1. **Prologo.**

Per non far torto ai modernisti, accantono il mio vecchio e incompleto **manuale** **As400** scritto con **Riccardo Battistutta** e riscrivo di getto questo, riguardante ancora **CL (Control Language)** e **RPG (Report Program Generator)**.  
Il vecchio corso è stato la mia principale referenza nell’ultimo decennio e deve la sua notorietà al fatto che chi ha approcciato As400 nello stesso periodo per prima cosa ha cercato sul web un manuale gratuito in italiano.  
Incredibile ma vero, ha trovato solo il mio.  
Per questo nuovo, riparto da un elenco scritto per dare la traccia a un paio di corsi che sto tenendo sull’argomento con modalità diverse.  
Ripasserò più volte l’elenco sostanziando gli argomenti fino ad esserne, per quanto possibile, soddisfatto.  
Queste note vogliono essere l’elenco delle nozioni necessarie per affrontare la programmazione in ambito As400.  
Avendo osservato che l’uditorio non è più formato da semplici sbarbati, il manuale non è rivolto al principiante assoluto ma a chi abbia già familiarità almeno scolastica con un qualunque database ed un qualunque linguaggio di programmazione.  
Come se si dovesse insegnare a qualcuno non ad andare in bicicletta ma ad usare anche una bicicletta diversa.  
  
Cadorago, 8 dicembre 2014

Claudio Neroni

[www.neroni.it](http://www.neroni.it)

1. **Raccomandazione.**
   1. Molte delle azioni qui raccontate sono potenzialmente pericolose per sorgenti, programmi e dati. All’inizio dello studio, fatti assegnare un profilo utente di bassa autorità e fai riferimento ad un tutore esperto per accedere alle risorse man mano necessarie.
   2. Per ottenere gratuitamente un utente su un ambiente di prova di 50 Mb su una macchina pubblica, consulta il link [www.pub1.de](http://www.pub1.de) .  
      L’utente fornito ha una autorità abbastanza bassa e non permette tutto quello che è autorizzato al programmatore su un As400 dotato dell’installazione standard. E’ cioè sufficiente per svolgere le attività di programmazione applicativa ma non permette molte attività di tipo sistemistico.
2. **Utilizzo del video comandi.**
   1. Per maneggiare un As400 sono presenti sul sistema molte interfacce parziali ma privilegiamo il video comandi perché fornisce al programmatore l’accesso più ampio e la migliore retroazione.
   2. Il video comandi viene chiamato dal comando  
      **CALL QCMD**oppure, meglio, come programma iniziale del profilo utente del programmatore sul quale si sia preventivamente compiuta la modifica  
      **CHGPRF INLPGM(QSYS/QCMD) INLMNU(\*SIGNOFF)**
   3. Il video comandi permette di:
      1. Digitare un comando nell’area di digitazione in forma completa  
         **COMANDO PAROLACHIAVE1(PARAMETRO1) PAROLACHIAVE2(PARAMETRO2) ...**
      2. Utilizzare il prompter: digitare il nome di un comando e premere ***F4*** per accedere al video di servizio che permette la ulteriore digitazione assistita dei parametri del comando.
      3. Inviare il comando digitato per l’esecuzione premendo il tasto ***Invio*** (***Enter***).
      4. Leggere i messaggi di risposta inviati al video comandi dall’esecuzione del comando immesso.
      5. Visualizzare il log a video contenente tutti i messaggi (normalmente nascosti) inviati a programmi diversi dal video comandi premendo ***F10***. Lo stesso comando si usa per tornare alla forma precedente.
      6. Visualizzare le pagine precedenti del log generato dal video comandi facendo uso del tasto ***PageUp*** (in passato chiamato ***RollDown***) o le successive rispetto a quella visualizzata facendo uso del tasto ***PageDown*** (in passato chiamato ***RollUp***).
      7. Visualizzare un messaggio del log al massimo dettaglio posizionando il cursore sulla riga del messaggio e premendo ***F1*** per accedere ad una videata contenente tutte le informazioni aggiuntive sul messaggio stesso. Dal dettaglio tornare al log con ***Invio***, ***F12*** o ***F3***.
      8. Recuperare l’ultimo comando eseguito nella sessione corrente premendo ***F9***. Il comando viene messo nell’area di digitazione pronto ad eventuali modifica ed esecuzione. La prima volta ***F9*** recupera l’ultimo comando. L’immediatamente successivo ***F9*** recupera il comando precedente e così via.
      9. Recuperare in modo mirato un comando già eseguito. Se si rolla (***PageUp***, ***PageDown***) per vedere pagine precedenti e si posiziona il cursore su un comando già eseguito, ***F9*** lo recupera come già detto.  
         Se invece di quest’ultimo ***F9*** si digita ***F4*** il comando recuperato compare direttamente prompterizzato.
   4. In sostanza il video comandi permette l’esecuzione dei comandi estemporanei e contemporaneamente visualizza il log dei comandi digitati ed eseguiti e dei messaggi da essi generati.
   5. Infine, a beneficio dei disgraziati costretti all’uso di **RDI** (**Rational Developer for i**), occorre qui mettere in guardia sul surrogato di log del video comandi che viene usato nella finestra comandi di **RDI**.  
      La finestra in questione evidenzia un log simile a quello ottenibile dopo la pressione di ***F10*** sul video di **QCMD** ma senza possibilità di minore o maggiore dettaglio.  
      Alla faccia del web, manca completamente la navigabilità del log con perdita irrimediabile della chiarezza originale. Come dico spesso, siamo caduti nelle mani del nemico.
3. **Utilizzo dei programmi e dei comandi.**
   1. I programmi sono il mezzo per ottenere dal sistema le elaborazioni dei dati che costituiscono lo scopo finale del sistema stesso.
   2. Nei casi più semplici, per chiamare un programma basta il comando generico **CALL**, seguito dai parametri che il programma richiede.  
      In forma parole chiave:  
      **CALL PGM(PGM1) PARM(PARM1 PARM2)**Oppure in forma posizionale:  
      **CALL PGM1 (PARM1 PARM2)**
   3. I comandi sono invece l’interfaccia evoluta verso i programmi e sono destinati soprattutto al programmatore e al sistemista.
   4. Ogni specifico comando è l’interfaccia di chiamata ad uno specifico programma.
   5. I programmi chiamati dai comandi possono essere quelli del sistema ma anche programmi di utilità costruiti dal programmatore o, in via eccezionale, programmi applicativi per uso dell’utente finale.
   6. Gli applicativi destinati all’utente finale non fanno uso normalmente dell’interfaccia comandi ma usano solitamente dei menù congegnati in modo da fornire tutte e solo le funzioni che competono all’utente stesso. I menù sono programmi rpg o cl che usano un video che elenca le chiamate possibili. Chiedono la digitazione di un’opzione alla volta corrispondente alla chiamata desiderata e usano una call o una submit job per eseguire o sottomettere l’azione corrispondente.
   7. Un menù di menù chiama solo altri menù.  
      Un menù di azioni chiama solo programmi applicativi.
   8. Il comando che il programmatore può costruire permette una chiamata assistita al medesimo programma richiamabile con la semplice **CALL**. Il comando si fa riconoscere non solo con il nome ma, tramite il prompter, anche con una descrizione. Aiuta inoltre la digitazione dei parametri con nome, descrizione, lunghezza, tipo, valori ed altro. I parametri sono validati autonomamente dal comando prima di chiamare il programma esecutore. E’ possibile anche usare, se previsto, un apposito validity checker.  
      La chiamata del comando avrà la forma con parole chiave:  
      **COMANDO1 PAROLACHI1(PARM1) PAROLACHI2(PARM2)**oppure la forma posizionale   
      **COMANDO1 PARM1 PARM2**oppure la forma mista  
      **COMANDO1 PARM1 PAROLACHI2(PARM2)**Il numero massimo di parametri posizionali è definito nel comando.  
      Nella forma mista, dopo la prima parola chiave si possono indicare solo altre parole chiave.
4. **Utilizzo del prompter comandi.**
   1. Il prompter (suggeritore) è il servizio che utilizza la definizione del comando per facilitare l’immissione dei parametri del comando stesso.

**Testo del comando (COMANDO1)**

**Immettere le scelte e premere Invio.**

**Testo di aiuto del parametro 1 . PAROLACHI1 PARM1**

**Testo di aiuto del parametro 2 . PAROLACHI2 PARM2**

* 1. Il comando che esce dal prompter è sempre nella forma con parole chiave.
  2. Per il comando è possibile definire, con vari metodi, un testo d’aiuto strutturato in intestazione e parametri.
  3. Se per il comando è definito il testo d’aiuto, premendo ***F1*** se ne ottiene la visualizzazione. La richiesta è sensibile al contesto: se il cursore si trova sulla riga di un parametro, il testo di aiuto è quello relativo al parametro. Altrimenti il testo di aiuto è quello generale, comprensivo di intestazione e parametri.
  4. Il prompter supporta alcune regole di minutazione che permettono di ottenerne le prestazioni.  
     Per ritovare queste regole, ***F13*** sul prompter.  
     Digitando nelle prime due posizioni di un parametro di immissione e premendo invio, si ottiene una variazione nella visualizzazione del parametro. Se si ricalca il contenuto emesso dal prompter, questo non viene perso ma in vario modo riproposto inalterato nel parametro opportuno.  
     Qui di seguito **Ѣ** sta per **blank**.
     1. **&Ѣ** allunga il campo a disposizione. La richiesta è ripetibile e provoca allungamento fino ad un limite legato al tipo di campo.
     2. **+Ѣ** sull’elemento di un parametro capace di contenere una lista, propone un gruppo di elementi vuoti in coda all’ultimo elemento pieno della lista.
     3. **>Ѣ** sull’elemento di un parametro capace di contenere una lista, propone un gruppo di elementi vuoti da inserire nella lista prima dell’elemento che ha ricevuto la digitazione.
  5. Anche il comando **CALL** ha un prompter che però non fornisce alcun soccorso a chi non conosce con esattezza ogni singolo parametro richiesto dal programma chiamato.
  6. Il prompter può essere chiamato non solo premendo ***F4*** dopo aver digitato almeno il nome comando ma anche facendo uso del simbolo **?** (interrogativo). Queste regole valgono all’invio in interattivo sul video comandi o all’esecuzione nei programmi interattivi.
     1. Anteporre **?** al nome del comando per ottenere il prompt con modificabili tutti i parametri.  
        **?CHGLIBL**
     2. Anteporre ?- ad una parola chiave, solo se già anteposto ? al comando, per ottenere il prompt con modificabili tutti i parametri all’infuori di quello corrente.  
        **?CHGLIBL ?-LIBL(QGPL QTEMP)**
     3. Anteporre ?? ad una parola chiave per ottenere il prompt con il relativo parametro evidente e modificabile.   
        **CHGLIBL ??LIBL(QGPL QTEMP)**
     4. Anteporre ?\* ad una parola chiave per ottenere il prompt con il relativo parametro evidente ma non modificabile.  
        **CHGLIBL ?\*LIBL(QGPL QTEMP)**
     5. **?&**, **?%**, **?/** , **?<** sono da trascurare influendo in maniera molto opinabile sull’esecuzione del comando poiché mostrano valori che possono anche non essere presi in considerazione dall’esecuzione.

1. **Definizione di lavoro.**
   1. Il protagonista fondamentale dell’elaborazione dati è il Job (Lavoro).
   2. Il job è l’attività del processore che esegue i programmi.
   3. Job interattivo è quello che interagisce con l’utente al terminale, fisico una volta, virtuale oggi. Di solito il terminale virtuale è un pc dotato di un programma di emulazione di terminale fisico. Può essere considerato job interattivo anche l’ammucchiata di interazioni Pc-As400 costituito da una sessione RDI (Rational Developer for i).
   4. Job batch è quello che nasce da un altro job, interattivo o batch, ma che non interagisce direttamente con un utente al terminale.
   5. Interattivo e batch hanno criteri di sviluppo e tecniche elaborative radicalmente diverse.
   6. Un Job viene indagato e, se necessario, gestito dal comando **WRKJOB**(Gestione Lavoro).
2. **Work Job.**
   1. Documentare WRKJOB.
3. **Principali comandi di gestione dei lavori.**
   1. Work with User Jobs (Lavora coi Lavori Utente)
      1. Prompt.

**Gestione lavori utente (WRKUSRJOB)**

**Immettere le scelte e premere Invio.**

**Utente di cui visual. lavori . . USER \***

**Stato dei lavori utente . . . . STATUS \*ALL**

**Emissione . . . . . . . . . . . OUTPUT \***

**Tipo di lavoro . . . . . . . . . JOBTYPE \*ALL**

* + 1. Premendo invio senza alterare i default del comando, si ottiene l’elenco di lavori dell’utente corrente.

**Gestione lavori utente S65D69DA**

**09/12/14 15:39:10**

**Immettere le opzioni e premere Invio.**

**2=Modif. 3=Congelamento 4=Fine 5=Gest. 6=Ril.**

**7=Visualizzazione messaggio 8=Gestione file di spool 13=Scollegamento**

**Opz Lavoro Utente Tipo -----Stato------ Funzione**

**H1NERO UFNERONI INTER ACTIVE COM-WRKUSRJOB**

**H2NERO UFNERONI INTER ACTIVE \* -CMDENT**

**JLIBL. UFNERONI BATCH OUTQ**

**Fine**

**Parametri o comando**

**===>**

**F3=Fine F4=Richiesta F5=Rivisual. F9=Duplicaz.**

**F11=Visualizzazione dati pianificazione F12=Annull. F24=Altri tasti**

* + 1. Su ciascuno dei job elencati è possibile eseguire un’opzione a scelta tra quelle elencate nell’intestazione. Ciascuna opzione esegue a sua volta un comando sopra il job corrispondente precompilando un adeguato comando.
       - 2 = CHGJOB Change Job
       - 3 = HLDJOB Hold Job
       - 4 = ENDJOB End Job
       - 5 = WRKJOB Work Job
       - 6 = RLSJOB Release Job
       - 7 = Visualizza messaggio pendente del job (senza prompt)
       - 8 = WRKJOB OPTION(\*SPLF) Work Job Spool Files
       - 13 = DSCJOB Disconnect Job
  1. Work with Submitted Jobs (WRKSBMJOB) (Gestione lavori sottomessi).
     1. Prompt.

**Gestione lavori immessi (WRKSBMJOB)**

**Immettere le scelte e premere Invio.**

**Lavori immessi da . . . . . . . SBMFROM \*USER**

**Emissione . . . . . . . . . . . OUTPUT \***

* + 1. Premendo invio senza alterare i default del comando, si ottiene l’elenco di lavori sottomessi dall’utente corrente.

