di Francesco Petroni

DB III e DB IV Tecniche avanzate: Macro e Funzioni di sistema

In attesa della prossima ondata di articoli sul dBASE IV e sulle novità in termini di funzioni in più apportate rispetto al «vecchio» dBASE III, consolidiamo alcune conoscenze di base dell'ambiente dBASE, che valgono non solo in ambedue i prodotti ma anche in tutti gli altri che utilizzano il «linguaggio» dBASE

Nel dBASE III e nel dBASE IV si possono distinguere due livelli di comandi, quelli fondamentali, richiamabili dall'ambiente dotprompt (il famoso puntino del dBASE III) ed eseguibili direttamente, e quelli sovrastrutturali, che lavorano con un'interfaccia utente più «amichevole», barra di menu e menu a tendina, ed impediscono, specie all'utente alle prime armi, un contatto diretto e brutale con il «puntino».

Esempio tipico di queste funzioni sovrastrutturali è l'ASSIST che tramite la tecnica a menu consente di costruire, pezzo per pezzo, un comando anche complesso. Tale comando, alla fine della costruzione, è esattamente quello che si può scrivere dal puntino.
Per la cronaca nel dBASE IV sono state potenziate sia le funzioni dirette (quelle chiamabili dal puntino), sia quelle sovrastrutturali, che operano in ambienti operativi amichevoli e che generano varie tipologie di entità (i Report, le Maschere, le Query, le Label, ecc.).

Tale sovrastrutture faciliteranno il compito agli utenti iniziali e velocizzeranno il lavoro, specie quello iniziale, del tecnico che sviluppa applicazioni. Non esauriscono però le modalità di lavoro con il dBASE, che dispone come noto di numerose altre funzioni, non accessibili da comandi sovrastrutturali e non facilmente maneggiabili dall'inesperto.

In questo articolo parleremo di due categorie di funzionalità, e cioè del concetto di Macro, legato al carattere & (detta «Ampersend» in inglese o «E commerciale» in italiano) e delle numerose funzioni di sistema che il linguaggio dBASE attiva e mette a disposizione sia per un uso in comandi diretti, sia e soprattutto, per un uso in programmazione.

L'articolo sarà al solito molto pratico, in modo da consentire a tutti di mettere immediatamente a frutto le teorie esposte e gli esempi mostrati.

Il concetto di Macro

Se assegnamo ad una variabile una stringa di caratteri che hanno anche il significato di un comando o di parte di un comando sintatticamente corretto e riconosciuto dal dBASE potremo, tramite il comando &, «mandare in esecuzione il contenuto della variabile». Ad esempio:

```
A="3*3"          assegniamo ad A la stringa 3*2
? A              visualizziamo il contenuto di A
  3*2
? &A             visualizziamo l'esecuzione del
                 contenuto di A
  3
```

Tale logica vale anche al contrario, nel senso che è possibile assegnare un valore non solo ad una variabile, ma anche al contenuto di una variabile, che deve essere una seconda variabile. Ad esempio:

```
PIPPO="PLUTO"
&PIPPO="KATIA"
? PLUTO
  KATIA
```

Nella riga &PIPPO="KATIA", viene assegnato un valore ad una variabile che non appare esplicitamente, in quanto è contenuta nella variabile PIPPO.

L'esperimento può essere spinto su più livelli, trattando ad esempio il contenuto del contenuto di una variabile, come in un gioco di scatole cinesi.

Inoltre il nome di una variabile, essendo una stringa, può essere manipolato all'occorrenza tramite funzioni di stringa e questa possibilità allarga ulteriormente i campi applicativi.

Es.

```
N = 12
```

```
VAR = "VR"+STR(N,2)
&VAR = "TOPOLINO"
? VAR,VR12
  VR12 TOPOLINO
```

In questo caso il contenuto di VAR è VR12, al contenuto di VAR è assegnato il valore TOPOLINO, e quindi VR12 è TOPOLINO. Questi concetti, apparentemente banali, trovano campi di applicazioni enormi, specie se collegati, come vedremo dopo, alle altre sofisticate funzioni del dBASE.

