++
    A* File ANPER13L
    A          R ANPER
    A                                    PFILE(ANPER00F)
    A                                    FORMAT(ANPER12L)
    A          K P1NMPE
    A          K P1CDPE
    A          O P1ANNU                  CMP(EQ 'A')
```

7.5 Tipi di chiavi.

Il primo e più importante logico sarà sempre quello che porta la chiave unica principale e non esclude gli annullati. Esso sarà particolarmente utile nel programma fondamentale di manutenzione anagrafica.

```
LF  A.........T.Nome++++++.Lun++TPdB......Funzioni++++++++++++++++++++++++++++++
    A* File ANPER01L
    A                                    UNIQUE
    A          R ANPER
    A                                    PFILE(ANPER00F)
    A          K P1CDPE
```

Dove UNIQUE =Le chiavi definite nelle specifiche sono uniche, cioè non possono esistere due record con lo stesso valore nei campi chiave.

Nessun altro logico che comprenda tra le sue chiavi tutti i campi della chiave unica qui definita avrà necessità di dichiararsi a chiave unica perché non potrà, a causa del primo vincolo, trovare nel fisico valori che contraddicano la regola. Il vincolo di unicità, cioè, affligge il file fisico in maniera permanente e anche gli altri logici ne saranno beneficiari senza che questo comporti la ripetizione della clausola altrove. L'unicità delle chiavi è garantita dal motore del database AS/400.

Altra chiave, sempre presente sui fisici di tipo anagrafico dotati di codice e descrizione, sarà quella che porta la descrizione come primo campo. Seguirà sempre la ripetizione della chiave unica del primo file logico. Tale logico sarà utile nelle ricerche per descrizione.

```
LF  A.........T.Nome++++++.Lun++TPdB......Funzioni++++++++++++++++++++++++++++++
    A* File ANPER02L
    A          R ANPER
    A                                    PFILE(ANPER00F)
    A          K P1NMPE
    A          K P1CDPE
```

Anche questo file non escluderà alcunché.

7.6 Alterazione della sequenza di comparazione nelle chiavi.

Per evitare che maiuscole e minuscole incidano nelle chiavi su campi descrittivi, si chiede al motore del database di costruire le chiavi descrittive come se i caratteri della descrizione fossero tutti maiuscoli. L'esempio precedente diventa.

```
LF  A.........T.Nome++++++.Lun++TPdB......Funzioni++++++++++++++++++++++++++++++
    A* File ANPER02L
    A                                    ALTSEQ(QSYSTRNTBL)
    A          R ANPER
    A                                    PFILE(ANPER00F)
    A          K P1NMPE
    A          K P1CDPE                  NOALTSEQ
```

Dove ALTSEQ =Le chiavi vengono manipolate da una tabella che trasforma i caratteri.
QSYSTRNTBL =Tabella che rialza le minuscole a maiuscole, già presente nel sistema.
NOALTSEQ =Il campo corrente non subisce l'influenza della parola chiave ALTSEQ.

7.7 Omissioni e Selezioni.

Tramite il logico, è possibile permettere l'accesso ad un sottoinsieme di record del fisico.

Allo scopo, dopo i campi chiave, si definiscono le regole di selezione e di omissione.

```
LF  A.........T.Nome++++++.Lun++TPdB......Funzioni++++++++++++++++++++++++++++++
    A* File ANPER02L
    A          R ANPER
    A                                    PFILE(ANPER00F)
    A          K P1CDPE
    A          O P1ANNU                  CMP(EQ 'A')
    A          S P1CDPE                  RANGE('A  ' 'C99')
    A            P1DTNS                  CMP(GE 19500101)
    A          S P1CDPE                  RANGE('E  ' 'F99')
    A            P1DTNS                  CMP(LT 19500101)
    A          S P1CDPE                  RANGE('M  ' 'P99')
    A            P1DTNS                  RANGE(19700101 20001231)
```

Le specifiche seguenti eliminano dal logico i record annullati.

```
A          O P1ANNU                  CMP(EQ 'A')
```

Le successive selezionano i record con codice persona compreso tra A e C e anno nascita uguale o maggiore di 1950.

```
A          S P1CDPE                  RANGE('A  ' 'C99')
A            P1DTNS                  CMP(GE 19500101)
```

Le successive selezionano i record con codice persona compreso tra E e F e anno nascita minore di 1950.

```
A          S P1CDPE                  RANGE('E  ' 'F99')
A            P1DTNS                  CMP(LT 19500101)
```

Le successive selezionano i record con codice persona compreso tra M e P e anno nascita compreso tra 1970 e 2000.

```
A          S P1CDPE                  RANGE('M  ' 'P99')
A            P1DTNS                  RANGE(19700101 20001231)
```

Si nota che il record segue il destino di selezione od omissione della prima regola soddisfatta nell'ordine di elencazione.

Il compilatore assume che dopo l'ultimo gruppo:

- se di selezione, sia sottinteso un gruppo di omissione totale;
- se di omissione, sia sottinteso un gruppo di selezione totale.

Nell'ultimo esempio si ritenga perciò aggiunta l'istruzione.

```
LF  A.........T.Nome++++++.Lun++TPdB......Funzioni++++++++++++++++++++++++++++++
    A          O *ALL
```

7.8 Strategie di creazione dei logici.

Della necessità dei fisici è raro discutere perché anche l'analista più inesperto sa individuare i soggetti da anagraficizzare con discreta facilità, se la cava sui movimenti e, quasi sempre, indovina i documenti.

I logici provocano invece gli istinti più creativi del neofita apprendista stregone, che carica la macchina di doppioni, di chiavi uniche ridondanti, di chiavi a proseguire, di logici per stampa di liste o tabulati statistici, di omissioni esagerate, di multiformato e di join dovuti ad ignoranza di banali tecniche di lettura e ad improvvisazione a valle di una cattiva conoscenza del database da implementare.

Seguono alcune regole igieniche sulla creazione e modifica dei file logici, sulle chiavi da usare in un logico e sull'uso dei logici nei programmi.

Circa l'univocità delle chiavi.

- Creare in ogni file logico chiavi virtualmente uniche, sufficienti cioè all'ordinamento univoco del file. Qualunque chiave deve includere, almeno in coda, i campi di una chiave unica e senza esclusioni, già applicata al file precedentemente, ad esempio per la gestione. Il concetto di chiave virtualmente unica è perfettamente opposto a quello di ordinamento incerto.

Circa il riutilizzo dei logici.

- Riutilizzare nei programmi i logici già esistenti scegliendo la chiave più opportuna nella batteria di quelle già presenti sul fisico indagato. Usare, se disponibile, una utility che mostri sinteticamente le chiavi di tutti i file logici sovrastanti il fisico, ad esempio il Twf.

Circa la creazione dei logici mancanti.

- Creare il logico se il programma che lo chiede è un interattivo e non esiste altro modo di rispondere rapidamente alla richiesta formulata.
- Creare il logico se il programma che lo chiede è la stampa immediata di un singolo documento, data la chiave univoca del documento.
- Non creare il logico se la procedura che dovrebbe usarlo è un batch e la risposta urgente non è necessaria. Usare invece uno schema batch (42).
- Non creare il logico se uno già esistente ha la stessa chiave più altri campi chiave indifferenti in coda

Circa l'implementazione di logici esistenti.

- Modificare un logico preesistente se una preesistente chiave ad ordinamento incerto può trasformarsi in quella richiesta tramite la semplice aggiunta di campi chiave in coda alla chiave esistente. I programmi preesistenti possono usare il nuovo file senza essere modificati.

Circa la duplicazione di logici esistenti con altro nome.

- Il sistema riconosce automaticamente una chiave già esistente e non si sovraccarica di doppia gestione. Se necessario, quindi, via libera a qualche rara doppia definizione se necessaria in un programma che esegue doppia apertura per non sposizionare un ciclo di lettura con riletture dello stesso logico.

7.9 File logici monoformato.

Il file logico monoformato contiene l'indirizzamento ad un solo file fisico. Dei monoformato si è parlato sino ad ora e costituiscono la scelta più riutilizzabile e più generalizzata.

7.10 File logici multiformato.

Un file logico può indirizzare anche i dati di più file fisici. I multiformato sono file poco riutilizzabili e quindi da evitare.

Il loro uso più frequente è a semplificare i cicli di lettura da parte di programmi sbrigativi.

Profittandone per svolgere un esercizio di generazione di database, si fornisce l'esempio dei due fisici campionati qua sotto, testate e righe, legati dalla chiave di testata.

	Testate	
	Chiave testata	Importo totale
Record 1	A	150
Record 2	K	70
Record 3	M	200

	Righe		
	Chiave testata	Chiave riga	Importo riga
Record 1	A	10	70
Record 2	A	20	30
Record 3	A	30	50
Record 4	K	10	45
Record 5	K	20	25
Record 6	M	10	200

Il sorgente del dizionario.

```
PF  A..........T.Nome++++++RLun++TPdB......Funzioni++++++++++++++++++++++++++++++
     * Dizionario delle parole.
     *·  IM = Importo
     *·  KY = Chiave
     *   RI = Riga
     *·  TE = Testata
     *·  TO = Totale
     *
     * Dizionario dei campi.
    A          R DIZIOR
    A                                      TEXT('Dizionario.')
    A            KYTE           5
    A                                      COLHDG('Chiave' +
    A                                             'testata')
    A            KYRI           3P 0
    A                                      COLHDG('Chiave' +
    A                                             'riga')
    A            IMRI           9P 0
    A                                      COLHDG('Importo' +
    A                                             'riga')
    A                                      EDTCDE(K)
    A            IMTO           9P 0
    A                                      COLHDG('Importo' +
    A                                             'totale')
    A                                      EDTCDE(K)
```

Il sorgente del fisico testate documenti.

```
PF  A..........T.Nome++++++RLun++TPdB......Funzioni++++++++++++++++++++++++++++++
     * File DOCTE00F
    A                                      REF(DIZIO)
    A          R DOCTE
    A                                      TEXT('Testate.')
    A            TEKYTE    R             REFFLD(KYTE)
    A            TEIMTO    R             REFFLD(IMTO)
```

Il sorgente del fisico righe documenti.

```
PF  A..........T.Nome++++++RLun++TPdB......Funzioni++++++++++++++++++++++++++++++
     * File DOCRI00F
    A                                      REF(DIZIO)
    A          R DOCRI
    A                                      TEXT('Righe.')
    A            RIKYTE    R             REFFLD(KYTE)
    A            RIKYRI    R             REFFLD(KYRI)
    A            RIIMRI    R             REFFLD(IMRI)
```

Il sorgente del logico fondamentale testate.

```
LF  A..........T.Nome++++++.Lun++TPdB......Funzioni++++++++++++++++++++++++++++++
     A* File DOCTE01L
    A                                      UNIQUE
    A          R DOCTE
    A                                      PFILE(DOCTE00F)
    A          K TEKYTE
```

Il sorgente del logico fondamentale righe.

```
LF  A..........T.Nome++++++.Lun++TPdB......Funzioni++++++++++++++++++++++++++++++
     * File DOCRI01L
    A                                      UNIQUE
    A          R DOCRI
    A                                      PFILE(DOCRI00F)
    A          K RIKYTE
    A          K RIKYRI
```

Ed infine si espone il sorgente del logico multiformato.

```
LF  A.........T.Nome++++++.Lun++TPdB......Funzioni++++++++++++++++++++++++++++++
     * File DOCRI01C
A                                       UNIQUE
A           R DOCTE                     PFILE(TESTA00F)
A           K TEKYTE
A           K *NONE
A           R RIGHE                     PFILE(RIGHE00F)
A           K RIKYTE
A           K RIKYRI
```

Si noti che i record dei due file figurano ordinati nel logico multiformato come se il meno dotato in campi chiave fosse invece provvisto dei campi mancanti. In tali campi chiave, di caratteristiche uguali ai campi invece presenti negli altri tracciati in quella medesima posizione, si deve supporre la presenza del valore ***LOVAL** (vedi costanti figurative). Con i presupposti indicati, si può allora capire l'ordinamento assunto dal sistema: nel caso in esempio, percorrendo il logico dall'inizio verso la fine, si trovano in ordine ascendente tutti i record della prima chiave di testata, prima la testata (come se avesse chiave riga *loval) e poi le righe corrispondenti in ordine di chiave riga; segue la seconda testata e poi le sue righe, eccetera.

L'esempio illustrato non verrebbe meno se nella testata ci fosse un campo ragionevolmente in grado di tenere la parte del campo supposto mancante.

Tramite il multiformato, i dati dell'esempio si vedrebbero nella forma e nell'ordine seguenti.

Record 1	Testata	A	150	
Record 2	Riga	A	10	70
Record 3	Riga	A	20	30
Record 4	Riga	A	30	50
Record 5	Testata	K	70	
Record 6	Riga	K	10	45
Record 7	Riga	K	20	25
Record 8	Testata	M	200	
Record 9	Riga	M	10	200

7.11 File join.

Il tracciato di un file join semplice risulta composto dalla giustapposizione di un tracciato principale e di un tracciato secondario agganciato tramite una chiave presente sul principale. La chiave permette il recupero univoco del record secondario. Si diffida dall'aggancio non univoco perché moltiplica la vista dei record primari in funzione dell'eccesso di corrispondenze.

In buona sostanza un file join evita al programma applicativo RPG di recuperare in proprio dei record tramite operazioni di CHAIN perché tali record vengono forniti di corredo alla lettura del record principale, che ne porta le chiavi complete.

L'uso illustrato nell'esempio permette di leggere una riga ed avere di corredo la testata corrispondente.

Un altro uso potrebbe essere la costruzione di un record composito che si trascina appresso tutte le decodifiche di cui porta le chiavi. Attenzione però: i campi base dell'aggancio possono risiedere soltanto sul record principale e questo limita molto la flessibilità dello strumento.

Un uso massiccio di questo tipo di file costringerebbe comunque il sistema ad una overdose di ricerche di database a fronte di semplici letture che non sempre applicativamente necessitano di decodifiche complete.

Il seguente esempio fa uso degli stessi file fisici del paragrafo precedente sul multiformato.

```
LF  A.........T.Nome++++++.Lun++TPdB......Funzioni++++++++++++++++++++++++++++++
     * File DOCRI01J
     * Richiede di assumere i valori di difetto per i campi vuoti
    A                                     JDFTVAL
     * Nome Tracciato.
    A          R DOCRI
     * Elenco dei file da congiungere.
    A                                     JFILE(DOCRI00F DOCTE00F)
     * Dettaglio di una congiunzione file congiunti - chiavi di congiunzione
    A          J                          JOIN(DOCRI00F DOCTE00F)
    A                                     JFLD(RIKYTE TEKYTE)
     * Elenco campi dalle Righe.
    A            RIKYTE
    A            RIKYRI
    A            RIIMRI
     * Elenco campi dalle Testate.
    A            TEKYTE
    A            TEIMTO
     * Elenco campi chiave del tracciato Righe.
    A          K RIKYTE
    A          K RIKYRI
```

Tramite il join, i dati dell'esempio si vedrebbero nella forma e nell'ordine seguente.

	Campi dalla riga			Campi dalla testata	
	Chiave testata	Chiave riga	Importo riga	Chiave testata	Importo totale
Record 1	A	10	70	A	150
Record 2	A	20	30	A	150
Record 3	A	30	50	A	150
Record 4	K	10	45	K	70
Record 5	K	20	25	K	70
Record 6	M	10	200	M	200

8 Query. (08R)

Il Query è uno strumento molto efficace per produrre delle stampe, effettuare delle estrazioni di dati ed effettuare delle analisi sui dati presenti nel database.
Il motore che il Query utilizza per effettuare le elaborazioni è lo stesso dell'SQL, soltanto meno potente e limitato unicamente alla lettura di un database preesistente (non è possibile effettuare delle operazioni di aggiornamento o cancellazione ai dati.

8.1 Definizione interattiva.

Per accedere alle funzioni del Query il sistema mette a disposizione un menù che si ottiene digitando il comando **STRQRY**.
Queste sono le opzioni del menù del Query.

```
Query per AS/400
1. Gestione query
2. Esecuzione di un query esistente
3. Cancellazione di un query
```

Le tre opzioni del menù corrispondono ai seguenti tre comandi.

WRKQRY
RUNQRY QRY(NomeLib/NomeQuery) QRYFILE((NomeLib/NomeFile *FIRST))
DLTQRY QRY(NomeLib/NomeQuery)

Mentre i comandi di esecuzione e di cancellazione di un Query risultano essere intuitivi, soffermiamoci invece sul comando WRKQRY.
Il comando WRKQRY non prevede parametri e questo è il menù a cui si accede dal quale è possibile operare con i Query.

```
Immettere le scelte e premere Invio.
 Opzione  . . _          1=Creaz.  2=Modifica  3=Copia  4=Cancel.
                         5=Visualizzazione  6=Stampa definizione
                         8=Esecuzione in batch  9=Esecuzione
 Query  . . . NOMEQRY    Nome, F4 (Elenco)
 Libreria . . NOMELIB    Nome, *LIBL, F4 (Elenco)
```

Anche in questo caso le opzioni 2…9 sono di elementare comprensione.
Analizziamo piuttosto l'opzione 1=Creazione che permette l'accesso al video con le opzioni per la creazione interattiva di un Query. Si noti che anche per l'opzione 2=Modifica viene presentato lo stesso video.

```
Immettere le opzioni e premere Invio. Premere F21 per sceglierle tutte.
1=Scelta
 Opz    Opzione di definizione query
  1     Specifica delle scelte dei file
  _     Definizione dei campi dei risultati
  _     Scelta e sequenza dei campi
  _     Scelta dei record
  _     Scelta dei campi di ordinamento
  _     Scelta della sequenza di ordinamento
  _     Specifica della formattazione delle colonne del prospetto
  _     Scelta delle funzioni di riepilogo del prospetto
  _     Definizione delle separazioni del prospetto
  _     Scelta del tipo e del formato di emissione
  _     Specifica delle opzioni di elaborazione
```

8.2 Dettagli della definizione interattiva.

Queste sono tutte le opzioni possibili per la creazione e/o modifica di un Query.

Specifica delle scelte dei file.

Questa opzione si utilizza per specificare (dichiarare) i file di database che si andranno ad utilizzare nel Query. Si deve specificare almeno un file di database, ma se ne possono specificare quanti necessari all'elaborazione. Solo nel caso in cui vengano specificati più file di database, si dovranno specificare anche i criteri di unione tra i file ed i campi di ciascun file mediante i quali avvengono gli agganci tra di loro.
Per prima cosa chiariamo il concetto di campi mediante i quali avvengono gli agganci tra i diversi file; sono concettualmente riconducibili ai campi chiave che costituiscono una chiave di accesso

ad un file nell'ambito di un programma RPG. Un record in un file con chiavi viene reperito con una operazione CHAIN solo se tutti i campi della chiave trovano corrispondenza in almeno un record nel file. L'aggancio tra i record dei file specificati nel Query avviene esattamente allo stesso modo.
Vediamo ora di chiarire meglio il concetto di criterio di unione tra i file; è possibile scegliere tra tre diversi tipi di unione tra i file.

```
Tipo di unione . . . . 1     1=Record corrispondenti
                             2=Record corrispondenti con file principale
                             3=Record non corrispondenti con file principale
```

Il primo criterio è quello per il quale verranno selezionati solo i record che trovano corrispondenza in tutti i file specificati. Ad esempio se si sono specificati due file (anagrafica clienti ed anagrafica fornitori) e si è deciso di unirli per il campo Provincia, l'esecuzione del Query selezionerà e presenterà solamente quei clienti e quei fornitori che hanno la medesima provincia. Tutti gli altri clienti e fornitori verranno scartati.
Il secondo criterio, che è quello che ci sentiamo di suggerirvi di utilizzare sempre, è quello per il quale il primo file tra quelli specificati viene considerato il file primario e tutti gli altri dei file secondari. In questo modo, se non specificate selezioni od omissioni di record per il file primario, tutti i record dello stesso saranno selezionati e presentati, mentre solo i record degli altri file che troveranno le opportune corrispondenze saranno selezionati e presentati (agganciati) ai record del file primario.
Il terzo criterio, difficilmente comprensibile e del quale sconsigliamo vivamente l'utilizzo, effettua un ragionamento inverso rispetto a quello che comunemente viene fatto dalle nostre menti per quanto attiene al concetto di unione tra record per i file di database. Se scelto quest'ultimo criterio il Query selezionerà e presenterà tutti i record che NON trovano alcuna corrispondenza tra i file specificati. Nel nostro esempio delle anagrafiche clienti e fornitori unite per il campo Provincia, l'esito del Query sarà quello di presentarci un elenco di tutti i clienti e fornitori che hanno una provincia che non viene trovata nell'altro anagrafico.

Definizione dei campi dei risultati.

In questa sezione si possono definire dei campi di lavoro che sono frutto dell'elaborazione del Query stesso. Ad esempio la moltiplicazione di due campi numerici di un record, l'estrazione di una sottostringa alfanumerica da un campo di un record di database etc.
Per definire un campo di questo tipo è sufficiente dichiararne il nome, a completa discrezione del programmatore, e l'operazione che deve essere eseguita per la sua costituzione (es. CampoA + CampoB).

```
Campo       Espressione                       Intestazione colonna  Lun   Dec
__________  ________________________________  ____________________  _____ ___
            ________________________________  ____________________
            ________________________________  ____________________
```

Scelta e sequenza dei campi.

In questa sezione si selezionano i campi che si desidera costituiscano il risultato finale dell'elaborazione, sia esso una stampa od un file di database; è la struttura del record del risultato del Query. In prima istanza vengono presentati tutti i campi di tutti i file specificati. A questo punto è a carico del programmatore selezionare i campi di interesse e specificarne la sequenza di ordinamento.

Scelta dei record.

In questa sezione si specificano le condizioni di selezione ed omissione dei record.

```
Imm. i confronti, premere Invio. Specificare OR per avviare ogni nuovo gruppo
   Test:  EQ, NE, LE, GE, LT, GT, RANGE, LIST, LIKE, IS, ISNOT...

AND/OR   Campo           Test   Valore (campo, Numero, 'caratteri', o ...)
```

Scelta dei campi di ordinamento.

In questa sezione si specificano i campi con i quali il Query dovrà effettuare l'ordinamento nel momento della presentazione dei risultati dell'elaborazione. Si possono specificare i campi per i quali si intende ottenere l'ordinamento ed anche il tipo di ordinamento richiesto; ascendente o discendente.

Scelta della sequenza di ordinamento.

In questa sezione si può specificare il tipo di sequenza con la quale effettuare l'ordinamento dei dati. Il default è Esadecimale ed è quella che abitualmente si utilizza. Vi sono altre opzioni precostituite fino ad arrivare ad una opzione che consente di crearsi una propria tabella di sequenza di caratteri con i quali effettuare l'ordinamento dei dati. Utilizzate il default!

Specifica della formattazione delle colonne del prospetto.

In questa sezione si possono specificare le intestazioni di colonna per i campi che costituiscono il prospetto Query ed eventualmente modificare le dimensioni dei campi di output. Si possono inoltre specificare il numero di spazi tra un campo e l'altro ed anche il modo con cui i campi numerici debbono essere editati.

Scelta delle funzioni di riepilogo del prospetto.

In questa sezione si possono specificare le eventuali funzioni di riepilogo del prospetto Query; queste le funzioni messe a disposizione.

```
Immettere le opzioni e premere Invio.
     _   1=Totale  2=Media     3=Minimo    4=Massimo   5=Conteggio
```

Dove:
- 1=Totale Esegue la totalizzazione (somma) dei valori presenti in un campo numerico.
- 2=Media Calcola la media algebrica dei valori presenti in un campo numerico.
- 3=Minimo Evidenzia il valor minimo tra i valori elaborati per un campo.
- 4=Massimo Evidenzia il valor massimo tra i valori elaborati per un campo.
- 5=Conteggio Calcola il numero di record elaborati.

Definizione delle separazioni del prospetto.

In questa sezione si possono specificare le eventuali rotture di livello per il Query. Affinché avvengano correttamente le rotture di livello i campi con cui si definiscono le rotture di livello devono essere gli stessi con i quali è stato effettuato l'ordinamento dei dati. Per ciascuna rottura di livello si può specificare un testo di descrizione ed eventualmente di visualizzare il contenuto delle relative variabili, se si desidera che in un prospetto di stampa si effettui un cambio pagina etc.

Questo è quanto viene mostrato nella sezione.

```
Immettere il livello di separazione (1-6) per massimo 9 nomi di campi e premere
Invio. Usare tanti campi quanti sono necessari per ogni livello di separazione.
```

Scelta del tipo e del formato di emissione.

In questa sezione si può specificare il tipo ed il formato di emissione per il Query.

```
 Immettere le scelte e premere Invio.
   Tipo di emissione  . . . . . .   1     1=Video
                                          2=Stampante
                                          3=File di database

   Formato di emissione . . . . . .   1     1=In dettaglio
                                          2=Solo riepilogo

   Suddivisione e incolonnamento righe  N     Y=Sì   N=No
     Largh. suddivis. e incolonnamento        Spazio, 1-378
     Record su una pagina . . . . . .   N     Y=Sì   N=No
```

Tipo di emissione.

- 1=Video L'emissione del Query è richiesta a video.

- 2=Stampante L'emissione del Query è richiesta su di un file di stampa: si potranno specificare varie opzioni sul file di stampa ricevente l'emissione e comporre l'intestazione di tabulato.
- 3=File di database L'emissione del Query è richiesta su di un file di database; si dovranno specificare le informazioni relative al nome ed alla libreria del file di database ricevente l'emissione, l'opzione con la quale il Query effettuerà l'aggiornamento dei dati, il testo del file e quant'altro necessario.

Formato di emissione.

- 1=In dettaglio Tutti i record selezionati dal Query faranno parte della emissione.
- 2=Solo riepilogo Faranno parte della emissione del Query solo i record di totale o riepilogo. Questo vale solo se sono state definite delle rotture di livello.

Specifica delle opzioni di elaborazione.

In questa sezione si possono specificare alcune opzioni relative all'elaborazione del Query.

```
Immettere le scelte e premere Invio.
 Arrotondamento . . . . . .          Spazio in bianco, Y=Sì, N=No

 Ignorare errori
   dati decimali  . . . . . .        Spazio in bianco, Y=Sì, N=No

 Ignorare avvertenze
   sostituzione caratteri . . . . .   Y      Y=Sì   N=No

 Usare sequenza di ordinamento per
   tutti i confronti tra caratteri    Y      Y=Sì   N=No
```

8.3 Generazione di stampe veloci.

Uno dei tipici impieghi del Query è quello di utilizzarlo per produrre dei prospetti in stampa. Siccome è molto semplice realizzare delle stampe con questo strumento, lo si utilizza per confezionare delle versioni estemporanee anche di stampe applicative. Per esempio, nella fase di realizzazione di una procedura statistica, una volta terminata la fase di sviluppo e verifica dei programmi di estrazione dati su archivi di lavoro, si utilizzerà dapprima un Query per la stampa dei dati estratti e successivamente si procederà alla realizzazione di un programma di stampa vero e proprio sostituendo la versione provvisoria precedentemente realizzata con il Query.

8.4 Scarico su disco per ulteriori elaborazioni.

Come visto in precedenza il Query può essere anche utilizzato per generare file di database. Utilizzando lo scarico su disco di file estratti con il Query si possono realizzare delle mini procedure di estrazione, completamento e stampa dei dati. Potendo anche specificare il formato di emissione si può decidere di ottenere dei dati già riepilogati a livello di totale ai quali sarà poi sufficiente aggiungere le opportune decodifiche per realizzare il prospetto finale.

8.5 Utilizzo per analisi sui dati: verifiche di contenuto dei file; popolazione di un codice,…

Un altro impiego del Query è quello per effettuare delle analisi sui dati. Essendo uno strumento facile ed efficace consente al programmatore di potere verificare il contenuto dei file di database, censire e quantificare la presenza di valori nei campi. A differenza di altri strumenti che consentono la gestione per singolo record e su un singolo file di database, il Query può effettuare elaborazioni per tutti i record di uno o più file di database. Si può affermare senza timore di essere smentiti che il Query è diventato lo strumento per eccellenza per effettuare le analisi sui dati.

9 Data File Utility (DFU). (09C)

Il Dfu è un programma di utilità del software di base che permette la manipolazione dei dati di un file.

9.1 Dfu di default.

Il modo più facile di usare il Dfu è chiamarlo con l'opzione 18 su un elenco Pdm di file fisici e logici. Tale opzione chiama il comando

```
UPDDTA      FILE(Libreria/File)
```

ad esempio

```
UPDDTA      FILE(NERONI1/PERSO00F)
```

che crea un Dfu eseguibile estemporaneo e lo esegue immediatamente.

Alla fine dell'esecuzione il Dfu estemporaneo viene cancellato.

9.2 Dfu di default su un file senza chiavi.

Il primo video, presentato quando si esamina un file fisico senza chiavi, è simile al seguente. Chiameremo tale video Guida.

Video Guida RRN

```
WORK WITH DATA IN A FILE                        Mode . . . . :   CHANGE
Format . . . . :   PERSO_____                   File . . . . :   PERSO00F

*RECNBR:             _________


F3=Exit                 F5=Refresh              F6=Select format
F9=Insert               F10=Entry               F11=Change
                                      (C) COPYRIGHT IBM CORP. 1980, 2002.
```

Tramite esso è possibile visualizzare il video Dati di un certo record fornendone il Numero Relativo di Record (Relative Record Number) e poi premendo Invio.

```
WORK WITH DATA IN A FILE                        Mode . . . . :   CHANGE
Format . . . . :   PERSO_____                   File . . . . :   PERSO00F

*RECNBR:             5________


F3=Exit                 F5=Refresh              F6=Select format
F9=Insert               F10=Entry               F11=Change
                                      (C) COPYRIGHT IBM CORP. 1980, 2002.
```

Partendo dalla Guida è possibile anche rollare (battere i tasti di Roll) per accedere al primo record dell'archivio (RollUp) o all'ultimo (RollDown).

Video Dati RRN

```
WORK WITH DATA IN A FILE                        Mode . . . . :   CHANGE
Format . . . . :   PERSO_____                   File . . . . :   PERSO00F

*RECNBR:                    5
Persona:             B01
Cognome:             Brambilla___________
Nome:                Mario_______________
Data nascita:        19800815_
Provincia nascita:   CO_
Provincia residenza: MI_


F3=Exit                 F5=Refresh              F6=Select format
F9=Insert               F10=Entry               F11=Change
```

Sul video Dati è possibile modificare i dati che all'Invio o al Roll vengono riportati sul record di database.

Se un solo video dati non basta a contenere tutti i dati, il Dfu ne crea a sufficienza. Per accedere ai video dati supplementari, occorre premere invio. Per rivedere il video dati precedente, si preme F12.

Se il record contiene molti campi, il dfu genera più di una videata. Si osserva quindi che, se il video Dati non è l'ultimo del record, l'invio presenta il video Dati successivo mentre, se corre l'ultimo, l'invio provoca l'aggiornamento del database e poi presenta il video Guida.

Roll su Guida in bianco visualizza il record precedente o successivo all'ultimo visto.

9.3 Dfu di default su un file con chiavi.

Nel caso di utilizzo del Dfu per manipolare un file con chiavi, viene invece presentato un video guida contenente tutte le chiavi del file. Ad esempio

```
UPDDTA      FILE(NERONI1/PERSO01L)
```

Video Guida Chiavi

```
 WORK WITH DATA IN A FILE                    Mode . . . . :   CHANGE
 Format . . . . :   PERSO_____               File . . . . :   PERSO01L

 Persona:             ___


F3=Exit                F5=Refresh              F6=Select format
 F9=Insert              F10=Entry               F11=Change
                                      (C) COPYRIGHT IBM CORP. 1980, 2002.
```

Per accedere al video Dati è sufficiente fornire la chiave e premere Invio.

```
 WORK WITH DATA IN A FILE                    Mode . . . . :   CHANGE
 Format . . . . :   PERSO_____               File . . . . :   PERSO01L

 Persona:             B01


F3=Exit                F5=Refresh              F6=Select format
 F9=Insert              F10=Entry               F11=Change
                                      (C) COPYRIGHT IBM CORP. 1980, 2002.
```

Anche i Roll hanno conseguenze simili al caso RRN. RollUp presenta il record con chiave uguale o successiva a quella digitata sulla Guida. RollDown presenta il record uguale o precedente alla chiave digitata sulla Guida.

Roll su Guida in bianco visualizza il record precedente o successivo all'ultimo visto.

Video Dati Chiavi

```
 WORK WITH DATA IN A FILE                    Mode . . . . :   CHANGE
 Format . . . . :   PERSO_____               File . . . . :   PERSO01L

 Persona:             B01
 Cognome:             Brambilla__________
 Nome:                Mario______________
 Data nascita:        19800815_
 Provincia nascita:   CO_
 Provincia residenza: MI_


 F3=Exit                F5=Refresh              F6=Select format
 F9=Insert              F10=Entry               F11=Change
```

Il video Dati di un file con chiavi differisce da quello per RRN solo per la mancanza della visualizzazione del Relative Record Number e per l'evidenziazione dei campi chiave.

9.4 Controlli eseguiti dal Dfu di default.

I campi del video dati vengono controllati solo se modificati e solo con riguardo ai controlli richiesti sulle DDS del file in corso di manipolazione.

Il Dfu rispetta naturalmente anche i vincoli di unicità delle chiavi imposti dal database ed eventuali trigger applicati al file, né potrebbe fare altrimenti.

Il Dfu evita inoltre che l'aggiornamento avvenga alla cieca. Cioè, se il record è visualizzato contemporaneamente da due lavori diversi ed entrambi tentano l'aggiornamento, vince il primo mentre il secondo lavoro riceve un rifiuto.

9.5 Dfu compilato.

È possibile, mediante una procedura interattiva originata dal comando STRDFU (Start Data File Utility), generare un programma Dfu permanente compilato e rieseguibile a piacere tramite un altro comando nel quale si specifica il programma Dfu da eseguire.

```
CHGDTA      DFUPGM(Libreria/Programma)
```

Il Dfu si esegue sullo stesso file menzionato durante la definizione interattiva, a meno che non si specifichi diversamente.

```
CHGDTA      DFUPGM(Libreria/Programma)  FILE(LibreriaFile/File)
```

Essendo particolarmente raro l'uso del Dfu definito interattivamente, se ne rimanda la trattazione completa ad altra occasione.

9.6 Alternative al Dfu.

Esistono in commercio prodotti alternativi più potenti del Dfu, ma che non possono competere per disponibilità con il Dfu. Procurarseli richiede tempo, fatica e, se necessario soldi. Sono tuttavia utilissimi per migliorare drasticamente l'approccio al database dell'AS/400.

Se ne consiglia l'uso per superare la povertà del Dfu.

Tra questi si ricordano TWF e WRKDBF.

TWF

TWF (Tool that Works on Files) di Luigi Rizzi & Patrizio Perego, ora gestito e venduto da Datasys. Non ha subito nessun aggiornamento significativo dal 1995 circa ma è ancora molto attuale grazie alla lungimiranza dei primi autori.

WRKDBF

WRKDBF (Work Data Base File) del canadese Bill Reger, più recente e più curato ma molto meno potente e molto macchinoso. Ha il grande pregio di essere reperibile gratuitamente su Internet all'indirizzo www.wrkdbf.com.

Da quando l'autore ha rinunciato a gestirlo (grande esempio di suicidio mediatico) è scaricabile, contro il suo parere, da www.iseriestools.com.

10 Control Language (CL) nei programmi di controllo azione. (10R)

Come si può evincere dal nome, questo è il linguaggio di programmazione principale del sistema AS/400 con numerosissime istruzioni ed una sintassi propria particolare. Usa infatti dei codici operativi, o per meglio dire, dei comandi di sistema validi unicamente in questo contesto (riservati).

Il Control Language è il primo linguaggio che si usa per dialogare con la macchina, come DOS o Windows sui personal. Solitamente il suo uso è poco visibile all'utente che accede a menù rigidi ed è poco usato anche dal programmatore interattivo, anche se i programmi generalizzati di accesso che costoro utilizzano ne fanno largo uso.

Il CL ha un uso prevalentemente sistemistico ma serve anche per governare l'esecuzione di programmi applicativi prevalentemente di tipo batch. Può elaborare sequenzialmente uno e soltanto un file database. Può anche gestire una interfaccia video senza subfile ma quest'ultima possibilità è veramente poco usata.

Dai sorgenti di tipo Clp (Control Language Program) o Clle (Control Language integrated Language Environment), si generano gli oggetti di tipo *PGM eseguibili. I comandi per la creazione dei programmi Clp e Clle sono i seguenti.

```
CRTCLPGM    PGM(NomeLib/NomePgm) SRCFILE(NomeLib/QCLSRC) SRCMBR(*PGM)
            TEXT(*SRCMBRTXT)
CRTBNDCL    PGM(NomeLib/NomePgm) SRCFILE(NomeLib/QCLSRC) SRCMBR(*PGM)
            TEXT(*SRCMBRTXT)
```

Gli stessi comandi valgono per entrambe le tipologie salvo pochissime eccezioni di cui si ricorda il TFRCTL (Transfer Control), utilizzabile solo nel Clp. La tipologia Clle si presta ad assiemaggi di moduli compilati che utilizzano l'inutile link dell'ambiente Ile (Integrated Language Environment). Di tale argomento non si tratta in questo corso. Si raccomanda tuttavia la tipologia Clle perché debuggabile contemporaneamente all' RpgIle. Di qui in avanti non si distinguerà più tra le due tipologie.

La prima istruzione di un programma CL deve sempre essere PGM, mentre l'ultima deve sempre essere ENDPGM.

10.1 Parametri e variabili.

Definizione di parametro: porzione di memoria condivisa tra diversi programmi.
Il programma CL può essere richiamato passandogli dei parametri, in questo caso, l'istruzione PGM dovrà essere corredata dall'indicazione di tutti i parametri ricevuti.
Esempio di programma CL che riceve il parametro architetturale KPJBA.

```
PGM         PARM(&KPJBA)
```

Come si può notare, il nome del parametro è preceduto dal carattere &. Questo carattere identifica il nome di una variabile definita in un programma CL. Infatti oltre a dichiarare che il programma CL riceve il parametro KPJBA, si dovrà procedere alla dichiarazione(definizione) dello stesso. Esempio di dichiarazione della variabile KPJBA

```
DCL         VAR(&KPJBA) TYPE(*CHAR) LEN(502)
```

In questo modo si è definita una variabile di tipo carattere (alfanumerica) di lunghezza 502 byte. Sono previste tre tipologie di variabili. Quella carattere appena descritta, quella decimale (numerica) e quella logica.

```
DCL         VAR(&VALORE) TYPE(*DEC) LEN(15 2)
```

In questo modo si è definita una variabile di tipo decimale (numerico) di lunghezza 15 byte di cui 2 decimali.

```
DCL         VAR(&SINO) TYPE(*LGL)
```

In questo modo si è definita una variabile di tipo logico. La lunghezza non necessita di essere specificata poiché può essere unicamente lunga 1 byte, e può assumere solo i valori "0" ed "1".

10.2 Operazioni con le variabili.

Nell'esempio che segue, si procede alla definizione di tre variabili. La prima è la variabile architetturale KPJBA, la seconda è la variabile architetturale parte utente KPJBU, e la terza è una variabile per verificare se l'utente ha richiesto la stampa. Quest'ultimo parametro si presume che sia stato immesso dal programma di lancio della funzione batch.

Esempio di dichiarazione della variabile KPJBA

```
DCL        VAR(&KPJBA) TYPE(*CHAR) LEN(502)
```

Esempio di dichiarazione della variabile KPJBU

```
DCL        VAR(&KPJBU) TYPE(*CHAR) LEN(256)
```

Esempio di dichiarazione della variabile PPSTA

```
DCL        VAR(&PPSTA) TYPE(*CHAR) LEN(1)
```

Le operazioni che seguono sono le operazioni necessarie per ricavare dapprima la variabile architetturale parte utente e quindi la variabile di richiesta stampa da parte dell'utente.

Ricava variabile architetturale parte utente

```
CHGVAR     VAR(&KPJBU) VALUE(%SST(&KPJBA 247 256))
```

In questo caso si ricava la variabile KPJBU estraendola dalla variabile KPJBA dalla posizione 247 per 256 byte.

Ricava variabile di eventuale richiesta stampa da parte dell'utente

```
CHGVAR     VAR(&PPSTA) VALUE(%SST(&KPJBU 15 1))
```

In questo caso si ricava la variabile PPSTA estraendola dalla variabile KPJBU dalla posizione 15 per 1 byte. A questo punto, dopo avere reperito le variabili, il programma deve stabilire se l'utente sul video di lancio ha richiesto o meno che venga eseguita la stampa. Per verificare una condizione con variabili CL si procede come segue:

```
IF         COND(&PPSTA *EQ 'S') THEN(DO)
CALL       PGM(*LIBL/NomePgm) PARM(&KPJBA)
ENDDO
```

In questo caso, se è stata richiesta la stampa, verrà richiamato il programma opportuno al quale viene passato come parametro la variabile architetturale KPJBA.

10.3 Chiamate.

Come già sommariamente visto al paragrafo precedente, si può richiamare l'esecuzione di uno o più programmi, siano essi RPG o CL. Il comando per richiamare l'esecuzione di un altro programma è CALL. Se occorre passare dei parametri al programma chiamato, gli stessi si dovranno specificare sul comando CALL. Tipico impiego delle chiamate a diversi programmi è quello per i programmi di stampa che richiedono una estrazione di dati in file di lavoro. Il primo programma richiamato sarà un estrattore di dati, il secondo un programma di completamento dati precedentemente estratti ed il terzo programma di stampa.

Esempio di chiamata al programma NOMEPGM con passaggio del parametro architetturale KPJBA.

```
CALL       PGM(NOMEPGM) PARM(&KPJBA)
```

Ricordarsi sempre che un programma, sia esso CL o RPG, non deve mai richiamare se stesso né direttamente né tramite un altro programma poiché si incorrerebbe in una chiamata ricorsiva, non gestita dal software di base.

10.4 Reindirizzamenti.

A volte si rende necessario specificare esattamente il/i file di database da utilizzare da parte del programma che viene richiamato. Per fare ciò il sistema mette a disposizione un comando specifico nel quale si può dare questa indicazione. Il comando è OVRDBF. Nel caso che di seguito viene illustrato si presuppone che il file di database ANCLI00F sia il file dei clienti relativo ad un sistema informativo, ma per questo lavoro specifico si ha la necessità di utilizzare il file dei clienti di un altro sistema informativo la cui libreria dati non è al momento

nella lista librerie del job corrente. Per potere utilizzare il file clienti che non è in lista librerie e puntare direttamente ad esso, si dovrà reindirizzare l'oggetto tramite il seguente comando.

```
OVRDBF      FILE(ANCLI00F)  TOFILE(NomeLib2/ANCLI00F)
```

In questo caso si specifica che il programma che verrà successivamente richiamato dovrà utilizzare il file ANCLI00F che si trova nella libreria NomeLib2.

10.5 Utilizzo e riempimento della libreria temporanea.

Ad ogni job viene associata una libreria temporanea **QTEMP**. Questa libreria è legata univocamente a ciascun job che nasce sul sistema. Questa libreria dovrebbe essere specificata come prima libreria nella lista librerie parte utente. Per potere verificare se la libreria temporanea QTEMP è la prima della lista librerie parte utente, si può utilizzare il comando

```
DSPSYSVAL   SYSVAL(QUSRLIBL)
```

Il fatto di avere una libreria che nasce e muore nell'ambito di un singolo job, ha fatto in modo che le tecniche di programmazione ne tenessero conto. Infatti, specialmente per le procedure che utilizzano un programma di controllo CL, nel caso si debba procedere ad una estrazione di dati su archivi di lavoro i quali vengono utilizzati unicamente nel job che li genera, si usa effettuare una duplicazione degli oggetti necessari alla elaborazione nella libreria temporanea QTEMP. In questo modo l'occupazione di spazio sul sistema si limita al tempo di elaborazione della funzione poiché, come già detto, al termine dell'esecuzione del job la libreria QTEMP e tutto il suo contenuto andrò perso. Di seguito viene descritto come un programma CL può utilizzare la libreria temporanea.

Pulizia preliminare (precauzionale) degli oggetti nella QTEMP.

```
DLTF        FILE(QTEMP/ANCLI*)
```

Duplicazione degli oggetti necessari all'esecuzione nella QTEMP.

```
CRTDUPOBJ   OBJ(ANCLI*) FROMLIB(NomeLib)  OBJTYPE(*FILE)  TOLIB(QTEMP)
            NEWOBJ(*OBJ)  DATA(*NO)
```

Reindirizzamento archivi di lavoro.

```
OVRDBF      FILE(ANCL01L)  TOFILE(QTEMP/ANCLI01L)
```

Richiamo del programma di elaborazione dei dati, ad esempio un RPG.

```
CALL        PGM(NomePgm) PARM(&KPJBA)
```

Se non si è sicuri che la QTEMP sia la prima libreria in lista librerie ci si può difendere in questo modo: si procede alla sua rimozione dalla lista librerie, quindi si procede al suo reinserimento specificando la collocazione come prima libreria.

```
RMVLIBLE    LIB(QTEMP)
MONMSG      MSGID(CPF0000)
ADDLIBLE    LIB(QTEMP)  POSITION(*FIRST)
```

10.6 Intercettazione e gestione degli errori.

Nell'ambito di un programma CL è possibile intercettare e gestire degli errori che occorrono durante l'esecuzione del programma. Vi sono diversi gruppi/tipologie di errori che si possono intercettare, i più diffusi e di comune utilizzo sono i CPFxxxx, dove per xxxx si intende il numero di messaggio di sistema. Questi messaggi sono contenuti in un file messaggi MSGF specifico di sistema, nel caso dei messaggi CPFxxxx gli stessi sono contenuti nel file messaggi QCPFMSG. Nel caso in cui non si conosca l'esatto codice di errore, si può utilizzarne uno generico; CPF0000. Vale a dire che tutti i messaggi di tipo CPF vengono intercettati e monitorizzati. Per chiarire meglio il concetto, supponiamo di dovere verificare la presenza di un oggetto necessario all'elaborazione ad esempio nella libreria temporanea QTEMP. In questo caso si vuole verificare la presenza del file di database ANCLI00F nella libreria QTEMP

```
CHKOBJ      OBJ(QTEMP/ANCLI00F)  OBJTYPE(*FILE)
```

Nel caso in cui l'oggetto non sia presente, il sistema risponderà con un opportuno messaggio di errore CPFxxxx. Per evitare che il programma, nel caso in cui il comando CHKOBJ non trovi

l'oggetto ANCLI00F nella libreria QTEMP vada in errore, è possibile intercettare e monitorizzare tale messaggio di errore specificando, immediatamente dopo il comando CHKOBJ il comando

```
MONMSG      MSGID(CPF0000)
```

Così facendo il programma non andrà in errore e continuerà l'esecuzione. Si può utilizzare l'intercettazione degli errori anche per potere intraprendere delle opportune contromisure. Nell'esempio precedente si verificava l'esistenza del file di database ANCLI00F nella libreria QTEMP. Ma cosa fare se il file di database, necessario all'esecuzione, non si trovasse nella libreria QTEMP? Ebbene, si può utilizzare l'intercettazione dell'errore e gestire opportunamente la situazione.

```
MONMSG      MSGID(CPF0000)  EXEC(DO)
CRTDUPOBJ   OBJ(ANCLI00F)  FROMLIB(NOMELIB)  OBJTYPE(*FILE)
            TOLIB(QTEMP)  NEWOBJ(*OBJ)  DATA(*NO)
ENDDO
```

In questo caso oltre che ad avere intercettato l'errore si è deciso di procedere alla creazione del file di database nella libreria temporanea QTEMP per il successivo utilizzo.

10.7 Utilizzo di un file di database.

Come già accennato in precedenza in un programma CL si può utilizzare uno e soltanto un file di database. Il file di database può essere letto una sola volta ed in modo sequenziale. Innanzitutto si deve dichiarare il file di database:

```
DCLF        FILE(ANCLI00F)
```

Con questa operazione il programma, all'atto della compilazione e quindi della creazione dell'oggetto eseguibile, conosce tutte le caratteristiche del file dichiarato, e mette a disposizione del programma CL tante variabili quanti sono i campi nel file di database senza la necessità di dichiararle una ad una; semplicemente si intendo sottintese. Di seguito viene illustrato il loop di lettura di un file di database nell'ambito di un programma CL.

```
READBEG:
      RCVF
      MONMSG      MSGID(CPF0000)  EXEC(DO)
           GOTO   CMDLBL(READEND))
      ENDDO
      /*  Elaborazione del record letto.  */
      GOTO        CMDLBL(READBEG)
READEND:
```

Analizziamo istruzione per istruzione quanto sopra descritto.

```
READBEG:
```

Questa è semplicemente una label di arrivo per una istruzione GOTO; è l'equivalente della istruzione TAG nell'ambito di un programma RPG. Fare attenzione nello specificare sempre i due punti finali (:).

```
RCVF
```

Legge un record dal file di database dichiarato con l'istruzione DCLF. Come si può notare non è necessario menzionare il nome del file da cui si sta leggendo essendo consentito l'utilizzo di un unico file.

```
MONMSG      MSGID(CPF0000)  EXEC(DO)
```

Nel caso in cui si verifichi un errore durante la lettura di un record nel file di database (fine file), lo stesso viene intercettato (il programma non va in errore) ed il pgm esegue l'istruzione DO, cioè tutte le istruzioni comprese tra il DO e il correlato ENDDO. Se l'errore non si verifica, le istruzioni comprese tra DO ed ENDDO non vengono eseguite.

```
GOTO        CMDLBL(READEND)
```

Indica al programma di riprendere l'esecuzione dalla label di arrivo READEND.
Si nota in questa occasione che il Control Language non contiene nessuna istruzione di programmazione strutturata adatta alla costruzione di cicli né qualcosa di simile alle subroutine, Ne segue un rilevante uso del GOTO, altrove bandito.

ENDDO

E' la chiusura del precedente DO. E' analoga alla stessa istruzione dell' RPG.

GOTO CMDLBL(READBEG)

Riporta l'esecuzione del programma alla label di arrivo READBEG e quindi, nel nostro esempio, alla lettura del file di database.

READEND:

Nel nostro esempio è la label di arrivo quando si verifica la segnalazione di fine record nel file di database.

11 Command Execution (QCMDEXC) nei programmi RPG. (11R)

QCMDEXC è un modulo precostituito, una Api (Application Program Interface), che viene messo a disposizione dal sistema, non modificabile, ed utilizzabile nell'ambito di un programma RPG. Il suo scopo è quello di consentire l'utilizzo di comandi CL in un programma RPG. Il suo richiamo avviene mediante il comando CALL esattamente come per il richiamo di qualsiasi altro programma. La parametrizzazione di questo modulo invece deve seguire un metodo ben preciso. A questo modulo si devono passare due e soltanto due parametri. Il primo parametro deve contenere la stringa relativa al comando CL che dovrà essere eseguito. Il secondo parametro deve contenere la lunghezza della stringa/comando che gli si sta passando da eseguire. Mentre la lunghezza del primo parametro è di lunghezza variabile, dipende dal comando CL che si intende venga eseguito, il secondo è di lunghezza fissa; è un campo numerico lungo 15 di cui 5 decimali.

11.1 Utilizzo di QCMDEXC per reindirizzamento.

Un tipico impiego del modulo QCMDEXC è quello di utilizzarlo per eseguire dei reindirizzamenti per un file di database. Siccome il programma RPG, se non diversamente specificato, nel momento in cui viene richiamato esegue un'apertura implicita di tutti i file di database definiti nelle specifiche di definizione dei file, ed il reindirizzamento per i file di database ha effetto soltanto se il file su cui si compie tale operazione è chiuso, si dovrà specificare che l'apertura (OPEN) per il file di database avverrà in modo esplicito (ovvero a programma tramite l'istruzione OPEN).

Di seguito viene riportata la definizione del file di database per il quale si richiede l'apertura esplicita a programma.

```
FX  FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++++++
    FANCLI00F  IF   E           K DISK    USROPN
```

Da notare la parola chiave USROPN la quale indica al programma che l'apertura del file ANCLI00F sarà a carico del programma stesso tramite il comando di apertura esplicita OPEN. Da notare che fino a quando il file non sarà aperto (OPEN) non sarà possibile eseguire alcuna operazione sullo stesso, altrimenti il programma restituirà un errore per il tentativo di utilizzare un file chiuso. Prima di procedere al richiamo del modulo QCMDEXC si deve comporre la stringa relativa al comando CL per il reindirizzamento. Ci sono almeno due metodi per costituire la stringa: il primo è quello di specificare il comando CL in una schiera interna al programma RPG; il secondo è quello di utilizzare sulle specifiche C estese l'istruzione EVAL e le Built In Function ad essa collegate (19.14). Dato che vi è un capitolo specifico nel quale si da ampia documentazione relativa alle specifiche C in forma estesa, in questa sede ci limiteremo a trattare il primo dei due metodi appena citati.

Innanzitutto si definisce la schiera CMD a tempo di compilazione di un solo elemento di 80 caratteri.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
     * Comandi CL
    D CMD             S             80    DIM(2) CTDATA PERRCD(1)
```

Quindi, in fondo al sorgente, si caricano i dati della schiera.

```
**   ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
** CMD - Comandi CL
OVRDBF     FILE(ANCLI00F) TOFILE(NOMELIB/ANCLI00F) MBR(NOMEMBR)
CLRPFM     FILE(WORK)
```

Come si può notare gli elementi della schiera contengono i singoli comandi CL che si intendono eseguire.

Definiti tutti gli elementi, il richiamo a QCMDEXC si esegue così.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Ridirige archivio ANCLI00F.
C                   MOVEL(P)  CMD(1)        QCMD             80
C                   Z-ADD     80            QCMDL            15 5
C                   CALL      'QCMDEXC'
C                   PARM                    QCMD
C                   PARM                    QCMDL
     * Apre il file ANCLI00F.
C                   OPEN      ANCLI00F
```

Sfruttando le MOVEL(P) e le Z-ADD implicite nelle PARM, la minutazione può essere più sintetica.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Ridirige archivio ANCLI00F.
C                   CALL      'QCMDEXC'
C                   PARM       CMD(1)       QCMD             80
C                   PARM       80           QCMDL            15 5
     * Apre il file ANCLI00F.
C                   OPEN      ANCLI00F
```

In questo modo si è proceduto al richiamo del QCMDEXC per eseguire il reindirizzamento per il file di database ANCLI00F, quindi si è eseguita l'apertura esplicita dello stesso. Da questo momento il programma può utilizzare il file di database ANCLI00F.

11.2 Altre operazioni.

Il modulo QCMDEXC può essere impiegato per eseguire tutti i comandi che si rendono necessari alla programmazione ma che il linguaggio RPG in senso stretto non consente. Se per esempio ad un certo punto del programma si rende necessario inviare un messaggio per informare circa l'andamento della elaborazione, le alternative al programmatore sono soltanto due. La prima è quella di richiamare un programma CLP, magari con opportuno parametro per individuare il tipo di messaggio da inviare; questa soluzione implica la codifica di un sorgente CLLE e la generazione di un oggetto sul sistema. L'altra è quella di utilizzare il modulo QCMDEXC richiamato dal programma RPG ed opportunamente parametrizzato.

12 Creazione di Comandi. (12C)

L'AS/400 mette a disposizione anche del programmatore applicativo la già illustrata interfaccia comandi.

La creazione di applicazioni governata da comandi chiede però una serie di competenze abbastanza distanti dalla normale programmazione interattiva e batch.

Per evitare una trattazione lunga e complessa, si esemplifica la creazione di un comando semplice ed utile.

12.1 Programma di utilità §SIF.

Il presente esempio illustra una tipologia di sviluppo caratteristica dei programmi di utilità dove l'interfaccia verso l'utente, che è solitamente un programmatore, è omogenea ai comandi di sistema.

Il problema da risolvere è il seguente.

Si vuole impostare nel lavoro interattivo la stessa lista librerie che il Modulo Base Ibm (altrimenti detto *Architettura*) imposta quando un utente di Modulo Base si collega ad un certo Sistema Informativo.

Si costruirà il comando §SIF (Lista librerie da modulo base) nella forma seguente.

§SIF SISINF(SisInf) TYPE(Type) OMITLOST(OmitLost)

Ovvero, nella forma più breve.

§SIF SisInf

I valori permessi sono illustrati di seguito. I valori sottolineati sono assunti per difetto.

SisInf Sistema Informativo.
=Nome del sistema informativo di Modulo Base.

Type Tipo ambiente.
=*PROD Cerca i sistemi informativi di produzione nel file ACGGAA/KFSIF01L.
=*TEST Cerca i sistemi informativi di prova nel file ACGTST/KFSIF01L.

OmitLost Ometti librerie inesistenti.
=*NO Non omettere le librerie mancanti e, nel caso, restituisci errore.
=*YES Ometti le librerie mancanti e, nel caso, non restituire errore.

Ad utility compilata, se "STAREND" è un sistema informativo presente e funzionante sotto Architettura e se ne vuole utilizzare la lista librerie nell'interattivo in corso, il comando da eseguire, nella sua forma più breve, sarà il seguente.

§SIF STAREND

Di seguito, si elencano ordinatamente i sorgenti necessari, anche se il concepimento delle varie parti dell'utility ha una cronologia diversa. Si riportano.

- Il comando.
- Il programma di elaborazione del comando
- Il testo d'aiuto del comando
- La stringa di creazione del comando

12.2 Comando §SIF.

Le caratteristiche del comando sono riportate integralmente nel paragrafo precedente: si osservino le corrispondenze puntuali tra l'analisi ed il sorgente sotto illustrato che verrà compilato dal comando CRTCMD presente nella stringa di creazione.

Command Source.

Libreria	**§CNUSRC**
File	**§SIF**
Membro	**§SIF**
Testo del membro:	**Lista librerie da modulo base. Cmd**
Tipo seu del membro:	**CMD**

```
/* Claudio Neroni 03/10/1998 Creato.                                  */
             CMD        PROMPT('Lista librerie da modulo base')
             PARM       KWD(SISINF) TYPE(*NAME) LEN(10) DFT(ACGGAA) +
                          SPCVAL((ACGGAA)) PROMPT('Sistema +
                          informativo')
             PARM       KWD(TYPE) TYPE(*CHAR) LEN(5) RSTD(*YES) +
                          DFT(*PROD) VALUES(*PROD *TEST) +
                          PROMPT('Tipo ambiente')
             PARM       KWD(OMITLOST) TYPE(*CHAR) LEN(4) RSTD(*YES) +
                          DFT(*NO) VALUES(*NO *YES) PROMPT('Ometti +
                          librerie inesistenti')
```

12.3 Programma di elaborazione del comando §SIFC.

Il comando è in realtà una semplice interfaccia di chiamata al programma esecutore. Invece di eseguire una faticosa CALL mirata e consapevole, si preferisce accedere al programma tramite un comando facile che non richiede né mira né consapevolezza circa il numero, la natura, la dimensione e i valori dei parametri che l'utente deve fornire al programma esecutore.

Il programma esecutore riceve cioè i parametri che gli sono apparecchiati dal comando che lo dichiara come proprio esecutore.

Il sorgente è completamente commentato, anzi, contiene l'analisi di dettaglio che lo ha generato e non lascia margini interpretativi.

Il sorgente sotto illustrato verrà compilato dal comando CRTBNDCL presente nella stringa di creazione.