**Gestione lavori inoltrati S65D69DA**

**09/12/14 16:20:38**

**Inoltrati da . . . . . . . . . : \*USER**

**Immettere le opzioni e premere Invio.**

**2=Modif. 3=Congelamento 4=Fine 5=Gest. 6=Ril.**

**7=Visualizzazione messaggio 8=Gestione file di spool**

**Opz Lavoro Utente Tipo -----Stato------ Funzione**

**JLIBL. UFNERONI BATCH OUTQ**

**Fine**

**Parametri o comando**

**===>**

**F3=Fine F4=Richiesta F5=Rivisual. F9=Duplicaz.**

**F11=Visualizzazione dati pianificazione F12=Annull. F17=Inizio F18=Fine**

* + 1. Su ciascuno dei job elencati è possibile eseguire un’opzione a scelta tra quelle elencate nell’intestazione. Ciascuna opzione esegue a sua volta un comando sopra il job corrispondente precompilando un adeguato comando, similmente a quanto esposto sul comando WRKUSRJOB.
  1. Work with Active Jobs (WRKACTJOB) (Gestione lavori attivi).
     1. Prompt.

**Gestione lavori attivi (WRKACTJOB)**

**Immettere le scelte e premere Invio.**

**Emissione . . . . . . . . . . . OUTPUT \***

* + 1. Premendo invio senza alterare i default del comando, si ottiene l’elenco di tutti i lavori attivi sulla macchina.  
       L’elenco può essere molto lungo

**Gestione lavori attivi S65D69DA**

**09/12/14 16:46:47**

**% CPU: 14,4 Tempo trascorso: 00:05:05 Lavori attivi: 991**

**Immettere le opzioni e premere Invio.**

**2=Modif. 3=Congelamento 4=Fine 5=Gest. 6=Ril.**

**7=Visualiz. messaggio 8=Gestione file di spool 13=Scollegamento...**

**Utente**

**Opz Sottosis/Lav corrente Tipo % CPU Funzione Stato**

**QINTER QSYS SBS 0,0 DEQW**

**+AS\_400\_H1 UFSIGNORE INT 0,0 PGM-MENPGMRR DSPW**

**AS\_400\_H2 UFSIGNORE INT 0,0 COM-STRPDM DSPW**

**BAGGI01 UFYZEIRI INT 0,0 PGM-MENPGMRR DSPW**

**BIFFI01 UFPIACENT INT 0,0 \* -CMDENT DSPW**

**CARRA01 UFPIACENT INT 0,0 \* -CMDENT DSPW**

**CEPPI01 UFMDUCA INT 0,0 COM-STRSEU DSPW**

**COSTA01 UFDEGRAND INT 0,0 COM-STRPDM DSPW**

**DACCI01 UFDEGRAND INT 0,0 COM-STRPDM DSPW**

**Segue...**

**Parametri o comando**

**===>**

**F3=Fine F5=Rivisual. F10=Riavvio statistiche**

**F11=Visualizzazione dati scaduti F12=Annull. F24=Altri tasti**

* + 1. Su ciascuno dei job elencati è possibile eseguire un’opzione a scelta tra quelle elencate nell’intestazione. Ciascuna opzione esegue a sua volta un comando sopra il job corrispondente precompilando un adeguato comando, similmente a quanto esposto sul comando WRKUSRJOB.
    2. Il comando WRKACTJOB è interessante per le parzializzazioni possibili in chiamata ma, soprattutto, perché permette l’analisi del peso di ciascun lavoro sulle risorse del processore.  
       In particolare portando il cursore sulla colonna **% CPU** e premendo ***F16***(Resequence), si ottiene l’ordinamento delle righe sulla base del valore riportato nella colonna: il risultato permette di trovare i lavori di maggiore impatto sul sistema.  
       In testata sono presenti anche i valori totali dell’occupazione processore. Le righe di alcuni dei lavori di sistema o non compaiono o non documentano correttamente l’occupazione e perciò il totale non raggiunge mai il 100%.
    3. Si osservino anche i comandi di piede della visualizzazione, in particolare ***F5*** (Refresh) e ***F10*** (Restart).

1. **Principali comandi di gestione del sistema.**
   1. Work with System Status (WRKSYSSTS) (Gestione stato del sistema).
      1. Prompt.

**Gestione stato del sistema (WRKSYSSTS)**

**Immettere le scelte e premere Invio.**

**Emissione . . . . . . . . . . . OUTPUT \***

**Ripristino statistiche stato . . RESET \*NO**

* + 1. Premendo invio senza alterare i default del comando, si ottengono informazioni sull’occupazione delle risorse e l’elenco dei lotti di memoria della macchina con i valori correnti di memoria assegnata.

**Gestione stato sistema S65D69DA**

**09/12/14 17:30:01**

**% CPU utilizzata . . . . : 15,6 Memoria ausiliaria:**

**Tempo trascorso . . . . : 00:00:01 Sistema ASP . . . . . : 9314 G**

**Lavori nel sistema . . . : 102161 % sistema ASP utiliz . : 95,0006**

**% indirizzi permanenti . : 0,439 Totale . . . . . . . . : 9314 G**

**% indirizzi temporanei . : 2,503 Utilizzo non protetto**

**corrente . . . . . . : 22354 M**

**Non protetto massimo . : 22516 M**

**Immettere modifiche (se consentito), premere Invio.**

**Lotto Dim (M) Dim (M) Attivo -----DB----- ---Non-DB---**

**sist lotto riservata max Err Pag Err Pag**

**1  9442,83 571,13 +++++ 0,0 0,0 13,9 211,5**

**2 4171,36 35,88  1231 284,7 585,1 840,3 10325**

**3  1000,00 0,00    76 0,0 0,0 0,0 0,0**

**4  1623,79 0,00   406 40,3 40,3 74,0 107,8**

**Fine**

**Comando**

**===>**

**F3=Fine F4=Richiesta F5=Rivisualiz. F9=Duplicaz. F10=Riavvio**

**F12=Ann. F19=Stato sistema esteso F24=Altri tasti**

**Rilevata condizione critica di memoria.**

* + 1. Se si è dotati di autorità sufficiente, si possono modificare i valori sottolineati.

1. **Librerie, file e membri sorgenti.**
   1. Il contenitore di un sorgente è normalmente il membro di un file. Può essere anche un file IFS, del quale, per ora, non parlerò.
   2. Anticipo queste note rispetto alla definizione di librerie, file, membri per dare subito una sede alla scrittura dei sorgenti da cui nascono il database ed i programmi oggetto mediante apposite attività di “compilazione”.
   3. Creiamo innanzitutto una libreria che, pur potendo contenere ogni tipo di oggetto, destineremo a contenitore dei sorgenti.  
      **CRTLIB LIB(CORLIB) TEXT(‘Libreria sorgente’)**
   4. Creiamo ora un file sorgente **CORSO** dentro la libreria sorgente **CORLIB**.  
      **CRTSRCPF FILE(CORLIB/CORSO) RCDLEN(112) TEXT('Corso As400')**La lunghezza record è 112 (100 dati, 6 sequenza, 6 data) perché in tale lunghezza sono ospitabili tutti i tipi di sorgenti, in particolare gli RPG ILE con un campo dati di 100 caratteri. Il tipo sorgente RPGLE ha sfondato per primo l’antica misura aurea di 80 caratteri per il campo dati.
      1. I file sorgenti possono essere considerati una sottospecie di file dati ma hanno un comando di creazione specifico e non necessitano di un sorgente.
      2. Tutti i file sorgente hanno i campi seguenti.  
          **Data Field Buffer  
         Field Type Length Length  
         SRCSEQ ZONED 6 2 6  
         SRCDAT ZONED 6 0 6  
         SRCDTA CHAR 100 100**ma possono differire per la lunghezza dell’ultimo campo contenente una riga di dati che, in questo caso, è una riga di programma sorgente.
      3. Un file sorgente è gestito soprattutto col SEU ma nulla vieta di trattarlo come un file dati tramite un programma che potrebbe leggerlo e scriverlo.
   5. Aggiungiamo infine un membro CLS al file sorgente.  
      **ADDPFM FILE(CORLIB/CORSO) MBR(CLS) TEXT('Stringa CL') SRCTYPE(CL)**
   6. Da ultimo, aggiungiamo righe al sorgente tramite il SEU (Source Entry Utility).  
      **STRSEU SRCFILE(CORLIB/CORSO) SRCMBR(CLS)**Il SEU, in assenza del membro, lo aggiunge.  
      In tal caso sono opportuni altri parametri, come segue.  
      **STRSEU SRCFILE(CORLIB/CORSO) SRCMBR(CLS) TYPE(CL) TEXT('Stringa CL')**
2. **Scrittura di una stringa Control Language (CL).**
   1. Un sorgente di tipo CL (Control Language), detto anche Job Stream, può contenere un elenco di comandi eseguibili uno dopo l’altro senza passaggio di variabili da un comando all’altro.  
      Supponiamo di avere inserito con il SEU le seguenti righe nel membro CLS  
      **//BCHJOB JOB(CLS)  
      DSPSYSSTS OUTPUT(\*PRINT)  
      DSPLIBL OUTPUT(\*PRINT)  
      ADDLIBLE LIB(NERONI2)  
      DSPLIBL OUTPUT(\*PRINT)  
      //ENDBCHJOB**Il primo e l’ultimo comando (**BCHJOB/ENDBCHJOB**) definiscono e delimitano il gruppo di comandi da eseguire.
   2. La Job Stream non necessita di creazione ma solo di esecuzione estemporanea tramite il comando Submit Data Base Jobs (**SBMDBJOB**).  
      **SBMDBJOB FILE(CORLIB/CORSO) MBR(CLS) JOBQ(QBATCH)**La sottomissione fa nascere un lavoro batch che viene inserito in una coda lavori ed eseguito da un sottosistema. A fine esecuzione si otterranno le stampe richieste dai comandi scritti nella Job Stream.
3. **Scrittura di un programma in Control Language (CLLE) con uso di variabili.**
   1. Avendo già creati libreria e file sorgente, carichiamo il nuovo sorgente direttamente con il SEU.  
      **STRSEU SRCFILE(CORLIB/CORSO) SRCMBR(CLVAR) TYPE(CLLE) TEXT('Pgm con variabili CL')**
   2. Il programma emette in joblog nome macchina e numero seriale della macchina.
   3. Ipotizziamo le seguenti istruzioni e abbondiamo di commenti a scopo didattico.

**/\* Claudio Neroni 14-12-2014 Creato \*/  
/\* Programma CL con variabili \*/  
  
/\* Riceve Testo obbligatorio \*/  
 PGM PARM(&TXT)  
/\* Definisce Testo \*/  
 DCL VAR(&TXT) TYPE(\*CHAR) LEN(32)  
/\* Definisce Messaggio \*/  
 DCL VAR(&MSG) TYPE(\*CHAR) LEN(100)  
/\* Definisce Numero seriale \*/  
 DCL VAR(&QSRLNBR) TYPE(\*CHAR) LEN(8)  
/\* Definisce Nome sistema \*/  
 DCL VAR(&SYSNAME) TYPE(\*CHAR) LEN(8)  
/\* Esige il parametro testo \*/  
 CHGVAR VAR(&TXT) VALUE(&TXT)  
 MONMSG MSGID(CPF0000 MCH0000) EXEC(DO)  
 SNDPGMMSG MSG('Non mi hai dato un testo') MSGTYPE(\*DIAG)  
 RETURN  
 ENDDO  
/\* Esige un contenuto per il testo \*/  
 IF COND(&TXT \*EQ ' ') THEN(DO)  
 SNDPGMMSG MSG('Testo in bianco') MSGTYPE(\*DIAG)  
 RETURN  
 ENDDO  
/\* Recupera Numero seriale \*/  
 RTVSYSVAL SYSVAL(QSRLNBR) RTNVAR(&QSRLNBR)  
/\* Recupera Nome sistema \*/  
 RTVNETA SYSNAME(&SYSNAME)  
/\* Compone Messaggio \*/  
 CHGVAR VAR(&MSG) VALUE(&TXT \*BCAT 'Sistema:"' \*CAT +  
 &SYSNAME \*CAT '" Macchina:"' \*CAT +  
 &QSRLNBR \*CAT '"')  
/\* Invia Messaggio \*/  
 SNDPGMMSG MSG(&MSG) MSGTYPE(\*COMP)  
 ENDPGM**

* 1. Il programma riceve parametri, contiene variabili, messaggi di completamento, intercettazione di errori e conseguenti messaggi diagnostici, recupero di valori di sistema, composizione di stringhe.
  2. Per poter essere eseguito, il programma va trasformato da sorgente ad oggetto eseguibile tramite il comando di creazione.  
     **CRTBNDCL PGM(CORLIB/CLVAR) SRCFILE(CORLIB/CORSO) SRCMBR(CLVAR) REPLACE(\*YES)**Per i programmi del corso assumiamo di creare gli oggetti nella stessa libreria in cui giacciono i sorgenti. Si tratta di una scelta semplificativa molto adatta ad esperimenti veloci e a perdere. Quando si tratta di produzione è invece più opportuno avere librerie diverse per sorgenti, programmi e dati. E tre serie di tutto per sviluppo, test e produzione.
  3. La chiamatain esecuzione del programma oggetto senza parametri risponde con un messaggio diagnostico.  
     **> call clvar  
      Non mi hai dato un testo**
  4. La chiamata del programma con parametro bianco risponde con un messaggio diagnostico.  
     **> call clvar ‘ ‘  
      Testo in bianco**
  5. La chiamata del programma con un parametro valido risponde con un messaggio di completamento.  
     **> call clvar 'Prova su'  
      Prova su Sistema:"PUB1 " Macchina:" 652381B"**

1. **Scrittura di un programma in Control Language (CLLE) con uso di outfile dei comandi di sistema.**
   1. Come sopra, si carica un sorgente con il SEU, si compila e si esegue.  
      **STRSEU SRCFILE(CORLIB/CORSO) SRCMBR(CLOUT) TYPE(CLLE) TEXT('Pgm con outfile')**
   2. Il programma riceve un prefisso di nome libreria ed elenca in joblog le librerie con il prefisso ricevuto.
   3. Ipotizziamo le seguenti istruzioni e abbondiamo di commenti a scopo didattico.

