

Istituto Professionale di Stato per l'Industria e l'Artigianato
MORETTO
Via Luigi Apollonio, 21 BRESCIA

Progetto finalizzato per

**Impiego del codice a barre nella raccolta dati
relativi al funzionamento del Centro Stampa di Istituto**

Realizzazione

HERRI FAUSTI
CRISTIAN TURELLI
DANIELE BOGLIONI

della classe 5AI a.s. 1995-96

corso per Tecnici delle Industrie Elettriche ed Elettroniche

INTRODUZIONE GENERALE

Questa esperienza è finalizzata alla soluzione di una problematica concreta, quella di automatizzare la registrazione del servizio fotocopie dell'istituto.

Gli utenti di questo servizio (studenti, docenti, uffici, classi.) hanno a disposizione un numero massimo di fotocopie stabilito all'inizio di ogni anno scolastico. Quando uno di questi si presenta al suddetto ufficio richiedendo delle fotocopie, l'addetto alla stampa procede all'operazione ed alla registrazione di quest'ultima su più registri per ragioni di carattere statistico. E' ovvio che tale sistema abbia notevoli limiti rispetto ad un sistema di acquisizione dati automatico per le elevate possibilità di errore e per la notevole lentezza di elaborazione dei dati.

Il sistema da noi proposto per risolvere questo problema è simile a quello utilizzato nelle farmacie e nei supermercati per il riconoscimento delle merci, la movimentazione del magazzino ecc.

Il nostro sistema si basa quindi sull'utilizzo di un barcode e di un programma di gestione dati.

Il primo ci permette di codificare un dato caratteristico di ogni singolo soggetto e quindi in sostanza di riconoscerlo; il secondo invece serve a gestire tutti i dati memorizzati sotto forma d'archivio. Per attuare tale progetto è stato quindi necessario assegnare in modo automatico un codice a barre ad ogni singolo utente (associazione fatta attraverso il numero di matricola); ed inoltre creare due tipi di archivi: uno di tipo anagrafico nel quale sono contenuti tutti i dati utili di ogni singolo utente e un altro di tipo storico nel quale vengono registrate una dopo l'altra le operazioni compiute al centro stampa da ogni singolo utente. Per contenere lo spazio occupato su disco vengono tenute in memoria solo le ultime cinque operazioni compiute dall'utente.

Quando quest'ultimo si presenterà all'ufficio, per richiedere delle fotocopie, l'operatore procederà alla lettura, tramite penna ottica, del codice (stampato su appositi tesserini e per gli studenti sul libretto personale) e quindi all'acquisizione dei dati contenuti in esso. Il programma, scritto in dBASE, attraverso il numero di matricola risale ai dati caratteristici del soggetto riportando su video insieme a questi anche le ultime operazioni compiute. A questo punto l'operatore può solo prendere visione dei dati oppure passare all'inserimento della quantità di fotocopie fatte. Nella pagina seguente sono riportati i tracciati record degli archivi da noi utilizzati:

TRACCIATI RECORD:

ARCHIVIO STUDENTI :

MATRICOLA	COGNOME	NOME	CLASSE	INSERIMENTO FOTOCOPIE FATTE
NUMERICO 5 cifre	CARATTERE 15 Caratteri	CARATTERE 15 Caratteri	CARATTERE 4 Caratteri	NUMERICO 5 Cifre

ARCHIVIO DOCENTI :

MATRICOLA	COGNOME	NOME	MATERIA	INSERIMENTO FOTOCOPIE FATTE
NUMERICO 5 cifre	CARATTERE 15 Caratteri	CARATTERE 15 Caratteri	CARATTERE 4 Caratteri	NUMERICO 5 Cifre

ARCHIVIO UFFICI :

MATRICOLA	NOME	INSERIMENTO FOTOCOPIE FATTE
NUMERICO 5 cifre	CARATTERE 15 Caratteri	NUMERICO 5 Cifre

ARCHIVIO CLASSI :

MATRICOLA	NOME	INSERIMENTO FOTOCOPIE FATTE
NUMERICO 5 cifre	CARATTERE 15 Caratteri	NUMERICO 5 Cifre

ESEMPIO DI ARCHIVIO STORICO RIFERITO AGLI STUDENTI :

MATRICOLA	COGNOME	NOME	CLASSE	DATA	FOTOCOPIE FATTE	FOTOCOPIE RIMANENTI
NUMERICO 5 Cifre	CARATTERE 15 Caratteri	CARATTERE 15 Caratteri	CARATTERE 4 Caratteri	DATA	NUMERICO 5 CIFRE	NUMERICO 5 CIFRE

dBASE IV PER DOS

CHE COSA E' IL PROGRAMMA IN dBASE:

Il programma eseguito in dBASE viene utilizzato come strumento per la gestione di archivi; ogni archivio è visto come una tabella.

Ad esempio la tabella che segue nella pagina successiva:

ARCHIVIO STUDENTI :

MATRICOLA	COGNOME	NOME	CLASSE	TOTALE FOTOCOPIE
00001	Fausti	Herri	5AI	56
00201	Turelli	Cristian	5AI	32
00013	Bogliani	Daniele	5AI	12

Le righe e le colonne vengono chiamate rispettivamente “record” e “campi”. I record individuano oggetti diversi della tabella (nel nostro caso i vari studenti), mentre i campi individuano caratteristiche diverse di ciascun oggetto della tabella (la matricola, il cognome, il nome, la classe, il totale delle fotocopie svolte da ogni studente).

Le operazioni fondamentali che l’utente può compiere su ciascuna tabella sono essenzialmente tre:

Selezionare alcuni record della tabella in base a certe caratteristiche:

- Selezionare alcuni record in base a certe loro caratteristiche (ad esempio tutti gli studenti che abbiano fatto un numero di fotocopie superiore a di 10).
- Selezionare alcuni campi su cui operare.
- Creare una nuova tabella congiungendone altre già preesistenti.

Queste operazioni base sono comuni a tutti i programmi di gestione di archivi. La programmazione con dBASE è efficiente e flessibile; la lista che segue elenca alcune delle funzioni più importanti:

- Compilatore di programmi
- Editor di programmi
- Variabili di memoria e array
- Strutture di programmazione
- Inizializzazione del programma e variabili di sistema
- Finestre e gestioni colori
- Menu lineari (orizzontali) e popup (verticali)
- Schede video e validazione dei dati
- Trattamento speciale dei campi memo
- File indice di generazione
- Struttura speciale di ricerca per i file di database
- Relazioni multiple tra i file
- Report e stampe personalizzate
- Strumenti per garantire la sicurezza e l’integrità dei dati
- Debug di un programma
- Strumenti per distribuire un’applicazione

CONVENZIONI SUI PROGRAMMI:

Attenendosi alle convenzioni sotto riportate, i programmi dBASE risulteranno più facili da leggere e mantenere.

- Ogni programma dovrebbe cominciare con un’intestazione che include il nome del programma, il suo scopo, il nome del programmatore, la storia di eventuali modifiche e la data di creazione.
- I nomi assegnati ai file, ai campi, alle variabili di memoria e agli altri oggetti devono essere descrittivi: un nome come X non significherebbe nulla per gli altri addetti al programma e, con il tempo, neppure il programmatore che l’ha creato saprebbe interpretarlo. Non possono essere usati nomi di comandi o funzioni di dBASE per designare oggetti.
- E’ consigliabile commentare accuratamente i vari settori del programma. I commenti devono essere preceduti da un asterisco (*) se si trovano su una riga separata, oppure da due “e commerciali” (&&)

se si trovano sulla stessa riga di un'istruzione del programma. Seguendo tale operazione è possibile identificare immediatamente l'utilità di una determinata parte del programma.

- Al fine di seguire una convenzione spesso seguita dai programmatori, è preferibile che i comandi annidati all'interno di strutture di programmazione come IF...ENDIF, DO CASE...ENDCASE e SCAN... ENDSCAN siano rientranti di tre spazi. (Questo anche per una migliore lettura del programma).
- E' preferibile iniziare ogni nuova clausola del comando @....SAY...GET su una nuova riga, allineandola sotto la clausola precedente. Se un'istruzione è così lunga che non può essere scritta su una sola riga, essa può essere spezzata inserendo un punto e virgola (;) prima del ritorno a capo.

LE VARIABILI DI MEMORIA:

Una variabile di memoria corrisponde a una locazione di memoria temporanea dei dati. Il valore di una variabile di memoria viene registrato in memoria e designato con un nome che fa riferimento alla sua locazione. Le variabili di memoria possono contenere stringhe di caratteri, numeri, date o valori logici. Non vi sono variabili di memoria di tipo memo (sono costituiti da dati di tipo carattere, che dBASE IV conserva in un file separato).

Le variabili di memoria vengono utilizzate nei programmi per vari scopi: ad esempio, una variabile di memoria può contenere dati immessi direttamente dall'utente, dati operativi per i calcoli o messaggi visualizzati di frequente.

Il riferimento ad una variabile di memoria viene fatto attraverso il nome e non attraverso il contenuto. Si può pertanto variare il contenuto di una variabile di memoria senza cambiarne il nome. Questo significa che la stessa variabile di memoria può essere usata ripetutamente all'interno di un programma.

ASSEGNAZIONE DEL NOME ALLE VARIABILI DI MEMORIA:

Il nome di una variabile di memoria può essere costituito al massimo da 10 caratteri, e può comprendere lettere, cifre e caratteri di sottolineatura (gli spazi vuoti non sono consentiti). Il nome deve iniziare con una parola e non deve essere una parola riservata di dBASE (cioè un nome di comando o di funzione).

E' bene stabilire delle convenzioni per l'assegnazione del nome alle variabili di memoria :

- Il nome scelto deve descrivere la funzione svolta dalla variabile, ad esempio *costo* descrive una variabile contenente dati sui costi.
- Se la variabile corrisponde a un campo del file di database, è opportuno assegnarle un nome simile a quello del campo stesso.
- E' preferibile che i nomi delle variabili inizino con la lettera *m* o con i caratteri *m_* per evitare di confonderli con i nomi di campo.

Dal nome dovrebbe essere immediatamente chiaro il tipo di dato corrispondente alla variabile e se si tratta di una variabile globale o locale: ad esempio, il nome *lc_titolo* potrebbe indicare una variabile locale (l) con dati di tipo carattere (c) e contenente un titolo .

INIZIALIZZAZIONE DELLE VARIABILI DI MEMORIA :

Per inizializzare una variabile di memoria, occorre assegnarle a un valore. Il tipo della variabile corrisponderà automaticamente a quello del dato in esso inserito. Inizializzando una variabile di memoria si sovrascrive a qualsiasi variabile preesistente con lo stesso nome; se si vuole modificare una variabile di memoria, basta modificarne il valore. Il tipo della variabile corrisponderà a quello del nuovo dato.

Le variabili di memoria si possono essere inizializzare in due diversi modi:

- Usando il comando STORE.
- Usando l'equivalente istruzione di assegnazione : *promtl*

La tabella di seguito mostra alcune definizioni tipiche di variabili di memoria, si noti che le stringhe di tipo carattere sono delimitate e che le date vengono convertite nel tipo data con la funzione CTOD().

Tipo	Esempio
Carattere	<pre>mnome = "Giovanni " prompt1 = "Errato riprovare" STORE SPACE (20) TO mcognome STORE " " TO scelta, risposta</pre>
Numerico	<pre>qta_artic = 20 m_costo = 0.00 STORE 0.00 TO subtotale, totale, contanti</pre>
Data	<pre>oggi = DATE () compleanno = CTOD (" 23/05/96 ") data_trans = CTOD ()</pre>
Logico	<pre>finito = . T. STORE .F. TO cancellato, non_valido</pre>

INIZIALIZZAZIONE DEGLI ARRAY DI VARIABILI DI MEMORIA :

Un array di memoria è un insieme di variabili di memoria disposte in righe e colonne. Gli array di variabili di memoria possono essere unidimensionali o bidimensionali: nel primo caso, l'array ha una sola colonna, nel secondo caso ha due o più colonne. Il numero totale di elementi di un array è uguale al numero di righe moltiplicato per il numero di colonne; un array può contenere fino a 1.170 elementi.

