di Andrea de Prisco

BASAL 2.1

Un linguaggio strutturato per il vostro VIC-20

Nel comune lessico dell'informatica, oltre ai termini "BASIC", "PERSONAL" e... "MC" (!), ne esistono altri come "Programmazione strutturata", "Codice oggetto", "Compilatore" che, pur essendo indubbiamente meno diffusi, non sono meno importanti. In questo articolo ci occuperemo per l'appunto del "nidificato mondo delle strutture tipo scatole cinesi" del quale probabilmente un po' tutti avranno sentito parlare, ma purtroppo ben pochi avranno provato l'ebbrezza dell'esperienza diretta. Se non si ha come minimo a disposizione un sistema Apple II + Language Card + Pascal, niente da fare: si resta in castigo a "giocare" col BASIC. Per tentare di ovviare a questo inconveniente, sempreché ci perdoniate questo piccolo peccato di gioventù, abbiamo inventato ex nono per voi un mini-mini-linguaggetto strutturato, supportato nientepopodimeno che dal BASIC.

A ragione, qui qualcuno si sarà già messo le mani nei capelli. Di fatto però il BASAL 2.1 (questo il nomignolo-fritto-misto fra BASIC e PASCAL) non ha il più pallido intento di sembrare un mezzo serio per programmare. Lo scopo è solo quello di diffondere queste benedette scatole cinesi al di là delle varie pagine di libri e riviste che trattano questo tema, proponendo il programma BASIC listato in queste pagine che creerà l'ambiente adatto. Sarà così possibile adoperare il BASAL direttamente sul vostro Personal: compito del programma è appunto quello di trasformare in BASIC i vostri elaborati e di inserirli direttamente in memoria.

La versione presentata è adatta all'ultradiffuso VIC-20 + 16K; essendo però scritta in BASIC abbastanza (ma non totalmente) standard, l'adattamento su altri Personal non dovrebbe essere molto difficile.

In particolar modo per la subroutine principale (linee 11420-13060) che, accettando in ingresso un programma BASAL contenuto nell'array A$(N), restituisce all'interno dello stesso il programma BASIC corrispondente. Ci occuperemo ora, prima di parlare del BASAL, di rispolverare alcuni concetti propri della programmazione strutturata nell'intento (speriamo) di chiarire a chi è completamente a digiuno, cosa diavolo c'entrano le scatole cinesi...

Due parole per incominciare

I vantaggi della programmazione attraverso un linguaggio di tipo strutturato, come il Pascal, il PL/1 e l'Algol W, sono innumerevoli. Grazie, ad esempio, alla possibilità di definire ricorsivamente le subroutine, è possibile risolvere determinate classi di problemi che con strutture meno potenti (vedi BASIC) risulterebbero molto più impegnativi. Senza scendere nel merito (tanto più che, come vedremo, non riguarda il BASAL), citiamo soltanto casi come il calcolo del fattoriale, il problema delle torri di Hanoi, la ricerca binaria in un albero, che diventano problemi-bazzecola se programmati ricorsivamente.

L'essenza della programmazione strutturata sta comunque nella possibilità di vedere intere sezioni di programma come un'unica istruzione, e quindi nella possibilità di scrivere qualsiasi programma senza usare istruzioni di salto condizionato o incondizionato. Sembra strano ma è vero. Dopotutto è una semplice conseguenza del parlare umano: avete mai visto, o meglio, sentito qualcuno in un discorso pronunciare parole del tipo "GOTO BLA.BLA.BLA"?

I linguaggi strutturati sono felicemente molto più vicini al linguaggio umano che a quello di macchina. Facciamo un esempio: abbiamo due numeri: A e B. Se A è maggiore di B stampiamo A, altrimenti stampiamo B. In BASIC una possibile soluzione sarà:

```
10 IF A>B THEN 40
20 PRINT B
30 GOTO 50
40 PRINT A
50 ...
60 ...
```

Con un linguaggio di tipo strutturato, come il Pascal, avremo:

```
IF A>B THEN WRITE(A)
       ELSE WRITE(B);
```

che è esattamente la traduzione inglese di quanto scritto sopra.

