Consiglio Nazionale delle Ricerche

Programmi per la Gestione dei Codici di lavoro degli Utenti OS/VS2

R. Bandinelli - E. Bracci - A. Ceccarelli

150

CNUCE

Divisione Servizio Elaborazione Dati

A cura di: Rolando Bandinelli
Edoardo Bracci
Alfredo Ceccarelli

Programmi per la gestione dei codici di lavoro degli utenti OS/VS2.

150

CNUCE

INDICE

INTRODUZIONE

Il lavoro descritto nel seguito e' stato ideato allo scopo di rendere automatica la gestione delle richieste di risorse da parte degli utenti per quanto riguarda l'uso del sistema operativo VS/HASP. Visto la grande varieta' di esigenze diverse, l'utenza e' stata suddivisa in vari gruppi il piu' omegenei possibile ciascuno dei quali e' contraddistinto da una lettera dell'alfabeto. All'interno di un gruppo ogni utente ha un numero progressivo che va da 0 a 999. In questo modo gia' dal codice completo dell'utente (lettera + numero di tre cifre) e' possibile avere un'idea del tipo di utenza. L'insieme di programmi scritti permettono di gestire automaticamente le risorse concesse a ciascun utente, contabilizzare il tempo a disposizione e di impedire l'uso di talune risorse a utenti che non ne siano autorizzati. Il sistema da noi costruito consiste di:

1) un archivio contenente tutte le informazioni dettagliate per ogni utente (o codice utente). Tale archivio nel seguito sara' indicato come data set dei codici.

2) tre procedure che gestiscono il data set dei codici piu' esattamente:

   a) creacod crea tale dataset

   b) caricod attiva i codici, inserisce le risorse associate a ciascuno codice e fornisce su carta le informazioni del data set.

   c) dumpcod fornisce su carta le informazioni del data set.

3) Un programma che viene chiamato dal sistema operativo ogni volta che si presenta all'ingresso del calcolatore un lavoro da eseguire. Tale programma esegue un controllo se le risorse richieste dal lavoro sono permesse all'utente (ovvero al codice) del lavoro stesso e permette o meno l'accesso al calcolatore. Tale programma e' stato inserito nel sistema operativo come EXIT del sottosistema SMF IEFUJV. Nel seguito indicheremo questo programma con il nome di EXIT.

Scopo di questa nota interna e' lo spiegare l'uso dei programmi di manutenzione del data set dei codici per cui solo alcune volte sara' nominata la EXIT.
Alla fine della lettera di questa nota interna si dovrebbe essere in grado di poter definire un codice con tutte le risorse permesse e non, modificare un codice (le risorse ad esso associate), leggere la situazione di un codice ecc..

CAP. 1.

Programma di creazione CREACOD

Il programma CREACOD provvede al formattamento di records destinati a contenere le informazioni dei codici abilitati a lavorare in BATCH/OS. I records sono costruiti sul data set dei codici indicato nella scheda DD di nome FILONE.

Questo e' il primo programma che deve essere eseguito nel caso si debba creare, o ricreare, il data set dei codici ed e' omesso allorche' il data set esiste gia' e deve essere solo aggiornato.

Il programma di creazione del data set richiede come scheda dati una scheda con il seguente tracciato:

P a colonna 1
W a colonna 2
C(....40 caratteri) a colonna 3 e successive contenente la password per accedere al data set dei codici.

Tale programma crea il file formattandolo in records e blocchi avendo cura di creare tutti i codici disattivati. (bit 1 del byte 1 del campo LIMITI posto uguale a 0).

I records formattati sono 1040 e ciascun record e' formato da 25 blocchi. Infine ogni blocco e' formato da 40 byte, destinato a contenere le informazioni associate ad un codice.

Il motivo per cui abbiamo scelto questi numeri piuttosto che altri, deriva dal fatto che si debbono costruire 1000 blocchi per ognuna delle lettere dell' alfabeto perche' i codici sono formati da una lettera (26 possibilita') e tre cifre (0-999). (26*1000=1040*25).

La stampa di controllo prodotta da questo programma e' regolata dal parametro specificato sulla scheda EXEC. (Cfr. Cap. 3).

Modo Batch

Per girare in modo batch il programma ha bisogno di una scheda controllo

```
//SYSIN DD *
```

seguita da tutte le schede indicanti le operazioni da fare.
Il tracciato di tali schede e' il seguente:

```
colonna 1      tipo di operazione  (X|E|O|A|I|D|P|*)
colonna 2      spazio
colonna 34÷72  operandi (1)
colonna 73÷80  numerazione facoltativa
```

NOTA (1) Qualora lo spazio non fosse sufficiente e' possibile continuare da colonna 3 della scheda successiva dopo aver specificato "C" nel campo operazione. La scansione del campo operandi si arresta comunque al primo spazio trovato, a meno che l'operazione non sia E nel qual caso viene considerato un campo lungo 40 caratteri anche se contiene spazi.

Modo conversazionale

Per girare in modo conversazionale il programma deve essere eseguito senza avere la scheda

```
//SYSIN DD *
```

in questo caso il programma chiedera' a console i comandi nel seguente modo:

```
OPERAZIONE? RISPOSTA: (X|D|I|O|A|P|?|END)
```

E' necessario rispondere correttamente alle domande tenendo presente che il primo comando deve essere "p" e l'ultimo deve essere "END".

La parte conversazionale del programma (vedi diagramma a blocchi), e' realizzata in modo da richiedere a console tutte le informazioni riguardanti un codice e fornire per ogni domanda tutte le possibili risposte. In questo modo viene creato un record contenente tutte le modifiche che si vogliono apportare ad un codice. Tale modifiche vengono visualizzate per ottenere o meno la conferma da console. Allorche' il programma ottiene la conferma, viene lanciata la macro di ENQ e viene acceduto il data set dei codici. Il codice prescelto e' modificato, aggiunto o eliminato a seconda del comando lanciato. Al termine della modifica viene lanciata la macro di DEQ ed il data set e' rilasciato.

Come si vede per ogni codice si effettua un solo colpo di lettura e il data set e' acceduto per il ristretto tempo necessario alla modifica. Questo per non intralciare l'accesso al data set dei codici fatto dalle EXIT che causerebbe ritardi al normale processo sul 370/158.

CAP. 2.

Operazioni permesse dal programma di Manutenzione in modo BATCH

X Tale operazione e' permessa solo su codici nuovi, cioe' su codici che hanno il primo bit dei limiti uguale a 0. Viene letta la scheda da colonna 3 a colonna 72 e la eventuale continuazione per un totale di 80 caratteri a meno che non vi sia un carattere bianco incontrato prima. I suddetti ottanta caratteri vengono tradotti da esadecimale in EBCDIC, viene ricercato il blocco a cui si riferisce (i primi 4 caratteri devono contenere il codice) e viene sostituito al blocco se tale blocco ha in OFF il bit di attivazione (cioe' quello che indica se il codice e' attivo o meno). Viene dato un messaggio di errore nei seguenti casi:

1) I caratteri della scheda da colonna 3 a 72 non contengono spazi bianchi e la scheda successiva non ha a colonna 1 una C.

2) I caratteri non sono solo quelli ammessi dal codice esadecimale (1 2 3 4 5 6 7 8 9 A B C D E F).

3) Il codice a cui ci si riferisce e' gia' attivo nel file.

4) I primi 8 caratteri tradotti non indicano un codice corretto.

N.B.: Qualora il data set fosse combinato con una maschera (vedere il comando P) le stringhe di quaranta caratteri devono essere operazionate in modo opportuno perche' a tale stringa non verra' fatta nessuna operazione prima di essere scritta.

E Tale operazione e' simile all'operazione X tranne che i caratteri che vanno dati da colonna 3 a 42 sono gia' tradotti in EBCDIC e che gli spazi bianchi incontrati in tale campo vengono considerati facenti parte del campo stesso e quindi la scansione non termina prima di colonna 42.
Tale comando da' errore nei seguenti casi:

1) Il codice a cui si riferisce e' gia' attivo nel file.

2) I primi 4 caratteri non indicano un codice corretto.

O Tale operazione serve ad aggiornare un codice facendo l'"OVERLAY" tra cio' che e' specificato ed il blocco che si trova nel data set.
Da colonna 3 in poi devono essere scritte le parole chiave (vedi piu' avanti) indicanti il campo da modificare, seguite dal segno "=", dalle lettere X oppure C e dal campo racchiuso tra parentesi tonde. Il contenuto del campo sara' tradotto da esadecimale in EBCDIC se e solo se tra il segno "=" e la parentesi aperta avremo specificato la lettera X.
I vari campi devono essere divisi tra loro con delle virgole.
Qualora i campi da modificare non stessero su di una scheda, basta interrompere prima di colonna 72 terminando con una chiusa parentesi tonda ed una virgola e continuare nella scheda seguente con C a colonna 1, spazio ed operandi.
Gli spazi bianchi provocano l'arresto della scansione della scheda a meno che non si trovino tra parentesi con C davanti alla parentesi.
Viene stampato un messaggio di errore nei seguenti casi:

1) Non esiste l'indicazione del codice a cui ci si riferisce

$$\text{CODICE=} \begin{matrix} X \\ C \end{matrix} (\text{.....}),\text{.....}$$

2) Non e' stata chiusa correttamente una parentesi.

3) Una scheda finisce con chiusa parentesi virgola e la successiva non e' una continuazione.

4) Un campo tra parentesi e' di lunghezza diversa da quello che si aspetta il programma.

5) Il codice a cui si riferisce non e' attivo.

6) I caratteri tra parentesi non sono quelli previsti dal codice esadecimale nonostante che davanti alla parentesi vi sia la X.

A Questo comando si comporta esattamente come il comando O tranne che la sostituzione del tempo residuo (RESIDUAL) viene fatta per somma con il tempo trovato nel blocco e gli altri campi aggiornati con l'operazione di OR.

I Tale comando si comporta esattamente come il comando O tranne che il tipo di errore 5 viene dato quando il blocco a cui si riferisce e' gia' attivo.

D Questo comando rende inattivo un blocco di un codice, deve essere specificata a colonna 3 solo la parola CODICE, il segno "=" ed il codice specificato con X(.....) o C(.....), viene dato un messaggio di errore nei seguenti casi:

1) Il codice a cui si riferisce non e' attivo.

2) Il formato della scheda non e' corretto.

3) E' stata data una scheda continuazione a questo record.

4) I caratteri tra parentesi non indicano un codice corretto.

5) I caratteri tra parentesi non sono quelli usati dal codice esadecimale nonostante che davanti alla parentesi vi sia la X.

P Questo comando e' obbligatorio e deve esser daTO COME PRIMO COMANDO.
A colonna 1 deve essere specificata la lettera P, seguita da uno spazio ed a colonna tre la lettera C oppure X seguita da una stringa di caratteri tale che non superi i quaranta caratteri (o 80 se e' scritta esadecimale).
Tale stringa di caratteri viene concatenata con tanti caratteri aventi tutti i bits = 0 (LOW(..)) fino ad arrivare a quaranta caratteri e tutti i blocchi successivamente elaborati dal programma vengono trattati come se fossero stati operazionati con la suddetta stringa con l'operazione di EXCLUSIVE OR.
Percio' i blocchi prima di essere elaborati vengono decodificati e ridecodificati prima della scrittura.
Questo comando da' errore nei seguenti casi:

1) Non viene chiusa la parentesi prima di colonna 72 e la scheda successiva non e' una continuazione.

2) E' usato il carattere X che indica esadecimale ed i caratteri tra parentesi non sono solo quelli permessi dal codice esadecimale.

3) La stringa di caratteri non e' quella usata al momento della creazione del file. Questo errore provoca la fine del programma.

* Tale operazione serve per inserire commenti.

Le parole chiave che si possono usare sono :

CODICE specifica il codice che deve essere aggiornato. Tale specifica e' obbligatoria.

RESIDUAL specifica il limite di tempo in sec. a disposizione per passare applicazioni con quel codice.

LIMITI specifica come devono essere settati i limiti per il codice in questione. Per una dettagliata spiegazione di ciascun limite vedere Appendice B.

PASSWORD specifica la parola chiave associata ad un codice da inserire in tutti i programmi, secondo certe regole, dagli utilizzatori di quel codice (vedere Appendice C).

VOLUMI specifica su quali volumi un codice puo' allocare volumi di nome standard. I nomi dei volumi si trovano in una tabella (data set) e corrispondono ai bits a seconda di come sono ordinati (es. il primo bit corrisponde al primo volume in tabella, il secondo al secondo ...) (per spiegazione piu' dettagliata vedere nota interna CNUCE 106).