Command Processing Program Source

Libreria	**§CNUSRC**
File	**§SIF**
Membro	**§SIFC**
Testo del membro:	**Lista librerie da modulo base. Cpp**
Tipo seu del membro:	**CLLE**

```
/* Lista librerie da ArcSisInfo.                                      */
/* Claudio Neroni 03/10/1998 Creato.                                  */
/*    Lista librerie da sistema informativo architetturale.           */

/* Riceve Sistema informativo.                                        */
/* Riceve Tipo del sistema informativo.                               */
/* Riceve Omissione delle librerie inesistenti.                       */
             PGM        PARM(&SISINF &TYPE &OMITLOST)
/* File dei Sistemi informativi.                                      */
             DCLF       FILE(ACGTST/KFSIF01L)
/* Sistema informativo.                                               */
             DCL        VAR(&SISINF) TYPE(*CHAR) LEN(10)
/* Tipo del sistema informativo.                                      */
             DCL        VAR(&TYPE) TYPE(*CHAR) LEN(5)
/* Omissione delle librerie inesistenti.                              */
             DCL        VAR(&OMITLOST) TYPE(*CHAR) LEN(4)
/* Parte comune lista librerie architettura                           */
/* (7 librerie per 12 caratteri).                                     */
             DCL        VAR(&KDPTCM) TYPE(*CHAR) LEN(84)
/* Lista librerie                                                     */
/* (24 librerie per 12 caratteri).                                    */
             DCL        VAR(&LIBL) TYPE(*CHAR) LEN(288)
/* Lista librerie compattata.                                         */
             DCL        VAR(&LIBL3) TYPE(*CHAR) LEN(288)
```

```
/* Libreria in elenco librerie.                                      */
            DCL        VAR(&LIBELE) TYPE(*CHAR) LEN(10)
/* Libreria in elenco parte comune.                                  */
            DCL        VAR(&COMELE) TYPE(*CHAR) LEN(10)
/* Comando da eseguire con QCMDEXC.                                  */
            DCL        VAR(&CMD) TYPE(*CHAR) LEN(400)
/* Trovato nel file il sistema informativo cercato.                  */
            DCL        VAR(&TROVATO) TYPE(*LGL) VALUE('0')
/* Numero massimo di librerie in lista.                              */
            DCL        VAR(&MAX) TYPE(*DEC) LEN(3) VALUE(24)
/* Numero massimo di librerie in sistema informativo.                */
            DCL        VAR(&MAXSIF) TYPE(*DEC) LEN(3) VALUE(14)
/* Numero massimo di librerie in parte comune.                       */
            DCL        VAR(&MAXCOM) TYPE(*DEC) LEN(3) VALUE(7)
/* Indici di servizio.                                               */
            DCL        VAR(&X) TYPE(*DEC) LEN(3)
            DCL        VAR(&X12) TYPE(*DEC) LEN(3)
            DCL        VAR(&Y) TYPE(*DEC) LEN(3)
            DCL        VAR(&Y12) TYPE(*DEC) LEN(3)
/* Prenotazione del CPF0001.                                         */
            DCL        VAR(&CPF0001) TYPE(*LGL)
/* Intercetta gli errori.                                            */
            MONMSG     MSGID(CPF0000) EXEC(GOTO CMDLBL(ERRORE))
/* Se il tipo ambiente richiesto è produzione, indirizza             */
/* la lettura sul file sistemi informativi di produzione.            */
            IF         COND(&TYPE *EQ *PROD) THEN(DO)
            OVRDBF     FILE(KFSIF01L) TOFILE(ACGGAA/KFSIF01L) +
                         LVLCHK(*NO)
            ENDDO
/* Legge tutti i record del file fino a trovare                      */
/* il sistema informativo cercato.                                   */
 READBEG:
            RCVF
            MONMSG     MSGID(CPF0864) EXEC(GOTO CMDLBL(READEND))
            IF         COND(&KNMSI *NE &SISINF) THEN(GOTO +
                         CMDLBL(READBEG))
            IF         COND(&ATKI1 *NE ' ') THEN(GOTO CMDLBL(READBEG))
            CHGVAR     VAR(&TROVATO) VALUE('1')
 READEND:
/* Se non trova il sistema informativo, diagnostica e rilascia.     */
            IF         COND(*NOT &TROVATO) THEN(DO)
            SNDPGMMSG  MSGID(§SI0001) MSGF(§SIF) MSGDTA(&SISINF +
                         *CAT &TYPE) MSGTYPE(*DIAG)
            GOTO       CMDLBL(CPF0001)
            ENDDO
/* Se richiesto sistema informativo di produzione.                   */
            IF         COND(&TYPE *EQ *PROD) THEN(DO)
/*    Recupera la parte comune lista librerie architettura.          */
            RTVDTAARA  DTAARA(ACGGAA/KDPTCM) RTNVAR(&KDPTCM)
/*    Filtra parte comune lista librerie.                            */
/*    Ogni libreria della parte comune viene cercata tra le 14 del   */
/*    sistema informativo. Se trovata, l'elemento viene sbiancato.   */
            DO
/*    Azzera l'indice di loop.                                       */
            CHGVAR     VAR(&X) VALUE(0)
/*    Inizio loop.                                                   */
 L1BEG:
/*    Incrementa l'indice.                                           */
            CHGVAR     VAR(&X) VALUE(&X + 1)
/*    Se l'indice supera il numero massimo, abbandona.               */
            IF         COND(&X *GT &MAXCOM) THEN(GOTO CMDLBL(L1END))
/*    Calcola la posizione dell'elemento lungo la stringa.           */
            CHGVAR     VAR(&X12) VALUE((&X - 1) * 12 +1)
/*    Estrae una libreria comune dall'elenco.                        */
            CHGVAR     VAR(&COMELE) VALUE(%SST(&KDPTCM &X12 10))
/*    Se la libreria comune è valorizzata.                           */
            IF         COND(&COMELE *NE ' ') THEN(DO)
/*       Esamina tutte le librerie del sistema informativo.          */
```

```
/*       Cerca la libreria comune tra quelle del sistema informativo   */
/*       e, se la trova, la sbianca per evitare doppia libreria.       */
             DO
/*           Azzera l'indice di loop.                                  */
             CHGVAR     VAR(&Y) VALUE(0)
/*           Inizio loop.                                              */
 L2BEG:
/*           Incrementa l'indice.                                      */
             CHGVAR     VAR(&Y) VALUE(&Y + 1)
/*           Se l'indice supera il numero massimo, abbandona.          */
             IF         COND(&Y *GT &MAXSIF) THEN(GOTO CMDLBL(L2END))
/*           Calcola la posizione dell'elemento lungo la stringa.      */
             CHGVAR     VAR(&Y12) VALUE((&Y - 1) * 12 +1)
/*           Estrae una libreria dal sistema informativo.              */
             CHGVAR     VAR(&LIBELE) VALUE(%SST(&KLBSI &Y12 10))
/*           Se la libreria sistema informativo è uguale               */
/*           alla libreria comune cercata.                             */
             IF         COND(&LIBELE *EQ &COMELE) THEN(DO)
/*             Sbianca la libreria della parte comune.                 */
             CHGVAR     VAR(%SST(&KDPTCM &X12 10)) VALUE(' ')
/*             Abbandona.                                              */
             GOTO       CMDLBL(L2END)
/*           Se la libreria sistema informativo è uguale               */
/*           alla libreria comune cercata.                             */
             ENDDO
/*           Ricicla.                                                  */
             GOTO       CMDLBL(L2BEG)
/*           Fine loop.                                                */
 L2END:
/*       Esamina tutte le librerie del sistema informativo.            */
             ENDDO
/*    Se la libreria comune è valorizzata.                             */
             ENDDO
/*    Ricicla.                                                         */
             GOTO       CMDLBL(L1BEG)
/*    Fine loop.                                                       */
 L1END:
/* Filtra parte comune lista librerie.                                 */
             ENDDO
/*    Compone la lista librerie.                                       */
/*    Ciascun nome occupa esattamente 12 caratteri,                    */
/*    compresi due spazi finali.                                       */
/*    Notare il troncamento di due carattere inutili in fondo          */
/*    alla lista proveniente dal file sistemi informativi              */
/*    (campo &KLBSI del file KFSIF01L accorciato da 170 a 168).        */
/*    Anche le costanti contengono 12 posizioni per ogni libreria.     */
             CHGVAR     VAR(&LIBL) VALUE('QTEMP       ' *CAT +
                          %SST(&KLBSI 1 168) *CAT 'ACGGAA      +
                          QGPL        ' *CAT &KDPTCM)
/* Se richiesto sistema informativo di produzione.                     */
             ENDDO
/* Se richiesto sistema informativo di test.                           */
             ELSE       CMD(DO)
/*    Compone la lista librerie con i soli dati provenienti            */
/*    dal record sistemi informativi di test.                          */
             CHGVAR     VAR(&LIBL) VALUE(%SST(&KLBSI 1 168))
/* Se richiesto sistema informativo di test.                           */
             ENDDO
/* Compatta e filtra la lista librerie.                                */
             DO
/*    Azzera l'indice di loop.                                         */
             CHGVAR     VAR(&X) VALUE(0)
/*    Inizio loop.                                                     */
 L3BEG:
/*    Incrementa l'indice.                                             */
             CHGVAR     VAR(&X) VALUE(&X + 1)
/*    Se l'indice supera il numero massimo, abbandona.                 */
             IF         COND(&X *GT &MAX) THEN(GOTO CMDLBL(L3END))
```

```
/*    Calcola la posizione dell'elemento lungo la stringa.             */
            CHGVAR     VAR(&X12) VALUE((&X - 1) * 12 +1)
/*    Estrae l'elemento dalla stringa.                                 */
            CHGVAR     VAR(&LIBELE) VALUE(%SST(&LIBL &X12 10))
/*    Se richiesto lo scarto delle librerie inesistenti                */
/*    e se l'elemento è valorizzato.                                   */
            IF         COND((&OMITLOST *EQ *YES) *AND (&LIBELE *NE +
                         ' ')) THEN(DO)
/*       Controlla l'esistenza della libreria.                         */
            CHKOBJ     OBJ(&LIBELE) OBJTYPE(*LIB)
/*       Se la libreria non esiste, sbianca l'elemento.                */
            MONMSG     MSGID(CPF0000) EXEC(CHGVAR VAR(&LIBELE) +
                         VALUE(' '))
/*    Se richiesto lo scarto delle librerie inesistenti                */
/*    e se l'elemento è valorizzato.                                   */
            ENDDO
/*    Se non corre il primo elemento, aggiunge l'elemento alla lista.  */
            IF         COND(&X *NE 1) THEN(CHGVAR VAR(&LIBL3) +
                         VALUE(&LIBL3 *BCAT &LIBELE))
/*    Se corre il primo elemento, inizializza la lista con esso.       */
            ELSE       CMD(CHGVAR VAR(&LIBL3) VALUE(&LIBELE))
/*    Ricicla.                                                         */
            GOTO       CMDLBL(L3BEG)
/*    Fine loop.                                                       */
 L3END:
/* Compatta e filtra la lista librerie.                               */
            ENDDO
/* Se la lista superstite è in bianco, inserisce *NONE.               */
            IF         COND(&LIBL3 *EQ ' ') THEN(CHGVAR VAR(&LIBL3) +
                         VALUE(*NONE))
/* Compone il comando eseguibile di modifica                          */
/* della lista librerie.                                              */
            CHGVAR     VAR(&CMD) VALUE('CHGLIBL LIBL(' *CAT &LIBL3 +
                         *CAT ')')
/* Esegue il comando di sostituzione lista librerie.                  */
            CALL       PGM(QCMDEXC) PARM(&CMD 400)
/* Se errore, diagnostica e rilascia.                                 */
            MONMSG     MSGID(CPF0000 MCH0000) EXEC(DO)
            SNDPGMMSG  MSGID(§SI0002) MSGF(§SIF) MSGDTA(&SISINF +
                         *CAT &TYPE *CAT &LIBL3) MSGTYPE(*DIAG)
            GOTO       CMDLBL(CPF0001)
            ENDDO
/* Manda messaggio di felice esecuzione.                              */
            SNDPGMMSG  MSGID(§SI0003) MSGF(§SIF) MSGDTA(&SISINF +
                         *CAT &TYPE *CAT &LIBL3) MSGTYPE(*COMP)
/* Label di esecuzione delle attività finali.                         */
 RCLRSC:
/* Riacquisisce le risorse.                                           */
            RCLRSC
            MONMSG     MSGID(CPF0000 MCH0000)
/* Se richiesto, rilascia il CPF0001.                                 */
            IF         COND(&CPF0001) THEN(DO)
            SNDPGMMSG  MSGID(CPF0001) MSGF(QCPFMSG) MSGDTA(§SIF) +
                         MSGTYPE(*ESCAPE)
            MONMSG     MSGID(CPF0000 MCH0000)
            ENDDO
/* Ritorna.                                                           */
            RETURN
/* Label di errore.                                                   */
 ERRORE:
/* Restituisce i messaggi al chiamante,                               */
/* trasformando eventuali escape in diagnostici.                      */
            §RSNMSG
            MONMSG     MSGID(CPF0000 MCH0000)
/* Label di prenotazione del CPF0001.                                 */
 CPF0001:
/* Prenota il CPF0001.                                                */
            CHGVAR     VAR(&CPF0001) VALUE('1')
```

```
             MONMSG     MSGID(CPF0000 MCH0000)
/* Salta alle attività finali.                                        */
             GOTO       CMDLBL(RCLRSC)
             ENDPGM
```

Si noti la chiamata del comando §RSNMSG (Resend Message), un'altra utility che non viene illustrata e che serve a restituire i messaggi del programma chiamante al programma ancora prima in lista di chiamata. Può essere asteriscata senza danno o può essere compilata come prerequisito dai sorgenti che accompagnano il corso.

12.4 Testo d'aiuto del comando.

Per la miglior comprensione del comando, gli si associa, durante la creazione, un testo di aiuto interattivo richiamabile dal prompter premendo F1. Viene allora esposta quella parte del testo di aiuto attribuita al parametro su cui si trova il cursore ed è poi possibile navigare in tutto il testo.

Pur essendo quello del Panel Group un linguaggio farraginoso ereditato dai mainframe, è anche l'unico modo di attribuire l'aiuto ad un comando e si fa buon viso a cattivo gioco.

Il sorgente sotto illustrato verrà compilato dal comando CRTPNLGRP presente nella stringa di creazione.

Panel Group Source.

Libreria	**§CNUSRC**
File	**§SIF**
Membro	**§SIFP**
Testo del membro:	**Lista librerie da modulo base. Hlp**
Tipo seu del membro:	**PNLGRP**

```
:PNLGRP.
.*--------------------------------------------------------------------
:HELP NAME=CMD.
:H3.Comando §SIF
:H2.Lista librerie da modulo base
:P.Il comando permette di impostare la lista librerie
di un sistema informativo di modulo base.
:H3.Messaggi di rilascio specifici che possono essere intercettati.
:PC.CPF0001 Nel comando è stato trovato un errore.
:EHELP.
.*--------------------------------------------------------------------
:HELP name='CMD/sisinf'.
:H3.Sistema informativo (SISINF)
:P.Nome del sistema informativo del modulo base
del quale impostare la lista librerie.
:P.Valori permessi:
:PARML.
:PT.:PK DEF.ACGGAA:EPK.Ä
:PD.Nome del sistema informativo di controllo del modulo base.
:PT.nome-sistema-informativo
:PD.Nome di un sistema informativo codificato nel modulo base.
:EPARML.
:EHELP.
.*--------------------------------------------------------------------
:HELP name='CMD/type'.
:H3.Tipo ambiente (TYPE)
:P.Sceglie il file dei sistemi informativi di produzione o di prova
del modulo base.
:P.Valori permessi:
:PARML.
:PT.:PK DEF.*PROD:EPK.Ä
:PD.Il file dei sistemi informativi da utilizzare è
quello di produzione del modulo base:
KFSIF01L della libreria ACGGAA.
:PT.*TEST
:PD.Il file dei sistemi informativi da utilizzare è
```

```
quello di prova del modulo base,
normalmente utilizzato tramite il comando KLANCIA:
KFSIF01L della libreria ACGTST.
:EPARML.
:EHELP.
.*-------------------------------------------------------------------------
:HELP name='CMD/omitlost'.
:H3.Ometti librerie inesistenti (OMITLOST)
:P.Se una o più librerie non esistono
attualmente sul sistema, è possibile ometterle
dalla lista librerie in costruzione.
:P.Valori permessi:
:PARML.
:PT.:PK DEF.*NO:EPK.Ä
:PD.Non omette le librerie inesistenti dalla lista librerie
in costruzione.
:PT.*YES
:PD.Omette le librerie inesistenti dalla lista librerie
in costruzione.
:EPARML.
:EHELP.
.*-------------------------------------------------------------------------
:EPNLGRP.
```

12.5 Stringa di creazione del programma di utilità.

Il sorgente sotto illustrato verrà mandato in esecuzione come job batch dal comando SBMDBJOB (Submit Database Job). Si presume l'esistenza sul sistema di compilazione dei sorgenti precedentemente descritti, nonché l'esistenza della libreria **§CNU** contenitrice dei programmi di utilità.

Nella stringa sono contenuti anche i comandi di creazione di un ulteriore oggetto che non è dotato di un sorgente proprio ma solo di comandi di creazione ed aggiunta: il file messaggi §SIF. È buona abitudine considerare la stringa come sorgente di tale oggetto ed averne la stessa cura degli altri sorgenti.

A proposito di questi si osservi che i parametri di compilazione non coincidono con quelli che il Pdm (Program Developement Manager) (5.5) assume per difetto quando sull'elenco membri si esegue l'opzione 14 di creazione. Si consiglia perciò a tutti gli effetti di avere per la stringa di creazione la stessa cura che si elargisce a tutti gli altri sorgenti.

Creation Job Stream Source.

Libreria	**§CNUSRC**
File	**§SIF**
Membro	**§SIF§**
Testo del membro:	**Lista librerie da modulo base. JobStr**
Tipo seu del membro:	**CL**

```
//BCHJOB   JOB(§SIF.) JOBD(QBATCH) INLLIBL(QTEMP §CNU §CNUSRC QGPL) +
           ENDSEV(60) LOG(4 00 *SECLVL) MSGQ(*USRPRF)

/* Claudio Neroni 03/10/1998 Creato.                                  */

/* §SIF.                                                             */
/* Lista librerie da ArcSisInfo.                                     */
/* Prerequisiti: §RSNMSG.                                            */

DLTCMD     CMD(§CNU/§SIF)
DLTPNLGRP  PNLGRP(§CNU/§SIFP)
DLTPGM     PGM(§CNU/§SIFC)
DLTMSGF    MSGF(§CNU/§SIF)

CRTBNDCL   PGM(§CNU/§SIFC) SRCFILE(§SIF) DBGVIEW(*ALL)
CRTPNLGRP  PNLGRP(§CNU/§SIFP) SRCFILE(§SIF)
CRTCMD     CMD(§CNU/§SIF) PGM(§SIFC) SRCFILE(§SIF) ALLOW(*ALL) +
           HLPPNLGRP(§SIFP) HLPID(CMD) PRDLIB(§CNU)
```

```
CRTMSGF    MSGF(§CNU/§SIF) TEXT('Lista librerie da modulo +
             base. Msgf')
ADDMSGD    MSGID(§SI0001) MSGF(§CNU/§SIF) MSG('Il sistema +
             informativo architetturale &2 &1 non esiste.') +
             SECLVL('Il sistema informativo &2 &1 non esiste nel +
             file architetturale KFSIF01L della libreria ACGGAA (se +
             richiesto tipo ambiente produzione *PROD) o della +
             libreria ACGTST (se richiesto ambiente test *TEST).') +
             FMT((*CHAR 10) (*CHAR 5))
ADDMSGD    MSGID(§SI0002) MSGF(§CNU/§SIF) MSG('SisInfo &2 &1 errore +
             su Libl &3') SECLVL('Dal sistema informativo +
             architetturale &2 &1 è stata prelevata ma è in errore +
             la lista librerie &3.') FMT((*CHAR 10) (*CHAR 5) +
             (*CHAR 276))
ADDMSGD    MSGID(§SI0003) MSGF(§CNU/§SIF) MSG('SisInfo &2 &1 Libl +
             &3') SECLVL('Dal sistema informativo architetturale &2 +
             &1 è stata prelevata ed usata la lista librerie &3.') +
             FMT((*CHAR 10) (*CHAR 5) (*CHAR 276))
//ENDBCHJOB
```

13 File Video. (13C)

Tutti i linguaggi dell'AS/400, e in particolare l' RPG, utilizzano il video (in vecchio e stagionato ambiente carattere) definendo un file di interfaccia con il dispositivo. La tecnica utilizzata prevede l'invio al video di record, ciascuno con un tracciato in uscita, e la ricezione di ritorno dei medesimi record, ciascuno con un proprio tracciato di ritorno. L'attività nei confronti del video si limita perciò a scrivere e leggere dei record sul e dal video.

L'argomento video è molto importante, e quindi esteso, perché permette la realizzazione dell'interfaccia interattiva. La trattazione si limiterà tuttavia alle tecniche essenziali, evitando sofismi resi obsoleti dall'evoluzione delle prestazioni del software di base. Ad esempio, nelle prime versioni del software, si verificavano facilmente sfarfallamenti evitabili con l'uso di certe parole chiave. Gli sfarfallamenti sono in seguito spariti rendendo superflui gli accorgimenti precedenti.

13.1 Specifiche DDS.

I record scritti sul video e letti dal video sono definiti esternamente ai programmi tramite il medesimo linguaggio con cui si definiscono anche i file fisici, logici e di stampa: le DDS (Data Description Specification). In verità le DDS delle varie tipologie hanno in comune il tracciato e poche istruzioni e utilizzano comandi di creazione diversi, per il video CRTDSPF (Create Display File).

Non si esamina la possibilità, pure esistente, di definire record video altro che esterni.

13.2 Screen Design Aid.

In realtà è raro scrivere direttamente le DDS del video a causa dell'esistenza di una utility di grande efficacia che permette la definizione interattiva del video e che, alla fine, scrive automaticamente le opportune istruzioni DDS.

Normalmente l'SDA (Screen Design Aid) viene usato senz'altro per la prima stesura del video. La manutenzione del sorgente video provoca però la perdita delle date di manutenzione preesistenti. In molti casi tale perdita è dannosa e la si schiva evitando l'uso dell'Sda e preferendo il SEU o, meglio, operando il riporto manuale da un sorgente di lavoro, trattato con l'Sda, all'originale, trattato con il solo SEU.

Per una breve guida all'Sda, si consulti l'apposito capitolo (36).

13.3 Video pieno.

Si esemplificano alcuni schemi tipici di formati video.

Il primo ed indispensabile è il video pieno, monolitico, che può occupare tutte le righe dello schermo. Il formato inizia con la prima riga dotata di campi o costanti e termina con l'ultima riga egualmente dotata. Sono comprese tutte le righe vuote intermedie.

Le righe lasciate libere possono essere occupate con altri formati non interferenti emessi con la clausola **OVERLAY**.

Immagine del Video pieno A01W.

Sorgente del Video pieno A01W.

```
....+.*..1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
.....A*N01N02N03T.Name++++++RLen++TDpBLinPosFunctions++++++++++++++++++++++++++++
     A                                      DSPSIZ(24 80 *DS3)
     A          R W1
     A                                      CA03
     A                                      TEXT('Video pieno.')
     A                                    1 33'Titolo del video'
     A                                      DSPATR(UL)
     A                                      DSPATR(HI)
     A                                    3  2'Persona..:'
     A            W1PERS          3   B  3 14DSPATR(HI)
     A                                    4  2'Nome.....:'
     A            W1NOME         25   B  4 14DSPATR(HI)
     A                                    5  2'Cognome..:'
     A            W1COGN         25   B  5 14DSPATR(HI)
     A                                   23  2'F3=Fine'
```

Commenti al sorgente del Video pieno A01W.

```
Definisce la dimensione del video. Sono previste 24x80 e 27x132.
     A                                      DSPSIZ(24 80 *DS3)
Assegna un nome al formato video.
     A          R W1
Permette il comando F3.
     A                                      CA03
Assegna un testo al formato video.
     A                                      TEXT('Video pieno.')
Definisce un titolo sul video alla riga 1 colonna 33, sottolineato e in alta intensità.
     A                                    1 33'Titolo del video'
     A                                      DSPATR(UL)
     A                                      DSPATR(HI)
Definisce una costante esplicativa del campo successivo.
     A                                    3  2'Persona..:'
Definisce un campo per immissione ed emissione (B) di un dato "Persona".
     A            W1PERS          3   B  3 14DSPATR(HI)
Altro campo.
     A                                    4  2'Nome.....:'
     A            W1NOME         25   B  4 14DSPATR(HI)
Altro campo.
     A                                    5  2'Cognome..:'
     A            W1COGN         25   B  5 14DSPATR(HI)
Definisce una costante esplicativa del comando F3.
     A                                   23  2'F3=Fine'
```

13.4 Recupero del nome del record e del nome del campo su cui si trova il cursore.

Al rientro dal video, se è dichiarata la parola chiave **RTNCSRLOC**, il sistema fornisce nei campi menzionati dalla parola chiave il nome del campo e del record su cui si trova il cursore, utili per chiamare finestre di elenchi e di aiuto.

13.5 Recupero della posizione su cui si trova il cursore.

Al rientro dal video, il sistema fornisce le coordinate del cursore in modo quasi criptato all'interno della File Information Data Structure. Per ottenere le coordinate si utilizza con profitto e facilità un programma di utilità apposito. Le coordinate che si ottengono possono essere riutilizzate in chiaro nella parola chiave **CSRLOC** al momento di riemettere il video.

Sorgente A02W commentato per l'uso della locazione cursore.

```
     A                                      DSPSIZ(24 80 *DS3)
     A          R W1
     A                                      CA03
     A                                      TEXT('Video pieno.')
La riga seguente attiva il Return cursor location.
     A                                      RTNCSRLOC(&RCD &FLD)
La riga seguente attiva il Cursor location.
     A                                      CSRLOC(LIN POS)
Le due righe seguenti definiscono i campi usati nel Return cursor location.
     A            RCD           10   H
     A            FLD           10   H
Le due righe seguenti definiscono i campi usati nel Cursor location.
     A            LIN            3  0H
     A            POS            3  0H
     A                                 1 33'Titolo del video'
     A                                     DSPATR(UL)
     A                                     DSPATR(HI)
     A                                   3  2'Persona..:'
     A            W1PERS         3   B  3 14DSPATR(HI)
     A                                   4  2'Nome.....:'
     A            W1NOME        25   B  4 14DSPATR(HI)
     A                                   5  2'Cognome..:'
     A            W1COGN        25   B  5 14DSPATR(HI)
     A                                  23  2'F3=Fine'
```

13.6 Programma di utilità per l'interpretazione della posizione del cursore.

Sorgenti ed esempi di chiamata reperibili al link: www.neroni.it/Scaricabili/CRSL.txt

Sorgente di CRSLDS (Data structure Cursor location)

```
CN    * Cursor location.
     A          R CRSLDS
     A                                      TEXT('Cursor location. +
     A                                            File information +
     A                                            data structure')
     A            CRSL01         8
     A                                      COLHDG('...')
     A            CRSLOP         1
     A                                      COLHDG('File' +
     A                                             'open')
     A            CRSL02         1
     A                                      COLHDG('...')
     A            CRSLST         5
     A                                      COLHDG('Status' +
     A                                             'code')
     A            CRSL03        30
     A                                      COLHDG('...')
     A            CRSLER         7
     A                                      COLHDG('Error' +
     A                                             'code')
     A            CRSL04        30
     A                                      COLHDG('...')
     A            CRSLFI        10
     A                                      COLHDG('File')
     A            CRSLLB        10
     A                                      COLHDG('Library')
     A            CRSL05        26
     A                                      COLHDG('...')
     A            CRSLMB        10
     A                                      COLHDG('Member')
     A            CRSL06        13
     A                                      COLHDG('...')
     A            CRSLFL         4B
     A                                      COLHDG('Forms lines or' +
     A                                             'workstation lines')
     A            CRSL07         2
```

```
     A                                      COLHDG('...')
     A            CRSLNR         9B
     A                                      COLHDG('Record quantity' +
     A                                             'in input file' +
     A                                             'at open')
     A            CRSLAT         2
     A                                      COLHDG('Access' +
     A                                             'type')
     A            CRSL08         1
     A                                      COLHDG('...')
     A            CRSLSO         1
     A                                      COLHDG('Source' +
     A                                             'file' +
     A                                             'flag')
     A            CRSL09        24
     A                                      COLHDG('...')
     A            CRSLOL         4B
     A                                      COLHDG('Overflow' +
     A                                             'line')
     A            CRSL10        71
     A                                      COLHDG('...')
     A            CRSLRE        10
     A                                      COLHDG('Last record' +
     A                                             'format' +
     A                                             'processed')
     A            CRSLDC         2
     A                                      COLHDG('Device' +
     A                                             'class')
     A            CRSL11        10
     A                                      COLHDG('...')
     A            CRSLLE         9B
     A                                      COLHDG('First read' +
     A                                             'record' +
     A                                             'length')
     A            CRSL12        80
     A                                      COLHDG('...')
     A            CRSLCL         4B
     A                                      COLHDG('Current' +
     A                                             'line')
     A            CRSL13         1
     A                                      COLHDG('...')
     A            CRSLLP         2
     A                                      COLHDG('Line/' +
     A                                             'position')
```

Sorgente di CRSLOC (Cursor location)

```
CN    /TITLE Cursor location.
      *--------------------------------------------------------------------
      * Riceve in CRSL la file information data structure della
      *    workstation da cui estrae:
      *    in CRSLDC il codice del dispositivo su cui si opera.
      *    in CRSLLP il codice della locazione del cursore
      *              su di un file display.
      * Restituisce in PPLIN e PPPOS linea e posizione decodificati.
      *--------------------------------------------------------------------
      * Restituisce H1 se il primo byte di CRSLDC e' diverso
      *    dall'esadecimale 01 (Keyboard display).
      *    Tramite il secondo byte di CRSLDC distingueva mezzo video
      *    e console 38 da tutti gli altri.
      *    Esadecimale 00 (5250 Display Station, 960 characters).
      *    Esadecimale 01 (System console, 1024 characters).
      *    Informazioni sul contenuto del byte in oggetto
      *    potevano essere desunte per il S/38 dalla CPF Programmer's Guide
      *    appendix D (Open and I/O Feedback Area) pagina D-16
      *    nel release 8.0.
      *    SU AS/400 VEDI "PROGRAMMING: DATA MANAGEMENT GUIDE" APPENDIX A.
      * Restituisce H3 se la linea calcolata per il cursore
      *    supera la misura imposta dal video utilizzato.
      * Restituisce H4 se la posizione del cursore lungo la linea
      *    supera la misura imposta dal video utilizzato.
      *--------------------------------------------------------------------
```

```
     * File information data structure della workstation.
    D crslds        e ds
     *-------------------------------------------------------------------
     * Scambia parametri.
    C     *entry        plist
     *    Riceve la file information data structure del video.
    C                   parm                    crslds                    I FIDS Workstn
     *    Restituisce la linea (riga) del cursore.
    C                   parm                    pplin             3 0     O Line
     *    Restituisce la posizione (colonna) del cursore.
    C                   parm                    pppos             3 0     O Position
     * Pulisce i parametri di ritorno.
    C                   clear                   pplin
    C                   clear                   pppos
     *-------------------------------------------------------------------
     * Decodifica Line/Position.
    C                   do
     *    Se il device non e' Keyboard Display, errore.
    C                   movel     crsldc        test
    C     test          ifne      x'01'
    C                   seton                                         h1
    C                   leave
    C                   endif
     *    Decodifica il byte che indica la linea del cursore.
    C                   movel     crsllp        test
    C                   exsr      bynr
    C                   z-add     n             lin               3 0
     *    Se la linea trovata e' superiore al massimo previsto, errore.
    C     lin           ifgt      27
    C                   seton                                         h3
    C                   leave
    C                   endif
     *    Decodifica il byte che indica la posizione del cursore.
    C                   move      crsllp        test
    C                   exsr      bynr
    C                   z-add     n             pos               3 0
     *    Se la posizione trovata e' superiore al massimo previsto, errore.
    C     pos           ifgt      132
    C                   seton                                         h4
    C                   leave
    C                   endif
     *    Restituisce la linea trovata.
    C                   z-add     lin           pplin
     *    Restituisce la posizione trovata.
    C                   z-add     pos           pppos
     * Decodifica Line/Position.
    C                   enddo
     * Ritorna.
    C                   return
     *-------------------------------------------------------------------
     * Interpreta un byte come binario assoluto.
    C     BYNR          BEGSR
    C                   clear                   N                 3 0
    C                   TESTB     '7'           TEST              1      50
    C   50              ADD       1             N
    C                   TESTB     '6'           TEST                     50
    C   50              ADD       2             N
    C                   TESTB     '5'           TEST                     50
    C   50              ADD       4             N
    C                   TESTB     '4'           TEST                     50
    C   50              ADD       8             N
    C                   TESTB     '3'           TEST                     50
    C   50              ADD       16            N
    C                   TESTB     '2'           TEST                     50
    C   50              ADD       32            N
    C                   TESTB     '1'           TEST                     50
    C   50              ADD       64            N
    C                   TESTB     '0'           TEST                     50
    C   50              ADD       128           N
    C                   ENDSR
     *-------------------------------------------------------------------
```

13.7 Posizionamento del cursore sul video in emissione.

Il cursore individua la posizione sulla quale si imprime il carattere digitato. Il suo miglior posizionamento facilita l'attività dell'utente.

Al momento dell'emissione di un record video, il cursore si posiziona seguendo una serie di regole. Vale la prima soddisfatta.

a. Parola chiave a livello di record **CRSLOC** con entrambe le coordinate in essa rappresentate diverse da zero.
b. Parola chiave DSPATR(PC) a livello di campo condizionata favorevolmente.
c. Primo campo di immissione del video.

13.8 Visualizzazione degli errori.

Quando tramite un video si vogliono immettere o modificare dei dati, si rende necessario che il programma gestore del video stesso, prima di modificare il database, esegua il controllo di validità formale e sostanziale e, in caso di rifiuto, presenti un sintetico messaggio che spieghi l'errore rilevato.

Gli errori sono di due tipi.

- Warning (lieve): non rifiuta la modifica.
- Terminal (grave): rifiuta la modifica.

Si discute qui la modalità di segnalazione degli errori gravi. Una modifica afflitta da errori gravi viene sempre rifiutata integralmente.

Subito dopo il campo da segnalare si inserisce una riga di errore. Quando l'emissione del video W1 (EXFMT W1) avviene con un messaggio di errore condizionato favorevolmente (indicatori 60 e 61 accesi), la videata si presenta con il campo in rovesciamento di immagine, con il cursore posizionato, con il testo d'errore "Dato obbligatorio" al piede del video e con l'immissione bloccata.

L'emissione di errore deve avvenire sempre a ricalco della stessa videata (W1) già presente a video, pena la totale illeggibilità del video,perché l'emissione con un ERRMSG eccitato soffre di una antica stranezza tecnica, la PUTRETAIN implicita: vengono emessi soltanto i caratteri di controllo dei campi per provocare reverse dove comanda l'errore e il testo del messaggio dove previsto da ?????.

Ad evitare il detto pasticcio, se acceso l'errore, vale l'espediente di emettere preliminarmente W1 (WRITE W1) con l'indicatore 60 (errore generico) provvisoriamente spento e poi EXFMT W1 con 60 di nuovo acceso. Questa tecnica permette anche di presentare sempre le decodifiche.

```
 DP  A*N01N02N03T.Name++++++RLen++TDpBLinPosFunctions++++++++++++++++++++++++++++
     A                                       DSPSIZ(24 80 *DS3)
     A          R W1
     A                                       CA03
     A                                       TEXT('Video pieno.')
     A                                       RTNCSRLOC(&RCD &FLD)
     A                                       CSRLOC(LIN POS)
     A            RCD           10   H
     A            FLD           10   H
     A            LIN            3  0H
     A            POS            3  0H
     A                                   1 33'Titolo del video'
     A                                       DSPATR(UL)
     A                                       DSPATR(HI)
     A                                   3  2'Persona..:'
     A            W1PERS         3   B  3 14DSPATR(HI)
Assegna al campo un messaggio di errore condizionato da indicatori.
     A  60 61                                ERRMSG('Dato obbligatorio')
     A                                   4  2'Nome.....:'
     A            W1NOME        25   B  4 14DSPATR(HI)
     A                                   5  2'Cognome..:'
     A            W1COGN        25   B  5 14DSPATR(HI)
     A                                  23  2'F3=Fine'
```

La parola chiave ERRMSG con il testo può essere sostituita dalla parola ERRMSGID con il testo presente solo nel file messaggi precostituito.

O ancora da un generico ERRMSGID(CPF9897 QCPFMSG &MSG) passando il testo nel campo MSG.

L'uso di un testo a programma e di un solo errore generico a video per ciascun campo permette di sprecare un solo indicatore di errore per campo invece che un indicatore per ogni errore-

Per altro, tale tecnica va complicata per estirpare i testi dal sorgente del programma, specialmente in vista di una internazionalizzazione dell'applicazione. Cioè, si deve fare in modo che lo stesso programma giri in tutte le lingue senza moltiplicare sorgenti, compilazioni ed oggetti eseguibili.

Esempio d'uso Pgm emissione errori..

13.9 Subfile.

Uno degli argomenti più ostici dell' AS/400 è questo. Affrontalo con attenzione. Anche a chi lo scrive sta costando più fatica degli altri perché ogni parola va pesata più volte per ottenere un racconto meglio leggibile.

Il subfile è semplicemente un elenco a video che si gestisce con istruzioni in parte specifiche e in parte condivise col database.

Specifiche le parole chiave sulle DDS che descrivono il subfile e l'operazione RPG READC.

Condivise le operazioni RPG WRITE, CHAIN, UPDATE, EXFMT

Un elenco a video potrebbe essere costruito con molta più fatica in altri due modi mediante l'uso di due tecniche che non si approfondiscono in questa sede.

- Un formato video pieno con tante righe simili con nomi di campi ripetitivi con desinenza progressiva (PRI01, PRI02, …; SEC01, SEC02, …) fino a riempire il numero di righe disponibili, quindi componendo una sola pagina a video.
- Un formato video a riga variabile, introdotto su As/400 per facilitare le conversioni da Ibm Sistema 36 e che permette egualmente una sola pagina a video.

Entrambi i metodi richiedono un'area di memoria in cui accantonare i dati presentati a video, specialmente se ne occorre una successiva manipolazione. L'area può essere un gruppo di schiere o una struttura dati a ricorrenze multiple.

I tipi di subfile dati si differenziano per una delle seguenti caratteristiche.

- Il numero di record contenuti nel subfile rispetto a quelli contenuti in una pagina.
- Il riempimento iniziale completo del subfile o il riempimento graduale a richiesta.

Sono tre i tipi utili di subfile dati.

- Subfile a riempimento globale.
  Il subfile è riempito prima della visualizzazione.
- Subfile a riempimento graduale.
  Si riempie una sola pagina del subfile prima della visualizzazione. La pagina successiva viene caricata e visualizzata a richiesta.
- Subfile quadrato.
  Il subfile è di una pagina sola e gode della possibilità di avere righe disuguali, negata agli altri. Quadrato è un nome apocrifo inventato da un mio eccellente capo, Casagrande, che aveva osservato che questo tipo di definizione prevedeva due misure uguali (SFLSIZ = SFLPAG), da cui "quadrato".

13.10 Regole comuni ai vari tipi di subfile.

Il subfile è un file diretto (indirizzato mediante il numero relativo di record) che viene preparato in memoria e poi emesso e manipolato a video. Il programma gestore del subfile compie più tipi di attività.

- Preparazione del contenuto del subfile.
- Presentazione a video del subfile.
- Elaborazione del subfile manipolato.

Nel file video si definiscono due record per ogni subfile.

- Il record di subfile vero e proprio che chiameremo S1.
- Il record di controllo che chiameremo C1.

La definizione di S1 contiene due informazioni importanti.

- Il tracciato del record, cioè l'elenco dei campi.
- La posizione a video degli stessi campi.

La definizione del controllo C1 contiene invece.

- In SFLSIZ (Subfile size) quanti record S1 sono contenuti nel subfile.
- In SFLPAG (Subfile page) quanti record S1 sono contemporaneamente presentati a video prima di rollare in avanti per vedere i successivi.

S1 ha un tracciato solitamente composto da un sottoinsieme dei campi di un record di database, naturalmente modificati i nomi in maniera sistematica.

La definizione delle posizioni occupate dal record S1 riguarda solo il primo record a video.

Il primo record S1 occupa una o più righe del video, quelle indicate nella definizione, mentre i successivi occupano a video le posizioni immediatamente seguenti. Definiamo *pagina di subfile* il numero di record S1 visibili alla prima uscita del video e *corpo del subfile* l'insieme delle righe video corrispondenti alla pagina.

SFLSIZ discrimina tra il subfile quadrato e gli altri. Quadrato è quello con SFLPAG è uguale a SFLPAG, ovvero quel subfile che presenta a video tutto il suo contenuto in una sola volta e senza lasciare nulla fuori vista.

Se il record S1 occupa una sola riga video, il corpo occupa a video un numero di righe uguale a SLFPAG; se occupa due righe, il corpo occupa due volte il numero espresso in SFLPAG. E' perciò necessario che SFLPAG non provochi un corpo più grande del numero di righe video disponibili per lo scopo.

Nelle righe del video sopra il corpo si posiziona un record che chiameremo T1 (Testa), nel quale giaceranno solitamente le intestazioni di colonna e le opzioni di parzializzazioni dell'elenco.

Sotto il corpo si posiziona il record che chiameremo P1 (Piede), nel quale giaceranno solitamente le descrizioni dei comandi e le legende.

Il record di controllo può essere emesso sopra o sotto il gruppo di righe video destinate alla visualizzazione delle righe del subfile.

Se nel record C1 si mettono le intestazioni, si elimina T1 incorporandolo in C1.

Se nel record C1 si mette il piede, si elimina P1 incorporandolo in C1.

13.11 Subfile a riempimento globale.

Il subfile globale viene riempito completamente prima dell'emissione. È il più facile da programmare perché non richiede la gestione a programma dello scorrimento (parole chiave ROLLUP e ROLLDOWN).

La "paginazione" è gestita dal sistema e l'unico sfizio che il programmatore si toglie è emetterlo ad una pagina diversa dalla prima, non tanto alla prima emissione, quanto al momento di riemettere il subfile posizionato su di un record in errore che non si trova in prima pagina.

Il posizionamento alla pagina avviene riempiendo il campo nascosto ??? dichiarato nella parola chiave SFLRRN.

E' meglio che SFLSIZ sia un multiplo di SFLPAG, normalmente il doppio, anche se SFLSIZ può essere in realtà un qualunque valore superiore a SFLPAG perché la definizione sia soddisfacente. Il valore SFLSIZ determina soltanto quante pagine sono riservate al subfile in memoria al momento in cui avviene l'apertura del video, anche se poi la misura può essere sfondata fino al massimo di 9999 senza problemi e senza avvisi durante la scrittura dei record S1 a calcolo.

Per testimoniare la presenza di altre pagine di subfile oltre quella in vista, si può chiedere al sistema di mettere un simbolo di continuazione in basso a destra nel corpo tramite la parola chiave SFLEND. Basta condizionarla con un indicatore negativo ad esempio N04 con 04 mai acceso (misterium fidei!).

Sul controllo C1 è utile la presenza dei campi chiave evidenziati anche in S1. Tali campi possono permettere il posizionamento alla pagina che contiene la chiave stessa o quella immediatamente precedente. In questo caso non si ricarica il subfile ma si riemette la pagina giusta, come già esposto.

Se il subfile resta a video a lungo e i dati esposti sono molto volatili, la visualizzazione diventa presto una finestra sul passato. Per questo occorre sempre fornire un comando (F5) che provochi il ricaricamento del subfile senza passare per il video guida che fa da prologo e porta scelte di parzializzazione.

13.12 Riempimento del subfile globale.

Si ipotizza che le letture del database avvengano in avanti, cioè con posizionamenti a base di SETLL e letture a base di READ e READE. Non si illustrano le lettura all'indietro, con i simmetrici SETGT, READP e READPE, ma la simmetria è perfetta. Questa nota vale anche per il riempimento del subfile graduale.

Si elencano le istruzioni utili al caricamento del subfile globale.

- Presentazione del video guida G1 contenente le informazioni di scelta del gruppo di record da caricare in elenco. Occorre soprattutto una chiave parziale della vista logica di accesso ai record di database.
- Pulizia totale del contenuto mediante WRITE del record C1 con condizionamento favorevole della parola chiave SFLCLR. Per finezza e per ingannare l'attesa, si può nell'occasione emettere anche le intestazioni del subfile accendendo la chiave di visualizzazione del controllo SFLDSPCTL. Azzeramento del contatore da usare nel seguito come numero relativo di record scrivendo S1.
- Ciclo di lettura del file di database del quale si vogliono presentare i record trascrivendoli nel subfile. Si fa uso della chiave parziale proveniente da G1.
  - Preparazione a calcolo del numero relativo di record da emettere nel subfile. Di solito è un semplice incremento.
  - Trascrizione del record dai campi del record di database ai corrispondenti campi della riga di subfile.
  - Sbiancamento dell'eventuale campo scelta,
  - Scrittura del record S1 di subfile mediante WRITE.
  - Accantonamento del numero relativo di record ultimo caricato nel subfile.
  - Il ciclo continua fino alla completa trascrizione del gruppo di record da database a riga di subfile. Il subfile in memoria risulta quindi costituito da più pagine.
- Emissione a video dei dati accumulati: EXFMT del record C1 con condizionamento favorevole delle parole chiave SFLDSP e SFLDSPCTL.

- Viene mostrata la prima pagina del subfile o, se presente il SFLRRN, la pagina che contiene il record S1 corrispondente.
- La visione della pagina successiva è ottenuta dal tasto ROLLUP, salvo che sull'ultima pagina, dove è impedito.
- La visione della pagina precedente è ottenuta dal tasto ROLLDOWN, salvo che sulla prima pagina, dove è impedito.
- Non si devono dichiarare né ROLLUP né ROLLDOWN perché sono gestiti dal sistema senza intervento del programma applicativo.
- Il passo successivo è l'esame delle scelte.

13.13 Subfile a riempimento graduale.

Il subfile graduale viene riempito solo parzialmente prima dell'emissione, normalmente solo la prima pagina. Al primo ROLLUP, il programma provvede al riempimento della seconda pagina e alla sua presentazione a video. Così via per le pagine successive.

Si definisce *ROLLUP di sfondamento* quello che viene richiesto per vedere una pagina non ancora caricata nel subfile.

La paginazione all'indietro è sempre gestita dal sistema. Quella in avanti è gestita dal sistema solo se non è di sfondamento.

L'uso di SFLEND è particolarmente utile esclusivamente se invita a rollare in avanti in presenza di altri record da vedere. Il problema è che prima di un roll di sfondamento l'ultimo record di subfile è già visualizzato e perciò non compare il simbolo di continuazione, se si è predisposto l'espediente "N04 SFLEND" senza la manovra di 04. Il programma di caricamento deve settare 04 in modo che sia spento se non ci sono altri record e, di contro, sia acceso se ci sono altri record. Ne segue che il programma, caricato l'ultimo record di una pagina, deve preleggere il record immediatamente successivo senza riportarlo nel subfile perché rovinerebbe l'impaginazione e il successivo riconoscimento dello sfondamento. Per non rovinarsi il ciclo di lettura in avanti a base di READ o READE, occorre subito riposizionarlo con un SETLL con l'ultima chiave letta. Simmetricamente, se il ciclo è di READP o di READPE, il riposizionamento è con un SETGT.

Ad eccezione di queste, tutte le altre considerazioni fatte per il globale valgono anche per il graduale.

Circa l'attualità dei dati presentati, occorre notare che le pagine di subfile sono caricate in momenti diversi e quindi temporalmente disomogenee. Una finestra storta sul passato! In pratica però non importa molto quando i dati sono poco volatili e la multiutenza non è frenetica.

Esempio d'uso.

13.14 Riempimento del subfile graduale.

Si elencano le istruzioni utili al caricamento del subfile graduale.

Lo spiegone è molto simile a quello del subfile globale ma viene ripetuto perché la miscellanea dei due sarebbe poco chiara.

- Presentazione del video guida G1 contenente le informazioni di scelta del gruppo di record da caricare in elenco. Occorre soprattutto una chiave parziale della vista logica di accesso ai record di database.
- Pulizia totale del contenuto mediante WRITE del record C1 con condizionamento favorevole della parola chiave SFLCLR. Per finezza e per ingannare l'attesa, si può nell'occasione emettere anche le intestazioni del subfile accendendo la chiave di visualizzazione del controllo SFLDSPCTL. Azzeramento del contatore da usare nel seguito come numero relativo di record scrivendo S1.

- Ciclo di lettura del file di database del quale si vogliono presentare i record trascrivendoli nel subfile. Si fa uso della chiave parziale proveniente da G1.
  - Preparazione a calcolo del numero relativo di record da emettere nel subfile. Di solito è un semplice incremento.
  - Trascrizione del record dai campi del record di database ai corrispondenti campi della riga di subfile.
  - Sbiancamento dell'eventuale campo scelta,
  - Scrittura del record S1 di subfile mediante WRITE.
  - Accantonamento del numero relativo di record ultimo caricato nel subfile.
  - Il ciclo continua fino al riempimento di una pagina per volta ch viene accodata elle pagine già riempite. Il subfile in memoria risulta quindi costituito da più pagine.
  - Ad ogni record letto e caricato nel subfile, occorre annotare l'ultima chiave letta dal database e l'ultimo relative record number caricato nel subfile. Quest'ultimo direttamente nel campo dichiarato per la chiave SFLRRN.
  - Se il ciclo di lettura non carica niente, si riemette la pagina ultima emessa in errore.
- Emissione a video dei dati accumulati: EXFMT del record C1 con condizionamento favorevole delle parole chiave SFLDSP e SFLDSPCTL.
- Viene mostrata la prima pagina del subfile o, se presente la chiave SFLRRN, la pagina che contiene il record S1 corrispondente.
- La visione della pagina successiva è ottenuta dal tasto ROLLUP, ma il caricamento della nuova pagina è un compito del programma applicativo. Facendo uso dell'ultima chiave letta per caricare l'ultimo record della pagina precedente, il pgm si posiziona sul file in esame con SETLL e continua la lettura. Salvo che sull'ultima pagina, dove è impedito. Quindi si storna al ciclo di lettura di database per caricare la nuova pagina.
- La visione della pagina precedente è ottenuta dal tasto ROLLDOWN, gestito automaticamente dal sistema, salvo che sulla prima pagina, dove è impedito.
- Alla luce di quanto detta, si deve dichiarare ROLLUP e non si deve dichiarare ROLLDOWN.

13.15 Subfile quadrato.

L'uso del quadrato è consigliabile quando occorre presentare a video dati aggiornatissimi.

Tuttavia il suo uso richiede uno schema di programmazione estremamente complesso rispetto a globale a graduale, giacché occorre annotare ad ogni uscita a video tutte le informazioni necessarie per rollare in avanti e all'indietro. E inoltre le scelte o le modifiche eseguite non possono essere accumulate su più pagine del subfile ma vanno risolte pagina per pagina.

Si puntualizza che è il sistema a comportarsi in maniera molto diversa rispetto a globale e graduale quando il compilatore riconosce il quadrato.

In particolare è permesso il condizionamento dei campi, assolutamente impedita sugli altri tipi di subfile.

Sono inoltre tollerati record di tracciato diverso fino al riempimento dello spazio riservato al corpo del subfile. Il primo record può, ad esempio essere lungo una riga e gli alri due righe. L'importante è non sfondare il corpo limitandosi a scrivere record condizionati in modo da non scrivere più righe di quelle ogni volta disponibili.

In un record di subfile, come in ogni altro record, i nomi di campo devono essere unici. Ne segue che ognuna dei vari sottorecord possibili avrà tutti i campi condizionati da un solo indicatore riservato. Se uno stesso contenuto di campo deve figurare più volte in più tracciati, verrà duplicato in caricamento in un campo diverso nel nome.

APPROFONDIMENTI

Esempio d'uso.

13.16 Riempimento del subfile quadrato.

Si elencano le istruzioni utili al caricamento del subfile quadrato.

- Si ipotizza l'uso di una vista logica ascendente letta in avanti durante le azioni conseguenti ai ROLLUP. Altre modalità si ricavano da questa con semplici e sistematici rovesciamenti della terminologia.
- La chiave della vista logica che permette la lettura dei dati deve essere univoca, cioè unica in senso stretto (UNIQUE) o virtualmente unica perché comprendente i campi di una chiave unica operante sul file fisico esaminato. L'univocità è necessaria perché i posizionamenti richiesti all'inizio di ogni ciclo di lettura siano precisi al record e non al gruppo di record. In sostanza, la pagina di subfile presentata a seguito di una richiesta di roll non deve presentare i record già visti nella pagina di partenza, né perderne tra una pagina e l'altra.
- Presentazione del video guida G1 contenente le informazioni di scelta del gruppo di record da caricare in elenco. Occorre soprattutto una chiave parziale della vista logica di accesso ai record di database.
- Pulizia totale del contenuto mediante WRITE del record C1 con condizionamento favorevole della parola chiave SFLCLR. Per finezza e per ingannare l'attesa, si può nell'occasione emettere anche le intestazioni del subfile accendendo la chiave di visualizzazione del controllo SFLDSPCTL. Azzeramento del contatore da usare nel seguito come numero relativo di record scrivendo S1.
- Si approntano due chiavi di posizionamento.
  - Una contenente la chiave unica del primo record della pagina. Inizializzata con la chiave parziale proveniente dal video guida G1.
  - Una contenente la chiave unica dell'ultimo record della pagina.
- Si scrivono due distinte routine di riempimento della pagina corrente di subfile.
  - Riempimento in avanti. Chiamata in ingresso a partire dalla chiave del video guida G1. Chiamata inoltre per eseguire la richiesta di ROLLUP.
  - Riempimento all'indietro. Chiamata per eseguire la richiesta di ROLLDOWN.
- La routine di caricamento in avanti esegue un ciclo di lettura del file di database del quale si vogliono presentare i record trascrivendoli nel subfile. Si fa uso della chiave parziale proveniente da G1 ma anche del posizionamento basato sui campi chiave accantonati per l'ultimo record presentato nella pagina di partenza.
  - Si esaminano i dati per decidere l'indicatore che condiziona il tracciato desiderato. L'indicatore scelto dice quante righe saranno occupate nel corpo. Se il numero di righe disponibili è inferiore al richiesto, si congeda l'emissione della pagina di subfile.
  - Preparazione a calcolo del numero relativo di record da emettere nel subfile. Di solito è un semplice incremento.
  - Trascrizione del record dai campi del record di database ai corrispondenti campi della riga di subfile.
  - Sbiancamento dell'eventuale campo scelta,
  - Scrittura del record S1 di subfile mediante WRITE.
  - Accantonamento del numero relativo di record ultimo caricato nel subfile.
  - Il ciclo continua fino al riempimento di una pagina per volta ch viene accodata elle pagine già riempite. Il subfile in memoria risulta quindi costituito da più pagine.
  - Ad ogni record letto e caricato nel subfile, occorre annotare l'ultima chiave letta dal database e l'ultimo relative record number caricato nel subfile.
  - Se il ciclo di lettura non carica niente, si riemette la pagina ultima emessa in errore.
- Emissione a video dei dati accumulati: EXFMT del record C1 con condizionamento favorevole delle parole chiave SFLDSP e SFLDSPCTL.
- Viene mostrata l'unica pagina del subfile.

- La visione della pagina successiva è ottenuta dal tasto ROLLUP che chiama la routine di caricamento in avanti.
- La visione della pagina precedente è ottenuta dal tasto ROLLDOWN che chiama la routine di caricamento all'indietro.
- Alla luce di quanto detta, si devono dichiarare sia ROLLUP che ROLLDOWN.

13.17 Subfile di visualizzazione con scelta.

Se sul subfile coricato ed emesso si è previsto un campo di scelta, subito dopo l'EXFMT che lo ha messo a video ed il test di uscita o di ritorno a video precedente, si eseguono dei cicli di esame del subfile per valutare la correttezza delle scelte e per provocarne l'esecuzione.

Si rinunci all'uso di VALUES sul campo scelta poiché le segnalazioni che ne scaturiscono avvengono al roll e non all'invio, quando vengono manifestati gli errori più complessi che possono essere individuati solo durante il calcolo.

Esiste un'istruzione che semplifica drasticamente il primo ciclo di lettura del subfile e che viene illustrata perché gli incauti ne fanno uso nelle logiche semplici e poco implementabili.

La READC (Read Changed) restituisce il primo record modificato dalla digitazione e, di seguito, gli altri ma contemporaneamente neutralizza lo stato di change del record letto che non viene più trovato ad un eventuale secondo giro. Si badi bene che il campo scelta di un certo record di subfile può essere digitato e poi sbiancato. La READC trova tutti i record digitati, anche se con un blank

Il rimedio alla disattivazione esiste: al primo giro di READC, su quei record S1 che si vogliono trovare anche al secondo giro, si esegue un UPDATE con accesa la chiave SFLNXTCHG. Poi, però, prima del secondo giro occorre senz'altro passare per il video che reimposta anche il posizionamento all'inizio, non altrimenti eseguibile.

Il rimedio è peggio del problema.

L'esplorazione delle scelte di un subfile va invece eseguita con un ciclo di CHAIN con numero relativo di record. Quando le macchine erano sottodimensionate, leggere tanti record di subfile era pesante. Era.

Eventualmente il primo giro, che esegue da 1 a riempimento il suo ciclo di lettura, può annotare il primo e l'ultimo da esaminare nei cicli successivi senza passare per il video.

Si vuole, insomma, dissuadere da pratiche apparentemente semplici e purtroppo diffuse ma sicuramente faticose e fallaci alla prima implementazione.

Esempio d'uso.

13.18 Subfile di immissione.

L'immissione eseguita da subfile ha un difetto gravissimo, tanto grave quanto sono numerosi i record presenti nel subfile al momento dell'invio di accettazione dei dati.

Se è vero che ogni errore deve essere segnalato, è chiaro che dopo aver immesso cento record, all'invio verranno segnalati per primi gli errori relativi ai primi record dell'elenco.

Se l'immissione è avvenuta dalla lettura di documenti, un errore segnalato sul primo record richiede una seconda manipolazione di un documento già sepolto da 99 altri.

Ecco perché merita immettere da subfile solo testi e mai dati. Il testo non richiede segnalazioni di errore e comunque non chiede il maneggio di molti documenti.

L'argomento quindi non verrà approfondito perché utile soltanto in programmi di gestione testi, raramente scritti in ambito applicativo gestionale.

Esempio d'uso.

14 File di stampa. (14R)

I file di stampa, allo stesso modo dei file di database, possono essere definiti sia internamente che esternamente al programma RPG.
La descrizione interna viene spiegata precocemente per evidenziare il parallelismo con la descrizione esterna. Volendo, potrà essere letta dopo il capitolo Specifiche RPG di emissione (20).

14.1 Spool.

Le stampe As/400 nascono sempre con l'intermediazione del supporto nativo di spool. La stampa in presa diretta su una stampante è possibile ma chi scrive non ci ha mai provato e non pensa che meriti farlo.
La gestione dei file di spool presenti nelle code di emissione è argomento a sé stante.
Mentre le stampe godono di opzioni grafiche non banali, la visualizzazione di un file di spool è in ambiente carattere e non mostra ciò che verrà stampato (non è WYSIWYW). L'unica via per vedere una stampa così come verrà stampata è la trasformazione in PDF, dove possibile, con una utility mirata e la sua visualizzazione con i pgm PC.

14.2 Stampa esterna.

Il file di stampa descritto esternamente è un oggetto che contiene tutte le definizioni necessarie ad ottenere una certa stampa.
Esso viene concepito prima ed autonomamente rispetto al programma che lo userà.
Il programma si limiterà a dichiarare il file di stampa, a riempire i campi menzionati nel printer file e a comandare la scrittura dei record. E' compito del printer file posizionare i campi in emissione, aggiungere le costanti, conosciute solo dal printer file, ed attribuire le caratteristiche ad ogni carattere stampato.
Le specifiche necessarie vengono scritte in un sorgente che viene compilato tramite il comando CRTPRTF (Create Printer File).
Dalla compilazione, dotata di lista di errori, come tutte le compilazioni da sorgente, nasce un oggetto (Printer File) di tipo *FILE e attributo PRTF.
Nel programma utilizzatore del file di stampa esterno vengono automaticamente dichiarati i record, i campi e gli indicatori di condizionamento presenti nel file di stampa. Il programma si limita a riempire i campi, accendere gli opportuni condizionamenti e comandare, in opportuna sequenza, l'emissione dei record per ottenere la stampa.

Sorgente della stampa esterna

I record, i campi, gli indicatori e le costanti del file di stampa vengono definiti tramite le DDS (Data Description Specification) nell'opportuno membro di tipo PRTF di un file sorgente. Per la codifica delle DDS relative ad un file di stampa si possono utilizzare almeno due metodi. Il primo è quello di andare in editazione del membro sorgente tramite il SEU (Source Entry Utility), l'altro è quello di utilizzare l'RLU (Report Layout Utility), spiegato in apposito capitolo.

Di seguito vengono riportate le DDS del file di stampa CLIP100P.

```
DP   AAN01N02N03T.Nome++++++RLun++TPdBRigColFunzioni+++++++++++++++++++++++++++++++
     A          R RCD001
     A                                      TEXT('Intestazione di tabulato')
     A                                      SKIPB(002)
     A                                     45'Stampa Anagrafico Clienti'
     A                                    121'Pagina'
     A                                    128PAGNBR
     A                                      EDTCDE(2)
     A            LIN01P       132A        1
     A                                      SPACEB(001)
     A                                     1'Cliente'
     A                                      SPACEB(001)
     A                                    +1'Ragione sociale cliente'
     A                                    +13'Località'
     A                                    +28'Indirizzo'
     A                                    +27'Pr'
     A            LIN02P       132A        1
     A                                      SPACEB(001)
DP   AAN01N02N03T.Nome++++++RLun++TPdBRigColFunzioni+++++++++++++++++++++++++++++++
     A          R RCD002
     A                                      TEXT('Dettaglio stampa')
     A            CODICL         7A        1
     A                                      SPACEB(001)
     A            RASOCL        35A       +1
     A            LOCACL        35A       +1
     A            INDICL        35A       +1
     A            PROVCL         2A       +1
DP   AAN01N02N03T.Nome++++++RLun++TPdBRigColFunzioni+++++++++++++++++++++++++++++++
     A          R RCD999
     A                                      TEXT('Ultimo record di stampa')
     A                                      SPACEB(002)
     A                                    50'*** Fine stampa ***'
```

L'esempio definisce un file di stampa con tre record.

RCD001 Intestazione di tabulato.
RCD002 Dettaglio stampa.
RCD999 Ultimo record di stampa.

Laddove si definiscono i record di stampa, si devono specificare i salti a riga prima o dopo avere effettuato la stampa e gli spazi prima o dopo avere effettuato la stampa. Sul record RCD001 si è deciso di saltare alla riga di stampa 2 e solo allora stampare.
Si noti che una riga può richiedere nell'ordine elencato tutti gli spazi e i salti possibili.

- Salto immediato SKIPB
- Spazio immediato SPACEB
- Spazio ritardato SPACEA
- Salto ritardato SKIPA

Oltre a spazi e salti a livello di record, si possono specificare anche spazi e salti a livello di campo. Nel primo caso il record di stampa giace su un'unica riga fisica. Nel seconda occupa più righe fisiche della stampa.
Per i file di stampa esterni si specifica la posizione iniziale del campo. Infatti il campo LIN01P, lungo 132 byte e contenente tutti caratteri "-" (una riga di trattini), viene stampato alla posizione 1. Peraltro è possibile evitare di specificare la posizione iniziale del campo in stampa bastando indicare quanti spazi devono intercorre tra un campo ed un altro. Nel record RCD002 ci si è limitati ad indicare che il campo RASOCL deve essere stampato ad 1 byte di distanza dal campo precedente CODICL.

Creazione dal sorgente della stampa esterna

Il comando di creazione del file di stampa esterno non designa soltanto quale sorgente prendere per creare l'oggetto, ma attribuisce al file di stampa generato un elevato numero di

caratteristiche che non hanno posto nel sorgente ma sono da considerare egualmente informazioni di tipo sorgente.
Ne segue la necessità di conservare il comando di creazione o in un sorgente apposito, detto allora stringa di creazione, o nel medesimo membro sorgente del file di stampa (come fa l' RLU) sotto forma di commento, eseguibile però come comando da apposito programma (l' RLU medesimo).
Di seguito il comando.
Decidere la forma e aggiungere parametri: tipo modulo, lunghezza, larghezza,...

```
CRTPRTF                                Create Printer File
      FILE(PRTLIB/PRTFILE)                   Printer Library/Printer File
      SRCFILE(SRCLIB/QDDSSRC)                Source Library/Source File
      SRCMBR(*FILE)                          Source Member
      SIZE(*NOMAX)                           Size
```

Segue una breve illustrazione delle singole parole chiave usate nella creazione del file di stampa.
Printer File Il nome del file di stampa da creare.
Printer Library Il nome della libreria destinata a contenere il file di stampa da creare.

CRTPRTF FILE(ObjLib/CLIP100F) SRCFILE(SrcLib/QDDSSRC) SRCMBR(*FILE) TEXT(*SRCMBRTXT)

Dichiarazione della stampa esterna nel programma RPG

Di seguito viene dichiarato un file di stampa descritto esternamente.

```
FX  FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++++
    Fclip100p  o    e             printer oflind(*in90)
```

Dove: clip100p =Nome del file di stampa.
O =Designazione del file, ovvero Output.
E =File descritto esternamente.
printer =Indica che è un file di stampa.
oflind(*in90) =L'indicatore di overflow è *IN90.

Emissione della stampa esterna nel programma RPG

L'istruzione RPG per emettere un record di stampa è WRITE.
Di seguito vengono riportate le istruzioni di calcolo.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Se è il caso, emette intestazione di tabulato.
C                   if        *in90
C                   write     rcd001
C                   setoff                                         90
C                   endif
     * Emette record di dettaglio.
C                   write     rcd002
     * A fine elaborazione, emette record di fine stampa.
Clr                 write     rcd999
```

14.3 Stampa interna.

Il file di stampa descritto internamente è un oggetto che non contiene tutte le definizioni necessarie ad ottenere una certa stampa. In particolare, esso viene generato senza fare uso di un sorgente ma solo del comando di creazione CRTPRTF (Create Printer File) con opportuni parametri.
Dalla creazione nasce un oggetto (Printer File) di tipo *FILE e attributo PRTF.
Nel programma utilizzatore del file di stampa interno, tramite apposite specifiche sorgente, vengono esplicitamente dichiarati i record, i campi e gli indicatori di condizionamento. Il programma riempie i campi, accende gli opportuni condizionamenti e comanda, in opportuna sequenza, l'emissione dei record per ottenere la stampa.

I file di stampa descritti internamente al programma RPG presuppongono l'esistenza dell'oggetto di tipo file (PRTF) sul sistema non necessariamente al momento della compilazione del programma, ma solo all'atto della esecuzione del programma stesso.
Nei programmi estemporanei, per semplificare, si fa spesso uso dei file di stampa precostituiti sul sistema. Ad esempio.

- QPRINT Fornito dal sistema per uso dell'utente.
- QSYSPRT Fornito dal sistema per uso proprio.

Tuttavia, all'interno di pacchetti con un minimo di qualità, per gestire al meglio le caratteristiche delle stampe conviene creare tanti file di stampa quanti sono i programmi di stampa del pacchetto, non importa se stampe interne o esterne. Questo evita di modificare il programma utilizzatore del file di stampa quando solo quella stampa cambia caratteristiche. Non solo, ma questo comportamento rende omogeneo il trattamento di tutti i file di stampa, sia esterni che interni.
La tecnica di definizione delle emissioni di record di stampa interna è quella fondata sulle emissioni comandate esplicitamente dal calcolo tramite l'operazione EXCEPT. Tale tecnica è prevalsa perché molto simile all'unica possibile per le emissioni di record di stampa esterni tramite l'operazione WRITE.
La tecnica alternativa, d'uso più antico e che non viene qui illustrata, prevede vari tipi di emissioni non comandabili direttamente dal calcolo ma eseguite dal ciclo RPG, quando applicabile, nei vari tempi di emissione che il ciclo prevede: prima pagina, intestazione, dettaglio, totale.

Creazione della stampa interna

Il comando di creazione del file di stampa interno non usa alcun sorgente ma attribuisce al file di stampa generato un elevato numero di caratteristiche.
Ne segue la necessità di conservare il comando di creazione in un sorgente apposito, detto allora stringa di creazione.
Di seguito il comando.
Decidere la forma e aggiungere parametri: tipo modulo, lunghezza, larghezza,...

```
CRTPRTF                                Create Printer File
      FILE(PRTLIB/PRTFILE)                   Printer Library/Printer File
      SRCFILE(*NONE)                         No source
      SIZE(*NOMAX)                           Size
      RCDLEN(RCDLEN)                         Record length
```

Segue una breve illustrazione delle singole parole chiave usate nella creazione del file di stampa.
Printer File Il nome del file di stampa da creare.
Printer Library Il nome della libreria destinata a contenere il file di stampa da creare.

CRTPRTF FILE(ObjLib/CLIP100F) SRCFILE(SrcLib/QDDSSRC) SRCMBR(*FILE) TEXT(*SRCMBRTXT)

Dichiarazione della stampa interna nel programma RPG

Di seguito viene dichiarato un file di stampa descritto internamente.