Le funzioni di sistema

Il linguaggio dBASE dispone di numerose funzioni, che forniscono dei valori riguardanti vari aspetti dello stato del sistema, e che sono utilizzabili sia in comandi diretti sia, in maniera molto più produttiva, in una situazione di programmazione.

Sono riconoscibili in quanto utilizzano una particolare simbologia costituita da due parentesi una aperta e una chiusa «()». Nel caso la funzione richieda parametri numerici questi vanno immessi tra le due parentesi.

Le funzioni di sistema possono essere raggruppate in più categorie. Ad esempio funzioni generali d'ambiente:

? DATE()	data del sistema
? TIME()	ora del sistema
? DISKSPACE()	spazio disponibile su disco

Funzioni che riguardano lo stato dell'archivio in uso:

? DBF()	nome
? NDX(1)	nome del primo indice aperto
? RECCOUNT()	numero di record
? RECSIZE()	dimensione della struttura

Funzioni che permettono di individuare la struttura:

? FIELD(1)	nome del primo campo

Funzioni legate al record in uso dell'archivio in uso:

? RECNO()	numero del record
? DELETED()	è cancellato SÌ/NO
? EOF()	ci si trova in condizioni di fine file S/N
? BOF()	ci si trova in condizioni di inizio file S/N

Nel caso di funzioni che riconoscono parametri numerici può succedere che il risultato sia nullo. In ogni caso non si verificano situazioni di errore. Ad esempio:

```
? FIELD(100)
```

Nel caso in cui esista il campo numero 100, ne viene fornito il nome. Nel caso invece in cui non esista viene restituita una stringa nulla. Si può quindi facilmente sia testare se il campo 100 esiste e, se esiste, leggerne il nome. È infine evidente che le varie funzionia di sistema possono collaborare con le altre funzioni «normali», e quindi allargare l'insieme delle informazioni reperibili dall'ambiente. Ad esempio è possibile leggere integralmente i dati della struttura.

C01	=FIELD(1)	nome del primo campo
T01	=TYPE(&C01)	tipo del primo campo (C=carattere, N=numerico, D=data, L=logico, ecc.)
L01	=LEN(&C01)	lunghezza del primo campo (se tipo=C)

I nostri esercizi

Passiamo alla pratica. Come al solito non si tratta di programmi da copiare ed usare, ma di esercizi che trattano problematiche sulle quali vogliamo stimolare il vostro interesse. Si tratta tralaltro di programma molto corti e, anche copiandoli semplicemente, occorrono pochi minuti.

Utilizzeremo due semplici archivi la cui struttura è leggibile nelle illustrazionia e nel testo. Il primo si chiama CLIENTI ed ha due indici CLIENT1 e CLIENT2, rispettivamente costruiti sui campi CODR e DITT. Il secondo PROVINCE con un indice PROVINCE realizzato sul campo SIGLA.

Per poter eseguire correttamente gli esercizi è necessario costruirli e riempirli con dati a piacere.

Cominceremo con la trattazione delle MACRO, vero e proprio mondo da esplorare, con le quali realizziamo uno pseudo-vettore.

Proseguiremo con il testare le varie funzioni di sistema che permettono di leggere lo «Status» dell'applicazione e di