In definitiva, grazie a particolari istruzioni strutturate (nel caso dell'IF abbiamo anche l'ELSE), è possibile spiegare quasi a parole, al calcolatore ciò che dovrà fare. Nei casi in cui il "ciò che dovrà fare" non è una singola istruzione (come WRITE), ma qualcosa di più complesso come due istruzioni o duemila, linguaggi strutturati come il Pascal e l'Algol usano delimitare l'intero blocco con le parole-chiave "BEGIN" (Inizio) e "END" (Fine). Ed è qui che "scatta" il concetto di scatola cinese. All'interno del blocco BEGIN-END è possibile racchiudere qualsiasi altra cosa, anche un intero programma zeppo di altri sottoblocchi "nidificati". È come se con la parola BEGIN si aprisse una nuova parentesi e con END si chiudesse l'ultima parentesi aperta. Altro esempiuccio: indovinate cosa fa questa porzione di programma:

```
IF ALFA>BETA THEN
BEGIN
 MAX:=ALFA;
 MIN:=BETA
END
ELSE
BEGIN
 MAX:=BETA;
 MIN:=ALFA
END;
```

È chiaro a questo punto che bene o male i salti ci sono comunque: è solo che non bisogna esplicitamente nominarli. Per coloro che non credono a ciò e vogliono a tutti i costi mortificare il Pascal o l'Algol inserendo all'interno di un programma volutamente dei salti del tipo GOTOETICHETTA, niente paura: tanto l'Algol quanto il Pascal (e vedremo... il BASAL), dispongono di quest'istruzione nonostante sia stato dimostrato che se ne può fare comodamente a meno.

Il minilinguaggio BASAL 2.1

Prima di descrivere l'intero set di istruzioni, diamo uno sguardo alla generica struttura di un programma BASAL. Useremo esempi con istruzioni a noi più familiari quali il FOR, l'IF e il GOSUB, tenendo però presente che quanto detto vale anche per le altre. Ogni programma BASAL si compone di due parti: il programma principale e, se esistono, le sue subroutine. Ciascuna di queste due parti è a sua volta composta da istruzioni semplici (le operazioni di INPUT, PRINT, gli assegnamenti ecc.) identiche al BASIC, e linee contenenti parole-chiave proprie del BASAL che necessitano delle opportune modifiche per diventare semplici istruzioni BASIC (fase di precompilazione). Bisogna inoltre chiarire che in BASAL non sono ammesse linee multiple (con più statement). Unica eccezione fanno quelle linee che non contengono parole chiave del BASAL ma solo comandi BASIC. Il programma principale inizia sempre con la parola-chiave "BEGIN" e termina con "END". All'interno del programma posso-

no starci altre compound (trad. blocchi: insieme di istruzioni racchiuse da BEGIN e END) anche nidificate l'una dentro l'altra sul tipo delle scatole cinesi. Ogni compound è vista dal BASAL come un'unica istruzione. Dato che tutti gli statement del BASAL accettano come argomento un'unica istruzione BASIC, nel caso sia necessario utilizzare, per esempio all'interno di un ciclo FOR, più istruzioni, basterà aprire una nuova compound e inserire all'interno quante linee si vogliono e di che tipo si vuole.

Facciamo due esempi:

```
1)      BASAL                   BASIC

        BEGIN                   10 FOR I=1 TO 100
         FOR I=1 TO 100         20 T=T+1
         T=T+1                  30 NEXT
        END.                    40 END

2)      BASAL                   BASIC

        BEGIN                   10 FOR I=1 TO 100
         FOR I=1 TO 100         20 PRINT I
         BEGIN                  30 T=T+1
          PRINT I               40 Q=Q+I
          T=T+1                 50 NEXT
          Q=Q+I                 60 END
         END
        END.
```

Come si può notare, nel primo caso, per I che assume valori da 1 a 100, si è dovuta ripetere una sola istruzione: T=T+1. Nel secondo caso, dato che le istruzioni da eseguire erano più di una, è stato necessario aprire una nuova compound BEGIN-END.