```
F   FFilename++IPEASFRlen+LKlen+AIDevice+.Par.chi.++++++++++++++++++++++++++++++
    Fqsysprt   o    f  132        printer oflind(*in90)
```

Dove: qsysprt =Nome del file di stampa generico.
O =Designazione del file, ovvero Output.
F =File descritto internamente.
132 =Lunghezza del record in byte.
printer =Indica che è un file di stampa.
oflind(*in90) =L'indicatore di overflow è *IN90.

Sorgente della stampa interna

Tutti i record e i campi del file di stampa vengono definiti nelle specifiche di tipo **O** (Output) in fondo al programma RPG. Si noti che la stampa ottenibile dal sorgente seguente è esattamente

la stessa ottenibile dal file di stampa descritto esternamente illustrato nel paragrafo precedente.

```
O   ONomefile++DF..N01N02N03Excnam++++B++A++Sb+Sa+.............................
P   O.............N01N02N03Field+++++++++YB.End++PConstant/editword/DTformat++
    Oqsysprt   e            rcd001         1 02
    O                       lin01p             132
    O          e            rcd001         1
    O                                          69 'Stampa Anagrafico Clienti'
    O                                         126 'Pagina'
    O                       pagenbr       2    132
    O          e            rcd001         1
    O                       lin02p             132
     *
    O          e            rcd001         1
    O                                           7 'Cliente'
    O                                          31 'Ragione sociale cliente'
    O                                          52 'Località'
    O                                          89 'Indirizzo'
    O                                         118 'Pr'
     *
    O          e            rcd002         1
    O                       codicl               7
    O                       rasocl            +  1
    O                       locacl            +  1
    O                       indicl            +  1
    O                       provcl            +  1
     *
    O          e            rcd999      2  1
    O                                          64 '*** Fine stampa ***'
```

L'esempio definisce un file di stampa con tre record.

RCD001 Intestazione di tabulato.

RCD002 Dettaglio stampa.

RCD999 Ultimo record di stampa.

Laddove si definiscono i record di eccezione, si devono specificare i salti a riga prima o dopo avere effettuato la stampa e gli spazi prima o dopo avere effettuato la stampa. Sul record RCD001 si è deciso di saltare alla riga di stampa 2, di stampare e, infine, di spaziare una riga. Diversamente dai file di stampa esterni, in quelli interni si specifica la posizione finale del campo e non quella iniziale. Infatti il campo LIN01P (lungo 132 byte) che contiene il carattere "-" per stampare la riga di trattini, viene stampato alla posizione 132. Peraltro è possibile evitare di specificare la posizione finale del campo in stampa, ed il record RCD002 ne è un chiaro esempio. In questo caso ci si è limitati ad indicare che il campo RASOCL deve essere stampato ad 1 byte di distanza dal campo precedente CODICL.

Emissione della stampa interna nel programma RPG

L'istruzione RPG per emettere un record di stampa è EXCEPT.

Di seguito vengono riportate le istruzioni di stampa.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Se è il caso, emette intestazione di tabulato.
    C                   if        *in90
    C                   except    rcd001
    C                   setoff                                         90
    C                   endif
     * Emette record di dettaglio.
    C                   except    rcd002
     * A fine elaborazione, emette record di fine stampa.
    Clr                 except    rcd999
```

14.1 Report Layout Utility.

L' RLU è di modesto utilizzo perché scrivere file di stampa da zero con tale strumento provoca un numero di complicazioni eccessivo rispetto alla semplicità dell'altro strumento disponibile: il SEU.

Così lo si usa soltanto per allineare i campi di una stampa già abbozzata col SEU.
Per una breve guida all' RLU, si consulti l'apposito capitolo (37).

15 Program status data structure (PSDS). (15C)

15.1 Dichiarazione.

Il programma RPG in corso di esecuzione conosce alcune informazioni su se stesso tramite la struttura dati informativa sullo stato del programma così definita.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
D PSDS           ESDS
```

Dove:

D	=Specifica dati.
PSDS	=Struttura dati informativa sul file definita nel programma come segue.
E	=Esternamente descritta.
S	=Contiene informazioni sullo stato del programma.
DS	=Struttura dati.

Il contenuto della struttura dati è fornito dal sistema senza ulteriori istruzioni.

15.2 Definizione dettagliata dei campi.

La struttura dati PSDS è riportata di seguito ma è meglio descritta nei sorgenti di esempio www.neroni.it/Scaricabili/JPSDS.txt .

```
Program Status Data Structure for RPG ILE.
 1   1  10 PS_PROC     A  10    Procedure name
 2  11  15 PS_STATUS   S   5 5  Status code
 3  16  20 PS_PRVSTS   S   5 5  Previous status code
 4  21  28 PS_SRCLST   A   8    RPG IV source listing line number
 5  29  36 PS_ROUTINE  A   8    RPG IV routine in which error occurred
 6  37  39 PS_PARMS    S   3 3  Parameters number
 7  40  42 PS_EXCTYP   A   3    Exception type
 8  43  46 PS_EXCNBR   A   4    Exception number
 9  47  50 PS_RESERV   A   4    Reserved
10  51  80 PS_WRKARA   A  30    Messages work area
11  81  90 PS_PGMLIB   A  10    Program library
12  91 170 PS_EXCDTA   A  80    Retrieved exception data
13 171 174 PS_EXCIDE   A   4    Exception cause of RNX9001
14 175 190 PS_UNUSED1  A  16    Unused
15 191 198 PS_ENTDAT8  A   8    Entrance date in *date fmt
16 199 200 PS_CENTURY  S   2 2  Entrance century
17 201 208 PS_LSTFIL   A   8    Last error file
18 209 243 PS_LSTFILS  A  35    Last error informations
19 244 253 PS_JOB      A  10    Job name
20 254 263 PS_USER     A  10    User
21 264 269 PS_JOBNBR   S   6 6  Job number
22 270 275 PS_ENTDAT6  S   6 6  Entrance date in udate fmt
23 276 281 PS_RUNDAT6  S   6 6  Run date in udate fmt
24 282 287 PS_RUNTIM   S   6 6  Run time in hhmmss fmt
25 288 293 PS_CPLDAT   A   6    Compile date in udate fmt
26 294 299 PS_CPLTIM   A   6    Compile time in hhmmss fmt
27 300 303 PS_CPLLVL   A   4    Compiler level
28 304 313 PS_SRCFIL   A  10    Source file
29 314 323 PS_SRCLIB   A  10    Source library
30 324 333 PS_SRCMBR   A  10    Source member
31 334 343 PS_PROGRAM  A  10    Program containing procedure
32 344 353 PS_MODULE   A  10    Module containing procedure
33 354 429 PS_UNUSED2  A  76    Unused
```

Si nota che la Program Status Data Structure del vecchio RPG differisce da quella per l' RPG Ile. In particolare, i primi dieci caratteri della vecchia sono il nome del programma che nella nuova sono in posizione 334-343.

Quando si converte un sorgente da vecchio RPG a Ile con il comando CVTRPGSRC è opportuno cambiare la PSDS e il nome del campo dal quale di preleva il nome del programma, anche se un pgm generato con CRTBNDRPG contiene alla 1 e alla 343 la stessa informazione fino a quando non si rinomina l' RPG Ile compilato. Il nome nuovo lo si trova allora soltanto alla 343.

Esiste anche la possibilità di definire la struttura dati internamente al programma, ma per questo si rimanda al manuale. Si nota tuttavia che la pratica più diffusa è la definizione interna dei soli campi necessari.

Si ricorda anche la seguente forma che dà accesso soltanto ad alcuni dei campi.

*PROGRAM

*PARMS

15.3 Utilizzo delle informazioni contenute.

L'informazione più frequentemente usata è il nome programma (PS_PROGRAM) che permette, ad esempio, di conoscere il nome corrente del programma (e dichiararlo magari a video) anche dopo che l'oggetto eseguibile compilato ha subito una ridenominazione.

Frequentemente è utile anche sapere quanti parametri ha ricevuto il run corrente del programma (PS_PARMS) per condizionare quelle operazioni che userebbero altrimenti i parametri non ricevuti, provocando un errore non correggibile.

16 Specifiche RPG di File esterni e interni. (16C)

L' RPG accede ai dati sempre tramite oggetti precostituiti

- o completamente descritti al di fuori dell' RPG (file descritti esternamente)
- o dettagliati soltanto all'interno dell' RPG che li usa (file descritti internamente).

Il file di tipo esterno è il più frequente e il più coerente con la logica del sistema.

Altri tipi di oggetti diversi dai file (aree dati, code dati, …) contengono dati e vengono usati in modo diverso con comandi mirati e scopi specifici.

Per i file database si usano sia interni che esterni.

Per i file video si usano solo esterni.

Per i file stampa, che sono solo di emissione, si usano sia interni che esterni.

Per i file esterni la lettura e la scrittura avvengono tramite le specifiche di immissione e di emissione automaticamente dichiarate nel programma al seguito della definizione del file.

Per i file interni la lettura e la scrittura avvengono tramite apposite specifiche di immissione e di emissione, alle quali si rimanda, minutate internamente all' RPG.

16.1 Definizione file di database esterni.

Un file di tipo esterno di database da leggere tramite chiavi in modo libero viene così definito.

```
FX  FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++++
    FESTER01L  IF   E           K DISK
```

Dove:

	F	=Specifica di file
	ESTER01L	=Nome del file
	I	=Input
	F	=Full procedural
	E	=External described
	K	=Keyed
	DISK	=File di database su disco

Un file di tipo esterno di database solo da scrivere viene così definito.

```
FX  FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++++
    FESTER00F  O    E             DISK
```

Dove:

	F	=Specifica di file
	ESTER00F	=Nome del file
	O	=Output
	E	=External described
	DISK	=File di database su disco

16.2 Definizione file di database interni.

Il file database di tipo interno è poco frequente ed è usato solitamente nell'ambito dei programmi di utilità.

Un file di tipo interno di database da leggere tramite numero relativo di record viene così definito.

```
F   FFilename++IPEASFRlen+LKlen+AIDevice+.Par.chi.++++++++++++++++++++++++++++++
    FINTER00F  IF   F  150        DISK
```

Dove:

	F	=Specifica di file
	INTER01L	=Nome del file
	I	=Input
	F	=Full procedural
	F	=Internal described
	150	=Lunghezza record
	DISK	=File di database su disco

Un file di tipo interno di database da leggere tramite chiave viene così definito.

```
F   FFilename++IPEASFRlen+LKlen+AIDevice+.Par.chi.++++++++++++++++++++++++++++++
    FINTER01L  IF   F  150     3AIDISK
```

Dove: F =Specifica di file
INTER01L =Nome del file
I =Input
F =Full procedural
F =Internal described
150 =Lunghezza record
3 =Lunghezza della chiave
AI =Chiave alfanumerica (Alphameric Index)
DISK =File di database su disco

16.3 Definizione file video esterni.

Un file video da leggere e scrivere in modo libero viene così definito.

```
FX  FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++++
    FVIDEO00V  CF   E            WORKSTN
```

Dove: F =Specifica di file
VIDEO00V =Nome del file
C =Combined (Input/Output)
F =Full procedural
E =External described
WORKSTN =File su video

16.4 Definizione file di stampa esterni.

Un file di tipo esterno di stampa viene così definito.

```
FX  FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++++
    FSTAMP00P  O    E            PRINTER
```

Dove: F =Specifica di file
STAMP00P =Nome del file
O =Output
E =External described
PRINTER =File di stampa

16.5 Definizione file di stampa interni.

Un file di tipo interno di stampa viene così definito.

```
F   FFilename++IPEASFRlen+LKlen+AIDevice+.Par.chi.++++++++++++++++++++++++++++++
    FSTAMP00P  O    F  198       PRINTER
```

Dove: F =Specifica di file
STAMP00P =Nome del file
O =Output
F =Internal described
198 =Lunghezza record
PRINTER =File di stampa

16.6 Apertura controllata.

Quando l' RPG si avvia, apre automaticamente i file dichiarati nel programma, a meno che non sia richiesta l'apertura controllata.

Riprendendo un esempio precedente, l'apertura controllata viene così richiesta.

```
FX  FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++++
    FESTER01L  IF   E          K DISK    USROPN
```

Nel calcolo sarà poi necessario aprire esplicitamente il file.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                     OPEN      ESTER01L
```

Apertura controllata o no, un file può essere esplicitamente chiuso nel seguente modo.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                     CLOSE     ESTER01L
```

L'apertura ritardata può servire quando, in mancanza di un programma cl chiamante, occorre eseguire un reindirizzamento del file di database un attimo prima di aprirlo, tramite di solito un Override Data Base File eseguito tramite QCMDEXC.

In altro esempio, la chiusura di un printer file congeda lo spool file di una stampa prima della chiusura dell' RPG che la esegue. La successiva riapertura può generare un altro spool file durante lo stesso run dell' RPG.

Anche il file video di un RPG–finestra può giovarsi di una chiusura e di una immediata riapertura per prelevare un'immagine nuova della base della finestra.

16.7 Ridenominazione dei campi mediante prefisso.

Per i file di tipo esterno dichiarati nel programma vengono automaticamente dichiarati anche i campi presenti nella descrizione esterna dei file. Ad esempio: CAMPO1, CAMPO2. Se però altri file, dichiarati nello stesso RPG, portano i medesimi nomi di campo che non vanno confusi con quelli definiti nel primo, si può porre rimedio rinominando globalmente i campi del primo file con un prefisso.

```
FX  FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++++
    FESTER01L  IF   E           K DISK    PREFIX(XX)
```

In questo caso i campi sopra esemplificati verranno conosciuti dentro il programma come XXCAMPO1, XXCAMPO2.

C'è un modo alternativo di rinominare i singoli campi tramite le specifiche di immissione a cui si rimanda.

16.8 File information data structure.

Spesso è utile conoscere alcune informazioni di servizio che il programma RPG gestisce circa ciascuno dei file aperti. Per evidenziare quelle su di un file, occorre richiederlo come segue.

```
FX  FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++++
    FESTER01L  IF   E           K DISK    INFDS(FIDS)
```

Dove: FIDS =Struttura dati informativa sul file definita nel programma come segue.

```
D   DName++++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
    D FIDS           E DS
```

Dove: D =Specifica dati.
FIDS =Struttura dati informativa sul file definita nel programma come segue.
E =Esternamente descritta.
DS =Struttura dati.

La struttura dati FIDS è riportata di seguito ma è meglio descritta nei sorgenti di esempio www.neroni.it/Scaricabili/JFIDS.txt .

Si nota che la File Information Data Structure del vecchio RPG non differisce da quella per l' RPG Ile.

```
  1   8 FIDS01  A   8       ...
  9   9 FIDSOP  A   1       File open
 10  10 FIDS02  A   1       ...
 11  15 FIDSST  A   5       Status code
 16  45 FIDS03  A  30       ...
 46  52 FIDSER  A   7       Error code
 53  82 FIDS04  A  30       ...
 83  92 FIDSFI  A  10       File
 93 102 FIDSLB  A  10       Library
103 128 FIDS05  A  26       ...
129 138 FIDSMB  A  10       Member
139 151 FIDS06  A  13       ...
152 153 FIDSFL  B   2  4    Forms lines or workstation lines
154 155 FIDS07  A   2       ...
156 159 FIDSNR  B   4  9    Record quantity in input file at open
160 161 FIDSAT  A   2       Access type
162 162 FIDS08  A   1       ...
163 163 FIDSSO  A   1       Source file flag
164 187 FIDS09  A  24       ...
188 189 FIDSOL  B   2  4    Overflow line
190 260 FIDS10  A  71       ...
261 270 FIDSRE  A  10       Last record format processed
271 272 FIDSDC  A   2       Device class
273 282 FIDS11  A  10       ...
283 286 FIDSLE  B   4  9    First read record length
287 366 FIDS12  A  80       ...
367 368 FIDSCL  B   2  4    Current line
369 369 FIDS13  A   1       ...
370 371 FIDSLP  A   2       Line/ position
372 396 FIDS14  A  25       ...
397 400 FIDSRR  B   4  9    Relative record number in data member
```

17 Specifiche RPG di Immissione dati per file esterni ed interni. (17R)

Le specifiche di immissione dati si utilizzano.

- Per effettuare la ridenominazione di singoli campi di un file.
- Per specificare le rotture di livello (Lx) oppure per definire i controlli di sequenza (Mx) quando si utilizza il ciclo RPG,.
- Per definire il tracciato record di un file descritto internamente al programma.

17.1 File Esterni.

Ridenominazione campi.

La necessità di ridenominare i campi in lettura si presenta soprattutto perché l' RPG trasporta subito i dati letti dal buffer di input, non direttamente accessibile come in COBOL, nei campi in memoria e la ridenominazione permette di non sporcare i campi omonimi di un altro record.

Di seguito viene riportato un esempio di ridenominazione di alcuni campi per il file SALDI01L.

```
FX  FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++++
     * Saldi quantità articoli.
    Fsaldi01l  if   e           k disk
IX  IRcdname+++....Ri.....................................................
    Isaldi
JX  I.............Ext-field+..................Field.........L1M1..PlMnZr......
    I              salq01                     salq01x
    I              salq02                     salq02x
    I              salq03                     salq03x
    I              salq04                     salq04x
    I              salq05                     salq05x
```

I campi SALQnn all'interno del programma RPG si chiameranno SALQnnX.

Ridenominazione campi in schiera.

E' possibile anche effettuare la ridenominazione dei campi di un file riempiendo una schiera contestualmente alla lettura. Di seguito l'esempio.

```
FX  FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++++
     * Saldi quantità articoli.
    Fsaldi01l  if   e           k disk
D   DName++++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
     * Schiera quantità ovvero campi del file SALDI01L.
    D qt              s              7  2 dim(5)
IX  IRcdname+++....Ri.....................................................
     * Ridenominazione campi per il file SALDI00F.
    Isaldi
JX  I.............Ext-field+..................Field.........L1M1..PlMnZr......
    I              salq01                     qt(01)
    I              salq02                     qt(02)
    I              salq03                     qt(03)
    I              salq04                     qt(04)
    I              salq05                     qt(05)
```

Ad ogni operazione di lettura di un record nel file SALDI01L, la schiera QT di 5 elementi sarà automaticamente riempita con i valori dei campi SALQnn.

Rotture di livello.

Come spiegato nell'opportuno capitolo (22), l' RPG mette a disposizione la possibilità di elaborare un file, definito "primario" sulla specifica "F", mediante una lettura automatica sequenziale detta "Ciclo RPG". Per potere sfruttare gli automatismi più interessanti del ciclo, si devono utilizzare le Rotture di livello (Lx) che si definiscono sulle specifiche di immissione.

```
FX  FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++++
     * Archivio statistico clienti.
    Fstati00f  ip   e            disk
```

```
IX  IRcdname+++....Ri.....................................................
     * Definizione delle rotture di livello per il file STATI00F.
    Istati
JX  I.............Ext-field+.................Field+++++++++L1M1..PlMnZr......
    I                                        continente    L4
    I                                        nazione       L3
    I                                        regione       L2
    I                                        provincia     L1
```

Il file STATI00F è designato input primario ed alcuni suoi campi sono definiti come campi di rottura da L4 a L1.

Si presume che i record siano preventivamente ordinati sui campi indicati nella sequenza da L4 a L1: continente, nazione, regione, provincia.

Si definisce "gruppo di controllo" di L4 l'insieme dei record che portano uno stesso valore nel campo L4. Nel file esistono tanti gruppi di L4 quanti sono i valori del campo continente presenti nei record del file.

Si definisce "gruppo di controllo" di L3 l'insieme dei record che portano uno stesso valore nel campo L4 e uno stesso valore nel campo L3. Nel file esistono tanti gruppi di L3 quanti sono i valori della coppia di campi continente—nazione presenti nei record del file. Ovviamente il gruppo di L3 è più piccolo del gruppo di L4

Preme osservare che lo stesso valore di Nazione presente su due record non li fa appartenere allo stesso gruppo di controllo di L3 se non è uguale anche il continente: Continente Eurasia e Nazione Georgia designano un gruppo di controllo diverso da quello definito da Continente America e Nazione Georgia.

Per L2 si considera la terna continente—nazione—regione e le considerazioni sono facilmente derivabili dalle precedenti.

Stesso discorso per L1, considerando la quaterna continente—nazione—regione—provincia.

I ragionamenti esposti mostrano infine che, se rompe L4, rompono anche L3, L2 e L1. Se rompe L3 (e non L4) rompono anche L2 e L1. Generalizzando, la rottura più alta provoca anche tutte le rotture inferiori.

E' indispensabile che il file sia ordinato in maniera che tutti i record di ciascun gruppo siano sottoposti alla lettura del ciclo RPG nell'ordine detto: continente—nazione—regione—provincia. L'ordinamento fa in modo che i record di un gruppo di controllo qualunque (L4, L3, L2, L1) siano elaborati tutti consecutivamente escludendo così che un record appartenente ad un gruppo si trovi isolato dagli altri dello stesso gruppo. L'incongruenza tra ordinamento e rotture di livello scatena rotture che il programma RPG rileva ed elabora ma che sono un vero e proprio errore logico.

Dal pasticcio, se l'ordinamento è ascendente su tutti i campi, ci salva la richiesta di controllo della sequenza che è bene affiancare alla definizione delle rotture.

```
JX  I.............Ext-field+.................Field+++++++++L1M1..PlMnZr......
    I                                        continente    L4M4
    I                                        nazione       L3M3
    I                                        regione       L2M2
    I                                        provincia     L1M1
```

Il controllo di sequenza è un cadavere dell'elaborazione "Matching Record", che qui non trattiamo, utile nelle elaborazioni batch ai tempi delle schede.

Il ciclo RPG legge a suo carico i record ordinati e per ciascuno esegue il calcolo di dettaglio. Durante il calcolo di dettaglio sono automaticamente disponibili gli speciali indicatori L4, L3, L2 ed L1 che, se accesi, permettono di riconoscere il primo record di ogni gruppo.

Prima di leggere ed elaborare il successivo record, il ciclo RPG mette a disposizione un tempo speciale di calcolo, quello di totale, che viene eseguito subito dopo il calcolo di dettaglio e

durante il quale sono accesi i soliti indicatori a significare però che l'ultimo record elaborato è anche l'ultimo di uno dei gruppi di controllo. Al tempo di totale si accede indicando L4, L3, L2 o L1 nelle posizioni 7-8 della specifica di calcolo.

Facendo uso degli indicatori di rottura a calcolo di dettaglio e di totale è possibile corredare una lista bruta di record con intestazioni di gruppo di L4, L3, L2, L1 e di totali di gruppo di L1, L2, L3, L4. Prima l'intestazione di L4. Ultimo il totale di L4.

Similmente su una lista di totali di L1 senza i dettagli.

Non è ovviamente sempre necessario stampare tutti i livelli delle intestazioni e dei totali ma, per decidere quali, occorre valutare il riempimento dei dati e le necessità dell'utente.

Il grande pregio del ciclo è che le rotture sono sempre tutte disponibili senza altri sforzi.

Normalmente si stampano i dati in maniera scalata, con bordi a sinistra più grandi per i livelli più bassi.

Su una riga di totale ci può stare anche il totale di livello più alto bastando condizionare il totale con l'indicatore corrispondente. E' previsto il taglio delle mani per quei modernisti che, non volendo usare gli indicatori di livello in emissione, si affannano in calcolo a riempire variabili con costanti. Se la stampa è esterna gli indicatori di livello, non menzionabili sulle DDS del printer file, vanno duplicati in altrettanti indicatori numerici, utilizzabili senza problemi..

Di seguito le specifiche CL che precedono la chiamata del programma RPG.

```
/* Ridirige archivio di input.                                        */
             OVRDBF     FILE(STATI00F) TOFILE(NOMELIB/STATI00F) +
                          SHARE(*YES) SEQONLY(*YES 5000)
/* Ordina il file di input.                                           */
             OPNQRYF    FILE((NOMELIB/STATI00F *FIRST)) OPTION(*ALL) +
                          KEYFLD((CONTINENTE) (NAZIONE) (REGIONE) +
                          (PROVINCIA) (COMUNE))
```

La prima istruzione ridirige l'archivio STATI00F in modo che venga letto sequenzialmente a blocchi di 5.000 record per volta e specifica che la prossima apertura del file potrà essere condivisa con altri programmi.

La seconda istruzione esegue l'apertura del file STATI00F tramite OPNQRYF che ne ordina opportunamente il contenuto secondo i campi chiave specificati. Si noti che i primi 4 campi chiave specificati sull' OPNQRYF sono i medesimi che costituiscono Rottura di livello nel successivo programma RPG.

Di seguito le specifiche CL che seguono la chiamata del programma RPG.

```
/* Chiude il file di input.                                           */
             CLOF       FILE(STATI00F)
/* Annulla il redirizzamento del file di input.                       */
             DLTOVR     FILE(STATI00F)
```

Si deve precisare che la tecnica delle rotture di livello era usata in illo tempore per batch complessi e che la transazionalità tipica dell'interattivo l'ha resa quasi completamente inutile.

Tuttavia le rotture di livello automatiche sono ancora molto utili nelle stampe batch, non egualmente sparite, e nei batch di generazione dei file da trasferire ad altri pacchetti.

La tecnica alternativa di utilizzo delle rotture di livello senza ricorso al ciclo è faticosissima quando i gruppi di controllo sono più di uno e, nel tempo, ho visto soluzioni macchinose e quasi sempre fallose praticate per modernismo malinteso.

Naturalmente, la tecnica delle rotture di livello è utile, oltre che per le stampe di elenchi, anche per programmi che generano archivi di totale a scopo statistico, magari da passare a Excel o ad Access.

ESEMPIO

17.2 File Interni.

```
     .*..1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
I    IFilename++SqNORiPos1+NCCPos2+NCCPos3+NCC.................................
J    I.......................Fmt+SPFrom+To+++DcField+++++++++L1M1FrPlMnZr......
```

Definizione tracciato.

L' RPG conosce il tracciato di un file descritto esternamente tramite la sola dichiarazione del file nell'apposita specifica di file (F).
Per i file descritti internamente, invece, la specifica F serve unicamente ad indicare al programma RPG il nome del file e la lunghezza del record
Il programma RPG deve dichiarare al suo interno quanto serve per identificare ogni singolo campo in quanto a posizione, dimensioni e caratteristiche.
Per chiarire meglio il concetto, all'interno del programma RPG si dovranno specificare i campi che costituiscono il tracciato record del file.
Nel vecchio RPG le specifiche erano analoghe alla definizione dei sottocampi di una struttura dati definita internamente al programma.
Di seguito vengono riportate le specifiche di immissione con le quali si definiscono i campi di un record per un file descritto internamente.

Esempio di input interno

Si coglie l'occasione per raccomandare di non fare uso di archivi senza tracciato salvo che nei seguenti casi:

- Programmi di utilità che, dovendo manipolare tracciati infiniti, li devono riconoscere con meccanismi più complessi.
- Programmi di utilità che leggono ed interpretano stampe, solitamente di sistema, per ricavarne informazioni. Prima che le Api traditrici (dell'interfaccia unica) fossero, l'unico modo per reperire informazioni che non si trovavano nei vari comandi di Retrieve era stampare su file e rileggere per estrarre. Le prime Api furono più veloci di questo giro del fumo. Oggi le Api sono invece spesso più lente in esecuzione e,comunque, più faticose da capire e da usare.

Un esempio di utility di lettura interna di stampe su file è www.neroni.it/Scaricabili/JSPOL.txt.

18 Specifiche RPG Dati. (18C)

Se il programma RPG usa file descritti esternamente, la maggior parte dei campi viene dichiarata automaticamente all'interno del programma in conseguenza della specifica F di definizione del file. Per altro, i campi letti dai file descritti internamente sono definiti nelle specifiche di immissione, alle quali si rimanda.

Le specifiche D (dati) permettono invece di definire i campi di cui l' RPG necessita in aggiunta ai campi appartenenti ai file.

Altri modi di definire campi aggiuntivi si realizzano tramite il campo del risultato di una specifica di calcolo o tramite l'istruzione *LIKE DEFINE. Per questo si vedano le specifiche di calcolo.

18.1 Definizione di Costanti.

Una costante alfanumerica viene così definita.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
D CstAlf          C                   'TENNIS'
```

Dove: D =Specifica di definizione dati.
CstAlf =Nome della costante alfanumerica.
C =Specifica di definizione di costante.
'TENNIS' =Valore assegnato alla costante alfanumerica.

Una costante numerica viene così definita.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
D CstNum          C                   12345,67
```

Dove: D =Specifica di definizione dati.
CstNum =Nome della costante numerica.
C =Specifica di definizione di costante.
1234,56 =Valore assegnato alla costante numerica.

Si nota che la costante non è modificabile nel corso del programma.

18.2 Definizione di Campi.

Un campo alfanumerico inizializzato a blank viene così definito.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
D FldAlf          S             15
```

Un campo alfanumerico inizializzato ad un valore a piacere viene così definito.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
D FldAlf          S             15    INZ('SCACCHI')
```

Dove: D =Specifica di definizione dati.
FldAlf =Nome del campo alfanumerico.
S =Specifica di definizione di campo.
15 =Lunghezza del campo alfanumerico.
'SCACCHI' =Valore iniziale assegnato al campo alfanumerico.

Un campo numerico inizializzato a zero viene così definito.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
D FldNum          S              7  2
```

Un campo numerico inizializzato ad un valore a piacere viene così definito.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
D FldNum          S              7  2 INZ(76543,21)
```

Dove: D =Specifica di definizione dati.
FldNum =Nome del campo numerico.
S =Specifica di definizione di campo.
7 =Lunghezza del campo numerico.
2 =Numero di decimali del campo numerico.

76543,21 =Valore iniziale assegnato al campo numerico.

Qualunque campo può essere definito sulla base di un altro, come nell'esempio seguente.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
    D FldNew          S                   LIKE(FldBas)
```

Dove: D =Specifica di definizione dati.
FldNew =Nome del campo.
S =Specifica di definizione di campo.
LIKE(FldBas) =Richiesta di qualità e dimensione del campo come il campo base FldBas, comunque definito altrove.

Naturalmente è possibile usare contemporaneamente entrambe le opzioni viste.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
    D FldNew          S                   LIKE(FldBas) INZ('MERINGA')
```

Dove: FldBas =Campo di riferimento definito nel relativo esempio precedente.

18.3 Definizione di Schiere.

Qualunque campo di qualunque tipo può essere multiplo e viene allora chiamato Schiera (in altri linguaggi: Vettore).

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
    D ArrAlf          S             15    DIM(10)
```

Dove: ArrAlf =Nome di schiera alfanumerica.
DIM(10) =Numero di ricorrenze presenti nella schiera.

Il numero di ricorrenze può essere un nome di costante con zero decimali, usato magari più volte per garantire che più schiere abbiano lo stesso numero di elementi e che tale numero sia globalmente e facilmente modificabile cambiando solo il valore della costante. Ad esempio.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
    D CstMax          C                   10
    D ArrAlf1         S             15    DIM(CstMax)
    D ArrAlf2         S              2    DIM(CstMax)
    D ArrNum          S              3  0 DIM(CstMax)
```

Dove: DIM(CstMax) =Numero di ricorrenze presenti nelle schiere.

18.4 Definizione di Strutture dati descritte internamente.

I campi di un gruppo possono essere definiti in modo da occupare in memoria uno spazio contiguo ed essere gestibili come i sottocampi di un insieme, detto allora struttura dati, sempre alfanumerico e definito come segue.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
    D DtaStr          DS
    D  FldAlf1                    15
    D  FldAlf2                     3
    D  FldNum1                     5S 0
```

Dove: DtaStr =Nome della struttura dati.
FldAlf1 =Sottocampo alfanumerico 1 lungo 15. (caratteri 01-15)
FldAlf2 =Sottocampo alfanumerico 2 lungo 3. (caratteri 16-18)
FldNum1 =Sottocampo numerico segnato 1 lungo 5,0. (caratteri 19-23)

Si noti che i campi sono definiti tramite la semplice dimensione e figurano giustapposti nella struttura dati.

La stessa zona di memoria può essere ridefinita più volte con nomi diversi, stavolta tramite la posizione iniziale e finale. La modifica di uno dei campi implica la modifica dei campi sovrapposti. Un esempio di ridefinizione si ottiene accodando la seguente specifica all'esempio precedente.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
    D  FldAlf3                 7     18
```

Dove: FldAlf3 =Sottocampo alfanumerico 3 lungo 12. (caratteri 07-18)

Occorre comunque rispettare la qualità dei campi numerici ad evitare che, manipolandone l'area di memoria tramite altri sottocampi, si provochino errori nei dati decimali che il sistema riscontrerebbe al primo uso del campo numerico.

18.5 Definizione di Strutture dati descritte esternamente.

Si supponga di avere sul sistema un file descritto esternamente di nome StrEst che, in linea di massima, non sia destinato a contenere record di dati e, quindi, nemmeno membri. Ad esempio.

```
Struttura Dati STREST – Parametri per la stampa degli ordini.
 1   1   1 PRT1        A   1    Esegui stampa 1.
 2   2   2 PRT2        A   1    Esegui stampa 2.
 3   3   8 DATBEG      S   8 0  Data inizio ccyymmdd.
 4   9  14 DATEND      S   8 0  Data fine ccyymmdd.
```

La definizione di una struttura dati dentro l' RPG può allora anche essere ottenuta dal file fisico, utilizzabile a tale scopo nei seguenti due modi.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++
D StrEst         E DS
```

Dove: StrEst =Nome, interno all' RPG, della struttura dati coincidente col nome del file esternamente descritto usato come prototipo.
E =External described.
DS =Data structure.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++
D Str            E DS                  EXTNAME(StrEst)
```

Dove: Str =Nome interno all' RPG della struttura dati.
E =External described.
DS =Data structure.
EXTNAME(StrEst) =Nome del file esternamente descritto usato come prototipo.

In conseguenza di una delle due definizioni sopra esemplificate, nel programma sarà definito l'insieme dei campi dichiarati nel file prototipo, riunito in una struttura dati che può coincidere col nome del file prototipo (primo esempio) o godere di un proprio nome interno all' RPG (secondo esempio).

Anche le strutture dati esterne sopportano l'aggiunta di righe interne per la ridefinizione dei campi. Vale lo stesso esempio dato poco sopra per le ds interne.

Come per i file dichiarati esternamente nelle specifiche F, è possibile rinominare i campi di una struttura dati descritta esternamente o tramite un prefisso che cambia tutti i nomi, o tramite una riga apposita che cambia un singolo nome.

Ridenominazione tramite prefisso.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++
D StrEst         E DS                  PREFIX(RN)
```

Dove: PREFIX(RN) =Costante con cui prefissare i campi della struttura dati.

In questo caso, nell' RPG i campi presenti nel prototipo (nell'esempio: PRT1, PRT2, DATBEG, DATEND) verranno conosciuti nell' RPG con il prefisso RN (nell'esempio: RNPRT1, RNPRT2, RNDATBEG, RNDATEND).

Ridenominazione di singoli campi.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++
D StrEst         E DS
D  RNDATBEG      E                     EXTFLD(DATBEG)
```

Dove: RNDATBEG =Nome del campo all'interno dell' RPG.
E =External described.
EXTFLD(DATBEG) =Nome del campo come presente nella definizione del file.

In questo caso, nell' RPG i campi non rinominati verranno conosciuti col nome presente nel prototipo (nell'esempio: PRT1, PRT2, DATEND) mentre i campi rinominati verranno conosciuti nell' RPG con i nomi modificati (nell'esempio: RNDATBEG).

18.6 Inizializzazione di una Struttura dati.

All'inizio di un programma RPG le variabili in esso contenute vengono inizializzate conformemente al loro tipo: blank nei campi alfanumerici e zeri nei numerici. Le strutture dati, sia interne che esterne, vengono però considerate come campi alfanumerici e ricevono blank che cadono quindi anche nei sottocampi numerici provocando errori al successivo utilizzo dei sottocampi.

Tuttavia, durante la definizione della struttura dati, se ne può richiedere l'inizializzazione campo per campo nel modo seguente.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
D StrEst         E DS                    INZ
```

Dove: INZ =Richiesta di inizializzazione campo per campo.

Se i campi di una struttura dati interna o i campi interni di una ds esterna si sovrappongono, vale l'inizializzazione del campo elencato per ultimo.

Resta salva la possibilità di inizializzare la ds anche in calcolo mediante la CLEAR con le stesse regole.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                     clear     StrEst
```

18.7 Definizione di Strutture dati a ricorrenze multiple.

L'area di memoria di una struttura dati, comunque definita, può contenere anche più ricorrenze di valori, come nell'esempio.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
D StrEst         E DS                    OCCURS(20)
```

Dove: OCCURS(20) =Numero delle ricorrenze della struttura dati.

In luogo del numero di ricorrenze si può usare anche il nome di una costante, ad esempio la medesima costante CstMax definita più sopra.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
D StrEst         E DS                    OCCURS(CstMax)
```

Dove: OCCURS(CstMax) =Numero delle ricorrenze della struttura dati.

Nel programma, inizialmente, il nome di un campo facente parte di una struttura dati punta alla prima ricorrenza. Dopo un'istruzione OCCUR, il puntamento passa sulla ricorrenza scelta.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     Index         OCCUR                   StrEst
```

Dove: C =Specifica di calcolo.
Index =Numero della ricorrenza scelta per la struttura dati.
OCCUR =Operazione di scelta ricorrenza.
StrEst =Nome della struttura dati definita con ricorrenze multiple.

Naturalmente Index può essere un numero, un nome di costante numerica o un campo numerico, tutti con zero decimali.

18.8 Definizione di Schiere affiancate all'interno di Strutture dati.

Allo scopo di affiancare in memoria gli elementi di una schiera con i corrispondenti di altre e di riunire gli elementi omologhi in un maxielemento talché che si possa gestire la corrispondente maxischiera, si esemplificano tre schiere (AffA, AffB. AffC) come sottodefinizioni di una maxischiera (Aff) di 20 elementi, giacente a sua volta in una struttura dati (AffDs).

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
    D AffDs           DS
    D Aff                            12    DIM(20)
    D AffA                            5    OVERLAY(Aff:1)
    D AffB                            5S 0 OVERLAY(Aff:6)
    D AffC                            2    OVERLAY(Aff:11)
```

Dove: AffA 5 OVERLAY(Aff:1) =Sottodefinisce ogni elemento della schiera AffA come sottostringa del corrispondente maxielemento della schiera Aff a partire dalla posizione 1 per 5 caratteri.

Aff =Schiera di 20 elementi alfanumerici lunghi 12.

AffA =Schiera di 20 elementi alfanumerici lunghi 5.

AffB =Schiera di 20 elementi numerici lunghi 5,0.

AffC =Schiera di 20 elementi alfanumerici lunghi 2.

La definizione di tali schiere si giustifica con la necessità di sortare (mettere in ordine) una schiera e tutte le parallele con un'unica operazione SORTA. Così, nel seguente esempio, le schiere sopra definite possono essere sortate tutte secondo la schiera AffB.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
    C                   SORTA                   AffB
```

Oppure secondo la schiera AffC.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
    C                   SORTA                   AffC
```

Oppure secondo la maxischiera Aff.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
    C                   SORTA                   Aff
```

Dove: SORTA =Operazione di messa in ordine.

19 Specifiche RPG di Calcolo e di Calcolo esteso. (19R)

Le specifiche di calcolo C e di calcolo esteso CX costituiscono la parte più importante di un programma RPG. Sono le specifiche che di fatto costituiscono e determinano il comportamento del programma.

Per curiosità, si nota che un programmino di stampa tramite il ciclo RPG, che emetta una stampa interna mediante emissioni di Intestazione (H), Dettaglio (D) e Totale (T), può fare completamente a meno delle specifiche di calcolo. Non ce ne impicciamo perché tali emissioni sono meno flessibili delle emissioni di Eccezione (E) richieste in calcolo mediante l'operazione EXCEPT.

19.1 Operazioni di spostamento dati.

Questi codici operativi consentono di eseguire lo spostamento dei dati tra i campi.

MOVE Muove a destra
MOVEL Muove a sinistra

MOVE muove il contenuto del campo datore (fattore1) nel campo ricevente (risultato) spostando il contenuto a partire dal byte più a destra e riempiendo il ricevente byte per byte a partire da destra.

MOVEL muove il contenuto del campo datore (fattore2) nel campo ricevente (risultato) spostando il contenuto a partire dal byte più a sinistra e riempiendo il ricevente byte per byte a partire da sinistra.

Si noti che l'estensione (P), facoltativa su entrambe le istruzioni, pulisce preliminarmente il campo ricevente riempiendolo di *BLANK se alfanumerico o di *ZERO se numerico. Se non è necessario altrimenti, MOVE(P) e MOVEL(P) sono da considerare le istruzioni da preferire.

Un campo numerico deve ricevere sempre e solo dati numerici validi anche se provenienti da un campo alfanumerico, pena un errore di sistema.

Se il datore è più corto del ricevente, una parte dei dati originali presenti nel ricevente non viene alterata.

Dati per gli esempi

Ciascuno degli esempi seguenti si deve immaginare eseguito a partire dai sottoelencati valori iniziali e quindi senza influenze dei vari esempi tra loro. Subito sotto l'istruzione è riportato il valore assunto dal campo del risultato dopo l'esecuzione dell'istruzione medesima. La presente nota è valida per tutti gli esempi del capitolo corrente.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
    D Alfa3           s               6    inz('987')
    D Alfa6           s               6    inz('PIAZZA')
    D Alfa8           s               8    inz('BISCOTTO')
    D Alfa9           s               9    inz('777888999')
    D Nume3           s               3  0 inz(789)
    D Nume6           s               6  0 inz(123456)
```

Esempi

Move tra campi alfanumerici.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                     MOVE      Alfa8          Alfa6
C                     MOVE(P)   Alfa8          Alfa6
 *                                                    "SCOTTO"
C                     MOVEL     Alfa8          Alfa6
C                     MOVEL(P)  Alfa8          Alfa6
 *                                                    "BISCOT"
C                     MOVE      Alfa6          Alfa8
 *                                                    "BIPIAZZA"
C                     MOVEL     Alfa6          Alfa8
 *                                                    "PIAZZATO"
C                     MOVE(P)   Alfa6          Alfa8
 *                                                    "  PIAZZA"
C                     MOVEL(P)  Alfa6          Alfa8
 *                                                    "PIAZZA  "
```

Move da campi alfanumerici a campi numerici con esito di "Errore nei dati decimali".

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                     MOVE      Alfa8          Nume6
C                     MOVEL     Alfa8          Nume6
C                     MOVE(P)   Alfa8          Nume6
C                     MOVEL(P)  Alfa8          Nume6
```

Move da campi alfanumerici a campi numerici con esito favorevole.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                     MOVE(P)   Alfa9          Nume6
 *                                                    888999
C                     MOVEL(P)  Alfa9          Nume6
 *                                                    777888
C                     MOVE      Alfa3          Nume6
 *                                                    123987
C                     MOVEL     Alfa3          Nume6
 *                                                    987456
C                     MOVE(P)   Alfa3          Nume6
 *                                                    000987
C                     MOVEL(P)  Alfa3          Nume6
 *                                                    987000
```

Move tra campi numerici.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                     MOVE      Nume6          Nume3
C                     MOVE(P)   Nume6          Nume3
 *                                                    456
C                     MOVEL     Nume6          Nume3
C                     MOVEL(P)  Nume6          Nume3
 *                                                    123
C                     MOVE      Nume3          Nume6
 *                                                    123789
C                     MOVE(P)   Nume3          Nume6
 *                                                    000789
C                     MOVEL     Nume3          Nume6
 *                                                    789456
C                     MOVEL(P)  Nume3          Nume6
 *                                                    789000
```

19.2 Operazioni matematiche.

Questi codici operativi consentono di eseguire le operazioni matematiche tra campi numerici.

ADD *Aggiunge*

Somma il valore del campo specificato sul fattore1 al valore del campo specificato sul fattore2 e lo pone nel campo specificato sul risultato.

Se il fattore1 non viene espresso, viene assunto uguale al risultato e la forma dell'operazione viene detta "breve".

```
D   DName++++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
    D Nume3a          s               3  0 inz(25)
    D Nume3b          s               3  0 inz(10)
    D Nume3c          s               3  0 inz(40)
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
    C     Nume3a        ADD       Nume3b        Nume3c
     *                                                  035
    C                   ADD       Nume3b        Nume3c
     *                                                  050
```

SUB Sottrae

Sottrae il valore del campo specificato sul fattore2 al valore del campo specificato sul fattore1 e lo pone nel campo specificato sul risultato.

Se il fattore1 non viene espresso, viene assunto uguale al risultato e la forma dell'operazione viene detta "breve".

```
D   DName++++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
    D Nume3a          s               3  0 inz(25)
    D Nume3b          s               3  0 inz(10)
    D Nume3c          s               3  0 inz(40)
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
    C     Nume3a        SUB       Nume3b        Nume3c
     *                                                  015
    C                   SUB       Nume3b        Nume3c
     *                                                  030
```

MULT Moltiplica

Moltiplica il valore del campo specificato sul fattore1 per il valore del campo specificato sul fattore2 e lo pone nel campo specificato sul risultato.

Se il fattore1 non viene espresso, viene assunto uguale al risultato e la forma dell'operazione viene detta "breve".

```
D   DName++++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
    D Nume3a          s               3  0 inz(25)
    D Nume3b          s               3  0 inz(10)
    D Nume3c          s               3  0 inz(40)
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
    C     Nume3a        MULT      Nume3b        Nume3c
     *                                                  250
    C                   MULT      Nume3b        Nume3c
     *                                                  400
```

DIV Divide

Divide il valore del campo specificato sul fattore1 per il valore del campo specificato sul fattore2 e lo pone nel campo specificato sul risultato.

Se il fattore1 non viene espresso, viene assunto uguale al risultato e la forma dell'operazione viene detta "breve".

```
D   DName++++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
    D Nume3a          s               3  0 inz(20)
    D Nume3b          s               3  0 inz(10)
    D Nume3c          s               3  0 inz(40)
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
    C     Nume3a        DIV       Nume3b        Nume3c
     *                                                  002
    C                   DIV       Nume3b        Nume3c
     *                                                  004
```

È importante verificare sempre che il divisore (fattore2) sia diverso da zero, altrimenti il programma andrà in errore tentando di dividere per zero. Nel campo della matematica e della

fisica il controllo è dovuto. Nel campo gestionale si assume sempre che il quoziente sia zero. Peccato che l' RPG non permetta, ad esempio, un'estensione operazione che risolva il problema in maniera più semplice di quella qui illustrata.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     Divisore      IFNE      *ZERO
C     Dividendo     DIV       Divisore      Quoziente
C                   ELSE
C                   CLEAR                   Quoziente
C                   ENDIF
```

MVR Muove il resto della divisione (Move reminder)

Posto subito dopo una istruzione DIV, trascrive il resto della divisione in un campo a sé stante.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
D Dividendo       s               3  0 inz(25)
D Divisore        s               3  0 inz(10)
D Quoziente       s               3  0
D Resto           s               3  0
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     Dividendo     DIV       Divisore      Quoziente
C                   MVR                     Resto
 * Quoziente = 002
 * Resto     = 005
```

SQRT Calcola radice quadrata (Square root)

Calcola la radice quadrata del valore del campo sul fattore2 e lo pone nel campo sul risultato.

Naturalmente un fattore2 negativo provoca un errore di sistema.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
D Quadrato        s               3  0 inz(9)
D Radice          s               3  0
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                   SQRT      Quadrato      Radice
 *                                                003
```

Si noti che il campo del risultato può essere lo stesso campo specificato sul fattore2.

19.3 Operazioni sulle schiere.

MOVEA Muove schiera (Move array)

Il codice operativo MOVEA tratta le schiere alfanumeriche come se fossero campi senza riguardo ai singoli elementi. Per capirne il funzionamento, si pensi ognuna delle schiere menzionate sul fattore2 o sul risultato come se fosse un unico grande campo ottenuto accostando tra loro tutti gli elementi della schiera. Fatta questa premessa, la MOVEA si comporta verso tali campi come se fosse una MOVEL.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
D Sc2x4           S               2    DIM(4) CTDATA PERRCD(1)
D Sc3x6           S               3    DIM(6) CTDATA PERRCD(1)
D Ca7             S               5    INZ('1234567')
D Ca5             S               5    INZ('ABCDE')
**   ....*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
** Sc2x4
11!
22!
33!
44!
** Sc3x6
AAA!
BBB!
CCC!
DDD!
EEE!
FFF!
```

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Muove la prima schiera nella seconda e presenta il risultato.
C                     MOVEA     Sc2x4          Sc3x6
     * "112"  "233"  "44C"  "DDD"  "EEE"  "FFF"
     * Muove la seconda schiera nella prima e presenta il risultato.
C                     MOVEA     Sc3x6          Sc2x4
     * "AA"  "AB"  "BB"  "CC"
     * Muove il primo campo nella seconda schiera e presenta il risultato.
C                     MOVEA     Ca7            Sc3x6
     * "123"  "456"  "7CC"  "DDD"  "EEE"  "FFF"
     * Muove il secondo campo nella prima schiera e presenta il risultato.
C                     MOVEA     Ca5            Sc2x4
     * "AB"  "CD"  "E3"  "44"
```

Se una delle schiere porta un indice di partenza, il campone su cui agisce la MOVEA inizia con quell'elemento e termina con l'ultimo mentre gli elementi precedenti non sono coinvolti.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Muove la prima schiera nella seconda e presenta il risultato.
C                     MOVEA     Sc2x4(2)       Sc3x6(3)
     * "AAA"  "BBB"  "223"  "344"  "EEE"  "FFF"
     * Muove la seconda schiera nella prima e presenta il risultato.
C                     MOVEA     Sc3x6(3)       Sc2x4(2)
     * "11"  "CC"  "CD"  "DD"
     * Muove il primo campo nella seconda schiera e presenta il risultato.
C                     MOVEA     Ca7            Sc3x6(3)
     * "AAA"  "BBB"  "123"  "456"  "7EE"  "FFF"
     * Muove il secondo campo nella prima schiera e presenta il risultato.
C                     MOVEA     Ca5            Sc2x4(2)
     * "11"  "AB"  "CD"  "E4"
```

LOOKUP Cerca nella schiera (Look up)

Il codice operativo LOOKUP consente di ricercare il valore specificato sul fattore1 negli elementi della schiera specificata sul fattore2. Se non diversamente indicato, la ricerca comincerà dal primo elemento della schiera.

Per semplicità si esamina solo il caso di ricerca per uguale rimandando al manuale RPG per le altre poco probabili ricerche.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
    D Schiera         S              2    DIM(6) CTDATA PERRCD(1)
    D Parola          S              2    INZ('XY')
```

```
**   ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
    ** Schiera
    CD!
    X5!
    ZZ!
    XY!
    BW!
```

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Cerca Parola in Schiera e accende l'indicatore 50 perché trova.
C     Parola        LOOKUP    Schiera                                  50
     * Cerca Costante in Schiera e non accende l'indicatore 50 perché non trova.
C      'NN'         LOOKUP    Schiera                                  50
```

Si può anche utilizzare un indice per conoscere il numero di elemento di schiera che contiene il valore ricercato. L'indice è un campo numerico intero che deve comunque essere preventivamente impostato con l'indirizzo dell'elemento di schiera dal quale si desidera che la ricerca abbia inizio.

Con gli stessi valori del caso precedente, nell'esempio seguente la ricerca comincia dal primo elemento della schiera, l'indicatore risultante *IN50, posto nella posizione di uguale, dopo l'operazione lookup risulterà acceso poiché sarà trovato il contenuto di Parola nella Schiera e l'indice X conterrà l'indirizzo 4 dell'elemento nel quale sarà ritrovata la parola cercata.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Cerca Parola in Schiera a partire dal primo elemento,
     * accende l'indicatore 50 perché trova
     * e mette X uguale all'indice trovato (X=4).
C                   Z-ADD     1             X                 3 0
C     Parola        LOOKUP    Schiera(X)                            50
```

SORTA Ordina schiera (Sort array)

Questo codice consente di effettuare l'ordinamento degli elementi della schiera secondo il tipo di ordinamento indicato sulla specifica di definizione della schiera. L'ordinamento può essere ascendente o discendente.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++
     * Elenco sigle provinciali.
D PRV             S              2    DIM(8) CTDATA PERRCD(8) ASCEND
     * Elenco sigle nazionali.
D NAZ             S              3    DIM(7) CTDATA PERRCD(7) DESCEND
```

```
**   ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
** PRV - Elenco sigle provinciali
MIVASOCOMNLCLOPV!
** PRV - Elenco sigle nazionali
CH F  I  E  D  SLOHR !
```

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Mette in ordine alfabetico ascendente l'elenco sigle provinciali.
C                   SORTA     PRV
     * Risulta "CO" "LC" "LO" "MI" "MN" "PV" "SO" "VA"
     * Mette in ordine alfabetico discendente l'elenco sigle nazionali.
C                   SORTA     NAZ
     * Risulta "SLO" "I  " "HR " "F  "  "E  " "D  " "CH "
```

19.4 Operazioni di condizionamento.

Il programma RPG esegue le istruzioni di calcolo una dopo l'altra nell'ordine in cui si presentano. Provocano eccezione le istruzioni di condizionamento e di flusso.

Questi codici operativi si utilizzano per condizionare l'esecuzione di un gruppo di istruzioni nel programma. Per semplicità gli esempi che seguono utilizzeranno dei numeri anziché dei campi; questo solo per rendere immediati i concetti.

IFxx Se vero esegue

ENDIF Fine di Se vero esegue

Il codice operativo IFxx compara il fattore1 e il fattore2 facendo uso dell'operatore relazionale xx e l'esito della comparazione condiziona l'esecuzione delle istruzioni successive. All'istruzione IF deve sempre corrispondere una istruzione di chiusura ENDIF. Se la condizione posta sulla IF viene soddisfatta, le istruzioni sino alla specifica ENDIF vengono eseguite.

Salvo quanto si dirà per la specifica di calcolo estesa, l'istruzione IF è sempre completata da uno dei seguenti operatori relazionali.

EQ uguale a
NE diverso da
LT minore di
LE minore od uguale a
GT maggiore di
GE maggiore od uguale a

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     1             IFEQ      1
 * Istruzioni eseguite perché condizione soddisfatta.
 *                  ..........
C                   ENDIF
```

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     1             IFEQ      2
 * Istruzioni non eseguite perché condizione non soddisfatta.
 *                  ..........
C                   ENDIF
```

Si possono nidificare svariate operazioni IF/ENDIF facendo in modo che a ciascuna IF corrisponda sempre una istruzione di chiusura ENDIF. All'estrema destra delle istruzioni è annotato il livello di annidamento in forma simile a quella della lista di compilazione.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     CAMPO1        IFEQ      CAMPO2                                  B01
 *                  ..........                                         01
C     CAMPO3        IFNE      CAMPO4                                  B02
 *                  ..........                                         02
C                   ENDIF                                             E02
 *                  ..........                                         01
C     CAMPO5        IFGT      CAMPO6                                  B02
 *                  ..........                                         02
C     CAMPO7        IFLE      CAMPO8                                  B03
 *                  ..........                                         03
C                   ENDIF                                             E03
 *                  ..........                                         02
C                   ENDIF                                             E02
 *                  ..........                                         01
C                   ENDIF                                             E01
```

Naturalmente l'annidamento realizza una logica complessa da tenere sotto controllo con molta attenzione. Riteniamo che un annidamento oltre il terzo livello risulti poco facile da capire. In tal caso si frammenterà il ragionamento facendo un oculato uso delle subroutine. Nelle subroutine, infatti, l'annidamento riparte visivamente da zero, facilitando la comprensione della logica.

ELSE Altrimenti esegue

Questa istruzione può essere utilizzata solo ed esclusivamente all'interno di un gruppo di istruzioni IF/ENDIF. La ELSE crea una alternativa alle condizioni non soddisfatte specificate sulla IF.

Se la IF è soddisfatta vengono eseguite le istruzioni poste tra la IF e la ELSE.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     1             IFEQ      1
 * Istruzioni eseguite perché condizione soddisfatta.
 *                  ..........
C                   ELSE
 * Istruzioni non eseguite perché condizione soddisfatta.
 *                  ..........
C                   ENDIF
```

Se invece la IF non è soddisfatta vengono eseguite le istruzioni poste tra la ELSE e la ENDIF.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     1             IFEQ      2
 * Istruzioni non eseguite perché condizione non soddisfatta.
 *                  ..........
C                   ELSE
 * Istruzioni eseguite perché condizione non soddisfatta.
 *                  ..........
C                   ENDIF
```

ANDxx E se vero

ORxx O se vero

Per esprimere i condizionamenti complessi, all'istruzione IF occorre aggiungere le istruzioni AND e OR subito dopo la IF.

Anche le istruzioni AND e OR sono sempre abbinate ad uno degli operatori relazionali elencati per l'istruzione IF.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Condizione soddisfatta.
C     1             IFEQ      1
C     2             ANDGT     1
     * Istruzioni eseguite.
     *                .........
C                    ENDIF
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Condizione non soddisfatta.
C     1             IFEQ      2
C     2             ANDEQ     1
     * Istruzioni non eseguite.
     *                .........
C                    ENDIF
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Condizione non soddisfatta.
C     1             IFEQ      2
C     2             ORLT      1
     * Istruzioni non eseguite.
     *                .........
C                    ENDIF
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Condizione soddisfatta.
C     1             IFEQ      1
C     2             AMDEQ     1
C     2             ORGT      1
     * Istruzioni eseguite.
     *                .........
C                    ENDIF
```

Da notare che ogni volta che compare una istruzione OR, il gruppo di condizioni precedentemente specificate termina ed un nuovo gruppo di condizioni comincia.

SELECT Sceglie

WHENxx Quando vero esegue

OTHER Altrimenti esegue

ENDSL Fine di Sceglie

Il codice operativo WHEN condiziona l'esecuzione delle istruzioni successive ed è perciò molto simile al codice IF.

Esso è tuttavia utilizzabile solo nell'ambito dei codici operativi SELECT/ENDSL per realizzare quel che si può chiamare "IF multiplo".

A fronte di una istruzione SELECT deve sempre corrispondere una istruzione ENDSL. Subito dopo una SELECT è necessaria una WHEN che condiziona l'esecuzione del successivo gruppo di istruzioni che si concludono subito prima di un'altra WHEN, della OTHER o della ENDSL.

All'interno di un gruppo SELECT/ENDSL si possono specificare quante si vogliono operazioni WHEN.

Nell'ambito di un gruppo SELECT/ENDSL la prima condizione WHEN che viene soddisfatta è anche l'unica che verrà eseguita o, per meglio dire, le istruzioni sottostanti alla condizione WHEN saranno eseguite. Anche nel caso in cui più condizioni WHEN fossero soddisfatte, solo la prima di esse verrà eseguita.

Salvo quanto si dirà per la specifica di calcolo estesa, anche l'istruzione WHEN è sempre abbinata ad uno degli operatori relazionali elencati per l'istruzione IF.

Al seguito di una WHEN si possono utilizzare anche le istruzioni ANDxx e ORxx con le stesse regole segnalate per l'istruzione IF,

Nel seguente caso solo la 3° condizione è soddisfatta e l'indicatore *IN50 risulterà spento. Lo spegnimento preliminare dell'indicatore è necessario per renderne ininfluente lo stato precedente (e perché la SELECT è priva della OTHER).

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                     SETOFF                                        50
C                     SELECT
C     1               WHENEQ    2
C                     SETON                                         50
C     1               WHENGT    2
C                     SETON                                         50
C     1               WHENLT    2
C                     SETOFF                                        50
C                     ENDSL
```

Nel caso successivo entrambe le condizioni sono soddisfatte ma soltanto la prima di esse viene eseguita; pertanto l'indicatore *IN50 risulterà acceso.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                     SETOFF                                        50
C                     SELECT
C     1               WHENLT    2
C                     SETON                                         50
C     1               WHENEQ    1
C                     SETOFF                                        50
C                     ENDSL
```

All'interno di un gruppo SELECT/WHEN/WHEN.../ENDSL, un altro codice operativo può essere utilizzato: OTHER. Si può specificare questo codice operativo solo dopo avere specificato almeno una condizione WHEN e, dopo OTHER, non ci possono essere più altre WHEN prima di ENDSL. Il codice OTHER è simile al codice operativo ELSE della sequenza IF/ELSE/ENDIF. Ovvero, in alternativa alle condizioni WHEN, se non soddisfatte, esegue le istruzioni comprese tra OTHER ed ENDSL.

La sequenza sarà quindi SELECT/WHEN/WHEN.../OTHER/ENDSL, dove OTHER è facoltativo ed è obbligatoria una sola WHEN.

La presenza della OTHER garantisce che almeno uno dei gruppi di istruzioni all'interno del gruppo SELECT/ENDSL sarà eseguito.

Si nota che i due casi meno complessi sono meglio risolti dalle sequenze di IF.

SELECT/WHEN/.../OTHER/.../ENDSL = IF/../ELSE/.../ENDIF

SELECT/WHEN/.../ENDSL = IF/.../ENDIF

Nel caso seguente la condizione posta sull'istruzione WHEN non viene soddisfatta e perciò viene eseguita l'istruzione tra OTHER e ENDSL. L'indicatore *IN50 risulterà spento.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                     SELECT
C     1               WHENEQ    2
C                     SETON                                         50
C                     OTHER
C                     SETOFF                                        50
C                     ENDSL
```

19.5 Operazioni di flusso.

DO Esegue

ENDDO Fine di Esegue

LEAVE Abbandona

ITER Ricicla

La sequenza DO/ENDDO permette di raggruppare ed eventualmente ripetere in maniera controllata il gruppo di istruzioni compreso tra DO ed ENDDO.

Il codice operativo DO è l'operatore di flusso per eccellenza. Lo si utilizza per eseguire dei cicli (loop) di elaborazione sia sui file di database che su schiere e/o stringhe di dati.
L'esempio mostra un doppio ciclo di elaborazione dei valori di una schiera. Il contenuto iniziale della schiera viene fornito tramite le specifiche apposite da posizionare in fondo al sorgente del programma. Non sono forniti esplicitamente gli ultimi tre valori vuoti.

```
D   DName++++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
     * Elenco città.
    D CITTA           S             20    DIM(10) CTDATA PERRCD(1) ASCEND
**   ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
    ** CITTA - Elenco città
    COMO               !
    BELLAGIO           !
    CARATE URIO        !
    MENAGGIO           !
    CERMENATE          !
    ERBA               !
    ASSO               !
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Cambia in valore alto gli elementi bianchi dell'elenco città.
    C                   DO        10            X                3 0
    C     CITTA(X)      IFEQ      *BLANK
    C                   MOVEL(P)  *HIVAL   CITTA(X)
    C                   ENDIF
    C                   ENDDO
     * Mette in ordine alfabetico l'elenco città.
     * Gli elementi contenenti i valori alti si posizionano in fondo.
    C                   SORTA     CITTA
     * Ripristina i valori bianchi.
    C                   DO        10            X                3 0
    C     CITTA(X)      IFEQ      *HIVAL
    C                   CLEAR               CITTA(X)
    C                   ENDIF
    C                   ENDDO
```

Per completezza si riporta il contenuto della schiera dopo l'elaborazione elencando anche gli elementi vuoti.

```
**   ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
    ASSO               !
    BELLAGIO           !
    CARATE URIO        !
    CERMENATE          !
    COMO               !
    ERBA               !
    MENAGGIO           !
                       !
                       !
                       !
```

L'esempio successivo mostra un ciclo di lettura di un file di database con chiavi.