Figura 1
Programma di simulazione di vettore - Utilizzo.
In attesa della diffusione del dBASE IV, che dispone di entità specifiche vettori e matrici, si può simulare un vettore di dimensioni limitate anche con il dBASE III, per mezzo di una serie di variabili il cui nome viene composto tramite una somma di stringhe in cui entra un valore numerico crescente (VAR1, VAR2,... oppure VARO1, VARO2,...).

```
* MC1 simulazione vettore - prima parte
*                          utilizzo
set talk off
v1="Roma          "
v2="Milano        "
v3="Torino        "
v4="Napoli        "
v5="Genova        "
v6="Caltannissetta"
clear
do while .t.
  v0=0
  @ 20,10 say "Immetti Numero della Citta': :"
  @ 20,38 get v0 picture "9" range 0,6
  read
  if v0=0
    exit
  endif
  v0="v"+str(v0,1)
  @ 20,50 say &v0
enddo
```

```
* MC2 simulazione vettore - seconda parte
*                          caricamento
*
set talk off
clear
cm=1
do while .t.
 ct="          "
 v00="v"+str(cm+10,2)
 @ 10,10 say "Citta' numero "+str(cm,3)
 @ 10,30 get ct pict "@!"
 read
 if len(trim(ct))=0
  exit
 endif
 &v00=ct
 cm=cm+1
enddo
release cm,v00,ct
display memory
```

Figura 2
Programma di simulazione di vettore - Caricamento.
Il vettore utilizzato nell'esercizio precedente può essere caricato via funzioni di input (GET e READ), in cui si definisce non il valore di una variabile, ma il valore del contenuto, di una stessa variabile V00, che assume valori successivi V11,V12,V13. In coda visualizzano l'elenco delle variabili caricate, depurata dalle tre variabili necessarie alla costruzione dello pseudo vettore.

leggere tutte le caratteristiche della struttura dell'archivio in uso, qualsiasi esso sia.

Poi proveremo ad attribuire ad una variabile delle stringhe equivalenti a formule di calcolo. Tali formule, in cui possono entrare numeri, segni matematici e nomi di variabili o di campi noti, vengono poi mandate in esecuzione.

Alla fine faremo un esercizio di parametrizzazione di un programma, proveremo cioè a specificare in variabili le varie caratteristiche di un programma di gestione archivio. Il comportamento delle varie istruzioni dipende quindi dal contenuto delle variabili.

Occorre ricordare che il dBASE consente di gestire archivi di variabili (quelli con desinenza *·MEM), in cui è possibile immagazzinare e leggere, anche in maniera selettiva, informazioni non strutturate, come quelle che vanno negli archivi *·DBF, ma che possono essere altrettanto utili.

Il numero massimo di variabili definibili è di 256 nel dBASE III e 2048 nel dBASE IV.

Simulazione di un vettore

Figure 1 e 2

Il primo esercizio fa riferimento al programma di figura 1 e consiste nell'utilizzo di uno pseudo-vettore. Sono state definite una serie di variabili con nome in sequenza. V1, V2, ecc. Ad ogni numero corrisponde una precisa variabile, così come, in un vettore, ad un numero corrisponde un singolo elemento del vettore stesso.

Nel nostro esercizio il vettore è caricato direttamente via istruzioni di assegnazione. Poteva essere anche caricato leggendo i valori progressivi in un archivio. Il caso invece di caricamento con dati provenienti via input è analizzato nella successiva figura 2.

La routine di scodifica è costituita da un ciclo «eterno»:

```
DO WHILE .T.
...
ENDDO
```

dal quale si esce solo quando si digita il valore 0.

I valori numerici accettati vanno dallo 0 (che provoca l'uscita) al 6. Il controllo dell'input avviene tramite le specifiche PICTURE «9», che accetta solo caratteri numerici, e la specifica RANGE 0,6 (dal significato ovvio, che si possono aggiungere all'istruzione @ r,c GET.

Da notare la modalità con la quale, dato un numero V0, viene ricostruita la variabile "V"+STR(V0,1), ad esempio V1, e con la quale viene scodificata la relativa variabile SAY &V0. Nel caso in cui siano necessarie più di 10 variabili si può costruire un vettore che va da V10 a V99, oppure da V100 a V999.

La funzione di costruzione della variabile intermedia contiene la somma di due stringhe, la «V» e un valore numerico che viene tradotto in stringa mediante una funzione STR(V0,n) in cui va indicato «n» il numero di caratteri voluto. Ed è bene che questo numero rimanga fisso, per evitare che in questa costruzione debbano entrare anche funzioni di IF, che testino la lunghezza del numero.