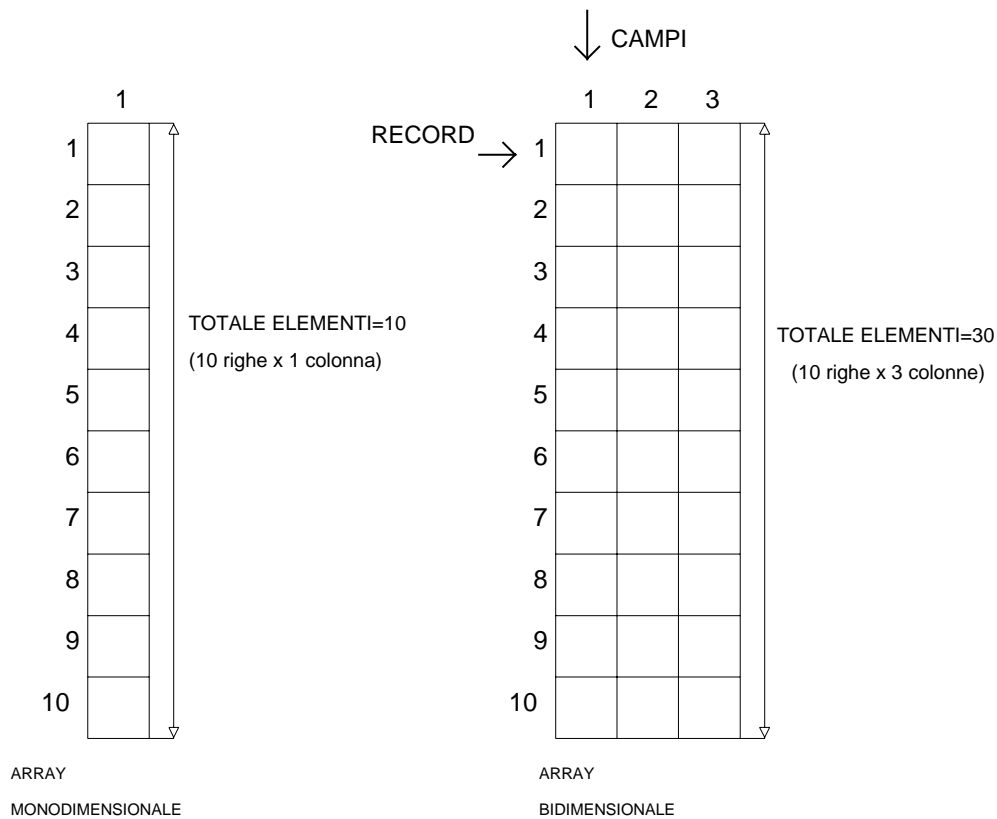
Per inizializzare un array di memoria, occorre definire la struttura con DECLARE e assegnare poi i valori degli elementi. I nomi degli array seguono le stesse convenzioni usate per i nomi delle variabili di memoria.

Il riferimento a un elemento di un array viene fatto mediante il nome array seguito dalle coordinate dell'elemento fra parentesi quadra : gli elementi array unidimensionali hanno una sola coordinata (la riga), mentre gli array bidimensionali ne hanno due (riga e colonna), separate da una virgola.

Per dichiarare un array bidimensionale di 33 righe si procede come segue:

```
DECLARE COSTO_arr [ 33,2 ]
```

Oltre a memorizzare una variabile di memoria in un array, è anche possibile assegnare il valore di un elemento di array a un altro elemento.



USO DI VARIABILI DI MEMORIA E DI ARRAY NEI PROGRAMMI :

Scopo

Gestire i messaggi presentati a video

Esempio

```
stamp_mess = "stampante non pronta "
m_suono = CHR (7)
@ 10,10 SAY m_suono + stamp_mess
```

Conservare i dati per la convalida e l'immissione in un file .bdf

```
mcod_artic = SPACE (10)
@ 10,10 SAY "indicare n. art."GET mcod_artic READ
DO Chk_dup WITH mcod_artic
```

VARIABILI PUBBLICHE E PRIVATE :

dBASE IV fornisce due tipi di variabili: pubbliche e private. Una variabile è pubblica quando disponibile in tutti i moduli di programma, indipendentemente da dove è stata inizializzata. Una variabile privata può essere usata solo nel modulo in cui è stata inizializzata e in tutti i moduli ad esso subordinati.

Nei programmi dBASE, le variabili di memoria sono private se non vengono dichiarate pubbliche.

Gli array sono pubblici per default: per creare un array privato, occorre usare l'istruzione **PRIVATE** prima di chiamare l'array stesso.

Se si vuole che la variabile sia disponibile nell'interno del programma, occorre dichiararla pubblica oppure inizializzarla nel modulo principale.

USO DEI FILE DI MEMORIA :

Per poter riutilizzare le variabili di memoria, occorre salvarle (con SAVE) in un file di memoria. In seguito, sarà possibile ripristinarle nella memoria attiva dal file di memoria utilizzando il comando RESTORE. dBASE IV dà a tutti i file di memoria l'estensione .mem, che può essere cambiata quando si salvano le variabili di memoria.

Il comando RESTORE serve per leggere nuovamente in memoria il contenuto di un file di memoria. Le variabili e gli array verranno ripristinati in memoria così come erano stati definiti.

Quando si ripristinano delle variabili di memoria da un file, tutte le variabili presenti in memoria vengono automaticamente cancellate, a meno che non venga esplicitamente chiesto da dBASE IV di conservarle aggiungendo la clausola ADDITTIVE. Per ripristinare il contenuto di Setup. mem senza cancellare altre variabili presenti in memoria, occorre quindi eseguire il comando:

RESTORE FROM Setup ADDITTIVE

dBASE IV non salva lo stato di una variabile (pubblica o privata) nel file di memoria. Se un file di memoria viene ripristinato in un programma, le variabili tornano private; occorrerà perciò utilizzare il comando PUBBLIC per renderle pubbliche.

E' possibile aggiungere e cancellare variabili in un file di memoria esistente. Dal punto, si ripristina il file di memoria con restore. Si inizializzano poi tutte le variabili o array supplementari che si desiderano cancellare con REALEASE tutte le variabili che devono essere eliminate. Infine, con SAVE, si salvano nuovamente le variabili nel file di memoria, che rifletterà i cambiamenti effettuati. Questa tecnica può essere utilizzata per la revisione dei file di memoria in fase di sviluppo di un programma.

ARCHITETTURA DI UN PROGRAMMA :

Le tecniche di programmazione strutturata consentono di scrivere programmi efficienti, facili da comprendere e da mantenere. A questo scopo dBASE IV permette di scrivere procedure e file di procedura, oltre a mettere a disposizione diverse strutture di programmazione.

LA PROGRAMMAZIONE STRUTTURATA :

La programmazione strutturata è un approccio organico alla programmazione che consente di produrre un programma di facile comprensione e manutenzione. Di seguito viene descritta una trattazione generale della programmazione strutturata: "Sviluppo top-down e programmazione modulare".

Sviluppo top-down :

Lo sviluppo top-down consiste nel suddividere l'applicazione nei suoi aspetti specifici; il frazionamento è gerarchico: le funzioni di livello superiore attivano quelle di livello inferiore. Lo sviluppo top-down conduce al concetto di programmazione modulare, in cui ogni operazione è codificata come singolo blocco di programma. Con lo sviluppo top-down di un applicazione, il menu principale di massimo livello controlla l'intero sistema e delega il lavoro a rami periferici. I menu intermedi o moduli sovrintendono al lavoro, svolto unicamente dai moduli di programma di livello inferiore. Questi si occupano degli aspetti relativi all'immissione, alla validazione, all'elaborazione e all'output dei dati.

Questo tipo di sviluppo presenta diversi vantaggi: le operazioni brevi sono più semplici da controllare e da codificare di quelle lunghe; suddividere un progetto di grandi dimensioni in una unità più maneggevole consente quindi di evitare errori di codifica. Inoltre con questo metodo i diversi moduli possono essere affidati a diversi programmatori, se il lavoro è svolto da un gruppo.

La programmazione modulare:

Quando si imposta un'applicazione procedendo dall'alto verso il basso, occorre identificare l'insieme dei compiti specifici che l'applicazione deve seguire. Ciascuno di questi compiti è l'elemento costitutivo di uno specifico modulo di programma.

Un programma modulare possiede le seguenti caratteristiche:

- Ogni modulo svolge un'unica mansione elementare
- Ogni modulo può essere richiamato solo dai moduli sovrastanti e può richiamare solo i moduli sottostanti.
- Ogni modulo dispone di un solo punto di accesso
- Completata l'esecuzione di un modulo, il controllo del programma generalmente torna al programma chiamante
- L'interazione tra i moduli è mantenuta al minimo
- Se possibile, il programma riutilizza i moduli

Contrariamente all'impostazione, che procede dall'alto verso il basso, la codifica del programma dovrebbe partire dalla base per arrivare al vertice. Ogni modulo deve essere progettato e scritto separatamente, cominciando dai moduli più specifici e procedendo verso quelli di livello più alto.

Quando si è completato un certo numero di moduli, essi potranno essere richiamati da un menu principale, in modo che gli utenti possano provare l'applicazione.

Codificando dal basso verso l'alto si potranno facilmente identificare i moduli usati ripetutamente nel corso dell'applicazione. Tali moduli possono essere inseriti in una libreria. Ad esempio, `libreria.prg` nell'applicazione Gestione moduli richiamati dalle sotto applicazioni per aggiungere, modificare e cancellare i record. L'uso delle librerie dà come risultato un programma più efficiente, file di dimensioni più ridotte e tempi di realizzazione più brevi, in quanto ogni modulo potrà essere documentato, verificato, su di esso potranno essere effettuate manutenzioni o revisioni lavorando su di un modulo alla volta.

PROCEDURE E FILE DI PROCEDURA :

dBASE IV rende possibile la programmazione strutturata mediante procedure, funzioni definite dall'utente e file di procedura. Una procedura è un modulo di programma dBASE IV eseguito da una procedura di un livello superiore, o procedura chiamante. Ogni procedura comincia con il comando :

`PROCEDURE <nome procedura>` e termina con l'istruzione `RETURN`.

Un programma può anche richiamare procedure provenienti da una libreria esterna, detta file di procedura. Solitamente, le procedure di un file di procedura sono routine essenziali richiamate da più di un programma in applicazioni diverse. Anche se è possibile aprire un solo file di procedura alla volta, si può accedere a un numero illimitato di procedure aprendo successivamente file di procedura diversi. Usando file di procedura si semplifica il compito di correggere o revisionare un programma, in quanto basta sostituire il file di procedura errato con uno appropriato. Per risparmiare memoria, un file di procedura non deve essere aperto finché il programma non lo utilizza effettivamente. Esso andrà poi tenuto aperto per tutto il tempo necessario, invece di essere chiuso e riaperto in continuazione : in tale modo si ridurrà al minimo il tempo di accesso al disco. Nella parte finale di un programma, è bene usare le istruzioni `CLOSE PROCEDURE TO` senza nome di file per chiudere l'ultimo file di procedura.

STRUTTURE DI PROGRAMMAZIONE :

Nella sua forma più semplice, un programma in dBASE è una sequenza di comandi dBASE memorizzati in un file. Il vero punto di forza di questo programma risiede nelle sue strutture di programmazione, queste controllano la logica o il flusso di un programma, ovvero il modo in cui il controllo passa da una parte all'altra del programma man mano che questo viene eseguito. In tal modo si può stabilire l'ordine, le condizioni e il numero di volte in cui i comandi vengono seguiti.

Un programma può seguire quattro tipi di schemi logici: processi sequenziali, selezioni, e interruzioni di un programma. dBASE fornisce strutture di programmazione per ognuno dei seguenti tipi:

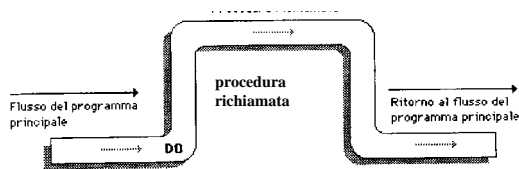
- Processi sequenziali -DO, RUN e CALL
- Strutture di selezione -IF...ENDIF e DO CASE...ENDCASE
- Strutture di iterazione -DO WHILE...ENDDO e SCAN...ENDSCAN

PROCESSI SEQUENZIALI :

La forma più semplice di processo sequenziale consiste nell'esecuzione dei comandi di un programma in sequenza. I processi sequenziali più complessi possono comprendere il richiamo di una procedura interna al programma o quella di un programma esterno.

Come abbiamo visto sopra abbiamo tre comandi sequenziali, noi analizziamo solo i primi due:

- DO esegue una procedura del programma principale o di un programma separato, ad esempio un file di procedura. Il comando DO [nome file] esegue la procedura indicata, che può trovarsi nel file di programma corrente o in un programma o file di procedura separata. Quando il compilatore incontra il comando DO, richiama la procedura o il programma, esegue le istruzioni contenute in quella routine e restituisce poi il controllo al livello appropriato.
- Il comando RUN esegue programmi esterni o programmi (.EXE) a livello di sistema operativo. La funzione RUN() può scaricare dBASE dalla RAM, consentendo di eseguire programma esterno.