```
FX  FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++++
     * File da elaborare.
    FANAGR01L  IF   E           K DISK
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Si posiziona prima della chiave più bassa sul file.
    C     *LOVAL        SETLL     ANAGR
     * Esegue infinite volte. Inizio.
    C                   DO        *HIVAL
     *    Legge un record dal file.
    C                   READ      ANAGR                                  50
     *    Se non ci sono altri record, abbandona.
    C   50              LEAVE
     *    Elabora il record.
     *                  ..........
     * Esegue infinite volte. Fine.
    C                   ENDDO
```

Sul fattore2 si può specificare il numero di volte che il ciclo DO dovrà essere eseguito (limite finale). Può essere una costante numerica, una variabile numerica od il valore speciale *HIVAL. Se non specificato, il sistema assume 1. Sul fattore1 si può specificare un valore (limite iniziale) da cui incominciare il conteggio del numero di volte che il ciclo DO dovrà essere eseguito. Se non specificato nulla, il sistema assume 1.
Esempio di un ciclo di esecuzione per 30 volte.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                     DO        30
 * Esegue 30 volte.
 *                    ..........
C                     ENDDO
```

Esempio di un ciclo di esecuzione per 16 volte.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     15              DO        30
 * Esegue 16 volte.
 *                    ..........
C                     ENDDO
```

Sul campo del risultato si può specificare una variabile numerica. Questo indice assume al primo ciclo il valore espresso nel fattore1 del DO. Al ciclo successivo avviene l'incremento di una unità dell'indice. Eseguito il ciclo con l'indice uguale al fattore2 del DO, l'iterazione termina. Lo schema è utile quando si opera con le schiere o per effettuare un ciclo di analisi dei record di subfile video.
Esempio di un ciclo di esecuzione per 16 volte con indice.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     15              DO        30            X                 3 0
 * Esegue 16 volte.
 *                    ..........
C                     ENDDO
```

Durante l'esecuzione del primo ciclo, l'indice X varrà 15, incrementando di 1 ad ogni ciclo, sino ad arrivare al valore 30 durante l'esecuzione dell'ultimo ciclo.
Sul fattore2 del codice operativo ENDDO, si può specificare un valore di incremento utilizzato per il conteggio durante l'esecuzione del ciclo. Se non specificato nulla, il sistema assume come valore 1. Normalmente non si specifica alcun valore.
PURA FOLLIA
Uno dei casi in cui si specifica è però un valore negativo (-1). E' una tecnica che si utilizza quasi esclusivamente per operare sulle schiere, per analizzare il contenuto delle schiere leggendone il contenuto partendo dall'ultimo elemento al primo.
Esempio di lettura degli elementi di schiera dall'ultimo al primo. Schiera di 3 elementi.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     3               DO        3             X                 3 0
 * Esegue 3 volte.
 *                    ..........
C                     ENDDO     -1
```

In questo caso si è specificato di eseguire un ciclo di DO per 3 volte (fattore 2), si è specificato il limite iniziale (fattore 1) con il valore 3. Si è specificata la variabile X che verrà incrementata ciclo dopo ciclo. Si è specificato il valore di incremento –1. Di seguito analizziamo, ciclo per ciclo, l'esecuzione: alla prima esecuzione del ciclo X vale 3, al secondo X vale 2, al terzo ed ultimo ciclo X vale 1.
Il codice operativo ITER può essere utilizzato solo nell'ambito di un ciclo DO/ENDDO. L'effetto di questo codice operativo è che il programma riprende l'esecuzione con l'istruzione immediatamente successiva al codice operativo DO. Viene utilizzato, ad esempio, nel ciclo di lettura di un file di database quando si devono effettuare delle selezioni sui record. Nel caso in cui il record corrente non debba essere processato, si utilizza il codice operativo ITER per riprendere l'esecuzione del programma sulla istruzione di lettura.

Esempio di lettura di un file di database usando la sequenza DO/LEAVE/ITER/ENDDO.

```
     FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++++
      * File da elaborare.
     FANAGR01L  IF   E           K DISK
C    CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
      * Si posiziona prima della chiave più bassa sul file.
     C     *LOVAL        SETLL     ANAGR
      * Esegue infinite volte. Inizio.
     C                   DO        *HIVAL
      *    Legge un record dal file.
     C                   READ      ANAGR                                  50
      *    Se non ci sono altri record, abbandona.
     C   50              LEAVE
      *    Se il record è annullato, ricicla.
     C     A1ANNU        IF        'A'
     C                   ITER
     C                   ENDIF
      *    Elabora il record.
      *                   ..........
      * Esegue infinite volte. Fine.
     C                   ENDDO
```

Il codice operativo LEAVE può essere utilizzato solo nell'ambito di un ciclo DO/ENDDO. L'effetto di questo codice operativo è che il programma riprende l'esecuzione con l'istruzione immediatamente successiva al codice operativo ENDDO. Viene utilizzato per esempio in un ciclo di lettura di un file di database. Quando l'operazione di lettura provoca l'accensione dell'indicatore di fine file e non si devono effettuare ulteriori elaborazioni ma abbandonare il ciclo di DO/ENDDO, l'istruzione LEAVE abbandona il ciclo e fa riprendere l'esecuzione del programma dopo la istruzione ENDDO.

GOTO Salta

TAG Arrivo salto

Un programma può essere visto come un insieme sequenziale di istruzioni da eseguire una dopo l'altra, nell'ordine in cui sono elencate. Il codice operativo TAG identifica una istruzione lungo il programma alla quale il codice operativo GOTO può trasferire l'esecuzione.

Sul fattore1 della TAG si menziona un nome univoco. Il trasferimento dell'esecuzione all'istruzione TAG avviene quando si percorre una GOTO che porta quel medesimo nome nel fattore2. Per completezza, si osserva che una TAG incontrata lungo l'esecuzione (senza cioè arrivarci tramite una GOTO) risulta indifferente all'elaborazione.

Il salto così realizzato può essere sia in avanti che all'indietro lungo la sequenza delle istruzioni del programma.

In passato, in assenza delle istruzioni di condizionamento e di flusso, con questa coppia di istruzioni si realizzava ogni tipo di ciclo.

In manutenzione di vecchi programmi se ne trovano in quantità e l'ultimo vizio dei vecchi programmatori è ancora quello di uscire da cicli molto annidati abbandonando con un GOTO.

Si menziona anche l'uso di saltare da una subroutine alla routine principale.

Brutte tendenze da evitare ma tollerate dall'' RPG che non si intruppa con gli annidamenti.

Pur essendo GOTO e TAG ormai cadute in disuso nella moderna programmazione strutturata, se ne esemplifica l'uso in un ciclo di lettura da database.

Esempio di lettura di un file di database con il solo uso di GOTO/TAG invece che della sequenza DO/LEAVE/ITER/ENDDO

```
FX   FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++++
      * File da elaborare.
     FANAGR01L  IF   E           K DISK
```

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Si posiziona prima della chiave più bassa sul file.
C      *LOVAL        SETLL     ANAGR
     * Inizio ciclo.
C      ANAGRBEG      TAG
     * Legge un record dal file.
C                    READ      ANAGR                                  50
     * Se non ci sono altri record, abbandona.
C     50             GOTO      ANAGREND
     * Se il record è annullato, ricicla.
C      A1ANNU        IF        'A'
C                    GOTO      ANAGRBEG
C                    ENDIF
     * Elabora il record.
     *              ..........
     * Salta a Inizio ciclo.
C                    GOTO      ANAGRBEG
     * Fine ciclo.
C      ANAGREND      TAG
```

19.6 Operazioni di subroutine.

BEGSR Inizio subroutine
ENDSR Fine subroutine
EXSR Esegui subroutine

Per subroutine si intende un gruppo di istruzioni dotate di un nome e delimitate da un codice operativo BEGSR che ne indica l'inizio ed un altro ENDSR che ne indica la fine.
Il nome della subroutine risiede sul fattore1 della BEGSR.
Le subroutine cominciano dopo l'ultima istruzione di calcolo del flusso principale del programma (Main routine). Dopo la prima BEGSR si incontreranno le istruzioni della prima subroutine chiuse da una ENDSR. Dopo una ENDSR si può trovare soltanto un'altra BEGSR oppure la fine del programma.
Per la verità, dopo trent'anni di RPG, ho scoperto che istruzioni di definizione vengono compilate anche se elencate oltre l'ultima subroutine!
Il codice operativo EXSR si utilizza per richiamare l'esecuzione della subroutine menzionata nel fattore2.
Esempio di utilizzo di una subroutine con passaggio di parametri.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Main routine.
     *              ..........
     * Calcola il giorno della settimana.
C                    Z-ADD     DATA          WKDDIN
C                    EXSR      WKDD
C                    MOVEL(P)  WKDDOU        GIORNO
     *              ..........
     *---------------------------------------------------------------------
     * Calcola il giorno della settimana.
     * Riceve la data in WKDDIN.
     * Ritorna il giorno della settimana in WKDDOU (lunedì, martedì, ...).
C      WKDD          BEGSR
     * Opportuni calcoli.
     *              ..........
C                    ENDSR
     *---------------------------------------------------------------------
```

Facoltativamente si può specificare al fattore1 di una ENDSR una label di arrivo, come se fosse una TAG. Questa indicazione è fornita per la leggibilità di vecchie istruzioni e non se ne consiglia l'uso. Se necessario, ci si rivolgerà all'uso di LEAVESR, che svolge la stessa funzione in programmazione strutturata ma solo dai rilasci più recenti.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     WKDD          BEGSR
 *                  ..........
C                   GOTO      WKDDEND
 *                  ..........
C     WKDDEND       ENDSR
```

Oppure.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     WKDD          BEGSR
 *                  ..........
C                   LEAVESR
 *                  ..........
C                   ENDSR
```

Le subroutine nascono per due ordini di motivi.

1) Per evitare la minutazione ripetuta di gruppi di istruzioni da eseguire più volte.
   Di questa prima specie è, ad esempio, la subroutine di controllo della data, da riportare identica in tutti i programmi di un pacchetto (magari con un /COPY). Dato che in un programma applicativo le date da controllare possono essere molte, è chiaro come, facendo lavorare la subroutine di controllo su una interfaccia bene evidente, forniti i dati all'interfaccia di input, si eseguirà la subroutine e si ricaverà l'esito dall'interfaccia di output.
   A volte questo tipo di calcoli merita di evolvere in programmi separati da richiamare dinamicamente con la stessa interfaccia sopra descritta tramite una CALL ad un programma di utilità (o modulo) invece che tramite una EXSR.
   Di passaggio, si nota che le date nell' RpgIle si controllano, sotto certe condizioni, con TEST(D) che sostituisce la routine di controllo date ancora da scrivere in Rpg400.
2) Per frammentare il flusso in gruppi di istruzioni destinate a scopi standard.
   Di questa seconda specie sono le subroutine specializzate come, ad esempio, la pulizia del video, il controllo del video, la decodifica del video, l'inizio programma etc. che svolgono all'interno di un programma funzioni ritrovabili in molti altri programmi simili. Isolare tali funzioni standard in certe routine permette al manutentore del pacchetto di orientarsi con facilità in un programma strutturalmente simile a molti altri.
   Tendenzialmente, i primi prototipi di un nuovo pacchetto, dai quali si deriveranno per copiatura e modifica molti degli altri, saranno strutturati in modo tale che, nelle copie, il flusso principale resti inalterato e le modifiche riguardino quasi esclusivamente le routine specializzate.

CASxx Compara ed esegue subroutine
ENDCS Fine di Compara ed esegue subroutine

Il codice operativo CASxx permette l'esecuzione condizionata di una subroutine. Se è soddisfatta tra il fattore1 e il fattore2 la condizione espressa dall'operatore relazionale, esegue la subroutine indicata nel risultato.
Ad uno o più codici operativi CAS contigui deve corrispondere un codice operativo ENDCS di chiusura. Si possono cioè specificare più CAS uno di seguito all'altro, ma si deve specificare un solo codice operativo ENDCS a chiusura delle operazioni soprastanti.
L'istruzione CAS è sempre abbinata ad uno degli operatori relazionali elencati per l'istruzione IF.

Come per il WHEN del SELECT, dopo la prima condizione soddisfatta e la relativa esecuzione di subroutine, il controllo passa oltre la ENDCS. Cioè, in un gruppo CAS/CAS.../ENDCS si esegue nessuna o una sola delle subroutine menzionate nel gruppo.
Tra la prima CAS e la ENDCS non sono permesse altre istruzioni che CAS.
Si osserva che la sequenza CAS/CAS.../ENDCS è poco usata perché molto rigida.
Nell'esempio seguente, pur essendo soddisfatte le condizioni relative sia alla subroutine SR12 che alla SR22, solo la prima (SR12) viene eseguita.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
    D Prova1          s              1    inz(2)
    D Prova2          s              1    inz(2)
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
    C     Prova1        CASEQ     1             SR11
    C     Prova1        CASEQ     2             SR12
    C     Prova1        CASEQ     3             SR13
    C     Prova2        CASEQ     1             SR21
    C     Prova2        CASEQ     2             SR22
    C                   ENDCS
```

19.7 Subroutine speciali.

***PSSR** Subroutine di gestione errori

L' RPG possiede una gestione di default delle situazioni di errore che prevede l'uscita di un messaggio di interrogazione al video o, se il lavoro è batch, alla coda messaggi dell'operatore di sistema QSYSOPR. Se invece si specifica la routine *PSSR, la gestione di default non interviene e ci si deve fare carico della gestione degli errori in proprio.
Non si approfondisce però intenzionalmente la *PSSR perché serve ai programmatori pigri a fregarsene di tutti i problemi. Veramente l'intenzione e le potenzialità della subroutine sono orientate al perfetto controllo degli eventi imprevisti. Tuttavia l'uso di tale subroutine delocalizza l'incidente e perciò demotiva il programmatore dal prevedere e risolvere gli eventi dove accadono.
Ad esempio, in un interattivo che scrive nuovi record, se si tenta l'aggiunta di una chiave unica che esiste già e non si intercetta la write con l'indicatore centrale (o con la built in function equivalente %err), il controllo passa alla *PSSR e il programmatore trascura la soluzione migliore: tornare a video e denunciare il problema. Si osserva che il controllo preliminare di non esistenza nel file di una certa chiave unica non schiva la possibilità di incidente in un contesto di forte multiutilizzo (o sulle vecchie macchine in coma di prestazioni) perché al momento della scrittura la chiave potrebbe essere già stata scritta da altri.
Se poi si vuole scrivere una buona *PSSR che risolva i problemi e non si limiti a sostituirsi al default, si va incontro alla scrittura di un numero elevato di istruzioni. Ogni volta che ho iniziato a scrivere una *PSSR ho poi risolto al meglio senza di quella.

***INZSR** Subroutine di inizializzazione

E' la subroutine che il programma RPG esegue automaticamente prima di ogni altro calcolo e subito dopo avere aperto i file. Non è quindi necessario né utile richiamarla esplicitamente in calcolo.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
    C     *INZSR        BEGSR
     *                  ..........
    C                   ENDSR
```

19.8 Operazioni su indicatori.

L' RPG mette a disposizione del programmatore 99 indicatori logici 01-99. Gli indicatori possono assumere solo due stati: *ON attivo, *OFF disattivo. I codici più immediati ed utilizzati per attivare e disattivare gli indicatori sono SETON e SETOFF.

SETON Attiva indicatori

Il codice operativo SETON accende (attiva) tutti gli indicatori elencati tra gli indicatori risultanti. È ininfluente la posizione dei tre indicatori possibili.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Accende gli indicatori 01, 02 e 03
C                   SETON                                          010203
     * Accende l'indicatore 50 in tre modi diversi
C                   SETON                                          50
C                   MOVEL     '1'           *IN50
C                   MOVEL     *ON           *IN50
```

Una volta un Red Book sull'uso delle istruzioni per migliorare le prestazioni ammoniva il programmatore fantasioso a non usare forme alternative per istruzioni molto economiche nel compilato. SETON e SETOFF erano tra queste. Poi su quanti byte occupa una singola istruzione è calato il silenzio. Ma io, che non sono mancino, continuo ad usare la destra anche se altri si sforzano di essere sinistri. Invece che ambidestri diventeranno ambisinistri.

SETOFF Disattiva indicatori

Il codice operativo SETOFF spegne (disattiva) tutti gli indicatori elencati tra gli indicatori risultanti. È ininfluente la posizione dei tre indicatori possibili.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Spegne gli indicatori 01, 02 e 03
C                   SETOFF                                         010203
     * Spegne l'indicatore 50 in tre modi diversi
C                   SETOFF                                         50
C                   MOVEL     '0'           *IN50
C                   MOVEL     *OFF          *IN50
```

Tanto per contraddire la tirata sulle forme alternative, uso volentieri lo spegnimento di un gruppo di indicatori contigui, come esemplifico.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Spegne gli indicatori da 60 a 90.
C                   MOVEL     *OFF          OFF31            31
C                   MOVEA     OFF31         *IN(60)
```

Uso degli indicatori nel condizionamento delle operazioni

Per verificare lo stato dell'indicatore si possono utilizzare due metodi.

Il primo, di uso frequente quando le istruzioni condizionate sono più d'una, è quello di utilizzare l'istruzione IF e considerando l'indicatore come una variabile di programma.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Se 50 è acceso, esegue.
C     *IN50         IFEQ      *ON
C                   READ      DATABASE
C                   EXFMT     VIDEO
C                   ENDIF
     * Se 51 è spento, esegue.
C     *IN51         IFEQ      *OFF
C                   READ      DATABASE
C                   EXFMT     VIDEO
C                   ENDIF
```

L'altro metodo, comodo se l'istruzione da eseguire è una sola, usa l'indicatore di condizionamento, presente su ogni istruzione di calcolo.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Se 50 è acceso, esegue.
C   50              READ      DATABASE
     * Se 51 è spento, esegue.
C  N51              READ      DATABASE
```

Uso degli indicatori nel condizionamento del video

Non è possibile sopprimere totalmente gli indicatori nella gestione di un video, né vale la pena di sobbarcarsi il tentativo di eliminarli usando tecniche alternative di segnalazione dei messaggi o

di parzializzazione dell'emissione, così facile con l'uso degli indicatori. L'odio per gli indicatori risale ad una colonia di profughi dai mainframe che nel Cobol usavano i flag, ma solo con nomi romantici e significativi invece che volgarmente numerici. L'odio si è acuito da quando sull'As/400 sono saliti i ragazzotti del Pascal e, poi, del Visual Basic.

19.9 Operazioni di comparazione.

COMP Compara

Il codice operativo COMP, ormai caduto in disuso, compara il contenuto dei campo specificato sul fattore1 con quello specificato sul fattore2. L'esito della comparazione è fornito dallo stato degli indicatori risultanti. Da notare che l'istruzione spegne preliminarmente tutti gli indicatori risultanti.

fattore1	maggiore di	fattore2	accende primo indicatore	Hi = Higth
fattore1	minore di	fattore2	accende secondo indicatore	Lo = Low
fattore1	uguale a	fattore2	accende terzo indicatore	Eq = Equal

```
D   DName++++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
    D Misura1         s                5  0 inz(1)
    D Misura2         s                3  0 inz(2)
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * La comparazione accende l'indicatore 49.
    C     Misura1       COMP      Misura2                        514950
```

CABxx Compara e salta (Compare and branch)

Il codice operativo CABxx, pure in disuso, si comporta come il codice COMP per quanto riguarda fattore1, fattore2 e indicatori risultanti ma, in più, esegue un eventuale salto alla TAG indicata nel risultato in modo condizionato dalla comparazione tra fattore1 e fattore2 espressa dal solo operatore relazionale. Si noti che gli indicatori risultanti non influenzano in alcun modo il salto, determinato esclusivamente dall'operatore relazionale.

```
D   DName++++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
    D Misura3         s                5  0 inz(3)
    D Misura4         s                3  0 inz(4)
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * La seguente istruzione accende l'indicatore 49 e non salta.
    C     Misura3       CABGT     Misura4       VIA              514950
     * La seguente istruzione accende l'indicatore 51 e salta a VIA.
    C     Misura4       CABGT     Misura3       VIA              514950
```

Sulla CAB è possibile non usare la label di salto nel risultato ottenendo così un'istruzione identica a COMP. Come pure è possibile non usare gli indicatori risultanti ottenendo il solo salto condizionato.

```
D   DName++++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
    D Misura3         s                5  0 inz(3)
    D Misura4         s                3  0 inz(4)
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * La seguente istruzione accende l'indicatore 49.
    C     Misura3       CABGT     Misura4                        514950
     * La seguente istruzione salta a VIA.
    C     Misura4       CABGT     Misura3       VIA
```

19.10 Operazioni sui bit. 19C

BITON Accendi i bit
BITOFF Spegni i bit
TESTB Esamina i bit

Dato un campo alfanumerico lungo uno, è possibile trattarne gli 8 bit che compongono il byte in maniera indipendente mediante le istruzioni di accensione e spegnimento di gruppi di bit.

La manipolazione dei bit non è una tecnica moderna che valga la pena di applicare oggi ma era eseguita in illo tempore per due scopi.

a. Apparecchiarsi dei byte particolari non presenti nel set caratteri della tastiera o comunque non visibili o non riconoscibili anche se, con opportune regole, digitabili.
b. Risparmiare spazio nel database usando flag di un bit invece che di un byte. È a volte il sistema che utilizza tale espediente nelle informazioni restituite da una Api o da un Dspxxx su outfile.

Per comprendere gli esempi, si illustra brevemente l'equivalenza tra notazione binaria ed esadecimale. Per migliori lumi, consultare il capitolo Formato dati (0).

La notazione binaria esprime direttamente i bit accesi e spenti di un semibyte. L'esadecimale corrispondente permette la notazione sintetica del semibyte.

Bin	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Se si vuole capire come sono messi i bit del carattere A, in esadecimale C1, si accostano gli equivalenti binari di C (1100) e di 1 (0001) nella stringa 11000001. Se gli otto bit si numerano da 0 a 7, si ottiene il nome dei bit usato dalle operazioni BITON, BITOFF e TESTB dell' RPG.

Stesso esempio sul carattere asterisco sopralineato, non presente nei caratteri pc ma presente nei caratteri EBCDIC (Tasto Dup): 1C=0001-1100

Si esemplifica la costruzione di un esadecimale, sia nel modo antico, a suon di BITOFF e BITON, che nel modo più moderno, con l'uso di costanti esadecimali.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Compone l'esadecimale 1C.
C                   bitoff    '01234567'    hex1c             1
C                   biton     '345'         hex1c
     * Compone l'esadecimale 1C.
C                   movel(p)  x'1c'         hex1c             1
```

Si esemplifica il test di un flag lungo un bit.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Se il terzo bit da sinistra di FLAG8 è acceso, accende l'indicatore 50.
C                   testb     '2'           flag8             1      50
```

19.11 Tipologie delle Specifiche RPG di Calcolo.

L' RPG permette tre tipologie di istruzione di calcolo.

a. Il formato libero (nel SEU ???).
b. Il formato posizionale (nel SEU C).
c. Il formato esteso (nel SEU CX).

19.12 Specifica di Calcolo in formato libero.

Il formato libero è possibile solo dal rilascio V5R1M0 e perciò, in attesa della necessaria pratica, si rimanda ai manuali RpgIle di tale rilascio.

Si nota che tale formato è permesso in un sorgente soltanto tra le istruzioni di confine /free e /end-free che, per altro, possono ripetersi quante volte si vuole nel corso dello stesso sorgente.

Si nota inoltre che tale formato non fa uso di indicatori risultanti e nemmeno di indicatori di condizionamento in forma breve: permesso "*IN01" ma non "01".

```
     * Istruzioni in formato posizionale o posizionale esteso.
C/free
     * Istruzioni in formato libero.
C/end-free
     * Istruzioni in formato posizionale o posizionale esteso.
```

19.13 Specifica di Calcolo in formato posizionale.

È la forma più antica. Viene usata poco più frequentemente del calcolo esteso per ragioni storiche.

Si richiede nel SEU mediante il comando di riga IPC (Insert Prompt Calculation).

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
```

Dove: C =Specifica RPG di definizione calcolo.
L0 =Livello di totale.
N01 =Indicatore di condizionamento.
Factor1 =Fattore 1 dell'operazione.
Opcode&Ext =Codice operazione ed estensione dell'operazione.
Factor2 =Fattore 2 dell'operazione.
Result =Risultato dell'operazione.
Len =Lunghezza del campo del risultato.
D =Numero di decimali del campo del risultato.
Hi =Indicatore 1 risultante dell'operazione (Fattore1>Fattore2 in COMP).
Lo =Indicatore 2 risultante dell'operazione (Fattore1<Fattore2 in COMP).
Eq =Indicatore 3 risultante dell'operazione (Fattore1=Fattore2 in COMP).

Segue un esempio di calcolo numerico.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Scrive 80 nel campo numerico nbr1 di 15 cifre di cui 5 decimali.
    C                   z-add     80            nbr1             15 5
```

Segue un esempio di calcolo con tutti i campi del tracciato valorizzati.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Se corre il calcolo di totale dopo l'ultimo record del gruppo di
     * controllo di L3 e se l'indicatore 50 è spento.
     *    Divide il campo NBR1 per il campo NBR2 e lo scrive nel campo numerico
     *    NBR3 di 10 cifre con 0 decimali arrotondando il risultato.
     *    Accende l'indicatore 10 se il risultato vale 0,
     *    lo spegne in caso contrario.
     *    Accende l'indicatore 11 se il risultato è maggiore di 0,
     *    lo spegne in caso contrario.
     *    Accende l'indicatore 09 se il risultato è minore di 0,
     *    lo spegne in caso contrario.
    Cl3n50nbr1          div(h)    nbr2          nbr3             10 0110910
```

Il successo dell'RPG è dipeso dalla sinteticità delle istruzioni posizionali che possono essere lette a colpo sicuro anche dalle bestie.

Il successo del free dipende dalla sinteticità permessa dalle built in function. La sua leggibilità non dipende più da regole posizionali ma dallo stile di minutazione del programmatore che deve indentare, spaziare, nomenclare, usare procedure, eccetera.

L'esperienza dice che in programmazione le forme strane, illeggibili, pericolose e scommentate trovano sempre qualche estimatore che ne abusa. Conosco chi usa il free per non farsi capire dagli anziani ignoranti e mi viene il paio con un pacchetto pc in basic interpretato che nei primi anni 80 gestiva le casse del negozio Fiorucci di via Passarella, scritto senza spazi e senza a capo, tanto per scoraggiare i poveri compratori dal capirlo.

Se state apprezzando il free, ricordate che la sua leggibilità dipende da chi lo scrive e non dal linguaggio in sé e che un casinista fa più danni in free che in posizionale.

Il posizionale ha permesso di allevare per trent'anni una schiera di programmatori che si capivano al volo. Compatisco chi non potrà prossimamente fare più conto su qualcosa di equivalente.

Segue un esempio di calcolo alfanumerico.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Se è acceso l'indicatore 77.
     *    Pulisce il campo del risultato ALFA3.
     *    Trascrive il contenuto del campo ALFA2 nel campo ALFA3
     *    di 128 caratteri allineandolo a sinistra.
    C   77              movel(p)  alfa2         alfa3           128
```

Segue un esempio di condizionamento complesso con l'uso di IF.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Se il campo OPTION vale "1" e il campo TYPE vale "ADD" oppure
     * se il campo OPTION vale "2" e il campo TYPE vale "CHG".
     *    Esegue la subroutine SRADDCHG.
    C     option        ifeq      '1'
    C     type          andeq     'ADD'
    C     option        oreq      '2'
    C     type          andeq     'CHG'
    C                   exsr      sraddchg
    C                   endif
```

Segue un esempio di condizionamento complesso con l'uso di WHEN.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Se il campo OPTION vale "1" e il campo TYPE vale "ADD" oppure
     * se il campo OPTION vale "2" e il campo TYPE vale "CHG".
     *    Esegue la subroutine SRADDCHG
     * Se il campo OPTION vale "3" e il campo TYPE vale "DLT".
     *    Esegue la subroutine SRDLT
     * In tutti gli altri casi.
     *    Esegue la subroutine SRERR
    C                   select
    C     option        wheq      '1'
    C     type          andeq     'ADD'
    C     option        oreq      '2'
    C     type          andeq     'CHG'
    C                   exsr      sraddchg
    C     option        wheq      '3'
    C     type          andeq     'DLT'
    C                   exsr      srdlt
    C                   other
    C                   exsr      srerr
    C                   endsl
```

19.14 Specifica di Calcolo in formato esteso (con fattore 2 esteso e libero).

Permette una maggiore leggibilità perché utilizza un fattore 2 più largo sopprimendo i restanti campi e viene usata da alcune istruzioni importanti.

Si richiede nel SEU mediante il comando di riga IPCX (Insert Prompt Calculation eXtended).

```
CX  CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++++
    C                   EVAL      phrase=%trim(phrase) + ` ' + %trim(word1)
```

Dove: C =Specifica RPG di definizione calcolo.
L0 =Livello di totale.
N01 =Indicatore di condizionamento.
Factor1 =Fattore 1 dell'operazione.
Opcode&Ext =Codice operazione ed estensione dell'operazione.
Extended-factor2 =Fattore 2 dell'operazione.

Solo alcune istruzioni possono usare sia il formato posizionale che quello esteso (IF, WHEN) mentre altre esistono solo nel formato esteso (EVAL).

Segue il medesimo esempio di condizionamento complesso IF sopra riportato, ma in forma estesa.

```
CX  CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++++
     * Se il campo OPTION vale "1" e il campo TYPE vale "ADD" oppure
     * se il campo OPTION vale "2" e il campo TYPE vale "CHG".
     *    Esegue la subroutine SRADDCHG
    C                   if        option='1' and type='ADD' or
    C                             option='2' and type='CHG'
    C                   exsr      sraddchg
    C                   endif
```

Per maggiore chiarezza o per necessità si può far uso di parentesi.

```
CX  CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++++
C                   if        (option='1' and type='ADD') or
C                             (option='2' and type='CHG')
C                   exsr      sraddchg
C                   endif
```

Segue il medesimo esempio di condizionamento complesso WHEN sopra riportato, ma in forma estesa.

```
CX  CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++++
 * Se il campo OPTION vale "1" e il campo TYPE vale "ADD" oppure
 * se il campo OPTION vale "2" e il campo TYPE vale "CHG".
 *    Esegue la subroutine SRADDCHG
 * Se il campo OPTION vale "3" e il campo TYPE vale "DLT".
 *    Esegue la subroutine SRDLT
 * In tutti gli altri casi.
 *    Esegue la subroutine SRERR
C                   select
C                   when      option='1' and type='ADD' or
C                             option='2' and type='CHG'
C                   exsr      sraddchg
C                   when      option='3' and type='DLT'
C                   exsr      srdlt
C                   other
C                   exsr      srerr
C                   endsl
```

L'uso della specifica CX permette anche un uso semplificato degli indicatori nelle IF.

L'esempio vecchio tracciato.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     *in50         ifeq      *on
C     *in51         andne     *on
```

Si trasforma nel seguente.

```
CX  CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++++
C                   if        *in50 and not *in51
```

19.15 EVAL su stringhe con uso di Built In Function.

L'istruzione EVAL permette la composizione di stringhe in modo molto libero.

Dal rilascio V4R4M0, l' RpgIle è dotato anche dell'istruzione EVALR che allinea a destra il contenuto del campo ricevente,

Seguono alcuni esempi.

Si vogliano accodare le parole contenute nei campi WORD1 e WORD2 al contenuto del campo PHRASE intervallandole con un blank.

```
CX  CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++++
C                   eval      phrase=%trim(phrase) + ' ' + %trim(word1)
C                   eval      phrase=%trim(phrase) + ' ' + %trim(word2)
```

Dove %trim Scontorna i blank in testa ed in coda al contenuto del campo.

Si voglia, ad esempio, trascrivere il contenuto del campo LUNGO nel campo CORTO, troncando i primi due caratteri.

```
CX  CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++++
C                   eval      corto=%subst(lungo:3)
```

Dove %subst Estrae dal campo lungo il contenuto dal carattere 3 fino alla fine.

Si voglia trascrivere il contenuto del campo LUNGO nel campo CORTO, prelevandone i caratteri da 5 a 20 ed allineando a destra il contenuto del ricevente.

```
CX  CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++++
C                   evalr     corto=%subst(lungo:5:16)
```

Dove %subst Estrae dal campo lungo il contenuto dal carattere 5 per 16 caratteri fino al carattere 20.

19.16 EVAL numerici. Problemi di capienza.

L'istruzione EVAL usata sui campi numerici per eseguire calcoli algebrici, a differenza delle istruzioni matematiche più antiche (ADD, SUB, MULT, DIV, Z-ADD, Z-SUB, SQRT,...), denuncia errore se il campo del risultato non è abbastanza capiente da contenere tutti gli interi del risultato.

Ad esempio, a tempo di esecuzione, se sul sistema corre la data 13/04/2002, la seguente istruzione di calcolo riempie il campo CCYYMMDD (8 cifre, 0 decimali) con il valore 20020413.

```
D    DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
     D ccyymmdd        s               8   0
CX   CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++
     C                   eval       ccyymmdd=*year*10000+*month*100+*day
```

Se invece il campo è grande solo 6 cifre, l'istruzione EVAL termina in errore.

```
D    DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
     D ccyymmdd        s               6   0
CX   CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++
     C                   eval       ccyymmdd=*year*10000+*month*100+*day
```

Ancora, la seguente istruzione EVAL riempie il campo result con il valore 7200.

```
D    DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
     D result          s               5   0
     D factor1         s               2   0 inz(90)
     D factor2         s               2   0 inz(80)
CX   CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++
     C                   eval       result=factor1*factor2
```

Invece, la seguente istruzione EVAL termina in errore.

```
D    DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
     D result1         s               3   0
     D factor1         s               2   0 inz(90)
     D factor2         s               2   0 inz(80)
CX   CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++
     C                   eval       result1=factor1*factor2
```

Esistono tuttavia opzioni di compilazione che evitano la segnalazione dell'errore.

19.17 EVAL numerici. Problemi di arrotondamento intermedio.

L'istruzione EVAL usata sui campi numerici si inventa la dimensione dei campi intermedi nei calcoli che coinvolgono dei decimali. Tali campi intermedi sono dimensionati solo in base alle caratteristiche del campo del risultato. Ne segue che se il risultato non prevede decimali, anche i calcoli intermedi li perdono strada facendo.

Ad esempio, la seguente istruzione scrive nel campo result2 il valore 3000 invece che l'atteso 3333

```
D    DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
     D result2         s               4   0
CX   CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++
     C                   eval       result2=10/3*1000
```

Esistono tuttavia opzioni di compilazione che provocano calcoli intermedi più accurati.

20 Specifiche RPG di emissione. (20R)

Le specifiche di emissione **O**, si minutano dopo le specifiche di calcolo C e calcolo esteso CX. Non sono di frequente utilizzo ed hanno un impiego piuttosto specifico.

Si possono utilizzare.

- Per definire delle stampe descritte internamente al programma. In questo caso non è necessario definire un file di stampa specifico descritto esternamente.
- Per effettuare delle operazioni di aggiornamento e scrittura su file di database.

20.1 Stampe

Per un esempio significativo di stampa descritta internamente con uso di specifiche di emissione si rimanda al capitolo File di stampa (14.3).

20.2 Aggiornamento parziale su file di database esterno

Nel programma si definisce un file esterno di aggiornamento elaborato liberamente.

```
FX  FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++++
    Fester01l  uf   e           k disk
```

Letto un record, si modificano in memoria solo i campi che si vogliono aggiornare. Si comanda quindi la modifica del record letto richiamando un gruppo di specifiche di emissione contraddistinte da un apposito *nome di eccezione*.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
    C     kester        chain     ester                              50
    C                   if        not *in50
    C                   z-add     tota          x1tota
    C                   z-add     *date         x1dtch
    C                   except    upd
    C                   endif
```

Nelle specifiche di emissione, il *nome di eccezione* (UPD, nell'esempio) identifica una specifica di record seguita dalle specifiche di campo che elencano solo i campi da aggiornare.
Nell'esempio seguente solo i campi X1TOTA e X1DTCH del record ESTER del file ESTER01L verranno aggiornati.

```
O   ONomefile++DF..N01N02N03Excnam++++B++A++Sb+Sa+.............................
    Oester     e            upd
P   O.............N01N02N03Field+++++++++YB.End++PConstant/editword/DTformat++
    O                       x1tota
    O                       x1dtch
```

Se non specificato alcun campo, l'esito dell'operazione di emissione sarà il rilascio del record dal vincolo imposto dalla lettura per update.

```
O   ONomefile++DF..N01N02N03Excnam++++B++A++Sb+Sa+.............................
    Oester     e            upd
```

20.3 Aggiornamento e scrittura su file di database interno

Le specifiche di emissione servono anche a provocare l'aggiornamento o la scrittura su file di database descritti internamente, secondo una tipologia di programmazione ormai desueta, utile semmai solo in certi programmi di utilità molto tecnici.

Aggiornamento

Nel programma si definisce un file interno di aggiornamento elaborato liberamente con chiavi.

```
F   FFilename++IPEASFRlen+LKlen+AIDevice+.Par.chi.++++++++++++++++++++++++++++++
    Finter01l  uf   f   40     3aidisk
```

Specificare input interno

Letto un record, si modificano in memoria solo i campi che si vogliono aggiornare. Si comanda quindi la modifica del record letto richiamando un gruppo di specifiche di emissione contraddistinte da un apposito *nome di eccezione*.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     'A10'          chain     inter01l                            50
C                    if        not *in50
C                    movel(p)  'Bulloni'     y2desc           25
C                    z-add     148,32        y2quan            5 2
C                    except    modrec
C                    endif
```

Il record prima della modifica è il seguente.

```
....+....1....+....2....+....3....+....4
A10Dado a galletto          00100Marameo
```

Nelle specifiche di emissione, il *nome di eccezione* (MODREC, nell'esempio) identifica una specifica di record seguita dalle specifiche di campo che elencano tutti i campi da emettere. Il ricalco di dati è mirato alle posizioni finali e non copre necessariamente tutto il record.

```
O   ONomefile++DF..N01N02N03Excnam++++B++A++Sb+Sa+.............................
O   Ointer01l  e            modrec
P   O.............N01N02N03Field+++++++++YB.End++PConstant/editword/DTformat++
O                        y2desc              28
O                        y2quan         S    33
```

Il record dopo la modifica sarà il seguente. Si noti che i dati emessi non coprono tutta la lunghezza del record e che nella parte non coperta vengono lasciati i dati preesistenti.

```
....+....1....+....2....+....3....+....4
A10Bulloni                  14832Marameo
```

Scrittura

Nel programma si definisce un file interno di emissione.

```
F   FFilename++IPEASFRlen+LKlen+AIDevice+.Par.chi.++++++++++++++++++++++++++++++
F   Finter00f  o    f   40        disk
```

Si approntano in memoria i campi da scrivere e si comanda quindi l'emissione del record tramite un gruppo di specifiche di emissione contraddistinte da un apposito *nome di eccezione*.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                    movel(p)  all'x'        allx             40
C                    movel(p)  'A10'         y2codi            3
C                    movel(p)  'Bulloni'     y2desc           25
C                    z-add     148,32        y2quan            5 2
C                    except    addrcd
```

Nelle specifiche di emissione, il *nome di eccezione* (ADDRCD, nell'esempio) identifica una specifica di record seguita dalle specifiche di campo che elencano tutti i campi da emettere.

```
O   ONomefile++DF..N01N02N03Excnam++++B++A++Sb+Sa+.............................
O   Ointer00f  eadd         addrcd
P   O.............N01N02N03Field+++++++++YB.End++PConstant/editword/DTformat++
O                        y2codi
O                        y2desc
O                        y2quan         S
```

Il record emesso sarà il seguente. Si noti che i dati emessi non coprono tutta la lunghezza del record e che nella parte non coperta vengono scritti dei blank di default.

```
....+....1....+....2....+....3....+....4
A10Bulloni                  14832
```

In alternativa, invece che giustapporre i campi, se ne può specificare la posizione finale.

```
O   ONomefile++DF..N01N02N03Excnam++++B++A++Sb+Sa+.............................
    Ointer00f  eadd         addrcd
P   O.............N01N02N03Field+++++++++YB.End++PConstant/editword/DTformat++
    O                       y2codi               3
    O                       y2desc              28
    O                       y2quan         S    33
```

21 Specifiche Rpg400. (21R)

Il linguaggio RpgIle è l'ultima versione del linguaggio RPG ed è proprio la versione Ile l'unica ad evolvere dopo la sua apparizione. Nel passaggio all'Ile, alcune tipologie di specifiche si sono perse o sono mutate ma, essendo necessario conoscerle per scopi di manutenzione, in questo capitolo se ne tratta sommariamente.

21.1 Specifica Rpg400 di Controllo (H).

Questa, se utilizzata, deve essere la prima specifica nel programma Rpg400. Ne è permessa una sola e la sua funzione è quella di definire alcune caratteristiche di compilazione e di esecuzione del programma. Le funzionalità che analizzeremo sono: formato della data di sistema all'interno del programma, carattere separatore per la data, modalità di editazione per i campi numerici.

Esempio di specifica di controllo all'italiana, che richiede il formato data anno/mese/giorno, la barra come separatore della data e la regola europea per punto delle migliaia, virgola decimale e zero prima dei decimali.

```
H---H.....1..CDYI....S.............1.F.............................Id-pgm
H             D/J
```

Dove: H =Specifica RPG di controllo.
- D =Formato della data giorno/mese/anno.
  - M =mese/giorno/anno.
  - D =giorno/mese/anno.
  - Y =anno/mese/giorno.
- / =Carattere separatore della data.
  - & =Valore particolare che significa blank come separatore data.
- J =Editazione dei campi numerici.
  - blank =Punto decimale e virgola separatrice delle migliaia.
  - I =Virgola decimale e punto separatore delle migliaia,
  - J =Equivale a I, tranne per lo zero che viene scritto alla sinistra della virgola decimale quando il campo contiene un saldo zero.
  - D =Come blank.

La specifica di controllo ha assunto nell' RpgIle un tracciato completamente libero. L'esempio proposto equivale alla seguente specifica di controllo RpgIle.

```
H   HParole chiave++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
H decedit('0,') datedit(*dmy/)
```

21.2 Specifica Rpg400 di Estensione (E).

Questa specifica serve per definire le schiere (e le tabelle, che qui non trattiamo).

Esempio di definizione di una schiera di nome QT, di 16 elementi numerici, di dimensione 9 interi di cui 2 decimali.

```
E---E....Dalfile+Alfile++Nome++N°rN°elLunPDSNomescLunPDSCommenti+++++++++.....
E                       QT         16  9 2
```

Dove: E =Specifica RPG di estensione.
- QT =Nome della schiera.
- 16 =Numero degli elementi della schiera.
- 9 =Lunghezza di ogni singolo elemento.
- 2 =Numero di decimali di ogni singolo elemento.

Se la schiera viene fornita a tempo di compilazione, occorre indicare inoltre il numero di elementi scritti su ognuno dei record aggiuntivi in fondo al sorgente

Esempio di definizione di una schiera di nome LET, 4 elementi per record, 10 elementi in tutto, ciascuno di dimensione 20 caratteri. Sono esemplificati anche i record che permettono il riempimento della schiera a tempo di compilazione.

```
E---E....Dalfile+Alfile++Nome++N°rN°elLunPDSNomescLunPDSCommenti+++++++++......
    E                    LET     4  10 20
```

Dove: E =Specifica RPG di estensione.
LET =Nome della schiera.
4 =Numero di elementi per record.
10 =Numero degli elementi della schiera.
20 =Lunghezza di ogni singolo elemento.

```
**   ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
**
uno                 due                 tre                 quattro
cinque              sei                 sette               otto
nove                dieci
```

Dove: ** =Riga di inizio dei dati della schiera (asterisco, asterisco, blank).

La specifica di estensione si è trasformata nell' RpgIle nella specifica Dati (D). Gli esempi proposti equivalgono alle seguenti specifiche dati RpgIle.

```
D   DName++++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
D qt                s                9  2 dim(16)
D let               s               20    dim(10) ctdata perrcd(4)
```

21.3 Specifica Rpg400 di Struttura dati (DS).

La specifica DS serve per definire le strutture dati, tanto descritte esternamente quanto internamente al programma. Per la definizione di una DS descritta esternamente è sufficiente la sola istruzione DS affinché il programma conosca ed acquisisca tutta la definizione.

```
DS--INomeds....NODsFile-est++.........   Ric+Lun+..........................
    IKPJBA     E DS
```

Per una DS definita a programma si dovranno invece specificare anche i sottocampi di cui la stessa è composta utilizzando la specifica SS.

21.4 Specifica Rpg400 di Sottocampo di struttura dati (SS).

Questa specifica serve per definire i sottocampi di una struttura dati definita a programma. Nell'esempio che segue viene definita una struttura dati di nome PP, con due sottocampi di cui il primo è un campo alfanumerico lungo 6 ed il secondo è un campo numerico lungo 8,0.

```
DS--INomeds....NODsFile-est++.........   Ric+Lun+..........................
    IPP          DS
SS--I.............Campo-est+.........  PDa++A+++DCampo+.....................
    I                                       1   6 PPCLI
    I                                       7  140PPDAT
```

Le specifiche di struttura dati e di sottocampo si sono trasformate nell' RpgIle nella specifica Dati (D). Gli esempi proposti equivalgono alle seguenti specifiche dati RpgIle.

```
D   DName++++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
D kpjba          e ds
D pp               ds
D  ppcli                    1      6
D  ppdat                    7     14  0
```

22 Ciclo RPG. (22R)

CONFRONTARE CON CAPITOLO (17)

Normalmente, in RPG, per elaborare i record dei file, si utilizza la tecnica della lettura libera, comune a tutti i linguaggi della stessa generazione. Si dichiarano quindi i file come disponibili all'elaborazione interamente procedurale (full procedural) e, nel calcolo, si usano le operazioni di posizionamento, lettura, ricalco e scrittura secondo necessità.

Una vecchia tecnica dell' RPG permette tuttavia di gestire un singolo file, detto primario, ed altri in coda a quello, detti secondari, tramite una lettura automatica sequenziale. Questa funzionalità viene definita come ciclo RPG. Se i file sono ordinati è inoltre possibile automatizzare il riconoscimento dei gruppi di record con lo stesso valore in un campo tramite le rotture di livello.

È inoltre possibile elaborare il primario ed i secondari ordinati secondo campi omogenei (stesse caratteristiche, anche se nomi diversi) come se fossero giacenti in un solo archivio ordinato nello stesso modo del primario. Tale tecnica è detta matching record.

Il ciclo permetteva all' RPG dei vecchi sistemi batch di ottenere la maggior parte delle elaborazioni con un dispendio minimo di specifiche.

Non si tratterà in questa sede né del ciclo in tutti i particolari, né dei file secondari, né della tecnica di matching record. Per tutto ciò si rimanda ai manuali ufficiali che ormai lo spiegano poco e male.

Un solo uso è superstite e riguarda sostanzialmente le stampe con totalizzazioni di archivi bene ordinati.

Si riporta un database esemplificativo CITMU00F, già ordinato su Nazione, Regione, Città.

Nazione	Regione	Città	Nr musei
Francia	Normandia	Baraque	1
Francia	Normandia	Barbun	1
Francia	Normandia	Paluce	2
Francia	Normandia	Zufire	1
Francia	Provenza	Lione	7
Francia	Provenza	Nizza	3
Italia	Lombardia	Como	3
Italia	Lombardia	Lecco	2
Italia	Lombardia	Milano	20
Italia	Lombardia	Varese	5
Italia	Piemonte	Asti	3
Italia	Piemonte	Novara	6
Italia	Piemonte	Torino	18
Italia	Toscana	Arezzo	6
Italia	Toscana	Firenze	32
Italia	Toscana	Grosseto	5
Italia	Toscana	Livorno	4
Italia	Toscana	Lucca	4
Svizzera	Ticino	Chiasso	2
Svizzera	Ticino	Lugano	8
Svizzera	Ticino	Mendrisio	1
Svizzera	Geneve	Burlon	2
Svizzera	Geneve	Geneve	10

La stampa che si vuole ottenere è simile alla seguente.

```
....+....1....+....2....+....3....+....4....+...

STAREND      Elenco Città con Musei   25/03/2003

------------------------------------------------
Nazione       Regione    Città      Nr musei
------------------------------------------------

Francia       Normandia   Baraque          1
                          Barbun           1
                          Paluce           2
                          Zufire           1
                      Totale regione       5 *
              Provenza    Lione            7
                          Nizza            3
                      Totale regione      10 *
                      Totale nazione      15 **
Italia        Lombardia   Como             3
                          Lecco            2
                          Milano          20
                          Varese           5
                      Totale regione      30 *
              Piemonte    Asti             3
                          Novara           6
                          Torino          18
                      Totale regione      27 *
              Toscana     Arezzo           6
                          Firenze         32
                          Grosseto         5
                          Livorno          4
                          Lucca            4
                      Totale regione      51 *
                      Totale nazione     108 **
Svizzera      Ticino      Chiasso          2
                          Lugano           8
                          Mendrisio        1
                      Totale regione      11 *
              Geneve      Burlon           2
                          Geneve          10
                      Totale regione      12 *
                      Totale nazione      23 **
                      Totale generale    146 ***
*** Fine stampa ***
```

22.1 Definizione del file Primario.

Il file per il quale si intende utilizzare questa funzionalità deve essere dichiarato, nelle specifiche di definizione dei file, come file primario.

Un file di database esterno da leggere automaticamente con il ciclo RPG viene così definito.

```
FX  FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++++
    Fcitmu00f  IP A E            DISK
```

Dove: F =Specifica di file
CITMU00F =Nome del file
I =Input
P =Primario

A =Sequenza ascendente
E =External described
DISK =File di database su disco

22.2 Definizione della sequenza di ordinamento e dei gruppi di controllo.

Sulle specifiche di immissione si posizionano dei contrassegni (da M9 a M1) che indicano e controllano l'ordinamento dei record secondo i campi scelti. Sulle stesse righe si posizionano analoghi contrassegni (da L9 a L1) che identificano i gruppi di controllo.

Si nota che la terminologia relativa ai gruppi di controllo è puntata al confine tra due gruppi, detto rottura di livello.

È buona norma far corrispondere i contrassegni Lx agli Mx usando sulla stessa riga di immissione la stessa x.

Si nota che la x più alta si assegna al campo più importante, esattamente il contrario di quanto avviene nel Query, che utilizza concetti analoghi a quelli del ciclo RPG.

Esempio Mx Lx

I record contenuti nel file sopra descritto vengono letti sequenzialmente (dal primo all'ultimo record) in modo automatico dal programma. Ad ogni record letto, viene richiamato il calcolo di dettaglio.

22.3 Calcolo di dettaglio e Calcolo di totale.

Nei programmi dotati di file primario, dopo la lettura di ogni record, viene eseguito ogni volta il calcolo presente in quella che, per gli altri programmi, è la main routine. Tale calcolo viene chiamato *di dettaglio*.

Subito dopo il calcolo di dettaglio, l' RPG esegue una pre-lettura del record successivo e, confrontando il contenuto dei due record, determina se il record corrente è l'ultimo di un gruppo. Se è così, viene acceso l'indicatore corrispondente al gruppo in fase di rottura.

L'accensione di un indicatore di livello comporta l'accensione automatica di tutti gli indicatori di livello inferiore. Ad esempio, l'accensione di L3 comporta l'accensione automatica di L2 e L1.

Esempio calcolo di dettaglio.

A causa delle definizioni già illustrate (contrassegno Lx su una specifica di immissione di campo). durante il calcolo di dettaglio del primo record di un gruppo (lo stesso gruppo definito dal campo su cui si trova il contrassegno Lx) risulta acceso automaticamente un indicatore particolare, Lx, identico nella forma al contrassegno che lo provoca. Questo permette di eseguire calcolo di dettaglio condizionato dall'inizio di un gruppo. L'indicatore Lx si spegne automaticamente appena finito il calcolo di dettaglio.

Esempio calcolo di dettaglio di inizio gruppo.

Dopo ogni record di fine gruppo viene inoltre eseguito il calcolo *di totale* del gruppo corrispondente. Se Lx è il contrassegno, Lx è anche l'indicatore da scrivere nelle posizioni iniziali della specifica di calcolo da eseguire a fine del gruppo di Lx. Si noti che l'indicatore di livello ha una posizione riservata, ben distinta da quella dell'indicatore di condizionamento che si trova poco oltre lungo la specifica.

Esempio calcolo di totale.

L'accensione di un indicatore di livello comporta l'accensione automatica di tutti gli indicatori di livello inferiore. Ad esempio, l'accensione di L3 comporta l'accensione automatica di L2 e L1..

Lo spegnimento di tutti gli indicatori di livello eventualmente accesi avviene subito dopo ogni calcolo di dettaglio.

22.4 Rotture di livello.

L' RPG permette di specificare fino ad un massimo di 9 rotture di livello. Per rottura di livello si intende la segnalazione a programma dell'inizio (primo record) e della fine (ultimo record) di un gruppo di record aventi il medesimo contenuto del campo di rottura. La segnalazione di avvenuta rottura di livello consiste nell'attivazione di specifici indicatori (L1-L9).

La rottura di livello avviene poiché l' RPG, dopo il calcolo di dettaglio e prima del calcolo di totale, esegue una pre—lettura del record successivo. Se il record corrente è l'ultimo record di un gruppo l' RPG accende l'indicatore assegnato a tale livello di rottura (Lx).

Le rotture di livello devono essere definite in modo gerarchico partendo da quella di livello più basso (L1) a quella di livello più alto (L9).

Il file di database deve necessariamente essere ordinato almeno per i campi che costituiscono campi di rottura di livello.

Supponiamo di dovere utilizzare le rotture di livello per un file statistico opportunamente ordinato per Continente, Nazione, Regione, Provincia.

Il livello di rottura più alto è il	Continente	L4
Il livello di rottura immediatamente inferiore è la	Nazione	L3
Il livello di rottura immediatamente inferiore è la	Regione	L2
Il livello di rottura più basso è la	Provincia	L1

E' chiaro che nel momento in cui si leggesse un record con regione diversa da quella precedente (L2), anche la provincia cambierebbe (L1).

E' evidente che nel momento in cui si leggesse un record con nazione diversa da quella precedente (L3), anche la regione cambierebbe (L2) e così pure la provincia (L1).

E' lampante che nel momento in cui si leggesse un record con continente diverso da quello precedente (L4), anche la nazione cambierebbe (L3), la regione cambierebbe (L2) e così pure la provincia (L1).

Da cui si ricava che, ad eccezione del livello di rottura più basso (L1), per tutti gli altri livelli definiti, quando avviene la rottura, avviene anche per tutti i livelli sottostanti.

Ultimo livello di rottura che non necessita di definizione da parte del programmatore ma che viene messa a disposizione direttamente dall' RPG è la rottura di livello finale. Questo tipo di rottura avviene solo a totale e soltanto dopo che è stato letto l'ultimo record nel file. L'avvenuta rottura di livello viene segnalata a programma poiché viene attivato l'indicatore specifico LR (Last Record).

22.5 Totalizzazioni.

Un utilizzo efficace delle rotture di livello lo si può apprezzare sulle totalizzazioni. Infatti, supponendo di dovere utilizzare l'archivio statistico di cui al punto precedente, e di dovere totalizzare degli importi di fatturato per tutti i livelli di rottura specificati, si dovrà procedere come segue.

A rottura di dettaglio provincia (L1), si dovranno azzerare i campi di totalizzazione per il livello corrente. In questo momento si possono altresì eseguire delle decodifiche sempre relative al livello corrente.

A rottura di totale provincia (L1), si dovranno emettere i totali (su stampa, video o file di database), sommare i valori dei campi di totalizzazione del livello corrente (L1) nei campi di totalizzazione del livello superiore (L2).

Quanto appena descritto si deve effettuare per tutti i livelli di rottura definiti, ad eccezione della rottura di livello definita come livello di rottura più alta per la quale, non avendo altri livelli

di rottura superiori definiti, si dovrà eventualmente sommare i valori di totalizzazione nei campi, se previsti, di totale generale (LR).

23 Uso del database. (23RC)

L'insieme delle informazioni e dei dati organizzati che risiedono nei file costituiscono il database. Per potere effettuare le operazioni di lettura, scrittura ed aggiornamento del database, l' RPG mette a disposizione dei codici operativi specifici.

Qui si spiega l'accesso mirato al database usando le istruzioni native dell' RPG.

La tendenza essequellara dei giorni nostri, oltre a complicare più che spesso le cose semplici, introduce un aggravio di prestazioni per processori e dischi e richiede una capacità algoritmica elevata da parte del programmatore. Un esperto che si converte all'uso a tappeto dell'sql dentro l'RPG castra la sua logica applicativa perché dedica il suo tempo ad una cosa già risolta.

Usate l'sql solo per quel che le dds non fanno e non per riscoprire l'acqua calda.

I posizionamenti e le letture sono illustrati innanzitutto per un file logico con chiavi. Le altre tipologie di file saranno spiegate per differenza.

23.1 File esemplificativo.

Negli esempi successivi di posizionamento e lettura si ipotizza il file logico di database *Animali* con due chiavi: *Famiglia* e *Specie*. Per semplicità, il file non contiene altri campi che quelli che fungono da chiave unica. Viene riportato anche un campo di sequenza per esplicitare la posizione delle operazioni di lettura.

Più avanti, negli esempi successivi alla lettura (scrittura, ricalco, cancellazione) si userà anche il fisico senza chiavi ANIMA00F sul quale il logico si basa.

	File: ANIMA00F Record:.ANIMA (Animali)	
sequenza	**A1FAM (Famiglia)**	**A1SPE (Specie)**
0		
1	CANIDE	CANE
2	CANIDE	DINGO
3	CANIDE	IENA
4	CANIDE	LUPO
5	CANIDE	VOLPE
6	FELINO	GATTO
7	FELINO	LEONE
8	FELINO	PANTERA
9	FELINO	TIGRE
10	PESCE	ALBORELLA
11	PESCE	CAVEDANO
12	PESCE	LUCCIO
13	PESCE	STORIONE
14	PESCE	TROTA
15	UCCELLO	CORVO
16	UCCELLO	MERLO
17	UCCELLO	SASSELLO
18	UCCELLO	TORDO

La posizione cursore 0 è quella prima del record 1.
La posizione cursore 10 è quella tra i record 10 e 11.

Il file si definirà nel programma RPG come segue.

```
FX  FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++++
    Fanima01l  if   e           k disk
```

Le chiavi di lettura saranno così definite.

```
D   DName++++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++
    D kkfam           s                   like(a1fam)
    D kkspe           s                   like(a1spe)
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
    C     kanima        klist
    C                   kfld                    kkfam
    C     kanima2       klist
    C                   kfld                    kkfam
    C                   kfld                    kkspe
```

23.2 Cursore.

Si numerano i record in base all'ordine in cui si presentano lungo il file, sia se ordinato per chiave, sia se per numero relativo di record.

Si identifica la posizione dopo ogni record con lo stesso numero del record che la precede.

Si assegna il valore zero alla posizione prima del primo record.

Si definisce *cursore* lungo il file la posizione numerata che costituisce il punto di partenza per la prossima lettura in avanti (READ o READE) o all'indietro (READP o READPE).

Per cursore sposizionato si intende una situazione che non permette mai l'esito positivo di una successiva READxx. Si verifica sempre dopo una lettura fallita per qualunque ragione (record occupato se update, fine file, fine gruppo).

23.3 Posizionamento del cursore.

Il posizionamento del cursore lungo il file di database predispone un punto di partenza per le operazioni di lettura READ, READP, READE e READPE. Il posizionamento del cursore scavalca i record non interessanti ed evita numerose ed inutili letture di record, rendendo più prestazionale l'elaborazione.

SETLL Posizionati subito prima della chiave (Set Lower Limit)

Il codice operativo SETLL non acquisisce il record ma si limita ad eseguire il posizionamento del cursore immediatamente prima del record con chiave uguale o successiva a quella specificata a fattore1. In tal modo, ad esempio, l'eventuale successiva lettura in avanti (READ) acquisirebbe il record in questione.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
    C     kanima        setll     anima
```

Dopo una SETLL il cursore è sempre posizionato validamente, persino in caso di file vuoto.

La SETLL accetta come chiave di posizionamento i valori speciali predefiniti *LOVAL (valore più basso) ed *HIVAL (valore più alto).

Per quanto la SETLL non acquisisca alcun record, se durante il posizionamento viene trovato almeno un record avente la chiave espressa nel fattore1, si accende l'indicatore risultante nella posizione Equal.

Ad esempio, se nel file esiste un record con la chiave del posizionamento, si accende l'indicatore 50 che altrimenti si spegne.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
    C     kanima        setll     anima                                  50
```

Seguono esempi di posizionamento del cursore basati sul file esemplificativo.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
    C                   eval      kkfam='FELINO'
    C     kanima        setll     anima                                  50
     * Cursore risultante 5. Indicatore 50 acceso.
```

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                     eval      kkfam='INSETTO'
C     kanima          setll     anima                                    50
 * Cursore risultante 9. Indicatore 50 spento.
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                     eval      kkfam='UCCELLO'
C                     eval      kkspe='MERLO'
C     kanima2         setll     anima                                    50
 * Cursore risultante 15. Indicatore 50 acceso.
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                     eval      kkfam='PESCE'
C                     eval      kkspe='PERSICO'
C     kanima2         setll     anima                                    50
 * Cursore risultante 12. Indicatore 50 spento.
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     *loval          setll     anima                                    50
 * Cursore risultante 0. Indicatore 50 spento.
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     *hival          setll     anima
 * Cursore risultante 18. Indicatore 50 spento.
```

Non vi venga in mente di usare la SETLL al posto della CHAIN per sapere se una chiave esiste nel file. Si potrebbe pensare che la SETLL, che non acquisisce i dati, sia più leggera della CHAIN, che li acquisisce. La letteratura, però, dichiara esattamente il contrario: CHAIN leggera e SETLL pesante.

SETGT Posizionati subito dopo la chiave (Set Greater Than)

Il codice operativo SETGT ha caratteristiche simili alla SETLL ma il posizionamento del cursore avviene immediatamente dopo il record avente la chiave uguale o precedente a quella specificata a fattore1. In tal modo, ad esempio, l'eventuale successiva lettura all'indietro (READP) acquisirebbe il record in questione.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     kanima          setgt     anima
```

Dopo una SETGT il cursore è sempre posizionato validamente, persino in caso di file vuoto.

Anche la SETGT accetta come chiave di posizionamento i valori speciali predefiniti *LOVAL (valore più basso) ed *HIVAL (valore più alto).

Per quanto la SETGT non acquisisca alcun record, se durante il posizionamento viene trovato almeno un record avente la chiave espressa nel fattore1, si accende l'indicatore risultante nella posizione Equal.

Ad esempio, se nel file esiste un record con la chiave del posizionamento, si accende l'indicatore 50 che altrimenti si spegne.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     kanima          setgt     anima                                    50
```

Seguono esempi di posizionamento del cursore basati sul file esemplificativo.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                     eval      kkfam='FELINO'
C     kanima          setgt     anima                                    50
 * Cursore risultante 9. Indicatore 50 acceso.
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                     eval      kkfam='INSETTO'
C     kanima          setgt     anima                                    50
 * Cursore risultante 9. Indicatore 50 spento.
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                     eval      kkfam='UCCELLO'
C                     eval      kkspe='MERLO'
C     kanima2         setgt     anima                                    50
 * Cursore risultante 16. Indicatore 50 acceso.
```

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                     eval      kkfam='PESCE'
C                     eval      kkspe='PERSICO'
C     kanima2         setgt     anima                                    50
 * Cursore risultante 12. Indicatore 50 spento.
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     *loval          setgt     anima                                    50
 * Cursore risultante 0. Indicatore 50 spento.
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     *hival          setgt     anima                                    50
 * Cursore risultante 18. Indicatore 50 spento.
```

23.4 Lettura dei record.

Per acquisire un record di database in memoria, l' RPG mette a disposizione dei codici operativi specifici.

Tutte le operazioni di lettura riempiono i campi in memoria con i valori presenti sul record letto. È tuttavia importante notare che, in caso di fallimento della lettura, i campi in memoria non vengono puliti automaticamente ma conservano il valore precedente. In particolare, dopo una lettura valida seguita da una fallita, i campi continuano a contenere i valori dell'ultima lettura valida.

READ Legge record successivo (Read next)

L'istruzione ha come prerequisito un posizionamento valido lungo il file.

Il codice operativo READ esegue la lettura del record successivo al cursore lungo il file di database. Non richiede alcuna chiave specificata a fattore1. Se l'operazione di lettura del record successivo fallisce (fine file), si accende l'indicatore risultante nella posizione Equal che, nel caso contrario si spegne.

In luogo dell'indicatore e con gli stessi significati, si può usare il flag %EOF.

Nell'esempio, se nel file esiste un record oltre la posizione del puntatore, si spegne l'indicatore 50 che altrimenti si accende.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                     read      anima                                    50
```

Nelle stesse condizioni, la seguente spegne %EOF.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                     read      anima
```

Dopo una READ fallita il cursore è sposizionato. Dopo una READ riuscita il cursore è posizionato dopo il record letto.

READP Legge record precedente (Read Previous)

L'istruzione ha come prerequisito un posizionamento valido lungo il file.

Il codice operativo READP esegue la lettura del record precedente alla posizione del puntatore lungo un file di database. Non richiede alcuna chiave specificata a fattore1. Se l'operazione di lettura del record successivo fallisce (inizio file), si accende l'indicatore risultante nella posizione Equal che, nel caso contrario, si spegne.

In luogo dell'indicatore e con gli stessi significati, si può usare il flag %EOF.