Il secondo esercizio, quello di figura 2, consiste nel caricare, via funzione di input da tastiera (@ <riga>,<colonna> GET <var>) il vettore.

Le variabili progressive V11, V12, ecc. sono costruite con il metodo già noto. La variabile V00 assume via via tali valori e la stringa in input viene caricata nel contenuto di V00 (con l'istruzione &V00=CT).

Le variabili di servizio, che quindi non hanno nulla a che vedere con il vettore, sono CM, che è il contatore numerico, CT che è la variabile dove viene parcheggiata la stringa immessa, prima di essere trasferita nel vettore, V00 la variabile di appoggio.

Notare come al valore 1 del contatore numerico venga fatto corrispondere la variabile V11, e così via. In tal modo come già detto si possono caricare una novantina di valori.

Lettura delle funzioni di sistema

Il primo programma consiste nella semplice lettura di una serie di variabili di sistema (listato e output in figura 3) che indicano archivio e indice aperti nelle varie aree di lavoro, poi indicano le varie funzioni relative all'archivio in uso,

```
* MC3 - funzioni di sistema
*       riconoscimento dello STATUS
*
  clear
* apertura archivi
  select 1
  use clienti index clien1,clien2
  select 2
  use province index province
  select 1
* funzioni di sistema sullo status
  select 1
  ? "Funzioni di Sistema sullo Status"
  ? "  Primo Archivio    ",dbf(),ndx(1),ndx(2),ndx(3)
  select 2
  ? "  Secondo Archivio  ",dbf(),ndx(1),ndx(2),ndx(3)
  select 3
  ? "  Terzo Archivio    ",dbf(),ndx(1),ndx(2),ndx(3)
* informazioni sul FILE
  ?
  select 1
  ? "Funzioni di Sistama sull'Archivio"
  ? "  Lunghezza Record ",recsize()
  ? "  Numero Record    ",reccount()
* informazioni sul RECORD
  ?
  go 100
  ? "Funzioni di Sistema sul generico record"
  ? "  Numero Record    ",recno()
  ? "  Flag Cancellaz.  ",deleted()
  ? "  Flag Inizio File ",bof()
  ? "  Flag Fine File   ",eof()
clear all
```

```
Funzioni di Sistema sullo Status
  Primo Archivio     C:clienti.dbf C:clien1.ndx C:clien2.ndx
  Secondo Archivio   C:province.dbf C:province.ndx
  Terzo Archivio

Funzioni di Sistama sull'Archivio
  Lunghezza Record          107
  Numero Record             257

Funzioni di Sistema sul generico record
  Numero Record             100
  Flag Cancellaz.   .F.
  Flag Inizio File  .F.
  Flag Fine File    .F.
```

Figura 3 - Funzioni di riconoscimento dello stato.
Nell'ambiente interattivo del dBASE esiste la funzione DISPLAY STATUS che fornisce la situazione relativa ai vari archivi aperti e ai vari indici aperti su ognuno di essi. In un programma è possibile ottenere le stesse informazioni leggendo le numerose funzioni di sistema che il dBASE attiva via via. In coda vediamo l'output del programma dal quale si nota che tali funzioni non generano mai errori, anche nel caso che non vi siano archivi aperti.

```
* MC4 - funzioni di sistema
*       riconoscimento struttura
*
set talk off
clear
nf=space(8)
*
fl=1
do while fl=1
  @ 22,0  say "Immetti nome del File "
  @ 22,24 get nf picture "@!"
  read
  nfc=nf+".dbf"
  if file(nfc)
    fl=0
  else
    @ 20,34 say "Non Esiste"
  endif
enddo
*
use &nf
? "Nome del File in Uso "+dbf()
cn=1
do while .t.
  fi=field(cn)
  if len(fi)=0
    exit
  endif
  tp=type("&fi")
  ln=0
  if tp$"C"
   ln=len(&fi)
  endif
  ? cn,fi,tp,ln
  cn=cn+1
enddo
use
```

```
Immetti nome del File   CLIENTI

Nome del File in Uso C:CLIENTI.dbf
          1 CODC C          4
          2 RAGR C          1
          3 DITT C         14
          4 DTRG D          0
          5 INDI C         24
          6 NCAP C          5
          7 CITT C         14
          8 PROV C          2
          9 VTOT N          0
         10 SCNT N          0
         11 SLDO N          0
         12 LOGI L          0
         13 MEMO M          0
```