È obbligatorio inoltre aprire nuove compound anche quando l'argomento è un'istruzione singola e contemporaneamente parola-chiave del BASAL.

Per esempio:

```
E' SCORRETTO:           E' CORRETTO:

FOR I=1 TO 100          FOR I=1 TO 100
GOSUB *CICCIOBELLO*     BEGIN
                        GOSUB *CICCIOBELLO*
                        END
```

L'unica istruzione che può essere nidificata è il FOR all'interno di altri FOR. Esempio:

```
BASAL                   BASIC

FOR X=1 TO 5            10 FOR X=1 TO 5
FOR Y=2 TO 7            20 FOR Y=2 TO 7
FOR Z=3 TO 6            30 FOR Z=3 TO 6
A(X,Y,Z)=-1             40 A(X,Y,Z)=-1
                        50 NEXT Z,Y,X
```

Le parole-chiave del BASAL sono:

BEGIN, END, END., GOSUB, GOTO, REPEAT, UNTIL, CASE, OF, FOR, IF, THEN, ELSE, WHILE, DO, *...*.

Come dicevamo, dopo il programma vero e proprio vanno posizionate, se esistono tutte le subroutine chiamate dal programma. Ogni subroutine è identificata da un nome racchiuso fra 2 o più asterischi e vale la solita regola: dopo il nome si può porre un'unica istruzione BASIC o una compound BEGIN-END con dentro tutto quello che si vuole. Facciamo un esempio: questo programmino calcola, dati A e B, A!+B!:

```
BEGIN
  INPUT "[illegible]=";A
  INPUT "[illegible]=";B
  X=A
  GOSUB *X!*
  A=X
  X=B
  GOSUB *X!*
  B=X
  PRINT"[illegible]!+B!=";A+B
END.
*X!*
BEGIN
  FORI=X-1TO2STEP-1
  X=X*I-(X=0)
END
```

```
20 INPUT"[illegible]=";A
30 INPUT"[illegible]=";B
40 X=A
50 GOSUB140
60 A=X
70 X=B
80 GOSUB140
90 B=X
100 PRINT"[illegible]!+B!=";A+B
110 END
140 FORI=X-1TO2STEP-1
150 X=X*I-(X=0):NEXT
160 RETURN
```

Gli Statement del BASAL 2.1

Per descrivere correttamente il set di istruzioni del BASAL indicheremo con:

<ARGOMENTO>: un'istruzione semplice BASIC o un blocco BEGIN-END con dentro ciò che si vuole (compresi volendo anche altri sottoblocchi nidificati).

<EXP>: un numero o una variabile o un'espressione matematica composta di simboli, numeri e variabili (2-3*I+Z, ad esempio).

<VAR>: una variabile numerica intera o reale.

<COST>: una costante numerica.

<BOOLE>: un'espressione logica del tipo A>B o (A>B) AND (C=3) o complicata quanto si vuole.

Per ogni caso verrà indicato qualche esempio BASAL con relativa traduzione in BASIC che dovrebbe chiarire ogni dubbio più di ogni commento o spiegazione.

1) IF <BOOLE> THEN
   <ARGOMENTO 1>
   ELSE
   <ARGOMENTO 2>

se la prova ha dato esito vero sarà eseguito <ARGOMENTO 1> altrimenti <ARGOMENTO 2>. Il ramo ELSE è facoltativo. Esempio:

```
BEGIN
  IF A>0 THEN           20 IFA>0THEN40
  BEGIN                 30 GOTO90
    PRINT S             40 PRINTS
    D=D+R               50 D=D+R
  END                   60 GOTO120
  ELSE                  90 A=A+1
  BEGIN                 100 D=-4
    A=A+1               120 END
    D=-4
  END
END.
```