**/\* Claudio Neroni 14-12-2014 Creato \*  
/\* Programma CL con outfile \*/  
  
 PGM PARM(&LIBGEN)  
/\* Riceve un nome generico \*/  
 DCL VAR(&LIBGEN) TYPE(\*CHAR) LEN(10)  
/\* Elenco librerie \*/  
 DCLF FILE(QADSPOBJ)  
/\* Genera l'elenco librerie \*/  
 DSPOBJD OBJ(QSYS/&LIBGEN) OBJTYPE(\*LIB) +  
 OUTPUT(\*OUTFILE) OUTFILE(QTEMP/CLOUT)  
/\* Reindirizza l'elenco librerie \*/  
 OVRDBF FILE(QADSPOBJ) TOFILE(QTEMP/CLOUT)  
/\* Cicla \*/  
 DOWHILE COND('1')  
/\* Legge dall'elenco librerie \*/  
 RCVF  
/\* Se non ce ne sono altre, abbandona \*/  
 MONMSG MSGID(CPF0864) EXEC(LEAVE)  
/\* Se il testo manca, assume "..." \*/  
 IF COND(&ODOBTX \*EQ ' ') THEN(CHGVAR +  
 VAR(&ODOBTX) VALUE('...'))  
/\* Logga libreria e testo \*/  
 SNDPGMMSG MSG(&ODOBNM \*CAT ' ' \*CAT &ODOBTX)  
/\* Cicla \*/  
 ENDDO  
 ENDPGM**

* 1. Il programma riceve parametri, contiene variabili, dichiara un file di emissione, riempie il file di emissione in libreria temporanea, rilegge il file record per record, per ogni record letto emette un messaggio informativo.
  2. La chiamata del programma con un parametro valido risponde con una lista in joblog.  
     **> call clout ne\*  
      NEERAJ ...  
      NERONI1 Claudio Neroni www.neroni.it  
      NERONI2 Utility di Claudio Neroni  
      NERONI3 ...**
  3. La chiamata del programma con un parametro che provoca una lista vuota risponde con un errore che non viene intercettato e al quale si risponde con ***ENTER***. In fondo grigio i messaggi visibili solo premendo ***F10*** (Include detailed messages) sul video comandi.  
     **> call clout nnn\*  
      No specified objects can be displayed from library QSYS.  
      CPF2124 received by procedure CLOUT. (C D I R)  
     ? C  
      Application error. CPF2124 unmonitored by CLOUT at statement 0000001000,  
      instruction X'0000'.**
  4. A quest’ultimo pasticcio conviene porre rimedio posizionando un  
     **MONMSG CPF0000**subito dopo il DSPOBJD.

1. **Automazione delle decisioni di elaborazione tramite CL.**
   1. Da scrivere.
2. **Utilizzo di programmi di utilità interfacciati da comandi.**
   1. Da scrivere.
3. **Tipi di campi dati.**
   1. In questo paragrafo e nei successivi si spiegano caratteristiche che verranno utilizzate nei paragrafi relativi alla minutazione delle DDS (Data Description Specification).
   2. Campo alfanumerico.
      1. La specie più semplice di campo di dati è quello che prevede una stringa alfanumerica di una lunghezza a piacere fino a 32766 caratteri.  
         Prima riga: carattere. Seconda e terza: esadecimale EBCDIC.  
         **acqua piovana. ACQUA PIOVANA. char  
         889A84989A89844CCDEC4DCDECDC4 zone  
         1384107965151B01384107965151B digit**
   3. Campo numerico zonato (o segnato).
      1. Il campo numerico segnato permette quante cifre desiderate fino ad un massimo di 31 con un massimo di 31 decimali.
      2. Uno zonato di 9 cifre con non importa quanti decimali viene scritto nel record come segue.  
         Prima riga: carattere. Seconda e terza: esadecimale EBCDIC.  
         **123456789 char  
         FFFFFFFFF zone  
         123456789 digit**Se negativo, invece cambia l’ultimo zone.  
         **12345678R char  
         FFFFFFFFD zone  
         123456789 digit**Per inciso, i valori finali di un campo zonato negativo con l’ultima cifra da zero a 9. Zero negativo ha senso solo come cifra finale di un campo con almeno due cifre.  
         **è J K L M N O P Q R char  
         D D D D D D D D D D zone  
         0 1 2 3 4 5 6 7 8 9 digit**
   4. Campo numerico impaccato.
      1. Il campo numerico impaccato permette quante cifre desiderate fino ad un massimo di 31 con un massimo di 31 decimali.
      2. Un impaccato di 31 cifre con non importa quanti decimali viene scritto nel record come segue.  
         Valore numerico del campo  
         **1234567890111222333444555666123**Prima riga: carattere. Seconda e terza: esadecimale EBCDIC.  
          **îÌ[ {íîÃ char  
         1357911233455613 zone  
         246801223445662F digit**Se negativo, invece cambia l’ultimo digit.  
          **îÌ[ {íîÃ char  
         1357911233455613 zone  
         246801223445662D digit**Si noti la non significatività della riga carattere.
      3. A proposito dei campi impaccati, occorre ricordare che per il primo software della famiglia As400 (che girava dal 1979 su macchine Sistem/38 e poi sugli As400 CISC dal 1988 al 1996) era richiesto dai guru IBM che i campi numerici del database fossero impaccati e con un numero di cifre dispari perché questo era il formato che veniva portato in memoria di elaborazione ed usato nel calcolo interno di quel linguaggio macchina.  
         I parvenu dell’As400 non gradiscono l’impaccato che richiede strapazzi di conversione quando il file diventa un flusso da spedire senza tracciato via ftp verso destinatari che non trattano questo tipo di campo EBCDIC senza riscontro nel mondo ASCII.
   5. Campo Data - Campo Ora - Campo TimeStamp.
      1. Di recente, da quando lo spazio su disco costa poco, sono venuti di moda i campi data e simili, ora e timestamp (data-ora-microsecondo).
      2. Si riporta la forma carattere e quella esadecimale EBCDIC dei tre tipi di campi con un esempio di dati.  
         **Data Ora TimeStamp    
         2014-12-20 13.50.10 1949-06-13-12.30.00.000000  
         FFFF6FF6FF FF4FF4FF FFFF6FF6FF6FF4FF4FF4FFFFFF  
         2014012020 13B50B10 1949006013012B30B00B000000**
      3. L’unica fregatura di tali campi è che non prevedono il valore vuoto, cosa che complica un po’ il loro trattamento e il colpo d’occhio che forniscono con il valore di inizio, molto meno leggibile dei blank degli alfanumerici o degli zeri dei numerici.  
         **Data Ora TimeStamp    
         0001-01-01 00.00.00 0001-01-01-00.00.00.000000  
         FFFF6FF6FF FF4FF4FF FFFF6FF6FF6FF4FF4FF4FFFFFF  
         0001001001 00B00B00 0001001001000B00B00B000000**
   6. Campo binario.
      1. Non trattiamo i i campi binari che è difficile trovare nel database per la stessa ragione di scarsa leggibilità degli impaccati.
4. **Altre caratteristiche dei campi dati.**
   1. Nome campo.
      1. Ogni campo viene identificato da un nome unico all’interno del file e, preferibilmente, unico anche all’interno del database.
   2. Intestazioni di colonna e testo.
      1. L’assegnazione delle intestazioni di colonna e del testo per ogni campo migliora la leggibilità del database grazie all’uso che il sistema ne fa nelle utility quali le seguenti.
         1. QUERY per interrogare i file.
         2. SDA per disegnare le videate.
         3. DFU o WRKDBF per gestire estemporaneamente i record.
         4. Display File Field Descriptio e utility derivate che visualizzano i tracciati.
         5. Liste di compilazione dei programmi.
      2. Se il testo non viene definito, viene assunto uguale alle intestazioni di colonna.
   3. Edit dei campi numerici.
      1. Per favorire la leggibilità dei dati numerici è utile definire un codice o una maschera di edizione che si occupa di posizionare i punti di separazione delle migliaia, la virgola decimale, il segno negativo, il simbolo di valuta, eventuali altri simboli di separazione.
      2. Intestazioni di colonna, testo, edit vengono ereditati dai campi di altri file che usano i campi di un file come referenza.
5. **File Fisici.**
   1. Per file fisico si intende un insieme di registrazioni (record) riguardante soggetti diversi ma dotati di informazioni (campi) dello stesso tipo.
   2. Per ciascun campo, dopo l’analisi del contenuto, si assume che occupi un certo numero massimo di caratteri.
   3. I campi dei file fisici contengono un tipo di informazione non complessa esprimibile con un insieme di caratteri alfanumerici. Informazioni più complesse (come documenti, fogli elettronici, disegni, immagini, audio) sono residenti fuori dai file fisici che si limitano a conservarne nei propri campi un indirizzo o un puntatore.
   4. Il record di un un file fisico contiene dunque un insieme di campi organizzati in un tracciato interpretativo uguale per tutti i record del file.
   5. Un file fisico può essere descritto internamente al programma che lo legge o lo scrive ed è così soggetto al rischio di una interpretazione disomogenea da parte di vari programmi in conseguenza soprattutto della evoluzione nel tempo dei programmi e dei tracciati dei file.
6. **Descrizione esterna dei file.**
   1. Per eliminare il rischio di interpretazioni diseguali, la definizione del tracciato di un file è stata posta sin dall’inizio (a.d. 1979) all’interno del file stesso.
   2. Il file viene creato da uno specifico compilatore a partire (come per i programmi) da un omonimo sorgente.
   3. Poi, i programmi che leggono o scrivono il contenuto del file si limitano a dichiarare il nome del file e la compilazione incorpora nel programma il tracciato e il nome dei campi del record.
   4. In RPG (e negli altri linguaggi dell’As400) la riga di definizione del file e del suo uso (nel SEU RPG specifica FX) basta per mettere a disposizione del calcolo tutti i campi dell’ultimo record letto senza usare altre istruzioni oltre a quella che esegue la lettura.
7. **Ordinamento dei file fisici.**
   1. In sequenza di arrivo.
      1. I record che vengono scritti in un file fisico dai programmi vengono aggiunti progressivamente in coda ai precedenti nello spazio su disco riservato al file.
      2. I programmi che leggono il file fisico hanno sempre a disposizione i record nello stesso ordine in cui sono stati scritti.  
         Per elaborare il file in sequenza d’arrivo, un RPG dichiarerà il file Input Primario.
      3. Lungo il file fisico, ogni record può essere recuperato anche in base alla posizione relativa.  
         Per elaborare il file per numero relativo di record, un RPG dichiarerà il file Input FulProcedural senza chiavi.
      4. Per minimizzare lo spreco di spazio, quando si cancellano i record dal fisico, si può richiedere che i record aggiunti in seguito occupino lo spazio vuoto ma ancora occupato dei record cancellati.  
         Chiaramente va a farsi friggere l’ordine di arrivo.
   2. Con chiavi.
      1. Pur essendo possibile, è sconsigliabile che un file fisico sia dotato di chiavi.  
         Ad esempio, la copia di un file fisico eseguita per cautela prima di una elaborazione, è velocissima se il file fisico è senza chiavi; molto più lenta se il file originale è dotato di chiavi perché anche la copia sarà dotata di inutili chiavi.  
         Lo stesso dicasi per salvataggi e ripristini del fisico.
8. **File Logici.**
   1. Un file logico non contiene dati propri ma indirizza i record di un fisico.
   2. Raramente, per ragioni di sicurezza, il logico può utilizzare un formato ridotto rispetto al fisico, cioè vedere meno campi.
   3. Più frequentemente il logico può vedere meno record di quelli presenti nel fisico di base.
   4. Normalmente la prima chiave logica costruita sul fisico vede tutti i record del fisico e usa una chiave tecnica o applicativa che possa essere considerata unica nel file.  
      Questo permette di usare la chiave per recuperare il record che la riporta.
   5. Se si richiede che il logico dichiari esplicitamente che la chiave è unica (clausola UNIQUE), non ci possono essere nel file due record con la stessa chiave. La scrittura di un record con chiave duplicata viene rifiutata non dal programma che la tenta ma direttamente dal supporto di database.
   6. La chiave unica è indispensabile per gestire facilmente i record in update.
   7. Tuttavia le chiavi uniche sono pesanti per il sistema perché, per ogni record scritto o sovrascritto nel file fisico di base, il sistema deve controllare l’osservanza della clausola di univocità nei logici sovrastanti che la richiedono.  
      Il controllo è sincrono e ritarda l’operazione di scrittura o sovrascrittura dei record fino al consenso da tutti i file con clausola di univocità.
   8. E’ inoltre rischioso costruire logici con chiavi a ordinamento indifferente ovvero non uniche.  
      Può succedere che l’ordine di elaborazione di un run differisca dall’ordine di elaborazione in un successivo run.  
      Per evitare il rischio si utilizzano le chiavi virtualmente uniche.
   9. I file logici con chiavi virtualmente uniche non contengono la clausola di univocità ma la chiave, composta da più campi, comprende anche tutti i campi che figurano in una chiave unica, possibilmente quella del primo logico che dichiara la clausola UNIQUE.  
      Alla disperata i campi si aggiungono in coda alla chiave in fase di definizione.
   10. Il gruppo di logici sopra un fisico deve essere il più ridotto possibile non solo per problemi di dimensioni delle chiavi ma soprattutto per il peso elaborativo che significa lentezza nei tempi di risposta dei programmi.
   11. Prima di creare un nuovo logico sopra un fisico, occorre sempre valutarne l’opportunità considerando quelli già esistenti ad esempio tramite l’utility JFDR.
9. **File Logici Combinati.**
   1. Il file combinato vede più file fisici sequenziandoli secondo una chiave condivisa, cioè grazie ad un gruppo di campi con le stesse caratteristiche presenti in tutti i file fisici che vengono visti dal logico.
   2. La lettura dall’inizio alla fine presenta tutti i record in sequenza di chiave a prescindere da quale file provengano.
   3. I campi chiave devono essere presenti in tutti i file fisici ma possono mancare quelli in coda, sostituiti da \*NONE (come se il campo ci fosse ma con valore a esadecimali x’00’).
   4. Da usare solo con utility dfu.
   5. Ad esempio, sequenzia testate e righe di un documento.
10. **File Logici Join.**
    1. Il file join lega i file un file primario a dei file secondari.
    2. Il legame del primario con il secondario consiste in una chiave presente sia sul primario che sul secondario.
    3. Ogni file secondario del join può avere una chiave di aggancio diversa dagli altri.
    4. La lettura di un record primario legge automaticamente anche un solo record secondario che soddisfa la chiave di aggancio.
    5. Da usare solo con utility query e dfu.
    6. Ad esempio, collega un file di codici alle sue decodifiche.
    7. Ad esempio, collega le righe di un documento alle testate.
    8. Solo i campi del primario possono fungere da chiavi del fole join.
11. **Definizione di fisici e logici mediante SQL.**
    1. Fisici e logici possono essere definiti anche tramite una sessione interattiva di **STRSQL** (Start SQL) ma non esiste un modo univoco di ottenere un sorgente che possa essere rieseguito per ottenere la creazione del database come quello ottenuto nella sessione **STRSQL**.
    2. In particolare in **STRSQL** è disponibile una composizione assistita o prompter dei comandi SQL.
    3. Quando invece si trascrivono i comandi dalla sessione **STRSQL** ad un sorgente, non è più possibile riprendere i comandi memorizzati nel sorgente per modificarli immediatamente in un prompter.
    4. Tuttavia possiamo tracciare una strada per comporre e mantenere un sorgente SQL rieseguibile.
    5. In uscita dalla sessione **STRSQL** si scaricano i comandi eseguiti nella sessione direttamente in un file sorgente di comodo.