Figura 4 - Funzioni di riconoscimento della struttura. In teoria, e l'esercizio tende a dimostrarlo, utilizzando le funzioni che permettono ad un programma di leggere le caratteristiche della struttura (nomi dei campi, tipo e lunghezza) è possibile realizzare un programma di gestione che valga per qualsiasi archivio.

nonché le funzioni attive che ciascun record si... porta dietro.

Nulla di complicato.

È comodo, in previsione di un eventuale riconoscimento automatico da parte del programma dello Status del sistema, che tali variabili non generino mai errori, anche quando, ad esempio, l'archivio in uso non ci sia. In tal caso la funzione di sistema restituisce, fortunatamente, una stringa vuota facilmente intercettabile.

L'esercizio successivo consiste in un programmino che legge automaticamente la struttura dell'archivio in uso, qualsiasi esso sia (listato e output di prova in figura 4).

Il programma incomincia con un ciclo DO WHILE... ENDDO, dal quale si esce solo quando il nome immesso corrisponde ad un file effettivamente presente sul disco (il test logico, che quindi fornisce una variabile logica .T. o .F. (vero o falso) lo esegue la variabile di sistema FILE(«nomefile»).

Se il test è negativo il flag rimane uguale a 1, viene visualizzato il messaggio «Non Esiste» e non si esce dal ciclo, se positivo si va in sequenza.

Poi c'è un ciclo in cui viene utilizzato un contatore, che indica il numero del campo. Stavolta si esce dal ciclo quando si intercetta un nome di un campo che non c'è, quando cioè i campi sono finiti.

In ogni giro del ciclo viene indicato il nome del campo, il suo tipo (Carattere, Logico, Memo, Numerico e Data) e nel caso di appartenenza al primo tipo, la sua lunghezza. Solo quando è di tipo carattere perché Logico, Data e Memo hanno lunghezza fissa (1, 8 e 10 rispettivamente), mentre i campi numerici non... dichiarano altrettanto facilmente la loro lunghezza.

Pensate ad una utilizzazione in un programma di gestione archivio che funziona con qualsiasi archivio in quanto ne interpreta direttamente la struttura.

Operazioni in variabili

Questo terzo esercizio fa riferimento alla figura 5 che contiene un programma e il relativo output, che visualizza formule immesse via input e risultati.

In tale programma vengono definite tre variabili numeriche e ne viene chiesta, via funzione di input (in questo caso ACCEPT TO <var>), una quarta. In questa variabile va inserita una formula matematica, sintatticamente e... matematicamente corretta, in cui possono quindi essere immessi, oltre alle variabili numeriche definite prima, anche nomi di campi numerici, operatori matematici, numeri, e, volendo, anche funzioni riconosciute dal dBASE III.

Al di là della semplicità dell'esercizio, il campo di utilizzo di tale metodo è enorme. In una procedura, in cui occorre eseguire dei calcoli con valori numerici presenti negli archivi, le varie formule possono essere attribuite, sotto forma di stringa, a variabili memorizzate in un file di tipo *.MEM, modificabile «al di fuori» del programma. Oppure possono essere realizzate sottoprocedure di calcolo estemporaneo, o di simulazione, in cui l'utente «inventa» lì per lì l'algoritmo di calcolo cui sottoporre i dati dei suoi archivi. In tal caso l'utente deve conoscere i nomi delle variabili e eventualmente i nomi dei campi.