2) FOR <VAR> = <EXP1> TO <EXP2> STEP <EXP3>
   <ARGOMENTO>

solo in questo caso <ARGOMENTO> può essere un altro FOR nidificato all'interno di esso. Praticamente identico al BASIC; lo step può essere omesso se vale 1. Esempio:

```
BEGIN
  FOR I=1 TO 5          20 FOR I=1 TO 5
  S=S+3                 30 S=S+3:NEXT
END.
```

3) CASE <VAR> OF
   BEGIN
   <EXP1> ←<ARGOMENTO1>
   <EXP2>←< ARGOMENTO2>
   "
   "
   "
   OT←<ARGOMENTOn>
   END

è eseguita l'istruzione o la serie di istruzioni che ha come indice lo stesso valore della variabile indicata nello statement. L'indice OT sta per "otherwise", è facoltativo e indica l'istruzione o la serie di istruzioni da eseguire negli altri casi. È obbligatorio racchiudere l'insieme dei casi in un blocco BEGIN-END. Esempio:

```
BEGIN
  CASE A OF
  BEGIN                 40 IFA=3THEND=D*F:GOTO110
    3←D=D*F             50 IFA<>2THEN90
    2←BEGIN             60 D=D*F
      D=D*F             70 PRINTQ
      PRINT Q           80 GOTO110
    END                 90 C=C*F
    OT←C=C*F            110 END
  END
END.
```

4) REPEAT
"
"
"
UNTIL <BOOLE>

È l'unica istruzione che non necessita compound quando le istruzioni da ripetere sono più di una. Praticamente è un loop condizionato: l'insieme di istruzioni racchiuse fra REPEAT e UNTIL è ripetuto fino a quando la prova è vera. Dato che la prova è situata alla fine del blocco, esso sarà eseguito sempre almeno una volta. Esempio:

```
BEGIN
  REPEAT                  30 A=A+3
  A=A+3                   40 D=D-3
  D=D-3                   50 IFNOT(A+D<12)THEN30
  UNTIL A+D<12            60 END
END.
```

5) WHILE <BOOLE> DO
<ARGOMENTO>

L'istruzione o la serie di istruzioni sono ripetute fintantoché la prova è vera; al contrario del REPEAT... UNTIL... , se la prova risulta subito falsa è saltato tutto l'argomento. Esempio:

```
BEGIN
  WHILE P<>A-Z DO         20 IFNOT(P<>A-Z)THEN70
  BEGIN                   40 D=D*F
    D=D*F                 50 A=A-E
    A=A-E                 60 GOTO20
  END                     70 END
END.
```

6) FOR <VAR> = <COST1>; <COST2>; ...; <COSTn> DO
<ARGOMENTO>

Per la variabile indicata che assume i valori indicati nello statement e nell'ordine dato, è eseguito l'argomento. Può essere usato una sola volta nel programma a condizione che non vi siano DATA e non venga usata la variabile II: Esempio:

```
BEGIN
  FOR J=2;4;5;-1;0 DO     20 RESTORE:DATA2,4,5,
  BEGIN                   -1,0:FORII=1TO5:READJ
    S=SD-J                40 S=SD-J
    PRINT J+3;J-3         50 PRINTJ+3;J-3
  END                     60 NEXT
.END.                     70 END
```

7) *NOME ETICHETTA* oppure *NOME ETICHETTA*
<ARGOMENTO>

Serve per inserire etichette nel programma e per identificare le subroutine. È obbligatorio che ogni etichetta sia puntata da almeno un'istruzione di GOTO o GOSUB. Esempio:

```
BEGIN
  *ANDREA*                30 S=S-R
  S=S-R                   40 L=L-4
  L=L-4                   50 GOSUB90
  GOSUB *ORNELLA*         60 GOTO30
  GOTO *ANDREA*           70 END
END.                      90 PRINT"BASAL 2.1":RETURN
*ORNELLA*
PRINT "BASAL 2.1"
```

A questi vanno chiaramente aggiunti tutti gli altri statement (INPUT, PRINT, READ, DATA, REM, OPEN, CLOSE ecc.) che non necessitando precompilazione (solo il numero linea è aggiunto) saranno scritte come istruzioni BASIC nella abituale sintassi del BASIC.