**Session was saved and started again.  
STRSQL parameters were ignored.   
Current connection is to relational database PUB1.   
drop TABLE NERONI1/TBL1  
Drop of TBL1 in NERONI1 complete.   
CREATE TABLE NERONI1/TBL1 (DOCUMENTO FOR COLUMN DOC1 DEC ( 5, 0)   
NOT NULL WITH DEFAULT, RIGA FOR COLUMN RIG1 DEC ( 3, 0) NOT NULL  
WITH DEFAULT, DESCRIZIONE FOR COLUMN DES1 CHAR ( 50) NOT NULL WITH  
DEFAULT)   
Table TBL1 in NERONI1 created but could not be journaled.   
CREATE UNIQUE INDEX NERONI1/tbl11 ON NERONI1/TBL1 (DOCUMENTO, RIGA)  
Index TBL11 created in NERONI1 on table TBL1 in NERONI1.  
Session was saved and started again.  
STRSQL parameters were ignored.  
Current connection is to relational database PUB1.**

Nel sorgente emesso si trovano sia i comandi eseguiti che i messaggi di log.

* 1. Si copiano solo i comandi SQL in un sorgente di creazione.

**drop TABLE NERONI1/TBL1  
CREATE TABLE NERONI1/TBL1 (DOCUMENTO FOR COLUMN DOC1 DEC ( 5, 0)   
NOT NULL WITH DEFAULT, RIGA FOR COLUMN RIG1 DEC ( 3, 0) NOT NULL  
WITH DEFAULT, DESCRIZIONE FOR COLUMN DES1 CHAR ( 50) NOT NULL WITH  
DEFAULT)   
CREATE UNIQUE INDEX NERONI1/tbl11 ON NERONI1/TBL1 (DOCUMENTO, RIGA)**

* 1. Si aggiungono i comandi **BCHJOB** e **ENDBCHJOB** per l’esecuzione batch.  
     Si imposta la lista librerie.  
     Si imposta la libreria in cui creare i file come current library.  
     Si esegue la stringa SQL con il comando **RUNSQLSTM**.  
     Si antepone al gruppo dei comandi SQL il comando **DATA** e si pospone il tag **//ENDSRC**.  
     Si chiude ogni comando SQL con “;” (punto e virgola).  
     Si eliminano dai comandi SQL i nomi librerie trasformando i nomi di file qualificati in nomi semplici. Come libreria di creazione i comandi SQL assumono la current library, già impostata.  
     Si aggiustano i comandi SQL per aumentarne la leggibilità.

**//BCHJOB JOB(DB5) JOBD(NERONI2/NERONI2) OUTQ(QPRINT) ENDSEV(60) +  
 LOG(4 00 \*SECLVL) MSGQ(\*USRPRF)  
/\* Imposta la lista librerie. \*/  
CHGLIBL LIBL(QTEMP NERONI2 QGPL)  
/\* Imposta la libreria corrente per crearvi il database. \*/  
CHGCURLIB CURLIB(NERONI1)  
/\* Esegue il sorgente SQL scritto nella stringa. \*/  
RUNSQLSTM SRCFILE(FIRST) SRCMBR(FIRST) ERRLVL(20)  
/\*---------------------------------------------------------------------\*/  
//DATA FILE(FIRST) FILETYPE(\*SRC) ENDCHAR('//ENDSRC')  
DROP TABLE TBL5;  
CREATE TABLE TBL5 (  
 DOCUMENTO FOR COLUMN DOC1 DEC ( 5, 0) NOT NULL WITH DEFAULT,  
 RIGA FOR COLUMN RIG1 DEC ( 3, 0) NOT NULL WITH DEFAULT,  
 DESCRIZIONE FOR COLUMN DES1 CHAR ( 50 ) NOT NULL WITH DEFAULT  
);  
CREATE UNIQUE INDEX TBL51  
 ON TBL5 (DOCUMENTO, RIGA);  
//ENDSRC  
/\*---------------------------------------------------------------------\*/  
//ENDBCHJOB**

* 1. Si noti il parametro **ERRLVL(20)** nel comando **RUNSQLSTM** per superare l’eventuale errore di file inesistente rilevato dal comando SQL **DROP**.
  2. La stringa di lancio di un job batch che si è ottenuta si sottomette all’esecuzione tramite il comando **SBMDBJOB**.
  3. A fine esecuzione si esamina il log SQL e la joblog del lavoro batch.
  4. In tutto questo la fregatura sta nel fatto che i comandi SQL non sono più prompterizzabili. Per farlo, copiaincollarli uno alla volta in una sessione SQL interattiva e riportali nel sorgente con un copiaincolla a rovescio.
  5. C’è da chiedersi chi è la bestia che ha ucciso la logica solare del sorgente che genera un oggetto in maniera semplice e nativa.
  6. Qualche furbo aggira il problema libreria con **CREATE SCHEMA** ma non è coerente con la creazione del database una botta alla volta.
  7. Questa tecnica di sorgentizzazione permette perlomeno di memorizzare SQL in modo prossimo al solito: creazione di sorgenti trasportabili, esecuzione di attività ripetibili di generazione, oggetti utilizzabili.
  8. Si osserva che molte strutture non sono costruibili con le sole DDS e che è quindi necessaria una gestione di sorgenti per quegli oggetti di database ottenibili solo con istruzioni SQL (o altro genere di comandi). Ad esempio, per viste logiche di tipo join con chiavi su file secondari o per vincoli CONSTRAIN o per l’aggiunta di trigger.
  9. L’espediente illustrato permette anche, con poche variant i, di memorizzare in sorgenti quegli oggetti che non hanno sorgente nativo ma solo un insieme di comandi di creazione. Ad esempio i file messaggi con il contenuto di definizioni di messaggi o, ancora, i sottosistemi.
  10. Una versione più complessa, collaudata sul release V7R1M0.  
      In blu le righe non sopportate da V5R3M0.

**//BCHJOB JOB(DB5B) JOBD(NERONI2/NERONI2) OUTQ(QPRINT) ENDSEV(60) +  
 LOG(4 00 \*SECLVL) MSGQ(\*USRPRF)  
/\* Imposta la lista librerie. \*/  
CHGLIBL LIBL(QTEMP NERONI2 QGPL)  
/\* Imposta la libreria corrente per crearvi il database. \*/  
CHGCURLIB CURLIB(UFNERONI)  
/\* Esegue il sorgente SQL scritto nella stringa. \*/  
RUNSQLSTM SRCFILE(FIRST) SRCMBR(FIRST) ERRLVL(20)  
/\*---------------------------------------------------------------------\*/  
//DATA FILE(FIRST) FILETYPE(\*SRC) ENDCHAR('//ENDSRC')  
DROP TABLE TBL5  
;  
CREATE TABLE TBL5  
 ( DOC1 DEC ( 5, 0) NOT NULL WITH DEFAULT,  
 RIG1 DEC ( 3, 0) NOT NULL WITH DEFAULT,  
 DES1 CHAR ( 50 ) NOT NULL WITH DEFAULT)  
 RCDFMT TBL5R  
;  
LABEL ON TABLE TBL5  
 IS 'Riga documento'  
LABEL ON COLUMN TBL5  
 ( DOC1 TEXT IS 'Numero documento' ,  
 RIG1 TEXT IS 'Numero riga' ,  
 DES1 TEXT IS 'Descrizione riga' )  
;  
LABEL ON COLUMN TBL5  
 ( DOC1 IS 'Documento' ,  
 RIG1 IS 'Riga' ,  
 DES1 IS 'Descrizione' )  
;  
CREATE UNIQUE INDEX TBL51  
 ON TBL5 (DOC1, RIG1);  
;  
LABEL ON INDEX TBL51  
 IS 'Riga documento unique doc-rig'  
;  
CREATE INDEX TBL52  
 ON TBL5 (DES1, DOC1, RIG1)  
;  
LABEL ON INDEX TBL52  
 IS 'Riga documento per des-doc-rig'  
;  
//ENDSRC  
/\*---------------------------------------------------------------------\*/  
//ENDBCHJOB**

* 1. Dulcis in fundo, su V7R1M0 ho anche riscontrato che l’interfaccia prompter di STRSQL non sopporta le stesse specifiche rifiutate dal V5R3M0. Misterium fidei.
  2. Per quanto mi riguarda, non userò SQL per il tracciato dei fisici: troppo faticoso.  
     Indispensabile invece per usufruire di quello che solo SQL fa.

1. **File XML.**
   1. Da scrivere.
2. **Librerie, file, membri, file dati.**
   1. Da scrivere.
3. **Utilizzo del PDM (Programming Development Manager).**
   1. Da scrivere.
4. **Utilizzo del SEU (Source Entry Utility).**
   1. Il SEU è il programma di sistema che permette di gestire il contenuto dei membri sorgente.
   2. Il tracciato di un sorgente è spiegato a .
   3. La chiamata del SEU avviene tramite il comando **STRSEU** ma di solito tale comando è intermediato dall’opzione 2=Edit sull’elenco membri di un file sorgente visualizzato dal PDM.
   4. L’elenco membri PDM è effettivamente il miglior visualizzatore per la ricerca di un membro tramite nome o nome parziale ma anche tramite testo o testo parziale.

**Work with Members Using PDM PUB1   
   
 File . . . . . . CORSO        
 Library . . . . NERONI1    Position to . . . . .              
   
 Type options, press Enter.   
 2=Edit 3=Copy 4=Delete 5=Display 6=Print 7=Rename   
 8=Display description 9=Save 13=Change text 14=Compile 15=Create module...  
   
 Opt Member Type Text   
    CLOUT CLLE       Pgm con outfile                                     
    CLS CL         Stringa CL                                          
    CLVAR CLLE       Pgm con variabili CL                                
    PF1 PF         Fisico uno                                          
   
 Bottom   
 Parameters or command   
 ===>                                                                             
 F3=Exit F4=Prompt F5=Refresh F6=Create   
 F9=Retrieve F10=Command entry F23=More options F24=More keys**

* 1. Per illustrare il comportamento del SEU
  2. Entriamo nell’aggiunta di un nuovo membro con***F6***.

**Start Source Entry Utility (STRSEU)   
   
 Type choices, press Enter.   
   
 Source file . . . . . . . . . . SRCFILE > CORSO   
 Library . . . . . . . . . . . > NERONI1   
 Source member . . . . . . . . . SRCMBR \*PRV         
 Source type . . . . . . . . . . TYPE \*SAME        
 Text 'description' . . . . . . . TEXT \*BLANK                           
                      
   
 Bottom   
 F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display   
 F24=More keys**

* 1. Completiamo il prompt dello **STRSEU** con   
     **SRCMBR(PRTOPT)  
     TYPE(RPGLE)  
     TEXT(‘Stampa opzioni PDM’)**e premiamo invio.
  2. Prima di proseguire, occorre annotare che il membro sorgente viene aggiunto e allocato dalla sessione interattiva ma non viene scritto fino all’uscita dal SEU e solo se si conferma la scrittura.  
     Per tutta la durata dell’attività, il SEU scrive in un’area di lavoro interna senza rischi di perdita in caso di caduta del lavoro o del sistema.  
     In tali circostanze, l’area di lavoro si ripropone a chi riapre il membro a caduta finita.  
     Le stesse considerazioni valgono anche durante le successive attività di modifica del membro sorgente.
  3. Torniamo a descrivere l’esito dello start seu.  
     Si presenta il video di immissione a tracciato libero del SEU.  
     Di passaggio, si osservi la terza riga che riporta intestazioni applicabili alla specifica H (inizia con **FMT H**)  
     Non essendo però il video adatto all’immissione da parte di un principiante, rifiutiamo subito il tracciato libero con un invio.

**Columns . . . : 6 100 Edit NERONI1/CORSO**

**SEU==>                                                             PRTOPT**

**FMT H HKeywords++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++Comments++++++++++++**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Beginning of data \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**'''''''**

**'''''''**

**'''''''**

**'''''''**

**'''''''**

**'''''''**

**'''''''**

**'''''''**

**'''''''**

**'''''''**

**'''''''**

**'''''''**

**'''''''**

**'''''''**

**'''''''**

**'''''''**

**'''''''**

**'''''''**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of data \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**F3=Exit F4=Prompt F5=Refresh F9=Retrieve F10=Cursor F11=Toggle**

**F16=Repeat find F17=Repeat change F24=More keys**

**Member PRTOPT added to file NERONI1/CORSO. +**

* 1. Si chiudono le righe a immissione libera e sulla riga di servizio digitiamo la richiesta IPFX (Insert Prompt File eXternal described).

**Columns . . . : 6 100 Edit NERONI1/CORSO**

**SEU==>                                                             PRTOPT**

**FMT H HKeywords++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++Comments++++++++++++**

**IPFX \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Beginning of data \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of data \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**F3=Exit F4=Prompt F5=Refresh F9=Retrieve F10=Cursor F11=Toggle**

**F16=Repeat find F17=Repeat change F24=More keys**

* 1. Si presenta il prompt della specifica di utilizzo di un file descritto esternamente.

**Columns . . . : 6 100 Edit NERONI1/CORSO**

**SEU==>                                                              PRTOPT**

**FMT H HKeywords++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++Comments++++++++++++**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Beginning of data \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**'''''''**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of data \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Prompt type . . . FX Sequence number . . . '''''''**

**File File End of File**

**Filename Type Designation File Addition Sequence**

**File Limits Record**

**Format Processing Address Type Device**

**Keywords Comment**

**F3=Exit F4=Prompt F5=Refresh F11=Previous record**

**F12=Cancel F23=Select prompt F24=More keys**

* 1. Completiamo il prompt FX con i valori specificati nella tabella seguente e premiamo invio.

|  |  |  |
| --- | --- | --- |
| **Intestazione campo** | **Valore** | **Significato del valore** |
| **Filename** | **QAUOOPT** | **Nome del file fisico opzioni PDM** |
| **File Type** | **I** | **Input** |
| **File Designation** | **F** | **Full procedural** |
| **File Format** | **E** | **External described** |
| **Device** | **DISK** | **Disk** |

* 1. Se la specifica non presenta errori formali, il SEU la scrive nell’area di lavoro e sul video nella zona dati. Ripresenta inoltre il prompt FX che non ci serve più e che eliminiamo con un invio senza digitazione.  
     Si osservi la terza riga che riporta ora le intestazioni applicabili alla specifica FX (inizia con **FMT FX**).

**Columns . . . : 6 100 Edit NERONI1/CORSO**

**SEU==>                                                              PRTOPT**

**FMT FX FFilename++IPEASF.....L.....A.Device+.Keywords+++++++++++++++++++++++++++++Comments++++++++++++**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Beginning of data \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**0001.00 FQAUOOPT IF E DISK 141227**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of data \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**F3=Exit F4=Prompt F5=Refresh F9=Retrieve F10=Cursor F11=Toggle**

**F16=Repeat find F17=Repeat change F24=More keys**

* 1. Sulla riga immediatamente precedente alla posizione in cui vogliamo inserire la prossima specifica, digitiamo la richiesta **IPF** (Insert Prompt File internal described). La richiesta va digitata in sostituzione delle cifre del numero di sequenza. Oppure a ricalco delle stesse, preferibilmente partendo dalla prima posizione e aggiungendo un blank in coda.