Essendo tralaltro programmi di solo calcolo non esiste nessun pericolo di perdita di dati in caso di operazioni sbagliate. Anche il caso di errore nella

```
* MC5 - variabili con formule
*       riconoscimento struttura
*
set talk off
clear
a=3
b=5
c=6
? " A ",a
? " B ",b
? " C ",c
?
do while .t.
  accept "Immetti la Formula  " to d
  if len(d)=0
    exit
  endif
  ? &d
enddo
```

```
 A           3
 B           5
 C           6

Immetti la Formula  a+10
        13
Immetti la Formula  a*b
                15
Immetti la Formula  a/b+c+200
          206.60
Immetti la Formula  (a*b)^c
                              11390625.00
Immetti la Formula  int(a*b*c/3)
                                      30
Immetti la Formula
```

Figura 5 - Programma di operazioni tra variabili. Tramite le Macro è possibile assegnare a delle variabili, oppure inserire in un archivio, stringhe il cui significato è equivalente ad una espressione matematica, che può essere calcolata come Macro. In tal modo ad esempio in un programma che esegue dei calcoli tutte le formule possono essere messe in un archivio di variabili, modificabile ben più facilmente del programma stesso.

formula digitata dall'utente può essere previsto e rimediato con una piccola routine di gestione degli errori che può inviare messaggi del tipo:

MANCA UNA PARENTESI
VARIABILE NON TROVATA
DIVISIONE PER ZERO

e far ridigitare la variabile.

Parametrizzazione

L'ultimo esercizio lavora sull'archivio PROVINCE, che ha la seguente struttura:

SIGLA	carattere da 2
CITTÀ	carattere da 15
CAP	carattere da 5
PRTEL	carattere da 4

Ci si propone di definire 4 variabili, in cui sono accorpate tutte le informazioni necessarie per costruire una miniapplicazione di gestione di una maschera per l'acquisizione dei dati.

Tali variabili sono a11,a12... e contengono le seguenti informazioni:
due caratteri che indicano la picture del campo;
due gruppi di due caratteri che indicano le coordinate X1,Y video del messaggio;
due caratteri che indicano la coordinata X2 della Get del campo;
due caratteri che indicano la lunghezza del messaggio;
più caratteri con il messaggio;
il nome del campo. Altre variabili contengono le Picture usate nella maschera. A queste fanno riferimento i primi due caratteri delle variabili a11,a12...

Nella variabile Cnm viene indicato il numero dei campi, nel nostro caso 4, che rappresenta il limite entro il quale cicla Cnc, che è il contatore dei campi (Cnc quindi va da 1 a Cnm).

Con tale ciclo vengono realizzate tre routine. La prima visualizza la maschera vuota (@ RG, C1 SAY <messaggio>), la seconda la riempie con i valori presenti nei campi (@ RG,C2 SAY <campo>) e la terza permette la editazione dei singoli campi.

Poiché quello che ci interessa è l'aspetto parametrizzazione abbiamo semplicemente collegato le nostre routine allo scorrimento dell'archivio, realizzato, al solito con un ciclo:

```
DO WHILE .NOT. EOF()
....
....
SKIP
ENDDO
```

Supponiamo infatti che chi sperimenta tecniche avanzate sappia da solo costruire un programma di gestione record cui applicare queste teorie.

È evidente inoltre che in caso di archivio con più campi, occorre variare Cnm (uguale al numero dei campi), e inserire altre variabili Axx ed, eventualmente, altre variabili Px, quelle con le picture.

Conclusioni

Se il dBASE (ripetiamo che non ha senso specificare DBIII o DBIV o Clipper o Foxbase o Quicksilver) è diventato il linguaggio più diffuso per chi gestisce archivi su PC lo deve anche a questa sua versatilità, che consente vari tipi di utilizzo, da quello più elementare, via strutture di assistenza (comando ASSIST e assimilabili), a quello più «puro», dal comando puntino.