La fase di precompilazione

Per trasformare un programma BASAL nel corrispondente "fratello" in BASIC, il precompilatore compie essenzialmente i seguenti cinque passi:

1) È individuata l'istruzione da tradurre (le linee non contenenti parole-chiave del BASAL non sono modificate).

2) È analizzato l'argomento di tale istruzione (semplice o compound?).

3) Nel caso di compound è ricercato l'indirizzo dell'END relativo al BEGIN.

4) A seconda del tipo di istruzione (for, if, while, ecc.) avvengono le specifiche trasformazioni del caso.

5) È aggiunto il numero linea.

Facciamo un primo esempio: vediamo come il precompilatore tradurrebbe questa porzione di programma:

```
BEGIN
  FOR I=1 TO 10
  H=H+I
END.
```

Esso è memorizzato all'interno dell'array A$(I), nelle prime 4 locazioni; quindi A$(1) = "BEGIN"; A$(2) = "FORI = 1TO10" A$(3) = "H = H + 1"; A$ (4) = "END". La fase iniziale della

```
10000 REM        ++++++++++++++++++++++++++++++++++++++++++++++++
10010 REM        +  ***    **    ****   **   *       ****     *  +
10020 REM        +  *  *  *  *  *   *  *  *  *      *    *     *  +
10030 REM        +  ***   ****  ****   ****  *      ****     *  +
10040 REM        +  *  *  *  *     *   *  *  *      *        *  +
10050 REM        +  ***   *  *  ****   *  *  ****   ****  *  *  +
10060 REM        +                                              +
10070 REM        +            (C) 1983 ADP-SOFTWARE             +
10080 REM        +              MC  MICROCOMPUTER               +
10090 REM        ++++++++++++++++++++++++++++++++++++++++++++++++
10100 POKE56,91:POKE36879,29:GOTO10430
10110 DIMA$(100):FL=0
10120 PRINT"□":A$(1)="BEGIN":PRINT"  BEGIN"
10130 FORI=2TO100:N=I
10140 IFA$(I)<>""THENPRINT"  "+A$(I):GOTO10390
10150 INPUTR$(I)
10160 IFA$(I)=""THENN=N-1:I=999:NEXT:GOTO10630
10170 POX=0:FORKL=1TOLEN(A$(I))
10180 IFMID$(A$(I),KL,1)="*"THENPOX=KL
10190 NEXT
10200 KL$=MID$(A$(I),POX+1,100)
10210 GOSUB10280
10220 A$(I)=LEFT$(A$(I),POX)+KL$
10230 FORZ=0TOHX:PRINT"■";:NEXT:PRINT"  ";A$(I)
10240 GOTO10390
10250 REM ************************
10260 REM * ROUTINE ABBREVIAZIONI *
10270 REM ************************
10280 HX=(LEN(KL$)+2)/22
10290 IFLEFT$(KL$,2)="C "THENKL$="CASE"+RIGHT$(KL$,LEN(KL$)-1)
10300 IFLEFT$(KL$,2)="G "THENKL$="GOSUB"+RIGHT$(KL$,LEN(KL$)-1)
10310 IFLEFT$(KL$,2)="I "THENKL$="INPUT"+RIGHT$(KL$,LEN(KL$)-1)
10320 IFLEFT$(KL$,2)="B"ORRIGHT$(KL$,2)="+B"THENKL$=KL$+"EGIN"
10330 IFLEFT$(KL$,2)="U "THENKL$="UNTIL"+RIGHT$(KL$,LEN(KL$)-1)
10340 IFLEFT$(KL$,2)="W "THENKL$="WHILE"+RIGHT$(KL$,LEN(KL$)-1)
10350 IFLEFT$(KL$,2)="IF"ANDRIGHT$(KL$,1)="T"THENKL$=KL$+"HEN"
10360 IFLEFT$(KL$,2)="R"THENKL$="REPEAT"