**Columns . . . : 6 100 Edit NERONI1/CORSO**

**SEU==>                                                              PRTOPT**

**FMT FX FFilename++IPEASF.....L.....A.Device+.Keywords+++++++++++++++++++++++++++++Comments++++++++++++**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Beginning of data \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**IPF .00 FQAUOOPT IF E DISK 141227**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of data \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**F3=Exit F4=Prompt F5=Refresh F9=Retrieve F10=Cursor F11=Toggle**

**F16=Repeat find F17=Repeat change F24=More keys**

* 1. Sulla riga immediatamente precedente alla posizione in cui vogliamo inserire la prossima specifica, digitiamo la richiesta IPF (Insert Prompt File internal described). La richiesta va digitata in sostituzione delle cifre del numero di sequenza. Oppure a ricalco delle stesse, preferibilmente partendo dalla prima posizione e aggiungendo un blank in coda.

**Columns . . . : 6 100 Edit NERONI1/CORSO**

**SEU==>                                                              PRTOPT**

**FMT FX FFilename++IPEASF.....L.....A.Device+.Keywords+++++++++++++++++++++++++++++Comments++++++++++++**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Beginning of data \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**0001.00 FQAUOOPT IF E DISK 141227**

**'''''''**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of data \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Prompt type . . . F Sequence number . . . '''''''**

**File File End of File**

**Filename Type Designation File Addition Sequence**

**\_\_\_\_\_\_\_\_\_\_ \_ \_ \_ \_ \_**

**File Record Limits Length of Record**

**Format Length Processing Key Field Address Type**

**\_ \_\_\_\_\_ \_ \_\_\_\_\_ \_**

**File**

**Organization Device Keywords**

**\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Comment**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**F3=Exit F4=Prompt F5=Refresh F11=Previous record**

**F12=Cancel F23=Select prompt F24=More keys**

* 1. Completiamo il prompt F con i valori specificati nella tabella seguente e premiamo invio.

|  |  |  |
| --- | --- | --- |
| **Intestazione campo** | **Valore** | **Significato del valore** |
| **Filename** | **QSYSPRT** | **Nome del file di stampa** |
| **File Type** | **O** | **Output** |
| **File Format** | **F** | **Internal described** |
| **Record Length** | **132** | **Lunghezza della riga di stampa** |
| **Device** | **PRINTER** | **Stampante** |

* 1. Stessa procedura per la specifica di calcolo. La richiesta è IPC (Insert Prompt Calculation).

**Columns . . . : 6 100 Edit NERONI1/CORSO**

**SEU==>                                                              PRTOPT**

**FMT FX FFilename++IPEASF.....L.....A.Device+.Keywords+++++++++++++++++++++++++++++Comments++++++++++++**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Beginning of data \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**0001.00 FQAUOOPT IF E DISK 141227**

**0002.00 FQSYSPRT O F 132 PRINTER 141227**

**'''''''**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of data \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Prompt type . . . C Sequence number . . . '''''''**

**Level N01 Factor 1 Operation Factor 2 Result**

**\_\_ \_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ DO\_\_\_\_\_\_\_\_ \*HIVAL\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Decimal**

**Length Positions HI LO EQ Comment**

**\_\_\_\_\_ \_\_ \_\_ \_\_ \_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**F3=Exit F4=Prompt F5=Refresh F11=Previous record**

**F12=Cancel F23=Select prompt F24=More keys**

* 1. Digitando il calcolo è normale utilizzare il nuovo prompt emesso dal SEU ad ogni invio perché in un programma RPG sono normalmente molte molte le specifiche consecutive dello stesso tipo.
  2. Tuttavia è raro digitare di seguito tutte le specifiche di calcolo.  
     Ad esempio dopo un **DO** di solito si digita nella successiva **ENDDO** perché è buona norma digitare subito la chiusura di un ciclo per non dimenticarsene.  
     Si provvede poi ad inserire le altre istruzioni necessarie tra il **DO** e l’**ENDDO**.  
     Stessa cosa tra **IF** ed **ENDIF** e molte altre coppie ancora.
  3. Per inserire dopo una specifica un prompt dello stesso formato basta digitare sopra il numero di sequenza semplicemente **IP** (**I**nsert **P**rompt) e procedere come già visto.
  4. Se si vuole inserire un prompt diverso, invece di **IP** si digiterà **IPxx**, intendendo con **xx** l’identificativo del formato desiderato.
  5. Il programma RPGLE di nome PRTOPT contiene infine le seguenti righe (in rosso). In verde le intestazioni visualizzate a richiesta dal SEU ma non incorporate dal sorgente.  
     Il tracciato **O** è la specifica di riga di output descritta internamente.  
     Il tracciato **P** è la specifica di campo di output descritta internamente.

**.FX..FFilename++IPEASF.....L.....A.Device+.Keywords+++++++++++++++++++++++++++++Comments++++++++++++  
 FQAUOOPT IF E DISK  
.F...FFilename++IPEASFRlen+LKlen+AIDevice+.Keywords+++++++++++++++++++++++++++++Comments++++++++++++  
 FQSYSPRT O F 132 PRINTER  
.C...CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq....Comments++++++++++++  
 C SETON LR  
 C DO \*HIVAL  
 C READ QAUOOPT  
 C IF %EOF  
 C LEAVE  
 C ENDIF  
 C ADD 1 CNT 4 0  
 C MOVEL COMMAND COMMAND100 100  
 C EXCEPT RIGA  
 C ENDDO  
.O...OFilename++DF..N01N02N03Excnam++++B++A++Sb+Sa+.............................Comments++++++++++++  
 OQSYSPRT E RIGA  
.P...O..............N01N02N03Field+++++++++YB.End++PConstant/editword/DTformat++Comments++++++++++++  
 O CNT K  
 O OPTION + 1  
 O COMMAND100 + 1**

* 1. Qui di seguito **Ѣ** sta per **blank**.
  2. E’ possibile copiare specifiche singole in un altro punto del sorgente.  
     Sopra il numero di sequenza della specifica da copiare si digita **CѢ** (Copy).  
     Sopra il numero di sequenza della specifica che precede il punto di arrivo si digita **AѢ** (After).  
     Si preme invio.  
     Invece di **AѢ** (After) si può digitare **BѢ** (Before) per fare atterrare le specifiche copiate prima e non dopo la specifica di destinazione.  
     Se la copiatura riguarda un gruppo di specifiche, si contrassegnano la prima e l’ultima con **CCѢ** (Copy Beg/End).  
     Se si vogliono muovere e non copiare le specifiche, invece di **CѢ** (Copy) si usa **MѢ** (Move) e invece di **CCѢ** (Copy Beg/End) si usa **MMѢ** (Move Beg/End).
  3. Per avere l’elenco delle numerose opzioni digitabili sopra il numero di sequenza, portare il cursore sopra la sequenza e premere ***F1*** (Help).  
     Qui l’elenco sintetico.
* **A=After**
* **An=After with repeat n times**
* **B=Before**
* **Bn=Before with repeat n times**
* **C=Copy a line**
* **Cn=Copy n lines**
* **CC=Block Copy**
* **CR=Copy records and retain command**
* **CRn=Copy n records and retain command**
* **CCR=Copy block records and retain command**
* **COLS=Column**
* **D=Delete a line**
* **Dn=Delete n lines**
* **DD=Block Delete**
* **F=Display format line**
* **Fxx=Display the xx format line**
* **F?=Show the Select Format display**
* **I=Insert a line**
* **In=Insert n lines**
* **IF=Insert line and display format**
* **IFn=Insert n lines and display format**
* **IFxx=Insert a blank line and display xx format**
* **IFxxn=Insert n blank lines and display xx format**
* **IF?=Show the Select Format display**
* **IF?n=Show the Select Format display and insert lines**
* **IP=Insert line and prompt**
* **IPxx=Insert line with an xx prompt**
* **IP?=Show Select Prompt Display**
* **IS=Insert skeleton line**
* **ISn=Insert n skeleton lines**
* **L=Shift data 1 char left**
* **Ln=Shift data n chars left**
* **LL=Shift block 1 char to the left**
* **LLn=Shift block n chars to the left**
* **LP=Line Print a line**
* **LPn=Line Print n lines**
* **LLP=Block Line Print**
* **LT=Shift data 1 char left with truncate**
* **LTn=Shift data n chars left with truncate**
* **LLT=Shift block left with truncate**
* **LLTn=Shift block n chars left with truncate**
* **M=Move a line**
* **Mn=Move n lines**
* **MM=Block Move**
* **O=Overlay**
* **On=Overlay n lines**
* **OO=Block overlay**
* **P=Display prompt**
* **Pxx=Display xx prompt**
* **P?=Show the Select Prompt display**
* **R=Shift data 1 char right**
* **Rn=Shift data n chars right**
* **RR=Block shift 1 char to the right**
* **RRn=Block shift n chars to the right**
* **RRT=Shift block right with truncate**
* **RRTn=Shift block n chars right with truncate**
* **RT=Shift data 1 char right with truncate**
* **RTn=Shift data n chars right with truncate**
* **RP=Repeat line**
* **RPn=Repeat line n times**
* **RPP=Block repeat**
* **RPPn=Block repeat n times**
* **S=Skeleton line**
* **SF=Show first record**
* **SFn=Show first n records**
* **SL=Show last record**
* **SLn=Show last n records**
* **TABS=Display tabs record**
* **W=Display member from column 1**
* **Wn=Display member from column n**
* **X=Exclude a line**
* **Xn=Exclude n lines**
* **XX=Block Exclude**
* **+ (plus sign)=Roll member forward 1 line**
* **+n=Roll member forward n lines**
* **- (minus sign)=Roll member backward 1 line**
* **-n=Roll member backward n lines**
* **n=Absolute Positioning**
  1. Durante la gestione di un membro sorgente, è possibile visualizzarne un secondo nella parte bassa del video. Premendo F15 (Browse/Copy options) si ottiene il video sul quale scegliere un membro riempiendo i campi Browse/copy member.

**Browse/Copy Options**

**Type choices, press Enter.**

**Selection . . . . . . . . . . . 1 1=Member**

**2=Spool file**

**3=Output queue**

**Copy all records . . . . . . . N Y=Yes, N=No**

**Browse/copy member . . . . . . PRTOPT Name, F4 for list**

**File . . . . . . . . . . . . CORSO Name, F4 for list**

**Library . . . . . . . . . . NERONI1 Name, \*CURLIB, \*LIBL**

**Browse/copy spool file . . . . PRTOPT Name, F4 for list**

**Job . . . . . . . . . . . . . PRTOPT Name**

**User . . . . . . . . . . . NERONI Name, F4 for list**

**Job number . . . . . . . . \*LAST Number, \*LAST**

**Spool number . . . . . . . . \*LAST Number, \*LAST, \*ONLY**

**Display output queue . . . . . QPRINT Name, \*ALL**

**Library . . . . . . . . . . . \*LIBL Name, \*CURLIB, \*LIBL**

**F3=Exit F4=Prompt F5=Refresh F12=Cancel**

**F13=Change session defaults F14=Find/Change options**

* 1. Il video si spezza in due per visualizzare il membro scelto. Anche il secondo membro può ora essere fonte di copiature.

**Columns . . . : 6 100 Edit NERONI1/CORSO**

**SEU==>                                                              PRTOPT**

**FMT FX FFilename++IPEASF.....L.....A.Device+.Keywords+++++++++++++++++++++++++++++Comments++++++++++++**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Beginning of data \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**0001.00 FQAUOOPT IF E DISK 141227**

**0002.00 FQSYSPRT O F 132 PRINTER 141227**

**0003.00 C SETON LR 141227**

**0004.00 C DO \*HIVAL 141227**

**0005.00 C READ QAUOOPT 141227**

**0006.00 C IF %EOF 141227**

**0007.00 C LEAVE 141227**

**0008.00 C ENDIF 141227**

**------------------------------------------------------------------------------------------------------------------------------------**

**Columns . . . : 1 100 Browse NERONI1/CORSO**

**SEU==>                                                              CLOUT**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Beginning of data \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**0001.00 /\* Claudio Neroni 14-12-2014 Creato \*/ 141214**

**0002.00 /\* Programma CL con outfile \*/ 141214**

**0003.00 141214**

**0004.00 PGM PARM(&LIBGEN) 141214**

**0005.00 /\* Riceve un nome generico \*/ 141214**

**0006.00 DCL VAR(&LIBGEN) TYPE(\*CHAR) LEN(10) 141214**

**0007.00 /\* Elenco librerie \*/ 141214**

**F3=Exit F4=Prompt F5=Refresh F9=Retrieve F11=Toggle F12=Cancel**

**F16=Repeat find F17=Repeat change F24=More keys**

* 1. A partire dal video **Browse/Copy options**, riempiendo i campi **Browse/copy spool file** e aiutandosi con ***F4 for list***, si ottiene lo spool file di compilazione del sorgente.  
     Sul video split che visualizza nella parte bassa lo spoo, digitando **\*ERR** nel campo di ricerca basso e premendo ***F16*** (Repeat find) si esplorano gli errori elencati nello spool.

1. **Utilizzo di RDI (Rational Developer for i).**
   1. E' il SEU moderno.
2. **Creazione di oggetti dei vari tipi a partire da sorgenti (compilazioni).**
   1. Da scrivere.
3. **Nomenclature.**
   1. Per costruire una applicazione occorre adottare una serie di regole per l’assegnazione di ogni specie di nomi.