Tabella riassuntiva delle operazioni permesse in modo BATCH

Per iniziare la seduta di lavoro e' necessario fare come prima operazione l'esecuzione del comando "p".

Inserimento di un codice nuovo	operazione "X"
" " "	" "I"
" " "	" "E"
Aggiornamento di un codice esistente	" "O"
" " "	" "A"
Cancellazione di un codice esistente	" "D"
Nessuna azione	" "*"

Operazioni permesse dal programma di Manutenzione in modo Conversazionale

E Tale comando non e' permesso perche' non e' possibile digitare tutte le 256 combinazioni necessarie per questa operazione dalla tastiera di un terminale video.

* Tale comando non e' permesso perche' e' inutile fare dei commenti dal momento che le richieste sono guidate dal programma stesso.

P Questo comando deve ancora essere il primo comando da dare ma la password con cui viene dato il permesso a proseguire il lavoro puo' essere cambiata in qualsiasi momento senza toccare ne' le EXIT ne' il data set dei codici ne' la password con cui e' protretto il data set stesso.

? Tale comando permette di ottenere a console le informazioni principali riguardanti un codice, la password di quel codice, sia in forma esadecimale che carattere, i limiti, il tempo residual espresso in secondi e l'ultimo Job passato con quel codice. Da notare che questo comando e' tipico del modo conversazionale e quindi non e' permesso in Batch.

X (cfr. modo Batch)

D (cfr. modo Batch)

I (cfr. modo Batch)

O (cfr. modo Batch)

A (cfr. modo Batch)

Nel caso del comando "X" il programma chiedera' di digitare la stringa esadecimale completa, cioe' di 40 caratteri. Bisogna fare attenzione all'uso di questo comando in quanto non vengono fatti controlli formali su quanto digitato ad esclusione della lunghezza della stringa e della presenza di eventuali caratteri diversi da 0 1 2 ..... E F. Il codice verra' caricato solo se non risulta gia' abilitato sul data set dei codici.

Per gli altri comandi verra' sempre richiesto il codice che si intende inserire o variare o eliminare. In quest'ultimo caso la sola informazione del codice e' sufficiente per la sua eliminazione.

Digitato il codice, il programma richiedera' la password che come il codice puo' essere digitato sia in caratteri che in esadecimale. Da notare che in tutte le domande e' descritta anche la risposta o le risposte che si debbono dare per eliminare il piu' possibile errori. Vengono comunque effettuati anche controlli formali che controllano le risposte e che in caso di errori causano la ripetizione della domanda che ha causato l'errore.

Dopo la password vengono richiesti i LIMITI. Poiche' richiedere sempre tutti i limiti uno a uno diverrebbe laborioso e spesso inutile, si e' cercato, quando e' possibile di evitare cio'. Infine viene richiesta una stringa di 8 caratteri indicanti i VOLUMI.

Per i dettagli delle singole operazioni vedere appendice D.

CAP. 3.

Procedura CREACOD

La procedura e' utilizzata per richiamare il programma CREACOD. Tale programma esegue il formattamento dei records destinati a contenere le informazioni dei codici autorizzati a lavorare in BATCH/OS. La sk DD di nome filone punta al data set da formattare. Vedremo il significato dei vari parametri nella descrizione successiva.

La procedura puo' esser fatta partire sia da console del 158, sia tramite schede.

```
S IR,J=CREACOD,P='V=volume,D=dataset,DP=DISP,S=stampa'

// EXEC CREACOD,V=volume,D=dataset,DP=disp,S=stampa
```

dove:

volume .... indica il volume dove si trova il data set specificato nel parametro D, tale parametro e' obbligatorio. (Default DUMMY)

dataset ... indica il data set che contiene i records formattati, tale parametro e' obbligatorio. (Default DUMMY)

disp ..... indica il disposition del data set (NEW e' il valore per default e puo' essere cambiato in OLD).

stampa .... indica se vogliamo la stampa del data set (il valore assunto se omesso tale parametro e' NOSTAMPA e puo' essere cambiato in STAMPA). Il parametro puo' essere fornito anche nella forma abbreviata "STA".

```
//CREACOD  PROC S=NOSTAMPA,V=DUMMY,D=DUMMY,DP=NEW
//CREACOD  EXEC PGM=CREACOD,PARM='/&S'
//STEPLIB  DD   DSN=SYS1.MISURE,DISP=SHR
//         DD   DSN=SYS1.PLOLINK,DISP=SHR
//SYSPRINT DD   SYSOUT=A
//PLIDUMP  DD   SYSOUT=A
//FILONE   DD   UNIT=SYSDA,VOL=SER=&V,DSNAME=&D,
// SPACE=(CYL,(5)),DCB=(DSORG=DA),DISP=(&DP,KEEP)
```

Procedura CARICOD

Questa procedura viene utilizzata per richiamare il programma CARICOD. Tale programma, provvede a caricare, nel data set puntato dalla DD di nome FILONE, i codici autorizzati a lavorare in BATCH/OS.

Tali codici portano anche le informazioni sull' uso che ciascuno di essi puo' fare delle risorse del sistema.

La procedura puo' esser fatta partire sia da console del 158 sia da schede.

```
S IP,J=CARICOD,P='V=volume,D=dataset,S=stampa,P=perfora'

// EXEC CREACOD,V=volume,D=dataset,S=stampa,P=perfora
```

dove:

volume .... indica il volume che contiene il dataset dei codici, parametro obbligatorio. (Default DUMMY)

dataset ... indica il nome del dataset relativo ai codici, parametro obbligatorio. (Default DUMMY)

stampa .... indica se vogliamo la stampa del data set (il valore assunto se omesso tale parametro e' NOSTAMPA e puo' essere cambiato in STAMPA). Il parametro puo' essere fornito anche nella forma abbreviata "STA".

perfora ... indica se vogliamo la perforazione delle schede tipo "F" (il valore assunto se omesso tale parametro e' DUMMY e puo' essere cambiato in perfora). Il parametro puo' essere fornito anche nella forma abbreviata "PERF".

Inoltre, e' possibile eseguire la stampa e la perforazione del dataset specificando, al momento in cui e' richiamata la procedura, i parametri S=STAMPA e P=PERFORA. Anche per questi parametri i valori per default sono DUMMY, stando a significare che in tal caso non viene eseguita ne' la stampa ne' la perforazione.

```
//CARICOD  PROC V=DUMMY,D=DUMMY,S=DUMMY,P=DUMMY
//CARICOD  EXEC PGM=CARICOD,PARM='/&S,&P',REGION=128K
//STEPLIB  DD DSN=SYS1.MISURE,DISP=SHR
//         DD DSN=SYS1.PLOLINK,DISP=SHR
//         DD DSN=SYS1.PLRLIB2,DISP=SHR
//FILONE   DD UNIT=SYSDA,VOL=SER=&V,DSNAME=&D,
//          DCB=(DSORG=DA),DISP=(SHR,KEEP)
//SYSPRINT DD SYSOUT=A
//PLIDUMP  DD SYSOUT=A
//SCHEDE   DD SYSOUT=B,DCB=(RECFM=F,BLKSIZE=80)
```

Procedura DUMPCOD

La procedura e' utilizzata per richiamare il programma DUMPCOD. Lo scopo del programma e' di eseguire una stampa nel formato carattere e quindi di facile consultazione del data set che contiene i codici. La stampa e' percio' utilizzata per controlli, da parte del sistemista addetto, e per l' aggiornamento dei codici stessi.

La procedura puo' esser fatta partire sia da console del 158 sia da schede.

```
S IR,J=DUMPCOD,P='V=volume,D=dataset,S=stampa,P=perfora'

// EXEC DUMPCOD,V=volume,D=dataset,S=stampa,P=perfora
```

dove:

volume .... indica il volume che contiene il dataset dei codici, tale parametro e' obbligatorio. (Default DUMMY)

dataset ... indica il nome del dataset relativo ai codici, tale parametro e' obbligatorio. (Default DUMMY)

stampa .... indica se vogliamo la stampa del data set (il valore assunto se omesso tale parametro e' NOSTAMPA e puo' essere cambiato in STAMPA). Il parametro puo' essere fornito anche nella forma abbreviata "STA".

perfora ... indica se vogliamo la perforazione (il valore assunto se omesso tale parametro e' DUMMY e puo' essere cambiato in perfora). Il parametro puo' essere fornito anche nella forma abbreviata "PERF".

Inoltre, e' possibile eseguire la stampa e la perforazione del dataset specificando, al momento in cui e' richiamata la procedura, i parametri S=STAMPA e P=PERFORA, oppure nella forma abbreviata S=STA e P=PERF. Anche per questi parametri i valori per default sono DUMMY, stando a significare che in tal caso non viene eseguita ne' la stampa ne' la perforazione.

```
   //DUMPCOD  PROC S=DUMMY,P=DUMMY,D=DUMMY,V=DUMMY,OUT=B
//          EXEC PGM=DUMPCOD,PARM='/&S,&P',REGION=128K
//STEPLIB   DD   DSN=SYS1.MISURE,DISP=SHR
//          DD   DSN=SYS1.PLOLINK,DISP=SHR
//          DD   DSN=SYS1.PLRLIB2,DISP=SHR
//FILONE    DD   UNIT=SYSDA,VOL=SER=&V,DSNAME=&D,DISP=SHR
//SCHEDE    DD   SYSOUT=&OUT,
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSPRINT  DD   SYSOUT=A
//PLIDUMP   DD   SYSOUT=A
```