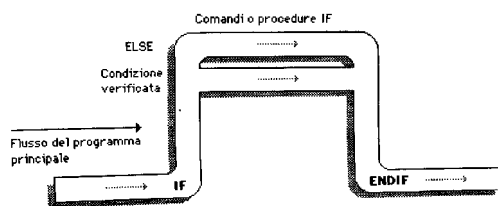
Richiamo di un programma o di una procedura

STRUTTURE DI SELEZIONE:

Un programma in dBASE può eseguire operazioni diverse in base alle circostanze o alle condizioni. dBASE IV possiede due strutture di programmazione condizionali:

IF...ENDIF e DO CASE...ENDCASE.

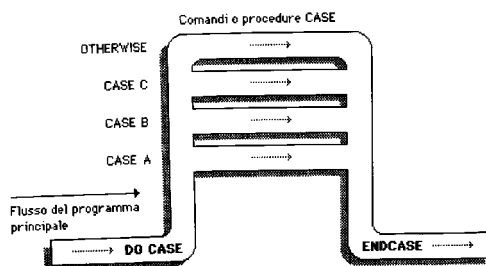
IF...ENDIF sono condizioni singole, quando dBASE incontra una struttura di questo tipo, esegue i comandi specificati nella struttura se si verifica la condizione (IF), in caso contrario li ignora.



Condizione singola (IF...ENDIF)

DO CASE...END CASE sono invece condizioni multiple, quando dBASE incontra una struttura di questo tipo, determina quale condizione è verificata tra quelle elencate, e segue i comandi associati. DO CASE...ENDCASE permette di controllare più alternative, mentre IF...ENDIF ne tratta al massimo due.

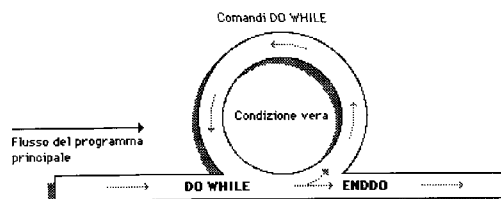
Le istruzioni CASE sono solitamente usate per eseguire un diverso comando o una diversa procedura per ogni voce di un menu.



Condizioni multiple (DO CASE...ENDCASE)

STRUTTURE DI ITERAZIONE :

Il processo denominato ciclo è quello in cui un programma ripete le stesse operazioni finché una determinata condizione si mantiene vera. Il programma esegue i comandi, torna al punto di partenza e li esegue nuovamente, ripetendo questo procedimento finché la condizione cessa di essere vera: a questo punto viene eseguito il comando immediatamente successivo al ciclo; dBASE tratta i cicli per mezzo la struttura DO WHILE...ENDDO.



Ciclo (DO WHILE...ENDDO)

L'uso forse più frequente dei cicli è la stampa o l'elaborazione di tutti i record di un file in data base. A tal fine occorre indicare a dBASE IV di continuare a eseguire i comandi del ciclo finché non viene raggiunta la fine del file (DO WHILE. NOT. EOF()).

Un altro impiego frequente dei cicli è la condizione DO WHILE.T.: questa provoca un ciclo infinito, in quanto .T. è sempre vero.

E' possibile uscire da un ciclo DO WHILE prima che dBASE abbia ultimato tutti i relativi comandi: l'opzione EXIT trasferisce il controllo nel punto immediatamente successivo al comando ENDDO che chiude il ciclo. In tal modo l'utente ha la possibilità di interrompere un ciclo prima che esso sia terminato.

IMPOSTAZIONE DELL'AMBIENTE :

Il codice d'impostazione è quella parte di un programma che si occupa di configurare l'ambiente operativo e inizializza le variabili di memoria globali usate in tutto il programma. Esso visualizza inoltre il menu principale e richiama i sottoprogrammi attivati da quest'ultimo.

LA DEFINIZIONE DI UN AMBIENTE OPERATIVO :

L'ambiente operativo di un programma comprende la configurazione di default dello schermo, della stampante, della tastiera e altri aspetti dell'ambiente in cui opera il programma. Usando i comandi SET e le variabili di memoria del sistema, è possibile creare un ambiente operativo di default del programma.

N.B. Quando ON o OFF sono scritti a caratteri maiuscoli indicano l'impostazione iniziale del comando o impostazione di default, nel caso contrario (on o off sono scritti a caratteri minuscoli) deve essere il programmatore a specificare la condizione desiderata.

Nel nostro caso i comandi SET utilizzati sono i seguenti:

- SET PROCEDURE TO [nome file procedura]

Apri un file di procedure e dà ai file di comando accesso alle procedure contenute.

- SET COLOR ON/OFF

Commuta tra i monitor a colori e monocromatici nei sistemi in cui sono entrambi disponibili.

- SET COLOR TO [normale] [, [luminoso]]
[, [bordo] [, [sfondo]]]]

Dove normale, luminoso, bordo e sfondo sono aree dello schermo alle quali si possono assegnare diversi attributi. Un attributo è una combinazione di caratteri che rappresentano il colore, la luminosità, il lampeggio; alcuni caratteri che si possono utilizzare in un attributo sono:

W= Bianco; G=Verde; N=Nero; B= Blu...

+ = Evidenziato; * = Lampeggiante.

- SET CURSOR ON /off

Determina se visualizzare il cursore (default) o se nascondere dallo schermo. Quando SET CURSOR è off il cursore è nascosto, per visualizzarlo si usa il comando SET CURSOR ON.

- SET HELP ON / off

Indica se, all'immissione di un comando sbagliato, compare un messaggio di aiuto o meno. Appariranno infatti le opzioni ANNULLA, MODIFICA e HELP; Si può scegliere di annullare il comando, di modificare la sintassi o di accedere al sistema di HELP, per verificare la corretta impostazione del comando.

- SET DEBUG on/ OFF

Serve a identificare la posizione degli errori nei programmi. Indica se l'output di SET ECHO viene inviato allo schermo o alla stampante, quando SET DEBUG è attivo l'output di SET ECHO viene inviato alla stampante evitando così interferenze tra le operazioni sullo schermo svolte dai programmi e i messaggi e/o le immagini generati dal processo di messa a punto del programma. Con SET DEBUG OFF l'output di SET ECHO viene mandato allo schermo. Con SET DEBUG ON e SET ECHO ON, non vengono inviati alla stampante i messaggi di errore.

- SET CLOCK TO [riga, colonna]

Determina la posizione dell'orologio sullo schermo.

- SET CLOCK on / OFF

Attiva e disattiva la visualizzazione dell'orologio in posizione default, a meno che ne sia stata indicata un'altra con SET CLOCK TO. (normalmente, SET CLOCK è OFF, quando è attivo la posizione di default è riga 0, colonna 68).

- **SET SCOREBOARD ON / off**

Se SET SCOREBOARD è attivo (ON) e SET STATUS OFF, dBASE IV visualizza gli indicatori dei tasti di scoreboard sulla riga 0 dello schermo, cioè gli indicatori dei tasti DEL, INS, CAPS, NUM, Rec Bloc e File Bloc. Se SET SCOREBOARD è impostato OFF dall'interno di un programma, lo schermo corrente viene cancellato.

Quando sono disattivati sia SET SCOREBOARD che SET STATUS, gli indicatori non vengono visualizzati.

- **SET STATUS on / OFF**

Attiva e disattiva la visualizzazione della riga di stato in fondo al video mentre si sta lavorando al Punto, in un programma e con i comandi di editing a tutto schermo come BROWSE, EDIT, APPEND e READ.

Con SET STATUS ON, la linea di stato visualizza il nome del comando impartito, il file attivo e il numero di record rispetto al numero totale dei record. Le informazioni non appaiono quando SET STATUS è OFF.

- **SET ECHO on /OFF**

Durante l'esecuzione di un file di comandi, SET ECHO invia allo schermo o alla stampante le singole righe di istruzioni man mano che vengono eseguite.

- **SET TALK ON / off**

Visualizza i numeri di record e le variabili di memoria man mano che vengono elaborati i comandi e visualizza i risultati di comandi come APPEND FROM, COPY, PACK, STORE e SUM.

- **SET TYPEAHEAD TO [esprNum]**

Specifica le dimensioni del buffer di tastiera. Le dimensioni per default sono 20 caratteri.

Gli utenti in grado di digitare molto velocemente possono ricorrere a questo comando per aumentare la capacità del buffer di tastiera, in modo che non vadino perdute le battute troppo in anticipo sul programma.

- **Riportiamo anche un altro comando: SELECT**

Sceglie un'area di lavoro in cui aprire un file di database o seleziona un'area di lavoro in cui è già aperto un file di database. Le aree di lavoro valide vanno da 1 a 40 e da A a J. Per aree di lavoro più grandi di 10, specificare l'area tramite numero o alias.

INSERIMENTO DI DATI MEDIANTE SCHEDE :

Le schede video dispongono i messaggi e gli spazi vuoti sullo schermo simulando i moduli di carta già famigliari agli utenti. Lo schermo di impostazione delle schede consente di definire i formati video senza bisogno di specificare le coordinate esatte dello schermo. Il generatore di schede produce tre file: un file di impostazione della scheda (.scr) contenente il codice interno impiegato dal generatore, un file formato (fmt.) contenente il codice dBASE e infine un file oggetto (.fmo) contenente il codice dBASE compilato. Il codice del file .fmt può essere modificato direttamente con l'editor di programmi.

DEFINIZIONE DI UN MENU POPUP :

Un menu popup è una lista di opzioni selezionabili racchiuse in una cornice. Per selezionare un'opzione, si digita la corrispondente lettera iniziale, oppure la si evidenzia portandovi sopra il cursore e si preme il tasto "return".

DEFINE POPUP e DEFINE BAR definiscono un menu popup e le corrispondenti opzioni, ma non definiscono le operazioni svolte da queste ultime. L'istruzione DEFINE POPUP specifica le coordinate dell'angolo superiore sinistro e dell'angolo superiore destro della cornice del menu. Le istruzioni DEFINE BAR definiscono le voci del menu, riferendosi alle righe del menu per numero: la prima riga (superiore)

corrisponde all'opzione 1, la seconda riga opzione 2, le voci del menu sono numerate in ordine verticale; la selezione si sposta nel menu per valori numerici crescenti. Si noti che l'opzione 1 in questo esempio definisce il titolo del menu e non un'operazione eseguibile. Con la parola SKIP si evita che la selezione si fermi sul titolo, dato che esso non è una voce selezionabile.

Riportiamo ora un esempio concreto :

```
PROCEDURE BAR.def
  DEFINE POPUP main-mnu FROM 2,58 TO 22,78;
  DEFINE BAR 1 OF main menu PROMPT"==MENU OPZIONI==" SKIP
  DEFINE BAR 2 OF main menu PROMPT "INSERIMENTO CODICE "
  DEFINE BAR 3 OF main menu PROMPT "GENERAZIONE STAMPE"
  DEFINE BAR 4 OF main menu PROMPT "SALVATAGGIO DATI"
  DEFINE BAR 5 OF main menu PROMPT "USCITA AL DOS"
RETURN
```

Per attivare un menu popup dopo averlo definito, è necessaria una sola riga di programma:

```
ACTIVATE POPUP main-menu
```

Quando si attiva un menu popup, dBASE IV conserva i popup già attivati sullo schermo e posiziona il nuovo menu sopra di essi. Sarà il programmatore a dover specificare il comando DEACTIVE POPUP nel momento in cui non sarà più necessario visualizzare il menu.

I FILE INDICE :

Gli utenti immettono dei dati in un file di database nell'ordine naturale; l'ordine in cui lo ricevono. Tuttavia, è spesso preferibile lavorare con i dati disposti in ordine alfabetico, numerico o in ordine di data. I comandi e le funzioni di creazione degli indici di dBASE IV permettono di controllare l'ordine in cui appaiono i dati.

Quando si associa a un file di database, dBASE IV crea un file indice contenente per ciascun record l'espressione chiave più un puntatore al record corrispondente nel file di data base.

Con i file indice è possibile produrre un output nell'ordine desiderato, esgurne le ricerche di dati in modo più efficiente e collegare file di data base.

DATA BASE IV può creare due tipi di file indice: i file indice (.ndx) sono adatti per un utilizzo occasionale. I file di indice multipli (.mdx) sono preferibili quando l'uso del database richiede sistematicamente quello dell'indice o degli indici associati.