* Programmi CLLE
* Programmi RPGLE
* File fisici
* File logici
* File display
* File printer
* Aree dati
* Campi di file fisici
* Campi di file display
* Campi di file printer
* Campi di lavoro nei programmi
* …
  1. Per gli oggetti dotati sia di sorgente che di oggetto è imperativo che si mantengano rigorosamente uguali i due nomi sorgente-oggetto. Non lasciarsi mai tentare dall’opzione di compilazione che permette la dissociazione.
  2. I nomi migliori per i programmi di qualunque genere (più le aree dati, più i file di lavoro in libreria temporanea) sono quelli che prevedono un prefisso di fantasia a significare l’appartenenza ad una procedura complessa e una desinenza che distingue la funzione svolta dal programma. La lunghezza massima del nome è bene non superi 8 caratteri per lasciare la possibilità di aggiungere al nome un paio di caratteri per distinguere le versioni obsolete che si conservano per cautela, almeno in forma sorgente.
  3. Per i file di database corre l’uso di assegnare un prefisso di fantasia di 5 caratteri ad un fisico e ai logici che lo usano. Di solito la desinenza di un fisico è 00F e quella dei suoi logici 01L, 02L, 03L e via cantando.
  4. I nomi dei campi di file fisico sono solitamente corti per l’antico limite RPG di 6 caratteri, ora salito a 10 nell’RPGLE. Si compongono di solito di un prefisso di 2 caratteri comune a tutti i campi dello stesso fisico, possibilmente diverso da tutti gli altri dello stesso pacchetto applicativo. Se si sceglie di limitarsi ancora a 6, i 4 rimanenti si costruiscono come abbreviazione delle due parole che descrivono il campo: 2 tipo record + 2 parola uno + 2 parola due.
  5. Nei display file si usano i nomi del database ma con un prefisso di 2 che identifica il formato record all’interno del display file. Ad esempio: W0 per il video guida, W1 per il video dati, W3 per il subfile (elenco). I nomi risultano quindi di 8 caratteri.
  6. Per i printer file, stessa regola dei display file ma con prefisso diverso: P1, P2,…
  7. Tuttavia non è importante quali regole si scelgono purché permettano di riconoscere facilmente i gruppi di programmi che compongono una procedura, i campi che fanno parte di uno stesso record, i logici di un fisico,…
  8. Per i campi di lavoro del programma che derivano dal database non bisogna fare sforzi di fantasia ma derivare anche il nome dai campi a cui fanno riferimento. Il meglio è poter scrivere il calcolo componendo il nome dei campi di copia, di salvataggio, di comodo, del video, della stampa tenendo conto solo di poche regole di derivazione dal database senza vagare da un capo all’altro del programma.
  9. Altra necessità fondamentale che induce ad usare regole facili è restare leggibili da parte dei colleghi, anche quelli che leggeranno i programmi in futuro senza spiegazioni giustificative delle stranezze.
  10. Se nell’ambiente in cui operate sono già consolidate regole utilizzabili, non inventate l’acqua calda e adeguatevi.

1. **Scrittura di DDS (Data Description Specification) di file fisico.**
   1. Da scrivere.
2. **Scrittura di DDS di file logici con uso di chiavi uniche o virtualmente uniche.**
   1. Da scrivere.
3. **Strutturazione di un database adatto a recepire documenti (testata, righe, sottorighe, commenti).**
   1. Da scrivere.
4. **Gestione dati con DFU (Data File Utility).**
   1. Da scrivere.
5. **Lettura, analisi, trascrizione ed elaborazione dati con Query e SQL (Structured Query Language).**
   1. Da scrivere.
6. **Concetti e gestione di ambiente di sviluppo, ambiente di test e ambiente di produzione.**
   1. Da scrivere.
7. **Travaso di dati via FTP (File Transfer Protocol) tra PC e As400 e tra due As400.**
   1. Da scrivere.
8. **Comandi e stampe di analisi della struttura del database.**
   1. Da scrivere.
9. **Scrittura di un programma RPG ILE in formato posizionale e in formato free.**
   1. Da scrivere.
10. **Utilizzo di RPG ante ILE.**
    1. Da scrivere.
11. **Principali istruzioni RPG di lettura e scrittura del database.**
    1. Da scrivere.
12. **Principali schemi di elaborazione del database.**
    1. Da scrivere.
13. **DDS dei display file da usare nei programmi interattivi e utilizzo di SDA (Screen Design Aid).**
    1. Da scrivere.
14. **Schema di programma interattivo di gestione file.**
    1. Definiamo il seguente file fisico **CLITE00F**.

**A R CLITE TEXT('Testata Cliente')  
 A CLCOD 6A COLHDG('Codice' 'Cliente')  
 A CLSTA 2A COLHDG('Codice' 'Paese')  
 A CLRAS 50A COLHDG('Ragione' 'Sociale')  
 A CLIND 50A COLHDG('Indirizzo')  
 A CLFID 11S 2 COLHDG('Fido')  
 A EDTCDE(K)  
 A CLANN 1A COLHDG('Annullamento')  
 A VALUES('A' ' ')**

* 1. Definiamo il seguente file logico **CLITE01L**.

**A UNIQUE  
 A R CLITE PFILE(CLITE00F)  
 A K CLCOD**

* 1. I successivi paragrafi presentano versioni progressive del medesimo programma.  
     Nella prima versione si imposta la struttura di sola visualizzazione ma con caratteristiche adatte alla crescita progressiva prima verso la gestione monoutente e poi verso quella multiutente.  
     Le caratteristiche fondamentali pro crescita successiva sono:  
     1) La strutturazione in routine, una per ogni formato video da gestire.  
     2) Un ciclo infinito intorno ad ogni video.  
     3) Ricicli e abbandoni nei cicli con ITER e LEAVE.  
     4) Comandi di navigazione omogenei tra i formati del programma e con gli altri programmi simili del pacchetto che si sta sviluppando.
  2. Definiamo la prima versione del file video **CLIGES1W**.

**A DSPSIZ(24 80 \*DS3)**

**A R W0**

**A CF03**

**A CF12**

**A PAGEDOWN(01 'Primo')**

**A PAGEUP(02 'Ultimo')**

**A 3 32'ANAGRAFICA CLIENTE'**

**A DSPATR(HI)**

**A DSPATR(UL)**

**A 4 38'Guida'**

**A DSPATR(HI)**

**A 8 6'Codice Cliente:'**

**A W0CLCOD R B 8 23REFFLD(CLITE/CLCOD \*LIBL/CLITE00F)**

**A 21 2'F3/F12=Esce'**

**A DSPATR(HI)**

**A 21 15'PagDw=Primo'**

**A DSPATR(HI)**

**A 21 28'PagUp=Ultimo'**

**A DSPATR(HI)**

**A R W1**

**A\*%%TS SD 20150419 195225 COCIANI REL-V5R3M0 5722-WDS**

**A CF03**

**A CF12**

**A PAGEDOWN(01 'Successivo')**

**A PAGEUP(02 'Precedente')**

**A 3 32'ANAGRAFICA CLIENTE'**

**A DSPATR(HI)**

**A DSPATR(UL)**

**A 4 39'Dati'**

**A DSPATR(HI)**

**A 8 6'Codice Cliente:'**

**A W1CLCOD R O 8 23REFFLD(CLITE/CLCOD \*LIBL/CLITE00F)**

**A 9 8'Codice Paese:'**

**A W1CLSTA R O 9 23REFFLD(CLITE/CLSTA \*LIBL/CLITE00F)**

**A 10 5'Ragione Sociale:'**

**A W1CLRAS R O 10 23REFFLD(CLITE/CLRAS \*LIBL/CLITE00F)**

**A 11 11'Indirizzo:'**

**A W1CLIND R O 11 23REFFLD(CLITE/CLIND \*LIBL/CLITE00F)**

**A 12 16'Fido:'**

**A W1CLFID R O 12 23REFFLD(CLITE/CLFID \*LIBL/CLITE00F)**

**A 13 8'Annullamento:'**

**A W1CLANN R O 13 23REFFLD(CLITE/CLANN \*LIBL/CLITE00F)**

**A 21 2'F3=Esce'**

**A DSPATR(HI)**

**A 21 11'F12=Guida'**

**A DSPATR(HI)**

**A 21 22'PagDw=Successivo'**

**A DSPATR(HI)**

**A 21 40'PagUp=Precedente'**

**A DSPATR(HI)**

* 1. Definiamo la prima versione del programma **CLIGES1**.  
     Il programma in rosso, i commenti in eccesso per uso didattico in verde.  
     I commenti in eccesso appartengono alla versione 1.