CAP. 4.

```
STMT LEV NT
                /****     PROGRAMMA CREACOD     *****/

   1      0   CREACOD: PROC (PARM) OPTIONS (MAIN);
   2    1 0   ON ERROR
                      BEGIN;
   3    2 0           DCL ONCODE BUILTIN;
   4    2 0           IF ONCODE = 9050
                           THEN DO;
   5    2 1                PUT EDIT ('IL DISCO E'' PIENO: QUANTO MI FAI SCRIVERE?.',
                            'HO SCRITTO ',CHIAVE,' RECORDS') (A,SKIP,A,P'ZZZZZZZ9',A);
   6    2 1                GO TO FINE;
   7    2 1                END;
   8    2 0           PUT EDIT ('PECCATO!!   ONCODE=',ONCODE) (A) SKIP (3);
   9    2 0           GO TO FINE;
  10    2 0           END;
  11    1 0   ON ENDFILE       (SYSIN) GO TO NOPASSWORD;
  12    1 0   ON UNDEFINEDFILE (SYSIN) GO TO NOPASSWORD;
  13    1 0   GET FILE (SYSIN) EDIT (STR80) (A(80));
  14    1 0   IF SUBSTR(STR80,1,4)  = 'P C('
              THEN DO;
  15    1 1           PUT SKIP LIST('ERRORE: SCHEDA DI INPUT NON CORRETTA');
  16    1 1           PUT SKIP (3);
  17    1 1           GO TO FINE;
  18    1 1           END;
  19    1 0      ELSE DO;
  20    1 1           PASS=SUBSTR(STR80,5,40);
  21    1 1           PASSWORD='1'B;
  22    1 1           END;
  23    1 0   Q=ADDR(PASS);
  24    1 0      NOPASSWORD: DO CHIAVE = 0 TO 1039;
  25    1 1           DO J = 0 TO 24;
  26    1 2           BLOCCO(J).CODICE_LETTERA = ALFABETO (CHIAVE/40+1);
  27    1 2           BLOCCO(J).CODICE_NUMERO = MOD (CHIAVE,40) * 25 + J;
  28    1 2           BLOCCO(J).LIMITI(*) = '0'B;
  29    1 2           BLOCCO(J).LIMITI(1) = '0'B;
  30    1 2           BLOCCO(J).RESIDUAL = 0;
  31    1 2           BLOCCO(J).PASSWORD = LOW(8);
  32    1 2           BLOCCO(J).VOLUMI(*) = '0'B;
  33    1 2           BLOCCO(J).FILLER = '';
  34    1 2           P=ADDR(BLOCCO(J));
  35    1 2           IF PASSWORD & PARM='STAMPA'
                         THEN  CALL PUTDUMP (RECDEFSTR);
  36    1 2           IF PASSWORD
                         THEN  RECDEFBIT=(RECDEFBIT|PASSDEF)& (RECDEFBIT&PASSDEF);
  37    1 2           IF PARM='STAMPA'
```

```
                  THEN CALL PUTDUMP (RECDEFSTR);
30   1  2          END;
39   1  1       WRITE FILE (FILONE) FROM (REC) KEYFROM(CHIAVE);
40   1  1       END NOPASSWORD;
41   1  0    P2=ADDR(PAROLACCE);
42   1  0    IF PASSWORD
               THEN DO;
43   1  1          PUT SKIP EDIT(ESADECI(PASS),'<==== PASS PRIMA DELL''OP.')(A);
44   1  1          PASSDEF=(PASSDEF|PASS2BIT)& (PASSDEF&PASS2BIT);
45   1  1          PUT SKIP EDIT(ESADECI(PASS),'<==== PASS DOPO L''OP.')(A);
46   1  1          WRITE FILE (FILONE) FROM (PASS   ) KEYFROM (1040);
47   1  1          END;
48   1  0       ELSE WRITE FILE (FILONE) FROM (BITTI00) KEYFROM (1040);

49   1  0    DECLARE
               1 REC UNALIGNED,
                 2 BLOCCO (0:24),
                   3 CODICE,
                     4 LETTERA CHAR(1),
                     4 NUMERO  PICTURE '999',
                   3 LIMITI(32)   BIT(1),
                   3 RESIDUAL     BIN FIXED (31),
                   3 PASSWORD     CHAR(8) INIT(''),
                   3 VOLUMI(32)   BIT(1) INIT((32) '0'B),
                   3 FILLER  CHAR(16),
                 CHIAVE    PICTURE '(8)9',
                 PUTDUMP ENTRY(CHAR(4096) VARYING),
                 ESADECI ENTRY(CHAR(100) VARYING) RETURNS (CHAR(200)VAR),
                 FILONE    FILE RECORD OUTPUT KEYED
                             ENV (REGIONAL(1) F RECSIZE (1000) BLKSIZE (1000)),
                 RECDEFBIT BIT(320) BASED (P),
                 PASS2BIT BIT(320) BASED (P2),
                 BITTI00   CHAR(40) INIT(LOW(40)),
                 RECDEFSTR CHAR(40) BASED(P),
                 PAROLACCE CHAR(40)
                      INIT('  PAROLA CHIAVE  '),
                 PARM      CHAR (100) VAR,
                 STR80     CHAR (80),
                 PASS CHAR(1000),
                 PASSDEF BIT(320) BASED(Q),
                 PASSWORD       BIT(1) INITIAL ('0'B),
                 ALFABETO(26)   CHAR(1) DEF ALFABETONE,
                 ALFABETONE CHAR(26) INIT ('ABCDEFGHIJKLMNOPQRSTUVWXYZ');
50   1  0    FINE: END CREACOD;
```

```
STMT LEV NT

                /********** PROGRAMMA DI AGGIORNAMENTO DPL FILE SYS1.MOGCOD ******/

  1       0     CARICA: PROC (EXTERNAL_PARM) OPTIONS(MAIN);
                /*** EXTERNAL_PARM CONTROLLA LE STAMPE P LE PERFORAZIONI ****/
                /***                SE CONTIENE LA STRINGA "STA" STAMPA   ****/
                /***                SE CONTIENE LA STRINGA "PERF" PERFORA ****/
  2     1   0   ON ENDFILE (SYSIN) GO TO DUMPA;
  3     1   0   OPEN FILE(FILONE) DIRECT UPDATE;
  4     1   0   ON UNDEFINEDFILE(SYSIN)
                   BEGIN;
  5     2   0      BATCH='0'B;
  6     2   0      GO TO CICLO1;
  7     2   0      END;
  8     1   0   CICLO1: /**** LETTURA OPERAZIONE DA ESEGUIRE ****/
                  /*************************************************************************/
                IF FATTO_ENQ
                   THEN DO;
  9     1   1           CALL FAIDEQ;
 10     1   1           FATTO_ENQ='0'B;
 11     1   1           END;
                  /*************************************************************************/
 12     1   0   IF BATCH
                   THEN CALL LEGGI;
 13     1   0      ELSE CALL CONVERSA;
                /*************************************************************************/
 14     1   0   CALL FAIENQ;
 15     1   0   FATTO_ENQ='1'B;
                /*************************************************************************/
 16     1   0   GO TO OPERA(N_OPERAZIONE);

                /*** ESECUZIONE DELLE VARIE OPERAZIONI POSSIBILI ***/

 17     1   0   OPERA(0):  /*** OPERAZIONE SBAGLIATA ***/
                PUT EDIT('TIPO OPERAZIONE SCONOSCIUTO: "',OPERAZIONE,'"')(SKIP,3 A);
 18     1   0   CALL PUTDUMP(SCHEDA1);
 19     1   0   PUT SKIP(3);
 20     1   0   SCHEDA1=SCHEDA2;
 21     1   0   GO TO CICLO1;

 22     1   0   OPERA(1):  /**** OPERAZIONE X ****/

                PUT SKIP EDIT(OPERAZIONE,STR80)(A,X(1),A);
 23     1   0   STR80=EBCDIC (STR80);
```

```
24   1  0  OPERA(2):   /***** OPERAZIONE E *******/

           CODEX=SUBSTR(STR40,1,4);
25   1  0  LETTERA=CODEX;
26   1  0  NUMERO=SUBSTR(CODEX,2,3);
27   1  0  CHIAVE=((INDEX(ALFABETONE,LETTERA)-1)*40)+TRUNC(NUMERO/25);
           /** DISPLAY('INSERISCO CODICE '||CODEX||' CON CHIAVE '||CHIAVE); **/
28   1  0  READ FILE (FILONE) INTO (REC) KEY (CHIAVE);
29   1  0  INDICE=MOD(NUMERO,25);
           /** DISPLAY ('INDICE='||INDICE);                                **/
30   1  0  P0=ADDR(BLOCCO_LETTO);
31   1  0  STR40B=STR40;
32   1  0  P0=ADDR(BLOCCO(INDICE));
33   1  0  STR40B=ORX_40CHAR(PASS,STR40B);
34   1  0  IF BLOCCO(INDICE).LIMITI(1)
               THEN DO;
35   1  1            IF  BATCH THEN DISPLAY
                         ('*** ERRORE ***       TENTATA RISCRITTURA '||
                         'DEL CODICE '||CODEX);
36   1  1            PUT EDIT('*** ERRORE ***       TENTATA RISCRITTURA ',
                         'DEL CODICE ',CODEX)(SKIP(3),3 A);
37   1  1            GO TO FINE_OPERAZIONE_X;
38   1  1            END;
39   1  0  BLOCCO_LETTO.LIMITI(1)='1'B;  /** FORZO CONDIZIONE DI CODICE ATTIVO **/
40   1  0  BLOCCO(INDICE)=BLOCCO_LETTO, BY NAME;
41   1  0  CALL PUTDUMP(STR40B);
42   1  0  STR40B=ORX_40CHAR(PASS,STR40B);
43   1  0  REWRITE FILE (FILONE) FROM (REC) KEY (CHIAVE);
44   1  0  FINE_OPERAZIONE_X:
           GO TO CICLO1;

45   1  0  OPERA(3):    /**** OPERAZIONE O *****/
           OPERA(4):    /**** OPERAZIONE A *****/
           OPERA(5):    /**** OPERAZIONE I *****/

           PUT SKIP EDIT(OPERAZIONE,STR80)(A,X(1),A);
46   1  0  IF INDEX(STR80,'CODICE=')=0
              THEN DO;
47   1  1         IF  BATCH THEN
                    DISPLAY('CODICE MANCANTE O ERRONEAMENTE SCRITTO');
48   1  1         PUT SKIP EDIT('CODICE MANCANTE O ERRONEAMENTE SCRITTO')
                       (A);
49   1  1         GO TO CICLO1;
50   1  1         END;
51   1  0  K=INDEX(STR80,'CODICE=C(');
```

```
52   1  0  IF K=0
             THEN DO;
53   1  1          K=INDEX(STR80,'CODICE=X(');
54   1  1          IF K=0
                       THEN DO;
55   1  2                  IF  BATCH THEN
                           DISPLAY('CODICE MANCANTE O ERRONEAMENTE SCRITTO');
56   1  2                  PUT SKIP EDIT('"CODICE" MANCANTE O ERRONEAMENTE SCRITTO')
                               (A);
57   1  2                  GO TO CICLO1;
58   1  2                  END;
59   1  1          CODEX=EBCDIC(SUBSTR(STR80,K+9,8));
60   1  1          STR80=SUBSTR(STR80,1,K-1)||SUBSTR(STR80,K+19);
61   1  1          END;
62   1  0     ELSE DO;
63   1  1          CODEX=SUBSTR(STR80,K+9,4);
64   1  1          STR80=SUBSTR(STR80,1,K-1)||SUBSTR(STR80,K+15);
65   1  1          END;
66   1  0  CHIAVE=((INDEX(ALFABETONE,SUBSTR(CODEX,1,1))-1)*40)
                    +TRUNC(SUBSTR(CODEX,2,3)/25);
67   1  0  READ FILE (FILONE) INTO (REC) KEY (CHIAVE);
68   1  0  INDICE=MOD(SUBSTR(CODEX,2,3),25);
69   1  0  P0=ADDR(BLOCCO(INDICE));
70   1  0  STR40B=ORX_40CHAR(PASS,STR40B);
71   1  0  IF (OPERAZIONE='O' &  BLOCCO(INDICE).LIMITI(1)) |
             (OPERAZIONE='I' &  BLOCCO(INDICE).LIMITI(1)) |
             (OPERAZIONE='A' &  BLOCCO(INDICE).LIMITI(1))
             THEN DO;
72   1  1          IF  BATCH THEN
                   DISPLAY ('ESISTENZA O INESISTENZA CODICE INCOMPATIBILE'||
                               ' CON RICHIESTE "I" "O" "A"');
73   1  1          PUT SKIP EDIT('ESISTENZA O INESISTENZA CODICE INCOMPATIBILE',
                               ' CON RICHIESTE "I" "O" "A"')
                               (A);
74   1  1          IF  BATCH THEN
                   DISPLAY('RICHIESTA IGNORATA');
75   1  1          PUT SKIP LIST('RICHIESTA IGNORATA');
76   1  1          GO TO CICLO1;
77   1  1          END;
```

```
 78   1  0  IF BLOCCO(INDICE).LIMITI(1)='0'B  /*SE CODICE ESISTE GIA' SU FILE */
                THEN BLOCCO(INDICE)='';      /*DISABILITATO SI AZZERA PRIMA DI
                                               RICREARLO*/
 79   1  0  CICLO: L=LENGTH(STR80);
 80   1  0  IF L<3 THEN GO TO RISCRIVI;
 81   1  0  IF SUBSTR(STR80,1,8)=',LIMITI='
             THEN STR80=SUBSTR(STR80,2);
 82   1  0  IF SUBSTR(STR80,1,7)='LIMITI='
             THEN DO;
 83   1  1       IF SUBSTR(STR80,1,9)='LIMITI=('
                  THEN DO;
 84   1  2            STRINGABIT=UNSPEC(SUBSTR(STR80,10,4));
 85   1  2            BLOCCO(INDICE).LIMITI(*)=VETTOREBIT;
 86   1  2            IF LENGTH(STR80)>16
                       THEN STR80=SUBSTR(STR80,16);
 87   1  2            ELSE STR80='';
 88   1  2            END;
 89   1  1       ELSE DO;
 90   1  2            STRINGABIT=UNSPEC(EBCDIC(SUBSTR(STR80,10,8)));
 91   1  2            BLOCCO(INDICE).LIMITI(*)=VETTOREBIT;
 92   1  2            IF LENGTH(STR80)>20
                       THEN STR80=SUBSTR(STR80,20);
 93   1  2            ELSE STR80='';
 94   1  2            END;
 95   1  1       END;
 96   1  0  IF SUBSTR(STR80,1,8)=',RESIDUAL='
             THEN STR80=SUBSTR(STR80,2);
 97   1  0  IF SUBSTR(STR80,1,9)='RESIDUAL='
             THEN DO;
 98   1  1       IF SUBSTR(STR80,1,1)=',' THEN STR80=SUBSTR(STR80,2);
 99   1  1       K=INDEX(STR80,',');
100   1  1       IF K=0
                  THEN K=LENGTH(STR80)+1;
101   1  1       RESIDUAL_TIME=SUBSTR(STR80,10,K-10);
102   1  1       RESIDUAL_TIME=RESIDUAL_TIME*100; /*CENTESIMI DI SECONDO*/
103   1  1       IF OPERAZIONE  ='A'
                  THEN BLOCCO(INDICE).RESIDUAL=RESIDUAL_TIME;
104   1  1       ELSE BLOCCO(INDICE).RESIDUAL=BLOCCO(INDICE).RESIDUAL+
                                               RESIDUAL_TIME;
105   1  1       IF K+1<LENGTH(STR80)
                  THEN STR80=SUBSTR(STR80,K+1);
106   1  1       ELSE STR80='';
107   1  1       END;
```

```
108   1  0  IF SUBSTR(STR80,1,8)=',PASSWORD='
            THEN STR80=SUBSTR(STR80,2);
109   1  0  IF SUBSTR(STR80,1,9)='PASSWORD='
            THEN DO;
110   1  1        IF SUBSTR(STR80,1,11)='PASSWORD=C('
                  THEN DO;
111   1  2              BLOCCO(INDICE).PASSWORD=SUBSTR(STR80,12,8);
112   1  2              IF LENGTH(STR80)>22