Nell'esempio, se nel file esiste un record prima della posizione del puntatore, si spegne l'indicatore 50 che altrimenti si accende.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                     readp     anima                                    50
```

Nelle stesse condizioni, la seguente spegne %EOF.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                     readp     anima
```

Dopo una READP fallita il cursore è sposizionato. Dopo una READP riuscita il cursore è posizionato prima del record letto.

READE Legge record successivo se uguale (Read next Equal)

L'istruzione ha come prerequisito un posizionamento valido lungo il file.

Il codice operativo READE esegue la lettura del record successivo alla posizione del puntatore lungo un file di database purché abbia chiave uguale a quella specificata sul fattore1. Se l'operazione di lettura del record successivo fallisce (fine file o chiave non uguale), si accende l'indicatore risultante nella posizione Equal che, nel caso contrario, si spegne.

In luogo dell'indicatore e con gli stessi significati, si può usare il flag %EOF.

Nell'esempio, se nel file, oltre la posizione del puntatore, esiste un record con la chiave cercata, si spegne l'indicatore 50 che altrimenti si accende.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     kanima         reade     anima                                    50
```

Nelle stesse condizioni, la seguente spegne %EOF.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     kanima         reade     anima
```

Dopo una READE fallita il cursore è sposizionato. Dopo una READE riuscita il cursore è posizionato dopo il record letto.

Comunemente l'operazione READE viene eseguita dopo un posizionamento SETLL per leggere il gruppo di record identificato dalla stessa chiave. Il gruppo viene percorso dal primo all'ultimo record. Si noti che la SETLL non porta intenzionalmente l'indicatore risultante di trovato perché, comunque, la READE dopo una SETLL fallita termina con un non trovato.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     kanima         setll     anima
C                    do        *hival
C     kanima         reade     anima                                    50
C   50               leave
 *                   ..........
C                    enddo
```

READPE Legge record precedente se uguale (Read Previous Equal)

L'istruzione ha come prerequisito un posizionamento valido lungo il file.

Il codice operativo READPE esegue la lettura del record precedente la posizione del puntatore lungo un file di database purché abbia chiave uguale a quella specificata sul fattore1. Se l'operazione di lettura del record precedente fallisce (inizio file o chiave non uguale), si accende l'indicatore risultante nella posizione Equal che, nel caso contrario, si spegne.

In luogo dell'indicatore e con gli stessi significati, si può usare il flag %EOF.

Nell'esempio, se nel file, immediatamente prima della posizione del puntatore, esiste un record con la chiave cercata, si spegne l'indicatore 50 che altrimenti si accende.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     kanima         readpe    anima                                    50
```

Nelle stesse condizioni, la seguente spegne %EOF.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     kanima         readpe    anima
```

Dopo una READPE fallita il cursore è sposizionato. Dopo una READPE riuscita il cursore è posizionato prima del record letto.

Comunemente l'operazione READPE viene eseguita dopo un posizionamento SETGT per leggere il gruppo di record identificato dalla stessa chiave. Il gruppo viene percorso dall'ultimo al primo record.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     kanima        setgt     anima
C                   do        *hival
C     kanima        readpe    anima                                    50
C   50              leave
 *                  ..........
C                   enddo
```

Si coglie l'occasione per esemplificare l'uso del flag %EOF in luogo dell'indicatore 50.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     kanima        setgt     anima
C                   do        *hival
C     kanima        readpe    anima
C                   if        %eof
C                   leave
C                   endif
 *                  ..........
C                   enddo
```

CHAIN Legge record con la chiave cercata (Chain)

L'istruzione non ha come prerequisito un posizionamento valido lungo il file.

Il codice operativo CHAIN esegue la lettura nel file di database di un record avente chiave uguale a quella specificata sul fattore1. Se l'operazione di reperimento del record fallisce (record inesistente), si accende l'indicatore risultante nella posizione Higth che, nel caso contrario si spegne.

Nell'istruzione successiva, se nel file esiste un record con la chiave cercata, si spegne l'indicatore 50 che altrimenti si accende.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     kanima        chain     anima                                    50
```

Dopo una CHAIN fallita il cursore è sposizionato. Dopo una CHAIN riuscita il cursore è posizionato dopo il record letto.

In luogo dell'indicatore e con gli stessi significati, ma rovesciati, si può usare il flag %FOUND, come in esempi successivi.

23.5 Blocco dei record letti.

Quando una operazione di lettura del database è effettuata felicemente su un file definito nelle specifiche F come UPDATE, avviene sia la lettura del record che il blocco dello stesso. Un record bloccato non è disponibile ad altre letture per update né da parte dello stesso lavoro né da parte di altri. Qualunque sia l'operazione eseguita, chiameremo questa *lettura per update*.

Il file si definirà nel programma RPG come segue.

```
FX  FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++
    Fanima01l  uf   e           k disk
```

È egualmente possibile tuttavia leggere il record senza che lo stesso rimanga bloccato. Per fare ciò si deve specificare l'estensione N sull'operazione di lettura. Se ne farà necessario uso nel ciclo di sincronia (25.6). Si esemplifica la CHAIN.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     kanima        chain(n)  anima                                    50
```

Questo tipo di lettura è utile nei programmi interattivi (ad esempio, una gestione anagrafica) per evitare di fermarsi a video con un record bloccato. La sequenza corretta delle operazioni è allora la seguente.

- Leggere senza blocco un record.
- Metterlo a video.

- Modificare i dati sul video.
- Rileggere lo stesso record con blocco.
- Controllare che il record sia uguale alla prima lettura.
- Ricalcare il record per riportarvi le modifiche del video e per sbloccarlo.

Se non si rilegge il record per update, il ricalco del record provoca un errore.

Se non si controlla che il record sia inalterato, si rischia di ricoprire gli aggiornamenti altrui senza averli visti e rovinando, ad esempio, i totali accumulati sul record.

23.6 Tentativo di lettura di record bloccati.

Quando, invece, una operazione di lettura del database è tentata su un file definito nelle specifiche F come UPDATE, ma nei confronti di un record già soggetto a blocco altrui, non avviene né la lettura del record né il blocco dello stesso.

Tentata la lettura, se il record in questione è occupato, il programma attende per il tempo previsto nell'attributo WAITRCD (Tempo massimo attesa record) del file di update. Scaduto il tempo massimo, il programma riceve un errore.

È possibile intercettare tale errore su tutte le istruzioni di lettura tramite l'indicatore Low e adottare le opportune contromisure. Si esemplifica la CHAIN.

Se il record cercato esiste ma è occupato, la seguente istruzione spegne l'indicatore 50 e accende l'indicatore 51.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     kanima        chain     anima                                  5051
```

Nella stessa situazione, volendo usare i flag, la seguente istruzione accende %FOUND e %ERR.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     kanima        chain     anima
```

23.7 Scrittura dei record.

WRITE Scrivi record

Il codice operativo WRITE provoca la nascita di un nuovo record dentro il file di database. Il nome del record è quello indicato nel campo del risultato. I campi in memoria che fanno parte del tracciato del record vengono trascritti in un nuovo record in coda al file fisico.

Il file a cui aggiungere il record può essere dichiarato di emissione, fisico o logico.

```
FX  FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++++
     * Emissione su fisico.
Fanima00f  o    e             disk
     * Emissione su fisico tramite logico.
Fanima01l  o    e             disk
```

Il file, fisico o logico, già dichiarato per altri scopi di immissione o di update, può ricevere anche l'aggiunta di record se contestualmente si specifica l'apposita opzione: una "A" in posizione 20.

```
**  ... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
FX  FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++++
     * Aggiunta su fisico di immissione.
Fanima00f  if a e             disk
     * Aggiunta su logico di immissione.
Fanima01l  if a e           k disk
     * Aggiunta su fisico di update.
Fanima00f  uf a e             disk
     * Aggiunta su logico di update.
Fanima01l  uf a e           k disk
```

La corretta generazione (aggiunta) di un record al file avviene seguendo sempre tre passi logici indispensabili.

- Pulizia preliminare di tutti i campi del record.
- Impostazione dei campi del record.
- Scrittura del record nel file.

Esempio di aggiunta di un record ad un file logico con chiave, dichiarato update full procedural con aggiunta.

```
FX  FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++++
    Fanima01l  uf a e           k disk
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Pulizia di tutti i campi del tracciato.
C                   clear                   anima
     * Valorizzazione di tutti i campi che non devono restare puliti.
C                   movel(p)  'CANIDE'      a1fam
C                   movel(p)  'LINCE'       a1spe
     * Scrittura del record nel database.
C                   write                   anima
```

Si può inoltre intercettare ed opportunamente gestire il fallimento (chiave doppia, file pieno) dell'operazione di scrittura di un nuovo record di database. Se l'operazione di scrittura del record fallisce, si accende l'indicatore risultante specificato nella posizione Low, altrimenti spento.

Non gestire l'errore provoca l'intervento del sistema che, non potendo eseguire l'operazione richiesta, propone un messaggio di interrogazione che interrompe l'elaborazione e mette in difficoltà l'utente interattivo o l'operatore di sistema, se batch.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                   write                   anima                   51
```

23.8 Ricalco dei record.

UPDATE Ricalca record

Il codice operativo UPDATE esegue l'aggiornamento di un record già esistente nel file di database e precedentemente letto per update. Il nome del record è indicato al fattore2.

Il corretto aggiornamento di un record al file avviene seguendo sempre tre passi logici indispensabili.

- Lettura e blocco preliminare del record su cui si vuole effettuare l'aggiornamento.
- Impostazione dei campi del record di cui si vuole effettuare l'aggiornamento.
- Ricalco del record nel file e sblocco.

Esempio di modifica di un record di un file logico con chiave (e quindi nell'opportuno file fisico), dichiarato update full procedural.

```
FX  FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++++
    Fanima01l  uf   e           k disk
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Valorizza la chiave di ricerca.
C                   movel(p)  'CANIDE'      kkfam
C                   movel(p)  'IENA'        kkspe
     * Cerca il record.
C     kanima2       chain     anima                         50
     * Se il record è trovato.
C                   if        not *in50
     *    Valorizza i campi da modificare.
C                   movel(p)  'LINCE'       a1spe
     *    Ricalca il record nel database.
C                   update                  anima
     * Se il record è trovato.
C                   endif
```

Si ripete facendo uso del flag %FOUND in luogo dell'indicatore 50.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Valorizza la chiave di ricerca.
C                    movel(p)   'CANIDE'       kkfam
C                    movel(p)   'IENA'         kkspe
     * Cerca il record.
C       kanima2      chain      anima
     * Se il record è trovato.
C                    if         %found
     *    Valorizza i campi da modificare.
C                    movel(p)   'LINCE'        a1spe
     *    Ricalca il record nel database.
C                    update                    anima
     * Se il record è trovato.
C                    endif
```

Nell'esempio, data l'esiguità del tracciato esemplificativo, invece di modificare un qualsiasi campo dati, si è modificata una chiave del file di update, cosa possibile in RPG ma non, ad esempio, in Cobol. La stessa attività in Cobol richiede un'altra apertura dello stesso file nello stesso programma.

La UPDATE aggiorna il contenuto di tutti i campi del record, anche di quelli per i quali non si sono impostati nuovi valori. Questa caratteristica richiede che si faccia attenzione a non sporcare i campi che non devono essere alterati, ad esempio, leggendo altri record dotati di campi omonimi del corrente (database non univoco) tra la lettura e il ricalco del record in questione.

Si noti altresì, che immediatamente dopo la UPDATE, il record viene disallocato ed è quindi immediatamente disponibile per altre elaborazioni, ad esempio una lettura per update già pendente.

Si può inoltre intercettare ed opportunamente gestire il fallimento (chiave doppia) dell'operazione di ricalco del record di database. Se ciò accade, si accende l'indicatore risultante specificato nella posizione Low, altrimenti spento.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                    update                    anima                    51
C                    if         *in51
C                    ..........
C                    endif
```

In luogo dell'indicatore è possibile usare il flag %ERR.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                    update                    anima
C                    if         %err
C                    ..........
C                    endif
```

Non gestire l'errore provoca l'intervento del sistema che, non potendo eseguire l'operazione richiesta, propone un messaggio di interrogazione che interrompe l'elaborazione e mette in difficoltà l'utente interattivo o l'operatore di sistema, se batch.

23.9 Cancellazione dei record.

DELETE Cancella record

Il codice operativo DELETE esegue la cancellazione di un record già esistente nel file di database e precedentemente letto per update. Il nome del record è indicato al fattore2.

Esempio di cancellazione di un record da un file logico con chiave (e quindi dall'opportuno file fisico), dichiarato update full procedural.

```
FX  FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++++
    Fanima01l  uf   e           k disk
```

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Valorizza la chiave di ricerca.
C                   movel(p)   'UCCELLO'     kkfam
C                   movel(p)   'MERLO'       kkspe
     * Cerca il record.
C     kanima2       chain      anima                          50
     * Se il record è trovato.
C                   if         not *in50
     *    Cancella il record dal database.
C                   delete                  anima
     * Se il record è trovato.
C                   endif
```

Nel caso precedente, la DELETE non può fallire.

In alternativa alla lettura preliminare, l'istruzione può direttamente indicare nel fattore1 la chiave del record da cancellare.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Valorizza la chiave di ricerca.
C                   movel(p)   'UCCELLO'     kkfam
C                   movel(p)   'MERLO'       kkspe
     * Cancella il record dal database.
C     kanima2       delete                  anima
```

In questo caso la cancellazione può fallire ma si può intercettare ed opportunamente gestire il fallimento (record allocato). Se ciò accade, si accende l'indicatore risultante specificato nella posizione Low, altrimenti spento.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     kanima2       delete                  anima                    51
```

Non gestire l'errore provoca l'intervento del sistema che, non potendo eseguire l'operazione richiesta, propone un messaggio di interrogazione che interrompe l'elaborazione e mette in difficoltà l'utente interattivo o l'operatore di sistema, se batch.

23.10 Sblocco dei record. 23C

UNLOCK Sblocca record

Il codice operativo UNLOCK esegue lo sblocco di un record precedentemente letto per update. Il nome del file contenente il record da sbloccare è indicato al fattore2.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                   unlock     anagr01l
```

EXCEPT Eccezione

Il codice operativo EXCEPT esegue le funzioni di emissione descritte nelle corrispondenti specifiche di emissione. Sul fattore2 della EXCEPT si indica un *nome di eccezione* che si riporta anche nel nome di eccezione della specifica di emissione di record.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                   except     ecce

O   ONomefile++DF..N01N02N03Excnam++++B++A++Sb+Sa+.............................
O anagr     e            ecce
```

La EXCEPT serve perciò a numerosi scopi e le funzioni di rilascio qui illustrate non ne esauriscono l'uso. Per gli altri usi si vedano i capitoli *File di stampa* (14) e *Specifiche RPG di emissione* (20).

Metodi di sblocco

Quando la lettura di un record avviene da un file aperto per update, e in lettura non si impedisce il bloccaggio del record, resta possibile sbloccare il record in cinque modi.

a. Eseguire l'update atteso mediante l'operazione UPDATE.
b. Eseguire l'update atteso mediante l'operazione EXCEPT mirata ad alcuni campi.
c. Rilasciare il record mediante l'operazione EXCEPT senza campi.

d. Rilasciare il record mediante l'operazione UNLOCK.
e. Leggere un altro record per update, ma solo se la nuova lettura ha buon esito.

Si definisce un file esemplificativo per il quale, negli esempi successivi, si suppongono tre chiavi valide: key, key1 e key2.

```
FX  FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++++
     * Anagrafico da ricalcare.
    Fanagr01l  uf   e           k disk
```

Si esemplifica una lettura senza blocco.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Legge un record non per update.
    C     key           chain(n)  anagr                            50
```

Si esemplifica un rilascio mediante UPDATE.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Legge e aggiorna il record letto.
    C     key           chain     anagr                            50
    C  n50              eval      descri='Nuova descrizione del record'
    C  n50              update    anagr
```

Si esemplifica un rilascio mediante EXCEPT mirato.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Legge e aggiorna il record letto.
    C     key           chain     anagr                            50
    C  n50              eval      descri='Nuova descrizione del record'
    C  n50              except    upd
O   ONomefile++DF..N01N02N03Excnam++++B++A++Sb+Sa+.............................
    Oanagr     e            upd
P   O.............N01N02N03Field+++++++++YB.End++PConstant/editword/DTformat++
    O                       descr
```

Si esemplifica un rilascio mediante EXCEPT vuoto.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Legge e rilascia il record letto.
    C     key           chain     anagr                            50
    C  n50              except    ril
O   ONomefile++DF..N01N02N03Excnam++++B++A++Sb+Sa+.............................
    Oanagr     e            ril
```

Si esemplifica un rilascio mediante UNLOCK.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Legge e rilascia il record letto.
    C     key           chain     anagr                            50
    C  n50              unlock    anagr01l
```

Si esemplifica un rilascio mediante un'altra lettura.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Legge il primo record.
    C     key1          chain     anagr                            50
     * Legge il secondo record e rilascia il primo.
    C     key2          chain     anagr                            50
```

23.11 File esemplificativo per gli esempi di Loop di Lettura.

Per la miglior comprensione dei successivi esempi di lettura, si ipotizza l'esistenza di un altro file logico da usare negli esempi successivi.

Nome Record	Nome File Logico			Testo File
LIBRI	**LIBRI01L**			**Libri**
Nome Campo	**Key**	**Dim**	**Tipo**	**Descrizione Campo**
LIANNU		1	Alf	Annullamento
LITIPO		1	Alf	Tipo Libro
LICODI	**K2**	6	Alf	Codice Libro
LIDESC		50	Alf	Descrizione Libro

Si ipotizza altresì il contenuto in dati del medesimo.

Libri				
RRN	**Ann**	**Tipo**	**Codice**	**Descrizione**
1		FS	A10	Abissi d'Acciaio
2		FS	A20	Il Crollo della Galassia Centrale
3	A	FS	A30	Odissea nello Spazio
4		FS	A40	Io, Robot
5		FU	D10	Tex Willer
6		FU	D20	Sturm Truppen
7	A	FU	D30	Dylan Dog
8		FU	D40	Bibì e Bibò
9	A	PO	B10	Odissea
10		PO	B20	Divina Commedia
11		PO	B30	Iliade
12	A	RO	C10	Feria d'Agosto.
13		RO	C20	I Promessi Sposi
14		RO	C30	Il Nome della Rosa
15		RO	C40	Il Prato in Fondo al Mare

Per completezza, si menziona la specifica necessaria in RPG per l'uso del file.

```
FX  FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++++
    Flibri01l  if   e           k disk
```

23.12 Loop di Lettura in avanti.

Si esemplifica la lettura in avanti dell'intero file.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
    C     *loval        setll     rcd
    C                   do        *hival
    C                   read      libri                                  50
    C   50              leave
    C                   if        liannu='A'
    C                   iter
    C                   endif
    C                   write     stampa
    C                   enddo
```

Meglio ancora se nella forma commentata ed autoesplicativa. Nel seguito del capitolo si farà esclusivamente uso di tale forma.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Si posiziona all'inizio del file.
C      *loval        setll     libri
     * Elabora tutti i record del file.
C                    do        *hival
     *    Legge un record in avanti.
C                    read      libri                                  50
     *    Se non ce ne sono altri, abbandona.
C    50              leave
     *    Se il record è annullato, ricicla.
C                    if        liannu='A'
C                    iter
C                    endif
     *    Scrive il record in stampa.
C                    write     stampa
     * Elabora tutti i record del file.
C                    enddo
```

La routine stampa l'elenco di tutti i record validi del file, in ordine ascendente.

RRN	Ann	Tipo	Codice	Descrizione
1		FS	A10	Abissi d'Acciaio
2		FS	A20	Il Crollo della Galassia Centrale
4		FS	A40	Io, Robot
5		FU	D10	Tex Willer
6		FU	D20	Sturm Truppen
8		FU	D40	Bibì e Bibò
10		PO	B20	Divina Commedia
11		PO	B30	Iliade
13		RO	C20	I Promessi Sposi
14		RO	C30	Il Nome della Rosa
15		RO	C40	Il Prato in Fondo al Mare

23.13 Loop di Lettura all'indietro.

Si esemplifica la lettura all'indietro dell'intero file.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Si posiziona alla fine del file.
C      *hival        setgt     libri
     * Elabora tutti i record del file.
C                    do        *hival
     *    Legge un record all'indietro.
C                    readp     libri                                  50
     *    Se non ce ne sono altri, abbandona.
C    50              leave
     *    Se il record è annullato, ricicla.
C                    if        liannu='A'
C                    iter
C                    endif
     *    Scrive il record in stampa.
C                    write     stampa
     * Elabora tutti i record del file.
C                    enddo
```

La routine stampa l'elenco di tutti i record validi del file, in ordine discendente.

RRN	Ann	Tipo	Codice	Descrizione
15		RO	C40	Il Prato in Fondo al Mare
14		RO	C30	Il Nome della Rosa
13		RO	C20	I Promessi Sposi
11		PO	B30	Iliade
10		PO	B20	Divina Commedia
8		FU	D40	Bibì e Bibò
6		FU	D20	Sturm Truppen
5		FU	D10	Tex Willer
4		FS	A40	Io, Robot
2		FS	A20	Il Crollo della Galassia Centrale
1		FS	A10	Abissi d'Acciaio

23.14 Loop di Lettura per chiave uguale in avanti.

Si esemplifica la lettura in avanti dei record validi con la prima chiave uguale a "FU", ricevuta dalla seguente routine nel campo PPTIPO.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Definisce la chiave di lettura per il file.
C     klibri        klist
C                   kfld                    pptipo
C     *like         define    litipo        pptipo
     * Si posiziona all'inizio del gruppo.
C     klibri        setll     libri
     * Elabora tutti i record del gruppo.
C                   do        *hival
     *    Legge un record del gruppo in avanti.
C     klibri        reade     libri                                  50
     *    Se non ce ne sono altri, abbandona.
C   50              leave
     *    Se il record è annullato, ricicla.
C                   if        liannu='A'
C                   iter
C                   endif
     *    Scrive il record in stampa.
C                   write     stampa
     * Elabora tutti i record del gruppo.
C                   enddo
```

La routine stampa l'elenco dei record validi del solo gruppo scelto, in ordine ascendente.

RRN	Ann	Tipo	Codice	Descrizione
5		FU	D10	Tex Willer
6		FU	D20	Sturm Truppen
8		FU	D40	Bibì e Bibò

23.15 Loop di Lettura per chiave uguale all'indietro.

Si esemplifica la lettura all'indietro dei record con la prima chiave uguale a "FU", ricevuta dalla seguente routine nel campo PPTIPO.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Definisce la chiave di lettura per il file.
C     klibri        klist
C                   kfld                    pptipo
C     *like         define    litipo        pptipo
     * Si posiziona alla fine del gruppo.
C     klibri        setgt     libri
     * Elabora tutti i record del gruppo.
C                   do        *hival
     *    Legge un record del gruppo all'indietro.
C     klibri        readpe    libri                                 50
     *    Se non ce ne sono altri, abbandona.
C   50              leave
     *    Se il record è annullato, ricicla.
C                   if        liannu='A'
C                   iter
C                   endif
     *    Scrive il record in stampa.
C                   write     stampa
     * Elabora tutti i record del gruppo.
C                   enddo
```

La routine stampa l'elenco dei record validi del solo gruppo scelto, in ordine discendente.

RRN	Ann	Tipo	Codice	Descrizione
8		FU	D40	Bibì e Bibò
6		FU	D20	Sturm Truppen
5		FU	D10	Tex Willer

23.16 Loop di Lettura a salti in avanti.

Si esemplifica la lettura in avanti di tutti i record validi di inizio gruppo presenti nel file.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Definisce la chiave di posizionamento a fine gruppo.
C     klibri        klist
C                   kfld                    litipo
     * Si posiziona all'inizio del file.
C     *loval        setll     libri
     * Elabora tutti i record del file.
C                   do        *hival
     *    Legge un record in avanti.
C                   read      libri                                 50
     *    Se non ce ne sono altri, abbandona.
C   50              leave
     *    Se il record è annullato, ricicla.
C                   if        liannu='A'
C                   iter
C                   endif
     *    Scrive il record in stampa.
C                   write     stampa
     *    Si posiziona oltre il gruppo di cui fa parte il record corrente.
C     klibri        setgt     libri
     *    Ricicla.
C                   iter
     * Elabora tutti i record del file.
C                   enddo
```

La routine stampa l'elenco dei record validi di inizio gruppo del file, in ordine ascendente.

RRN	Ann	Tipo	Codice	Descrizione
1		FS	A10	Abissi d'Acciaio
5		FU	D10	Tex Willer
10		PO	B20	Divina Commedia
13		RO	C20	I Promessi Sposi

23.17 Loop di Lettura a salti all'indietro. Variante 1.

Si esemplifica la lettura all'indietro di tutti i record validi di **fine** gruppo presenti nel file.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Definisce la chiave di posizionamento a inizio gruppo.
C     klibri        klist
C                   kfld                    litipo
     * Si posiziona alla fine del file.
C     *hival        setgt     libri
     * Elabora tutti i record del file.
C                   do        *hival
     *    Legge un record all'indietro.
C                   readp     libri                                  50
     *    Se non ce ne sono altri, abbandona.
C   50              leave
     *    Se il record è annullato, ricicla.
C                   if        liannu='A'
C                   iter
C                   endif
     *    Scrive il record in stampa.
C                   write     stampa
     *    Si posiziona prima del gruppo di cui fa parte il record corrente.
C     klibri        setll     libri
     *    Ricicla.
C                   iter
     * Elabora tutti i record del file.
C                   enddo
```

La routine stampa l'elenco dei record validi di fine gruppo del file, in ordine discendente.

RRN	Ann	Tipo	Codice	Descrizione
15		RO	C40	Il Prato in Fondo al Mare
11		PO	B30	Iliade
8		FU	D40	Bibì e Bibò
4		FS	A40	Io, Robot

23.18 Loop di Lettura a salti all'indietro. Variante 2.

Si esemplifica la lettura all'indietro di tutti i record validi di **inizio** gruppo presenti nel file.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Definisce la chiave di posizionamento a inizio gruppo.
C     klibri        klist
C                   kfld                    litipo
     * Si posiziona alla fine del file.
C     *hival        setgt     libri
     * Elabora tutti i record del file.
C                   do        *hival
     *    Legge un record all'indietro.
C                   readp     libri                                  50
     *    Se non ce ne sono altri, abbandona.
C   50              leave
     *    Se il record è annullato, ricicla.
C                   if        liannu='A'
C                   iter
C                   endif
     *    Si posiziona prima del gruppo di cui fa parte il record corrente.
C     klibri        setll     libri
     *    Elabora tutti i record del gruppo.
C                   do        *hival
     *       Legge un record del gruppo in avanti.
C     klibri        reade     libri                                  50
     *       Se non ce ne sono altri, abbandona.
C   50              leave
     *       Se il record è annullato, ricicla.
C                   if        liannu='A'
C                   iter
C                   endif
     *       Scrive il record in stampa.
C                   write     stampa
     *       Abbandona.
C                   leave
     *    Elabora tutti i record del gruppo.
C                   enddo
     *    Si posiziona prima del gruppo di cui fa parte il record corrente.
C     klibri        setll     libri
     *    Ricicla.
C                   iter
     * Elabora tutti i record del file.
C                   enddo
```

La routine stampa l'elenco dei record validi di inizio gruppo del file, in ordine discendente.

RRN	Ann	Tipo	Codice	Descrizione
15		RO	C40	Il Prato in Fondo al Mare
11		PO	B30	Iliade
8		FU	D40	Bibì e Bibò
4		FS	A40	Io, Robot

Si osservi che una migliore minutazione, con la frammentazione in subroutine, aumenta la leggibilità dell'esempio e permette di riutilizzare uno schema già illustrato almeno per una parte del problema.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Definisce la chiave di posizionamento a inizio gruppo.
C     klibri        klist
C                   kfld                    litipo
     * Si posiziona alla fine del file.
C     *hival        setgt     libri
     * Elabora tutti i record del file.
C                   do        *hival
     *    Legge un record all'indietro.
C                   readp     libri                                  50
     *    Se non ce ne sono altri, abbandona.
C   50              leave
     *    Se il record è annullato, ricicla.
C                   if        liannu='A'
C                   iter
C                   endif
     *    Stampa il primo record del gruppo corrente.
C                   exsr      prtfst
     *    Si posiziona prima del gruppo di cui fa parte il record corrente.
C     klibri        setll     libri
     *    Ricicla.
C                   iter
     * Elabora tutti i record del file.
C                   enddo
     *---------------------------------------------------------------------
     * Stampa il primo record del gruppo corrente.
C     prtfst        begsr
     * Si posiziona prima del gruppo di cui fa parte il record corrente.
C     klibri        setll     libri
     * Elabora tutti i record del gruppo.
C                   do        *hival
     *    Legge un record del gruppo in avanti.
C     klibri        reade     libri                                  50
     *    Se non ce ne sono altri, abbandona.
C   50              leave
     *    Se il record è annullato, ricicla.
C                   if        liannu='A'
C                   iter
C                   endif
     *    Scrive il record in stampa.
C                   write     stampa
     *    Abbandona.
C                   leave
     * Elabora tutti i record del gruppo.
C                   endsr
C                   enddo
     *---------------------------------------------------------------------
```

24 Strutturazione di un RpgIle interattivo. (24C)

Concepimento, nascita e crescita di un flusso di gestione di un file anagrafico.

I flussi qui riportati sono dei semplici esempi e non l'unico modo di risolvere il problema. Tuttavia si raccomanda un esame attento della soluzione proposta che è il risultato di una lunga serie di approssimazioni successive.

Lo schema proposto è un elenco di spunti e riflessioni e non è in realtà perfettamente sufficiente a stendere un programma funzionante. Lo schema completo risulterebbe troppo tecnico per lo scopo del capitolo corrente. Per questo si rimanda all'esempio di *Gestione anagrafico* (38.2).

24.1 Tecniche di minutazione.

L'idea centrale.

La programmazione ha qualche non casuale simiglianza con il linguaggio della comunicazione verbale. Naturalmente l'accostamento è tra lo scrivere programmi e lo scrivere pensieri. A parte la logica più semplice ma più stringente della programmazione, le due attività hanno in comune un'intenzione iniziale che si sviluppa prima in un progetto e poi in una serie di particolari realizzativi.

La simiglianza non è solo dei costrutti logici. Il modo giusto di creare i due oggetti è egualmente simile: brevi frasi, solide e consequenziali, corredate da tutte le affermazioni di contorno al corpo centrale necessarie e sufficienti alla comprensione. La frase iniziale nasce in semplicità ed ha un'evidenza anche intuitiva ma si amplia subito di quel tanto che basta a renderla inequivocabile e viene corredata di premesse e conseguenze che la riallacciano ai presupposti iniziali e la conducono, dall'altro lato, alle conclusioni necessarie.

Un programma nasce bene intorno ad una idea che da iniziale diventa centrale e la sua minutazione avviene in ordine di importanza discendente, sparpagliando istruzioni prima, dentro e dopo l'idea principale.

Ordine di minutazione.

La scrittura di un sorgente non avviene quindi normalmente in maniera consecutiva, cioè nello stesso ordine in cui le istruzioni figureranno nell'edizione finale, ma in ordine di importanza o, meglio di concepimento.

Ad esempio, le varie parti della routine SRW1 si scrivono in un ordine molto diverso da quello che appare a routine completata.

- Si inizia con la EXFMT.
- La EXFMT viene circondata dalla sequenza DO/ENDDO.
- Si introduce l'abbandono del ciclo dopo la EXFMT se premuto F3.
- Si scrive l'acquisizione del record prima della EXFMT.
- Si inserisce la chiamata della routine DBW1 dopo l'acquisizione del record.
- Si condiziona l'acquisizione con la richiesta di acquisizione e si provvede anche ad eccitarla prima della DO.
- ...

La minutazione produce in successione degli schemi sempre più ricchi che risultano strutturalmente validi anche prima di essere completi. Questa cura nel conservare la funzionalità del programma anche durante lo sviluppo permette di minutare a memoria, compilare e provare anche prima del completamento e diminuisce drasticamente il tempo di debug.

Ordine di minutazione delle sequenze.

Dopo aver scritto l'operazione di apertura di una sequenza (BEGSR, DO, IF, ...), si consiglia di scrivere subito in anticipo anche quella di chiusura (ENDSR, ENDDO, ENDIF, ...) per evitare di dimenticarsene più tardi.

Naturalmente, l'istruzione pendente troverà il suo posto definitivo dopo aver flottato quanto basta mentre le si antepongono le istruzioni che vanno posizionate in mezzo alla sequenza.

Nei flussi seguenti si invita a scrivere le chiusure in anticipo ma se ne rammenta la presenza anche nel luogo opportuno.

24.2 Flusso.

Con il termine "flusso" si intendeva in illo tempore la diagrammazione della logica di un programma a suon di rettangoli (rappresentanti sequenze di istruzioni senza salti), di rombi (decisioni di salto), di linee in uscita dai rombi (salti su decisione), di linee in uscita dai rettangoli (giustapposizione di istruzioni), di ovali (punti di ingresso o di uscita).

Tale diagrammazione non è mai stata in uso in ambito S/38 e AS/400 a causa dei flussi particolarmente semplici che la frammentazione dei programmi permette e della linearità del flusso interno ai programmi così frammentati.

L'evoluzione è stata irreversibile dal momento della sparizione totale dell'uso del salto, sostituita dalle tecniche di programmazione che fanno uso di sequenze di DO/ENDDO, IF/ELSE/ENDIF, SELECT/WHEN/OTHER/ENDSL e di subroutine. Usa chiamare tale tecnica "programmazione strutturata" ma essa risulta emergere più dal buon senso comune dei programmatori che non da testi teorici.

A proposito di diagrammi, per un lungo periodo i corsi elementari IBM spiegavano una tecnica che, per quanto ne so, nessuno ha mai preso sul serio: la diagrammazione di Jackson. In corrispondenza di tale scoperta, smisi nell'80 di spedire scagnozzi ai corsi.

Il massimo della follia fu raggiunto, come mi raccontava certa Rustichelli che sostituivo in Esab, nei gruppi di sviluppo delle ADB.

I programmatori erano costretti a usare un tool che traduceva quella diagrammazione in un precompilato RPG pieno di GOTO, visto che tutto questo avveniva per il futuro 38 ma lavorando sul 34 disponibile.

I programmi che ne derivavano potevano essere capiti solo a partire dal metalinguaggio di partenza ma, (udite, udite) il tool sparì prima del test e i poveri disgraziati misero a punto non il metalinguaggio ma il precompilato.

Se qualcuno ha visto le ADB del 38 sa che schifezza ne uscì.

Quasi sempre i programmi funzionavano ma pochi sapevano come mai.

Per questa ragione non parlerò mai di Jackson.

24.3 Prerequisiti.

Prima di stendere il flusso di un RPG interattivo di gestione anagrafico è indispensabile avere concepito il file fisico ed i logici relativi, in particolare un logico con chiave unica.

In secondo luogo si richiede l'abbozzo dei video da usarsi nella gestione.

Per terza cosa si richiede la logica di avvicendamento dei video: deciso il primo video in uscita, da ciascuno dei video disegnati occorre indicare le azioni che verranno intraprese per passare agli altri video e che effetto ci sarà sui file di database.

24.4 Frammentazione in routine del flusso.

L'approccio migliore alla costruzione di un flusso è quello che mette intorno ad ogni video una routine e che per prima routine scrive quella che gestisce il video di ingresso nel programma.

Ogni routine ha una sua logica particolare ma con molti punti in comune con altre.

Si distinguono poi le routine di flusso e quelle di gestione dei dati. Le prime passano quasi inalterate da un programma all'altro mentre le seconde cambiano ogni volta tutto il loro contenuto.

È importante individuare con chiarezza quali parametri fornire ad una subroutine e quali ricavarne. Non tutta l'interfaccia è esplicitata come nella chiamata ad un programma esterno poiché tutti i campi di un programma sono disponibili all'interno della routine (salvo l'uso di Procedure, di cui questo corso non tratta). Tuttavia, pur senza ridondanti palleggi di valori tra variabili, occorre avere sempre chiara l'interfaccia veramente in uso, esplicitandola quando si corra il rischio di equivoco.

24.5 Denominazione delle routine.

Si consiglia un dizionarietto adatto alla denominazione delle routine per semplificare la comprensione delle funzioni svolte da ciascuna.

SRW0	Gestisce il video guida W0.
SRW1	Gestisce il video dati W1 per modifica.
SRW1A	Gestisce il video dati W1 per aggiunta.
DEW1	Decodifica il video dati W1.
CTW1	Controlla gli errori del video dati W1.
DBW1	Sposta i dati da database a video W1.
W1DB	Sposta i dati da video W1 a database.
ELW1	Visualizza gli elenchi del video dati W1.

Naturalmente i nomi son di fantasia, ma si suggerisce che nomi più lunghi somiglierebbero ai nomi dei campi di database e, più lunghi ancora, ai nomi dei campi di comodo. Salvo casi veramente particolari, non chiamerei mai "Donna che mi ha dato la vita" quella persona che posso chiamare "Mamma". Cioè, i nomi brevi distinguono tanto bene le routine da poterli usare come acronimi identificativi per intendersi tra colleghi programmatori dentro uno standard già altrimenti complesso e faticoso.

24.6 Logica della main routine.

La routine principale si limita a chiamare la routine del primo video (EXSR SRW0) e a terminare subito con la richiesta di chiusura (SETON LR).

In effetti la tecnica di frammentazione non necessiterebbe di una scelta così drastica: si potrebbe ragionevolmente conservare nella main routine tutto il trattamento del primo video. Il problema è che un programma non rispetta solo la propria logica interna ma anche quella del pacchetto di programmi di cui fa parte

24.7 Logica della routine SRW0 = Gestisce il video guida W0.

24.8 Logica della routine SRW1 = Gestisce il video dati W1 per modifica.

La routine di flusso SRW1 è la madre di tutte le routine di gestione dei video perché è la più completa del gruppo.

La routine SRW1 gestisce il video dati W1 che riporta tutti i dati presenti su un record anagrafico e ne permette la visualizzazione, la decodifica e la modifica controllata.

- La routine SRW1 riceve dalla chiamante la chiave del record da gestire (KANAGR).
- Se necessario, la routine restituirà un messaggio di errore alla chiamante (MSG).
- Si scrive l'istruzione di inizio routine (BEGSR) e, contestualmente, anche quella di fine (ENDSR).
- Si prenota il caricamento dei dati accendendo un flag di richiesta (SRW1LO).
- Si costituisce un ciclo infinito scrivendo l'istruzione di inizio (DO *HIVAL) e, contestualmente, anche quella di fine (ENDDO).

- Se richiesto il caricamento dei dati.
    - Si sprenota il caricamento dei dati spegnendo il flag di richiesta (SRW1LO).
    - Si recupera il record da presentare sulla base della chiave che la routine riceve dalla chiamante (KANAGR CHAIN ANAGR).
    - Se il record non esiste, si restituisce un messaggio di errore alla routine chiamante e si abbandona il ciclo infinito.
    - Si esegue la routine di spostamento dei dati da record di database a video (DBW1).
    - Si esegue la routine di decodifica (DEW1).
- Se corre un errore.
    - Si spegne l'errore generico (SETOFF 60).
    - Si emette il video per decodifica (WRITE W1).
    - Si riaccende l'errore generico (SETON 60).
- Si esegue l'emissione/rilettura del video (EXFMT W1). Quando non è presente un errore, questa è l'unica emissione necessaria. Altrimenti questa emissione va a ricalco della WRITE W1 vista sopra ([1]).
- Si testano le richieste di fine (F3) e di video precedente (F12) e, se richiesto, si esce dal ciclo infinito.
- Si esegue la routine Elenchi (ELW1). Se richiesto almeno un elenco, si prenota il riciclo accendendo l'indicatore di errore generico *IN60.
- Si esegue la routine di decodifica (DEW1) dei codici presenti sul video, assumendo decodifiche a base di interrogativi per i codici errati.
- Se è prenotato il riciclo, si ricicla.
- Si esegue la routine di controllo (CTW1) dei dati su video.
- Se la routine riscontra un errore su un campo del video, si ricicla con errore. Ovvero.
    - Si predispone un messaggio da portare a video e si accende l'errore specifico del campo (un indicatore da *IN61 a *IN90) e l'errore generico *IN60.
    - Si ricicla.
- Si recupera il record per ricalcarlo.
- Se il record non esiste più, si ricicla con errore (come sopra).
- Se il record non è disponibile, si ricicla con errore.
- Se il record è cambiato, si ricicla con errore.
- Si esegue la routine di spostamento dei dati da video al record di database (W1DB).
- Si ricalca il record (UPDATE ANAGR).
- Se chiave doppia, si ricicla con errore.
- Si chiude il ciclo infinito (ENDDO).
- Si conclude la routine (ENDSR).

24.9 Logica della routine SRW1A = Gestisce il video dati W1 per aggiunta.

24.10 Logica della routine DEW1 = Decodifica il video dati W1.

La routine dati DEW1 accosta ad ogni campo codice del video l'opportuno campo descrizione pure a video. La routine non si interrompe se qualche decodifica è in errore ma continua fino ad esaurire tutte le decodifiche.

Normalmente, per ogni codice si consulta l'anagrafico a chiave unica che lo definisce e se ne ricava la descrizione da riportare in video accanto al codice stesso.

[1] A giustificazione di questa stranezza tecnica, si sappia che la EXFMT di un video con l'attivazione di una ERRMSG o ERRMSGID provoca l'uscita solo del messaggio di errore e dei caratteri di controllo di inizio dei campi. Se tale emissione avviene su una base diversa dal video medesimo, avviene una devastazione dell'immagine a video. Ne segue la cautela illustrata: se sta per avvenire la EXFMT di un certo video con ERRMSG(ID), si premette una WRITE senza ERRMSG(ID) del medesimo video.

In altri casi, la decodifica può avvenire usando come anagrafico una schiera, nella quale, a inizio programma, si sono caricati i pochi valori necessari per una particolare decodifica molto ricorrente.

Altre volte si tratta di decodificare un flag a pochi stati (S = Sì, N = No) tramite una sequenza IF o SELECT con uso di costanti a programma.

Normalmente il codice vuoto provoca una decodifica vuota e non, come in altre efferate logiche, una decodifica di errore.

La decodifica non riuscita di un codice valorizzato provoca una descrizione di tutti interrogativi, utile più tardi nella routine di controllo errori. Da schivare gli asterischi, che per la loro abusata specie e mancata specializzazione, han finito per non significare più nulla.

- La routine DEW1 riceve dalla chiamante tutti i campi digitati o, comunque, digitabili presenti sul video W1. In particolare, tra questi, riceverà i codici da decodificare.
- La routine restituirà tutte le decodifiche a video dei codici ricevuti, anche se alcune non fossero destinate ad essere viste.
- Si scrive l'istruzione di inizio routine (BEGSR) e, contestualmente, anche quella di fine (ENDSR).
- Per ciascuna decodifica su anagrafico.
    - Pulisce la decodifica.
    - Se il codice è valorizzato.
        - Cerca il codice sull'anagrafico.
        - Se il codice manca.
            - Assume interrogativi nella decodifica.
        - Se il codice esiste ed è valido.
            - Decodifica prelevando il dato dal tracciato dell'anagrafico.
- Per ciascuna decodifica in schiera.
    - Pulisce la decodifica.
    - Se il codice è valorizzato.
        - Inizializza a 1 l'indice di ricerca.
        - Cerca il codice nella schiera dei codici facendo uso dell'indice di ricerca.
        - Se il codice manca.
            - Assume interrogativi nella decodifica.
        - Se il codice esiste.
            - Decodifica prelevando il dato dalla schiera delle decodifiche tramite l'indice trovato.
- Per ciascuna decodifica con l'uso di SELECT.
    - Pulisce la decodifica.
    - Se il codice è valorizzato.
        - Se il codice è uguale al primo valore di confronto.
            - Assume il primo valore di decodifica nella decodifica.
        - Se il codice è uguale al secondo valore di confronto.
            - Assume il secondo valore di decodifica nella decodifica.
        - ...
            - ...
        - Se non è uguale a nessuno dei codici precedenti.
            - Assume interrogativi nella decodifica.
- Si conclude la routine (ENDSR).

24.11 Logica della routine CTW1 = Controlla gli errori del video dati W1.

La routine dati CTW1 controlla tutti gli errori, sia formali che di decodifica, che si presentano durante la modifica o l'aggiunta di un record anagrafico mentre il suo contenuto si trova sul video. Il controllo, quindi, avviene sui campi del video e non sui campi del record di database.

- La routine CTW1 riceve dalla chiamante tutti i campi presenti sul video W1, compresi codici e decodifiche. In realtà deve ricevere anche codici e decodifiche eventualmente presenti sul record ma non sul video: in questo caso tali codici e decodifiche saranno preferibilmente presenti sul record video come campi nascosti.
- La routine restituirà il testo del messaggio di errore, l'indicatore di errore generico acceso e, pure accesi, uno o più indicatori di errore di campo.
- Si scrive l'istruzione di inizio routine (BEGSR) e, contestualmente, anche quella di fine (ENDSR).
- Si costituisce un ciclo singolo scrivendo l'istruzione di inizio (DO) e, contestualmente, anche quella di fine (ENDDO).
- Se un campo da controllare è vuoto ma obbligatorio.
    - Si accende l'indicatore di errore generico *IN60.
    - Si accende un indicatore di errore specifico di campo da *IN61 a *IN90.
    - Si riempie il messaggio MSG con un testo adatto: Dato obbligatorio.
    - Si abbandona il ciclo singolo.
- Se la decodifica di un campo contiene tutti interrogativi.
    - Si accende l'indicatore di errore generico *IN60.
    - Si accende un indicatore di errore specifico di campo da *IN61 a *IN90.
    - Si riempie il messaggio MSG con un testo adatto: Manca in anagrafico.
    - Si abbandona il ciclo singolo.
- Se un flag S/N o 0/1 non già decodificato ha un valore non previsto.
    - Si accende l'indicatore di errore generico *IN60.
    - Si accende un indicatore di errore specifico di campo da *IN61 a *IN90.
    - Si riempie il messaggio MSG con un testo adatto: Permessi i valori x/y.
    - Si abbandona il ciclo singolo.
- Se si presenta un qualsiasi altro errore.
    - Si accende l'indicatore di errore generico *IN60.
    - Si accende un indicatore di errore specifico di campo da *IN61 a *IN90.
    - Si riempie il messaggio MSG con un testo adatto: Errore tal dei tali.
    - Si abbandona il ciclo singolo.
- Si chiude il ciclo singolo (ENDDO).
- Si conclude la routine (ENDSR).

Interessa notare che i vari controlli di errore godono di assoluta autonomia e non si annidano l'uno nell'altro con condizionamenti complessi. Questa tecnica semplifica moltissimo sia l'aggiunta che la rimozione di singoli controlli nel corso delle manutenzioni successive, specialmente di quelle, numerose, che si rendono necessarie durante la prima messa a punto del programma.

24.12 Logica della routine DBW1 = Sposta i dati da database a video W1.

La routine dati DBW1 sposta tutti i dati dal record anagrafico ai corrispondenti campi del video.

- La routine DBW1 riceve dalla chiamante tutti i campi presenti sul record anagrafico.
- La routine restituirà il riempimento di tutti i campi del video corrispondenti ai campi del record anagrafico datore.
- Si scrive l'istruzione di inizio routine (BEGSR) e, contestualmente, anche quella di fine (ENDSR).

- Si muove ogni singolo campo del record anagrafico di database nel corrispondente campo del video.
- Si conclude la routine (ENDSR).

24.13 Logica della routine W1DB = Sposta i dati da video W1 a database.

La routine dati W1DB sposta tutti i dati dal video ai corrispondenti campi del record anagrafico.

La routine corrente è perfettamente simmetrica alla DBW1.

- La routine W1DB riceve dalla chiamante tutti i campi presenti sul video.
- La routine restituirà il riempimento di tutti i campi del record anagrafico corrispondenti ai campi del video datore.
- Si scrive l'istruzione di inizio routine (BEGSR) e, contestualmente, anche quella di fine (ENDSR).
- Si muove ogni singolo campo del video nel corrispondente campo del record anagrafico di database.
- Si conclude la routine (ENDSR).

24.14 Logica della routine ELW1 = Visualizza gli elenchi del video dati W1.

La routine dati ELW1 esegue le richieste di elenco formulate sul video chiamando, per ogni codice esaminato, l'apposito programma di elenco. Sull'elenco è permessa la scelta di un solo codice che viene riportato sul video nel campo codice di partenza della richiesta.

- La routine ELW1 riceve dalla chiamante tutti i campi codice presenti sul video.
- Si scrive l'istruzione di inizio routine (BEGSR) e, contestualmente, anche quella di fine (ENDSR).
- Se il campo in esame è dotato di un interrogativo in prima posizione o contiene il cursore video al momento dell'eventuale comando F2/F14 di richiesta elenco.
    - Prenota il riciclo accendendo l'indicatore di riciclo *IN60 che altrove svolge la funzione di indicatore di errore generico. Si nota che l'uso dell'indicatore 60 è dovuto alla sua disponibilità in questa fase dell'elaborazione ed al fatto che anche nella sua funzione principale di errore generico serve, tra l'altro, a provocare proprio un riciclo.
    - Se presente l'interrogativo nel codice.
        - Estrae il codice di partenza e pulisce il codice a video, rovinato dalla digitazione dell'interrogativo.
    - Se non presente l'interrogativo nel codice.
        - Se digitato F2 (Elenco dall'inizio)
            - Assume come inizio blank.
        - Se digitato F14 (Elenco dal codice attuale).
            - Assume come inizio il codice corrente.
    - Chiama il programma esterno di elenco passandogli il codice di partenza e ricevendo di ritorno il codice scelto.
    - Se il codice di ritorno è valorizzato, lo sostituisce
- Si ripete l'ambaradan per ogni campo, fino ad esaurimento.
- Si conclude la routine (ENDSR).

24.15 Copia di routine precostituite mediante /COPY.

Se la routine da eseguire nel corso di un programma ricorre in realtà inalterata in molti programmi, è dotata di un'interfaccia ben distinta e non utilizza altre variabili oltre alle proprie e a quelle dell'interfaccia, allora la routine può essere minutata in un sorgente separato e inclusa nel programma al momento della compilazione.

L'inclusione avviene grazie alla direttiva di compilazione /COPY, presente nell'RPG sin da notte dei tempi (Sistema 3 modello 10).

```
**  ... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
     * Forma estesa.
     /COPY FileSrc(LibSrc),MbrSrc
     * Forma breve.
     /COPY FileSrc,MbrSrc
```

Dove: /COPY =Direttiva di inclusione di un altro sorgente.
FileSrc =Nome del file contenente il membro sorgente.
LibSrc =Nome della libreria contenente il file sorgente.
MbrSrc =Nome del membro sorgente.

La forma breve è permessa solo se il file sorgente è presente in lista librerie al momento della compilazione. Si nota che in lista librerie si trova il primo dei file omonimi presenti in elenco. Occorre quindi porre particolare attenzione alla lista librerie di compilazione. D'altra parte è delittuoso includere in un sorgente un nome di libreria perché ne diminuisce la portabilità.

La /COPY acchiappa tutte le istruzioni del sorgente indirizzato e le riporta nella compilazione nell'esatto punto della /COPY. Nessuna modifica viene inflitta ai sorgenti coinvolti.

Per storia, si ricorda l'antico RPG Auto Report (presente almeno fino al rilascio 3.2.0) la cui /COPY, sempre senza toccare i sorgenti, sparpagliava le istruzioni del membro sorgente datore lungo il compilato del ricevente in funzione del tipo specifica. Si potevano così includere contemporaneamente una routine di calcolo e le specifiche di file, di estensione e di immissione necessarie alla routine stessa. Meraviglie perdute!

Se il contenuto di una routine inclusa nei programmi tramite la /COPY viene modificato, tutti i programmi che la usano vanno ricompilati ed eventualmente corretti.

24.16 Chiamate dinamiche a programmi esterni.

Le stesse funzioni che sono candidate ad essere risolte con una /COPY possono dare origine anche ad un programma esterno da richiamare con l'istruzione CALL.

In questo caso la funzione viene racchiusa in un programma costruito, compilato e testato separatamente. L'interfaccia è rigorosamente limitata ai parametri esplicitamente passati al programma chiamato tramite la CALL. Per la verità, nulla impedisce di usare un file, un'area dati o la Local Data Area per passare ulteriori informazioni tra programma chiamante e programma chiamato. L'importante è che i due programmi abbiano anche tale interfaccia supplementare in perfetta evidenza.

Si ipotizza di avere a disposizione un programma di servizio capace di ricevere una stringa di dati alfanumerici e di restituire la stessa stringa ma con gli apici raddoppiati.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Chiama Raddoppia apici.
C                   call      'DUPAPX'
     *    Riceve la stringa con gli apici semplici.
C                   parm                    ppapxi           80
     *    Restituisce la stringa con gli apici raddoppiati.
C                   parm                    ppapxo          160
```

Il programma chiamato (DUPAPX) ha istruzioni perfettamente simmetriche rispetto al chiamante là dove definisce i parametri di ingresso.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Raddoppia apici.
C     *entry        plist
     *    Riceve la stringa con gli apici semplici.
C                   parm                    apxi             80
     *    Restituisce la stringa con gli apici raddoppiati.
C                   parm                    apxo            160
```

I commenti descrivono i parametri come ricevuti o restituiti nello stesso modo in entrambi i programmi e come se i commento fossero quelli del programma chiamato.

- Il commento "riceve" significa che il chiamante passa i dati al chiamato.
- Il commento "restituisce" significa che il chiamato passa i dati al chiamante.

I parametri del chiamante e del chiamato devono corrispondere esattamente in numero, ordine e caratteristiche ma non nel nome.

Si noti che il nome del programma (DUPAPX) va scritto in maiuscolo nella CALL. Il minuscolo si compila ma poi la CALL non trova il programma.

Se il programma chiamato termina in errore, il chiamante riceve un errore terminale. Per intercettare l'errore nel chiamante, si usa l'indicatore Low, acceso se errore, spento altrimenti.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                     call       'DUPAPX'                              50
C                     parm                    ppapxi             80
C                     parm                    ppapxo            160
```

Sulla CALL si può indicare anche una variabile lunga fino a 10 caratteri e, anche in questo caso, il contenuto della variabile deve essere un nome maiuscolo.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                     movel(p)   'DUPAPX'       pgmnam             10
C                     call       pgmnam                                50
C                     parm                    ppapxi             80
C                     parm                    ppapxo            160
```

Volendo si può usare una variabile lunga 21 nella forma LIBRERIA/PROGRAMMA ma valgono le cautele nella menzione di librerie nei sorgenti applicativi. Il contenuto va allineato a sinistra senza blank intermedi.

Se il contenuto di un programma chiamato viene modificato, i programmi che la usano vanno ricompilati ed eventualmente corretti solo se l'interfaccia è cambiata.

Non si spiega la CALLB (Chiamata statica) che è il parto di un manipolo di reazionari calati dai mainframe che hanno espropriato il primigenio gruppo di sviluppo a Rochester.

25 Giornale. (25R)

Lo scopo del ***Giornale*** è duplice.

- Garantire l'integrità del database in occasione di cadute di singoli lavori o di sistema e documentare tutte le attività sui file di database scelti.
- Fare da base al ***Controllo di sincronia***, che permette di trattare le transazioni complesse, composte dall'aggiornamento contemporaneo di più record, come se fosse un aggiornamento unico.

Il meccanismo del giornale si basa su una serie di definizioni e sull'uso di un contenitore di dati particolari, il ricevitore, una sorta di log relativo a tutte le transazioni di aggiornamento effettuate sui file scelti. Il log non è altro che la raccolta delle immagini di tutti i record che hanno subito operazioni di aggiornamento, e non solo è consultabile, ma può anche essere utilizzato per ripristinare i record esattamente come erano prima di subire gli aggiornamenti, qualora si ritenesse opportuno rinunciarvi.

Infatti, in caso di caduta di sistema, ogni file subisce la perdita degli ultimi aggiornamenti, dall'ultimo all'indietro fino ad un tempo aleatorio. Purtroppo le perdite su file diversi non sono cronologicamente fasate. È così possibile che un file contenga aggiornamenti successivi ad altri perduti, che riguardano però un altro file.

Se invece i file sono protetti da giornale, quando la macchina caduta ritorna pronta per l'uso, sono garantiti, a cura del software di base, gli aggiornamenti ai file nell'esatta cronologia in cui sono originalmente avvenuti.

25.1 Componenti di un giornale.

Per potere sfruttare le potenzialità di questi strumenti e poterli utilizzare nei programmi, si deve innanzitutto costituire l'impianto, ovvero una serie di oggetti interdipendenti.

JRNRCV	Ricevitore di giornale.
JRN	Giornale.

25.2 JRNRCV Ricevitore di giornale.

È l'oggetto atto a ricevere la registrazione cronologica di tutte le operazioni di aggiornamento per i file di database per i quali è stato richiesto esplicitamente l'avvio della registrazione su giornale; contiene di fatto le immagini dei record dopo e, a richiesta, prima di ogni aggiornamento.

Il comando di sistema per la creazione di un JRNRCV è il seguente.

CRTJRNRCV JRNRCV(Libreria/Ricevitore) TEXT('Titolo dell'applicazione.')

Il comando di sistema per la cancellazione di un JRNRCV è il seguente.

DLTJRNRCV JRNRCV(Libreria/Ricevitore)

Normalmente il nome di un ricevitore è uguale a quello del giornale al quale è collegato con in più alcune cifre finali. Quando, come spiegato più oltre, i ricevitori si avvicendano, il nome del ricevitore subentrante è identico al precedente ma con il numero in coda aumentato di una unità. È perciò evidente l'opportunità di creare il primo ricevitore con un nome del tipo JJJJJJ0001 dove JJJJJJ è il nome del giornale e 0001 è un progressivo destinato ad incrementarsi a 0002 per il secondo ricevitore e così via.

25.3 JRN Giornale.

E' l'oggetto al quale si collegano, da un lato, il ricevitore e, dall'altro, i file fisici di database dei quali è richiesta la registrazione degli aggiornamenti. Si può specificare il tipo di registrazione al quale assoggettare ogni singolo file: la sola immagine prima dell'aggiornamento od entrambe le immagini prima e dopo l'aggiornamento del record.

Il comando di sistema CRTJRN (Create Journal) crea un giornale e gli attacca un ricevitore preesistente.

CRTJRN JRN(Libreria/Giornale) JRNRCV(Libreria/Ricevitore)
TEXT('Titolo dell'applicazione.')

Il comando di sistema DLTJRN (Delete Journal) cancella il giornale.

DLTJRN JRN(Libreria/Giornale)

Normalmente, il giornale ed i suoi ricevitori giacciono nella stessa libreria che può essere la stessa dei file fisici coperti dal giornale.

25.4 Impianto e cura di un giornale.

Il seguente comando avvia la registrazione di un file fisico sotto giornale con entrambe le immagini.

STRJRNPF FILE(FileLib/File) JRN(Libreria/Giornale) IMAGES(*BOTH)

Per terminare la registrazione dello stesso file, il comando è invece.

ENDJRNPF FILE(FileLib/File) JRN(Libreria/Giornale)

Con l'utilizzo il ricevitore si riempie ed è pertanto periodicamente necessario scollegarlo e sostituirlo. Il ricevitore scollegato resta in linea fino all'eventuale salvataggio e alla sicura cancellazione.

Ciò detto, per scollegare un ricevitore e collegarne uno nuovo vuoto e creato automaticamente con un nome progressivo come già spiegato (25.2), si dovrà utilizzare il seguente comando.

CHGJRN JRN(Libreria/Giornale) JRNRCV(*GEN) SEQOPT(*CONT)

Il parametro JRNRCV(*GEN) indica al sistema di scollegare il ricevitore corrente ma al contempo di generarne uno nuovo e di collegarlo al giornale. Il parametro SEQOPT(*CONT) indica al sistema di proseguire in sequenza con la numerazione progressiva delle voci di giornale.

Per operare con questi oggetti vi sono svariati comandi di sistema (GO CMDJRN).

Il primo, WRKJRNA (Work Journal Attribute), alla faccia del WRK che indicherebbe una capacità gestionale, permette di visualizzare le caratteristiche del giornale, l'elenco dei file registrati, l'elenco dei ricevitori.

WRKJRNA JRN(Libreria/Giornale)

Il secondo, WRKJRN (Work Journal), permette una serie di operazioni rischiose che hanno poco senso in ambiente produttivo. Se ne sconsiglia quindi l'uso all'inesperto. Entra, stranamente, senza parametri: altro segno di logica iniqua.

WRKJRNA

Il terzo, WRKJRNRCV (Work Journal Receiver), visualizza direttamente le caratteristiche dei ricevitori di giornale.

WRKJRNRCV JRNRCV(Libreria/Ricevitore)

Il quarto, DSPJRN (Display Journal), visualizza le voci contenute in un giornale, permettendo numerose parzializzazioni per tipo di voce, data e file.

Se invece di *CURCHAIN si usa *CURRENT, ci si limita al solo ricevitore collegato.

DSPJRN JRN(Libreria/Giornale) FILE((FileLib/File)) RCVRNG(*CURCHAIN)

25.5 Le transazioni.

Per transazione semplice si intende un intervento sul database riguardante l'aggiornamento di un solo record. Ad esempio.

- Un record anagrafico clienti.

Per transazione complessa si intende un intervento sul database riguardante l'aggiornamento di un gruppo correlato di vari record. Ad esempio.

- Un movimento di magazzino e i record riepilogativi di articolo e di ubicazione.
- Una riga documento e la relativa testata documento che totalizza il valore di tutte le righe.

25.6 Il controllo di sincronia.

Il giornale fa da base ad una importante tecnica di programmazione che riduce in modo drastico la difficoltà di gestione delle transazioni complesse: il controllo di sincronia (commitment).

Necessità del controllo di sincronia.

Nella transazione semplice è il sistema a garantire che la transazione sia tutta fatta o tutta non fatta a cavallo di una caduta di lavoro o di sistema. Ed è ancora il sistema a garantire che il record transatto sia coinvolto in una sola transazione per volta. In pratica, il record letto per aggiornamento non subisce altre modifiche prima della conclusione dell'aggiornamento in corso.

Se invece la transazione è complessa, il sistema non fornisce spontaneamente un servizio paragonabile a quello che riguarda la transazione semplice. Nondimeno è fortemente desiderabile che una caduta non lasci nessuna transazione complessa a mezza via. Il giornale, in quanto prerequisito, ed il ciclo di sincronia permettono di gestire la transazione complessa come richiesto.

In sostanza, la transazione complessa, grazie al ciclo di sincronia, è gestibile come una transazione semplice con un impatto minimo sulla minutazione.

In sovrappiù, sempre durante una transazione complessa, può capitare che, a metà degli aggiornamenti, si decida di rinunciare alla transazione a causa di una intervenuta difficoltà (record occupati da altri, dati formalmente errati a dispetto dei controlli preliminari). Nasce allora il problema di disfare la mezza transazione già fatta, arduo perché durante la recessione si possono incontrare difficoltà analoghe a quelle incontrate per prime: ma è chiaro che non esiste il rovescio del rovescio. Anche qui il ciclo di sincronia risolve perfettamente il problema e permette logiche di particolare semplicità.

Avvio e chiusura del controllo e del ciclo di sincronia.

Una volta costituiti ricevitore e giornale ed avviata la registrazione cronologica degli eventi per i file di database coinvolti nelle transazioni complesse da governare, si dovranno adottare tecniche specifiche di programmazione RPG per gestire e controllare le transazioni complesse.

Mentre l'attività di journaling, una volta innescata, resta attiva permanentemente e non richiede variazioni ai programmi RPG, l'attività di commitment deve essere innescata dentro ciascun lavoro ed i programmi devono dichiarare esplicitamente la soggezione di ciascun file al controllo di sincronia.

Sulla specifica di definizione del file è necessario informare il compilatore che il file sarà sottoposto a controllo di sincronia.

```
FX  FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++++
    Fester01l  uf   e           k disk    commit
```

Prima che i file soggetti vengano aperti, si dovrà attivare il controllo di sincronia mediante il comando.

STRCMTCTL LCKLVL(*CHG)

Al termine dell'azione, si disattiverà il controllo di sincronia con il comando.

ENDCMTCTL

Solitamente ciò avverrà nel CL chiamante. In obbrobriosa alternativa si userà il solito QCMDEXC. Si forniscono file, calcolo di apertura e calcolo di chiusura.