**\* Claudio Neroni 19-04-2015 Creato**

**\* Gestisce Clienti - Lettura**

**\*---------------------------------------------------------------------------------------------**

**\* Prima versione del programma di gestione.**

**\* Esegue solo lettura del file di dati.**

**1 \* Il programma non riceve parametri.**

**1 \* VERSIONE SUPERCOMMENTATA PER SCOPO DIDATTICO**

**1 \*---------------------------------------------------------------------------------------------**

**1 \* DEFINIZIONE DEI FILE UTILIZZATI**

**1 \* I file sono aperti implicitamente dall'RPG all'inizio del run.**

**\*---------------------------------------------------------------------------------------------**

**\* Testata Clienti per Codice.**

**1 \* Il file logico utilizza un file fisico.**

**1 \* Entrambi devono preesistere sia alla compilazione che alla esecuzione**

**1 \* del presente programma.**

**1 \* Il fisico deve preesistere anche alla composizione tramite SDA (Screen Design Aid)**

**1 \* del file video.**

**1 \* Visto inoltre che il programma visualizza soltanto i dati e non**

**1 \* ne permette l'aggiunta, prima dell'esecuzione occorre inserire**

**1 \* nel fisico non meno di 5 record facendo uso di DFU (Data File Utility)**

**1 \* tramite il comando UPDDTA FILE(CLITE01L)**

**1 \*CLITE01L Nome file logico con chiave unica**

**1 \* I Input Solo in lettura**

**1 \* F Full procedural Ad accesso mirato**

**1 \* E External described Descritto esternamente**

**1 \* K Keyed Con chiavi**

**1 \* DISK Disk Su disco**

**FCLITE01L IF E K DISK**

**\* Video.**

**1 \* If file contiene due formati video**

**1 \* Video guida W0**

**1 \* Video dati W1**

**1 \* Il file deve preesistere sia alla compilazione che alla esecuzione**

**1 \* del presente programma.**

**1 \* Il video guida contiene solo una chiave di posizionamento sui dati**

**1 \* e i comandi di navigazione.**

**1 \* Il video dati contiene tutti i campi del file fisico da gestire**

**1 \* e i comandi di navigazione. Il nome dei campi sul video dati si ottiene**

**1 \* prefissando con W1 il nome dei campi del file dati in modo da creare una**

**1 \* facile corrispondenza mnemonica tra le due serie di nomi.**

**1 \*CLIGES1W Nome file video**

**1 \* C Combined Combinato (tipico dei file video)**

**1 \* F Full procedural Ad accesso mirato**

**1 \* E External described Descritto esternamente**

**1 \* WORKSTN Workstation Su Video**

**FCLIGES1W CF E WORKSTN**

**\*---------------------------------------------------------------------------------------------**

**\* MAIN ROUTINE**

**1 \* Contiene la gestione del video guida W0.**

**\*---------------------------------------------------------------------------------------------**

**\* Prenota chiusura di file e programma a fine del corrente calcolo.**

**1 \* Accende indicatore LR = Last Record**

**C SETON LR**

**\* Cicla all'infinito intorno al video guida.**

**C DO \*HIVAL**

**\* Emette il video guida.**

**1 \* La guida contiene il campo di ricerca**

**1 \* e i comandi di navigazione.**

**1 \* EXFMT = Execute Format esegue prima WRITE e poi READ del video guida.**

**1 \* La READ ritorna il controllo al programma solo alla pressione**

**1 \* di un comando o dell'invio.**

**C EXFMT W0**

**\* Se premuto il comando F3, abbandona il ciclo intorno al video guida.**

**1 \* Il primo evento da testare è la richiesta di abbandono totale.**

**C KC LEAVE**

**\* Se premuto il comando F12, abbandona il ciclo intorno al video guida.**

**1 \* Il secondo evento da testare è la richiesta di ritorno al passo precedente.**

**1 \* In questo caso passo precedente e abbandono hanno lo stesso effetto.**

**C KL LEAVE**

**\* Sceglie tra PageDown, PageUp e Invio.**

**C SELECT**

**\* Se richiesto PageDown.**

**1 \* Sul video il tasto PageDown è associato all'indicatore \*IN01.**

**1 \* E' richiesto il record con chiave uguale o successiva a quella digitata.**

**C WHEN \*IN01**

**\* Se il codice di posizionamento del video guida è vuoto.**

**C IF W0CLCOD = \*BLANK**

**\* Posiziona il cursore di lettura all'inizio del file.**

**C \*LOVAL SETLL CLITE**

**\* Se il codice di posizionamento del video guida è valorizzato.**

**C ELSE**

**\* Posiziona il cursore di lettura prima del codice di posizionamento.**

**C W0CLCOD SETLL CLITE**

**\* Se il codice di posizionamento del video guida è valorizzato.**

**C ENDIF**

**\* Legge il record immediatamente successivo al cursore.**

**1 \* Il cursore non è mai su di un record ma prima o dopo un record.**

**1 \* La lettura in avanti legge il record dopo il cursore**

**1 \* e sposta il cursore oltre il record letto.**

**C READ CLITE**

**\* Se il record non esiste, ricicla.**

**1 \* Se ha incontrato fine file, torna all'emissione del video guida.**

**C IF %EOF**

**C ITER**

**C ENDIF**

**\* Esegue il salvataggio della chiave dell'ultimo record letto.**

**C EXSR KYSAV**

**\* Se richiesto PageUp.**

**1 \* Sul video il tasto PageUp è associato all'indicatore \*IN02.**

**1 \* E' richiesto il record con chiave precedente a quella digitata.**

**C WHEN \*IN02**

**\* Se il codice di posizionamento del video guida è vuoto.**

**C IF W0CLCOD = \*BLANK**

**\* Posiziona il cursore di lettura alla fine del file.**

**C \*HIVAL SETGT CLITE**

**\* Se il codice di posizionamento del video guida è valorizzato.**

**C ELSE**

**\* Posiziona il cursore di lettura dopo il codice di posizionamento.**

**C W0CLCOD SETGT CLITE**

**\* Se il codice di posizionamento del video guida è valorizzato.**

**C ENDIF**

**\* Legge il record immediatamente precedente al cursore.**

**1 \* Il cursore non è mai su di un record ma prima o dopo un record.**

**1 \* La lettura all'indietro legge il record prima del cursore**

**1 \* e sposta il cursore prima del record letto.**

**C READP CLITE**

**\* Se il record non esiste, ricicla.**

**1 \* Se ha incontrato inizio file, torna all'emissione del video guida.**

**C IF %EOF**

**C ITER**

**C ENDIF**

**\* Esegue il salvataggio della chiave dell'ultimo record letto.**

**C EXSR KYSAV**

**\* Se richiesto Invio.**

**1 \* Il tasto di Invio non ha un indicatore associato e viene individuato per esclusione.**

**C OTHER**

**\* Cerca il record con la chiave esatta indicata sul video guida.**

**C W0CLCOD CHAIN CLITE**

**\* Se il record non esiste, ricicla.**

**1 \* Torna all'emissione del video guida.**

**C IF NOT %FOUND**

**C ITER**

**C ENDIF**

**\* Esegue il salvataggio della chiave dell'ultimo record letto.**

**C EXSR KYSAV**

**\* Sceglie tra PageDown, PageUp e Invio.**

**C ENDSL**

**\* Acquisito un record, lo gestisce tramite la subroutine del video dati.**

**C EXSR SRW1**

**\* Se premuto il comando F3, abbandona.**

**1 \* Al ritorno dalla subroutine del video dati, testa la richiesta**

**1 \* di abbandono totale e abbandona anche il ciclo intorno al video guida.**

**C KC LEAVE**

**\* Cicla all'infinito intorno al video guida.**

**C ENDDO**

**\* Ritorna al chiamante.**

**1 \* Essendo acceso l'indicatore LR, chiude i file e termina completamente.**

**C RETURN**

**\*---------------------------------------------------------------------------------------------**

**\* Gestisce il video dati W1.**

**1 \* La subroutine entra su un record già acquisito e ne acquisisce a sua volta.**

**\*---------------------------------------------------------------------------------------------**

**C SRW1 BEGSR**

**\* Cicla all'infinito intorno al video dati.**

**C DO \*HIVAL**

**\* Copia i dati dal record al video dati.**

**C EXSR DBW1**

**\* Emette il video dati.**

**1 \* Il video dati contiene il record ultimo letto dal file**

**1 \* e i comandi di navigazione.**

**1 \* EXFMT = Execute Format esegue prima WRITE e poi READ del video dati.**

**1 \* La READ ritorna il controllo al programma solo alla pressione**

**1 \* di un comando o dell'invio.**

**C EXFMT W1**

**\* Se premuto il comando F3, abbandona.**

**1 \* Il primo evento da testare è la richiesta di abbandono totale.**

**1 \* Se richiesto, abbandona il ciclo intorno al video dati**

**1 \* e, al ritorno nel ciclo del video guida, il test di F3 viene rieseguito**

**1 \* per abbandonare anche quel ciclo.**

**C KC LEAVE**

**\* Se premuto il comando F12, abbandona.**

**1 \* Il secondo evento da testare è la richiesta di ritorno al passo precedente.**

**1 \* Se richiesto, abbandona il ciclo intorno al video dati**

**1 \* e ritorna sul video guida.**

**C KL LEAVE**

**\* Sceglie tra PageDown, PageUp e Invio.**

**C SELECT**

**\* Se richiesto PageDown.**

**1 \* Sul video il tasto PageDown è associato all'indicatore \*IN01.**

**1 \* E' richiesto il record con chiave successiva a quella**

**1 \* dell'ultimo record letto e visto.**

**C WHEN \*IN01**

**\* Si posiziona oltre l'ultimo record visto.**

**C KYCLCOD SETGT CLITE**

**\* Legge il record successivo.**

**C READ CLITE**

**\* Se il record non esiste, abbandona.**

**1 \* Se ha incontrato fine file torna all'emissione del video guida.**

**C IF %EOF**

**C LEAVE**

**C ENDIF**

**\* Esegue il salvataggio della chiave dell'ultimo record letto.**

**C EXSR KYSAV**

**\* Se richiesto PageUp.**

**1 \* Sul video il tasto PageUp è associato all'indicatore \*IN02.**

**1 \* E' richiesto il record con chiave precedente a quella**

**1 \* dell'ultimo record letto e visto.**

**C WHEN \*IN02**

**\* Si posiziona prima dell'ultimo record visto.**

**C KYCLCOD SETLL CLITE**

**\* Legge il record precedente.**

**C READP CLITE**

**\* Se il record non esiste, abbandona.**

**1 \* Se ha incontrato inizio file torna all'emissione del video guida.**

**C IF %EOF**

**C LEAVE**

**C ENDIF**

**\* Esegue il salvataggio della chiave dell'ultimo record letto.**

**C EXSR KYSAV**

**\* Se richiesto Invio.**

**1 \* Il tasto di Invio non ha un indicatore associato e viene individuato per esclusione.**

**C OTHER**

**\* Abbandona il video dati.**

**C LEAVE**

**\* Sceglie tra PageDown, PageUp e Invio.**

**C ENDSL**

**\* Cicla all'infinito intorno al video dati.**

**C ENDDO**

**C ENDSR**

**\*---------------------------------------------------------------------------------------------**

**\* Copia i dati dal record al video dati.**

**\*---------------------------------------------------------------------------------------------**

**C DBW1 BEGSR**

**\* Copia tutti i campi dal record dati ai campi omonimi del video dati.**

**1 \* I nomi del video sono uguali a quelli del file dati salvo il prefisso W1.**

**C EVAL W1CLCOD = CLCOD**

**C EVAL W1CLSTA = CLSTA**

**C EVAL W1CLRAS = CLRAS**

**C EVAL W1CLIND = CLIND**

**C EVAL W1CLFID = CLFID**

**C EVAL W1CLANN = CLANN**

**C ENDSR**

**\*---------------------------------------------------------------------------------------------**

**\* Salva la chiave dell'ultimo record letto dal file.**

**\*---------------------------------------------------------------------------------------------**

**C KYSAV BEGSR**

**\* Definisce la chiave di salvataggio.**

**C \*LIKE DEFINE CLCOD KYCLCOD**

**\* Copia la chiave dal record in canna al salvataggio.**

**C EVAL KYCLCOD = CLCOD**

**C ENDSR**

* 1. Definiamo la seconda versione del file video **CLIGES2W**.

**A DSPSIZ(24 80 \*DS3)**

**A R W0**

**A CF03**

**A CF12**

**A PAGEDOWN(01 'Primo')**

**A PAGEUP(02 'Ultimo')**

**A CF09**

**A 3 32'ANAGRAFICA CLIENTE'**

**A DSPATR(HI)**

**A DSPATR(UL)**

**A 4 38'Guida'**

**A DSPATR(HI)**

**A 8 6'Codice Cliente:'**

**A W0CLCOD R B 8 23REFFLD(CLITE/CLCOD \*LIBL/CLITE00F)**

**A 21 2'F3/F12=Esce'**

**A DSPATR(HI)**

**A 21 15'PagDw=Primo'**

**A DSPATR(HI)**

**A 21 28'PagUp=Ultimo'**

**A DSPATR(HI)**

**A 21 42'F9=Agiunge'**

**A DSPATR(HI)**

**A R W1**

**A CF03**

**A CF12**

**A PAGEDOWN(01 'Successivo')**

**A PAGEUP(02 'Precedente')**

**A 3 32'ANAGRAFICA CLIENTE'**

**A DSPATR(HI)**

**A DSPATR(UL)**

**A 4 39'Dati'**

**A DSPATR(HI)**

**A 8 6'Codice Cliente:'**

**A W1CLCOD R B 8 23REFFLD(CLITE/CLCOD \*LIBL/CLITE00F)**

**A 61 60 ERRMSG('Chiave doppia')**

**A 62 60 ERRMSG('Campo obbligatorio')**

**A 9 8'Codice Paese:'**

**A W1CLSTA R B 9 23REFFLD(CLITE/CLSTA \*LIBL/CLITE00F)**

**A 63 60 ERRMSG('Paese non valido')**

**A W1CLSTAD 20A O 9 27**

**A 10 5'Ragione Sociale:'**

**A W1CLRAS R B 10 23REFFLD(CLITE/CLRAS \*LIBL/CLITE00F)**

**A 11 11'Indirizzo:'**

**A W1CLIND R B 11 23REFFLD(CLITE/CLIND \*LIBL/CLITE00F)**

**A 12 16'Fido:'**

**A W1CLFID R B 12 23REFFLD(CLITE/CLFID \*LIBL/CLITE00F)**

**A 13 8'Annullamento:'**

**A W1CLANN R B 13 23REFFLD(CLITE/CLANN \*LIBL/CLITE00F)**

**A 21 2'F3=Esce'**

**A DSPATR(HI)**

**A 21 11'F12=Guida'**

**A DSPATR(HI)**

**A 21 22'PagDw=Successivo'**

**A DSPATR(HI)**

**A 21 40'PagUp=Precedente'**

**A DSPATR(HI)**

* 1. Definiamo la seconda versione del programma **CLIGES2**.  
     Il programma in rosso, i commenti in eccesso per uso didattico in verde.  
     I commenti in eccesso appartengono alla versione 1 o 2.