                        THEN STR80=SUBSTR(STR80,22);
113   1  2                 ELSE STR80='';
114   1  2              END;
115   1  1           ELSE DO;
116   1  2              BLOCCO(INDICE).PASSWORD=EBCDIC(SUBSTR(STR80,12,16));
117   1  2              IF LENGTH(STR80)>26
                        THEN STR80=SUBSTR(STR80,26);
118   1  2                 ELSE STR80='';
119   1  2              END;
120   1  1        END;
121   1  0  IF SUBSTR(STR80,1,8)=',VOLUMI='
            THEN STR80=SUBSTR(STR80,2);
122   1  0  IF SUBSTR(STR80,1,7)='VOLUMI='
            THEN DO;
123   1  1        IF SUBSTR(STR80,1,9)='VOLUMI=C('
                  THEN DO;
124   1  2              STRINGABIT=UNSPEC(SUBSTR(STR80,10,4));
125   1  2              BLOCCO(INDICE).VOLUMI(*)=VETTOREBIT;
126   1  2              IF LENGTH(STR80)>16
                        THEN STR80=SUBSTR(STR80,16);
127   1  2                 ELSE STR80='';
128   1  2              END;
129   1  1           ELSE DO;
130   1  2              STRINGABIT=UNSPEC(EBCDIC(SUBSTR(STR80,10,8)));
131   1  2              BLOCCO(INDICE).VOLUMI(*)=VETTOREBIT;
132   1  2              IF LENGTH(STR80)>20
                        THEN STR80=SUBSTR(STR80,20);
133   1  2                 ELSE STR80='';
134   1  2              END;
135   1  1        END;
```

```
136   1  0  IF L<=LENGTH(STR80)
               THEN DO;
137   1  1         PUT SKIP EDIT('NON RIESCO A INTERPRETARE LA STRINGA ',
                  STR80)(2 A);
138   1  1         IF  BATCH THEN DISPLAY
                  ('NON RIESCO A INTERPRETARE LA STRINGA ' || STR80);
139   1  1         END;
140   1  0      ELSE GO TO CICLO;
141   1  0  RISCRIVI:   /*** RISCRITTURA DEL BLOCCO CORRETTO ****/
             BLOCCO(INDICE).LIMITI(1)='1'B;  /* PORZO CONDIZIONE DI CODICE ATTIVO */
142   1  0  IF OPERAZIONE='I'
                     THEN DO;
143   1  1                   BLOCCO(INDICE).CODICE.LETTERE=CODEX;
144   1  1                   BLOCCO(INDICE).CODICE.NUMERI=SUBSTR(CODEX,2);
145   1  1                   END;
146   1  0  P0=ADDR(BLOCCO(INDICE));
147   1  0  CALL PUTDUMP(STR40B);
148   1  0  STR40B=ORX_40CHAR(PASS,STR40B);
149   1  0  REWRITE FILE (FILONE) FROM (REC) KEY (CHIAVE);
150   1  0  GO TO CICLO1;

151   1  0  OPERA(6):     /**** OPERAZIONE D *****/

             PUT SKIP EDIT(OPERAZIONE,STR80)(A,X(1),A);
152   1  0  IF  BATCH THEN DISPLAY (OPERAZIONE||STR80);
153   1  0  IF INDEX(STR80,'CODICE=')=0
               THEN DO;
154   1  1         IF  BATCH THEN
                  DISPLAY ('CODICE MANCANTE O ERRONEAMENTE SCRITTO');
155   1  1         PUT SKIP EDIT('CODICE MANCANTE O ERRONEAMENTE SCRITTO')
                        (A);
156   1  1         GO TO CICLO1;
157   1  1         END;
158   1  0  K=INDEX(STR80,'CODICE=C(');
159   1  0  IF K=0
               THEN DO;
160   1  1         K=INDEX(STR80,'CODICE=X(');
161   1  1         IF K=0
                     THEN DO;
162   1  2                IF  BATCH THEN
                         DISPLAY ('CODICE MANCANTE O ERRONEAMENTE SCRITTO');
163   1  2                PUT SKIP EDIT('"CODICE" MANCANTE O ERRONEAMENTE SCRITTO')
                              (A);
164   1  2                GO TO CICLO1;
165   1  2                END;
166   1  1         CODEX=EBCDIC(SUBSTR(STR80,K+9,8));
167   1  1         STR80=SUBSTR(STR80,1,K-1)||SUBSTR(STR80,K+19);
168   1  1         END;
```

```
169   1  0     ELSE DO;
170   1  1          CODEX=SUBSTR(STR80,K+9,4);
171   1  1          STR80=SUBSTR(STR80,1,K-1)||SUBSTR(STR80,K+15);
172   1  1          END;
173   1  0   CHIAVE=((INDEX(ALFABETONE,SUBSTR(CODEX,1,1))-1)*40)
                     +TRUNC(SUBSTR(CODEX,2,3)/25);
174   1  0   READ FILE (FILONE) INTO (REC) KEY (CHIAVE);
175   1  0   INDICE=MOD(SUBSTR(CODEX,2,3),25);
176   1  0   P0=ADDR(BLOCCO(INDICE));
177   1  0   STR40B=ORX_40CHAR(PASS,STR40B);
178   1  0   IF   BLOCCO(INDICE).LIMITI(1)
              THEN DO;
179   1  1          IF  BATCH THEN
                     DISPLAY ('INESISTENZA CODICE INCOMPATIBILE'||
                              ' CON RICHIESTA D');
180   1  1          PUT SKIP EDIT('INESISTENZA CODICE INCOMPATIBILE',
                              ' CON RICHIESTA "D"')
                              (A);
181   1  1          END;

                        /*** RISCRITTURA DEL BLOCCO CORRETTO ****/

182   1  0   BLOCCO(INDICE).LIMITI(1)='0'B;  /* FORZO CONDIZ. DI CODICE INATTIVO */
183   1  0   P0=ADDR(BLOCCO(INDICE));
184   1  0   CALL PUTDUMP(STR40B);
185   1  0   STR40B=ORX_40CHAR(PASS,STR40B);
186   1  0   REWRITE FILE (FILONE) FROM (REC) KEY (CHIAVE);
187   1  0   IF  BATCH  THEN DISPLAY ('***CODICE '||CODEX||' CANCELLATO***');
188   1  0   PUT SKIP EDIT ('***CODICE ',CODEX,' CANCELLATO***') (A);
189   1  0   GO TO CICLO1;
```

```
190   1  0  OPERA(7):    /******* PASSWORD ********/

             IF SUBSTR(SCHEDA1,4,1) ='('
                THEN GO TO ERRORE_DI_PARENTESI;
191   1  0  IF SUBSTR(SCHEDA1,3,1) ='C' & SUBSTR(SCHEDA1,44,1) =')'
                THEN GO TO ERRORE_DI_PARENTESI;
192   1  0  READ FILE (FILONE) INTO (STRINGACCIA) KEY (1040);
193   1  0  PASS=OBX_40CHAR(PAROLACCE,STRINGACCIA);
194   1  0  STR40=SUBSTR(SCHEDA1,5,40);
195   1  0  IF PASS=STR40
                THEN GO TO CICLO1;
196   1  0     ELSE DO;
197   1  1          PUT EDIT('PASSWORD NON CORRETTA')(A)SKIP;
198   1  1          CALL PLIRETC(16);
199   1  1          STOP;
200   1  1          END;

201   1  0  OPERA(8):          /***** SALTO UNA SCHEDA *****/

              /****** COMMENTO ******/
             GO TO CICLO1;

202   1  0  OPERA(9):          /***** INFORMAZIONI SUL CODICE *******/

             IF INDEX(STR80,'CODICE=')=0
                THEN DO;
203   1  1          IF  BATCH THEN
                     DISPLAY ('CODICE MANCANTE O ERRONEAMENTE SCRITTO');
204   1  1          PUT SKIP EDIT('CODICE  MANCANTE O ERRONEAMENTE SCRITTO')
                          (A);
205   1  1          GO TO CICLO1;
206   1  1          END;
207   1  0  K=INDEX(STR80,'CODICE=C(');
```

```
208   1  0  IF K=0
               THEN DO;
209   1  1          K=INDEX(STR80,'CODICE=X('');
210   1  1          IF K=0
                         THEN DO;
211   1  2                  IF  BATCH THEN
                           DISPLAY ('CODICE MANCANTE O ERRONEAMENTE SCRITTO');
212   1  2                  PUT SKIP EDIT('"CODICE" MANCANTE O ERRONEAMENTE SCRITTO')
                                 (A);
213   1  2                  GO TO CICLO1;
214   1  2                  END;
215   1  1          CODEX=EBCDIC(SUBSTR(STR80,K+9,8));
216   1  1          STR80=SUBSTR(STR80,1,K-1)||SUBSTR(STR80,K+19);
217   1  1          END;
218   1  0     ELSE DO;
219   1  1          CODEX=SUBSTR(STR80,K+9,4);
220   1  1          STR80=SUBSTR(STR80,1,K-1)||SUBSTR(STR80,K+15);
221   1  1          END;
222   1  0  CHIAVE=((INDEX(ALFABETONE,SUBSTR(CODEX,1,1))-1)*40)
                     +TRUNC(SUBSTR(CODEX,2,3)/25);
223   1  0  READ FILE (FILONE) INTO (REC) KEY (CHIAVE);
224   1  0  INDICE=MOD(SUBSTR(CODEX,2,3),25);
225   1  0  P0=ADDR(BLOCCO(INDICE));
226   1  0  STR40B=ORX_40CHAR(PASS,STR40B);
227   1  0  IF BLOCCO(INDICE).LIMITI(1) = '0'B
               THEN DO;
228   1  1          DISPLAY ('IL CODICE= ' || SUBSTR (STR40B,1,4)
                           || ' NON E'' ABILITATO ');
229   1  1                  GO TO CICLO1;
230   1  1          END;
231   1  0  DISPLAY ('CODICE= '   || SUBSTR (STR40B,1,4));
232   1  0  DISPLAY ('LIMITI= '   || ESADECI (SUBSTR (STR40B,5,4)));
233   1  0  KKK=UNSPEC(SUBSTR(STR40B,9,4));
234   1  0  KKK=KKK/100;
235   1  0  DISPLAY ('RESIDUAL= ' || KKK);
236   1  0  DISPLAY ('PASSWORD= ' || SUBSTR (STR40B,13,8)||
                '   '||ESADECI (SUBSTR (STR40B,13,8)));
237   1  0  DISPLAY ('SPYJOB=' || SUBSTR (STR40B,25,16));
238   1  0  GO TO CICLO1;
```

```
239   1  0  DUMPA:              /* DUMP DELLA SITUAZIONE FINALE */

            /*************************************************************************
           CALL FAIENQ;
           PASTO_ENQ='1'B;
            /*************************************************************************
           CLOSE FILE (FILONE);
           OPEN FILE (FILONE) SEQUENTIAL INPUT;
              DO CHIAVE=0 TO 5;
              READ FILE (FILONE) INTO (STR1000)  * KEY (CHIAVE) *  ;
                   DO K=1 TO 25;
                   P0=ADDR(STR1000DEF(K));
                   STR40B=ORX_40CHAR(PASS,STR40B);
                   END;
              PUT SKIP (5) EDIT ('RECORD NUMERO ',CHIAVE)(A,F(8));
              CALL PUTDUMP (STR1000);
              END;
           CLOSE FILE (FILONE);
           CALL DUMP(EXTERNAL_PARM);          ***** CHIAMATA DEL PGM DI DUMP ****/
           GO TO FINE;
240   1  0  ERRORE_DI_PARENTESI:
           PUT SKIP(5) EDIT('IMPARA A PERFORARE CORRETTAMENTE LE PARENTESI!!!',
              OPERAZIONE,SUBSTR(SCHEDA1,4,1),STR40,SUBSTR(SCHEDA1,44,1))
              (A,SKIP,A(1),X(1),A(1),A(40),A(1));
```

```
241   1  0  DECLARE
               1 REC UNALIGNED,
                    2 BLOCCO (0:24),
                         3 CODICE,
                              4 LETTERE CHAR(1),
                              4 NUMERI  PICTURE '999',
                         3 LIMITI(32)    BIT(1),
                         3 RESIDUAL      BIN FIXED (31),
                         3 PASSWORD      CHAR(8) INIT(''),
                         3 VOLUMI(32)    BIT(1)  INIT((32) '0'B),
                         3 FILLER  CHAR(16),
               1 BLOCCO_LETTO,
                         2 CODICE,
                              3 LETTERE CHAR(1),
                              3 NUMERI  PICTURE '999',
                         2 LIMITI(32)    BIT(1),
                         2 RESIDUAL      BIN FIXED (31),
                         2 PASSWORD      CHAR(8) INIT(''),
                         2 VOLUMI(32)    BIT(1) INIT((32) '0'B),
                         2 FILLER  CHAR(16),
                    FILONE     FILE RECORD KEYED
                                    ENV (REGIONAL(1) F RECSIZE (1000) BLKSIZE (1000)),
                    OPERAZIONE   CHAR (1),
                    PRIMAVOLTA     BIT(1) INIT('1'B),
                    FATTO_ENQ      BIT(1) INIT('0'B),
                    BATCH          BIT(1) INIT('1'B),
                    N_OPERAZIONE   BIN FIXED,
                    OPERA(0:10)    LABEL,
                    LETTERA        CHAR(1),
                    EXTERNAL_PARM  CHAR(100) VARYING,
                    DUMP           ENTRY(CHAR(100)VAR),
                    EBCDIC         ENTRY(CHAR(100)VAR) RETURNS(CHAR(50) VAR),
                    NUMERO  PICTURE '999',
                    PUTDUMP ENTRY (CHAR(2000)VAR),
                    CHIAVE    PICTURE 'ZZZZZZZ9' INIT (0),
                    CODEX     CHAR(4),
                    STR40     CHAR(40),
                    STR80     CHAR(800) VARYING,
                    SCHEDA1   CHAR(800) VARYING,
                    SCHEDA2   CHAR(800) VARYING,
                    STR40B    CHAR(40) BASED(P0),
                    PASS      CHAR(40),
                    STRINGABIT BIT(48),
                    RESIDUAL_TIME BIN FIXED (31,0),
                    KKK BIN FIXED (31,0),
                    VETTOREBIT(32) BIT(1) DEFINED STRINGABIT POSITION (17),
                   /*** ELIMINA 2 BYTES DI LUNGHEZZA DELL'UNSPEC***/
                    ESADECI ENTRY (CHAR(100) VAR) RETURNS(CHAR(200) VAR),
                    STRINGACCIA CHAR(1000),
```

```
                PAROLACCE CHAR(40)
                     INIT('   PAROLA CHIAVE  '),
                TIPO   CHAR(1),
                (P0,P1,P2)  POINTER,
                STR1000   CHAR(1000),
                STR1000DEF(25) CHAR(40) DEF STR1000,
                ALFABETO(26)   CHAR(1) DEF ALFABETONE,
                ALFABETONE CHAR(26) INIT ('ABCDEFGHIJKLMNOPQRSTUVWXYZ');
242   1  0  OFX_40CHAR: PROC (A,B) RETURNS(CHAR(40));
           /*****************************************************************
            *  QUESTA FUNZIONE ESEGUE L'OPERAZIONE DI OR ESCLUSIVO TRA DUE   *
            *  STRINGHE DI CARATTERI LUNGHE 40 OGNUNA                        *
            *****************************************************************/
243   2  0  PA=ADDR(A);