```
FX  FFilename++IPEASF.....L.....A.Device+.Par.chi.++++++++++++++++++++++++++++++
    Fester01l  uf   e           k disk    commit usropn
```

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Avvia il controllo di sincronia.
C                   eval      qcmd='strcmtctl lcklvl(*chg)'
C                   call      'QCMDEXC'
C                   parm                    qcmd            80
C                   parm       80           qcmdl           15 5
     * Apre il file ESTER01L.
C                   open      ester01l
```

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
    * Chiude il file ESTER01L.
C                   close     ester01l
    * Chiude il controllo di sincronia.
C                   call      'QCMDEXC'
C                   parm      'endcmtctl'   qcmd            80
C                   parm       80           qcmdl           15 5
```

Logica del ciclo di sincronia.

Si noti che, sotto controllo di sincronia, ogni record letto per update viene bloccato e quindi allocato in modo esclusivo dal programma in esecuzione non solo fino al suo ricalco (UPDATE) o rilascio (UNLOCK), ma fino alla esplicita chiusura del ciclo di sincronia (COMMIT) o rinuncia agli aggiornamenti (ROLBK).

Anche i record nuovi restano bloccati e vengono rilasciati dalla chiusura del ciclo o cancellati dalla rinuncia.

Un programma che utilizza il controllo deve nascere intorno a tecniche di programmazione che prediligano, ad esempio, l'esecuzione in un solo fiato di tutte le operazioni di aggiornamento proprie di una transazione complessa. In un fiato significa: senza inciampare nel video tra il primo e l'ultimo degli aggiornamenti della transazione.

Solitamente si andrà a video senza mai leggere per update e, dopo aver modificato i dati a video, si rileggeranno per update tutti i record da variare e si riscriveranno immediatamente. Visto il meccanismo di rilascio in massa di tutti i record operato dal COMMIT, non è necessario prima leggere tutti i record e poi ricalcarli, ma si potrà procedere in un ordine a piacere, magari leggendo per update e ricalcando i record uno alla volta.

Il primo ciclo di sincronia si apre quando nel lavoro viene aperto il primo programma che contiene file dichiarati sotto commitment. Un ciclo qualsiasi deve essere chiuso sempre o da COMMIT o da ROLBK. La chiusura del controllo di sincronia (ENDCMTCTL) deve avvenire dopo la chiusura esplicita dell'ultimo ciclo di sincronia, altrimenti il sistema interroga l'utente sul destino dell'ultimo ciclo.

Si osserva che non esiste una apertura esplicita del ciclo di sincronia ma solo la chiusura o la rinuncia. Non è pertanto possibile escludere qualche lettura per update da un ciclo: semplicemente si eviteranno a favore delle letture non per update (23.5).

Codici operativi del ciclo di sincronia.

Come anticipato, sono due i codici operativi specifici del controllo di sincronia.

COMMIT Conferma ciclo di sincronia

Dopo che in una transazione complessa sono state effettuate tutte le operazioni di aggiornamento e scrittura nei file (WRITE/UPDATE/EXCEPT/DELETE) e, verificato che non si siano verificati degli errori durante tali attività, il codice operativo COMMIT conferma gli aggiornamenti eseguiti sul database. Dopo la COMMIT, tutti i record di tutti i file aggiornati sotto commitment vengono sbloccati e resi disponibili ad altri programmi o lavori.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                   commit
```

ROLBK Rinuncia agli aggiornamenti del ciclo di sincronia

Il codice operativo ROLBK si utilizza in alternativa alla COMMIT quando durante le operazioni di aggiornamento o scrittura di una transazione complessa si verificano degli errori.

La ROLBK evita quindi di lasciare transazioni zoppe, incomplete od errate.

Una volta rilevato un errore, si utilizza questa operazione per rendere nulli tutti gli aggiornamenti e le scritture avvenute sui file sotto commitment dopo l'ultima COMMIT o ROLBK eseguite. Con la ROLBK, i record su cui è già stato effettuato l'aggiornamento vengono ripristinati allo stato antecedente la transazione in corso. Quest'ultima sparisce completamente dal sistema, come se non fosse mai stata intrapresa.

Il ripristino è possibile perché dentro il ricevitore di giornale sono presenti le immagini dei record prima dell'aggiornamento. Questo anche se i file interessati registrano normalmente solo le immagini dopo (*AFTER) e non entrambe (*BOTH). Infatti l'apertura del controllo di sincronia costringe il journaling a diventare provvisoriamente *BOTH in vista di eventuali ROLBK.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                       rolbk
```

25.7 Utility di lettura del giornale.

Se il giornale è costruito correttamente, nei suoi ricevitori si accumulano le modifiche di tutti i file delle librerie dati di un sistema informativo. Il comando DSPJRN (Display Journal) permette di esaminare, tra le altre cose, le immagini dei record prima (se previsto) e dopo gi aggiornamenti.

Ogni voce immagine contiene un mazzo di campi informativi relativi all'avvenimento che l'ha generata: file, programma, utente, tempo, tipo aggiornamento ed altro.

Tuttavia l'immagine dei dati giace in una stringa piatta, senza tracciato, né sarebbe possibile diversamente, visto il limite di un solo tracciato per file fisico di cui soffre il database descritto esternamente.

La stringa dati rispetta ovviamente il tracciato del file giornalato. Se nel DSPJRN ci si limita ad un solo file, è possibile copiarne l'emissione in un tracciato ad hoc che interpreta correttamente il campo dati per quel solo file.

Questo fa l'utility JJRN, scaricabile dalla pagina www.neroni.it/Scaricabili. Essa permette di scegliere un file dal giornale, costruisce un file ad hoc e lo riempie coi dati del giornale. Il file può quindi essere esaminato con query o sql per trovare chi e quando.

26 Debug sorgente e interattivo per la ricerca degli errori. (27C)

27 Debug è cercare un errore in un programma compilato al momento della esecuzione valendosi della lettura a vista o automatica del sorgente.

27.1 Tipi di debug disponibili.

In origine, ai tempi del 38, il debug era cieco. Guardando il sorgente in un'altra sessione, si appostavano dei punti di interruzione dove si sperava di scoprire qualcosa. Arrivato il pgm al punto, si esploravano le variabili e si apponevano altri punti di interruzione.

Meglio certamente che distribuire istruzioni aggiuntive da togliere a debug finito, come si era costretti a fare sul Sistema/3. Ricordo ancora con angoscia la mia prima ricerca di errore sul programma www.neroni.it/Scaricabili/JXLTNBR.txt che trasformava importi in descrizioni. Era il primo programma difficile, scritto scopiazzando espedienti che credevo di aver capito.

Quaranta ore per concepirlo e quaranta ore per trovare un errorino. Per la cronaca, l'istruzione di cui non conoscevo abbastanza era la MOVEL. Ancora oggi, come allora (meravigliosa coerenza dell'evoluzione), quando si muove a sinistra un campo alfa in uno numerico più corto, non vengono mosse solo le cifre di sinistra ma si coglie anche lo zone dell'ultimo byte di destra del datore per farlo diventare il segno del ricevente.

Avevo una schiera alfa che conteneva lunghezza e descrizione come la seguente.

```
D   DName++++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
     * LUNGHEZZA/DESCRIZIONE
    D LUDE            S              12    DIM(10) CTDATA PERRCD(1)
**   ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
    ** LUNGHEZZA/DESCRIZIONE
    03       UNO
    03       DUE
    O3       TRE
    07   QUATTRO
    06    CINQUE
    03       SEI
    05     SETTE
    04      OTTO
    04      NOVE
    05     DIECI
```

Il problema proveniva dalle seguenti istruzioni.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
    C                   MOVEL     LUDE(4)       LU                2 0
    C                   MOVE      LUDE(4)       DE               10
```

Il risultato non era, come sperato

```
LU=7
DE="   QUATTRO"
```

Ma piuttosto

```
LU=-7
```

Usando LU come numero di iterazioni da compiere (estrarre un byte alla volta dal campo DE per portarlo in un campo descrittivo) il loop con un limite negativo non usciva al momento giusto e il campo risultante ne risultava devastato.

Il rimedio fu rendere incondizionatamente positivo il campo LU con l'istruzione.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
    C                   MLLZO     0             LU
```

Move low to low zone di zero, cioè prendi lo zone di zero (x'F0') cioè "F" e portalo nel campo guasto LU (x'F0D7') per farlo diventare x'F0F7).

Sic!

27.2 Modalità di compilazione dei programmi per l'uso ottimale del debug.

27.3 Punti di interruzione, tipi di liste, watch,…

28 Utilizzo dei Manuali. (32C)

I manuali del sistema sono nati cartacei ma si sono evoluti nel tempo fino all'ultima versione, sicuramente la più comoda, scaricabile da Internet in formato PDF, quello di Adobe Reader, il cui nome in passato era Acrobat Reader.

Si ricordano soltanto per cultura le versioni precedenti.

28.1 Versioni dei Manuali.

- La versione cartacea è vissuta molto a lungo. Nel 1980 il cliente S/38 IBM riceveva tutti i manuali di carta, compresi nel prezzo del software per un peso che raggiunse una maturità di 30 o 40 chili. Un modesto numero era in Italiano. Era attivissimo un servizio di news letter che permetteva di conservare a lungo il primo pacco semplicemente sostituendo qualche centinaio di pagine all'anno.
- In un secondo tempo, poco dopo la comparsa del primo AS/400, alcuni manuali divennero a richiesta.
- In corrispondenza della versione 3 venne timidamente distribuito un CD-ROM contenente la Softcopy Library. Esso conteneva, oltre ai manuali, anche il programma IBM che ne permetteva la lettura, il BookManager Library Reader.
- Per costringere i programmatori ad andare su Internet, la Softcopy Library venne in seguito impoverita di alcuni manuali essenziali, come RPG e DDS, che furono portati nei siti Internet ma sotto una forma HMTL particolarmente disgraziata. Non si poteva stampare più di un capitoletto alla volta e non era previsto lo scarico. Sostanzialmente, nel corso della versione 3 e in parte della 4, Internet ha suggestionato l'IBM tanto da generare una versione in linea di manuali HTML poco portabile, poco leggibile, poco stampabile e poco congruente con le antiche sane abitudini.
- In compagnia della Softcopy Library veniva fornito anche un CD-ROM intitolato AS/400 Information Center che permetteva di andare rapidamente su Internet.
- La mutevolezza delle scelte successive testimonia una grave confusione da parte della testa pensante IBM che, non sapendo a che santo votarsi, ha tentato tutte le strade per rendersi gradevole (non si sa a chi) con le luci e gli effetti speciali di Internet. Poi il tranquillo ritorno alla forma libresca classica, naturalmente elettronica.
- Con la maturità della versione 4, i manuali erano disponibili più o meno completamente anche in formato PDF, scaricabile e quindi stampabile e consultabile anche senza la connessione Internet attiva.
- La versione 5 ha decretato l'uso esclusivo di Adobe Reader e del formato PDF. Il ritorno graduale alla forma di documento di formato standard e diffuso è un sollievo per chi deve usare davvero i manuali.
- La versione HTML resta utile per le ricerche estemporanee, quando si cerca un argomento ma non si sa in che manuale si trovi.
- Personalmente ad ogni release, scarico su PC in un indirizzario ManualiV5R4 tutti i manuali che oggi sono raggiungibili da un elenco internet unico. Il nome del file scaricato è di alcune lettere e cifre poco utili all'individuazione del contenuto. Perciò, subito dopo lo scarico, apro il file, copio il titolo, esco e lo incollo nel nome del file lasciando in testa il nome originale. E' importante non perdere il nome originale per l'evidenza della sua provenienza: altrimenti capita di riscaricare più volte la stessa cosa senza accorgersene, aumentando la confusione nella ricerca del manuale che si vuole consultare.
- Si nota che il nome dello stesso manuale cambia ad ogni release e che l'Ibm non risparmia inutili riorganizzazioni degli argomenti, giustificate una volta dalla necessità di contenere le

dimensioni fisiche del singolo cartaceo, necessità superata dal fatto che nessuno stampa più delle poche pagine dell'argomento che gli interessa leggere senza usare un PC.

Qualcuna delle affermazioni precedenti è inesatta e forse malevola. Forse, in illo tempore, chi scrive non ha tentato a sufficienza di strappare ad Internet i manuali, girando per pagine incomplete per certi versi e ridondanti per molti altri, combattendo contro l'oscura organizzazione degli argomenti, contro scarichi lunghi e faticosi (nemmeno zippati!). Resta solo l'osservazione che l'assenza del cartaceo, per quanto ecologica, ha disabituato i pivelli a farne uso e che una navigazione difficoltosa non è il modo migliore di invogliarli a familiarizzare con i manuali, fonte del sapere informatico.

Mi è capitato di rivelare a programmatori reduci dai corsi IBM avanzati che i manuali sono veramente in internet e come ci si arriva. Il tentativo solitario di arrivarci entrando in www.ibm.com è di solito così frustrante che la più parte rinuncia.

28.2 Ultima versione dei Manuali.

Di tutte le storie precedenti il sugo è che oggi è possibile collegarsi senza alcun altro intermediario agli opportuni indirizzi Internet per scaricare gratis i manuali originali IBM che interessano. Il collegamento a internet è meglio che sia un Adsl perché la misura di un manuale è di qualche mega e a 56k lo scarico dura troppo.

Non illudetevi di trovare granché in italiano. Qualcuno conserva ancora del cartaceo del secolo scorso (sic!) come reliquia ed il tragico è che lo usa. Il vituperato inglese maccheronico basta e avanza a capire tutti i manuali scritti in un maccheronico simile al nostro da gente che inglese non è quasi mai.

Si illustra, come esempio, lo scaricamento da Internet del rilascio V5R4 del seguente manuale in formato PDF. La versione 6.1 avrà bisogno di ulteriori spiegazioni perché, (udite, udite!) IBM ha ancora cambiato il supporto, l'impaginazione, la collezione di cose scaricabili, ma solo per il nuovo release. Contentatevi della documentazione per lo scarico di un solo manuale del 5.4, visto che ho dovuto buttare tutta la navigazione che avevo fotografato prima di questo e che le immagini di due giri appesantiscono le dimensioni del manuale in maniera eccessiva.

DDS for Physical and logical files

Si parte da un sito d'ingresso che dà accesso a tutti i manuali delle versioni 5.3, 5.4 e 6.1. Altre strade sono possibili ma si sconsiglia, nell'urgenza, di ravanare a vanvera dal sito www.ibm.com che è l'inizio di una foresta impenetrabile.

Ad evitare di perdere la traccia quando le pagine saranno obsolete ed altre simili saranno subentrate, se ne copia qui di seguito il contenuto a memoria ed ammaestramento dei posteri, sperando che chi farà le prossime non ne peggiori la navigabilità.

28.3 Ingresso su Internet (IBM System and i5/OS Information Center).

Digitare il seguente link, se si sta leggendo questo manuale in cartaceo, o cliccare di sinistro, se si sta leggendo il pdf su un PC.

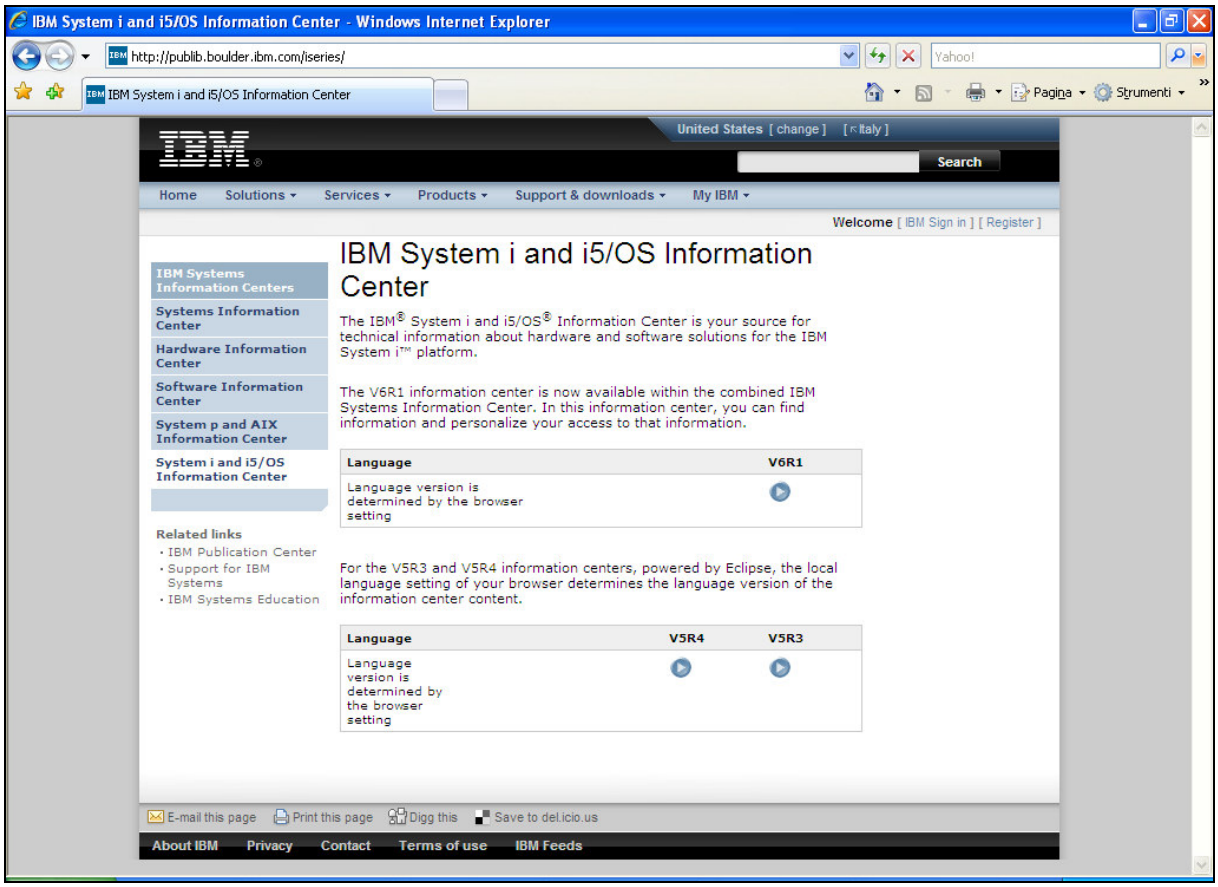
Si ottiene la seguente pagina 1.

http://publib.boulder.ibm.com/iseries/

Si nota che il collegamento è già abbastanza difficile da ricordare. Una volta entrati, conviene copiarlo dalla barra dell'indirizzo della pagina trascinandolo sul desktop del PC o dove altro comoda. Si rinuncia a partire da un indirizzo più semplice perché si rischia il panico da labirinto.

Si noti che non vale scrivere www al posto di http://.

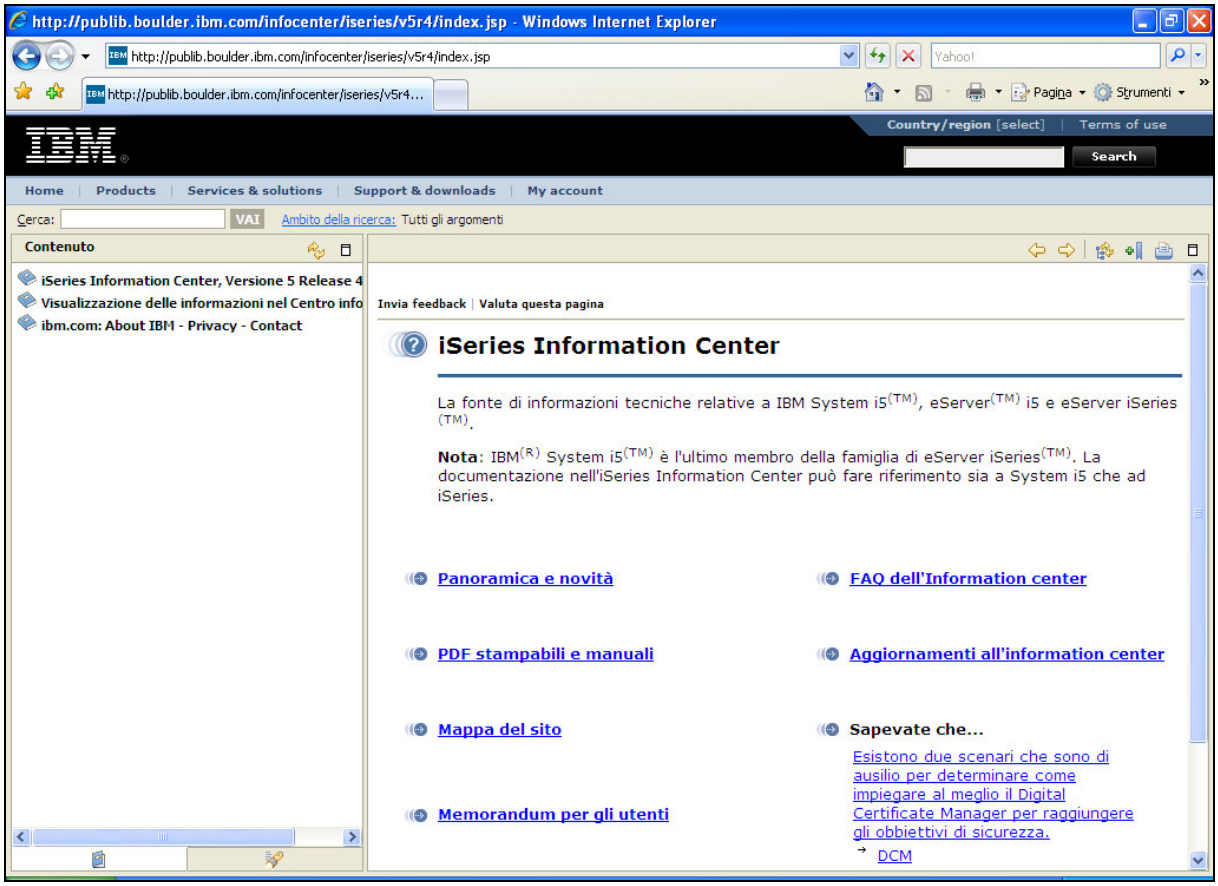
Pagina 1



Sulla pagina 1, cliccare l'icona V5R4 per ottenere la pagina seguente di cui viene evidenziato anche il collegamento.

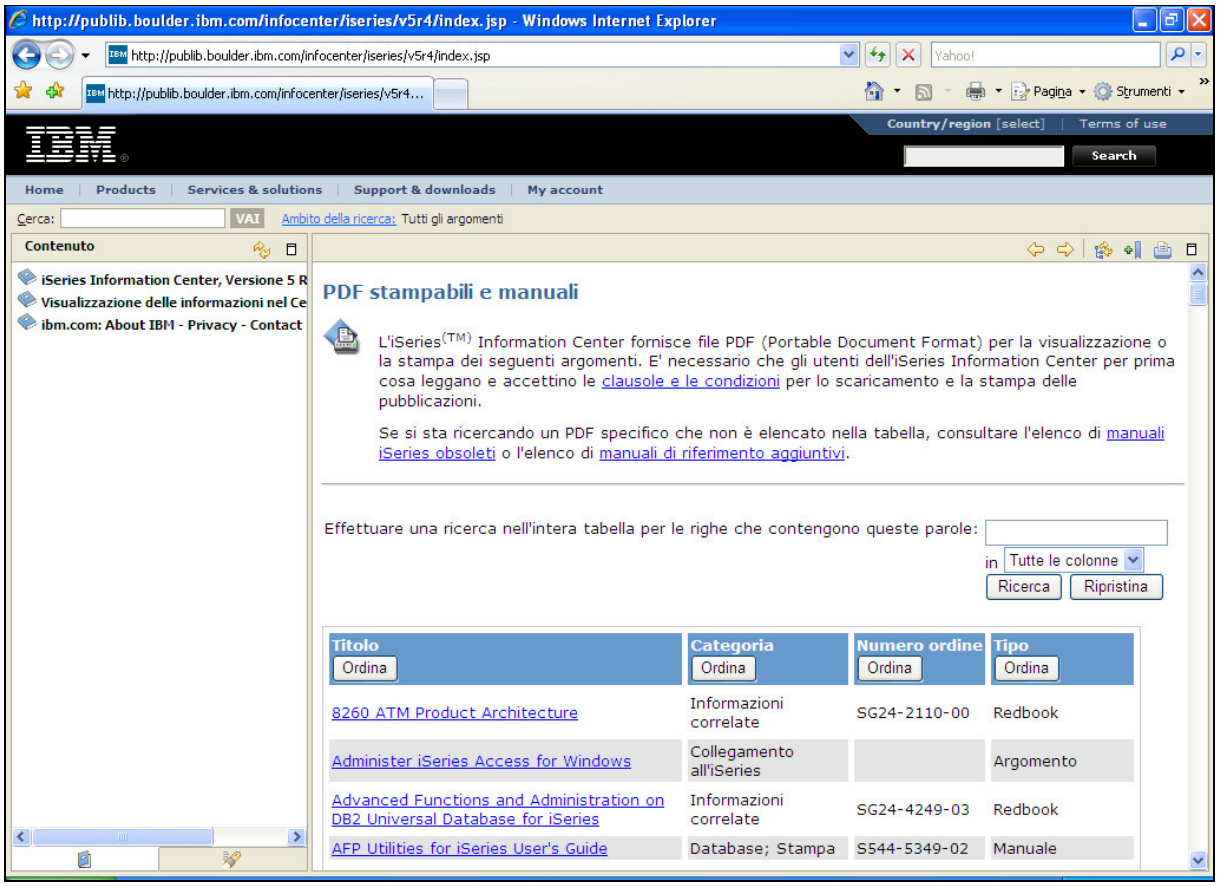
http://publib.boulder.ibm.com/infocenter/iseries/v5r4/index.jsp

Pagina 2



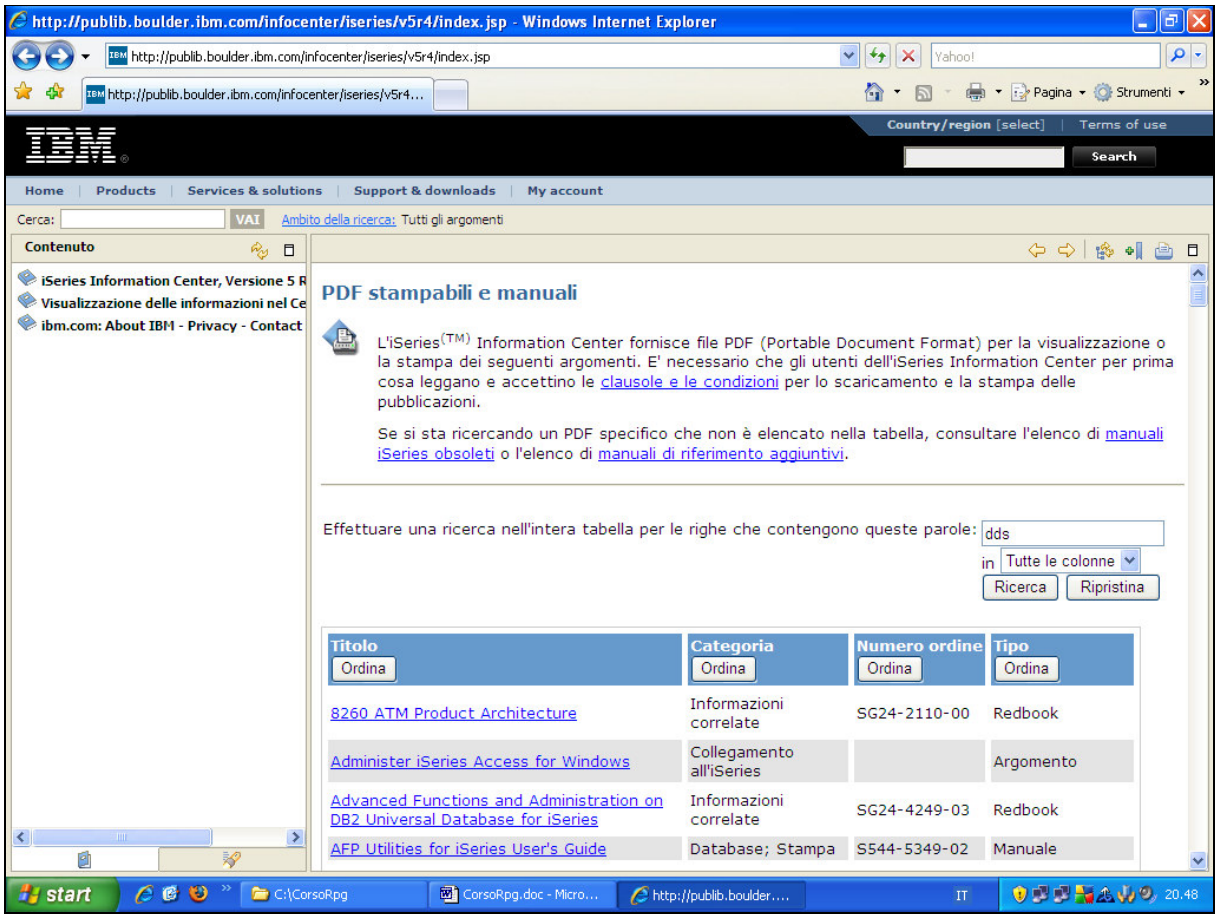
Sulla pagina 2, cliccare il link PDF stampabili e manuali per ottenere la pagina seguente, il cui link non è presentato in chiaro.

Pagina 3



Sulla pagina 3, digitare "dds" nel campo di ricerca, come si vede nella seguente.

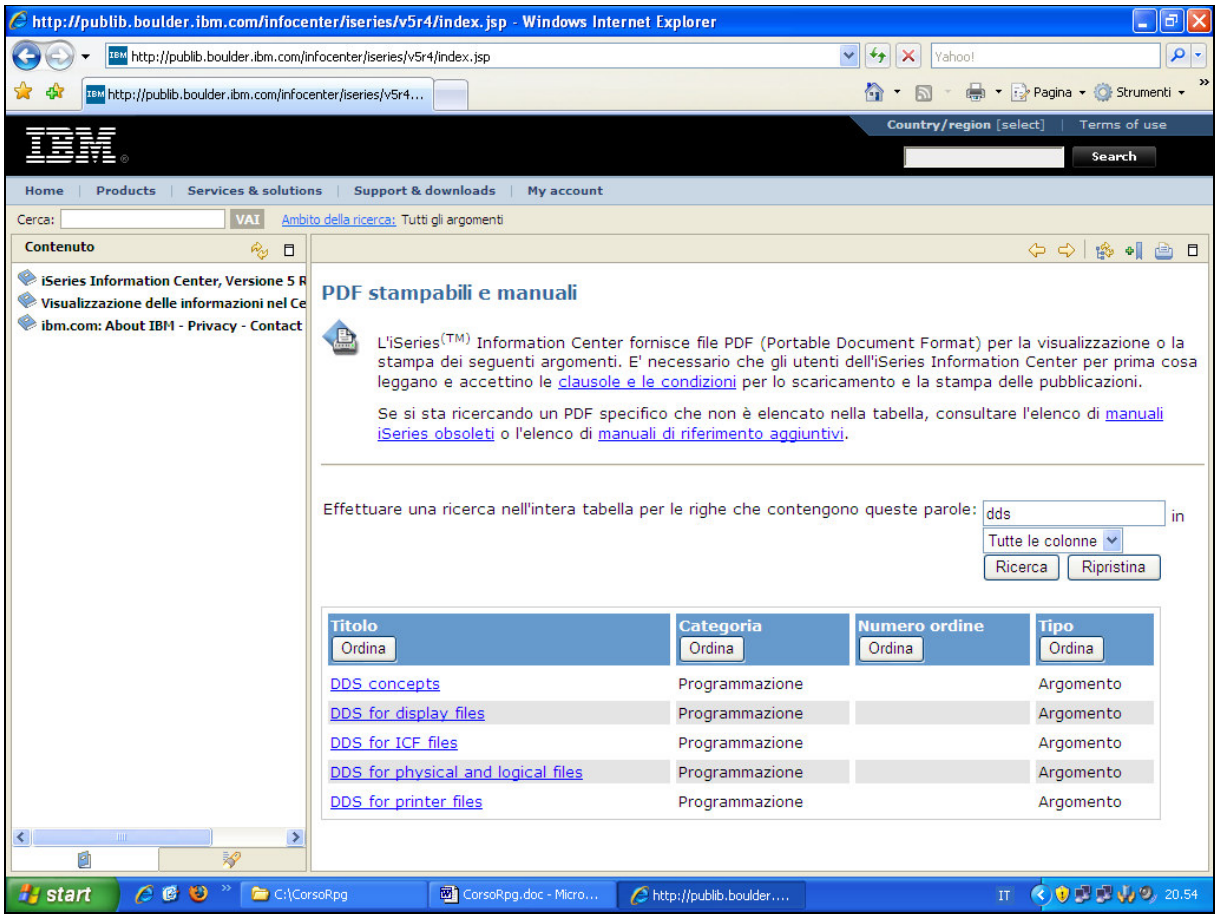
Pagina 4



Quindi premere il pulsante Ricerca.

Si presenta un sottoinsieme delle righe della pagina 3.

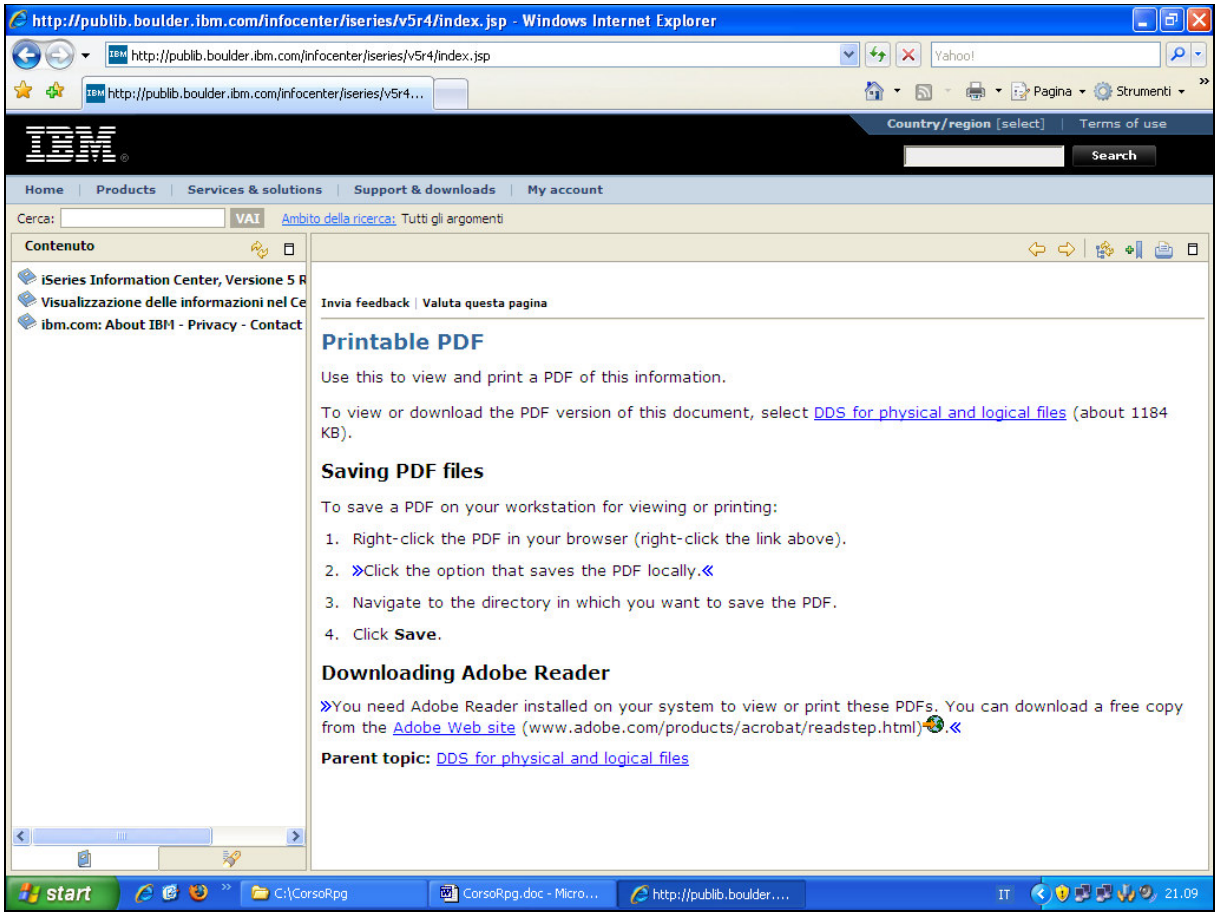
Pagina 5



Sulla pagina 5, cliccare sul collegamento DDS for physical and logical files

Si presenta una pagina che permette finalmente la visualizzazione e lo scarico PDF del manuale e anche la consultazione delle pagine HTML sul medesimo argomento.

Pagina 6



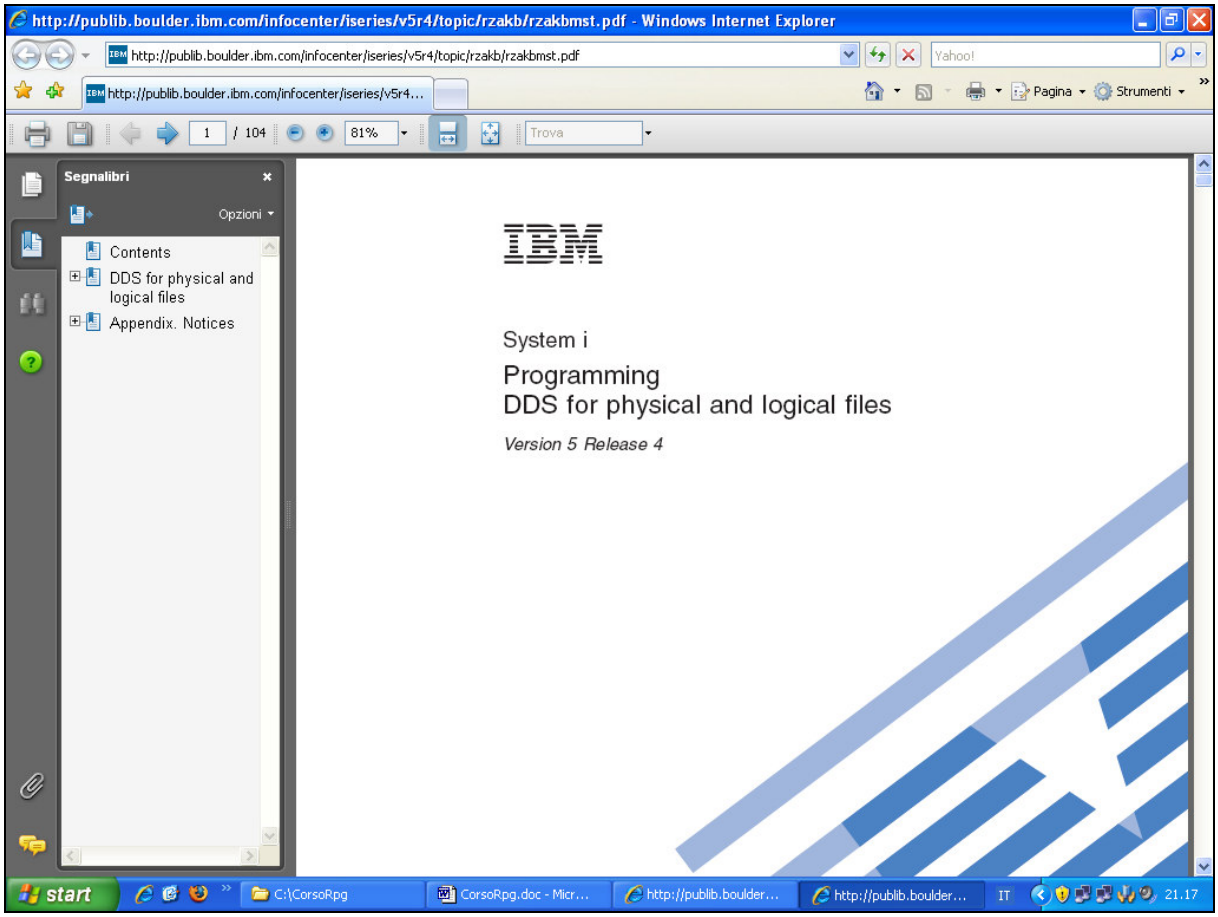
La pagina 6 contiene anche uno dei tanti collegamenti per installare Adobe Reader, se necessario.

La pagina 6 non si presenta per altri manuali e si passa direttamente dalla pagina 5 alla 7, ad esempio per il manuale **ILE RPG Language Reference** che corrisponde al collegamento http://publib.boulder.ibm.com/infocenter/iseries/v5r4/topic/books/sc092508.pdf

Sulla pagina 6, cliccare di sinistro sul primo dei collegamenti DDS for physical and logical files.
Si presenta il manuale PDF che corrisponde al collegamento seguente.

http://publib.boulder.ibm.com/infocenter/iseries/v5r4/topic/rzakb/rzakbmst.pdf

Pagina 7



La visualizzazione del PDF è gestita dal programma Adobe Reader che deve essere già installato sul PC. Dalla visualizzazione si può ottenere il salvataggio ma mi risparmio questa spiegazione che non rientra negli scopi di questo manuale.

In questo caso lo stesso collegamento chiamato col click di destra invece che di sinistra, permette anche lo scarico diretto senza passare per la visualizzazione.

Esce una finestra sulla quale occorre scegliere Salva oggetto con nome... che, decisa la cartella su cui scaricare, salva sul proprio personal il manuale desiderato con il nome **rzakbmst.pdf**

Non sempre è agibile lo scarico diretto; nel caso farlo dalla visualizzazione PDF, come detto nel paragrafo successivo.

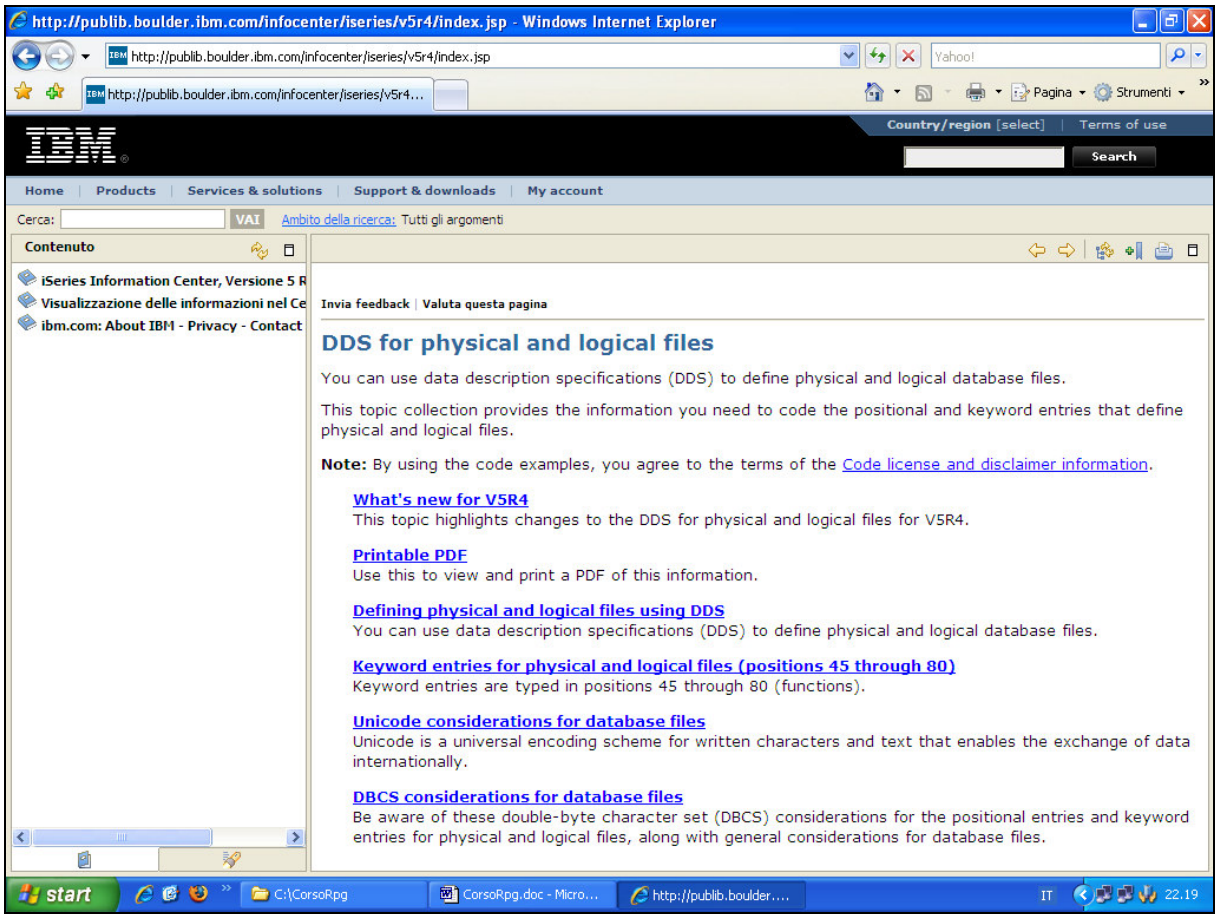
Per completezza si documenta anche l'ingresso sulle pagine del manuale HTML.

Sulla pagina 6, cliccare di sinistro sul collegamento

Parent topic: DDS for physical and logical files

Si presenta la pagina HTML che contiene le stesse informazioni del PDF.

Pagina 8



Dalla pagina 8, cliccare i vari collegamenti.

28.4 Se lo scarico diretto è problematico.

Installata una volta per tutte l'ultima edizione di Adobe Reader, se il sito non è di quelli disgraziati, il clic di sinistra su un manuale in formato PDF lo apre per la lettura. Il metodo è sconsigliabile perché ogni volta scarica il manuale intero (o quasi) e poi, chiuso Adobe Reader, lo getta.

Il metodo migliore è scaricare i manuali una tantum come già spiegato ai paragrafi precedenti.

Qualche volta però lo scarico è inagibile per le paturnie del sito o per quelle del proprio personal.

In tali occasioni si può tentare di salvare il file da Adobe Reader usando l'icona, visto che Adobe Reader chiamato da Explorer manca dei menù in alto.

Si faccia attenzione a conservare il nome giusto al manuale, tanto per riconoscerlo con facilità.

Ad esempio, proprio il manuale **ILE RPG Language Reference** richiede lo scarico da PDF.

Poi conviene entrare nel manuale per copiare il titolo

DDS for physical and logical files ,

uscire dal PDF e modificare il nome da

rzakbmst.pdf

a

rzakbmst DDS for physical and logical files.pdf

facendo uso di F2 sul file scaricato e della funzione incolla.

29 Formati dei dati. (35C)

I dati presenti nei file del database AS/400 vengono interpretati normalmente tramite la descrizione esterna assegnata al file fisico al momento della sua creazione. Ne segue che ogni singolo carattere presente nel file viene interpretato diversamente secondo la tipologia del campo che lo ospita.

Innanzitutto si espone la notazione esadecimale, che permette di esprimere il contenuto in bit di ogni spazio fisico elettromagnetico (il byte) che nella memoria dell'elaboratore è capace di rappresentare un carattere.

Non si trattano qui i formati più recenti, ad esempio virgola mobile o campi grafici, perché non abbastanza frequenti da essere necessari in questo livello di corso.

29.1 Notazione esadecimale.

Sull'AS/400, come su tutti gli elaboratori dotati di ambiente carattere, ogni carattere è associato ad un byte, composto da 8 bit, dove il bit è l'informazione binaria elementare (0-1 ovvero spento-acceso) gestibile dall'elaboratore. È perciò possibile definire fino a 256 caratteri o, meglio, simboli diversi. 256 è il numero di combinazioni diverse di 0 e 1 che si possono generare con 8 bit e coincide con l'ottava potenza del 2.

Su ogni tipo di elaboratore vige una certa corrispondenza tra caratteri e byte. Tale corrispondenza è espressa tramite una tabella che associa ad ogni carattere una certa configurazione di acceso-spento per gli otto bit del byte. Sui pc vige la tabella ASCII mentre l'AS/400 fa uso della tabella EBCDIC. Lo stesso carattere è rappresentato da byte diversi in tabelle diverse.

Per significare lo stato di quattro bit si conviene di usare la notazione esadecimale che associa un simbolo ad ognuna delle 16 combinazioni di quattro bit. Riunendo due simboli esadecimali si può quindi significare lo stato dei bit di un byte.

La corrispondenza tra semibyte e simboli esadecimali è la seguente.

0000	**0**
0001	**1**
0010	**2**
0011	**3**
0100	**4**
0101	**5**
0110	**6**
0111	**7**
1000	**8**
1001	**9**
1010	**A**
1011	**B**
1100	**C**
1101	**D**
1110	**E**
1111	**F**

Ciò significa, ad esempio, che l'esadecimale A descrive i quattro bit come acceso–spento–acceso–spento ovvero 1010.

Perciò, se la tabella EBCDIC assegna al carattere g (g minuscolo) l'esadecimale 87 significa che l'elaboratore archivierà il carattere g con la sequenza di bit 1000 0111.

Se, ancora, la tabella EBCDIC assegna al carattere B (B maiuscolo) l'esadecimale C2 significa che l'elaboratore archivierà il carattere B con la sequenza di bit 1100 0010.

Nell'esempio non si confonda il carattere B con il simbolo esadecimale B.

Per ragioni storiche legate alle rappresentazioni numeriche, si assegna al primo esadecimale che simboleggia lo stato dei primi quattro bit il nome di zone e al secondo il nome di digit. Riprendendo gli esempi.

Carattere	Zone	Digit
g	8	7
B	C	2
C	C	3
i	8	9
a	8	1
o	9	6
M	D	4
m	9	4

Una rappresentazione frequente e molto comprensibile di una stringa di dati con la relativa codifica esadecimale è quella verticale, di seguito esemplificata per una stringa di 12 caratteri.

```
Carattere   Ciao Mamma
Zone        C8894D899844
Digit       391604144100
```

29.2 Tracciato esemplificativo.

Si utilizza il medesimo tracciato dato in 6.1 e qui ripetuto in sintesi.

```
PF  A..........T.Nome++++++RLun++TPdB......Funzioni++++++++++++++++++++++++++++++
    A          R FMTRCD
    A            FLDALF        12
    A            FLDPAK         7P 2
    A            FLDSIG         5S 0
    A            FLDBIN         9B 0
```

Si può supporre ogni record del file come un insieme di campi contigui, ciascuno composto da un insieme di byte. Ogni tipologia di campo ha regole diverse di interpretazione. Dentro un RPG, la dichiarazione del file provoca la dichiarazione automatica dei campi in memoria. Al momento della lettura del record, ogni campo si riempie con i valori desunti dai byte che gli competono.

Si esemplifica un record del file, interpretato in esadecimale.

```
Colonna            ...+... 1 ...+... 2 ...+
Carattere          Ciao Mamma  ****12345****
Zone               C8894D8998441357FFFFF05C1
Digit              391604144100246F123457BD5
```

Il carattere asterisco barrato **~~****~~** è il simbolo usato per i caratteri casuali che si formano in corrispondenza di campi impaccati o binari.

Utilizzando il tracciato, si spezza il record in campi e si evidenzia il valore in memoria.

Campo	FLDALF	FLDPAK	FLDSIG	FLDBIN
Colonna	...+... 1 .	..+.	.. 2	...+
Carattere	Ciao Mamma	**~~****~~**	12345	**~~****~~**
Zone	C8894D899844	1357	FFFFF	05C1
Digit	391604144100	246F	12345	7BD5
Valore	Ciao Mamma	12345,67+	12345+	123456789+

Altro esempio di spezzamento. Si evita l'esemplificazione del record, facilmente deducibile.

Campo	FLDALF	FLDPAK	FLDSIG	FLDBIN
Colonna	...+... 1 .	..+.	.. 2	...+
Carattere	Bambalughino	**~~****~~**	1234N	**~~****~~**
Zone	C89889A88899	1357	FFFFD	FA3E
Digit	214213478956	246D	12345	842B
Valore	Bambalughino	12345,67-	12345-	123456789-

Altro esempio.

Campo	FLDALF	FLDPAK	FLDSIG	FLDBIN
Colonna	**...+... 1 .**	**..+.**	**.. 2**	**...+**
Carattere	**BRAMBILLONE**	********	**00001**	********
Zone	**CDCDCCDDDDC4**	**0001**	**FFFFF**	**0000**
Digit	**291429336550**	**000F**	**00001**	**0001**
Valore	BRAMBILLONE	00000,01+	00001+	000000001+

Altro esempio.

Campo	FLDALF	FLDPAK	FLDSIG	FLDBIN
Colonna	**...+... 1 .**	**..+.**	**.. 2**	**...+**
Carattere	**TRAPPOLACCIA**	********	**0000J**	********
Zone	**EDCDDDDCCCCC**	**0001**	**FFFFD**	**FFFF**
Digit	**391776313391**	**000D**	**00001**	**FFFF**
Valore	TRAPPOLACCIA	00000,01-	00001-	000000001-

29.3 Formato alfanumerico.

I campi definiti come alfanumerici non abbisognano di spiegazioni supplementari rispetto alla notazione esadecimale che ne descrive fedelmente il contenuto.

29.4 Formato numerico segnato.

I campi numerici segnati sono composti da tanti byte quante sono le cifre definite per il campo. Le cifre del numero vengono ordinatamente scritte nei digit dei byte del campo. Tutti gli zone dei byte sono impostati a F salvo l'ultimo che significa il segno del numero.

- F = Segno positivo.
- D = Segno negativo.

Caso limite, il campo di una sola cifra dove la cifra occupa il digit e il segno lo zone.

L'esempio riguarda quattro valori di un campo di 5 cifre con 0 decimali.

Carattere	**00001**	**0000J**	**12345**	**1234N**
Zone	**FFFFD**	**FFFFD**	**FFFFF**	**FFFFD**
Digit	**00001**	**00001**	**12345**	**12345**
Valore	00001+	00001-	12345+	12345-

La leggibilità di un campo numerico segnato, riscontrabile durante la visualizzazione della interpretazione esadecimale del record che lo ospita, è elevata conoscendone le posizioni di inizio e fine.

29.5 Formato numerico impaccato.

I campi numerici segnati sono composti da tanti byte quanti risultano dal seguente calcolo.

NumeroByte = ParteIntera(NumeroCifre / 2) + 1

In parole, per trovare il numero di byte necessari per contenere il dato numerico, si divide per due il numero di cifre del campo numerico, si prende la parte intera del risultato e si aggiunge uno.

Se il campo numerico ha un numero di cifre pari, gli si aggiunge una cifra zero in testa. Le cifre del numero vengono poi ordinatamente scritte a partire dallo zone del primo byte seguito (se ci sono altre cifre da esprimere) dal digit del medesimo byte e via così con lo zone e il digit del secondo byte e poi dei successivi. L'ultimo digit significa il segno del numero.

- F = Segno positivo.
- D = Segno negativo.

Caso limite, il campo di una sola cifra dove la cifra occupa lo zone e il segno il digit.

L'esempio riguarda quattro valori di un campo di 5 cifre con 0 decimali.

Carattere	*******	*******	*******	*******
Zone	**001**	**001**	**135**	**135**
Digit	**00F**	**00D**	**24F**	**24D**
Valore	00001+	00001-	12345+	12345-

La leggibilità di un campo numerico impaccato, riscontrabile durante la visualizzazione della interpretazione esadecimale del record che lo ospita, è elevata anche senza conoscere le posizioni di inizio e fine perché il digit che fa da segno è ben riconoscibile dopo tutte le cifre del numero.

29.6 Formato numerico binario.

I campi numerici binari hanno due lunghezze precostituite di 2 e di 4 byte.

Il primo bit dei 16 del campo lungo 2 byte, o dei 32 del campo lungo 4 byte, è riservato al segno.

- 0 = Segno positivo.
- 1 = Segno negativo.

Il resto dei bit (15 o 31 secondo il caso) contiene il numero nella esatta forma dettata dall'algebra binaria, su cui non ci soffermiamo.

Il campo di 2 byte contiene al massimo solo 4 cifre con segno.

Il campo di 4 byte contiene al massimo solo 9 cifre con segno.

Forte stranezza, se il segno è negativo, si sottrae 1 al numero prima di convertirlo in binario e tutti i bit restanti oltre il segno sono esattamente rovesciati.

L'esempio riguarda sei valori di un campo di 9 cifre con 0 decimali.

Carattere	********	********	********	********
Zone	**0000**	**FFFF**	**05C1**	**FA3E**
Digit	**0001**	**FFFF**	**7BD5**	**842B**
Valore	000000001+	000000001-	123456789+	123456789-
Carattere	********	********		
Zone	**0000**	**FFFF**		
Digit	**0002**	**FFFE**		
Valore	000000002+	000000002-		

La leggibilità di un campo numerico binario, riscontrabile durante la visualizzazione della interpretazione esadecimale del record che lo ospita, è praticamente nulla, dovendosi convertire il numero binario in numero decimale ed eseguire le manovre dette per i negativi.

29.7 Errori nei dati numerici.

Se nel database un campo numerico contiene dati non validi, al momento della lettura il sistema reagisce con un errore grave che impedisce il proseguimento del programma. La stessa cosa avviene al primo uso di un campo numerico che per qualsiasi ragione contiene un dato numerico non valido.

Esempio di errore in un campo impaccato 5,0 dove un esadecimale destinato a contenere i valori da 0 a 9 è sostituito da un altro esadecimale da A ad F.

	giusto	**sbagliato**
Zone	**135**	**135**
Digit	**24F**	**2AF**
Valore	12345+	

Esempio di errore in un campo impaccato 5,0 dove un esadecimale destinato a contenere il segno F o D contiene invece un altro esadecimale.

	giusto	**sbagliato**
Zone	**975**	**975**
Digit	**86D**	**86B**
Valore	98765-	

Se il record è male inizializzato a causa di una manipolazione che fa uso di descrizioni interne dei file, accade che dei blank (esadecimali x'40') cadano in un campo impaccato provocando il solito errore. Segno di tale errore è nei debug il ritrovamento di valori esadecimali del tipo x'404040' per un campo numerico, ad esempio lungo 5,0. Sotto condizioni particolari (direttiva di compilazione Ignora Errori Decimali) il valore dello stesso campo numerico in memoria può diventare 40404+.

	giusto	**sbagliato**
Zone	**000**	**444**
Digit	**00F**	**000**
Valore	00000+	40404+

Errori RPG nei campi numerici sono più facili quando un campo impaccato è il sottocampo di una struttura dati. La struttura dati è assimilata dall' RPG ad un campo alfanumerico ed è inizializzato a blank, da cui l'errore descritto che colpisce al primo uso del sottocampo ma non se usato come risultato. Altrettanto facilmente si rimedia chiedendo che la struttura dati venga inizializzato nei sottocampi come indicato a 18.6.

29.8 Formato numerico preferito dal sistema.

Il sistema rappresenta tutti i campi numerici in memoria come campi impaccati con numero di cifre dispari. Ne risulta che ogni altra rappresentazione numerica viene automaticamente convertita prima di essere usata e che tale fatica viene risparmiata se i dati numerici si trovano nel database già con tali caratteristiche.

30 Aree Dati. (37R)

L'area dati è un oggetto di tipo *DTAARA. Esso consiste in uno spazio fisico definito residente sulla memoria ausiliaria (disco) atto a contenere una sola stringa di dati.

L'area dati ha una capacità di contenimento simile a quella di un file descritto internamente dotato di un solo record ma è governata da tutt'altre istruzioni.

Con il comando seguente si crea un'area dati di tipo carattere la cui dimensione è di 125 byte.

CRTDTAARA DTAARA(NomeLib/NomeDtaAra) TYPE(*CHAR) LEN(125) TEXT('Testo descrittivo area dati. ')

In quest'altro modo si sostituisce il contenuto dell'area dati dalla posizione 25 per 11 byte con il valore specificato nel parametro VALUE.

CHGDTAARA DTAARA(NomeLib/NomeDtaAra (25 11)) VALUE('NuovoValore')

Così invece si cancella l'area dati.

DLTDTAARA DTAARA(NomeLib/NomeDtaAra)

30.1 Definizione area dati e interpretazione del tracciato.

Per potere utilizzare un'area dati in un programma RPG, si deve innanzitutto definirla; un po' come si fa per i file. L'area dati viene definita nelle specifiche di calcolo.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     *DTAARA       DEFINE                    NomeDtaAra
```

Essendo l'area dati uno spazio fisico riservato per l'appoggio di dati privo di un tracciato per la interpretazione, ne deriva la necessità di poterne correttamente interpretare il contenuto. Per potere fare questa operazione si ricorre prevalentemente alla definizione un struttura dati interna al programma.

Esempio di definizione struttura dati descritta internamente al programma.

```
D   DName++++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
D NomeDtaAra      DS
D  Campo1                 1      2
D  Campo2                 3      3
D  Campo3                 4      4
D  Campo4                 5      7
D  Campo5                 8     10
```

Così facendo si interpreterà il contenuto dell'area dati dalla posizione 1 alla posizione 10, avendo a disposizione del programma RPG i cinque campi che la compongono.

A volte è necessario ricorrere alla definizione di una struttura dati descritta esternamente al programma RPG e successivamente porre il suo contenuto in schiera. Di seguito vengono illustrate le istruzioni per la definizione della schiera, struttura dati, area dati ed acquisizione del contenuto dell'area dati in schiera di lavoro.

```
D   DName++++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
 * Zone Appartenenti CEE.
D CEE             S              2    DIM(200)
 * DS Interpretativa Area Dati Zone Appartenenti CEE.
D DTAISO          DS            400
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
 * Legge area dati DTAISO e carica schiera CEE.
 *    Definisce l'area dati.
C     *DTAARA       DEFINE                    DTAISO
 *    Legge il contenuto area dati.
C                   IN        DTAISO
 *    Pulisce la schiera di lavoro.
C                   CLEAR                     CEE
 *    Imposta il contenuto dell'area dati in schiera.
C                   MOVEA     DTAISO          CEE
```

30.2 Operazioni di lettura e scrittura.

L'accesso al contenuto di una area dati da parte di un programma RPG si esegue utilizzando il codice operativo IN ovvero lettura del contenuto di una area dati.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                     IN        NomeDtaAra
```

La modifica del contenuto di una area dati si esegue utilizzando il codice operativo OUT.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C                     OUT       NomeDtaAra
```

30.3 Blocco e sblocco.

E' possibile allocare in modo esclusivo un'area dati da parte di un programma RPG. L'allocazione dell'area dati avviene nel momento in cui si acquisisce il suo contenuto.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     *LOCK           IN        NomeDtaAra
```

In questo modo si acquisisce il contenuto dell'area dati e contestualmente la si alloca.

Per deallocare l'area dati e renderla disponibile ad altri programmi/lavori, si deve eseguire una operazione di aggiornamento, indipendentemente dalla modifica del contenuto dell'area dati.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     *LOCK           OUT       NomeDtaAra
```

Si noti la stretta analogia di comportamento con i file di database.

30.4 Utilizzo di un'area dati come controllo della sequenza di elaborazione.

Spesso una procedura batch viene eseguita tramite un solo programma supervisore di controllo che, secondo i risultati parziali dei primi passi, è in grado di scegliere la strada giusta ricorrendo soltanto alle decisioni dell'utente prese aprioristicamente al momento del lancio.

Altrettanto spesso, però, la procedura batch diventa complessa e si spezza in più passi che richiedono non solo decisioni intermedie prese dall'utente durante il lancio, ma anche che una certa parte sia eseguita solo dopo che un'altra certa parte ha avuto l'esito desiderato.

Un'area dati può essere impiegata come controllo di sequenza di elaborazione nell'ambito di una procedura complessa alla stregua di un file monorecord. A favore del file monorecord gioca solo il fatto di essere asservibile a giornale e sincronia. A favore dell'area dati gioca invece la possibilità di trattarla in modo semplice sia in RPG che in CL.

Naturalmente, i parametri che regolano il funzionamento di una procedura semplice si passeranno facilmente come parametri delle CALL (sia RPG che CL) e, simmetricamente, come parametri di ingresso (*ENTRY PLIST in RPG e PGM in CL) dei programmi. Avendo validità per una sola esecuzione, non occorre fissarli né in un file né in un'area dati.

In una procedura complessa, invece, un programma asincrono rispetto agli altri va informato degli esiti ottenuti da altre parti della procedura. Se tali esiti sono annotati su un'area dati, si possono automatizzare alcune decisioni, sottraendole alla fatica dell'utente ed evitandogli errori di sequenza elaborativa. È chiaro che i parametri palleggiati tra le varie parti della procedura complessa vanno appoggiati dove possono essere condivisi facilmente. Niente di meglio che un'area dati.