10370 IFLEFT$(KL$,1)="?"THENKL$="PRINT"+RIGHT$(KL$,LEN(KL$)-1)
10380 RETURN
10390 IFA$(I)<>"E"THENNEXT:GOTO10540
10400 A$(I)="":L=I-1:GOSUB10840:PRINT"□"
10410 FORK=1TON-1:PRINT"  ";A$(K):NEXT
10420 GOTO10150
10430 PRINT"□■■       BASAL 2.1       ■"
10440 PRINT"■ SCELTA OPZIONE"
10450 PRINT" ──────────"
10460 PRINT"■ L:LIST  SU  VIDEO  "
10470 PRINT"■ E:EDIT  DI  LINEA"
10480 PRINT"■ P:PRECOMPILAZIONE"
10490 PRINT"■ C:  CONCATENAMENTO"
10500 PRINT"■ ■■:NUOVO PROGRAMMA"
10510 PRINT"■ ■■:SAVE  PROGRAMMA"
10520 PRINT"■ ■■:LOAD  PROGRAMMA"
10530 PRINT"■■  (C)  ADP  SOFTWARE  ■";
10540 GETZ$:IFZ$=""THEN10540
10550 IFZ$="E"THENGOSUB10820:GOTO10430
10560 IFZ$="C"THEN10120
10570 IFZ$="/"THENCLR:GOTO10110
10580 IFZ$="P"THENPRINT"□":GOSUB11460:GOTO11230
10590 IFZ$="▼"THENPRINT"□":GOTO11090
10600 IFZ$="F"THENOH=80:OPEN1,4:CMD1:GOTO10640
10610 IFZ$="L"THENPRINT"□":GOTO11140
10620 IFZ$<>"L"THEN10430
10630 OH=22
10640 PRINT"□":H=0:FORI=1TON
10650 IFRIGHT$(A$(I-1),5)="BEGIN"THENH=H+1-(OH>30)
10660 IFA$(I)="END"ORA$(I)="END."THENH=H-1+(OH>30)
10670 IFLEN(A$(I))+H<=OHTHEN10740
10680 B$=A$(I)
10690 IFB$=""THENPRINT:GOTO10750
10700 FORII=1TOH:PRINT" ";:NEXT
10710 PRINTLEFT$(B$,OH-H);
10720 B$=MID$(B$,OH+1-H,100)
10730 GOTO10690
10740 PRINTSPC(H);A$(I)
10750 IFPEEK(653)<>0THEN10750
10760 NEXT
10770 GETZ$:IFZ$=""THEN10770
10780 CLOSE1:GOTO10550
10790 REM ************************
10800 REM * ROUTINE EDIT DI LINEA *
10810 REM ************************
10820 I=2
10830 L=N
10840 T=I:G=0
10850 T=T-1
10860 IFT>LTHENT=T-1
10870 IFT<1THENT=T+1
10880 PRINT"□■■■■";A$(T)
10890 PRINT"■";TAB(244);TAB(229);"■";T;"■■ "
10900 GETZ$:IFZ$=""THEN10900
10910 IFZ$=CHR$(145)THEN10850
10920 IFZ$="I"THEN11030
10930 IFZ$="D"THEN11060
10940 IFZ$="C"THEN10120
10950 IFZ$=CHR$(17)THENT=T+1:GOTO10860
10960 IFZ$=CHR$(13)THENRETURN
10970 IFZ$<>CHR$(29)THEN10900
10980 PRINT"□■■■";A$(T)
10990 PRINT"■";TAB(244);TAB(229);"■";T;"■■ ■"
11000 INPUTKL$:TY=I:I=T:GOSUB10280:A$(T)=KL$:I=TY:IFQ>1THENQ=Q-1:T=T+1:GOTO10980
11010 Q=0:GOTO10880
11020 RETURN
```

precompilazione inizia dalla stringa A$(2). Troviamo un FOR: è questa una parola chiave del BASAL, quindi da tradurre. L'argomento è un'istruzione semplice: l'unica trasformazione è data dall'assegnamento A$(3) = A$(3) + ":NEXT". Con l'aggiunta del numero linea e la cancellazione del BEGIN iniziale e dell'END finale otteniamo il corrispondente programma BASIC. Facciamo un altro esempio: è da tradurre il programma:

```
BEGIN
 IF A>0 THEN
 B=B+1
 ELSE
 BEGIN
  H=H+3
  A=A-1
 END
END.
```

Come sempre si trova memorizzato all'interno dell'array A$(I), questa volta nelle prime 9 locazioni. L'IF è un po' più complicato da tradurre in quanto vi sono più cose da controllare.