244   2  0  PB=ADDR(B);
245   2  0  PC=ADDR(C);
246   2  0  CBIT=(ABIT|BBIT)& (ABIT&BBIT);
247   2  0  RETURN (C);
248   2  0  DECLARE
                (PA,PB,PC)      POINTER,
                (A,B,C)         CHAR(40),
                ABIT       BIT(320) BASED(PA),
                BBIT       BIT(320) BASED(PB),
                CBIT       BIT(320) BASED(PC);
249   2  0  END OFX_40CHAR;
```

```
250   1  0  LEGGI: PROC;
251   2  0  IF PRIMAVOLTA
               THEN DO;
252   2  1       ON ENDFILE(SYSIN) SIGNAL UNDEFINEDFILE(SYSIN);
253   2  1       GET EDIT(SCHEDA1)(COL(1),A(80));
254   2  1       PRIMAVOLTA='0'B;
255   2  1       END;
256   2  0     ELSE SCHEDA1=SCHEDA2;
257   2  0  ON ENDFILE (SYSIN)  BEGIN;
258   3  0                      FINE_SCHEDE='1';
259   3  0                      GO TO PROSEGUI;
260   3  0                      END;
261   2  0  IF FINE_SCHEDE THEN GO TO DUMPA;
262   2  0  GET EDIT(SCHEDA2)(COL(1),A(80));
263   2  0  PROSEGUI:
             OPERAZIONE=SCHEDA1;
264   2  0  N_OPERAZIONE=INDEX('XEOAIDP*',OPERAZIONE);
265   2  0  STR80=SUBSTR(SCHEDA1,3,70);
266   2  0  IF SUBSTR(SCHEDA1,1,1)='E'
               THEN DO;
267   2  1       STR40,STR80=SUBSTR(SCHEDA1,3,40);
268   2  1       RETURN;
269   2  1       END;
270   2  0  SOPRA: K=INDEX(STR80,' ');
271   2  0  IF K>0
               THEN STR80=SUBSTR(STR80,1,K-1);
272   2  0  IF SUBSTR(SCHEDA2,1,1) ='C'
               THEN RETURN;
273   2  0  STR80=STR80||SUBSTR(SCHEDA2,3,70);
274   2  0  SCHEDA1=SCHEDA2;
275   2  0  ON ENDFILE (SYSIN)  BEGIN;
276   3  0                      FINE_SCHEDE='1'B;
277   3  0                      SCHEDA2='';
278   3  0                      GO TO SOPRA;
279   3  0                      END;
280   2  0  GET EDIT(SCHEDA2)(COL(1),A(80));
281   2  0  GO TO SOPRA;
282   2  0  END LEGGI;
```

```
283   1  0  CONVERSA: PROC;
            /*************************************************************************
            *  QUESTA ROUTINE FORNISCE GLI STESSI RISULTATI FINALI DELLA ROUTINE *
            *  L'UNICA DIFFERENZA E' CHE OTTIENE I DATI DA CONSOLE ANZICHE' DA   *
            *  SCHEDE. NON SONO PREVISTE LE OPERAZIONI E CHE RISULTANO           *
            *  SCOMODE SE NON IMPOSSIBILI DA CONSOLE, MA VOLENDO SI POSSONO FARE *
            *************************************************************************/
284   2  0  RICOMINCIA:
            DISPLAY ('OPERAZIONE? RISPOSTA (P|O|I|A|D|X|?|END)') REPLY (STRVAR);
285   2  0  IF PRIMA_RISPOSTA THEN DO;
286   2  1                            IF STRVAR ='P=PIPI'
                                           THEN DO;
287   2  2                                      DISPLAY ('BATTERE PASSWORD');
288   2  2                                      GO TO RICOMINCIA;
289   2  2                                      END;
290   2  1                            PRIMA_RISPOSTA='0'B;
            /*************************************************************************/
291   2  1                            CALL FAIENQ;
            /*************************************************************************/
292   2  1                            READ FILE(FILONE) INTO(STRINGACCIA) KEY('1040');
293   2  1                            PASS=ORX_40CHAR(PAROLACCE,STRINGACCIA);
            /*************************************************************************/
294   2  1                            CALL FAIDEQ;
            /*************************************************************************/
295   2  1                            END;
296   2  0  IF STRVAR='END' THEN GO TO DUMPA;
297   2  0  OPERAZIONE=STRVAR;
298   2  0  IF VERIFY(OPERAZIONE,'XDIOA?')>0
              THEN GO TO RICOMINCIA;
299   2  0  N_OPERAZIONE=INDEX('XEOAIDP*?',OPERAZIONE);
300   2  0  STR80='';
301   2  0  IF OPERAZIONE='X'
              THEN DO;
302   2  1       RIDAI: DISPLAY('DAMMI LA STRINGA') REPLY (STRVAR);
303   2  1       IF VERIFY(STRVAR,'0123456789ABCDEF')>0
                    THEN GO TO RIDAI;
304   2  1       IF LENGTH (STRVAR) =40 & LENGTH (STRVAR) =48
                    THEN DO;
305   2  2                 DISPLAY ('STRINGA TROPPO CORTA O TROPPO LUNGA');
306   2  2                 GO TO RIDAI;
307   2  2                 END;
308   2  1       STR80=STRVAR;
309   2  1       RETURN;
310   2  1       END;
```

```
311   2  0  RIPETI1:
            DISPLAY('CODICE? RISPOSTA.: C(....) OPPURE X(........)')
                    REPLY (STRVAR);
312   2  0  IF LENGTH(STRVAR)  =7 & SUBSTR(STRVAR,1,1)='C' THEN GO TO RIPETI1;
313   2  0  IF LENGTH(STRVAR)  =11 & SUBSTR(STRVAR,1,1)='X' THEN GO TO RIPETI1;
314   2  0  IF (SUBSTR(STRVAR,1,2) ='C(' & SUBSTR(STRVAR,1,2) ='X(') |
                    SUBSTR(STRVAR,LENGTH(STRVAR),1) =')' THEN GO TO RIPETI1;
315   2  0  SK.CODICE=TRADUCI(STRVAR);
316   2  0  IF VERIFY(SK.CODICE,'0123456789ABCDEF')>0 THEN GO TO RIPETI1;
317   2  0  STR80=STR80||'CODICE=C('||EBCDIC(SK.CODICE)||')';
318   2  0  PRIMA_LETTERA_CODICE=SUBSTR(STR80,10,1);
319   2  0  IF OPERAZIONE ='D' | OPERAZIONE='?' THEN RETURN;
320   2  0  IF OPERAZIONE='O' | OPERAZIONE='A'
               THEN DO;
321   2  1  SE_PASSWORD:
                    DISPLAY ('VUOI CAMBIARE LA PASSWORD? RISPONDI "SI" O "NO"')
                    REPLY (STRVAR);
322   2  1           IF STRVAR='NO' THEN GO TO SE_LIMITI;
323   2  1           IF STRVAR='SI' THEN GO TO RICHIEDI_PASSWORD;
324   2  1              ELSE GO TO SE_PASSWORD;
325   2  1           END;
326   2  0  RICHIEDI_PASSWORD:
            DISPLAY('PASSWORD (RISP.: X(..............) OPPURE C(........)')
                    REPLY(STRVAR);
327   2  0  IF LENGTH(STRVAR) < 11 | LENGTH(STRVAR) > 19 THEN
                  GO TO RICHIEDI_PASSWORD;
328   2  0  IF (SUBSTR(STRVAR,1,2) ='C(' & SUBSTR(STRVAR,1,2) ='X(') |
                    SUBSTR(STRVAR,LENGTH(STRVAR),1) =')' THEN
                  GO TO RICHIEDI_PASSWORD;
329   2  0  SK.PASSW=TRADUCI(STRVAR);
330   2  0  IF VERIFY(SK.PASSW,'0123456789ABCDEF')>0 THEN
                  GO TO RICHIEDI_PASSWORD;
331   2  0  STR80=STR80||',PASSWORD=C('||EBCDIC(SK.PASSW)||')';
332   2  0  SE_LIMITI:
            IF OPERAZIONE='O' | OPERAZIONE='A'
               THEN DO;
333   2  1           DISPLAY ('VUOI CAMBIARE I LIMITI? RISPONDI "SI" O "NO"')
                    REPLY (STRVAR);
334   2  1           IF STRVAR='NO' THEN GO TO ESAME_RESIDUAL;
335   2  1           IF STRVAR='SI' THEN GO TO LIMITI_PAGANTI;
336   2  1              ELSE GO TO SE_LIMITI;
337   2  1           END;
```

```
338   2  0  LIMITI_PAGANTI:
               IF VERIFY (PRIMA_LETTERA_CODICE,'ABCDGJLMNOPQRUVWXYZ') =0
                  THEN DO;
339   2  1                LIMITACCI=EBCDIC('A000FD7F');
340   2  1                DISPLAY ('VUOI LIMITI STANDARD PER CODICI PAGANTI?' ||
                          ' RISP.: "SI" O "NO"')
                          REPLY (STRVAR);
341   2  1                IF STRVAR   ='SI' & STRVAR  ='NO'
                             THEN GO TO LIMITI_PAGANTI;
342   2  1                IF STRVAR='SI' THEN GO TO METTI_LIMITI;
343   2  1                               ELSE GO TO RICHIEDI_LIMITI;
344   2  1             END;
345   2  0  LIMITI_F:
               IF VERIFY (PRIMA_LETTERA_CODICE,'F') =0
                  THEN DO;
346   2  1                LIMITACCI=EBCDIC('B000FD7F');
347   2  1                DISPLAY ('VUOI LIMITI STANDARD PER CODICI IN FRANCHIGIA?'||
                          ' RISP.: "SI" O "NO"')
                          REPLY (STRVAR);
348   2  1                IF STRVAR   ='SI' & STRVAR  ='NO'
                             THEN GO TO LIMITI_F;
349   2  1                IF STRVAR='SI' THEN GO TO METTI_LIMITI;
350   2  1                               ELSE GO TO RICHIEDI_LIMITI;
351   2  1             END;
352   2  0  LIMITI_S:
               IF VERIFY (PRIMA_LETTERA_CODICE,'S') =0
                  THEN DO;
353   2  1                LIMITACCI=EBCDIC('F0044D29');
354   2  1                DISPLAY ('VUOI LIMITI STANDARD PER CODICI S?' ||
                          ' RISP.: "SI" O "NO"')
                          REPLY (STRVAR);
355   2  1                IF STRVAR   ='SI' & STRVAR  ='NO'
                             THEN GO TO LIMITI_S;
356   2  1                IF STRVAR='SI' THEN GO TO METTI_LIMITI;
357   2  1                               ELSE GO TO RICHIEDI_LIMITI;
358   2  1             END;
359   2  0  LIMITI_T:
               IF VERIFY (PRIMA_LETTERA_CODICE,'T') =0
                  THEN DO;
360   2  1                LIMITACCI=EBCDIC('F0004D29');
361   2  1                DISPLAY ('VUOI LIMITI STANDARD PER CODICI T?' ||
                          ' RISP.: "SI" O "NO"')
                          REPLY (STRVAR);
362   2  1                IF STRVAR   ='SI' & STRVAR  ='NO'
                             THEN GO TO LIMITI_T;
363   2  1                IF STRVAR='SI' THEN GO TO METTI_LIMITI;
364   2  1                               ELSE GO TO RICHIEDI_LIMITI;
365   2  1             END;
```

```
366   2  0  LIMITI_E:
               IF VERIFY (PRIMA_LETTERA_CODICE,'E') =0
                  THEN DO;
367   2  1               LIMITACCI=EBCDIC('F0000100');
368   2  1               DISPLAY ('VUOI LIMITI STANDARD PER CODICI E?' ||
                        ' RISP.: "SI" O "NO"')
                        REPLY (STRVAR);
369   2  1               IF STRVAR   ='SI' & STRVAR  ='NO'
                           THEN GO TO LIMITI_E;
370   2  1               IF STRVAR='SI' THEN GO TO METTI_LIMITI;
371   2  1                              ELSE GO TO RICHIEDI_LIMITI;
372   2  1            END;
373   2  0  LIMITI_K:
               IF VERIFY (PRIMA_LETTERA_CODICE,'K') =0
                  THEN DO;
374   2  1               LIMITACCI=EBCDIC('A000FD7F');
375   2  1               DISPLAY ('VUOI LIMITI STANDARD PER CODICI K?' ||
                        ' RISP.: "SI" O "NO"')
                        REPLY (STRVAR);
376   2  1               IF STRVAR   ='SI' & STRVAR  ='NO'
                           THEN GO TO LIMITI_K;
377   2  1               IF STRVAR='SI' THEN GO TO METTI_LIMITI;
378   2  1                              ELSE GO TO RICHIEDI_LIMITI;
379   2  1            END;
            /*************RIEMPIMENTO LIMITI****************
            ************** IN MODO MANUALE  ****************/
380   2  0  RICHIEDI_LIMITI:
            DISPLAY('LIMITI (RISP.: X(........) OPPURE NO')
                     REPLY(STRVAR);
381   2  0  IF STRVAR = 'NO' THEN GO TO DETTAGLIA_LIMITI;
382   2  0  IF LENGTH(STRVAR) = 11 |
               SUBSTR(STRVAR,1,2) ='X('  |
               SUBSTR(STRVAR,LENGTH(STRVAR),1) =')' THEN
               GO TO RICHIEDI_LIMITI;
383   2  0  SK_LIMITI=TRADUCI(STRVAR);
384   2  0  IF VERIFY(SK_LIMITI,'0123456789ABCDEF')>0 THEN
                  GO TO RICHIEDI_LIMITI;
385   2  0  STR80=STR80||',LIMITI=X('||SK_LIMITI||')';
386   2  0  GO TO ESAME_RESIDUAL;
387   2  0  DETTAGLIA_LIMITI:
            LIMITIBIT(1)='1'B;
388   2  0  LIMITIBIT(5)='0'B;
389   2  0  LIMITIBIT(6)='0'B;
390   2  0  LIMITIBIT(7)='0'B;
391   2  0  LIMITIBIT(8)='0'B;
392   2  0  LIMITIBIT(9)='0'B;
393   2  0  LIMITIBIT(10)='0'B;
394   2  0  LIMITIBIT(11)='0'B;
395   2  0  LIMITIBIT(12)='0'B;
396   2  0  LIMITIBIT(13)='0'B;
```

```
397   2  0  CHIEDI_NOME_JOB:
             DISPLAY ('IL NOME JOB E'' CONCATENATO? RISP.: SI-NO-STOP')
             REPLY (STRVAR);
398   2  0    IF STRVAR='STOP' THEN  GO TO RICOMINCIA;
399   2  0    IF STRVAR='SI'
                THEN LIMITIBIT(2)='1'B;
400   2  0       ELSE IF STRVAR='NO'
                        THEN LIMITIBIT(2)='0'B;
401   2  0                ELSE GO TO CHIEDI_NOME_JOB;
402   2  0  CHIEDI_ESAME_PASSWORD:
             DISPLAY('E'' RICHIESTO L''ESAME DELLA PASSWORD? RISP.:SI-NO-STOP')
             REPLY (STRVAR);
403   2  0    IF STRVAR='STOP' THEN  GO TO RICOMINCIA;
404   2  0    IF STRVAR='SI'
                THEN LIMITIBIT(3)='1'B;
405   2  0       ELSE IF STRVAR='NO'
                        THEN LIMITIBIT(3)='0'B;
406   2  0                ELSE GO TO CHIEDI_ESAME_PASSWORD;
407   2  0  CHIEDI_ESAME_RESIDUAL:
             DISPLAY('E'' RICHIESTO L''ESAME DEL RESIDUAL? RISP.:SI-NO-STOP')
             REPLY (STRVAR);
408   2  0    IF STRVAR='STOP' THEN  GO TO RICOMINCIA;