L'appassionato o il tecnico, alla eterna ricerca non tanto di comandi nuovi (già li conoscono tutti) quanto di metodologie di utilizzo, trova nelle Macro un vero e proprio mondo da esplorare.

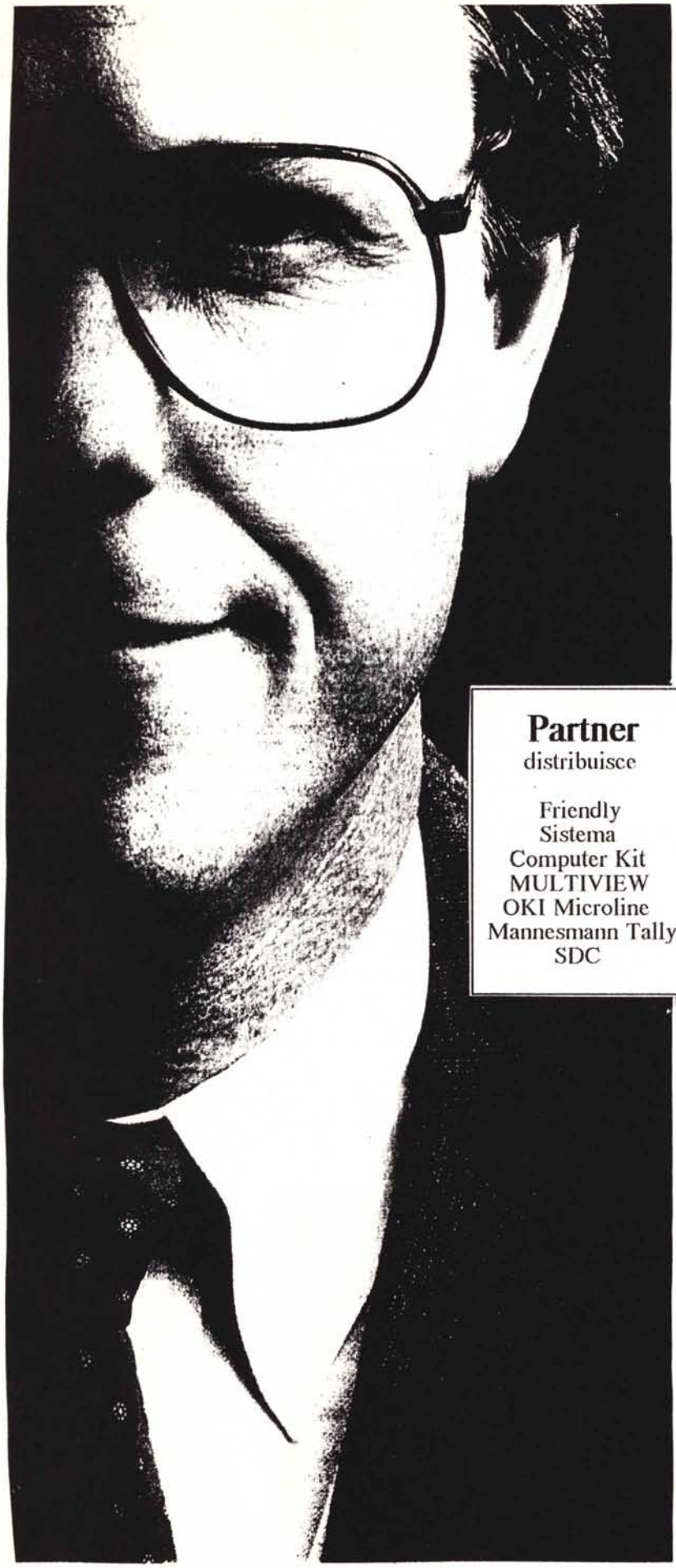
Si può affermare che in dBASE si può parametrizzare teoricamente tutto, al punto di poter costruire una procedura che funziona con qualsiasi archivio.