SPIEGAZIONE DEL PROGRAMMA :

In questo paragrafo vedremo di spiegare il funzionamento generale del programma:

Dopo essere entrati nel programma in DATA BASE IV appare sullo schermo il menu principale, il quale risulta costituito dalle seguenti opzioni:

- 1)INSERIMENTO CODICE
- 2)GENERAZIONE STAMPE
- 3)SALVATAGGIO DEI DATI BACK UP
- 4)USCITA AL DOS

Analizziamo ora il significato e la funzione di ogni opzione:

1) INSERIMENTO CODICE: premendo su questa opzione appare la schermata che richiede l'inserimento del codice; usufruendo di una penna ottica si legge il codice a barre che viene decodificato e i dati relativi vengono immessi nel calcolatore. Non appena il codice viene inserito appare il prospetto storico delle operazioni svolte dall'utente. In tale schermata vengono riportate le ultime cinque operazioni fatte (mano mano che ne vengono effettuate delle nuove la più vecchia viene cancellata) in questa videata vengono indicati la matricola, il nome, il cognome, la data e il numero delle fotocopie fatte volta per volta. Nel caso in cui il codice letto dalla penna ottica sia sconosciuto, appare un messaggio e viene emesso un segnale acustico che avvisano l'operatore che quel codice non può essere accettato.

Dopo aver visualizzato lo storico si passa alla scheda indicante i dati caratteristici dell'utente (nome, cognome, matricola...) e un apposito campo dove l'operatore deve digitare il numero delle fotocopie fatte in quella occasione.

Il conteggio delle complessive fotocopie fatte avviene in modo automatico, il calcolatore compie infatti la differenza fra le fotocopie ancora disponibili all'utente e quelle fatte in quella particolare occasione. Premendo il tasto F1 avviene il salvataggio dell'operazione, mentre premendo il tasto ESC si ritorna al menu principale senza salvare nulla.

2) GENERAZIONE STAMPE: selezionando questa opzione si passa ad un menu di scelta che richiede per quale categoria di utenti si vuole effettuare la stampa (se si vogliono stampare i dati degli alunni, dei docenti, degli uffici o delle classi. Successivamente il programma richiede anche la selezione della condizione di stampa; si possono infatti generare stampe in base a diversi criteri. Di seguito ne riportiamo alcuni esempi:

- in base al cognome o al nome di un utente;
- in base al numero complessivo delle fotocopie fatte;
- in base alla classe;
- oppure in base ad un certo campo di matricole.

N.B. sono possibili delle operazioni di stampa in base a uno o più dei criteri sopra esposti.

Una volta determinato il criterio si può passare all'operazione vera e propria.

3) SALVATAGGIO DEI DATI BACK UP: viene visualizzata una schermata indicante diverse opzioni, esse sono:

- Richiede se il salvataggio deve avvenire su dischetti;
- Richiede se è necessario il recupero da altri dischetti;
- Oppure se si vuole ritornare al menu principale.

In seguito il programma richiede se si vuole veramente procedere o meno al salvataggio; quindi basta digitare "SI" per avviare l'operazione oppure se non si intende salvare si digita "NO".

4) USCITA AL DOS: premendo su questo comando si esce dal programma e si ritorna al dos.

IL CODICE A BARRE :

Il codice a barre si può definire come una simbologia che permetta di acquisire informazioni in modo automatico. E' uno dei sistemi più usati per la identificazione automatica di un oggetto o di una persona. Il limite del sistema è costituito dall'identificazione automatica di un qualsiasi oggetto presentato ad un adeguato sensore. L'elemento fondamentale di questo sistema è un appropriato sistema/sensore in grado di rilevare e discriminare i dati caratteristici dell'oggetto in esame rispetto all'ambiente circostante. Quindi il codice a barre ci permette di acquisire dei dati ed introdurli nel calcolatore in modo automatico; dati che altrimenti dovrebbero essere introdotti manualmente con notevole lentezza ed elevate possibilità d'errore. I codici a barre si basano su una codifica binaria, come si usa nel linguaggio dei microprocessori. Le diverse esigenze e i miglioramenti via via apportati hanno portato allo sviluppo di diversi tipi di codici a barre.

STORIA DEL CODICE A BARRE :

Il codice a barre è nato come sistema grafico adatto a consentire la lettura automatica dei caratteri attraverso lettura ottica o decodifica elettronica. Le origini del codice risalgono al 1949. Nel 1960 iniziano negli U.S.A. i primi studi sulla possibilità di utilizzare il codice sulle confezioni dei prodotti di largo consumo per diminuire le file alle casse dei supermercati. Nel 1970 vengono varati i primi progetti della standardizzazione della codifica, che viene denominata UPC (Universal Product Code). In Europa gli studi iniziarono separatamente in diversi paesi nel 1967/68; nel 1967 in Germania diedero origine al sistema BAN e nel '68 in Francia al sistema GENCOD. Le esigenze che scaturirono dagli scambi internazionali diedero origine ad un sistema che fosse compatibile con i sistemi allora utilizzati; tale sistema venne chiamato EAN (European Article Numbering).

CARATTERISTICHE DEL CODICE A BARRE :

Oltre al codice a barre esistono altri sistemi di acquisizione dati che citiamo solamente come quello magnetico o il sistema OCR che sfrutta il riconoscimento di caratteri alfanumerici.

La scelta del sistema in codice a barre rispetto ad altri sistemi di acquisizione dati è determinata da diverse esigenze:

- riconoscimento, controllo e conteggio di oggetti o documenti;
- sicurezza nella leggibilità;
- velocità di acquisizione dati;
- economia di investimento ed esercizio;

Il codice può essere stampato direttamente sulla confezione oppure su etichette autoadesive, senza richiedere supporti speciali. Il codice a barre può essere inoltre stampato in diverse dimensioni, senza che ci siano problemi di lettura.

Esistono diversi tipi di lettori ottici che vengono scelti in base alle esigenze e sono:

- penne ottiche manuali;
- scanner a raggio laser manuali o automatici;

I sistemi di identificazione che si basano sui codici a barre hanno le seguenti caratteristiche:

- **Velocità:** la velocità di acquisizione di un dato da parte di un computer mediante lettura di un codice è sicuramente molto elevata rispetto agli altri sistemi di tipo tradizionale;
- **Precisione:** i codici a barre utilizzano un check digit (cifra di controllo) che li rende praticamente esenti da errori;
- **Automazione:** l'acquisizione dei dati con il codice a barre può essere totalmente automatica;
- **Affidabilità:** mediante l'uso di una cifra di controllo (check digit) si ottiene un'altissima affidabilità delle informazioni ricevute. L'errore di lettura per sostituzione di un carattere va circa da 1/10 milioni a 1/30 milioni a seconda del tipo di codice.

COSA SONO I CODICI A BARRE:

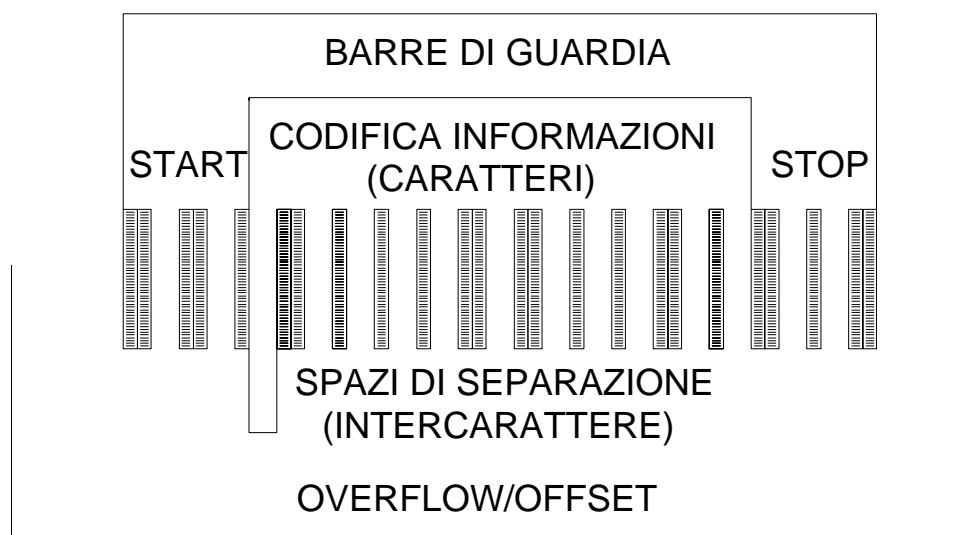
I codici a barre sono la codifica di caratteri numerici e alfanumerici sotto forma di barre scure e chiare di solito nere e bianche. Le codifiche sono costituite in modo binario. I caratteri possono essere codificati utilizzando sia le barre che gli spazi in questo ultimo caso diventano barre anche questi ultimi. In altri casi gli spazi fra le barre invece non sono di nessun significato, tranne quello di separare le barre nere. Il codice può essere di lunghezza fissa oppure variabile. Come unità di misura si assume la larghezza minima di una barra. I codici inoltre sono dotati di barre di start e di stop di modulo fisso caratteristiche per ogni codice e poste all'inizio e alla fine dello stesso. Se il codice è dotato di check digit viene anch'esso espresso in barre e posto in genere fra le barre di codifica e quelle di stop. Il sistema che effettua la lettura e la decodifica interpreta il check digit per soddisfare l'algoritmo di verifica del codice, che solitamente può essere letto in modo bidirezionale. Le applicazioni dei codici a barre sono numerose ma il settore nel quale sono più impiegati è quello del conteggio e movimentazione delle merci di un magazzino. All'atto dell'ingresso al magazzino o alla vendita (quindi all'uscita del magazzino) il codice viene letto, decodificato e inviato all'elaboratore il quale provvede a movimentare il magazzino e ad effettuare tutte le altre operazioni necessarie.

LA CODIFICA:

I codici a barre sono costituiti da una successione di barre e di spazi che contengono l'informazione codificata in modo binario. Gli elementi che costituiscono il codice a barre sono l'insieme di tutti gli elementi (sia barre che spazi). Questo insieme può essere diviso in sottoinsiemi che presentano caratteristiche diverse:

- Elementi del codice in cui sono contenute le informazioni (caratteri).
- Elementi di controllo: barre di guardia (start- stop) e spazi di separazione (intercarattere).
- Zona di overflow: zona bianca o di rispetto, zona di offset.

Questa zona di offset, è lo spazio che deve essere lasciato per consentire al decodificatore il corretto riconoscimento delle caratteristiche del codice. La larghezza deve essere di circa 10 volte la larghezza minima di una barra stretta, che per i codici ad alta risoluzione è di 2,5 mm. Ogni tipo di codice presenta una propria codifica e definizione caratteristica per ognuno di questi parametri. Sotto è riportato un esempio di codice a barre con i propri elementi significativi.



I codici si dividono in:

- codici con elementi bidimensionali
- codici con elementi pluridimensionali

Gli elementi bidimensionali del codice (barre e spazi) possono assumere due spessori, elemento largo (L) ed elemento stretto (S) o (X).

Gli elementi pluridimensionali dei codici possono assumere più spazi diversi e ciascuno multiplo dell'elemento stretto di spessore unitario (U) o (X).

Fattore importante, che caratterizza i codici a barre, per la loro lettura, è il contrasto. E' definito come il rapporto tra la riflettanza degli elementi scuri (barre) e gli elementi chiari (spazi) del simbolo.

Ogni tipo di codifica è caratterizzata da diversi fattori:

- risoluzione
- spessore degli elementi stretti o modulo unitario
- rapporto fra gli elementi stretti e quelli larghi (rapporto di stampa)
- sequenza di barre e spazi
- numero di caratteri rappresentabili ovvero lunghezza del codice (alcuni presentano infatti lunghezza fissa)
- tipo di caratteri rappresentabili
- sequenza delle barre di guardia
- tolleranza
- check digit
- lunghezza del simbolo
- mono/bidimensionalità
- codice continuo
- codice discreto
- codice self-checking

RISOLUZIONE:

Tra gli elementi stretti del codice e quelli larghi o multipli vi è un rapporto. Lo spessore dell'elemento stretto può assumere diversi valori e mano a mano che lo si diminuisce, il simbolo diviene sempre più compatto e dimensionalmente più corto mentre aumenta la sua risoluzione. Il parametro che indica i vari livelli di risoluzione è la densità, i codici possono essere rappresentati in:

- altissima densità o risoluzione (dove l'elemento stretto è inferiore a 0,15 mm e quello largo a meno di 0,375)
- alta densità (dove l'elemento stretto è uguale a 0,15 mm e quello largo 0,375)
- media densità (dove l'elemento stretto è uguale a 0,25 mm e quello largo a 0,635)
- bassa densità (dove l'elemento stretto è uguale a 0,38 mm e quello largo a 0,952)
- densità molto bassa (dove l'elemento stretto è superiore a 0,38 mm e quello largo a 0,952).

L'aumento della densità del codice è strettamente collegato anche alla tecnologia di stampa che deve essere in grado di stampare gli elementi con precisione maggiore quanto più è elevata la risoluzione che si desidera ottenere. La risoluzione del codice influenza anche il dispositivo di lettura in quanto la risoluzione del codice deve rientrare nel campo di risoluzione del lettore.