409   2  0    IF STRVAR='SI'
                THEN LIMITIBIT(4)='1'B;
410   2  0       ELSE IF STRVAR='NO'
                        THEN LIMITIBIT(4)='0'B;
411   2  0                ELSE GO TO CHIEDI_ESAME_RESIDUAL;
412   2  0  RICHIEDI_REMISI:
             DISPLAY('IL CODICE PUO'' FARE ''=REMISI''? RISP.: SI-NO-STOP')
                    REPLY(STRVAR);
413   2  0  IF STRVAR='STOP' THEN  GO TO RICOMINCIA;
414   2  0  IF STRVAR    ='SI' & STRVAR  ='NO' THEN GO TO RICHIEDI_REMISI;
415   2  0  IF STRVAR='SI'
                    THEN LIMITIBIT(14)='1'B;
416   2  0               ELSE LIMITIBIT(14)='0'B;
417   2  0  RICHIEDI_ACNO:
             DISPLAY('IL CODICE PUO'' USARE ''AC=NO''? RISP.: SI-NO-STOP')
                    REPLY(STRVAR);
418   2  0  IF STRVAR='STOP' THEN  GO TO RICOMINCIA;
419   2  0  IF STRVAR    ='SI' & STRVAR  ='NO' THEN GO TO RICHIEDI_ACNO;
420   2  0  IF STRVAR='SI'
                    THEN LIMITIBIT(15)='1'B;
421   2  0               ELSE LIMITIBIT(15)='0'B;
```

```
422   2  0  RICHIEDI_ADDRSPC:
            DISPLAY('IL CODICE PUO'' FARE ''ADDRSPC=REAL''? RISP.: SI-NO-STOP')
                  REPLY(STRVAR);
423   2  0  IF STRVAR='STOP' THEN  GO TO RICOMINCIA;
424   2  0  IF STRVAR     ='SI' & STRVAR  ='NO' THEN GO TO RICHIEDI_ADDRSPC;
425   2  0  IF STRVAR='SI'
                  THEN LIMITIBIT(16)='1'B;
426   2  0          ELSE LIMITIBIT(16)='0'B;
427   2  0  RICHIEDI_COPIE:
            DISPLAY('1 COPIA, 3 COPIE, O NO LIMIT? RISP.: 1-2-3-STOP')
                  REPLY(STRVAR);
428   2  0  IF STRVAR='STOP' THEN  GO TO RICOMINCIA;
429   2  0  IF LENGTH(STRVAR) =1 | VERIFY(STRVAR,'123')>0
               THEN GO TO RICHIEDI_COPIE;
430   2  0  LIMITIBIT(17),LIMITIBIT(18)='0'B;
431   2  0  IF STRVAR='2' THEN LIMITIBIT(18)='1'B;
432   2  0  IF STRVAR='3' THEN LIMITIBIT(18),LIMITIBIT(17)='1'B;
433   2  0  RICHIEDI_CPU:
            DISPLAY('1'',3'',10'',30'' DI CPU O NO LIMIT? RISP.: 1-2-3-4-5-STOP')
                  REPLY(STRVAR);
434   2  0  IF STRVAR='STOP' THEN  GO TO RICOMINCIA;
435   2  0  IF LENGTH(STRVAR) =1 | VERIFY(STRVAR,'12345')>0
               THEN GO TO RICHIEDI_CPU;
436   2  0  LIMITIBIT(19),LIMITIBIT(20),LIMITIBIT(21),LIMITIBIT(22)='0'B;
437   2  0  IF STRVAR>='2' THEN LIMITIBIT(22)='1'B;
438   2  0  IF STRVAR>='3' THEN LIMITIBIT(21)='1'B;
439   2  0  IF STRVAR>='4' THEN LIMITIBIT(20)='1'B;
440   2  0  IF STRVAR>='5' THEN LIMITIBIT(19)='1'B;
```

```
441   2  0  RICHIEDI_CLASSE:
            DISPLAY('NESSUNA CLASSE, TUTTE - W O NO LIMIT? RISP.: 1-2-3-STOP')
                  REPLY(STRVAR);
442   2  0  IF STRVAR='STOP' THEN  GO TO RICOMINCIA;
443   2  0  IF LENGTH(STRVAR) =1 | VERIFY(STRVAR,'123')>0
               THEN GO TO RICHIEDI_CLASSE;
444   2  0  LIMITIBIT(23),LIMITIBIT(24)='0'B;
445   2  0  IF STRVAR>='2' THEN LIMITIBIT(24)='1'B;
446   2  0  IF STRVAR>='3' THEN LIMITIBIT(23)='1'B;
447   2  0  RICHIEDI_REGION:
            DISPLAY('256K, 512K, 2048K DI REGION O NO LIMIT? RISP.: 1-2-3-4-STOP')
                  REPLY(STRVAR);
448   2  0  IF STRVAR='STOP' THEN  GO TO RICOMINCIA;
449   2  0  IF LENGTH(STRVAR) =1 | VERIFY(STRVAR,'1234')>0
               THEN GO TO RICHIEDI_REGION;
450   2  0  LIMITIBIT(25),LIMITIBIT(26),LIMITIBIT(27)='0'B;
451   2  0  IF STRVAR>='2' THEN LIMITIBIT(27)='1'B;
452   2  0  IF STRVAR>='3' THEN LIMITIBIT(26)='1'B;
453   2  0  IF STRVAR>='4' THEN LIMITIBIT(25)='1'B;
454   2  0  RICHIEDI_PUNCH:
            DISPLAY('0 SK, 200 SK DI PUNCH O NO LIMIT? RISP.: 1-2-3-STOP')
                  REPLY(STRVAR);
455   2  0  IF STRVAR='STOP' THEN  GO TO RICOMINCIA;
456   2  0  IF LENGTH(STRVAR) =1 | VERIFY(STRVAR,'123')>0
               THEN GO TO RICHIEDI_PUNCH;
457   2  0  LIMITIBIT(28),LIMITIBIT(29)='0'B;
458   2  0  IF STRVAR>='2' THEN LIMITIBIT(29)='1'B;
459   2  0  IF STRVAR>='3' THEN LIMITIBIT(28)='1'B;
460   2  0  RICHIEDI_LINEE:
            DISPLAY('2000, 5000, 10000 LINEE O NO LIMIT? RISP.: 1-2-3-4-STOP')
                  REPLY(STRVAR);
461   2  0  IF STRVAR='STOP' THEN  GO TO RICOMINCIA;
462   2  0  IF LENGTH(STRVAR) =1 | VERIFY(STRVAR,'1234')>0
               THEN GO TO RICHIEDI_LINEE;
463   2  0  LIMITIBIT(30),LIMITIBIT(31),LIMITIBIT(32)='0'B;
464   2  0  IF STRVAR>='2' THEN LIMITIBIT(32)='1'B;
465   2  0  IF STRVAR>='3' THEN LIMITIBIT(31)='1'B;
466   2  0  IF STRVAR>='4' THEN LIMITIBIT(30)='1'B;
            /**********FINE RIEMPIMENTO LIMITI************/
```

```
467   2  0  METTI_LIMITI:
            SK_LIMITI=ESADECI(LIMITACCI);
468   2  0  STR80=STR80||',LIMITI=X('||SK_LIMITI||')';
469   2  0  ESAME_RESIDUAL:
            DISPLAY('VUOI INSERIRE RESIDUAL? (RISPONDI IL NUMERO DI SEC O "NO")')
               REPLY(STRVAR);
470   2  0  IF STRVAR='NO' /*****& (OPERAZIONE='O' | OPERAZIONE='A')******/
               THEN GO TO ESAME_VOLUMI;
471   2  0  IF VERIFY(STRVAR,'0123456789')>0
               THEN GO TO ESAME_RESIDUAL;
472   2  0  STR80=STR80||',RESIDUAL='||STRVAR;
473   2  0  ESAME_VOLUMI:
            DISPLAY('VOLUMI? (RISPOSTA X(.........) O "NO"')REPLY(STRVAR);
474   2  0  IF STRVAR='NO' /****& (OPERAZIONE='O' | OPERAZIONE='A')*******/
               THEN GO TO RITORNO;
475   2  0  IF LENGTH(STRVAR) =11 THEN GO TO ESAME_VOLUMI;
476   2  0  IF  SUBSTR(STRVAR,1,2) ='X(' |  SUBSTR(STRVAR,LENGTH(STRVAR),1) =')'
               THEN GO TO ESAME_VOLUMI;
477   2  0  STRVAR=SUBSTR(STRVAR,3,8);
478   2  0  IF VERIFY(STRVAR,'0123456789ABCDEF')>0
               THEN GO TO ESAME_VOLUMI;
479   2  0  STR80=STR80||',VOLUMI=X('||STRVAR||')';
            /************FINE INSERIMENTO PARAMETRI**************/
480   2  0  RITORNO:
            STRVAR=STR80;
481   2  0  DISPLAY(STRVAR);
482   2  0  IF LENGTH(STR80)>72
               THEN DO;
483   2  1     STRVAR=SUBSTR(STR80,73);
484   2  1     DISPLAY(STRVAR);
485   2  1     END;
486   2  0  DISPLAY('OPERAZIONE='||OPERAZIONE||'; OK? (RISP. OK O KO)')
               REPLY(STRVAR);
487   2  0  IF STRVAR ='KO' & STRVAR ='OK'
               THEN GO TO RITORNO;
488   2  0  IF STRVAR='KO'
               THEN GO TO RICOMINCIA;
489   2  0  RETURN;
```

```
490   2  0  DECLARE
                 PERMOG FILE OUTPUT ENV(CONSECUTIVE F BLKSIZE(800) RECSIZE(80)),
                 LIMITIBIT(32)       BIT(1) DEF LIMITACCI,
                 LIMITACCI CHAR(4),
              1 SK,
                 2 CODICE CHAR(8),
                 2 LIMITI CHAR (6),
                 2 OTTOZERI     CHAR(8) INIT ('00000000'),
                 2 PASSW    CHAR(16),
                 2 FILLER1 CHAR(2) INIT (''),
                 2 CODCHAR CHAR(4),
                 2 ULTIM    CHAR(34) INIT (' -                              ???'),
                 STRVAR     CHAR(72)INIT('') VAR ,
                 PRIMA_RISPOSTA BIT(1) INIT ('1') STATIC,
                 FINE_SCHEDE BIT(1) INIT ('0'),
                 PRIMA_LETTERA_CODICE CHAR(1),
                 ESADECI ENTRY (CHAR(100)VAR) RETURNS (CHAR(200) VAR);
491   2  0  TRADUCI: PROC (CHARVAR) RETURNS(CHAR(50) VAR);
492   3  0  IF SUBSTR(CHARVAR,1,2)='X('
              THEN RETURN (SUBSTR(CHARVAR,3,LENGTH(CHARVAR)-3));
493   3  0  IF SUBSTR(CHARVAR,1,2)='C('
              THEN RETURN (ESADECI(SUBSTR(CHARVAR,3,LENGTH(CHARVAR)-3)));
494   3  0  DECLARE CHARVAR CHAR(50) VAR;
495   3  0  END TRADUCI;
496   2  0  END CONVERSA;
497   1  0  FINE:  END CARICA;
```

```
STMT LEV NT

             /*** PROGRAMMA DI DUMP DEI CODICI OS-VS2 ***/


  1       0   DUMP: PROC (PARM) OPTIONS(MAIN);
  2   1   0     ON ENDPAGE (SYSPRINT) BEGIN;
  3   2   0                           CONT=CONT+1;
  4   2   0                           PUT PAGE;
  5   2   0     PUT EDIT ('CODICE   LIMITI          RESIDUAL     PASSWORD') (A);
  6   2   0     PUT EDIT ('PASSWORD') (COL(59),A);
  7   2   0     PUT EDIT ('VOLUMI    FILLER',CONT) (COL(69),A,COL(109),A);
  8   2   0                           PUT SKIP(3);
  9   2   0                           END;
 10   1   0   ON UNDEFINEDFILE(SYSIN)
               BEGIN;
 11   2   0     DISPLAY('DAMMI I 40 CARATTERI DI PASSWORD') REPLY (VETT);
 12   2   0     GO TO CONTROLLA;
 13   2   0     END;
 14   1   0   ON ENDFILE(SYSIN) SIGNAL UNDEFINEDFILE(SYSIN);
 15   1   0   GET EDIT(SCHEDA1)(A(80));
 16   1   0   IF SUBSTR(SCHEDA1,4,1) ='('
               THEN GO TO ERRORE_DI_PARENTESI;
 17   1   0   IF SUBSTR(SCHEDA1,3,1) ='C' & SUBSTR(SCHEDA1,44,1) =')'
               THEN GO TO ERRORE_DI_PARENTESI;
 18   1   0   CONTROLLA: READ FILE (FILONE) INTO (STRINGACCIA) KEY (1040);
 19   1   0   PASS=0&X_40CHAR(PAROLACCE,STRINGACCIA);
 20   1   0   VETT=SUBSTR(SCHEDA1,5,40);
 21   1   0   IF PASS =VETT
               THEN DO;
 22   1   1        PUT EDIT('PASSWORD NON CORRETTA')(A)SKIP;
 23   1   1        CALL PLIRETC(16);
 24   1   1        RETURN;
 25   1   1        END;
 26   1   0     IF INDEX(PARM,'STA') =0
                  THEN SIGNAL ENDPAGE (SYSPRINT);
 27   1   0     READ FILE (FILONE) INTO (PASS    ) KEY (1040);
 28   1   0     Q=ADDR(PASS);
 29   1   0     R=ADDR(PAROLACCE);
 30   1   0     PASSDEF=(PASSDEF|PAROLACCEDEF)& (PASSDEF&PAROLACCEDEF);
 31   1   0          DO CHIAVE=0 TO 1039;
 32   1   1          READ FILE (FILONE) INTO (REC) KEY (CHIAVE);
 33   1   1            DO J=0 TO 24;
 34   1   2              IF BLOCCO(J).LIMITI(1) THEN GO TO CICLO1;
 35   1   2              P=ADDR(BLOCCO(J));
 36   1   2              Z=ADDR(VETT);
 37   1   2     CONDDO=(RECDEFBIT|PASSDEF)& (RECDEFBIT&PASSDEF);
```

```
38   1  2     IF INDEX(PARM,'STA') =0
              THEN DO;
39   1  3          KKK=UNSPEC(SUBSTR(VETT,9,4);
40   1  3          KKK=KKK/100;
41   1  3          PUT SKIP EDIT (SUBSTR(VETT,1,4),ESADECI(SUBSTR(VETT,5,4)),
                   KKK,ESADECI(SUBSTR(VETT,13,8)),
                   SUBSTR(VETT,13,8),
                   ESADECI(SUBSTR(VETT,21,4)),SUBSTR(VETT,25,16))
                  (A,COL(10),A,COL(20),A,COL(39),A,COL(59),A,COL(69),A,COL(79),A);
42   1  3          END;
43   1  2     IF INDEX(PARM,'PERF') =0
              THEN DO;
44   1  3          NUMERO=NUMERO+10;
45   1  3          IVETT='E ' || VETT || '                                    ' ||
                   'DUM'||NUMERO;
46   1  3          WRITE FILE (SCHEDE) FROM (IVETT);
47   1  3          END;
48   1  2     CICLO1: END;
49   1  1         END;
50   1  0   GO TO FINE;
51   1  0   ERRORE_DI_PARENTESI:
           PUT SKIP(5) EDIT('IMPARA A PERFORARE CORRETTAMENTE LE PARENTESI!!!',
                    'P',SUBSTR(SCHEDA1,4,1),STR40,SUBSTR(SCHEDA1,40,1))
              (A,SKIP,A(1),X(1),A(1),A(40),A(1));


52   1  0    DECLARE
              1  REC UNALIGNED,
                2 BLOCCO(0:24),
                  3 CODICE,
                    4 LETTERA CHAR(1),
                    4 NUMERO  PICTURE '999',
                  3 LIMITI(32) BIT(1),
                  3 RESIDUAL   BIN FIXED (31),
                  3 PASSWORD   CHAR(8),
                  3 VOLUMI(32) BIT(1),
                  3 FILLER     CHAR(16),
           FILONE FILE RECORD INPUT KEYED DIRECT
               ENV (REGIONAL(1) F RECSIZE (1000) BLKSIZE (1000)),
           RECDEFBIT    BIT(320) BASED (P),
           COMODO       BIT(320) BASED (Z),
           PAROLACCEDEF BIT(320) BASED (R),
           PASSDEF      BIT(320) BASED (Q),
           PARM  CHAR(100) VARYING,
           SCHEDA1 CHAR(80),
           VETT  CHAR(40),
           PASS  CHAR(1000),
           STRINGACCIA CHAR(1000),
           IVETT CHAR(80),
```

```
          ESADECI ENTRY (CHAR(10) VAR) RETURNS (CHAR(20) VAR),
          KKK BIN FIXED(31),
          NUMERO PICTURE '99999' INIT(0),
          CONT   PICTURE '999' INIT(0),
          CHIAVE PICTURE '(8)9',
          PAROLACCE CHAR(40)
               INIT('  PAROLA CHIAVE  ');

53   1  0  ORX_40CHAR: PROC (A,B) RETURNS(CHAR(40));
54   2  0  PA=ADDR(A);
55   2  0  PB=ADDR(B);
56   2  0  PC=ADDR(C);
57   2  0  CBIT=(ABIT|BBIT)& (ABIT&BBIT);
58   2  0  RETURN (C);


59   2  0  DECLARE
                    (PA,PB,PC)      POINTER,
                    (A,B,C)         CHAR(40),
                    ABIT       BIT(320)  BASED(PA),
                    BBIT       BIT(320)  BASED(PB),
                    CBIT       BIT(320)  BASED(PC);
60   2  0  END OFX_40CHAR;
61   1  0  FINE: END DUMPCOD;
```