**\* Claudio Neroni 19-04-2015 Creato**

**\* Gestisce Clienti - Lettura**

**\*---------------------------------------------------------------------------------------------**

**\* Seconda versione del programma di gestione.**

**\* Esegue modifica e aggiunta sul file di dati.**

**1 \* Il programma non riceve parametri.**

**1 \* VERSIONE SUPERCOMMENTATA PER SCOPO DIDATTICO**

**1 \*---------------------------------------------------------------------------------------------**

**1 \* DEFINIZIONE DEI FILE UTILIZZATI**

**1 \* I file sono aperti implicitamente dall'RPG all'inizio del run.**

**\*---------------------------------------------------------------------------------------------**

**\* Testata Clienti per Codice.**

**1 \* Il file logico utilizza un file fisico.**

**1 \* Entrambi devono preesistere sia alla compilazione che alla esecuzione**

**1 \* del presente programma.**

**1 \* Il fisico deve preesistere anche alla composizione tramite SDA (Screen Design Aid)**

**1 \* del file video.**

**1 \* Visto inoltre che il programma visualizza soltanto i dati e non**

**1 \* ne permette l'aggiunta, prima dell'esecuzione occorre inserire**

**1 \* nel fisico non meno di 5 record facendo uso di DFU (Data File Utility)**

**1 \* tramite il comando UPDDTA FILE(CLITE01L)**

**2 \* Permessa ora non solo Lettura ma anche Update (Ricalco) e Aggiunta record.**

**2 \* Quindi non più necessario precaricare record con DFU in questo file.**

**1 \*CLITE01L Nome file logico con chiave unica**

**2 \* U Update Lettura e ricalco record**

**1 \* F Full procedural Ad accesso mirato**

**2 \* A Add Con aggiunta record**

**1 \* E External described Descritto esternamente**

**1 \* K Keyed Con chiavi**

**1 \* DISK Disk Su disco**

**FCLITE01L IF E K DISK**

**\* Video.**

**1 \* If file contiene due formati video**

**1 \* Video guida W0**

**1 \* Video dati W1**

**1 \* Il file deve preesistere sia alla compilazione che alla esecuzione**

**1 \* del presente programma.**

**1 \* Il video guida contiene solo una chiave di posizionamento sui dati**

**1 \* e i comandi di navigazione.**

**1 \* Il video dati contiene tutti i campi del file fisico da gestire**

**1 \* e i comandi di navigazione. Il nome dei campi sul video dati si ottiene**

**1 \* prefissando con W1 il nome dei campi del file dati in modo da creare una**

**1 \* facile corrispondenza mnemonica tra le due serie di nomi.**

**2 \* Il file video cambia nome e diventa 2.**

**2 \* Modifiche al formato W0.**

**2 \* Compare il comando F9 per entrare nella modalità aggiunta.**

**2 \* Modifiche al formato W1.**

**2 \* Tutti i campi, che nella versione 1 erano di tipo O=Output,**

**2 \* diventano di tipo B=Both (Input-Output)**

**2 \* per permettere modifica e aggiunta dei record.**

**2 \* Per segnalare gli errori rilevati dalla subroutine di controllo,**

**2 \* sono usati gli ERRMSG con un testo da segnalare.**

**2 \* Ad ogni ERRMSG è assegnato un indicatore di errore generico**

**2 \* \*IN60 ed uno specifico da \*IN61 a \*IN90.**

**2 \* La comparsa della segnalazione si ottiene accendendo**

**2 \* entrambi gli indicatori.**

**2 \* Si introduce anche la decodifica di un campo codice**

**2 \* utilizzando il file anagrafico di quella codifica.**

**2 \*CLIGES2W Nome file video**

**1 \* C Combined Combinato (tipico dei file video)**

**1 \* F Full procedural Ad accesso mirato**

**1 \* E External described Descritto esternamente**

**1 \* WORKSTN Workstation Su Video**

**FCLIGES2W CF E WORKSTN**

**\* Paese per codice.**

**2 \* Serve alla decodifica del codice paese.**

**2 \* Il file è un fisico con chiave.**

**2 \* Pur essendo preferibile che i fisici non possiedano chiavi**

**2 \* e pur essendo di inciampo in vista di una futura trasformazione**

**2 \* in tabelle SQL, si esemplifica questa soluzione perché molto**

**2 \* adatta a decodifiche di un numero limitato di codici.**

**2 \* Prima dell'uso è indispensabile caricare nel file le decodifiche**

**2 \* dei paesi che si useranno tramite il comando UPDDTA FILE(PAESE00F)**

**2 \*PAESE00F Nome file fisico con chiave unica**

**2 \* I Input Solo in lettura**

**2 \* F Full procedural Ad accesso mirato**

**2 \* E External described Descritto esternamente**

**2 \* K Keyed Con chiavi**

**2 \* DISK Disk Su disco**

**FPAESE00F IF E K DISK**

**\*---------------------------------------------------------------------------------------------**

**\* MAIN ROUTINE**

**1 \* Contiene la gestione del video guida W0.**

**\*---------------------------------------------------------------------------------------------**

**\* Prenota chiusura di file e programma a fine del corrente calcolo.**

**1 \* Accende indicatore LR = Last Record**

**C SETON LR**

**\* Cicla all'infinito intorno al video guida.**

**C DO \*HIVAL**

**\* Emette il video guida.**

**1 \* La guida contiene il campo di ricerca**

**1 \* e i comandi di navigazione.**

**1 \* EXFMT = Execute Format esegue prima WRITE e poi READ del video guida.**

**1 \* La READ ritorna il controllo al programma solo alla pressione**

**1 \* di un comando o dell'invio.**

**C EXFMT W0**

**\* Se premuto il comando F3, abbandona il ciclo intorno al video guida.**

**1 \* Il primo evento da testare è la richiesta di abbandono totale.**

**C KC LEAVE**

**\* Se premuto il comando F12, abbandona il ciclo intorno al video guida.**

**1 \* Il secondo evento da testare è la richiesta di ritorno al passo precedente.**

**1 \* In questo caso passo precedente e abbandono hanno lo stesso effetto.**

**C KL LEAVE**

**\* Sceglie tra PageDown, PageUp e Invio.**

**C SELECT**

**\* Se richiesto F9.**

**2 \* Il comando F9 è automaticamente associato all'indicatore rpg \*INKI**

**2 \* Come dallo schema seguente.**

**2 \* F1=KA F2=KB F3=KC F4=KD F5=KE F6=KF F7=KG F8=KH F9=KI F10=KJ**

**2 \* F11=KK F12=KL F13=KM F14=KN F15=KP F16=KQ F17=KR F18=KS F19=KT F20=KU**

**2 \* F21=KV F22=KW F23=KX F24=KY**

**2 \* Per rendere mnemonica la sequenza, si noti l'assenza**

**2 \* del malaugurante "KO".**

**C WHEN \*INKI**

**\* Esegue Subroutine di Aggiunta.**

**C EXSR ADW1**

**\* Se premuto il comando F3, abbandona.**

**2 \* Al ritorno dalla subroutine di aggiunta, testa la richiesta**

**2 \* di abbandono totale e abbandona anche il ciclo intorno al video guida.**

**C KC LEAVE**

**2 \* Torna all'emissione del video guida.**

**C ITER**

**\* Se richiesto PageDown.**

**1 \* Sul video il tasto PageDown è associato all'indicatore \*IN01.**

**1 \* E' richiesto il record con chiave uguale o successiva a quella digitata.**

**C WHEN \*IN01**

**\* Se il codice di posizionamento del video guida è vuoto.**

**C IF W0CLCOD = \*BLANK**

**\* Posiziona il cursore di lettura all'inizio del file.**

**C \*LOVAL SETLL CLITE**

**\* Se il codice di posizionamento del video guida è valorizzato.**

**C ELSE**

**\* Posiziona il cursore di lettura prima del codice di posizionamento.**

**C W0CLCOD SETLL CLITE**

**\* Se il codice di posizionamento del video guida è valorizzato.**

**C ENDIF**

**\* Legge il record immediatamente successivo al cursore.**

**1 \* Il cursore non è mai su di un record ma prima o dopo un record.**

**1 \* La lettura in avanti legge il record dopo il cursore**

**1 \* e sposta il cursore oltre il record letto.**

**C READ CLITE**

**\* Se il record non esiste, ricicla.**

**1 \* Se ha incontrato fine file, torna all'emissione del video guida.**

**C IF %EOF**

**C ITER**

**C ENDIF**

**\* Esegue il salvataggio della chiave dell'ultimo record letto.**

**C EXSR KYSAV**

**\* Se richiesto PageUp.**

**1 \* Sul video il tasto PageUp è associato all'indicatore \*IN02.**

**1 \* E' richiesto il record con chiave precedente a quella digitata.**

**C WHEN \*IN02**

**\* Se il codice di posizionamento del video guida è vuoto.**

**C IF W0CLCOD = \*BLANK**

**\* Posiziona il cursore di lettura alla fine del file.**

**C \*HIVAL SETGT CLITE**

**\* Se il codice di posizionamento del video guida è valorizzato.**

**C ELSE**

**\* Posiziona il cursore di lettura dopo il codice di posizionamento.**

**C W0CLCOD SETGT CLITE**

**\* Se il codice di posizionamento del video guida è valorizzato.**

**C ENDIF**

**\* Legge il record immediatamente precedente al cursore.**

**1 \* Il cursore non è mai su di un record ma prima o dopo un record.**

**1 \* La lettura all'indietro legge il record prima del cursore**

**1 \* e sposta il cursore prima del record letto.**

**C READP CLITE**

**\* Se il record non esiste, ricicla.**

**1 \* Se ha incontrato inizio file, torna all'emissione del video guida.**

**C IF %EOF**

**C ITER**

**C ENDIF**

**\* Esegue il salvataggio della chiave dell'ultimo record letto.**

**C EXSR KYSAV**

**\* Se richiesto Invio.**

**1 \* Il tasto di Invio non ha un indicatore associato e viene individuato per esclusione.**

**C OTHER**

**\* Cerca il record con la chiave esatta indicata sul video guida.**

**C W0CLCOD CHAIN CLITE**

**\* Se il record non esiste, ricicla.**

**1 \* Torna all'emissione del video guida.**

**C IF NOT %FOUND**

**C ITER**

**C ENDIF**

**\* Esegue il salvataggio della chiave dell'ultimo record letto.**

**C EXSR KYSAV**

**\* Sceglie tra PageDown, PageUp e Invio.**

**C ENDSL**

**\* Acquisito un record, lo gestisce tramite la subroutine del video dati.**

**C EXSR SRW1**

**\* Se premuto il comando F3, abbandona.**

**1 \* Al ritorno dalla subroutine del video dati, testa la richiesta**

**1 \* di abbandono totale e abbandona anche il ciclo intorno al video guida.**

**C KC LEAVE**

**\* Cicla all'infinito intorno al video guida.**

**C ENDDO**

**\* Ritorna al chiamante.**

**1 \* Essendo acceso l'indicatore LR, chiude i file e termina completamente.**

**C RETURN**

**\*---------------------------------------------------------------------------------------------**

**\* Gestisce il video dati W1.**

**1 \* La subroutine entra su un record già acquisito e ne acquisisce a sua volta.**

**\*---------------------------------------------------------------------------------------------**

**C SRW1 BEGSR**

**\* Cicla all'infinito intorno al video dati.**

**C DO \*HIVAL**

**\* Copia i dati dal record al video dati.**

**C EXSR DBW1**

**\* Decodifica il video dati.**

**C EXSR DEW1**

**\* Se riscontrato errore, emette separatamente le decodifiche.**

**C IF \*IN60**

**C SETOFF 60**

**C WRITE W1**

**C SETON 60**

**C ENDIF**

**\* Emette il video dati.**

**1 \* Il video dati contiene il record ultimo letto dal file**

**1 \* e i comandi di navigazione.**

**1 \* EXFMT = Execute Format esegue prima WRITE e poi READ del video dati.**

**1 \* La READ ritorna il controllo al programma solo alla pressione**

**1 \* di un comando o dell'invio.**

**C EXFMT W1**

**\* Spegne gli indicatori di errore.**

**C EXSR $SETOFF**

**\* Se premuto il comando F3, abbandona.**

**1 \* Il primo evento da testare è la richiesta di abbandono totale.**

**1 \* Se richiesto, abbandona il ciclo intorno al video dati**

**1 \* e, al ritorno nel ciclo del video guida, il test di F3 viene rieseguito**

**1 \* per abbandonare anche quel ciclo.**

**C KC LEAVE**

**\* Se premuto il comando F12, abbandona.**

**1 \* Il secondo evento da testare è la richiesta di ritorno al passo precedente.**

**1 \* Se richiesto, abbandona il ciclo intorno al video dati**

**1 \* e ritorna sul video guida.**

**C KL LEAVE**

**\* Decodifica il video dati.**

**C EXSR DEW1**

**\* Controlla errori.**

**C EXSR CTW1**

**\* Se presenti errori, ricicla.**

**C 60 ITER**

**\* Trascrive dal video dati ai campi del record.**

**C EXSR W1DB**

**\* Ricalca il record.**

**C UPDATE(E) CLITE**

**\* Se errore chiave doppia, accende errore e ricicla,**

**C IF %ERROR**

**C SETON 6061**

**C ITER**

**C ENDIF**

**\* Sceglie tra PageDown, PageUp e Invio.**

**C SELECT**

**\* Se richiesto PageDown.**

**1 \* Sul video il tasto PageDown è associato all'indicatore \*IN01.**

**1 \* E' richiesto il record con chiave successiva a quella**

**1 \* dell'ultimo record letto e visto.**

**C WHEN \*IN01**

**\* Si posiziona oltre l'ultimo record visto.**

**C KYCLCOD SETGT CLITE**

**\* Legge il record successivo.**

**C READ CLITE**

**\* Se il record non esiste, abbandona.**

**1 \* Se ha incontrato fine file torna all'emissione del video guida.**

**C IF %EOF**

**C LEAVE**

**C ENDIF**

**\* Esegue il salvataggio della chiave dell'ultimo record letto.**

**C EXSR KYSAV**

**\* Se richiesto PageUp.**

**1 \* Sul video il tasto PageUp è associato all'indicatore \*IN02.**

**1 \* E' richiesto il record con chiave precedente a quella**

**1 \* dell'ultimo record letto e visto.**

**C WHEN \*IN02**

**\* Si posiziona prima dell'ultimo record visto.**

**C KYCLCOD SETLL CLITE**

**\* Legge il record precedente.**

**C READP CLITE**

**\* Se il record non esiste, abbandona.**

**1 \* Se ha incontrato inizio file torna all'emissione del video guida.**

**C IF %EOF**

**C LEAVE**

**C ENDIF**

**\* Esegue il salvataggio della chiave dell'ultimo record letto.**

**C EXSR KYSAV**

**\* Se richiesto Invio.**

**1 \* Il tasto di Invio non ha un indicatore associato e viene individuato per esclusione.**

**C OTHER**

**\* Abbandona il video dati.**

**C LEAVE**

**\* Sceglie tra PageDown, PageUp e Invio.**

**C ENDSL**

**\* Cicla all'infinito intorno al video dati.**

**C ENDDO**

**C ENDSR**

**\*---------------------------------------------------------------------------------------------**

**\* Copia i dati dal record al video dati.**

**\*---------------------------------------------------------------------------------------------**

**C DBW1 BEGSR**

**\* Copia tutti i campi dal record dati ai campi omonimi del video dati.**

**1 \* I nomi del video sono uguali a quelli del file dati salvo il prefisso W1.**

**C EVAL W1CLCOD = CLCOD**

**C EVAL W1CLSTA = CLSTA**

**C EVAL W1CLRAS = CLRAS**

**C EVAL W1CLIND = CLIND**

**C EVAL W1CLFID = CLFID**

**C EVAL W1CLANN = CLANN**

**C ENDSR**

**\*---------------------------------------------------------------------------------------------**

**\* Salva la chiave dell'ultimo record letto dal file.**

**\*---------------------------------------------------------------------------------------------**

**C KYSAV BEGSR**

**\* Definisce la chiave di salvataggio.**

**C \*LIKE DEFINE CLCOD KYCLCOD**

**\* Copia la chiave dal record in canna al salvataggio.**

**C EVAL KYCLCOD = CLCOD**

**C ENDSR**

**2 \*---------------------------------------------------------------------------------------------**

**2 \* LE SUBROUTINE SUCCESSIVE NASCONO DALLA VERSIONE 2**

**\*---------------------------------------------------------------------------------------------**

**\* Copia tutti i campi dal video dati ai campi omonimi del record dati.**

**\*---------------------------------------------------------------------------------------------**

**C W1DB BEGSR**

**C EVAL CLCOD = W1CLCOD**

**C EVAL CLSTA = W1CLSTA**

**C EVAL CLRAS = W1CLRAS**

**C EVAL CLIND = W1CLIND**

**C EVAL CLFID = W1CLFID**

**C EVAL CLANN = W1CLANN**

**C ENDSR**

**\*---------------------------------------------------------------------------------------------**

**\* Pulisce tutti i campi del video dati.**

**\*---------------------------------------------------------------------------------------------**

**C CLW1 BEGSR**

**\* Pulisce i campi dati.**

**C CLEAR W1CLCOD**

**C CLEAR W1CLSTA**

**C CLEAR W1CLRAS**

**C CLEAR W1CLIND**

**C CLEAR W1CLFID**

**C CLEAR W1CLANN**

**\* Pulisce i campi di decodifica.**

**C CLEAR W1CLSTAD**

**C ENDSR**

**\*---------------------------------------------------------------------------------------------**

**\* Gestisce l'aggiunta record.**

**\*---------------------------------------------------------------------------------------------**

**C ADW1 BEGSR**

**\* Pulisce tutti i campi del video dati.**

**C EXSR CLW1**

**\* Cicla all'infinito intorno al video dati.**

**C DO \*HIVAL**

**\* Se riscontrato errore, emette separatamente le decodifiche.**

**C IF \*IN60**

**C SETOFF 60**

**C WRITE W1**

**C SETON 60**

**C ENDIF**

**\* Emette il video dati.**

**C EXFMT W1**

**\* Spegne gli indicatori di errore.**

**C EXSR $SETOFF**

**\* Se premuto il comando F3, abbandona.**

**C KC LEAVE**

**\* Se premuto il comando F12, abbandona.**

**C KL LEAVE**

**\* Decodifica il video dati.**

**C EXSR DEW1**

**\* Controlla errori.**

**C EXSR CTW1**

**\* Se presenti errori, ricicla.**

**C 60 ITER**

**\* Trascrive dal video dati ai campi del record.**

**C EXSR W1DB**

**\* Scrive il record.**

**C WRITE(E) CLITE**

**\* Se errore chiave doppia, accende errore e ricicla,**

**C IF %ERROR**

**C SETON 6061**

**C ITER**

**C ENDIF**

**\* Pulisce tutti i campi del video dati.**

**C EXSR CLW1**

**\* Cicla all'infinito intorno al video dati.**

**C ENDDO**

**C ENDSR**

**\*---------------------------------------------------------------------------------------------**

**\* Spegne gli indicatori di errore.**

**\*---------------------------------------------------------------------------------------------**

**C $SETOFF BEGSR**

**\* Spegne tutti gli indicatori di errore da \*in60 a \*in90.**

**C MOVE \*ZERO $31 31**

**C MOVEA $31 \*IN(60)**

**C ENDSR**

**\*---------------------------------------------------------------------------------------------**

**\* Decodifica il video dati.**

**\*---------------------------------------------------------------------------------------------**

**C DEW1 BEGSR**

**\* Decodifica campo paese.**

**C DO**

**\* Pulisce la decodifica.**

**C CLEAR W1CLSTAD**

**\* Se il codice è valorizzato.**

**C IF W1CLSTA<>\*BLANK**

**\* Cerca il codice in anagrafica.**

**C W1CLSTA CHAIN PAESE**

**\* Se trova l'anagrafica.**

**C IF %FOUND**

**\* Trascrive la decodifica sul video dati.**

**C EVAL W1CLSTAD=PADES**

**\* Se non trova l'anagrafica.**

**C ELSE**

**\* Assume interrogativi nella decodifica.**

**C EVAL W1CLSTAD=\*ALL'?'**

**\* Se non trova l'anagrafica.**

**C ENDIF**

**\* Se il codice è valorizzato.**

**C ENDIF**

**\* Decodifica campo paese.**

**C ENDDO**

**C ENDSR**

**\*---------------------------------------------------------------------------------------------**

**\* Controlla il video dati.**

**\*---------------------------------------------------------------------------------------------**

**C CTW1 BEGSR**

**\* Codice cliente obbligatorio.**

**C IF W1CLCOD=\*BLANK**

**C SETON 6062**

**C ENDIF**

**\* Il codice paese se specificato deve essere in anagrafica.**

**C IF W1CLSTAD=\*ALL'?'**

**C SETON 6063**

**C ENDIF**

**C ENDSR**

* 1. Se hai caricato sul tuo As400 i sorgenti del corso, puoi confrontare i due sorgenti con il comando:  
     **CMPPFM NEWFILE(NERONI2/CORSO400) NEWMBR(CLIGES2) OLDFILE(NERONI2/CORSO400) OLDMBR(CLIGES1) OPTION(\*NARROW)**

1. **Schema di programma interattivo di elenco.**
   1. Da scrivere.
2. **Schema di programma batch.**
   1. Da scrivere.
3. **Schema di programma di stampa e utilizzo di printer file interni ed esterni.**
   1. Da scrivere.
4. **Istruzioni di calcolo numerico.**
   1. Da scrivere.
5. **Istruzioni di elaborazione di stringhe caratteri.**
   1. Da scrivere.
6. **Utilizzo dei manuali As400 dal web.**
   1. Da scrivere.
7. **Eventuale utilizzo delle API (Application Program Interface).**
   1. Da scrivere.

\*\*\*Fine Documento \*\*\*