In realtà occorre definire un limite entro il quale la parametrizzazione diventa un vantaggio, in termini sia di metodo per la soluzione di un problema altrimenti non risolubile sia in termini di alleggerimento della programmazione e della manutenzione. Oltre tale limite si entra nel campo della accademia. In ogni caso la pratica della parametrizzazione è sicuramente un eccellente esercizio per chi lavora con il dBASE III.

MC

```
* MC6 - codificazioni tramite macro
*       struttura, maschera
use province
set talk off
clear
cnm=4           && NUMERO DEI CAMPI
p1="@!"
p2="99999"
p3="9999"
a11="p110105026SIGLA DELLA CITTA         :SIGLA"
a12="p112105026NOME DELLA CITTA          :CITTA"
a13="p214105026CODICE POSTALE            :CAP"
a14="p316105026PREFISSO TELELESELETTIVO :PRTEL"
*
cnc=1           && MASCHERA VUOTA
do while cnc<=cnm
 ac ="a"+str(cnc+10,2)
 rg =val(subs(&ac,3,2))
 c1 =val(subs(&ac,5,2))
 ln =val(subs(&ac,9,2))
 @ rg,c1 say subs(&ac,11,ln)
 cnc=cnc+1
enddo
DO WHILE .NOT. EOF()
  cnc=1          && RIEMPIMENTO MASCHERA
  do while cnc<=cnm
    ac ="a"+str(cnc+10,2)
    rg =val(subs(&ac,3,2))
    c2 =val(subs(&ac,7,2))
    cmp=subs(&ac,37)
    @ rg,c2 say &cmp
    cnc=cnc+1
  enddo
  cnc=1          && CORREZIONE CAMPI
  do while cnc<=cnm
    ac ="a"+str(cnc+10,2)
    pi =subs(&ac,1,2)
    rg =val(subs(&ac,3,2))
    c2 =val(subs(&ac,7,2))
    cmp=subs(&ac,37)
    pi=&pi
    @ rg,c2 get &cmp picture "&pi"
    read
    cnc=cnc+1
  enddo
  SKIP
ENDDO

SIGLA DELLA CITTA         :          AG

NOME DELLA CITTA          :          AGRIGENTO

CODICE POSTALE            :          92100

PREFISSO TELELESELETTIVO :           0922
```

Figura 6 Problematiche di parametrizzazione. Altro vasto campo di applicazione delle MACRO consiste nella parametrizzazione di un programma, in cui quindi le varie istruzioni non sono rigide ma fanno riferimento ad elementi esterni che ne condizionano il valore e l'effetto. Nel nostro esercizio deleghiamo descrizione della Struttura, delle Picture e della Maschera di Data Entry a variabili gestibili esternamente rispetto al programma.



Partner
distribuisce

Friendly
Sistema
Computer Kit
MULTIVIEW
OKI Microline
Mannesmann Tally
SDC

Essere oggi Rivenditori o Consulenti Edp e' sempre piu' difficile.

E sempre piu' difficile e' anche essere dei buoni fornitori globali.

Noi della Partner stiamo da tempo provando a dare ai Professionisti di questo settore quello di cui hanno bisogno.

Prezzi di concorrenza, alta qualita' dei prodotti,gamma completa, pronto e qualificato service.

La Partner con le sue Divisioni Trade, Diffusion, Sistemi e Assistance cerca di "cucire" il rapporto a misura delle Vostre esigenze.

Con il Vostro aiuto potremo farlo sempre meglio.

Consultateci quindi per qualsiasi esigenza.

Partner

GROUP

Sede Centrale
00144 Roma V.le C. Pavese 410 Tel. 06.5003136
Telex 610366 Telefax 06.5002383
Filiali Regionali
20145 Milano P.za Giulio Cesare 5 Tel.02.48193551
09041 Dolianova (CA) Via Mazzini 10 Tel. 070.740569
98100 Messina P.za Trombetta Is. 106 Tel. 090.713819