APPENDICE A

Descrizione del file dei codici

```
RECFM    F
LRECL    1000
BLKSIZE  1000
DSORG DA   ---> REGIONAL(1)
Numero records 171 5 CYL
SPACE = (CYL,(5))
```

Tracciato record

Il record e' composto di 25 blocchi (lunghi 40) che descrivono ciascuno un codice. Il tracciato di tali blocchi e' il seguente:

```
2 CODICE
            3 LETTERA     CHAR (1)
            3 NUMERO      PICTURE '999'
2 LIMITI (32)             BIT (1)
2 RESIDUAL                BIN FIXED (31)
2 PASSWORD                CHAR (8)
2 VOLUMI (32)             BIT (1)
2 SPY_LOG
            3 DATA        FIXED (7)
            3 ORA         BIN FIXED (31)
            3 JOBNAME     CHAR (8)
```

FORMULE PER DETERMINARE RECORD E BLOCCO

Dato un codice e' possibile determinare il record ed il blocco del record in cui si trova la descrizione di detto codice.

```
NUMERO_RECORD=((INDEX(ALFABETO,LETTERA)-1*40)+TRUNC(NUMERO/25);
```

dove:

INDEX e TRUNC hanno il significato delle funzioni PL1

ALFABETO e' la stringa di 26 caratteri 'ABCDEFGHIJKLMNOPQ..YZ'

LETTERA e' il primo carattere del codice

NUMERO e' il numero formato dal 2°, 3° e 4° carattere del codice.