RAPPORTO TRA LO SPESSORE DEGLI ELEMENTI STRETTI E QUELLI LARGHI:

Il rapporto tra l'elemento stretto (X) e quello largo, detto anche rapporto di stampa, è generalmente entro un valore da 2:1 a 3:1.

SEQUENZA DI BARRE E SPAZI:

La sequenza di barre e spazi consente la codifica dei caratteri. Questa codifica, infatti, è ottenuta assegnando ad ogni carattere una sequenza di barre e di spazi.

NUMERO DI CARATTERI RAPPRESENTABILI:

E' il numero dei caratteri inteso come quantità. Ciò determina la lunghezza del simbolo. Alcuni codici, infatti presentano un numero fisso di caratteri codificabili, e quindi lunghezza fissa di quest'ultimo. Altri invece hanno un numero variabile di caratteri. Ciò però non va confuso con la lunghezza fisica del codice stampato.

TIPI DI CARATTERI RAPPRESENTABILI:

Ogni tipo di codice ha una caratteristica tabella di decodifica nella quale sono elencati i caratteri e la relativa codifica. Tutti i codici possono codificare caratteri numerici anche decimali, sono alcuni invece possono codificare caratteri speciali e alfabetici.

SEQUENZA DELLE BARRE DI GUARDIA:

La sequenza degli elementi (spazi e barre) che costituiscono le "barre di guardia" è caratteristica di ogni tipo di codice. Questa sequenza codifica caratteri di start, stop o center mark (di centro codice).

TOLLERANZA:

Per ogni tipo di codice sono stati stabiliti dei valori di tolleranza applicabili al singolo elemento del codice. La tolleranza è lo scostamento dimensionale ammesso dal valore nominale di larghezza assunto dall'elemento stretto.

CHECK DIGIT:

Il check digit, o cifra di controllo, viene utilizzato per aumentare la sicurezza di lettura del codice. Viene aggiunto ai caratteri informativi del codice ed è generato da algoritmi matematici comuni a diversi codici. Il check digit rende altamente affidabile il codice ed è esso può essere obbligatorio oppure opzionale.

LUNGHEZZA DEL SIMBOLO:

La lunghezza del simbolo è generalmente comprensiva delle zone di overflow e viene determinata da diversi fattori:

- numero dei caratteri rappresentati
- rapporto di stampa
- spessore del modulo unitario
- larghezza delle zone di overflow (minimo $10 \cdot$ lo spessore del modulo stretto)
- rapporti di spessore tra spazi e modulo della barra.

MONO-BIDIMENSIONALITA':

In questo caso si intende una mono-bidimensionalità dell'intero simbolo. I codici bidimensionali sono quelli che si sviluppano su diverse righe, ognuna correlata all'altra.

CODICE CONTINUO-DISCRETO

Un codice è continuo quando non vi sono spazi tra i caratteri codificati in barre. Diversamente da un codice discreto dove ogni carattere è separato dall'altro da uno spazio intercarattere non significativo, cioè che non contiene alcuna informazione.

CODICE SELF CHECKING

Un codice è self-checking quando nella codifica binaria di ogni carattere è contenuto un bit di parità con funzione di autocontrollo di ogni carattere del codice. Il livello di controllo non è molto elevato, è quindi sempre opportuno ricorrere al check-digit che effettua un controllo di livello superiore.

IL CODICE 39

-Questo codice è bidirezionale e di lunghezza variabile.

-Il codice oltre ad avere codifica alfanumerica di 10 cifre e 26 caratteri alfabetici, ha 7 caratteri speciali.

-I caratteri codificabili (maiuscoli) sono 43 di cui ogni singolo carattere è codificato in 9 elementi costituiti da 5 barre e 4 spazi, di cui 3 sono di doppia unità e i restanti 6 sono stretti.

-La serie delle barre di ogni carattere è separata da uno spazio di intercarattere sottile con la caratteristica di poter assumere uno spessore fino ad un multiplo di 3 volte.

-Il carattere "*" (asterisco) è quello di Start e di Stop che sono identici.

-L'altezza minima delle barre che compongono l'etichetta è di 6,5mm o il 15% della lunghezza del simbolo per letture con lettori ottici a contatto (penne ottiche).

-Lo spessore nominale delle barre sottili è 0,19mm ad alta risoluzione. Il rapporto fra le barre strette e quelle larghe varia da 2:1 a 3:1.

-Il codice è self-checking e di tipo discreto.

-La densità delle informazioni codificabili non è elevata per le caratteristiche di codifica del codice, cioè, con questo codice si occupa più spazio nella lunghezza del simbolo che non con altri codici che hanno un grado di compattezza più elevato.

La tolleranza di questo codice dipende dal rapporto di stampa e dalle dimensioni del modulo unitario (barra o spazio stretto) e può essere calcolata per mezzo di una espressione:

$$\pm T = \frac{4}{27} \cdot \frac{(R-2)}{3} \cdot X$$

La lunghezza del modulo si calcola con la seguente espressione:

$$L = (N \cdot (3R + 6) + (6R + 12) + (N + 1) + RS) + 2 \cdot O$$

Se si desidera conoscere il numero di caratteri che può essere rappresentato entro una determinata dimensione del simbolo, anche in relazione alla dimensione fisica dell'etichetta da applicare al prodotto o all'area disponibile per la stampa, l'espressione è la seguente:

$$N = \frac{\frac{(L-2O)}{X} - (6R + 12) - RS}{3R + 6 + RS}$$

Dove nelle formule i simboli significano:

L = Lunghezza del simbolo compresa la zona di overflow

N = Numero di caratteri codificati o da codificare

X = Spessore dell'elemento (modulo unitario)

R = Rapporto di stampa

RS = Rapporto spessori tra spazio intercarattere e elemento
 O = Larghezza della zona di overflow (minimo 10 volte X)
 T = Tolleranza %

La codifica del codice 39 è rappresentata nella tabella sottostante:

TABELLA DI CODIFICA

VN	CARATTERE	B	S	B	S	B	S	B	S	B
0	0	0	0	0	1	1	0	1	0	0
1	1	1	0	0	1	0	0	0	0	1
2	2	0	0	1	1	0	0	0	0	1
3	3	1	0	1	1	0	0	0	0	0
4	4	0	0	0	1	1	0	0	0	1
5	5	1	0	0	1	1	0	0	0	0
6	6	0	0	1	1	1	0	0	0	0
7	7	0	0	0	0	0	0	1	0	1
8	8	1	0	0	0	0	0	1	0	0
9	9	0	0	1	0	0	0	1	0	0
10	A	1	0	0	0	0	1	0	0	1
11	B	0	0	1	0	0	1	0	0	1
12	C	1	0	1	0	0	1	0	0	0
13	D	0	0	0	0	1	1	0	0	1
14	E	1	0	0	0	1	1	0	0	0
15	F	0	0	1	0	1	1	0	0	0
16	G	0	0	0	0	0	1	1	0	1
17	H	1	0	0	0	0	1	1	0	0
18	I	0	0	1	0	0	1	1	0	0
19	J	0	0	0	0	1	1	1	0	0
20	K	1	0	0	0	0	0	0	1	1
21	L	0	0	1	0	0	0	0	1	1
22	M	1	0	1	0	0	0	0	1	0
23	N	0	0	0	0	1	0	0	1	1
24	O	1	0	0	0	1	0	0	1	0
25	P	0	0	1	0	1	0	0	1	0
26	Q	0	0	0	0	0	0	1	1	1
27	R	1	0	0	0	0	0	1	1	0
28	S	0	0	1	0	0	0	1	1	0
29	T	0	0	0	0	1	0	1	1	0
30	U	1	1	0	0	0	0	0	0	1
31	V	0	1	1	0	0	0	0	0	1
32	W	1	1	1	0	0	0	0	0	0
33	X	0	1	0	0	1	0	0	0	1
34	Y	1	1	0	0	1	0	0	0	0
35	Z	0	1	1	0	1	0	0	0	0
36	-	0	1	0	0	0	0	1	0	1
37	.	1	1	0	0	0	0	1	0	0
38	SPACE	0	1	1	0	0	0	1	0	0
39	\$	0	1	0	1	0	1	0	0	0
40	/	0	1	0	1	0	0	0	1	0
41	+	0	1	0	0	0	1	0	1	0
42	%	0	0	0	1	0	1	0	1	0
START/STOP	*	0	1	0	0	1	0	1	0	0

B = Barra **S** = Spazio **Barre/Spazio stretti** = 0 **Barre/Spazio larghi** = 1

VN = Valore numerico assegnato a ogni carattere, utile per il calcolo del check digit. Il check digit viene introdotto con adeguato algoritmo matematico e stampato in codice a barre alla destra della codifica dei caratteri, prima delle barre di stop.

CALCOLO DEL CHECK DIGIT:

Per calcolare il check digit si fa la somma di tutti i valori numerici dei caratteri da codificare e la si divide per un valore fisso.

Se supponiamo ad esempio di voler rappresentare la parola BAR, dovremmo procedere nel seguente modo:

1) Sommare i valori numerici (VN) corrispondenti ai caratteri codificati

$$\begin{array}{r} \text{B} \quad \text{A} \quad \text{R} \\ 11 + 10 + 27 = 48 \end{array}$$

2) Dividere il risultato per il valore del modulo fisso

$$48 : 43 = 1 \quad \text{resto} = 5$$

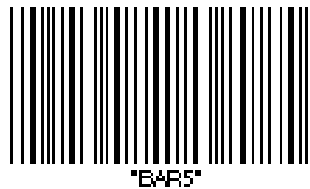
3) Ricercare nella tabella di codifica il carattere corrispondente alla cifra del resto trovato

$$5 = 5 \text{ check digit (CD)}$$

5) Serie di caratteri da codificare nel simbolo completo di check digit

BAR 5

6) Rappresentazione grafica in codice 39



Bibliografia :

L. Serasini, I CODICI A BARRE, Jackson libri
dBASE IV per dos 2.0 (il linguaggio), Borland
dBASE IV per dos 2.0 (per chi sviluppa), Borland

PROGRAMMA IN dBASE IV

LISTATO SORGENTE DEL PROGRAMMA GESTIONE

IL PROGRAMMA "GESTIONE" PROVVEDE AD IMPOSTARE L'AMBIENTE DI GESTIONE DEL PROGRAMMA. ESSO DETERMINA INNANZITUTTO LE IMPOSTAZIONI DEL MONITOR DETERMINANDO SE TRATTASI MONITOR A COLORI O MONOCROMATICO; SI PASSA POI ALLA DEFINIZIONE DEL MENU DI POPUP, CIOE' DI UNA LISTA DI OPZIONI SELEZIONABILI, E ALLA CONSEGUENTE ATTIVAZIONE DELLE PROCEDURE CONTENENTI LE ISTRUZIONI PER LE OPERAZIONI ASSOCIATE A CIASCUNA VOCE DEL MENU. ESSE SONO INSERIMENTO CODICE, GENERAZIONE STAMPE, BACKUP DEI DATI ED USCITA DAL PROGRAMMA E RITORNO AL DOS.

```
PROCEDURE Gestione
SET PROCEDURE TO Libreria
  CLEAR ALL
release all
DO Set_env
  PUBLIC gl_MainMenu

IF ISCOLOR()
  c_normal = "W/B,GR+/Bg,B"
  c_pop    = "B/W,GR+/b ,W+/R"
  red     = "R/W"
  blue    = "B/W"
  lt_blue = "W/BG"
ELSE
  STORE "W+/N,N/W" TO c_normal, c_pop
  STORE "W"      TO red, blue
  STORE "N/W" TO lt_blue
ENDIF

DO Main_def

PUBLIC gl_Error

gl_Error = .F.
DO WHILE BAR() <> 13 .and. .not. gl_error
  SET COLOR TO &c_normal.
  CLEAR
  DO Title
  SET COLOR TO &c_pop.
  ACTIVATE POPUP mainmenu
  DO Main
ENDDO
RELEASE gl_MainMenu
DO Rest_env
ON ERROR
ON KEY LABEL F1
CLEAR ALL
CLOSE ALL
CLEAR
RETURN

PROCEDURE Title
CLEAR
@ 2,13 TO 5,63 DOUBLE COLOR &blue.
```

```

@ 2,13 FILL TO 5,63 COLOR &BLUE.
SET COLOR TO &BLUE.
@ 3,23 SAY " I.P.S.I.A. MORETTO BS"
@ 4,15 SAY "SISTEMA DI GESTIONE CODICE A BARRE CON dBASE IV"
SET COLOR TO &c_normal.
RETURN