Nel caso più semplice, volendo controllare l'esecuzione di una procedura complessa composta da tre procedure semplici, si stabilirà un'area dati di controllo di tre caratteri che, all'inizio dei tempi sono in bianco.

- Il lancio della prima e la prima procedura pretenderanno che tutti i caratteri siano in bianco.
- Se la prima procedura va a buon fine, scrive una X nel primo carattere, altrimenti non scrive nulla.
- Il lancio della seconda e la seconda procedura pretenderanno che il primo carattere valga X

e gli altri siano in bianco.

- Se la seconda procedura va a buon fine, scrive una X nel secondo carattere, altrimenti non scrive nulla.

30.5 Local data area.

La local data area (*LDA) è un'area dati particolare che il sistema mette a disposizione di ciascun job. E' concettualmente analoga alla libreria temporanea QTEMP ma , a differenza di questa, viene passata anche ai lavori sottomessi tramite SBMJOB (Submit Job). Può perciò essere utilizzata per l'appoggio ed il passaggio di parametri tra un programma interattivo di lancio ed uno batch di esecuzione.

La *LDA è un'area dati di tipo carattere con dimensione fissa di 1024 byte.

Per poterla utilizzare in un programma RPG la si deve definire in questo modo.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
C     *DTAARA       DEFINE    *LDA          NOMEDS
```

Quindi la si può utilizzare esattamente come una qualsiasi area dati come precedentemente descritto in questo capitolo.

La *LDA non è cancellabile, è possibile visualizzarne o modificarne il contenuto. Se si lancia il comando di visualizzazione,

DSPDTAARA DTAARA(*LDA)

Oltre al suo contenuto, queste sono le informazioni visualizzate.

```
Area dati . . . . . :   *LDA
Tipo  . . . . . :   *CHAR
Lunghezza . . . . . :   1024
Testo . . . . . :   *LDA per il Lavoro 116730/xxxxxxxxxx/yyyyyyyyy
```

Per modificare in tutto od in parte il contenuto della *LDA

CHGDTAARA DTAARA(*LDA (1 11)) VALUE('NuovoValore')

31 Subroutine RPG. (38C)

32 Date. (39C)

32.1 Formato, Costanti, Inizializzazioni data.

32.2 Operazioni sulle date MOVE, TEST(D), ADDDUR, SUBDUR.

33 Salvataggi. (40C)

33.1 Necessità dei salvataggi.

Uno dei problemi principali di tutti i generi di elaboratori proviene dalla possibilità di guasto definitivo dei dischi che ne costituiscono la memoria di massa, detto altrimenti ***crash***. I dischi moderni hanno una affidabilità ed una capacità molto elevate ma non sono indenni da rotture che provocano la perdita dei dati agli utenti finali e dei sorgenti ai programmatori.

Il rimedio al crash è costituito dalle attività di salvataggio fuori linea di tutti i dati e i programmi esistenti sul sistema.

Il salvataggio può essere totale, come nel trasferimento da una macchina vecchia ad una nuova, fatto salvo il contemporaneo eventuale aggiornamento ad un rilascio di sistema più recente.

Oppure può essere parziale, riguardando di volta in volta una delle parti in cui è opportuno dividere l'insieme.

33.2 Oggetti da salvare.

I dati da salvare vengono catalogati in base alla loro mobilità. Tanto più sono mobili i dati tanto più spesso devono essere salvati. L'elenco che segue è in ordine crescente di mobilità e deve contenere tutto quello che si desidera possedere ancora dopo la distruzione della macchina salvata.

- Il sistema ed il software di base.
- L'eseguibile dei programmi applicativi di pacchetti acquistati.
- I sorgenti dei programmi applicativi di pacchetti acquistati.
- I dati aziendali storici.
- L'eseguibile dei programmi applicativi sotto sviluppo e manutenzione.
- I sorgenti dei programmi applicativi sotto sviluppo e manutenzione.
- I dati aziendali correnti.

Per decidere però la frequenza di salvataggio, occorre tenere conto di numerose altre circostanze.

- Se di un pacchetto programmi si possiede anche il sorgente, è sicuramente preferibile privilegiare il salvataggio dei sorgenti perché gli oggetti rinascono dalla semplice ricompilazione dei sorgenti. Si faccia però attenzione: molti sprovveduti creano oggetti senza traccia di sorgente, ad esempio un printer file descritto internamente con caratteristiche di creazione strane. In questo caso si conservi il comando di creazione in una job stream rieseguibile inserita tra i sorgenti.
- A volte gli oggetti eseguibili provengono dalla compilazione su una macchina diversa e la macchina corrente non contiene qualche elemento al posto giusto: una struttura dati mancante, una opzione di compilazione taciuta dal fornitore del software, una /COPY male indirizzata. Per questa ragione l'eseguibile ha una importanza da non disconoscere.
- A volte vecchi dati aziendali sono presenti anche in librerie di prova che rientrano nei salvataggi solo per particolari ragioni (come la prova di una chiusura contabile con programmi appena ricevuti dal fornitore dei programmi contabili). Grosse quantità impreviste di dati possono mandare in overflow il salvataggio su nastro e richiedere nottetempo un intervento operativo di cambio nastro che però non viene eseguito. Un salvataggio interrotto e non intercettato su una macchina senza presidio tecnico, ripetuto per disgrazia per più giorni, porta ad un elevato rischio di crash senza salvataggi completi. È opportuno che i salvataggi delle librerie di prova sia condotto di giorno in batch e a bassa priorità da chi ne ha necessità.
- I dati storici e gli eseguibili e sorgenti di pacchetti acquistati possono essere salvati in doppia copia e dimenticati. I sorgenti che si devono personalizzare verranno sempre copiati, modificati e ricompilati nell'ambiente di modifica.

- Le varie tipologie di dati si possono presentare in varie librerie per aree diverse. Ad esempio, a volte i dati contabili di un'azienda giacciono in una libreria dati diversa da quella gestionale della stessa azienda. Ma le due librerie subiranno gli stessi salvataggi per mantenere la sincronizzazione di quei dati di una delle due applicazioni che subiscono elaborazioni anche da parte dei programmi dell'altra applicazione. Per questo sarà bene che anche il giornale sia unico.

33.3 Piano di ripristino.

Dato che, allo stato attuale dell'arte, il crash è comunque un evento insolito e scarsamente conosciuto per esperienza diretta, è indispensabile che il responsabile del sistema appronti un piano preventivo di ripartenza dopo crash.

Si procede a rovescio: decisa la modalità di ripristino e la perdita tollerabile di dati per tempo e quantità, si pianificano i salvataggi che permettono la ripartenza desiderata.

Ripristino alla sera precedente.

Una delle migliori strategie di ripristino è preparata da un'azione di salvataggio globale di routine che impatti il meno possibile sul normale utilizzo dell'elaboratore ma che sia la più frequente possibile.

Di fatto, il tempo lungo di un salvataggio globale (alcune ore), permette un solo salvataggio al giorno da eseguire di notte senza l'assistenza di un operatore.

Questa è la tecnica più diffusa quando è possibile conservare presso ogni utente interattivo una copia di tutti i documenti cartacei che originano o testimoniano puntualmente i movimenti di una intera giornata.

In caso di crash, si ripristina integralmente il salvataggio globale e, riaperta la macchina agli utenti, questi provvedono al ricaricamento ordinato delle stesse transazioni già eseguite nella giornata del crash. Naturalmente la cronologia di ricaricamento può essere critica: ad esempio, prima si eseguiranno tutti i carichi di magazzino, poi gli scarichi. Anche i programmi interattivi dovranno permettere di scegliere i dati da elaborare senza automatismi troppo rigidi. La cosa più importante è riottenere uguali agli originali i documenti fiscali che sono già usciti di casa tra il momento del salvataggio e quello del crash.

I supporti di salvataggio sono sempre nastri di elevata capacità e devono essere almeno tre perché esistano almeno due copie dei salvataggi lontano dall'elaboratore fulminato proprio durante l'azione di salvataggio. La seconda copia delle due lontane, anche se più vecchia della prima, tutela dal rischio che la prima sia danneggiata. Preferibilmente i nastri verranno periodicamente sostituiti e saranno in numero maggiore ai tre minimi sia per evitare la naturale usura che per aumentare la probabilità di trovare una copia buona tra quelle conservate. Sarà anche buona norma accantonare un salvataggio al mese, il più fresco, per recuperare informazioni d'epoca, quando se ne presentasse la necessità.

Se la probabilità di trovare un nastro danneggiato proprio al momento del suo riutilizzo è, verosimilmente, all'incirca di uno su mille, la probabilità di trovare contemporaneamente danneggiato anche il precedente è il quadrato di un millesimo, cioè un milionesimo. È evidente che una piccola cautela in più migliora enormemente le chance di ripartenza.

Ripristino alla mezz'ora precedente.

La seconda tecnica di ripristino è un semplice supplemento alla precedente e si presta a quelle situazioni in cui il numero di transazioni giornaliere è così elevato da rendere troppo oneroso il loro completo rifacimento.

Per attrezzare questo tipo di ripristino occorre mettere sotto giornale (25) tutti i file fisici del database da ripristinare. Si provvede poi a lanciare un Never Ending Program che, ad ogni mezz'ora, stacca e salva i ricevitori di giornale in accoramento su una unità a nastri dedicata allo

scopo. In caso di ricaricamento, dopo il ripristino dell'ultimo salvataggio completo, si ripristinano tutti i ricevitori salvati fino all'ultimo. Infine, si provvede all'applicazione delle modifiche al database documentate nei ricevitori con il comando APYJRNCHG (Apply Journal Change), di durata faticosissima.

Si nota che il salvataggio su nastro dei ricevitori in singola copia è il punto debole del metodo poiché è troppo dispendioso usare due unità a nastri dedicate per tale tecnica. Si nota altresì che l'applicazione dei ricevitori salvati il giorno prima annullerebbe il danno proveniente dall'ultimo salvataggio globale danneggiato, a discapito però dei tempi di esecuzione.

Tecniche integrative.

Usando protocolli di gestione che utilizzano batterie di dischi ridondanti (uguali tra loro e con un disco in più del normale), è possibile resistere, senza interruzioni immediate, alla rottura di un disco per ogni batteria. È possibile anche l'uso di un duplicato integrale dei dati sensibili. Ma tali tecniche più complesse esulano dallo scopo di questo corso.

Tuttavia, nessuna delle tecniche integrative sostituisce le cautele di base e in particolare il salvataggio quotidiano.

33.4 Salvataggio del sistema.

Per ripartire dopo un crash, il primo passo è il ricaricamento del software di base e della configurazione della macchina esattamente come era prima del danno.

Per questo, ogni volta che si apporta un aggiornamento a tale software (nuovo rilascio, applicazione di un gruppo di modifiche, nuove opzioni, nuovo hardware) è opportuno eseguire un salvataggio di sistema tramite il comando SAVSYS (Save System).

Esso genera un nastro da fornire per primo al sistema vergine quando si provvede alla prima accessione della macchina recuperata dal tecnico di manutenzione. Durante questo primo IMPL (Initial Macro Program Loading) avviene, molto automaticamente, il ricaricamento del software e della configurazione salvati.

Non solo il ripristino, ma anche il salvataggio di sistema avviene necessariamente a macchina dedicata. Occorre pianificarlo in orari non lavorativi per gli utenti.

L'assenza del salvataggio di sistema comporta il ripercorrimento di tutte le azioni di manutenzione del software di base facendo uso dei supporti originali, dal caricamento dell'ultimo rilascio in poi seguito dall'applicazione dell'ultimo cumulativo di PTF (modifiche al sistema) e di ogni altra PTF applicata dopo quello. Vanno altresì rieseguite tutte le configurazioni che non avvengono automaticamente.

È fortunatamente possibile un salvataggio della configurazione anche per altre vie.

SAVCFG (Save Configuration) da cui è possibile eseguire il RSTCFG (Restore Configuration).

RTVCFGSRC (Retrieve Configuration Source) che genera un sorgente, rieseguibile come job stream, da salvare con i sorgenti.

33.5 Salvataggio dei dati.

Avendo già discusso di numerosi aspetti del salvataggio dati in tutti i paragrafi precedenti, si elencano alcune considerazioni aggiuntive.

- I dati hanno una dimensione fisica notevole e vanno salvati prestando la massima cura al loro ingombro su nastro. Il loro salvataggio sarà sempre DTACPR(*YES) (Data compression) per minimizzare tale ingombro.
- Il salvataggio deve avvenire mentre nessun utente ha aperto per update un qualunque file da salvare. Meglio se gli utenti sono a menù. Meglio ancora se sono scollegati. Meglio se di notte con il sottosistema interattivo chiuso e nessun altro lavoro batch in corso.

- È sconsigliabile salvare meno di una libreria intera ad evitare problemi di sincronia tra i vari file ma soprattutto per semplificare il già difficile aspetto gestionale dei salvataggi. Si userà quindi sempre SAVLIB (Save Library) e ben raramente SAVOBJ (Save Object).
- Il salvataggio quotidiano non è rinunciabile. Ogni mattina occorre verificare che i salvataggi notturni abbiano avuto buon esito. Se falliscono, occorre eseguirli nell'intervallo del pranzo. Se non è possibile, occorre allertare l'utenza circa la tracciabilità dell'attività di due giorni invece che del solito giorno corrente. Se fallisce per due notti di fila, non si può dare l'uso della macchina agli utenti se non si è provveduto almeno al salvataggio dei dati aziendali e dei sorgenti sotto manutenzione. E chi non è d'accordo con me, peste lo colga!

33.6 Salvataggio dell'area personal.

Pur non esistendo un coinvolgimento forte dei dati personal giacenti nell'apposita area dell'AS/400 con il database (salve specifiche applicazioni), il salvataggio di tali dati va eseguito quotidianamente, alla pari dei dati aziendali, se ne esiste un qualunque uso.

Lo si esegue con un SAV (Save).

33.7 Salvataggio dei sorgenti.

La prima motivazione al salvataggio dei sorgenti sta nella possibilità di perdita come succede per i dati utente.

Una seconda motivazione a frequenti salvataggi dei sorgenti è costituita dal danno software provocato da modifiche incaute o peggiorative ad un sorgente o dalla cancellazione accidentale di un sorgente ancora necessario.

Come per i dati, l'unica difesa da tali perdite risiede nei salvataggi periodici, solitamente quotidiani e notturni.

Si elencano alcune considerazioni sui sorgenti, notevoli per la strategia di salvataggio.

- I sorgenti hanno una modesta dimensione fisica rispetto agli oggetti compilati ed una scarsa densità di dati significativi (normalmente un terzo della loro dimensione). Un salvataggio compresso di sorgenti è perciò rapido ed occupa poco nastro o poco disco di personal collegato all'AS/400.
- I sorgenti relativi a nuove applicazioni possono essere isolati in librerie separate, salvabili con maggiore frequenza.
- I sorgenti possono e devono essere copiati prima di ogni modifica, anche se banale, e tali copiature, di preferenza in linea e con un testo esterno del tipo **ante modifica tale**, possono risiedere nella stessa libreria dei sorgenti principali e, per poco tempo, persino nello stesso file.
- I sorgenti devono essere salvati almeno quotidianamente insieme ai dati e saltuariamente dal programmatore interessato.
- Anche per i sorgenti possono essere utili i salvataggi alla data: una settimana prima, un mese prima, sei mesi prima, uno o due anni prima. Se questa strategia è già attuata per i dati, permette il recupero di sorgenti assassinati anche molto tempo prima.

Tuttavia il salvataggio dei dati può non esaurire il problema della ricostruzione sui nuovi dischi del vecchio ambiente di programmazione. Occorre sempre cioè salvare tutto quanto permette di rimettere in sella lo sviluppo in corso.

Le esigenze dei salvataggi influiscono quindi sulle strategie dello sviluppo.

Si distingueranno perciò la libreria dei sorgenti di produzione, salvata separatamente, e le librerie dello sviluppo in corso. Una struttura possibile e consigliabile per lo sviluppo in corso è la seguente.

- Una libreria sorgenti di sviluppo, molto mobile, salvata frequentemente.
- Una libreria menù, molto statica, salvata quando cambia.

- Una libreria dati di prova, utile ma non indispensabile nei salvataggi.

Al limite, si possono unificare le librerie sorgenti e menù ma è meglio tenere separata la libreria dati di prova. Naturalmente il salvataggio dovrà includere ogni tanto anche quest'ultima.

Si osserva che ogni volta che uno sviluppo viene concluso, testato e installato, il materiale sorgente che lo riguarda sparisce dall'ambiente di sviluppo perché viene migrato nella libreria dei sorgenti di produzione.

33.8 Unità di salvataggio.

L'unità a nastro su cui avvengono i salvataggi è dimensionata al momento dell'acquisto della macchina in modo tale che il singolo nastro di massima capacità sopportato dall'unità stessa contenga da un quinto a metà della capacità dei dischi.

La capacità ideale è quella che permette di usare un unico nastro di massima capacità per tutta la vita del sistema per contenere, compressi, almeno tutti i dati aziendali ed i sorgenti in manutenzione.

Le unità a nastro sono dotate di un nastro speciale che, semplicemente inserito nell'unità, provoca la pulizia delle testine. Il nastro si esaurisce e va sostituito dopo una cinquantina di utilizzi. In caso di segnalazioni di malfunzionamento dell'unità, si provvede a tale pulizia ma, in caso di persistenza degli errori, occorre chiamare l'assistenza tecnica.

33.9 Salvataggio delle librerie sorgente da As/400 a PC e ripristino su altro As/400,

Esistono oggi alcune circostanze che permettono una soluzione molto adatta al programmatore che necessita spesso di scaricare da una macchina e ricaricare altrove le librerie sorgente che costituiscono il suo patrimonio di esempi.

- Il costo modesto dello spazio su disco As/400, solo da 10 a 100 volte superiore a quello dello spazio su PC e tuttavia modesto rispetto al passato.
- Il costo irrisorio dei dischi Usb 2.0 per il PC. In borsa porto a spasso 320 GigaByte per circa 130 euro.
- Le frequenti lontananza fisica e segregazione delle macchine As/400, e quindi delle unità di salvataggio.
- L'accesso all'As/400 tramite un PC in rete, metodo ormai universale.

Lavorando su un PC collegato in rete all'As/400 datore e dotato di emulazione (non importa quale), le operazioni per scaricare una libreria sorgente dall'As/400 datore al PC sono le seguenti.

- Dal video comandi As/400 di una sessione di emulazione terminale, creare un file di salvataggio in linea.
  CRTSAVF SAVF(QGPL/MySrcLib)
- Sempre dal video comandi As/400, salvare la libreria sorgente dentro il file di salvataggio. Usare un target release uguale a quello presente sul ricevente, se inferiore al datore; corrente, se il ricevente è uguale o superiore al datore.
  SAVLIB LIB(MySrcLib) SAVF(QGPL/MySrcLib) TGTRLS(*CURRENT) DTACPR(*YES)
- Dall'immissione comandi del Dos, chiamare Ftp tra PC e As/400 datore per copiare il file di salvataggio sul PC.

```
FTP  NomeAs400Datore
```

Alla richiesta dell"Ftp, digitare utente e password, poi, sul video comandi ftp, immettere.

```
bin
get  /QSYS.lib/QGPL.lib/MySrcLib.savf  C:\MySrcLib.savf
quit
```

Il file PC che si ottiene può essere conservato o trasmesso o pubblicato in un sito perché altri ne fruiscano. Inutile invece tentare di visionarlo sul PC causa il formato particolare dei dati salvati e causa la differente codifica ASCII-EBCDIC.

Il ricaricamento è simmetrico allo scaricamento.

Lavorando su un PC collegato in rete all'As/400 ricevente e dotato di emulazione (non importa quale), le operazioni per ricaricare una libreria sorgente dal PC all'As/400 ricevente sono le seguenti.

- Dal video comandi As/400 di una sessione di emulazione terminale, creare un file di salvataggio in linea.
  CRTSAVF SAVF(QGPL/MySrcLib)
- Dall'immissione comandi del Dos, chiamare Ftp tra PC e As/400 ricevente per copiare il file di salvataggio dal PC.

```
FTP  NomeAs400Ricevente
```

  Alla richiesta dell''Ftp, digitare utente e password, poi, sul video comandi ftp, immettere.

```
bin
put  C:\MySrcLib.savf  /QSYS.lib/QGPL.lib/MySrcLib.savf
quit
```

- Sempre dal video comandi As/400, ripristinare la libreria dal file di salvataggio.
  RSTLIB SAVLIB(MySrcLib) SAVF(QGPL/MySrcLib) RSTLIB(MyRestoredSrcLib)

Nei comandi ftp si noti "bin" che evita la trascodifica tra ASCII ed EBCDIC e viceversa.

Si noti anche che il formato di denominazione dei file menzionati nei comandi put e get non è richiesto esplicitamente perché l'ftp di dos/windows ne è privo. Il formato che si evince dal primo put o get resta attivo e non modificabile fino all'abbandono della sessione ftp con "quit". C'est a dire: se si tenta in un modo (/qsy.lib/libreria.lib/savefile.savf), per cambiarlo nell'altro (savefile.libreria) occorre abbandonare l'ftp e riaprirlo.

Per completezza, ricordo che prima di mettere in rete è comunque opportuno zippare. Raccomando l'uso degli exe autoestraenti piuttosto che i formati da estrarre con le utility tipo zip o arj. Come tutte le utility gratuite, tendono a slittare verso versioni a pagamento o, peggio, verso l'abbandono da parte degli autori.

Salvo una new entry di zippaggio a bordo dell'As/400 ad opera di Perotti, troppo recente per farci pieno affidamento, il transito in rete tra As/400 e PC passa comunque in rete al massimo ingombro.

Ricordo ancora uno zippaggio ante litteram di file sorgenti scritto dal compianto Del Bravo, beta tester del primo S/38 italiano, semplice ma efficace di corredo all'architettura 38 e comunque dimenticato da tutti. Del Bravo fu autore anche del mitico KUPDF, grande successo della stessa epoca.

33.10 Salvataggio libreria dati da As/400 a PC,

Quanto detto per le librerie sorgente vale anche per le librerie dati.

La differenza sta nelle dimensioni dei file dati, normalmente molto più grandi dei file sorgenti. Il grande ingombro ammazza la velocità della rete su cui viaggia scoraggiando un uso sistematico di questo metodo per i salvataggi di routine dei dati di produzione.

Architettura. (41C)

I programmi devono essere raggiunti ed eseguiti tramite una interfaccia facile da costruire per il programmatore e facile da comprendere anche ad un terminalista sprovveduto. La costruzione di una procedura di utilità che risolva il problema in maniera omogenea permette al programmatore delle applicazioni gestionali di concentrarsi su aspetti più applicativi che formali e di risolvere il problema con un applicativo come gli altri.

33.11 Necessità dei menù.

Chiunque abbia sviluppato un pacchetto applicativo su AS/400 ha incontrato la necessità di fornirlo di una interfaccia di menù per farlo funzionare. I menù, grazie alla loro struttura ad albero, permettono di raggiungere tutte le funzioni eseguibili di un pacchetto partendo da un solo punto di ingresso. Non si corre quindi il rischio di perdere di vista qualche funzione come avverrebbe se le chiamate fossero singole. Restano inoltre autodocumentate stabilmente l'estensione stessa del pacchetto e la chiamabilità effettiva di ogni funzione.

Il software di base fornisce varie possibilità di confezionare menù specifici mediante un supporto inserito nell'SDA che, sull'AS/400, sputacchia una trappola inutilmente complessa in termini di oggetti e sorgenti mentre sul S/38 si limitava a generare un più semplice ed efficace video e CLP.

33.12 Modulo Base IBM.

L'IBM fornisce solo in Italia per l'AS/400 una serie di pacchetti applicativi standard per aziende metalmeccaniche e simili: le ACG (Applicazioni Contabili Gestionali). A cappello e corredo delle ACG è dato anche un supporto menù e gestione sistema detto ***Modulo Base*** o, con il vecchio nome di progetto sul primo S/38, ***Architettura***.

A dire il vero, il primo S/38 era scarso di menù di sistema e richiedeva una competenza operativa non banale. Oggi l'AS/400 è dotato nel software di base di numerosi menù per raggiungere funzionalità che, con il solo settaggio al momento della prima installazione della macchina, permettono di lasciare il sistema senza un operatore dedicato ed esperto.

L'Architettura si proponeva perciò di fornire nell'applicativo quel tanto di automatismi che supplissero alle carenze e tali funzionalità sono rimaste nell'Architettura a dispetto della evoluzione del software di base.

Nel capitolo corrente si tratta del Modulo Base IBM soprattutto dal lato del programmatore applicativo che ne deve fare uso per costruire programmi nello stile ACG richiamabili dai menù del Modulo Base stesso. Un cenno soltanto sarà fatto al Modulo come gestore del sistema.

33.13 Altre soluzioni.

Quasi tutti i pacchetti di un certo ingombro hanno una sezione simile all'Architettura ma di gran lunga più semplice.

Come esempio di pacchetto che risolve in modo autonomo la sezione menù e gestione del sistema, si menziona quello della Multi Consult Milano, dedicato a trasporti, logistica e viciniori.

Più frequentemente, in Italia è uso sviluppare in modo compatibile con il Modulo Base IBM soprattutto se il pacchetto non si impiccia direttamente di contabilità e coopera invece con la Contabilità ACG. Di questa categoria fanno sicuramente parte tutti i gestionali con un numero di installazioni modesto come pure i pacchetti integrativi alle ACG (in illo tempore ADB) che costituivano il cavallo di battaglia delle prime case di software di area S/38 come la Datasys e la Cata.

Altre volte, ancora, il pacchetto ha un suo proprio modulo base ma tutti i programmi sono in grado di comportarsi correttamente anche sotto Architettura. Si veda, ad esempio il pacchetto

tessile della Datatex che, con lo stesso compilato, funziona in Italia spessissimo sotto Architettura e all'estero solo sotto il proprio modulo base.

Qualcuno ha anche scopiazzato la parte menù dell'Architettura del vecchio S/38, della quale venivano forniti tutti i sorgenti, riportandola in vita anche sull'AS/400 in modo da renderla compatibile con i file dell'Architettura corrente.

In comense (c'est a dire: in comasco dentro le mura): "Cen cò, cen crap!" Cento teste, cento idee!

33.14 Uso applicativo del Modulo Base.

L'utente AS/400 che usa l'Architettura viene creato tramite un'apposita azione eseguita dal profilo ACGMASTER, installato dall'Architettura stessa. Al profilo utente viene inoltre assegnato un programma iniziale blindato BCI00 che presenta un menù iniziale con accesso esclusivamente alle azioni previste dai menù in catena. Altra caratteristica obbligatoria è la lista librerie, altrimenti detta sistema informativo, da usare nel lavoro interattivo e negli eventuali batch da esso sottomessi.

Fin qui un normale utente finale.

Naturalmente, se un'azione lo permette, sono agibili anche le funzioni operative e di programmazione ma la scelta è infelice perché le caratteristiche di un utente architetturale non coincidono con quelle auspicabili per un programmatore.

L'azione di modifica dell'utente architetturale, per giunta, si appropria dell'utente AS/400 in modo devastante rispetto ad eventuali personalizzazioni praticate da un programmatore sul proprio profilo.

Nulla vieta di creare il profilo da Architettura, modificarlo per programmare e non toccarlo più con la modifica utente architetturale.

Il miglior modo di andare sotto architettura per un programmatore è tuttavia usare il comando KUSOACG (Usa ACG). Prima di usarlo per la prima volta occorre che il responsabile della sicurezza garantisca all'utente interessato, o al suo capogruppo, il diritto d'uso con il comando GRTOBJAUT.

I parametri da fornire sono il sistema informativo e il menù iniziale, gli stessi che definiscono un utente architetturale, con la differenza che il programmatore sceglie ogni volta i parametri che gli comodano.

33.15 Sistemi informativi architetturali.

Il sistema informativo coincide concettualmente con la lista librerie (1.11) e viene definito in un record del file ACGGAA/KFSIF00F tramite una apposita azione eseguibile dall'utente ACGMASTER. Qui in realtà si definisce solo la parte centrale delle quattro che compongono la user library list completa.

- In testa si pone automaticamente la libreria QTEMP (Temporanea del job in corso).
- In seconda posizione l'elenco delle librerie in questione.
- In terza posizione un elenco di librerie comuni a tutti i sistemi informativi architetturali, definito a sua volta con azione apposita.
- In quarta e ultima posizione, fisse, ACGGAA (Programmi di servizio di Architettura) e QGPL (General Purpose Library).

Si esemplifica l'immissione di un sistema informativo composto da quattro librerie.

- ESEMNU Menù esempio.
- ESEDAT Dati esempio.
- ESEOBJ Programmi esempio.
- ESESRC Sorgenti esempio.

Esempio menù di chiamata gestione Sistemi Informativi.

```
                          Gestione degli oggetti con il PDM            S44524BA

Libreria. . . . .   NERONI____       Inizio elenco da. . . . . .   __________
                                     Inizio elenco da tipo . . .   __________

Immettere le opzioni e premere Invio.
  2=Modifica     3=Copia       4=Cancellaz.  5=Visualiz.     7=Ridemoninaz.
  8=Visualiz. descrizione      9=Salvatag.  10=Ripristino   11=Spostamento...

Opz  Oggetto     Tipo        Attributo   Testo
__   BNT01       *PGM        RPGLE       Bontà.
__   BTT01W      *FILE       DSPF        Bottoni.
__   BTTSRC      *FILE       PF-SRC      Bottoni. Source.
__   EVENT00F    *FILE       PF-DTA      Eventi.
__   RIGAD00F    *FILE       PF-DTA      Righe documenti.
__   TESTN01L    *FILE       LF          Testate dolenti.
__   TRAMV00P    *FILE       PRTF        Stampa del tram.

                                                                       Fine
Parametri o comando
===> ____________________________________________________________________________
F3=Fine           F4=Richiesta          F5=Rivisualizzaz.       F6=Creazione
F9=Duplicazione   F10=Immissione comandi    F23=Altre opzioni   F24=Altri tasti
```

Esempio video di aggiunta Sistema Informativo.

```
                          Gestione degli oggetti con il PDM            S44524BA

Libreria. . . . .   NERONI____       Inizio elenco da. . . . . .   __________
                                     Inizio elenco da tipo . . .   __________

Immettere le opzioni e premere Invio.
  2=Modifica     3=Copia       4=Cancellaz.  5=Visualiz.     7=Ridemoninaz.
  8=Visualiz. descrizione      9=Salvatag.  10=Ripristino   11=Spostamento...

Opz  Oggetto     Tipo        Attributo   Testo
__   BNT01       *PGM        RPGLE       Bontà.
__   BTT01W      *FILE       DSPF        Bottoni.
__   BTTSRC      *FILE       PF-SRC      Bottoni. Source.
__   EVENT00F    *FILE       PF-DTA      Eventi.
__   RIGAD00F    *FILE       PF-DTA      Righe documenti.
__   TESTN01L    *FILE       LF          Testate dolenti.
__   TRAMV00P    *FILE       PRTF        Stampa del tram.

                                                                       Fine
Parametri o comando
===> ____________________________________________________________________________
F3=Fine           F4=Richiesta          F5=Rivisualizzaz.       F6=Creazione
F9=Duplicazione   F10=Immissione comandi    F23=Altre opzioni   F24=Altri tasti
```

Un esempio di utilizzo un po' anomalo del file dei sistemi informativi può essere trovato nel capitolo sui comandi (12).

33.16 Azioni e menù architetturali.

Un sistema informativo architetturale è fortemente caratterizzato dai file di controllo della libreria in testa alla lista librerie che gli è propria. Tali file sono generati da apposita azione del sistema informativo di controllo (ACGGAA) e caricati di dati tramite altre azioni eseguite nel sistema informativo in causa.

I file di controllo fisici sono essenzialmente.

- KFAZN10F Azioni.
- KFKPJ00F Prolunga azioni.
- KFMNU10F Menù.

33.17 Azioni architetturali.

Il fulcro di una applicazione di gestione menù è l'associazione tra una scelta sul menù ed il programma da chiamare in attuazione della scelta.

L'intermediazione tra le due cose è svolta sotto Architettura dal file KFAZN00F (Azioni) che si deve trovare nella prima libreria del sistema informativo. Il file viene gestito da un apposito programma

Esempio video di aggiunta Azione.

```
                        Gestione degli oggetti con il PDM           S44524BA

Libreria. . . . .   NERONI____       Inizio elenco da. . . . . .   __________
                                     Inizio elenco da tipo . . .   __________

Immettere le opzioni e premere Invio.
  2=Modifica     3=Copia       4=Cancellaz.  5=Visualiz.     7=Ridemoninaz.
  8=Visualiz. descrizione      9=Salvatag.  10=Ripristino   11=Spostamento...

Opz  Oggetto     Tipo        Attributo   Testo
__   BNT01       *PGM        RPGLE       Bontà.
__   BTT01W      *FILE       DSPF        Bottoni.
__   BTTSRC      *FILE       PF-SRC      Bottoni. Source.
__   EVENT00F    *FILE       PF-DTA      Eventi.
__   RIGAD00F    *FILE       PF-DTA      Righe documenti.
__   TESTN01L    *FILE       LF          Testate dolenti.
__   TRAMV00P    *FILE       PRTF        Stampa del tram.

                                                                      Fine
Parametri o comando
===> ____________________________________________________________________________
F3=Fine           F4=Richiesta          F5=Rivisualizzaz.       F6=Creazione
F9=Duplicazione   F10=Immissione comandi    F23=Altre opzioni   F24=Altri tasti
```

Il record azione porta essenzialmente il nome del programma da chiamare e la descrizione con cui l'azione verrà menzionata nei menù. Qualche rara volta è utile anche la possibilità alternativa di chiamare comandi invece che programmi, con la possibilità supplementare di parametrizzare.

L'Architettura controlla le azioni in tre modi diversi.

- N L'azione è elencabile a menù per essere chiamata ma nessun altro servizio è fornito.
- X L'azione è anche tracciata da un log apposito, consultabile con apposite azioni.
- S Il programma riceve anche un maxi parametro, la KPJBA, che fornisce banali informazioni sul job ma, soprattutto, permette di sottomettere azioni batch egualmente governate dall'Architettura, con anche un facile passaggio di parametri.

Il record azione contiene anche la modalità di avvio del controllo di sincronia (25.6).

- N Non è richiesto il controllo di sincronia.
- A È richiesto il controllo di sincronia STRCMTCTL LCKLVL(*CHG).
- T È richiesto il controllo di sincronia STRCMTCTL LCKLVL(*ALL). Non usare mai questo valore senza attente disquisizioni sul controllo di sincronia.

Battendo Invio sul video Azione si ottiene il video Oggetti allocati.

Esempio video di aggiunta Azione-Oggetti allocati.

```
                          Gestione degli oggetti con il PDM           S44524BA

Libreria. . . . .   NERONI____       Inizio elenco da. . . . . .   __________
                                     Inizio elenco da tipo . . .   __________

Immettere le opzioni e premere Invio.
  2=Modifica     3=Copia       4=Cancellaz.  5=Visualiz.     7=Ridemoninaz.
  8=Visualiz. descrizione      9=Salvatag.  10=Ripristino   11=Spostamento...

Opz  Oggetto     Tipo        Attributo   Testo
__   BNT01       *PGM        RPGLE       Bontà.
__   BTT01W      *FILE       DSPF        Bottoni.
__   BTTSRC      *FILE       PF-SRC      Bottoni. Source.
__   EVENT00F    *FILE       PF-DTA      Eventi.
__   RIGAD00F    *FILE       PF-DTA      Righe documenti.
__   TESTN01L    *FILE       LF          Testate dolenti.
__   TRAMV00P    *FILE       PRTF        Stampa del tram.

                                                                      Fine
Parametri o comando
===> ____________________________________________________________________________
F3=Fine           F4=Richiesta          F5=Rivisualizzaz.       F6=Creazione
F9=Duplicazione   F10=Immissione comandi    F23=Altre opzioni   F24=Altri tasti
```

Se si vuole allocare qualche file per l'elaborazione compiuta dall'azione, invece che eseguire le allocazioni nel CL di controllo azione, è possibile elencare qui i file desiderati. Si ottiene il vantaggio di non scrivere un CL apposta ma, soprattutto, si ottiene un rifiuto a menù se la risorsa non è disponibile. Se poi l'azione è batch, l'azione che tenta di partire senza le risorse disponibili si autorisottomette congelata.

Battendo F13 sul video Azione si ottiene il video KPJBU iniziale.

Esempio video di aggiunta Azione-KPJBU iniziale.

```
                          Gestione degli oggetti con il PDM           S44524BA

Libreria. . . . .   NERONI____       Inizio elenco da. . . . . .   __________
                                     Inizio elenco da tipo . . .   __________

Immettere le opzioni e premere Invio.
  2=Modifica     3=Copia       4=Cancellaz.  5=Visualiz.     7=Ridemoninaz.
  8=Visualiz. descrizione      9=Salvatag.  10=Ripristino   11=Spostamento...

Opz  Oggetto     Tipo        Attributo   Testo
__   BNT01       *PGM        RPGLE       Bontà.
__   BTT01W      *FILE       DSPF        Bottoni.
__   BTTSRC      *FILE       PF-SRC      Bottoni. Source.
__   EVENT00F    *FILE       PF-DTA      Eventi.
__   RIGAD00F    *FILE       PF-DTA      Righe documenti.
__   TESTN01L    *FILE       LF          Testate dolenti.
__   TRAMV00P    *FILE       PRTF        Stampa del tram.

                                                                      Fine
Parametri o comando
===> ____________________________________________________________________________
F3=Fine           F4=Richiesta          F5=Rivisualizzaz.       F6=Creazione
F9=Duplicazione   F10=Immissione comandi    F23=Altre opzioni   F24=Altri tasti
```

Se si vuole richiamare un programma con parametrizzazioni diverse a seconda dell'azione chiamante, si può fornire un valore iniziale alla parte utente (KPJBU) del parametro architetturale KPJBA. Si nota che non basta l'Invio per aggiornare i dati di questo video ma occorrono o F13 per prendere i dati dalla riga caratteri o F14 per i dati esadecimali.

33.18 Il parametro architetturale KPJBA.

La struttura dati KPJBA, fornita dall'Architettura, è riportata di seguito per documentazione.

```
Program Status Data Structure for RPG ILE.
 1   1  10 PS_PROC     A  10    Procedure name
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
 2  11  15 PS_STATUS   S   5 5  Status code
```

La KPJBA contiene tre tipi di campi.

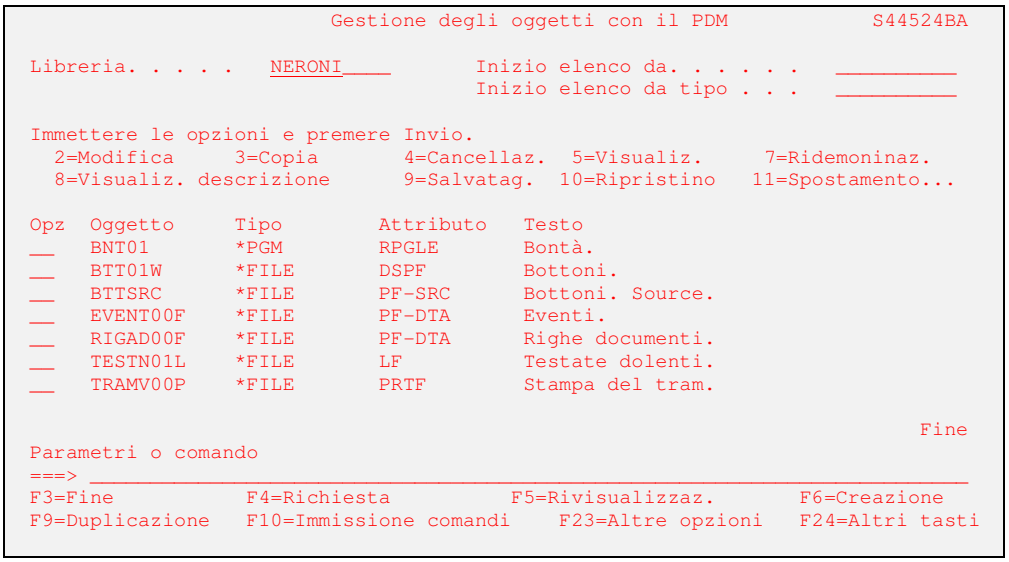
- A Da leggere per informazione.
- B Modificabili in valori predeterminati per cambiare le caratteristiche dei lavori da sottomettere.
- C Modificabili liberamente per annotare parametri da rileggere più tardi nel medesimo lavoro o nei lavori sottomessi. In realtà un solo campo (KPJBU) ha questa caratteristica.

Commentare qualche campo.

33.19 Menù architetturali.

Un menù architetturale è memorizzato tramite un apposito programma nel file KFMNU10F (Menù) che, come il file azioni KFAZN10F, si deve trovare nella prima libreria del sistema informativo.

Esempio video di aggiunta Menù.



Ogni menù raduna fino a 14 chiamate ad altri menù o alle azioni già codificate nel file azioni.

La chiamata in esecuzione di una azione o di un menù avviene digitando il numero progressivo (scelta o opzione) che l'Architettura associa automaticamente alle righe significative del menù.

Alcune regole facilitano fortemente la comprensione e la percorribilità di una catena di menù.

- La descrizione di ogni menù inizierà con la dicitura ***Menù*** o simile, purché fissa. Essa permette di intraprendere a cuor leggero l'opzione corrispondente, essendo evidente che non verrà chiamato in esecuzione nulla di irrevocabile.
- Ogni programma interattivo non eseguirà niente di significativo e di irripetibile prima di presentare il primo video. Normalmente un programma di gestione dati presenterà un inoffensivo video guida ed ogni programma di lancio batch richiederà un tasto di conferma prima di sottomettere in esecuzione alcunché.
- Un programma batch privo di prologo richiederà sempre il tasto di conferma prima di avviarsi. La definizione dell'azione permette di realizzare la conferma evitando così che gironzolando per i menù si possa scatenare un'elaborazione indesiderata.
- Preferibilmente i menù conterranno, in alternativa, o solo menù o solo azioni. Mescolare azioni e menù porta a squilibri nell'albero dei menù stessi.
- Il numero di scelte che portano ad una qualsiasi azione sarà preferibilmente uguale per tutte le azioni ed sarà inoltre il più basso possibile. È chiaro che questa regola diventa difficile da rispettare quando il numero delle azioni si allarga fino a richiedere la riorganizzazione dei menù.
- Tutte le funzioni necessarie o anche solo utili applicativamente saranno definite come azioni. Se non meglio organizzate in programmi più complessi, anche le attività più semplici, come la pulizia periodica di un file, prenderanno posto tra le azioni e compariranno a menù. L'esperienza insegna che un'azione non esplicitata in una riga di menù viene dimenticata o sospettata di incompletezza. Lo stesso vale per un programma non raggiunto tramite un'azione. La presenza di un'azione bene descritta in un menù bene ordinato ne testimonia l'eseguibilità e la necessità. Possono invece non andare a menù i seguenti.
  - Programmi di conversione la cui esecuzione non è normalmente ripetibile.
  - Programmi che agiscono su file non più presenti a regime.
  - Programmi di correzione estemporanea.

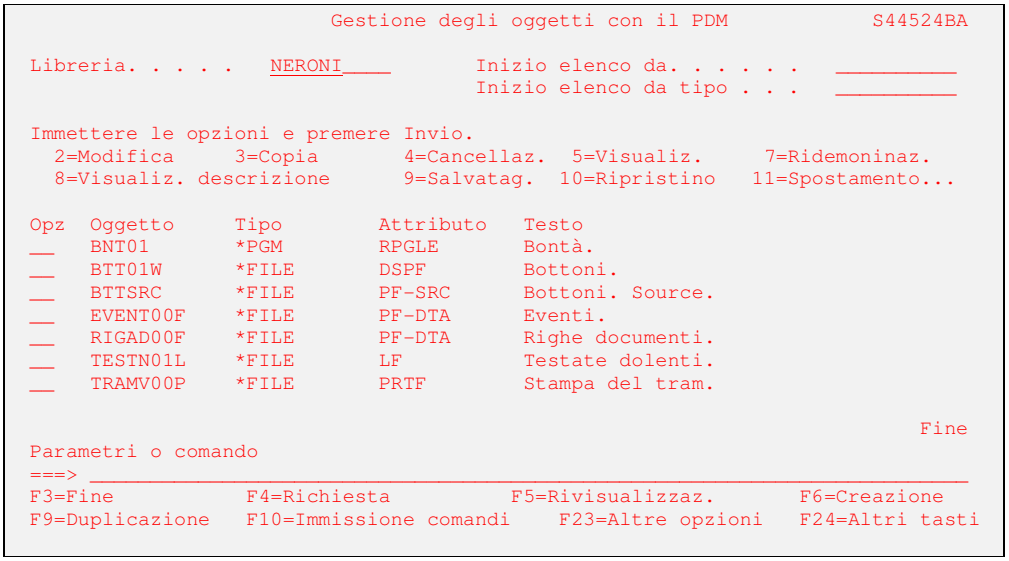
33.20 Azioni batch architetturali.

Con il solo uso del software di base, la sottomissione dei lavori batch è discretamente macchinosa. Nel caso peggiore, il programmatore risolve in maniera estemporanea il problema. Nel caso migliore, l'operazione viene intermediata da un programma CL generico di utilità costruito per il pacchetto. Sostanzialmente, la sottomissione avviene tramite il comando SBMJOB con una opportuna stringa inserita nel parametro CMD. La stringa viene composta in modo da contenere la CALL necessaria all'esecuzione batch e i PARM della CALL opportunamente riempiti. La difficoltà della cosa sta nella costruzione della stringa con riguardo alla presenza nei dati di apici, di campi numerici impaccati, di valori esprimibili solo con notazione esadecimale e quant'altre complicazioni si possono immaginare nell'interfaccia CALL.

L'Architettura fornisce invece un ottimo servizio al lancio delle azioni batch. Sostanzialmente, basta riempire di dati il campo KPJBU e chiamare il programma di servizio BCH10 passandogli il solo parametro KPJBA.

Prima del richiamo a BCH10, su richiesta l'interattivo può richiamare il modulo BCH09 che emette un video capace di modificare alcuni campi della KPJBA.

Esempio video di parametrizzazione del lancio batch BCH09.



Eventuali dati provenienti dal programma di lancio possono essere inseriti nel campo KPJBU mediante una struttura dati di palleggio. È la struttura dati, conosciuta anche dal programma lanciato, ad occuparsi del trasporto dei dati, qualunque sia la loro forma.

Se la lunghezza della KPJBU (256 byte) è troppo scarsa, si riaprono le danze. Seguendo il concetto, si inventerà un file di appoggio più ampio dotato di una chiave unica progressiva. I dati verranno appoggiati in un record di tale file. La KPJBU trasporterà esclusivamente la chiave unica con la quale il lanciato riagguanterà i dati inseriti dal lanciatore nell'ampio record di comodo.

33.21 Log architetturale.

Tutte le azioni chiamate in esecuzione tramite menù o sottomesse tramite il modulo di lancio batch BCH10 vengono annotate in un file di servizio centralizzato (ACGGAA/KFLOG00F). La registrazione è gestita da quei programmi architetturali che eseguono l'azione interattiva o l'azione batch. La registrazione avviene quindi in parte prima ed in parte dopo l'esecuzione del programma titolare dell'azione.

Le azioni batch transitano preliminarmente in un file separato (ACGGAA/KFBAT00F), costruito in modo analogo, ma, non appena avviate in esecuzione, entrano nel log principale.

Il file di log contiene, per ogni azione in esecuzione o eseguita, l'immagine del parametro architetturale più qualche informazione di servizio.

Pur essendo utilissimo alla gestione dei lavori mentre questi avvengono, normalmente il log viene spazzato automaticamente tutte le notti e serve quindi a poco per documentazione successiva.

Finché i dati del log sono in linea, tramite le interrogazioni sul log è possibile valutare ogni singola azione per quanto riguarda utente, inizio e fine ma non per i dati afflitti dall'azione.

In illo tempore i record ADB portavano un campo che diceva quale azione aveva aggiornato da ultimo il record stesso con l'ambizione quindi di sapere chi e quando avesse svolto l'ultimo aggiornamento. Tale campo non è mai stato di nessuna utilità perché affidato alla buona volontà di programmatori frettolosi ed è stato prima dimenticato e poi abolito. Il riferimento vale solo se si conserva il log indefinitamente, cosa non praticabile con semplicità. Molti espedienti suggeriti grossolanamente dall'impianto architetturale sono in realtà poco e niente sviluppati dentro l'Architettura stessa e forse non val la pena di portare troppi fiori freschi su una tomba vecchia.

In particolare, le azioni che visualizzano, ricercano e stampano il log sono di modesto aiuto nella individuazione e nella correzione dei problemi, ed il programmatore deve fare uso di mezzi più mirati come gli ambienti di prova, il giornale ed il debug.

33.22 Controllo del sistema da parte dell'Architettura.

L'architettura controlla, a richiesta, cose come le seguenti.

- Accensione e spegnimento del sistema in giorni e ad orari prefissati. Si interseca e si integra con il comando CHGPWRSCD (Change Power Schedule).
- Avvio ed arresto dei sottosistemi sia interattivi diurni che batch, diurni e notturni.
- Gestione delle code di immissione dei lavori batch con particolare riguardo all'innesco delle attività diurne e notturne.
- Esecuzione di salvataggi in linea e fuori linea ad inizio e a fine attività.
- Esecuzione dei programmi di ripartenza per aggiustare il database dopo una caduta di sistema.
- Esecuzione di programmi personalizzabili di accensione e spegnimento.

Preme osservare che, evolutosi il software di base del sistema, molte funzioni architetturali risultano oggi essere un doppione di prestazioni native dell'AS/400. Né si può fare a meno di dire che la diffusione italiana dell'AS/400, seconda solo a quella statunitense, dipende proprio dalla presenza dell'Architettura che ha automatizzato il funzionamento del sistema molto tempo prima che il software di base se ne facesse carico. Naturalmente un cliente italiano di buona volontà e con qualche aggancio americano avrebbe potuto reperire un surrogato di architettura tra le tante trappole scritte dagli ex programmatori dei centri di sviluppo IBM. Il fenomeno di attecchimento dell'AS/400 in Italia grazie alla presenza dell'Architettura è simile a quello che, in altro argomento, ha fatto prevalere lo stesso AS/400 sui mainframe di fascia bassa perché dotato di un database nativo e garantito a dispetto delle molte ma più onerose soluzioni possibili sui mainframe.

Non si documentano meglio le funzioni di controllo architetturali perché poco coinvolte nella programmazione applicativa al livello che stiamo trattando.

33.23 Il fissaggio del parametro architetturale KPJBA.

Normalmente il programma applicativo titolare dell'azione riceve il parametro architetturale KPJBA e lo manipola per proprio uso. Ad esempio, per passare parametri da un programma all'altro dentro la procedura corrente. Oppure, per sottomettere una azione batch.

Il record di log nasce prima della chiamata in esecuzione del titolare e viene aggiornato solo dopo la sua fine. Può tuttavia rendersi necessario un aggiornamento in corso di esecuzione del titolare per scopi di ripartenza dopo caduta di sistema.

Per questa necessità si usa il modulo BRK10 che riceve la variabile architetturale KPJBA e la fissa immediatamente nel log dell'azione.

Gran parte dell'utilità di questo modulo è sparita nei programmi interattivi dopo la nascita del controllo di sincronia, che rende superfluo annotare dati di ripartenza per una singola transazione. Resta invece utile fissare lo stato di avanzamento di una procedura batch tra un programma e l'altro per poter riavviare una procedura caduta. Ovviamente, usare il log architetturale come supporto per una procedura batch significa fidarsi delle vicende architetturali in caso di caduta. Le numerose ed incerte manipolazioni avvenute nel corso degli anni al motore architetturale hanno sgangherato non poco la primitiva fiducia ed il rinnovarsi di dolorose infanzie del controllo sistema hanno convinto molti a usare più solidi dati di appoggio.

34 Edit. (42C)

I campi alfanumerici vengono presentati senza intermediazioni.

Invece il contenuto dei campi numerici viene presentato a video o su stampa dopo essere stato trattato automaticamente da routine di sistema evocate da apposite parole chiavi nelle specifiche DDS o da opportune scelte sulle specifiche RPG di emissione.

Il trattamento consiste nelle seguenti azioni possibili.

- Interposizione di punti separatori delle migliaia.
- Interposizione della virgola decimale.
- Aggiunta del segno se il campo è negativo.
- Eliminazione degli zeri non significativi.
- Sbiancamento totale dell'emissione di un campo con valore zero.
- Copertura degli zeri non significativi con asterischi.
- Aggiunta di un simbolo di divisa.
- Interposizione del carattere separatore delle date (DATSEP del lavoro, di solito /).

Per ottenere il trattamento complessivo desiderato, si usano o codici o parole di editazione sia nelle specifiche di definizione dei file video e stampa che nelle specifiche RPG delle stampe interne.

Il termine editazione in italiano non americanofilo si potrebbe tradurre ***edizione*** o, più arcaicamente, ***acconciamento***.

Il segno meno sotto forma CR è di uso prettamente americano. Lo si riporta per completezza.

34.1 Edit code nelle DDS.

L'editazione di un campo del video o della stampa descritta esternamente si ottiene attribuendo al campo numerico un codice tra quelli previsti. La specifica DDS necessaria è, ad esempio.

```
DP  AAN01N02N03T.Nome++++++RLun++TPdBRigColFunzioni++++++++++++++++++++++++++++++
A                NUMERO          9  2  10 20EDTCDE(K)
```

La più parte dei codici permessi è elencata nella tabella sottostante. Tutti inseriscono la virgola decimale ed eliminano gli zeri non significativi.

Codice di edizione	Separatore delle migliaia	Segno negativo flottante a sinistra	Segno negativo a destra	Emissione del valore zero per i valori di sistema QDECFMT		
				Blank	I	J
1	si			.00 o 0	,00 o 0	0,00 o 0
2	si					
3				.00 o 0	,00 o 0	0,00 o 0
4						
A	si		CR	.00 o 0	,00 o 0	0,00 o 0
B	si		CR			
C			CR	.00 o 0	,00 o 0	0,00 o 0
D			CR			
J	si		-	.00 o 0	,00 o 0	0,00 o 0
K	si		-			
L			-	.00 o 0	,00 o 0	0,00 o 0
M			-			
N	si	-		.00 o 0	,00 o 0	0,00 o 0
O	si	-				
P		-		.00 o 0	,00 o 0	0,00 o 0
Q		-				

Gli altri codici richiedono diversa spiegazione.

Il codice W serve ad editare campi numerici da 5 a 8 cifre contenenti date e l'esito dipende dalla lunghezza del campo.

Nr cifre	5	6	7	8
C=Secolo Y=Anno M=Mese D=Giorno	YY/DDD	CCYY/MM	CCYY/DDD	CCYY/MM/DD
z=Elimina zeri n=Non elimina	zn/nnn	zzzn/nn	zzzn/nnn	zzzn/nn/nn

Il codice Y serve ad editare campi numerici da 3 a 8 cifre contenenti date e l'esito dipende dalla lunghezza del campo.

Nr cifre	3	4	5	6	7	8
C=Secolo Y=Anno M=Mese D=Giorno	MM/Y	YY/MM	DD/MM/Y	DD/MM/YY	CYY/MM/DD	DD/MM/CCYY
z=Elimina zeri n=Non elimina	zn/n	zn/nn	zn/nn/n	zn/nn/nn	zzn/nn/nn	zn/nn/nnnn

Sia per il codice W che per il codice Y, le ipotesi di utilizzo Secolo–Anno–Mese-Giorno a seconda della lunghezza del campo sono di pura fantasia.

Il codice Z elimina gli zeri e non evidenzia il segno.

Il codice X demanda l'edit del campo al keyboard shift (posizione 35 della specifica di definizione del campo). Tecnica fumosa che non viene descritta.

Nella tabella seguente si esemplifica qualche editazione, supponendo in vigore il valore di sistema QDECFMT = J, di effettivo uso sulle macchine italiane.

Codice	Positivo 5+2dec	Positivo 7+0dec	Negativo 4+3dec	Negativo 3+0dec	Zero 4+2dec	Zero 6+0dec
Valore	12345,67	1234567	0000,125-	125-	000000	000000
1	12.345.67	1.234.567	0,125	125	0.00	0
2	12.345,67	1.234.567	0,125	125		
3	12345,67	1234567	0,125	125	0.00	0
4	12345,67	1234567	0,125	125		
A	12.345,67	1.234.567	0,125CR	125CR	0.00	0
B	12.345,67	1.234.567	0,125CR	125CR		
C	12345,67	1234567	0,125CR	125CR	0.00	0
D	12345,67	1234567	0,125CR	125CR		
J	12.345,67	1.234.567	0,125-	125-	0.00	0
K	12.345,67	1.234.567	0,125-	125-		
L	12345,67	1234567	0,125-	125-	0.00	0
M	12345,67	1234567	0,125-	125-		
N	12.345,67	1.234.567	-0,125	-125	0.00	0
O	12.345,67	1.234.567	-0,125	-125		
P	12345,67	1234567	-0,125	-125	0.00	0
Q	12345,67	1234567	-0,125	-125		
W	no	1234/567	no	no	no	0/000
Y	no	123/45/67	no	12/5	no	0/00/00
Z	1234567	1234567	125	125		

La dicitura no sta per non possibile.

34.2 Edit code nell' RPG.

L'editazione di un campo emesso su un file descritto internamente (ad esempio, una stampa) si ottiene attribuendo al campo numerico un codice tra quelli previsti.

La specifica di controllo RpgIle necessaria è, ad esempio.

```
H    HParole chiave++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
H decedit('0,') datedit(*dmy/)
```

La specifica di emissione RpgIle necessaria è, ad esempio.

```
P    O.............N01N02N03Field+++++++++YB.End++PConstant/editword/DTformat++
O                       z2nr92        k     20
```

La tabella sottostante è praticamente identica a quella riportata nel paragrafo precedente per le DDS. La sola differenza è quella relativa al parametro che determina punti, virgole e zeri.

Codice di edizione	Separatore delle migliaia	Segno negativo flottante a sinistra	Segno negativo a destra	Emissione del valore zero per i valori del parametro DECEDIT sulla specifica H di controllo			
				'.'	','	'0,'	'0.'
1	si			.00 o 0	,00 o 0	0,00 o 0	0.00 o 0
2	si						
3				.00 o 0	,00 o 0	0,00 o 0	0.00 o 0
4							
A	si		CR	.00 o 0	,00 o 0	0,00 o 0	0.00 o 0
B	si		CR				
C			CR	.00 o 0	,00 o 0	0,00 o 0	0.00 o 0
D			CR				
J	si		-	.00 o 0	,00 o 0	0,00 o 0	0.00 o 0
K	si		-				
L			-	.00 o 0	,00 o 0	0,00 o 0	0.00 o 0
M			-				
N	si	-		.00 o 0	,00 o 0	0,00 o 0	0.00 o 0
O	si	-					
P		-		.00 o 0	,00 o 0	0,00 o 0	0.00 o 0
Q		-					

Il codice Y serve ad editare campi numerici da 3 a 9 cifre contenenti date e l'esito dipende dalla lunghezza del campo. Le ipotesi di utilizzo Secolo-Anno-Mese-Giorno a seconda della lunghezza del campo sono di pura fantasia.

Codice Y. Per i campi da 3 a 7 cifre vale la seguente tabella.

Nr cifre	3	4	5	6	7
C=Secolo Y=Anno M=Mese D=Giorno	MM/Y	YY/MM	DD/MM/Y	DD/MM/YY	CYY/MM/DD
z=Elimina zeri n=Non elimina	zn/n	zn/nn	zn/nn/n	zn/nn/nn	zzn/nn/nn

Codice Y. Per i campi di 8 e 9 cifre con DATFMT(*DMY/) vale la seguente tabella.

Nr cifre	8	9
C=Secolo Y=Anno M=Mese D=Giorno I=Insignificante	DD/MM/CCYY	IDD/MM/CCYY
z=Elimina zeri n=Non elimina	zn/nn/nnnn	zzn/nn/nnnn

Codice Y. Per i campi di 8 e 9 cifre con DATFMT(*YMD/) vale la seguente tabella.

Nr cifre	8	9
C=Secolo Y=Anno M=Mese D=Giorno I=Insignificante	CCYY/MM/DD	ICCYY/MM/DD
z=Elimina zeri n=Non elimina	zzzn/nn/nn	zzzzn/nn/nn

Il codice Z elimina gli zeri e non evidenzia il segno.

34.3 Edit word nelle DDS.

Se, nelle stesse circostanze viste per i codici di edizione, si desidera una editazione non ottenibile nel modo illustrato, è possibile costruire una editazione estemporanea con le caratteristiche desiderate attribuendo al campo numerico la parola chiave EDTWRD. La specifica DDS necessaria è, ad esempio.