APPENDICE B

Significato del campo dei LIMITI del blocco del dei codici

```
Byte 1 : |_____|_____|_____|_____|_____|_____|_____|_____|
   bit       1     2     3     4     5     6     7     8
```

Descrizione :

- Bit 1 : (1) = codice abilitato
  (0) = codice non abilitato
- Bit 2 : (1) = Il nome JOB deve essere formato da 8 caratteri, di cui i primi 4 sono il codice e gli ultimi quattro il codice-presentatore
  (0) = nessuna limitazione sul nome JOB
- Bit 3 : (1) = E' richiesto l'esame della password
  (0) = non E' richiesto l'esame della password
- Bit 4 : (1) = E' richiesto l'esame del residual
  (0) = non E' richiesto l'esame del residual
- Bit 5 : riservato
- Bit 6 : riservato
- Bit 7 : riservato
- Bit 8 : riservato

```
Byte 2 : |_____|_____|_____|_____|_____|_____|_____|_____|
   bit       1     2     3     4     5     6     7     8
```

Descrizione :

- Bit 1 : riservato
- Bit 2 : riservato
- Bit 3 : riservato
- Bit 4 : riservato
- Bit 5 : riservato
- Bit 6 : (1) = Il codice e' abilitato a usare '=REMISI'
  (0) = Il codice non e' abilitato a usare '=REMISI'
- bit 7 : (1) = Il codice e' abilitato a usare 'AC=NO'
  (0) = Il codice non e' abilitato a usare 'AC=NO'
- Bit 8 : (1) = Il codice e' abilitato a usare 'ADDRSPC=REAL'
  (0) = Il codice non e' abilitato a usare 'ADDRSPC=REAL'

```
Byte 3 : |_____|_____|_____|_____|_____|_____|_____|_____|
   bit       1     2     3     4     5     6     7     8

   campi  : |__ copie __|_____ tempo di CPU ____|_ classe  _|
```

Descrizione :

Campo 'copie'	: (00)	= massimo 1 copia
	(01)	= massimo 3 copie
	(11)	= nessun limite
Campo 'tempo di cpu'	: (0000)	= massimo 1 minuto
	(0001)	= massimo 3 minuti
	(0011)	= massimo 10 minuti
	(0111)	= massimo 30 minuti
	(1111)	= nessun limite
Campo 'classe'	: (00)	= nessunA ASSEGNAZIONE PERMESSA
	(01)	= PERMESSA OGNI ASSEGNAZIONE TRANNE 'W'
	(11)	= permessa qualsiasi assegnazione

```
Byte 4 : |_____|_____|_____|_____|_____|_____|_____|_____|
   bit       1     2     3     4     5     6     7     8

   campi  : |____ regione ____|  perforaz. |_____ stampa ____|
```

Descrizione :

Campo 'regione'	: (000)	= massimo 256k
	(001)	= massimo 512k
	(011)	= massimo 2048k
	(111)	= nessun limite
Campo 'perforaz.'	: (00)	= non ammessa perforazione
	(01)	= massimo 200 schede
	(11)	= nessun limite
Campo 'stampa'	: (000)	= massimo 2000 righe
	(001)	= massimo 5000 righe
	(011)	= massimo 10000 righe
	(111)	= nessun limite

APPENDICE C

Comunicazione agli Utenti del Batch OS.

E' disponibile per gli Utenti del servizio batch OS sul 370/158 un sistema di protezione del codice.

Tale sistema di protezione e' realizzato associando ad ogni codice una password. Quest'ultima compare in una qualsiasi scheda controllo EXEC o DD del job, in un campo commento.

La password e' formata da 8 bytes ed e' accettata qualsiasi configurazione esadecimale. Essa deve comparire nella seguente forma:

PASSWORD='xxxxxxxx'

in un campo commento di una scheda EXEC o DD del job. Non deve comparire nella scheda JOB, in una scheda PROC, in una scheda PEND, in una scheda NULL ne' in alcuna delle schede controllo comprese tra una PROC e una PEND.

Se la scritta PASSWORD='xxxxxxxx' compare piu' volte tra le schede controllo di un job, solo l'ultima e' ritenuta valida. Nessuna di tali scritte compare sul listing di uscita.

Se un job non ha, in una delle schede controllo, la password richiesta, o ne ha una sbagliata, il job non viene eseguito e si ha una segnalazione di 'JCL error'. L'uscita, comprendente la mascherina e il job control (in funzione del parametro MSGLEVEL), viene comunque stampata, o inviata al remoto a seconda del campo tra apici della scheda JOB.

ESEMPIO

```
//ESEMPIO JOB (0001,P999, ....
//STEP1 EXEC PGM=ANY              PASSWORD='NOPQ1111'
//STEPLIB DD DSN=DATASET,DISP=SHR   PASSWORD='X1Y2Z3W7'
//.....
```

Il job viene eseguito se la password associata al codice B999 e' 'X1Y2Z3W7'.

APPENDICE D

Viene qui descritta in maggior dettaglio la sequenza delle risposte necessarie per apportare modifiche ad un codice.

Le risposte in cui e' specificato

X (.....) oppure C (.....)

stanno ad indicare che la risposta puo' esser data in caratteri se si usa C(.....) oppure nella traduzione esadecimale dei caratteri se si usa X(.....).

1) Operazione P seguita dalla password che autorizza l'uso del programma.

2) Richiesta di una operazione, (END per terminare).

3) Richiesta del codice su cui apportare le modifiche (escluso operazione X per la quale occorre fornire l'intera stringa di 40 bytes).

4) Richiesta della password del codice. Nel caso di operazioni O/A e' opzionale la conferma della password gia' esistente .

5) Richiesta dei limiti per quel codice. Possono essere scelti dei limiti standard (definiti in un apposito data set) per le seguenti categorie di codici:

PAGANTI : codici la cui prima lettera e' "A,B,C,D,G,J,L,M,N,O,P,Q,R,U,V,W,X,Y,Z"

COLLABORAZIONI : codici "F"

TESI ISI : codici "S"

TESI NON ISI : codici "T"

CORSI UNIVERSITA' : codici "E"

CODICI GRATUITI : codici "K"

Se alla richiesta di limiti standard viene risposto

"NO", viene posta la domanda :

LIMITI (RISP: X(.....) oppure NO).

Se la risposta e' X(.....) la stringa battuta costituira' i nuovi limiti del codice senza porre altre domande, mentre se la risposta e' "NO" verranno chiesti in dettaglio tutti i limiti uno per uno.

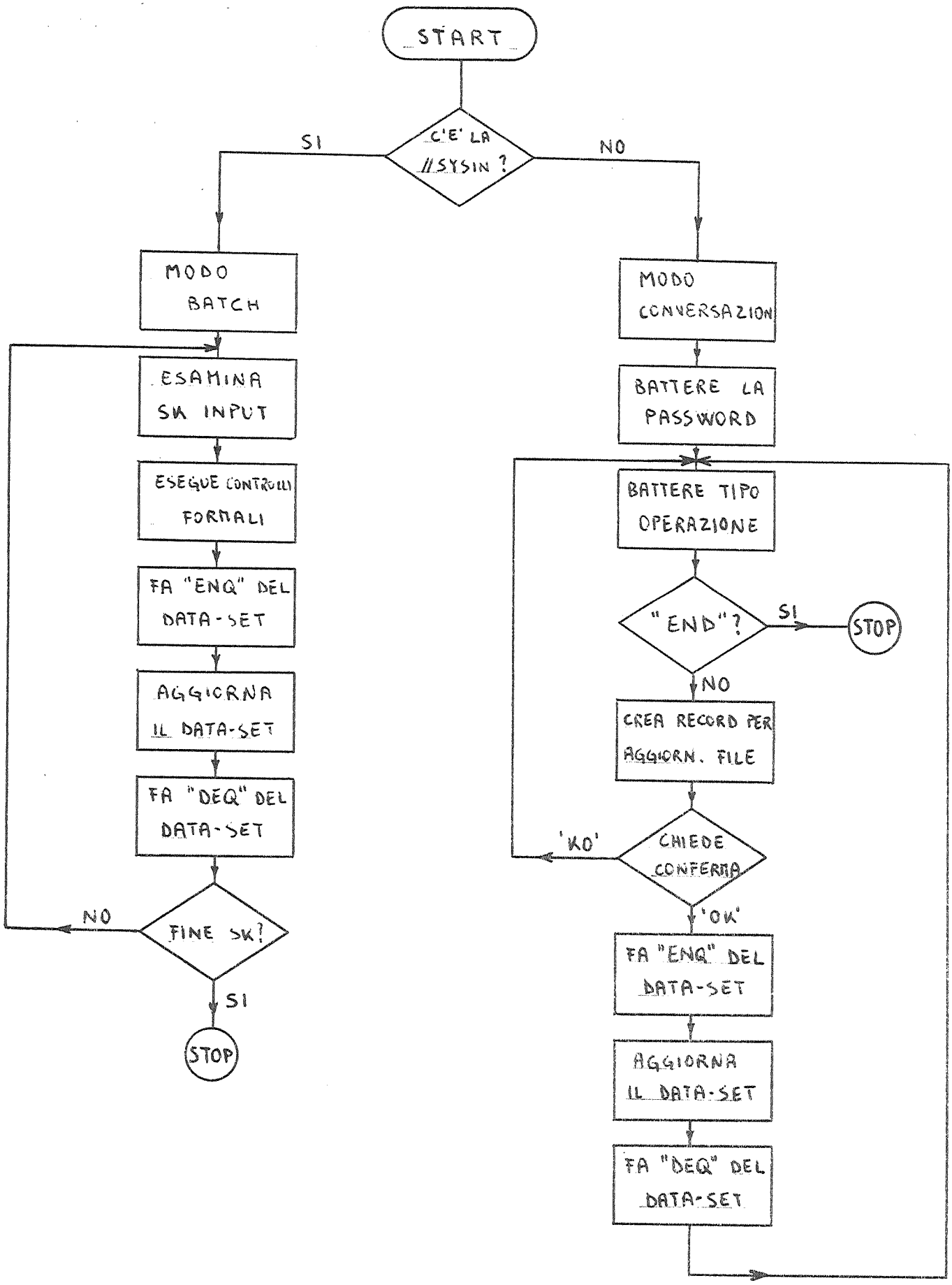
Nel caso venisse inviato un limite errato, e' possibile rispondere "STOP" ad una qualsiasi domanda successiva. Cio' consentira' di ripartire da capo specificando una nuova operazione ed il codice su cui si stava lavorando rimarra' inalterato su disco.

Infine nel caso di operazioni "O/A" si possono confermare i limiti gia' esistenti per quel codice (opzionale).

6) Viene richiesto il RESIDUAL da aggiungere (operazione "A") o da rimpiazzare a quello esistente. Nel caso si risponda "NO" il RESIDUAL per quel codice resta inalterato.

7) Volumi (per spiegazione dettagliata vedere nota interna CNUCE 106).

8) Richiesta di conferma. Se la richiesta e' OK il data set dei codici e' alterato, se la risposta e' "KO" si deve ripartire dal punto 2 con una nuova operazione.


START
C'E' LA //SYSIN ?
SI
NO
MODO BATCH
ESAMINA SK INPUT
ESEGUE CONTROLLI FORMALI
FA "ENQ" DEL DATA-SET
AGGIORNA IL DATA-SET
FA "DEQ" DEL DATA-SET
FINE SK?
NO
SI
STOP
MODO CONVERSAZION
BATTERE LA PASSWORD
BATTERE TIPO OPERAZIONE
"END"?
SI
STOP
NO
CREA RECORD PER AGGIORN. FILE
CHIEDE CONFERMA
'KO'
'OK'
FA "ENQ" DEL DATA-SET
AGGIORNA IL DATA-SET
FA "DEQ" DEL DATA-SET