```

```

PROCEDURE Main_def
DEFINE POPUP mainmenu FROM 7,19 TO 22,58 ;
MESSAGE "Premere l'iniziale dell'opzione del menu, o selezionare l'opzione e battere "+CHR(17)+CHR(217)
DEFINE BAR 1 OF mainmenu PROMPT "===== MENU PRINCIPALE =====" SKIP
DEFINE BAR 4 OF mainmenu PROMPT " INSERIMENTO CODICE"
DEFINE BAR 6 OF mainmenu PROMPT " GENERAZIONE STAMPE"
DEFINE BAR 8 OF mainmenu PROMPT " SALVATAGGIO DATI"
DEFINE BAR 13 OF mainmenu PROMPT " USCITA AL DOS"
ON SELECTION POPUP mainmenu DEACTIVATE POPUP
RETURN

```

```

PROCEDURE Main
DO CASE
CASE BAR() = 4
DO BARRE
CASE BAR() = 6
do richiest
DO STAMPA
CASE BAR() = 8
DO Salvatag
CASE BAR() = 13
RELEASE ALL
SET CLOCK ON
DO Colo_rese
QUIT
ENDCASE
RETURN

```

LISTATO SORGENTE DEL PROGRAMMA BARRE

IL PROGRAMMA BARRE CONTIENE LE ISTRUZIONI PER LA GESTIONE DEI DATABASE, IVI COMPRESI I DATABASE PER LA GESTIONE DELLE OPERAZIONI SULLO "STORICO". IL PROGRAMMA IMPOSTA I COMANDI SET DI DEFINIZIONE DELL'AMBIENTE OPERATIVO, L'ASSEGNAZIONE DELLE VARIABILI DI MEMORIA, LA PRESENTAZIONE DEL TITOLO, LA RICERCA NELLO "STORICO" DELLE ULTIME CINQUE OPERAZIONI EFFETTUATE E INFINE L'INSERIMENTO DATI RELATIVO ALLE FOTOCOPIE ESEGUITE.

```

SET PROCEDURE TO LIBRERIA
SET COLOR TO &C_STANDARD
set typeahead to 20
set escape on
SET BELL on
SET CURSOR ON
SET HELP OFF
SET DEBUG off
SET CLOCK ON
SET SCOREBOARD OFF
SET STATUS off
SET ECHO OFF
SET TALK off
select 1

```

```

use docenti index docenti.ndx
select 2
use matr9596 index matr9596.ndx
SELECT 3
USE UFFICI INDEX UFFICI.NDX
SELECT 4
USE CLASSI INDEX CLASSI.NDX
PUBLIC MAT1,mat2,mat3,mat4,mat5,OGGI
MAT = SPACE (7)
mat2 = space(1)
mat1 = space(5)
mat3 = space(5)
mat4 = SPACE(5)
mat5 = SPACE(5)
oggi=date()
@11,0 TO 11,79 DOUBLE
set color to &c_pop
@2,6 TO 5,72 DOUBLE
@2,6 FILL TO 5,72 COLOR B/W
@3,7 SAY " ISTITUTO PROFESSIONALE DI STATO PER L'INDUSTRIA E L'ARTIGIANATO" COLOR B/W
@4,21 SAY " 'M O R E T T O' - B R E S C I A " COLOR B/W
*@20,29 SAY "<ESC> PER USCIRE" COLOR W/B*
@9,26 say "INSERIMENTO CODICE:" color w+/b
@9,46 get mat picture "A99999A"
read
store substr(mat,1,1) to mat2

DO CASE
    case mat2 = "D"
        store substr(mat,2,5) to mat1
        do ricerca_DOCENTI
        case mat2= "A"
            store substr(mat,2,5) to mat3
            do ricerca_ALUNNI
            CASE MAT2= "U"
                STORE SUBSTR(MAT,2,5) TO mat4
            DO RICERCA_UFFICI
            CASE MAT2= "C"
                STORE SUBSTR(MAT,2,5) TO MAT5
            DO RICERCA_CLASSI
            CASE mat2=""
                DO warnbell
                DO ALERT
ENDCASE
return to barre

PROCEDURE ricerca_DOCENTI
    SELECT 1
    seek val(mat1)
    IF .NOT. FOUND()
        DO WARNBELL
        DO ALERT
    ENDIF
    DO STO
    DO STO1
PROCEDURE STO1
    SELECT 1
    USE docenti index docenti.ndx
    SEEK VAL(MAT1)
    SET FILTER TO MATR=VAL(MAT1)
    INDEX ON MATR TO DOCENTL.NDX

```

```

replace all data with oggi
replace FOTOCOPIE with 0
REPLACE all TOTale WITH totale+fotocopia
REPLACE FOTocopia WITH 0 FOR FOTocopia<>0
set fields to
MATR/R,COGNOME/R,NOME/R,MATERIA/R,fotocopia,DATA/R,TOTALE=fotocopia+totale
SET FORMAT TO VITOA01
ON KEY LABEL F1 KEYBOARD "{CTRL-END}"
EDIT NOAPPEND NOINIT
CLOSE INDEX
CLOSE DATABASES
RETURN
PROCEDURE STO
PUBLIC RECORD
OGGI=DATE()
USE STORICO order matr
APPEND FROM DOCENTI FOR MATR=VAL(MAT1)
REPLACE ALL DATA WITH OGGI
REPLACE all FOTocopia WITH FOTocopia
count for matr=val(mat1) to record
DO CASE
    case record<6
        USE STORICO
        index on DTOS(DATA) to storico
        SCAN FOR MATR=VAL(MAT1)
        COUNT FOR MATR=VAL(MAT1)
        ENDSKAN
        DELETE FOR FOTOCOPIE=0
        PACK
        DO WINDO

    case record=>6
        USE STORICO
        INDEX ON DTOS(DATA) TO STORICO
        set filter to matr=val(mat1)
        reindex
        DELETE ALL FOR FOTOCOPIE=0
        PACK
        go top
        delete
        pack
        reindex
        do windo
ENDCASE
CLOSE DATABASES

procedure windo
@12,0 to 12,79 double
@12,0 FILL TO 24,79 COLOR B/BG
SET COLOR TO W+/BG
define window parziale from 15,0 to 21,79
*set filter to matr=val(mat1)
SCAN FOR MATR=VAL(MAT1)
BROWSE noedit noappend noinit fields
matr/r,cognome/r,nome/r,materia/r/4,data/r,fotocopia/r,totale=fotocopia+totale;
compress window parziale
skip
RISPOSTA =" "
@ 23,28 SAY "SI VUOLE CONTINUARE? (S/N)";
GET RISPOSTA PICTURE "!";
VALID RISPOSTA $ "SN";

```

```

ERROR "DIGITARE S o N"
READ
DO CASE
CASE RISPOSTA ="S"
RETURN
CASE RISPOSTA ="N"
RETURN TO MASTER
ENDCASE
ENDSCAN
set filter to
SET CURSOR ON
RETURN

```

PROCEDURE ricerca_ALUNNI

```

SELECT 2
seek val(mat3)
IF .NOT. FOUND()
DO WARNBELL
DO ALERT
ENDIF
DO STO2
DO STO3

```

PROCEDURE STO3

```

SELECT 2
USE MATR9596 index MATR9596.ndx
SEEK VAL(MAT3)
SET FILTER TO MATR=VAL(MAT3)
INDEX ON MATR TO MATR9596.NDX
replace all data with oggi
REPLACE FOTOCOPIE WITH 0
REPLACE ALL TOTALE WITH TOTALE+FOTOCOPIE
REPLACE FOTOCOPIE WITH 0 FOR FOTOCOPIE <>0
set fields to

```

MATR/R,COGNOME/R,NOME/R,CLASSE/R,fotocopie,DATA/R,totale=FOTOCOPIE+TOTALE

```

SET FORMAT TO VITOA02
ON KEY LABEL F1 KEYBOARD "{CTRL-END}"
EDIT noappend noinit
CLOSE INDEX
CLOSE DATABASES
RETURN

```

PROCEDURE STO2

```

PUBLIC RECORD
OGGI=DATE()
USE STORICO2 ORDER MATR
APPEND FROM MATR9596 FOR MATR=VAL(MAT3)
REPLACE ALL DATA WITH OGGI
REPLACE ALL FOTOCOPIE WITH FOTOCOPIE
COUNT for matr=val(mat3) to record
DO CASE
case record<6
use storico2
index on DTOS(DATA) to storico2
SCAN FOR MATR=VAL(MAT3)
COUNT FOR MATR=VAL(MAT3)
ENDSCAN
delete for fotocopie=0
pack
do windo1

case record=>6
use storico2

```

```

        index on DTOS(DATA) to storico2
        set filter to matr=val(mat3)
        REINDEX
        delete all for fotocopie=0
        pack
        go top
        delete
        pack
        reindex
        index on dtoc(data) to storico2
        do windo1
    ENDCASE
CLOSE DATABASES

```

```

procedure windo1
USE STORICO2
@ 12,0 TO 12,79 DOUBLE
@ 12,0 FILL TO 24,79 COLOR B/BG
SET COLOR TO W+/BG
define window parziale from 15,0 to 21,79
set filter to matr=val(mat3)
SCAN FOR MATR=VAL(MAT3)
BROWSE NOAPPEND NOEDIT NOMENU COMPRESS FIELDS

```

```

MATR/R,COGNOME/R, NOME/R,CLASSE/R,FOTOCOPIE/R,DATA/R,TOTALE=FOTOCOPIE+TOTALE WINDOW
PARZIALE

```

```

        RISPOSTA = " "
        @ 23,28 SAY "SI VUOLE CONTINUARE? (S/N)";
        GET RISPOSTA PICTURE "!";
        VALID RISPOSTA $ "SN";
        ERROR "DIGITARE S o N"
        READ
        DO CASE
        CASE RISPOSTA ="S"
        RETURN
        CASE RISPOSTA ="N"
        RETURN TO MASTER
        ENDCASE
        ENDSCAN
        SET FILTER TO
        SET CURSOR ON
        RETURN

```

```

PROCEDURE ricerca_UFFICI

```

```

    SELECT 3
    seek val(mat4)
    IF .NOT. FOUND()
    DO WARNBELL
    DO ALERT
    ENDIF
    DO STO4
    DO STO5
    PROCEDURE STO5
        SELECT 3
        USE UFFICI index UFFICI.ndx
        SEEK VAL(mat4)
        SET FILTER TO MATR=VAL(mat4)
        INDEX ON MATR TO UFFICI.NDX
        replace all data with oggi
        REPLACE FOTOCOPIE WITH 0
        REPLACE all TOTale WITH totale+FOTOCOPIE
        REPLACE FOTOCOPIE WITH 0 FOR FOTOCOPIE <> 0
    
```

```

set fields to MATR/R,NOME/R,FOTOCOPIE,data/r,TOTALE/R
SET FORMAT TO VITOA03
ON KEY LABEL F1 KEYBOARD "{CTRL-END}"
EDIT NOAPPEND NOINIT
CLOSE INDEX
CLOSE DATABASES
RETURN
PROCEDURE STO4
PUBLIC RECORD
OGGI=DATE()
USE storico3 ORDER MATR
APPEND FROM UFFICI FOR MATR=VAL(MAT4)
replace all data with oggi
REPLACE ALL FOTOCOPIE WITH FOTOCOPIE
COUNT FOR MATR=VAL(MAT4) TO RECORD
DO CASE
    case record<6
        USE STORICO3
        index on dtoS(data) to storico3
        SCAN FOR MATR=VAL(MAT4)
        COUNT FOR MATR=VAL(MAT4)
        ENDSKAN
        DELETE FOR FOTOCOPIE=0
        PACK
        do windo2

    case record=>6
        USE STORICO3
        INDEX ON DTOS(DATA) TO STORICO3
        set filter to matr=val(mat4)
        REINDEX
        DELETE ALL FOR FOTOCOPIE=0
        PACK
        go top
        delete
        pack
        reindex
        INDEX ON DTOS(DATA) TO STORICO3
        do windo2
ENDCASE
CLOSE DATABASES

procedure windo2
USE STORICO3
@ 12,0 to 12,79 double
@ 12,0 FILL TO 24,79 COLOR B/BG
SET COLOR TO W+/BG
define window parziale from 15,0 to 21,79
set filter to matr=val(mat4)
SCAN FOR MATR=VAL(MAT4)
BROWSE NOAPPEND NOEDIT NOMENU fields
matr/r,nome/r,FOTOCOPIE/R,data/r,totale=FOTOCOPIE+TOTALE;
compress window parziale
RISPOSTA =" "
@ 23,28 SAY "SI VUOLE CONTINUARE? (S/N)";
GET RISPOSTA PICTURE "!";
VALID RISPOSTA $ "SN";
ERROR "DIGITARE S o N"
READ
DO CASE