```
DP   AAN01N02N03T.Nome++++++RLun++TPdBRigColFunzioni++++++++++++++++++++++++++++
A              NUMERO          9  2  10 20EDTWRD(' .   .   EURO&  CENT-')
```

La parola in esempio provocherebbe la emissione dei valori come segue.

Valore	Valore editato
1234567,89	1.234.567EURO 89CENT
1234567,89-	1.234.567EURO 89CENT-
0000012,34	12EURO 34CENT
0000012,34-	12EURO 34CENT-
0000000,12	12CENT
0000000,00	CENT

Al momento dell'emissione del campo, le cifre e la stringa di edizione contribuiscono alla formazione del valore editato risultante.

Per prima cosa, si identificano i caratteri della stringa di edizione che possono essere sostituiti ordinatamente dalle cifre del campo. Nella parola di edizione, sono caratteri sostituibili i blank, ed altri che poi vedremo.

In secondo luogo si sostituisce l'ultimo carattere sostituibile con la l'ultima cifra del campo numerico. Si prosegue verso sinistra trascrivendo le cifre ma anche i caratteri di servizio che si incontrano tra le cifre trascritte fino ad incontrare lo zero a sinistra della prima cifra significativa. Da tale zero in poi non si trascrive più nulla.

Nella stringa tra apici che fa da parametro alla parola chiave, alcuni caratteri hanno una funzione specifica.

- Il blank è sempre un carattere sostituibile.
- Il primo zero lungo la parola definisce la posizione di riferimento per la soppressione degli zeri non significativi. Gli zeri non significativi in questa posizione ed in quelle a sinistra sono sostituiti da blank nella stringa finale. È un carattere sostituibile.
- Il primo asterisco lungo la parola, purché non preceduto da uno zero, definisce la posizione di riferimento per la soppressione degli zeri non significativi. Gli zeri non significativi in questa posizione ed in quelle a sinistra sono sostituiti da asterischi nella stringa finale. È un carattere sostituibile.
- Il numero totale dei caratteri sostituibili deve essere uguale al numero delle cifre del campo.
- Se lo zero o l'asterisco sono in prima posizione e non si vuole la soppressione del primo zero non significativo, si scrive lo zero o l'asterisco in una posizione aggiuntiva all'estrema sinistra. Cioè, nel caso particolare che si vogliano emettere tutti gli zeri non significativi, non basta scrivere lo zero o l'asterisco di riferimento all'estrema sinistra nella parola ma bisogna allargare la parola a sinistra di una posizione e scriverlo lì. Il carattere di servizio impone così la conservazione di tutti gli zeri non significativi, non viene emesso, l'ingombro dell'emissione non aumenta e la posizione aggiuntiva della parola non entra nel conto dei caratteri sostituibili.
- La e-commerciale (&=ampersand) viene sostituita da un blank nella stringa finale. Se però il campo è di input/output i blank intermedi vengono trasformati in zeri, con pregiudizio del contenuto del campo riletto.
- Il primo simbolo di divisa (il carattere presente nel valore di sistema QCURSYM) provoca due fenomeni in alternativa.
    - Se è il primo carattere di sinistra, viene sempre stampato (divisa fissa).
    - Se è immediatamente a sinistra dello zero o dell'asterisco che definiscono la soppressione degli zeri, si inserisce nella stringa finale immediatamente prima della prima cifra significativa (divisa mobile).
- Punto o virgola viene inserito nella stringa finale nella stessa posizione che occupa nella parola, purché a destra della prima cifra significativa.
- Il segno meno (o l'acronimo CR) subito dopo l'ultima cifra del campo viene emesso solo se il campo è negativo.

Parola di edizione	Valore	Stringa di emissione
'....+....1....+....2....+	....+....1.	....+....1....+....2....+
' . . 0, &CR*'	0000000005-	,05 CR*
' . . $0 , CR*'	0000000005+	$0,05 *
' . . $0, CR**'	0034567890-	$345.678,90CR**
'$ . . 0, '	0000000000	$ .00
'$& . .0 . &-&GROSS'	1234567890-	$ 12.345.678,90 - GROSS
' . . * , &-'	0000567890	*****5.678,90
	0000567890	0000567890
	0000567890-	0000567890
' '	0000000000	
' '	0000567890+	567890
' '	0000567890-	567890
' 0'	0000567890-	567890
'0 '	0000567890+	000567890
'0 '	0000567890+	0000567890
'$ &-&NET'	0000567890+	$ 567890 NET
'$ &-&NET'	0000567890-	$ 567890 - NET
'$0 -&NET'	0000567890	$ 000567890 NET
' $0 &CR*'	0000567890-	$567890 CR*
' $0 &CR*'	1234567890-	$1234567890 CR*
' *&CR'	0000000000	**********
' * &CR'	0000000000	********00
'* '	0000567890-	*000567890
' . . , &CR*NET'	0000567890-	5.678,90 CR* NET
' . . , &CR*NET'	0000567890	5.678,90 * NET
' . . $0. '	0000000005	$.05
' . . $0. CR'	0004567890-	$45.678,90CR
' . . *. *CR**'	0000567890+	*****5.678,90* **
' . . DOLLARS CENTS'	0000567890	5.678DOLLARS90CENTS
'0 - - '	023456789	23-45-6789
'& .* . '	0234567	**234.567
' - - &LATER'	091202	9-12-02 LATER
' & & &LATER'	311203	31 12 03 LATER
' / / '	100199	10/01/99

Se il campo numerico ha dei decimali e se la parola non permette almeno uno spazio in più oltre il numero delle cifre, non sarà possibile digitarle tutte: o mancherà la prima cifra intera o l'ultimo decimale.

34.4 Edit word nell' RPG.

Nelle specifiche RPG di emissione di campi numerici su file descritti internamente è possibile personalizzare l'emissione di un campo numerico con regole assolutamente identiche a quelle previste dalle edit word delle DDS.

La sola differenza riscontrabile rispetto a quanto già visto è l'inevitabile eliminazione del primo zero non significativo di un campo, anche specificando lo zero nella prima posizione della maschera. Nelle DDS invece, se si vuole che un'emissione presenti anche il primo zero non significativo, basta inserire lo zro in testa alla maschera allungandola di un carattere sulla sinistra.

34.5 Modifica degli edit code personalizzabili (5-6-7-8-9).

Con gli edit code fissi e con le edit word è impossibile soltanto una cosa che si può fare invece con un edit code descritto a livello di sistema: si tratta del diverso comportamento di editazione per il contenuto zero.

Sul sistema, in tutte le occasioni in cui è possibile utilizzare gli edit code nativi, è possibile anche l'uso degli edit code personalizzati.

Nella libreria di sistema QSYS si trovano infatti 5 oggetti di tipo *EDTD (Edit description): da QEDIT5 a QEDIT9.

I comandi che manipolano questi 5 oggetti sono i seguenti.

DLTEDTD Delete Edit description.

CRTEDTD Create Edit description.

Il sistema contiene una definizione iniziale che non mi è mai servita a nulla e che è quindi possibile trascurare.

Se nello sviluppo di un pacchetto si riscontra l'uso frequente di una edit word, può essere conveniente fissarne il contenuto in un edit code personalizzabile.

L'edit code è sul sistema in versione unica e viene incorporato in modo permanente nel programma o nel file che lo usa: se dopo la compilazione l'edi code cambia, il compilato si comporterà ancora secondo le regole incluse nell'edit code che valeva al tempo della compilazione.

Non ostano difficoltà se, inventato un edit code, lo si usa senza ripensamenti in tutte le attività successive. A parte la necessità di ricrearlo su altri sistemi, resta però il casino che si presenta volendo alternare edit code diversi sullo stesso codice.

Evidentemente è possibile compilare un pacchetto intero nel corso di validità di una certa versione degli edit code, ma è evidente la difficoltà di gestire le compilazioni poi necessarie a causa delle manutenzioni. Prima di ricompilare un oggetto, occorre assicurarsi che gli edit code siano alla giusta versione, curandosi inoltre che l'attività concorrente di altre compilazioni sul sistema non contrasti con l'attività corrente.

Si esemplifica l'edit code X, adatto a gestire …

CRTEDTD X

34.6 DLTEDT e DLTCHK nelle DDS di un Display file.

Nelle DDS di un file fisico è sempre opportuno inserire per ogni campo il Column Heading e, se numerico l'Edit Code o l'Edit Word. Questa pratica permette di avere un edit pronto nell'SQL, nel QUERY e nell'SDA.

Altre keywords permettono di precisare i valori permessi, come Values o Range. Servono da promemoria per lo sviluppo dei controlli nei programmi. Mediante questo espediente anche l'immissione con un Dfu gode di qualche controllo, mentre altre utility (tipo WRKDBF) se ne fregano.

Queste, come le altre keyword, vengono ereditate dal video definito con campi che fanno riferimento ai fisici. Abbastanza spesso però nei programmi è preferibile non affidarsi ai controlli scatenati dalle dds. Si può interrompere il riporto dal fisico di riferimento al video specificando DLTEDT per eliminare gli edit e DLTCHK per eliminare i controlli sui campi.

35 Costanti figurative. (43C)

Si dicono costanti figurative quelle notazioni sintetiche che significano stringhe più ampie o che esprimono con più chiarezza valori standard, ad esempio: *ZERO invece che '000'.

35.1 Valore alto (*HIVAL, High Value).

Il valore della costante figurativa *HIVAL dipende dal contesto in cui viene usata.

Contesto Alfanumerico

Se la costante viene mossa in un campo alfanumerico, il suo valore è costituito da tanti esadecimali FF affiancati quanti sono le posizioni del campo ricevente.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Riempie alfa1 con tutti esadecimali FF.
C                    move      *hival        alfa1            10
     * Forme alternative equivalenti.
C                    movel     *hival        alfa1            10
C                    move(p)   *hival        alfa1            10
C                    movel(p)  *hival        alfa1            10
```

Oppure.

```
D   DName++++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
    D alfa1           s              10
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Riempie alfa1 con tutti esadecimali FF.
C                    eval      alfa1=*hival
     * Forme alternative equivalenti.
C                    evalr     alfa1=*hival
```

Il risultato di ogni singola operazione precedente è il riempimento del campo alfa1 con tutti esadecimali FF. Si osserva che le forme alternative provocano azioni aggiuntive rispetto alla operazione esemplificata per prima, azioni rese tuttavia inefficaci dalla lunghezza automatica della costante *hival.

La costante può essere utilizzata come valore di inizializzazione anche nella definizione di un campo sulla specifica D ottenendo nel campo definito lo stesso valore illustrato per le operazioni di calcolo.

```
D   DName++++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
    D alfa1           s              10    inz(*hival)
```

Anche la comparazione e i condizionamenti utilizzano il valore di tutti esadecimali FF di lunghezza automaticamente adattata al campo di confronto.

Compara

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Se alfa1 contiene tutti esadecimali FF, accende 50,
     * altrimenti accende 49. Non si potrà mai accendere 51
     * perché alfa1 non può essere superiore a *hival.
C      alfa1         comp      *hival                           514950
```

Se

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Se alfa1 contiene tutti esadecimali FF.
C                    if        alfa1=*hival
     *    Esegue i calcoli in questa posizione.
C                    ..........
C                    endif
```

Scegli Quando

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Sceglie.
C                   select
     * Se alfa1 contiene tutti esadecimali FF.
C                   when      alfa1=*hival
     *    Esegue i calcoli in questa posizione.
C                   ..........
     * Se alfa1 contiene tutti esadecimali 00.
C                   when      alfa1=*loval
     *    Esegue i calcoli in questa posizione.
C                   ..........
     * Se alfa1 contiene valori diversi da tutti esadecimali FF
     * e da tutti esadecimali 00
C                   other
     *    Esegue i calcoli in questa posizione.
C                   ..........
     * Sceglie.
C                   endsl
```

Contesto Numerico

Se invece la costante viene mossa in un campo numerico, il suo valore è costituito da tante cifre 9 quante sono le posizioni del campo ricevente e dal segno positivo.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Riempie nume1 con tutte cifre 9 e segno positivo.
C                   z-add     *hival        nume1            11 2
     * Forme alternative equivalenti ma sconsigliabili.
C                   move      *hival        nume1            11 2
C                   movel     *hival        nume1            11 2
C                   move(p)   *hival        nume1            11 2
C                   movel(p)  *hival        nume1            11 2
```

Oppure.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++
     D nume1           s             11  2
CX  CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++
     * Riempie nume1 con tutte cifre 9.
C                   eval      nume1=*hival
```

Il risultato di ogni singola operazione precedente è il riempimento del campo nume1 con tutte cifre 9 e segno positivo.

La costante può essere utilizzata anche nella definizione di un campo sulla specifica D.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++
     D nume1           s             11  2 inz(*hival)
```

Anche la comparazione e i condizionamenti utilizzano il valore di tutte cifre 9 con segno positivo di lunghezza automaticamente adattata al campo di confronto.

Compara

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Se nume1 contiene tutte cifre 9 e segno positivo, accende 50,
     * altrimenti accende 49. Non si potrà mai accendere 51
     * perché nume1 non può essere superiore a *hival.
C     nume1         comp      *hival                             514950
```

Se

```
CX  CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++
     * Se nume1 contiene tutte cifre 9 con segno positivo.
C                   if        nume1=*hival
     *    Esegue i calcoli in questa posizione.
C                   ..........
C                   endif
```

Scegli Quando

```
CX  CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++++
     * Sceglie.
C                    select
     * Se nume1 contiene tutte cifre 9 con segno positivo.
C                    when      nume1=*hival
     *    Esegue i calcoli in questa posizione.
C                    ..........
     * Se nume1 contiene tutte cifre 9 con segno negativo.
C                    when      nume1=*loval
     *    Esegue i calcoli in questa posizione.
C                    ..........
C                    endsl
     * Se alfa1 contiene valori diversi da tutte cifre 9 con segno positivo,
     * e da tutte cifre 9 con segno negativo.
C                    other
     *    Esegue i calcoli in questa posizione.
C                    ..........
     * Sceglie.
C                    endsl
```

35.2 Valore basso (*LOVAL, Low Value).

*Mutatis mutandis, si ripete la spiegazione valevole per *HIVAL.*

Il valore della costante figurativa *LOVAL dipende dal contesto in cui viene usata.

Contesto Alfanumerico

Se la costante viene mossa in un campo alfanumerico, il suo valore è costituito da tanti esadecimali 00 affiancati quanti sono le posizioni del campo ricevente.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Riempie alfa1 con tutti esadecimali 00.
C                    move      *loval        alfa1            10
     * Forme alternative equivalenti.
C                    movel     *loval        alfa1            10
C                    move(p)   *loval        alfa1            10
C                    movel(p)  *loval        alfa1            10
```

Oppure.

```
D   DName++++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
    D alfa1           s              10
CX  CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++++
     * Riempie alfa1 con tutti esadecimali 00.
C                    eval      alfa1=*loval
     * Forme alternative equivalenti.
C                    evalr     alfa1=*loval
```

Il risultato di ogni singola operazione precedente è il riempimento del campo alfa1 con tutti esadecimali 00. Si osserva che le forme alternative provocano azioni aggiuntive rispetto alla operazione esemplificata per prima, azioni rese tuttavia inefficaci dalla lunghezza automatica della costante *loval.

La costante può essere utilizzata come valore di inizializzazione anche nella definizione di un campo sulla specifica D ottenendo nel campo definito lo stesso valore illustrato per le operazioni di calcolo.

```
D   DName++++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
    D alfa1           s              10    inz(*loval)
```

Anche la comparazione e i condizionamenti utilizzano il valore di tutti esadecimali 00 di lunghezza automaticamente adattata al campo di confronto.

Compara

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Se alfa1 contiene tutti esadecimali 00, accende 50,
     * altrimenti accende 51. Non si potrà mai accendere 49
     * perché alfa1 non può essere inferiore a *loval.
C     alfa1         comp      *loval                             514950
```

Se

```
CX  CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++++
     * Se alfa1 contiene tutti esadecimali 00.
C                   if        alfa1=*loval
```

Scegli Quando

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Vedi l'esempio per *HIVAL.
```

Contesto Numerico

Se invece la costante viene mossa in un campo numerico, il suo valore è costituito da tante cifre 9 quante sono le posizioni del campo ricevente e dal segno negativo.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Riempie nume1 con tutte cifre 9 e segno negativo.
C                   z-add     *loval         nume1             11 2
     * Forme alternative equivalenti ma sconsigliabili.
C                   move      *loval         nume1             11 2
C                   movel     *loval         nume1             11 2
C                   move(p)   *loval         nume1             11 2
C                   movel(p)  *loval         nume1             11 2
```

Oppure.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
    D nume1           s              11  2
CX  CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++++
     * Riempie nume1 con tutte cifre 9 e segno negativo.
C                   eval      nume1=*loval
```

Il risultato di ogni singola operazione precedente è il riempimento del campo nume1 con tutte cifre 9 e segno negativo.

La costante può essere utilizzata anche nella definizione di un campo sulla specifica D.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
    D nume1           s              11  2 inz(*loval)
```

Anche la comparazione e i condizionamenti utilizzano il valore di tutte cifre 9 con segno negativo di lunghezza automaticamente adattata al campo di confronto.

Compara

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Se nume1 contiene tutte cifre 9 e segno negativo, accende 50,
     * altrimenti accende 51. Non si potrà mai accendere 49
     * perché nume1 non può essere inferiore a *loval.
C     nume1         comp      *loval                             514950
```

Se

```
CX  CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++++
     * Se nume1 contiene tutte cifre 9 con segno negativo.
C                   if        nume1=*loval
```

Scegli Quando

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Vedi l'esempio per *HIVAL.
```

35.3 Stringa di caratteri (*ALL'cc', All Characters).

*La spiegazione è simile a quanto già esposto per *HIVAL e per *LOVAL.*

In luogo di cc si possono indicare da 1 a ? caratteri. Ad esempio: *ALL'ABC' ; *ALL'<>--' .

Contesto Alfanumerico

*In luogo di 'cc' si possono anche indicare da 1 a ? caratteri esadecimali (ad esempio: *ALLX'1C' ; *ALLX'22C120') ma, per ulteriori delucidazioni, consultare poco oltre il paragrafo sulle costanti esadecimali.*

La definizione permette di costruire stringhe composte da uno o più caratteri che si ripetono fino al riempimento del campo ricevente.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Riempie alfa1 con la stringa ABC ripetuta fino al riempimento,
     * ottenendo in alfa1 il valore 'ABCABCABCA'.
C                    move      *all'ABC'     alfa1            10
```

Oppure.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
D alfa1           s             10
CX  CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++++
     * Riempie alfa1 con la stringa ABC ripetuta fino al riempimento,
     * ottenendo in alfa1 il valore 'ABCABCABCA'.
C                    eval      alfa1=*all'ABC'
```

La costante può essere utilizzata come valore di inizializzazione anche nella definizione di un campo sulla specifica D ottenendo nel campo definito lo stesso valore illustrato per le operazioni di calcolo.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
D alfa1           s             10    inz(*all'ABC')
```

In particolare, capita spesso di usare una riga di trattini in stampa.

```
D   DName++++++++++ETDsFrom+++To/L+++IDc.Par.chi.++++++++++++++++++++++++++++++
D trat1           s            132    inz(*all'-')
```

Anche la comparazione e i condizionamenti utilizzano il valore esemplificato di lunghezza automaticamente adattata al campo di confronto.

Compara

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Se alfa1 contiene 'ABCABCABCA', accende 50;
     * se contiene un valore superiore, accende 51;
     * se contiene un valore inferiore, accende 49.
C     alfa1         comp      *all'ABC'                          514950
```

Se

```
CX  CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++++
     * Se alfa1 contiene 'ABCABCABCA'.
C                    if        alfa1=*all'ABC'
     *    Esegue i calcoli in questa posizione.
C                    ..........
C                    endif
```

Scegli Quando

```
CX  CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++
     * Sceglie.
C                    select
     * Se alfa1 contiene 'ABCABCABCA'.
C                    when      alfa1=*all'ABC'
     *    Esegue i calcoli in questa posizione.
C                    ..........
     * Se alfa1 contiene 'WWWWWWWWWW'.
C                    when      alfa1=*all'W'
     *    Esegue i calcoli in questa posizione.
C                    ..........
     * Se alfa1 contiene valori diversi da 'ABCABCABCA' e da 'WWWWWWWWWW'.
C                    other
     *    Esegue i calcoli in questa posizione.
C                    ..........
     * Sceglie.
C                    endsl
```

Contesto Numerico

La definizione permette di costruire numeri composti da una cifra che si ripete fino al riempimento del campo ricevente. Non si usa, di solito, più di una cifra e, perciò, si tralasciano le esemplificazioni per la loro semplicità.

35.4 Spazi (*BLANK, Blank).

Valgono anche qui, e per tutte le costanti successive, considerazioni analoghe a quelle esposte per le costanti precedenti. Si illustrano perciò soltanto il valore specifico delle costanti ed eventuali usi particolari.

*BLANK (equivalente al prolisso *BLANKS), sta per una serie di caratteri blank (ovvero: spazi) lunga quanto basta per riempire il campo ricevente o per confrontarsi con un campo.

In particolare *BLANK si usa per riportare al valore di default le variabili alfanumeriche. Per valore di default si intende il valore a cui il programma inizializza le variabili.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Pulisce il campo alfa1.
C                    movel     *blank         alfa1             10
```

In questa funzione è tuttavia preferibile l'istruzione seguente, più moderna e sintetica, che risulta utilizzabile per scopi analoghi anche sui campi numerici.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Pulisce il campo alfa1.
C                    clear                    alfa1             10
```

35.5 Zeri (*ZERO, Zero).

*ZERO (equivalente al prolisso *ZEROS), sta per una serie di caratteri '0' (ovvero: zeri) lunga quanto basta per riempire il campo ricevente o per confrontarsi con un campo.

Si nota che *ZERO è utilizzabile sia nel contesto alfanumerico che in quello numerico.

In particolare *ZERO si usa per riportare al valore di default le variabili numeriche. Per valore di default si intende il valore a cui il programma inizializza le variabili.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Pulisce il campo nume1.
C                    z-add     *zero          nume1             11 2
```

In questa funzione è tuttavia preferibile l'istruzione seguente, più moderna e sintetica.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Pulisce il campo nume1.
C                    clear                    nume1             11 2
```

35.6 Valori esadecimali (X'cc').

Sul video AS/400 non si possono esprimere i caratteri con un corrispondente esadecimale inferiore a x'40' (ovvero: blank), né per tutti gli esadecimali del campo visibile esiste un tasto sulle tastiere.

La notazione esadecimale è il mezzo per esprimere tali caratteri.

La costante x'7C' corrisponde, ad esempio, al carattere § (paragrafo).

Altri esempi: x'C1' = A
x'F2' = 2
x'1C' = asterisco sopralineato
x'22' = carattere invisibile che provoca alta luminosità nei caratteri successivi sul video

È possibile indicare anche più esadecimali contemporaneamente.
x'818283C3F2' = abcC2

Si esemplifica la composizione di una dicitura contenente un carattere nei dati che provoca il reverse image lungo il testo.

```
CX  CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++++
     * Compone la dicitura "Ciao mamma mia".
C                    eval      txt='Ciao'+x'24'+'mamma'+x'20'+'mia'
```

Se un campo alfanumerico a video viene corredato sulle DDS della parola chiave DUP, il campo stesso diventa abile a ricevere la digitazione del tasto DUP (sui pc: Maiuscolo, Inserimento) che riempie il campo di asterischi sopralineati. Se non si vuole usare l'indicatore di risposta del DUP per ragioni di economia indicatori, è lo stesso possibile riconoscere la presenza degli asterischi sopralineati come segue.

```
CX  CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++++
     * Se è stata richiesta la duplicazione dei dati.
C                    if        alfavideo=*allx'1C'
```

Per completezza si nota che per digitare su terminale un carattere uguale o superiore a x'40' è possibile battere in sequenza; Esadecimale, Digit, Zone. Ad esempio, digitando "Esadecimale, 7, C", si ottiene il carattere § (paragrafo).

35.7 Valore logico vero (*ON, Logical On).

*ON sta per una serie di caratteri '1' lunga quanto basta per riempire il campo ricevente o per confrontarsi con un campo.

In particolare si usa confrontarlo con gli indicatori espressi sotto forma di campo.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Se l'indicatore 50 è acceso...
C     *in50         ifeq      *on
```

Serve anche per accendere un indicatore.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Accende l'indicatore 50.
C                    movel     *on           *in50
```

Se ne sconsiglia l'uso per confronti con campi che non siano dei flag logici di lunghezza 1. Per flag logico si intende un campo alfanumerico di lunghezza 1 destinato a contenere solo i valori '0' e '1' oppure, che è lo stesso, i valori *ON e *OFF. Frequentemente invece del valore *OFF si usa il valore *BLANK, più comodo perché è il valore di inizializzazione dei campi alfanumerici non altrimenti governati.

```
CX  CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++++
     * Se non ancora eseguito.
C                    if        eseguito<>*on
     *   Annota eseguito.
C                    movel     *on            eseguito          1
     *   Esegue.
C                    exsr      esecuzione
     * Se non ancora eseguito.
C                    endif
```

35.8 Valore logico falso (*OFF, Logical Off).

*OFF sta per una serie di caratteri '0' lunga quanto basta per riempire il campo ricevente o per confrontarsi con un campo.

In particolare si usa confrontarlo con gli indicatori espressi sotto forma di campo.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Se l'indicatore 50 è spento...
C     *in50          ifeq      *off
```

Serve anche per spegnere un indicatore.

```
C   CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....
     * Spegne l'indicatore 50.
C                    movel     *off           *in50
```

Se ne sconsiglia l'uso per confronti con campi che non siano dei flag logici di lunghezza 1.

36 Screen Design Aid (SDA). (44R)

L'SDA è lo strumento mediante il quale si definiscono i file video. Le DDS (Data Description Specification) per il membro sorgente vengono generate in automatico dall'SDA. E' analogo all'RLU (Report Layout Utility), che viene utilizzato per disegnare le stampe, soltanto che l'SDA definisce i file video. Ci sentiamo di affermare che l'SDA è ad oggi lo strumento da utilizzare per definire, creare e manutenere i file video.

36.1 Uso per prima generazione di un file video.

Il comando per avviare una sessione SDA è STRSDA, che corrisponde alla opzione 17 della gestione dei membri del PDM. Di seguito viene mostrato l'avvio di una sessione SDA dalla lista di gestione membri con il PDM.

```
Immettere le opzioni e premere Invio.
  16=Esecuz. procedura   17=Modifica tramite SDA   19=Modifica tramite RLU
  25=Ricerca stringa  54=Confronto              55=Integrazione ...

Opz  Membro      Tipo        Testo
17   VIDEO00V    DSPF        Corso di programmazione applicativa su AS/400.
```

Questo è il video SDA di gestione record video.

```
Immettere le opzioni e premere Invio.
  1=Aggiunta        2=Editare commenti       3=Copia         4=Cancellazione
  7=Ridenominazione 8=Scelta parole chiave  12=Preparazione immagine

Opz  Ordine   Record        Tipo      Subfile correlato Data        Errore DDS
1             W01

  (Nessun record nel file)
```

Immettere l'opzione 1=Aggiunta, il nome del record quindi premere Invio. Specificare il tipo di record che si sta andando ad aggiungere, nel nostro caso Tipo = RECORD, quindi premere Invio. L'SDA presenta un intero video senza alcuna costante o variabile precostituita. Al programmatore il disegno dell'intero video.

36.2 Definizione di una costante.

Una costante si definisce tra due apici (iniziale e finale). Se si è specificata l'intestazione del video, per esempio, per centrarla in automatico è possibile specificare AC sul byte precedente al primo carattere della costante e premere Invio; in questo modo la costante verrà centrata automaticamente dall'SDA.

36.3 Definizione di una variabile.

Vi sono due modi per definire una variabile. Il primo consiste nello specificare e definire una variabile senza alcun riferimento ad un campo di database, il secondo consiste nell'attingere direttamente a tutte le caratteristiche da un campo di database. A differenza della costante, per una variabile si devono necessariamente specificare i seguenti attributi; il formato dati, il tipo di utilizzo, le dimensioni ed il nome.

36.4 Definizione di una variabile senza riferimento al database.

<u>Variabile alfanumerica.</u>

Si deve specificare il carattere +, che sta ad indicare l'inizio della definizione di una variabile, quindi il tipo di campo (input, output, input/output) e le dimensioni.

+I(10) Variabile di Input lunga 10 caratteri
+O(10) Variabile di Output lunga 10 caratteri
+B(10) Variabile di Input/Output lunga 10 caratteri

Per attribuire un nome alla variabile si deve specificare il carattere ? sul byte precedente al primo carattere della variabile e premere Invio. L'SDA mostra il nome attribuito in automatico e la lunghezza della variabile; da qui è possibile attribuire il nuovo nome ed eventualmente una nuova dimensione.

```
FLD001     Lungh.: 00010 TEXT:
```

Variabile numerica.

Si deve specificare il carattere +, che sta ad indicare l'inizio della definizione di una variabile, quindi il tipo di campo (input, output, input/output) e le dimensioni.

+3(5,2) Variabile di Input lunga 5 di cui 2 decimali
+6(5,2) Variabile di Output lunga 5 di cui 2 decimali
+9(5,2) Variabile di Input/Output lunga 5 di cui 2 decimali

36.5 Definizione di una variabile con riferimento al database.

Premere F10 per ottenere il video di scelta file di database.

```
                          Scelta dei file database
Immettere le opzioni e i nomi e premere Invio.
  1=Visualizzazione dell'elenco dei campi database
  2=Scelta di tutti i campi per l'immissione (I)
  3=Scelta di tutti i campi per l'emissione (O)
  4=Scelta di tutti i campi sia per l'immissione che per l'emissione (B)

Opzione   File database   Libreria     Record
          _________      _________    _________
```

Se specificata l'opzione 1 ed il file di database, viene presentato l'elenco dei campi da cui effettuare la selezione. Nell'esempio che viene di seguito mostrato si è specificato il file ANCL200F in *LIBL nome record ANCL2.

```
Immettere le opzioni e premere Invio.
   1=Visualizzazione descrizione estesa del campo
   2=Scelta per immis. (I), 3=Scelta per emiss. (O), 4=Scelta per imm/emiss (B)

 Opzione Campo        Lungh.   Tipo    Intestazione di colonna
         ATA10             1    A      A n n
         NRA10           9,0    P      Azione
         DAA10           8,0    P      D.ult. manut.
         CDCLI             6    A      Client
         RASCL            35    A      Ragione sociale
         RASC2            35    A      Aggiunta rag.sociale
         CDRIC            15    A      CDRIC
         INDCL            35    A      Indirizzo
                                                                  Segue...
```

Se specificate le altre opzioni, si avranno a disposizione tutti i campi di un file di database.

I campi di database selezionati sono ora a disposizione per essere utilizzati sul video SDA.

Per utilizzare e collocare i soli campi di database sul video SDA si deve specificare il carattere & seguito dal numero relativo al campo; &1 indica di collocare il campo di database identificato come numero 1 sul video SDA.

Per utilizzare e collocare le sole intestazioni dei campi di database sul video SDA si deve specificare il carattere &, seguito dal numero relativo al campo, seguito dal carattere P.

Per utilizzare e collocare sia i campi di database che le intestazioni sul video SDA si deve specificare il carattere &, seguito dal numero relativo al campo, seguito dal carattere L.

Questo è quello che appare se specifichiamo il carattere ? su di un campo di database.

```
RASCL      Lungh.: 00035 COLHDG: Ragione sociale
```

Si rammenta che è sempre auspicabile non utilizzare sul file video i nomi di campo coincidenti con quelli di database.

36.6 Scelta parole chiave campo.

Per attribuire le parole chiave ad una variabile si deve specificare il carattere * sul byte precedente al primo carattere della variabile e premere Invio. L'SDA mostra quanto segue:

```
                          Scelta parole chiave campo

Campo . . . . . :   W1CDCLI             Uso . . . :  B
Lunghezza . . . :   6                   Riga. . . :  5    Colonna. . . :  19

Immettere le scelte e premere Invio.
                                    Y=Sì    Per tipo campo
  Attributi visualizzazione . . . .         Tutti eccetto non visualizzati
  Colori  . . . . . .               Tutti eccetto non visualizzati
  Opzioni di immissione . . . . . .         Nascosto, immissione o entrambi
  Controllo validità  . . . . . .           Imm. o Imm./Em., non virgola mob.
  Parole chiave immissione. . . . .         Immissione o Immiss./Emiss.
  Parole chiave comuni. . . . . .           Tutti i tipi

  Riferimento database. . . . . .           Non vis., Immiss., Emiss., Imm./Em.
  Messaggi di errore  . . . . . .           Immiss., Emiss., Immiss./Emiss.
  ID messaggio (MSGID)  . . . . . .         Emissione o Immissione/Emissione


  Parola chiave TEXT. . . . . .   Codice cliente
```

Da qui si possono specificare tutte le parole chiave per un campo video, nel nostro esempio, per il campo W1CDCLI.

Per attribuire le parole chiave ad una costante si deve specificare il carattere * sul byte precedente al primo carattere della costante e premere Invio. L'SDA mostra quanto segue:

```
                          Scelta parole chiave campo

Costante  . . . :   Intestazione Video
Lunghezza . . . :   18                  Riga. . . :  1    Colonna. . . :  32

Immettere le scelte e premere Invio.
                                    Y=Sì    Per tipo campo
  Attributi visualizzazione . . . .         Tutti eccetto non visualizzati
  Colori  . . . . . .               Tutti eccetto non visualizzati

  Parole chiave comuni. . . . . .           Tutti i tipi

  Parola chiave TEXT. . . . . .
```

Si noti che alcune parole chiave si possono specificare e quindi attribuirle ai campi ed alle costanti senza dovere richiamare il menù sopra visualizzato, ad esempio l'alta intensità o sottolineatura del campo o costante si attribuisce specificando una H od una U sul byte precedente al primo carattere della variabile o costante e premere Invio.

36.7 Spostamento, copia e cancellazione dei campi e delle costanti.

Per meglio comprendere quanto segue definiamo "Posizione di attributo del campo" il byte precedente al primo carattere della variabile o costante.

Trasferimento di un campo -, =

> Immettere – nella posizione di attributo del campo ed immettere = dove si desidera che il campo venga visualizzato.

Copia di un campo -, ==

> Immettere – nella posizione di attributo del campo ed immettere == dove si desidera venga visualizzata una copia del campo.

Cancellazione di un campo

> Immettere D nella posizione di attributo del campo oppure cancellare la posizione dell'intero campo cominciando dalla posizione di attributo.

Trasferimento di un blocco di campi -, -, =

Immettere – nell'angolo in alto a sinistra del blocco di campi, immettere – nell'angolo in basso a destra del blocco di campi in modo che – sia un carattere che si trova al di là del campo più lungo del blocco. Immettere = nel punto in cui si desidera che il gruppo di campi venga spostato.

Copia di un blocco di campi -, -, ==

Immettere – nell'angolo in alto a sinistra del blocco di campi, immettere – nell'angolo in basso a destra del blocco di campi in modo che – sia un carattere che si trova al di là del campo più lungo del blocco. Immettere == nel punto in cui si desidera avere una copia del gruppo di campi.

Eliminazione di un blocco di campi --, --

Immettere -- nell'angolo in alto a sinistra del blocco di campi, immettere -- nell'angolo in basso a destra del blocco di campi in modo che – sia un carattere che si trova al di là del campo più lungo del blocco. Quando si preme Invio, il pannello di lavoro viene rivisualizzato con segni indicanti i limiti di blocco che si sta cancellando. E' possibile premere Invio per cancellare il blocco o F12 per annullare la cancellazione.

Spostamento di un campo <<<, >>>

Immettere <<< nella posizione di attributo del campo od immettere >>> nella dopo il campo per spostare a sinistra o a destra il campo sino al limite indicato dai segni.

36.8 Costanti speciali di sistema.

Vi sono delle costanti speciali precostituite che vengono messe a disposizione del programmatore che possono essere utilizzate nell'SDA. Queste costanti si definiscono specificando le opportune parole chiave.

Parole chiave Data e Ora

Immettere *DATE o *TIME per definire costanti speciali sullo schermo su cui verranno automaticamente sostituite con la data e ora di sistema correnti.

Parole chiave Utente e Sistema

Immettere *USER o *SYSNAME per definire costanti speciali sullo schermo in cui verranno automaticamente sostituiti i nomi utente e di sistema.

36.9 Tasti di funzione.

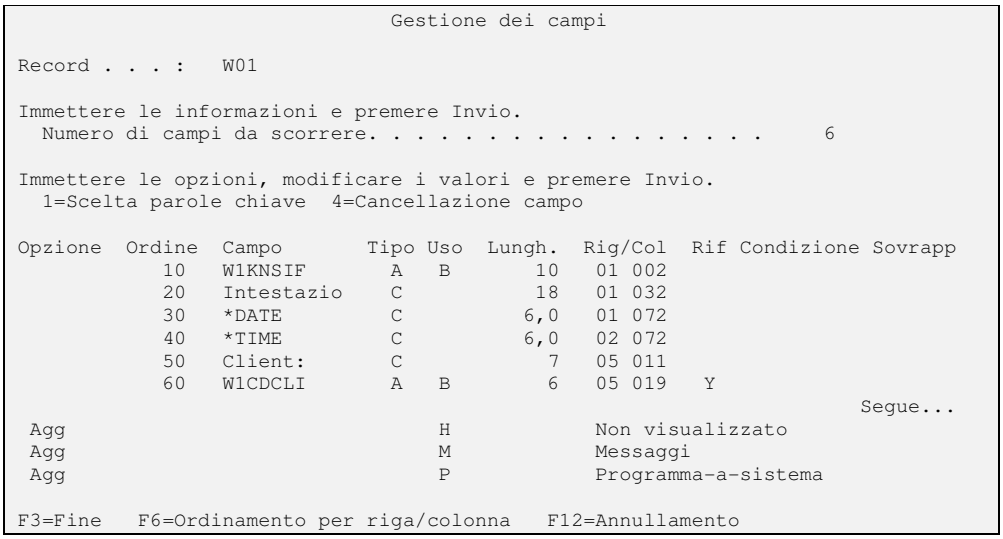
Di seguito vengono riportati i principali tasti di funzione disponibili sul video SDA. Si tenga conto che alcuni tasti di funzione sono stati volutamente omessi e che altri si possono utilizzare solo per determinate tipologie di record (subfile).

F3=Fine.

Premere F3=Fine per chiudere la sessione di Preparazione immagine schermo di lavoro. Appare lo schermo Fine preparazione immagine schermo di lavoro.

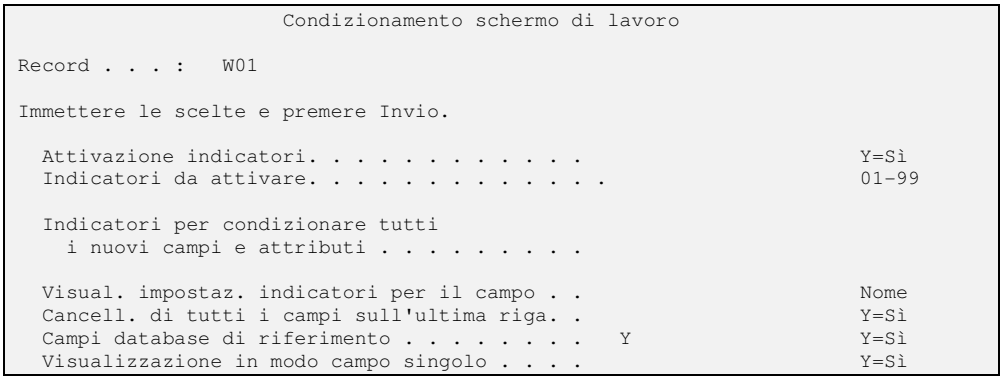
F4=Richiesta.

Premendo F4 comparirà lo schermo Gestione dei campi. Viene presentato l'elenco di tutti i campi e di tutte le costanti per il record corrente inclusi eventuali campi nascosti.



F6=Condizionamento schermo di lavoro.

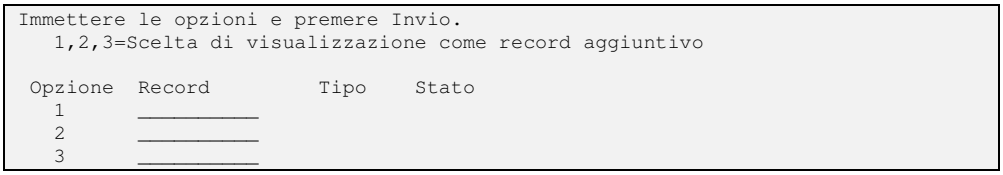
Premendo F6 comparirà lo schermo di Condizionamento schermo di lavoro dal quale si possono attivare o disattivare gli eventuali indicatori di condizionamento campi video.



F9=Scelta record aggiuntivi.

Premere F9 per scegliere altri record da visualizzare sullo schermo Preparazione immagine schermo di lavoro. Tipico utilizzo di questa opportunità è quello con i video subfile, oppure per visualizzare una Window che si deve sovrapporre ad un record per deciderne l'opportuno posizionamento.

Ovviamente devono esistere più record per il file video corrente.



F10=Data base.

Premere F10 per fare apparire lo schermo Scelta file di database. Il suo utilizzo lo abbiamo già visto proprio in questo capitolo.

F12=Annullamento.

Premere F12 per salvare il lavoro e tornare sullo schermo Gestione record video, tenendo conto che non è stato elaborato nulla di ciò che è stato immesso dall'ultima volta in cui è stato premuto Invio.

F15=Richiesta subfile.

Premere F15 per visualizzare una richiesta per mezzo della quale è possibile modificare le dimensioni di riga e pagina del subfile (rispettivamente SFLLIN e SFLPAG) attualmente valide per il record in corso di definizione.

```
 Par. chiave subfile: SFLPAG:     8         SFLLIN:     0
```

F17=Stampa.

Premere F17 per stampare il contenuto corrente dello schermo Preparazione immagine schermo.

F20=Inversione di fondo delle costanti.

Premere F20 per ottenere l'inversione di fondo delle costanti. L'inversione di fondo è temporanea e non influisce sulla immagine compilata. E' utilizzata per stabilire con esattezza l'inizio e la fine delle costanti (es. prima di spostarle).

F22=Comando di sistema.

Premere F22 affinché la finestra in cui si possono immettere dei comandi di sistema venga aperta.

36.10 Uscita e salvataggio sessione SDA.

Premendo F3=Fine, l'SDA presenta lo schermo di fine lavoro SDA dal quale si possono utilizzare le seguenti opzioni di uscita dalla sessione di lavoro.

```
                          Fine schermo di lavoro SDA

 Scegliere una delle seguenti opzioni:

     1. Salvataggio lavoro eseguito dall'ultimo Invio e fine schermo di lavoro
     2. Uscita senza salvataggio del lavoro eseguito sullo schermo di lavoro
     3. Ripresa della sessione schermo di lavoro



 Scelta
     1

 F12=Annullamento
```

Se non scelta l'opzione 3, ci si ritroverà sul video SDA di gestione record video.

```
Immettere le opzioni e premere Invio.
   1=Aggiunta        2=Editare commenti       3=Copia         4=Cancellazione
   7=Ridenominazione 8=Scelta parole chiave  12=Preparazione immagine

Opz  Ordine   Record        Tipo      Subfile correlato Data        Errore DDS

        10    W01           RECORD                      gg/mm/aa
```

36.11 Specifica parole chiave record.

Così come si è dovuto specificare alcune parole chiave per i singoli campi o costanti, si dovrà procedere alla specifica delle parole chiave a livello di record. Dal video SDA di gestione video, immettere l'opzione 8 e premere Invio.

```
                          Scelta parole chiave record

Record . . . :   W01

Immettere le scelte e premere Invio.

                                      Y=Sì
  Parole chiave comuni. . . . . .
  Parole chiave indicatori. . . . .
  Aiuto applicazione. . . . . .
  Parole chiave di aiuto  . . . . .
  Parole chiave emissione . . . . .
  Parole chiave immissione. . . . .
  Parole chiave sovrapposizione . .

  Parole chiave di stampa . . . . .
  Parola chiave ALTNAME . . . . . .

  Parola chiave TEXT. . . . . .
```

Da qui si possono specificare tutte le parole chiave per un record video, ad esempio i tasti di funzione utilizzabili se ammessa la stampa dello schermo etc., nel nostro caso per il record W01.

36.12 Specifica parole chiave file video.

Così come si è dovuto specificare alcune parole chiave per il record, si dovrà procedere alla specifica delle parole chiave a livello di file. Dal video SDA di gestione video, premere F14=Parole chiave livello-file.

```
                          Scelta parole chiave file

Membro . . . :   VIDEO00V

Immettere le scelte e premere Invio.

                                      Y=Sì
  Parole chiave comuni. . . . . .
  Parole chiave indicatori. . . . .
  Parole chiave di stampa . . . . .
  Parole chiave di aiuto  . . . . .
  Ampiezze schermo  . . . . . .
  Parole chiave alternative . . . .
  Conversione DBCS  . . . . . .
  Bordatura finestra. . . . . .
  Parole chiave barra menu. . . . .
```

Da qui si possono specificare tutte le parole chiave per un file video, ad esempio se gli indicatori risiedono in una area di memoria specifica, le dimensioni del formato video, eventuali tasti di funzione utilizzabili etc., nel nostro caso per il file VIDEO00F.

36.13 Creazione di un file video.

Una volta disegnato il formato video e specificate le opportune parole chiave a livello di singolo campo, record e file, dal video di Gestione record video si può premere F3=Fine. Viene presentato il video di Salvataggio DDS – Creazione file video.

```
                    Salvataggio DDS - Creazione file video

 Immettere le scelte e premere Invio.

   Salvataggio origine DDS  . . . . . .   Y            Y=Sì
     File origine . . . . . . . . . . .   QDSPFSRC     F4 per elenco
       Libreria . . . . . . . . . . . .   NOMELIB      Nome, *LIBL ...
     Membro . . . . . . . . . . . . . .   VIDEO00V     F4 per elenco
     Testo. . . . . . . . . . . . . . .   Corso di programmazione applicativ
a su AS/400.

   Creazione file video . . . . . . . .   Y            Y=Sì
     Richiesta per parametri. . . . . .                Y=Sì
     File video . . . . . . . . . . . .   VIDEO00V     F4 per elenco
       Libreria . . . . . . . . . . . .   NOMELIB      Nome, *CURLIB
     Sostituzione file esistente. . . . .              Y=Sì

   Sottomissione creazione lavoro in batch.   Y        Y=Sì
   Specifica di opzioni aggiuntive di
     salvataggio o creazione  . . . . . .              Y=Sì

 F3=Fine   F4=Richiesta F12=Annullamento
 Il membro VIDEO00V esiste già. Per la sostituzione, premere Invio.
```

I default che vengono presentati sono già tutti corretti e non vanno modificati, ad eccezione del parametro:

```
Creazione file video . . . . . .   Y            Y=Sì
```

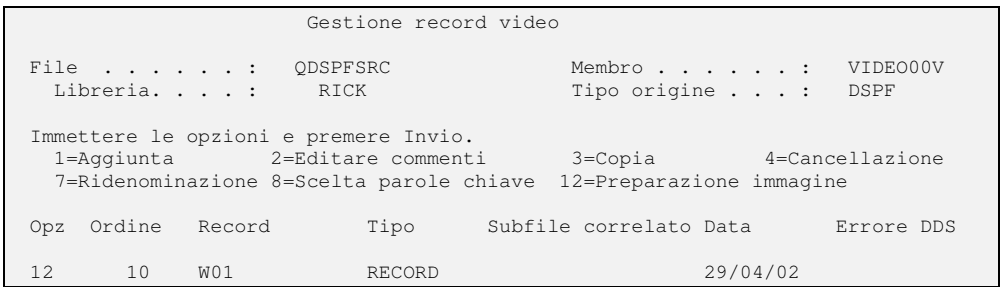
Per il quale suggeriamo di specificare sempre N = No ed evitare di creare il file video da questo punto. Le ragioni sono molteplici: ad esempio si può avere la necessità di andare in editazione con il SEU del membro sorgente per aggiungere specifiche od altro, si può avere la necessità di specificare dei parametri di compilazione diversi da quelli di default proposti dal sistema. In questo caso, poiché non vi è traccia degli eventuali parametri di compilazione imputati dall'utente se non ricavandoli dall'osservazione dell'oggetto, si rammenta e raccomanda l'utilizzo di stringhe CL di compilazione, in modo tale che dei parametri di creazione vi sia una traccia facilmente reperibile e riutilizzabile. Od ancora, si ha la necessità di effettuare la creazione del file video utilizzando un'opportuna Job Description etc. Per questi motivi suggeriamo di procedere alla creazione del file video o utilizzando l'opzione 14=Compilaz. Del PDM o utilizzando un'opportuna stringa CL di compilazione.

36.14 Manutenzioni successive di un file video.

L'SDA si utilizza anche per effettuare le manutenzioni successive per un file video. Dalla lista membri PDM si accede sempre utilizzando l'opzione 17=Modifica tramite SDA.

```
 Immettere le opzioni e premere Invio.
  2=Editazione   3=Copia  4=Cancel.   5=Visualiz.   6=Stampa    7=Ridenom.
  8=Visual. descriz.  9=Salvat. 13=Modif. testo 14=Compilaz. 15=Creaz. modulo...

 Opz  Membro      Tipo        Testo
 17   VIDEO00V    DSPF        Corso di programmazione applicativa su AS/400.
```

Dal video Gestione record video immettere l'opzione 12=Preparazine immagine sul record che si intende modificare; ci si ritrova sullo schermo di preparazione immagine schermo di lavoro dove è possibile effettuare ed utilizzare tutte le risorse messe a disposizione dall'SDA come già ampiamente visto e descritto nel paragrafo precedente.



Si raccomanda sempre, prima di iniziare le modifiche e/o implementazioni, di effettuare opportuna copia di salvataggio del membro sorgente qualunque tipologia esso sia; questo per garantirsi la possibilità di ripristinare il sorgente ed eventualmente generare nuovamente l'oggetto modificato.

37 Report Layout Utility (RLU). (45R)

L'RLU (Report Layout Utility) viene utilizzato per disegnare le stampe.

Da riscrivere integralmente. Questo è l'aiuto in linea.

Analogamente al programma di utilità per disegnare i file video SDA (Screen Design Aid), l'RLU (Report Layout Utility) serve per disegnare i file di stampa descritti esternamente. Dall'elenco dei membri sorgente di tipo PRTF con il PDM si accede all'RLU specificando l'opzione 19. Questo strumento ha tutte le funzionalità necessarie alla creazione e modifica di un file di stampa. Di seguito vengono riportati direttamente i testi dell'aiuto esteso accessibile da una sessione RLU; la prima parte è il testo di aiuto generico che vale ovunque. La seconda parte relativa ai tasti di funzione è suddivisa in due parti poiché l'RLU mette a disposizione due gruppi di tasti funzionali; uno base ed uno alternativo, si può passare dall'uno all'altro premendo il tasto F22.

```
L'RLU è utilizzato per preparare una immagine di prospetto, stampare
prospetti prototipo, creare file di stampa e generare le DDS dalle
quali sarà creato il prospetto.  Un prospetto prototipo è una copia
di stampa di un'immagine di prospetto che assomiglia al prospetto
che sarà generato da un programma applicativo.

Lo schermo Preparazione prospetto è lo schermo principale dal quale
viene editato il prospetto.  Da questo schermo è possibile
utilizzare i tasti funzionali per passare ad altri schermi nei qual
si possono aggiungere informazioni al file e ai livelli dei record
di campo.  E' possibile anche passare allo schermo Modifica dei
valori assunti per la sessione sul quale si può controllare le
caratteristiche della sessione di editazione.

Esistono due modi per stampare il prospetto prototipo.  Quando si
avvia l'RLU é possibile specificare l'opzione 6 per stampare dallo
schermo STRRLU. Dall'interno della sessione é possibile specificare
ciò che si desidera stampare dallo schermo Fine RLU.  Sullo schermo
Fine scegliere Y=Sì per la richiesta Prospetto prototipo ed il
prospetto sarà stampato.

Ci sono alcune cose da ricordare durante la sessione RLU:

o  Una volta che l'origine DDS è stata codificata per il prospetto,
   è possibile editare l'immagine utilizzando la riga campi, i
   tasti funzionali di contrassegno che consentono di spostare e
   copiare campi e dati e gli altri tasti funzionali che presentano
   gli schermi che saranno di ausilio per l'editazione del
   prospetto.

o  Prima di definire i campi, occorre definire un formato record.
   Per farlo, si può utilizzare il comando DR.

o  Assicurarsi che siano stati definiti tutti i campi per un
   record. Per farlo è possibile utilizzare il comando DF
   (definizione campo) oppure definire i campi sullo schermo
   Definizione informazioni di campo.  Per visualizzare lo schermo
   Definizione informazioni di campo premere F11 quando il cursore
   non è posizionato su un campo esistente.

o  Per visualizzare campi in sovrapposizione o campi per i quali il
   condizionamento è disattivato, usare il comando VF.  Esso
   visualizzerà una riga campi o righe campi multiple.

o  E' possibile eseguire una fusione dei record utilizzando il
   comando CL (modifica tipo riga).

o  In RLU vi sono due gruppi di tasti funzionali; un gruppo base e
   un gruppo alternativo; si può passare da uno all'altro con il
   tasto F22.  I tasti funzionali di base sono specifici dell'RLU.
   Si utilizzano per editare il prospetto con cui si sta lavorando
```

```
   e per accedere agli schermi RLU. I tasti funzionali alternativi
   sono di uso più comune rispetto a quelli di base e sono simili
   ai tasti funzionali dello schermo Editazione del SEU.
   Quando i tasti funzionali di base sono attivi, compare BASE
   sulla sinistra della riga di formattazione. Si tratta della
   terza riga partendo dall'inizio dello schermo che evidenzia le
   colonne nello schermo. Quando i tasti funzionali alternati sono
   attivi, compare ALT.

o  Non si dovrebbe spostare i campi nelle righe di riempimento.  Se
   ciò accade, tutto ciò che é stato definito per il campo andrà
   perso.

o  Nella parte superiore dello schermo Preparazione prospetto vi è
   una riga comandi utilizzabile per l'immissione di specifici
   comandi nella sessione RLU. Per immettere comandi di sistema,
   premere F21, uno dei tasti funzionali del gruppo alternativo.
   Sullo schermo comparirà una riga comandi di sistema.

Colonne
   Questo campo visualizza quali colonne sono attualmente visibili
   sullo schermo.  Il primo numero in questo campo rappresenta la prima
   colonna di righe di prospetto visualizzata.  Il secondo numero
   rappresenta l'ultima colonna visualizzata.

Libreria/file origine

   Il nome della libreria e del file origine contenente il prospetto
   che si sta editando.

Riga comando

   I comandi RLU forniscono un rapido accesso alle funzioni degli
   schermi Opzioni di ricerca/modifica, Opzioni di ricerca, Fine e
   Modifica dei valori assunti per la sessione, senza lasciare lo
   schermo di lavoro.  I comandi RLU sono a formato libero.  I
   parametri dei comandi sono obbligatori e posizionali oppure
   facoltativi, e possono essere immessi in qualsiasi ordine.
   Per eseguire un comando RLU. immetterlo sulla riga comandi e premere
   un tasto funzionale: ad esempio, Invio, Scorrimento in avanti,
   Scorrimento indietro, F19=Sinistra, F20=Destra.

   Alcuni comandi hanno delle abbreviazioni:  ad esempio, per eseguire
   il comando FIND, si può immettere F o FIND.  Per ricercare l'ultimo
   comando immesso, immettere F9=Duplicazione. Se il cursore si trova
   nell'area dati, si può usare F10=Cursore per spostare il cursore in
   avanti e indietro tra l'area dati e la riga comandi.  L'RLU
   memorizza fino a 50 comandi.  Per visualizzare l'aiuto per un
   comando RLU, immettere il comando sulla riga comandi e premere il
   tasto Aiuto.

   Nota:  Non si possono immettere comandi di sistema sulla riga
   comandi, ma lo si può fare direttamente sullo schermo di lavoro
   premendo F21=Comando di sistema. Sullo schermo appare una
   finestra in cui è possibile immettere un comando di sistema.
   Su una qualsiasi riga di uno schermo di lavoro (Editazione, Esame o
   Suddiviso), è possibile immettere i seguenti comandi RLU:

o  FIND o F

o  CHANGE o C

o  SAVE

o  CANCEL o CAN

o  FILE
```

o TOP

o BOTTOM

o SET o S

o HIDE o H

Nota: I comandi FILE, SAVE e CHANGE possono essere utilizzati soltanto in una sessione di editazione o sulla sessione superiore di uno schermo Editazione/esame suddiviso.

Il comando SET ha le seguenti opzioni:

o MATCH

o CAPS

o TABS

o ROLL

o EXPERT

o SHIFT

Nota: I comandi SHIFT e TABS possono essere utilizzati soltanto in una sessione di editazione o sulla sessione superiore di uno schermo Editazione/esame suddiviso.

Prospetto

Il nome del prospetto che si sta editando.

Tasti attivi

BASE indica che i tasti funzionali di base sono attivi. ALT indica che i tasti alternati sono attivi.

Riga formato

La riga graduata all'inizio dello schermo.
Zona controllo di riga

La zona dello schermo che consiste in numeri di riga e nomi di formati record. In questa zona è possibile immettere comandi di riga.

I seguenti comandi di riga sono disponibili in RLU, in aggiunta a molti altri comandi di riga SEU:

- Definizione formato record (DR)
- Modifica tipo riga (CL)
- Creazione dati campione (SD)
- Visualizzazione riga di campo (VF)

Nuova pagina (NP)

- Definizione costanti (DC)
- Definizione campi (DF)
- Centratura campi (CF)

```
   o  Spaziatura campi uniforme (SP)

Zona di lavoro

   L'area di lavoro per il prospetto che si sta preparando o editando.
   Questa area é formata dai seguenti tre tipi di righe:

Riga di prospetto

   Una riga di prospetto è una qualsiasi riga del prospetto facente
   parte di un formato record.  Le righe di prospetto sono
   interessate nella generazione dell'origine DDS. La prima riga di
   prospetto di un formato record è indicata con una R nell'area
   del numero riga.  Le righe di prospetto successive, denominate
   righe di continuazione, sono indicate con un + nell'area del
   numero riga.

Riga campione

   Una riga campione è una riga utilizzata per far si che il
   prospetto che si sta editando assomigli maggiormente al
   prospetto finale.  Le righe campione contengono dati campione
   associati al formato record precedente.  La prima riga di un
   gruppo di righe di esempio è indicata con una S nell'area del
   numero riga.  Le righe di esempio successive, denominate righe
   di continuazione, sono indicate con un segno + nell'area del
   numero riga. Queste righe non vengono implicate nella
   generazione dell'origine DDS.

Riga di riempimento
   Le righe di riempimento sono righe che separano un formato
   record dall'altro.   Esse vengono indicate con un punto (.)
   nell'area del numero righe.  Le righe di riempimento vengono
   implicate nella generazione dell'origine DDS.

Riga campo

   Una riga campo è un record temporaneo che può essere
   visualizzato sulla riga di prospetto dove è indicata con FLDn
   nell'area numero di sequenza  n=1, 2, o 3. La riga campo indica
   i limiti del campo all'interno del record, e consente
   l'editazione di campi nel record, ed accetta alcuni comandi per
   agevolare la generazione del campo.
```

ELENCO TASTI GRUPPO BASE

```
F1=Aiuto
   Premere F1 per ulteriore aiuto all'uso dello schermo.

F3=Fine
   Premere F3 per passare sullo schermo Fine RLU.

F4=Campi
   Premere F4 per utilizzare un elenco dei campi all'interno di un
   formato record DDS.

F5=Rivisualizzazione
   Premere F5 per ripristinare tutte le immissioni con il loro
   contenuto iniziale.

F6=Condizionamento schermo
   Premere F6 per passare sullo schermo Condizionamento
   preparazione prospetto in modo da attivare o disattivare gli
   indicatori per condizionare i campi e le parole chiave sul modo
   di visualizzarli sullo schermo di lavoro.

F9=Duplicazione
```

```
   Premere F9 per richiamare l'ultimo comando immesso.

F10=Campi data base
    Premere F10 per passare sullo schermo Gestione campi database,
    su cui è possibile creare un elenco di campi database da
    visualizzare in basso sullo schermo di lavoro per l'inclusione
    nel prospetto.

F11=Definizione campo
    Premere F11 quando il cursore non è su un campo esistente;
    compare lo schermo Definizione informazioni di campo, su cui è
    possibile creare un nuovo campo.

F12=Annullamento
    Premere F12 per annullare la sessione Suddivisione
    editazione/esame o la sessione Suddivisione
    editazione/richiesta.  Si tornerà su una singola sessione di
    Preparazione prospetto.

F13=Dati contrassegnati/non contrassegnati
    Premere F13 per contrassegnare un carattere, un campo o una
    parte del testo da copiare o spostare. Se qualcosa é stata
    contrassegnata ed il cursore viene spostato su un'altra zona e
    si preme nuovamente F13, la nuova zona verrà evidenziata ad alta
    intensità assieme alla prima zona che é stata evidenziata. Ora é
    possibile utilizzare F14 per copiare i dati nella nuova zona
    contrassegnata oppure F15 per spostarvi i dati.  Premendo
    nuovamente F13 con il cursore sull'area contrassegnata, verrà
    tolto il contrassegno al testo. Il modo di spostare o copiare
    può essere modificato passando sullo schermo Modifica dei valori
    assunti per la sessione.

F14=Copia dati contrassegnati
    Premere F14 per copiare una zona contrassegnata in un'altra
    zona.  Quando si preme F14 il proprio cursore dovrebbe essere
    nell'angolo in alto a sinistra della zona nella quale si sta
   eseguendo la copia.

F15=Spostamento dati contrassegnati
    Premere F15 per spostare una zona contrassegnata in un'altra
    zona. Posizionare il cursore nell'angolo in alto a sinistra
    della zona nella quale si sta spostando la zona contrassegnata e
    premere F15.

F16=Cancellazione campo
    Premere F16 su una riga di campo, una riga di prospetto o una
    riga campione per cancellare dallo schermo Preparazione
    prospetto il testo del campo e i dati campione associati e tutte
    le informazioni a livello campo associate.

F17=Parole chiave file
    Premere F17 per passare sullo schermo Gestione parole chiave di
    file, su cui è possibile modificare la definizione a livello
    file. Se l'ultima volta che si sono utilizzate parole chiave è
    stato usato lo schermo Immissione parole chiave, quando viene
    premuto F17 compare lo schermo Immissione parole chiave file.

F18=Parole chiave record
    Premere F18 sulla riga del prospetto o sulla riga di esempio per
    passare sullo schermo Gestione parole chiave record.  Se
    l'ultima volta che si sono utilizzate parole chiave è stato
    usato lo schermo Immissione parole chiave, quando viene premuto
    F18 compare lo schermo Immissione parole chiave record.

F19=Sinistra
    Premere F19 per spostare la zona record sulla sinistra dello
    schermo.
```

```
F20=Destra
    Premere F20 per spostare la zona record sulla destra dello
    schermo.

F21=Comando di sistema
    Premere F21 per visualizzare una finestra di immissione su cui
    poter immettere comandi di sistema.

F22=Tasti alternativi
    Premere F22 per visualizzare la serie alternativa dei tasti
    funzionali.

F23=Parole chiave campo
    Premere F23 con il cursore posizionato su un campo esistente;
    compare lo schermo Gestione parole chiave campo, su cui è
    possibile modificare le definizioni di campo.  Se l'ultima volta
    che si sono utilizzate parole chiave è stato usato lo schermo
    Immissione parole chiave, quando viene premuto F23 compare lo
    schermo Immissione parole chiave campo.

F24=Altri tasti
    Premere F24 per visualizzare gli altri tasti funzionali
    disponibili.
```

ELENCO TASTI GRUPPO ALTERNATIVO

```
F1=Aiuto
   Premere F1 per ulteriore aiuto all'uso dello schermo.

F3=Fine
   Premere F3 per passare sullo schermo Fine RLU.

F4=Richiesta
   Premere F4 con il cursore su una riga per visualizzare quella
   riga nella parte inferiore dello schermo.  E' possibile quindi
   editare la riga su cui é stata effettuata la richiesta.

F5=Rivisualizzazione
   Premere F5 per ripristinare tutte le immissioni con il loro
   contenuto iniziale.

F6=Spostamento della riga di divisione
   Premere F6 per ridistribuire lo spazio nelle parti superiore e
   inferiore dello schermo suddiviso.

F9=Duplicazione
   Premere F9 per richiamare l'ultimo comando immesso.

F10=Cursore
    Premere F10 per spostare il cursore tra la riga comandi e
    l'ultima posizione sullo schermo in cui si trovava il cursore.

F12=Annullamento
    Premere F12 per annullare la sessione Suddivisione
    editazione/esame o la sessione Suddivisione
    editazione/richiesta.  Si tornerà su una singola sessione di
    Preparazione prospetto.

F13=Modifica dei valori assunti per la sessione
    Premere F13 per passare allo schermo Modifica dei valori assunti
    per la sessione su cui é possibile inserire i valori assunti per
    il controllo della visualizzazione della sessione Preparazione
    prospetto.

F14=Opzioni ricerca/modifica
    Premere F14 per passare allo schermo Opzioni ricerca/modifica.
```

```
F15=Opzioni esame/copia
    Premere F15 per passare allo schermo Opzioni esame/copia.

F16=Ripetizione ricerca
    Premere F16 per ricercare la stringa carattere specificata nella
    richiesta Ricerca sullo schermo Opzioni ricerca/modifica.

F17=Ripetizione modifica
    Premere F17 per ricercare la stringa di caratteri specificata
    nella richiesta Ricerca sullo schermo Opzioni ricerca/modifica e
    cambiarla con la stringa di sostituzione specificata.

F18=Conversione DBCS
    Premere F18 per la conversione DBCS.  Il cursore deve trovarsi
    in una zona dello schermo di tipo DBCS e sotto un carattere a
    due-byte, il carattere 'shift-in' o una zona vuota tra i
    caratteri di controllo 'shift'.
    Per terminare la conversione DBCS premere nuovamente F18.

F19=Sinistra
    Premere F19 per spostare la zona record sulla sinistra dello
    schermo.

F20=Destra
    Premere F20 per spostare la zona record sulla destra dello
    schermo.

F21=Comando di sistema
    Premere F21 per visualizzare una finestra di immissione su cui
    poter immettere comandi di sistema.

F22=Tasti base
    Premere F22 per visualizzare la serie di base dei tasti
    funzionali.


F24=Altri tasti
    Premere F24 per visualizzare gli altri tasti funzionali
    disponibili.
```

46 Resti da riposizionare.

46.1 Specifica I/IX (I =File interni, IX =File esterni).

Questa specifica di immissione serve per potere specificare un nome di tracciato record di un file precedentemente descritto nelle specifiche di definizione dei file (F/FX) e ad associargli un indicatore che viene attivato automaticamente nel momento in cui un record viene acquisito dal programma. Questo espediente viene utilizzato nel caso in cui un file sia un combinato, ovvero veda diversi tracciati, per conoscere, dopo una operazione di acquisizione di un record, a quale file appartiene. Esempio di associazione di un indicatore a due record.

```
IANCLI        01
IANFOR        02
```

In questo caso, quando si acquisirà un record dal tracciato ANCLI, automaticamente la condizione dell'indicatore *IN01 sarà *ON . Analogamente quando un record verrà acquisito dal tracciato ANFOR, *IN02 sarà *ON .

Un altro impiego di questa specifica è quello per potere definire le rotture di livello per un file definito come primario. Esempio di definizione del record del file primario e relativi campi e rotture di livello associate.

```
tracciato I
IANCLI
I                                              FLDAA L5
I                                              FLDBB L4
I                                              FLDHH L3
I                                              FLDKK L2
I                                              FLDXX L1
```

46.2 Specifica JX (Definizione e ridenominazione campi file esterni).

Questa specifica di immissione serve per definire e rinominare i campi di un file. Esempio di ridenominazione di un campo di un file.

```
tracciato I
I               FLD01                          FLD01X
```

In questo caso il campo FLD01 nel programma si chiamerà FLD01X.

Esempio di ridenominazione di un campo di un file e contestuale utilizzo dello stesso in una schiera.

```
tracciato I
I               FLD01                          QT,01
```

In questo caso il campo FLD01 nel programma sarà il 1° elemento della schiera QT.

```
  ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
```

Fine del documento