```

```

CASE RISPOSTA ="S"
RETURN
CASE RISPOSTA ="N"
RETURN TO MASTER
ENDCASE
ENDSCAN
set filter to
SET CURSOR ON
RETURN

```

PROCEDURE ricerca_CLASSI

```

use classi order matr
seek val(mat5)
IF .NOT. FOUND()
DO WARNBELL
DO ALERT
ENDIF
DO STO6
DO STO7

```

PROCEDURE STO7

```

SELECT 4
USE CLASSI index CLASSI.ndx
SEEK VAL(mat5)
SET FILTER TO MATR=VAL(mat5)
INDEX ON MATR TO CLASSI.NDX
replace all data with oggi
REPLACE FOTOCOPIE WITH 0
REPLACE all TOTale WITH totale+FOTOCOPIE
REPLACE FOTOCOPIE WITH 0 FOR FOTOCOPIE <> 0

```

```

set fields to MATR/R,CLASSE/R,FOTOCOPIE,data/r,TOTALE/R
SET FORMAT TO VITOA04
ON KEY LABEL F1 KEYBOARD "{CTRL-END}"
EDIT NOAPPEND NOINIT
CLOSE INDEX
CLOSE DATABASES
RETURN

```

PROCEDURE STO6

```

PUBLIC RECORD
OGGI=DATE()
USE storico4 ORDER MATR
APPEND FROM CLASSI FOR MATR=VAL(MAT5)
replace all data with oggi
REPLACE ALL FOTOCOPIE WITH FOTOCOPIE
count for matr=val(mat5) to record
DO CASE
    case record<6
        USE STORICO4
        index on dtos(data) to storico4
        SCAN FOR MATR=VAL(MAT5)
        COUNT FOR MATR=VAL(MAT5)
        ENDSCAN
        DELETE FOR FOTOCOPIE=0
        PACK
        do windo3

    case record=>6
        USE STORICO4
        INDEX ON DTOS(DATA) TO STORICO4
        set filter to matr=val(mat5)

```

```

REINDEX
DELETE ALL FOR FOTOCOPIE=0
PACK
go top
delete
pack
reindex
do windo3
ENDCASE
CLOSE DATABASES
procedure windo3
USE STORICO4
@12,0 to 12,79 double
@12,0 FILL TO 24,79 COLOR B/BG
SET COLOR TO W+/BG
define window parziale from 15,0 to 21,79
set filter to matr=val(mat5)
SCAN FOR MATR=VAL(MAT5)
BROWSE NOAPPEND NOEDIT NOMENU fields
matr/r,CLASSE/r,FOTOCOPIE/R,data/r,totale=FOTOCOPIE+TOTALE;
compress window parziale
RISPOSTA = " "
@ 23,28 SAY "SI VUOLE CONTINUARE? (S/N)";
GET RISPOSTA PICTURE "!";
VALID RISPOSTA $ "SN";
ERROR "DIGITARE S o N"
READ
DO CASE
CASE RISPOSTA ="S"
RETURN
CASE RISPOSTA ="N"
RETURN TO MASTER
ENDCASE
ENDSCAN
set filter to
SET CURSOR ON
RETURN

```

LISTATO SORGENTE DEL PROGRAMMA LIBRERIA

NEL PROGRAMMA LIBRERIA SONO INSERITE LE PROCEDURE RICHIAMABILI DAI PROGRAMMI GESTIONE E BARRE. TALI PROCEDURE, OVVERO SOTTOPROGRAMMI, CONTENGONO LE ISTRUZIONI RELATIVE ALLA GESTIONE DEI COMANDI SET, DEI COLORI, DEL SALVATAGGIO DEI DATI, DELLA IMPOSTAZIONE DELLE VARIABILI, DELLA GESTIONE DEI MESSAGGI DI ERRORE, DEL MENU' E DEL PROGRAMMA DI STAMPA E IL BACKUP DEI DATI.

```
PROCEDURE Rest_env
  IF TYPE( "gl_MainMenu" ) = "L"
    RETURN
  ENDIF
SET COLOR TO &c_standard.
SET SCOREBOARD &scor.
SET DELIMITERS &deli.
SET HELP &help.
SET ESCAPE &esca.
SET DELETED &delee.
SET HEADING &head.
SET SAFETY &safe.
SET EXACT &exac.
SET BELL &bell.
SET NEAR &near.
DO Colo_rese
  SET CLOCK &clock.
  SET STATUS &stat.
  SET TALK &talk.
RETURN

PROCEDURE Sav_data
IF NodShake( " ; Salvare i dati su disco? ", ;
  9, 26, 2, 29, .F. )
  IF !AddNew
    APPEND BLANK
    record_num = RECNO()
  ELSE
    record_num = RECNO()
  ENDIF
DO Repl_fld
ELSE
GO record_num
ENDIF
RETURN

PROCEDURE Set_env
  IF TYPE( "FILTERS_ON" ) = "L"
    filters_on = .F.
  ENDIF
  IF TYPE( "gl_MainMenu" ) = "L"
    RETURN
  ENDIF
PUBLIC talk
IF SET( "TALK" ) = "ON"
  SET TALK OFF
  talk = "ON"
ELSE
  talk = "OFF"
ENDIF
```

```

PUBLIC c_Save
c_save = SET( "ATTRIBUTES" )

PUBLIC c_standard, c_data, c_fields, c_popup, c_alert, c_list
PUBLIC c_red, c_blue, c_yellow, c_yelowhit, c_green, c_blink

IF ISCOLOR()
c_standard = "W/B,BG+/R,B"
  c_data   = "B/W,GR/BG,B"
  c_fields = "B/BG"
  c_popup  = "B/W,GR+/BG"
  c_alert  = "GR+/R,B/W,R/G"
  c_list   = "W+/G,GR+/B,GR+/GR"
  c_red    = "R/W"
  c_blue   = "B/W"
  c_yellow = "GR+/B"
  c_yelowhit = "GR+/W"
  c_green  = "G/W"
  c_blink  = "GR+*/B"
ELSE
STORE "W+/N" TO c_standard, c_data, c_popup, c_alert, c_list
  STORE "W" TO c_red, c_blue, c_yellow, c_yelowhit, c_green, c_fields
  c_blink = "W+*/N,N/W"
ENDIF
SET COLOR OF MESSAGES TO &c_blue.
SET COLOR TO &c_standard.

PUBLIC scor, deli, hellp, clock, esca, delee, head, stat, safe
PUBLIC exac, bell, near
scor = SET("SCOREBOARD")
deli = SET("DELIMITERS")
hellp = SET("HELP")
clock = SET("CLOCK")
esca = SET("ESCAPE")
delee = SET("DELETED")
head = SET("HEADING")
stat = SET("STATUS")
safe = SET("SAFETY")
exac = SET("EXACT")
bell = SET("BELL")
near = SET("NEAR")

SET SCOREBOARD off
SET DELIMITERS off
SET HELP off
SET CLOCK off
SET ESCAPE on
SET DELETED on
SET HEADING on
SET STATUS off
SET SAFETY off
SET TALK off
SET EXACT off
SET BELL off
SET NEAR off

PUBLIC erased, not_valid, rec_is_dup, filters_on, lookup_ok, choice
PUBLIC record_num, net_choice
PUBLIC target, look_dbf, matchchar, scanfield
STORE .F. TO erased, not_valid, rec_is_dup, filters_on

```

```

lookup_ok = .T.
STORE "" TO choice,subset
STORE 0 TO record_num, net_choice

RETURN

```

```

PROCEDURE Gen_Err
PARAMETERS pn_Error, pc_Message
DO Err_Box WITH pc_Message
gl_Error = .T.
ON ERROR
ON KEY LABEL F1
ON KEY LABEL F9
ON KEY LABEL F10
RETURN TO MASTER

```

```

IF TYPE( "gl_MainMenu" ) <> "L"
DO Rest_env.
ON ERROR
ON KEY LABEL F1
ON KEY LABEL F9
ON KEY LABEL F10
CLEAR ALL
CLOSE ALL
CLEAR
CANCEL
ENDIF
RETURN TO MASTER

```

```

PROCEDURE Show_msg
PARAMETERS u_message
MyIndent = _indent
MyMargin = _rmargin
_indent = 0
_rmargin = 40
_wrap = .T.
ACTIVATE WINDOW alert
@ 1,0
?? u_message
?
WAIT "Premere SPAZIO per continuare..."
_indent = MyIndent
_rmargin = MyMargin
DEACTIVATE WINDOW alert
RETURN

```

```

PROCEDURE Err_Box                                &&PROCEDURA GESTIONE ERRORI
PARAMETERS pc_msg

```

```

PRIVATE lc_anykey, lc_msg, lc_msglen, lc_win, ln_press, ln_width, ll_trap,;
ll_escape

```

```

lc_anykey = [Un tasto per continuare...]
ln_press = LEN( lc_anykey )
lc_win = WINDOW()
lc_msg = LTRIM( RTRIM( pc_msg ) )
ln_msglen = LEN( lc_msg )
ln_width = 0
ll_escape = SET("ESCAPE") = "ON"

```

```

SET ESCAPE OFF

IF ln_msglen <= ln_press
  ln_width = ln_press
ELSE
IF ln_msglen > 76
  lc_msg = LEFT( lc_msg, 76 )
  ln_msglen = 76
ENDIF
  ln_width = ln_msglen
ENDIF
DEFINE WINDOW _err_box FROM 9, ((76 - ln_width) + .5) / 2 ;
  TO 15, (ln_width + 83) / 2 DOUBLE
ln_width = ( ln_width + 2 )

ACTIVATE WINDOW _err_box
@ 1, ( ln_width - ln_msglen ) / 2 SAY lc_msg
@ 3, ( ln_width - ln_press ) / 2 SAY lc_anykey
SET CONSOLE OFF
WAIT
SET CONSOLE ON

RELEASE WINDOW _err_box
IF ISBLANK( lc_win )
  ACTIVATE SCREEN
ENDIF

IF ll_escape
  SET ESCAPE ON
ELSE
  SET ESCAPE OFF
ENDIF

RETURN

PROCEDURE Colo_rese          &&PROCEDURA COLORI MESSAGGI, TITOLI E BOX
PRIVATE old_color, c_messages, c_titles, c_box, c_info, c_fields

old_color = c_save

SET COLOR TO
SET COLOR TO &old_color.
CLEAR

old_color = STUFF(old_color, 1, AT("&",old_color)+2, "")

comma = AT(", ",old_color)
c_messages = LEFT(old_color, comma-1)
old_color = STUFF(old_color, 1, comma, "")

comma = AT(", ",old_color)
c_titles = LEFT(old_color, comma-1)
old_color = STUFF(old_color, 1, comma, "")
comma = AT(", ",old_color)
c_box = LEFT(old_color, comma-1)
old_color = STUFF(old_color, 1, comma, "")

comma = AT(", ",old_color)
c_info = LEFT(old_color, comma-1)
old_color = STUFF(old_color, 1, comma, "")

```

```
comma = AT(",",old_color)
c_fields = old_color
```

```
SET COLOR OF MESSAGES TO &c_messages.
SET COLOR OF TITLES TO &c_titles.
SET COLOR OF BOX TO &c_box.
SET COLOR OF INFORMATION TO &c_info.
SET COLOR OF FIELDS TO &c_fields.
RETURN
```

```
procedure warnbell
  private mwrap
  mwrap = _wrap
  _wrap = .f.
  set bell to 880,4
  ?? chr(7)
  set bell to 1400,4
  ?? chr(7)
  set bell to 880,4
  ?? chr(7)
  set bell to
  _wrap = mwrap
  return
```

```
PROCEDURE ALERT &&PROCEDURA DI ERRORE
  DEFINE WINDOW ALERT FROM 16,21 TO 23,58
  ACTIVATE WINDOW ALERT
  @2,3 SAY "CODICE INESISTENTE O INESATTO" COLOR W+/B*
  ?
  ?
  WAIT " RETURN PER CONTINUARE"
  DEACTIVATE WINDOW ALERT
  close databases
  RETURN to barre
```

```
procedure alert1 &&PROCEDURA DI ERRORE1
  define window alert1 from 16,21 to 23,58
  activate window alert1
  store " " to scelta
  @2,3 say "si vuole continuare? (S/N)" get scelta picture "!X"
  read
  do case
  case scelta ="S"
  DEACTIVATE WINDOW ALERT1
  CLEAR
  RETURN
  case scelta ="N"
  use storico
  DELETE
  PACK
  deactivate window alert1
  close databases
  DO barre
  endcase
  return
```

```
PROCEDURE STAMPA &&MENU' E PROGRAMMA DI STAMPA
  SET CURSOR ON
  clear
  set color to &c_STANDARD
  @0,0 say "É"+replicate("Í",19)+»"
```

```

riga=1
do while len(field(riga))>0
@riga,0 say ""+str(riga,2,0)+". "+field(riga)+space(14-len(field(riga)))+""
riga=riga+1
enddo
@riga,0 say "È"+replicate("Í",19)+"¼"
num_campi=riga-1

condiz=""
ciclo=.t.
@2,23 say "SELEZIONE CONDIZIONE"
@3,23 SAY "É"+REPLICATE("Í",52)+"»"
@7,23 SAY "È"+REPLICATE("Í",52)+"¼"
@8,23 SAY "OPERATORI: =,<,>,>=,<=,<>,$"
@13,0 SAY "CONDIZIONE"
@14,0 SAY REPLICATE("Í",80)
@22,0 SAY REPLICATE("Í",80)

DO WHILE CICLO
OPERATORE=" "
N_CAMPO=0
@4,23 SAY ""+SPACE(50)+" ""
@5,23 SAY ""+SPACE(50)+" ""
@6,23 SAY ""+SPACE(50)+" ""
@4,25 SAY "N. DEL CAMPO:" GET N_CAMPO PICT "99"
@5,25 SAY "OPERATORE...:" GET OPERATORE PICT "!!"
READ
@23,1 SAY SPACE(78)
OPERATORE=TRIM(OPERATORE)

DO CASE
CASE N_CAMPO>0 .AND. N_CAMPO<=NUM_CAMPI .AND. OPERATORE $ "=<><=>=$"
IF LEN (TRIM(CONDIZ))>0
CONDIZ=CONDIZ+" .AND. "
ENDIF

NOME_CAMPO=FIELD(N_CAMPO)
TIPO_CAMPO=TYPE("&NOME_CAMPO")
DO CASE
CASE TIPO_CAMPO="C"
VALORE_C=SPACE(LEN(&NOME_CAMPO))
@6,25 SAY "VALORE.....:" GET VALORE_C
READ

VALORE_C=UPPER(TRIM(VALORE_C))
CONDIZ=CONDIZ+"UPPER(TRIM(&NOME_CAMPO))"+OPERATORE+CHR(34)+"&VALORE_C"+CHR(34)

CASE TIPO_CAMPO="N"
VALORE_N=0
@6,25 SAY "VALORE.....:" GET VALORE_N
READ
CONDIZ=CONDIZ+"&NOME_CAMPO"+OPERATORE+STR(VALORE_N)
case tipo_campo="D"
VALORE_D=CTOD(" / / ")
@6,25 SAY "VALORE.....:"GET VALORE_D
READ
CONDIZ=CONDIZ+"&NOME_CAMPO"+OPERATORE+"CTOD("+CHR(34)+DTC(VALORE_D)+CHR(34)+")"

ENDCASE
@15,0 SAY CONDIZ
CASE N_CAMPO=0

```

```

CICLO= .F.
OTHERWISE
@23,1 SAY "ERRORE DI SELEZIONE. RIPETERE"
ENDCASE
ENDDO
IF LEN(TRIM(CONDIZ))>0
index on matr to stampa
clear
DO CASE
CASE RIS=4
REPORT FORM RUBRICA1 NOEJECT FOR &CONDIZ TO PRINT
CASE RIS=1
REPORT FORM RUBRICA NOEJECT FOR &CONDIZ TO PRINT
CASE RIS=2
REPORT FORM RUBRICA2 NOEJECT FOR &CONDIZ TO PRINT
CASE RIS=3
REPORT FORM RUBRICA3 NOEJECT FOR &CONDIZ TO PRINT
ENDCASE
ENDIF
CLOSE DATABASES
RETURN

```

```

PROCEDURE RICHIEST
CLEAR
SET COLOR TO &C_STANDARD
PUBLIC RIS
RIS=" "
define popup sta_menu from 7,20 to 20,50
define bar 1 of sta_menu prompt "===== MENU' DI STAMPA =====" skip
define bar 3 of sta_menu prompt " STAMPA ARCHIVIO ALUNNI "
DEFINE BAR 5 OF STA_MENU PROMPT " STAMPA ARCHIVIO DOCENTI"
define bar 7 of sta_menu prompt " STAMPA ARCHIVIO UFFICI "
DEFINE BAR 9 OF STA_MENU PROMPT " STAMPA ARCHIVIO CLASSI "
DEFINE BAR 11 OF STA_MENU PROMPT " USCITA"
ON SELECTION POPUP STA_MENU DEACTIVATE POPUP
ACTIVATE POPUP STA_MENU
DO CASE
CASE BAR()=3
USE STORICO2
RIS=1
DO STAMPA
CASE BAR()=5
USE storico
RIS=2
DO STAMPA
CASE BAR()=7
USE STORICO3
RIS=3
DO STAMPA
CASE BAR()=9
USE STORICO4
RIS=4
DO STAMPA
CASE BAR()=11
RETURN TO MASTER
ENDCASE
RETURN

```

```

PROCEDURE Salvatag
CLOSE DATABASES
PUBLIC mpath

```

```
&&PROCEDURA SALVATAGGIO DATI
```

```

STORE "" TO choice, answer
DO M_popdef
SET COLOR TO &c_normal.
CLEAR
SET COLOR TO &c_pop.
SET COLOR OF HIGHLIGHT TO &lt_blue.
ACTIVATE POPUP backmenu
RETURN TO Gestione

```

```

PROCEDURE Bbackup  &&MENU' DI BACKUP
ACTIVATE WINDOW backup
@ 0,0 SAY "-----SALVATAGGIO DATI (BACKUP)-----"
@ 1,2 SAY "Inserire un dischetto formattato nel drive A: e"
WAIT "premere un tasto qualsiasi per iniziare il salvataggio"
RUN msBACKUP &mpath.*.DB? A:
RUN msBACKUP &mpath.*.MDX A: /A
?? CHR(7)
CLEAR
? "***** SALVATAGGIO CONCLUSO *****"
WAIT
DEACTIVATE WINDOW backup
SET COLOR TO &c_normal.
CLEAR
SET COLOR TO &c_pop.
RETURN

```

```

PROCEDURE Back
DO CASE
CASE BAR() = 2
DO Sure
IF choice $ "SY"
DO Bbackup
ENDIF
CASE BAR() = 3
DO Sure
IF choice $ "SY"
DO Rrestore
ENDIF
CASE BAR() = 4
RETURN TO Gestione
ENDCASE
RETURN

```

```

PROCEDURE M_popdef          && BACKUP DATI
DEFINE POPUP backmenu FROM 7,20 to 12,56 ;
MESSAGE "Premere l'iniziale dell'opzione del menu, o selezionare l'opzione e battere "+CHR(17)+CHR(217)
DEFINE BAR 1 OF backmenu ;
PROMPT "====SALVATAGGIO/RIPRISTINO DATI====" SKIP
DEFINE BAR 2 OF backmenu ;
PROMPT " Salvataggio dati su dischetti "
DEFINE BAR 3 OF backmenu ;
PROMPT " Ripristino dati da dischetti "
DEFINE BAR 4 OF backmenu ;
PROMPT " Torna al menu principale "
ON SELECTION POPUP backmenu DO Back
DEFINE WINDOW backup FROM 14,15 TO 20,70 COLOR &c_pop.
RETURN

```

```

PROCEDURE Rrestore          &&MENU' RIPRISTINO DATI
ACTIVATE WINDOW backup
@ 0,0 SAY "-----RIPRISTINO DATI (RESTORE)-----"

```

```

@ 1,2 SAY "Inserire il primo dischetto di backup nel drive A: e"
WAIT "premere un tasto qualsiasi per iniziare il ripristino"
! RESTORE A: &mpath.*.DB?
! RESTORE A: &mpath.*.MDX
?? CHR(7)
CLEAR
? "***** RIPRISTINO CONCLUSO *****"
WAIT
DEACTIVATE WINDOW backup
SET COLOR TO &c_normal.
CLEAR
SET COLOR TO &c_pop.
RETURN

PROCEDURE Sure
CLEAR
choice = " "
ACTIVATE WINDOW backup
@ 0,0 SAY "-----AVVERTENZA-----"
@ 1,0 SAY "Si desidera veramente salvare o ripristinare i dati?"
@ 2,0 SAY "Premere S per continuare o N per rinunciare"
@ 3,0 SAY "SCELTA:" GET choice PICTURE "Y"
?? CHR(7)
READ
IF choice = "N"
SET COLOR TO &c_pop.
DEACTIVATE WINDOW backup
RETURN
ENDIF
IF UPPER(choice) $ "SY"
?? CHR(7)
mpath = "C:\dBASE\" + SPACE(10)
CLEAR
@ 0,0 SAY "-----PERCORSO DEI FILE DI ESEMPIO-----"
@ 1,0 SAY "Immettere il drive e il percorso DOS ove sono"
@ 2,0 SAY "registrati i file"
@ 3,0 GET mpath VALID "" <> TRIM(mpath) ;
MESSAGE "Immettere la lettera del drive e il percorso DOS"
READ
mpath = TRIM(mpath)
ENDIF
DEACTIVATE WINDOW backup
RETURN

```

SOMMARIO

INTRODUZIONE GENERALE.....	2
<i>TRACCIATI RECORD</i> :	3
dBASE IV PER DOS.....	3
<i>CHE COSA E' IL PROGRAMMA IN dBASE</i> :	3
<i>CONVENZIONI SUI PROGRAMMI</i> :	4
<i>LE VARIABILI DI MEMORIA</i> :	5
<i>ASSEGNAZIONE DEL NOME ALLE VARIABILI DI MEMORIA</i> :	5
<i>INIZIALIZZAZIONE DELLE VARIABILI DI MEMORIA</i> :	5
<i>INIZIALIZZAZIONE DEGLI ARRAY DI VARIABILI DI MEMORIA</i> :	6
<i>USO DI VARIABILI DI MEMORIA E DI ARRAY NEI PROGRAMMI</i> :	7
<i>VARIABILI PUBBLICHE E PRIVATE</i> :	7
<i>USO DEI FILE DI MEMORIA</i> :	8
<i>ARCHITETTURA DI UN PROGRAMMA</i> :	8
<i>LA PROGRAMMAZIONE STRUTTURATA</i> :	8
<i>PROCEDURE E FILE DI PROCEDURA</i> :	9
<i>STRUTTURE DI PROGRAMMAZIONE</i> :	9
<i>PROCESSI SEQUENZIALI</i> :	10
<i>STRUTTURE DI SELEZIONE</i> :	10
<i>STRUTTURE DI ITERAZIONE</i> :	11
<i>IMPOSTAZIONE DELL'AMBIENTE</i> :	11
<i>LA DEFINIZIONE DI UN AMBIENTE OPERATIVO</i> :	11
<i>INSERIMENTO DI DATI MEDIANTE SCHEDE</i> :	13
<i>DEFINIZIONE DI UN MENU POPUP</i> :	13
<i>I FILE INDICE</i> :	14
<i>SPIEGAZIONE DEL PROGRAMMA</i> :	14
IL CODICE A BARRE :	16
<i>STORIA DEL CODICE A BARRE</i> :	16
<i>CARATTERISTICHE DEL CODICE A BARRE</i> :	16
<i>COSA SONO I CODICI A BARRE</i> :	17
<i>LA CODIFICA</i> :	17
<i>RISOLUZIONE</i> :	18
<i>RAPPORTO TRA LO SPESSORE DEGLI ELEMENTI STRETTI E QUELLI LARGHI</i> :	18
<i>SEQUENZA DI BARRE E SPAZI</i> :	19
<i>NUMERO DI CARATTERI RAPPRESENTABILI</i> :	19
<i>TIPI DI CARATTERI RAPPRESENTABILI</i> :	19
<i>SEQUENZA DELLE BARRE DI GUARDIA</i> :	19
<i>TOLLERANZA</i> :	19
<i>CHECK DIGIT</i> :	19
<i>LUNGHEZZA DEL SIMBOLO</i> :	19
<i>MONO-BIDIMENSIONALITA'</i> :	19
<i>CODICE CONTINUO-DISCRETO</i>	20
<i>CODICE SELF CHECKING</i>	20
IL CODICE 39	20
<i>TABELLA DI CODIFICA</i>	21
<i>CALCOLO DEL CHECK DIGIT</i> :	22
PROGRAMMA IN dBASE IV.....	23
<i>LISTATO SORGENTE DEL PROGRAMMA GESTIONE</i>	23
<i>LISTATO SORGENTE DEL PROGRAMMA BARRE</i>	24
<i>LISTATO SORGENTE DEL PROGRAMMA LIBRERIA</i>	32