In questo caso il ramo THEN è composto da una istruzione: (A$(3) = "B = B +1"). Occorre concatenare la stringa 3 alla stringa 2 e cancellare il contenuto di A$(3), quindi A$(2) = A$(2) + A$(3) e A$(3) = "". Il secondo passo è controllare se esiste il ramo ELSE e in caso positivo, il nostro, calcolare (leggi: "cercare a tentoni") dove termina. Essendo questo ramo caratterizzato da una compound BEGIN-END, è cercato l'indirizzo (il numero di stringa) dell'END relativo al nostro BEGIN. In questo caso: 8. Dato che, nel caso in cui si verifica che A > 0, si dovrà eseguire B = B + 1 e saltare tutto il ramo ELSE, basta aggiungere ulteriormente alla stringa 2 un "GOTO (fine ramo else)", quindi:

A$(2) = A$(2) + "GOTO" + STR$ ((J + 1)*10) dove, in questo

```
11030 Q=Q+1:FORW=NTOTSTEP-1
11040 A$(W+1)=A$(W)
11050 NEXT:A$(T)="":N=N+1:L=L+1:I=I+1:GOTO10980
11060 FORW=TTOL
11070 A$(W)=A$(W+1)
11080 NEXT:N=N-1:L=L-1:I=I-1:GOTO10880
11090 OPEN2,1,1,"PROG"
11100 PRINT#2,N
11110 FORI=1TON
11120 PRINT#2,A$(I)
11130 NEXT:CLOSE2:GOTO10430
11140 CLR:OPEN2,1,0,"PROG"
11150 INPUT#2,N:DIMA$(100)
11160 FORI=1TON
11170 INPUT#2,A$(I)
11180 NEXT:CLOSE2:GOTO10430
11190 REM ************************************************
11200 REM * QUESTA ROUTINE  TRASFERISCE AUTOMATICAMENTE *
11210 REM * IL PROGRAMMA GENERATO NELLA MEMORIA DEL VIC *
11220 REM ************************************************
11230 S=24320:Z=0:PRINT"□":FORI=1TON
11240 FORK=1TOLEN(A$(I))
11250 PO=ASC(MID$(A$(I),K,1))
11260 IFPO=34THENZ=1-Z
11270 IF(PO=32ORPO=160)ANDZ=0THEN11290
11280 S=S-1:POKES,PO
11290 NEXT
11300 S=S-1:POKES,0
11310 NEXT:POKE56,S/256-2:POKE-1,1:POKE-2,2:POKE-3,3
11320 I=24320
11330 I=I-1
11340 IFPEEK(I)<>0THENA$=A$+CHR$(PEEK(I)):GOTO11330
11350 IFPEEK(I-1)=1ANDPEEK(I-2)=2ANDPEEK(I-3)=3THENPRINT"□□□□";A$:PRINT"□RUN":G
OTO11400
11360 POKE24319,I/256
11370 POKE24318,(I/256-PEEK(24319))*256
11380 PRINT"□□□□";A$:A$=""
11390 PRINT"GOTO11410":
11400 POKE198,10:POKE631,13:POKE632,13:END
11410 I=PEEK(24319)*256+PEEK(24318):GOTO11330
11420 REM ************************************
11430 REM * TRADUZIONE DA BASAL A BASIC DEL *
11440 REM * PROGRAMMA  CONTENUTO  IN  A$(N) *
11450 REM ************************************
11460 FORI=2TON
11470 FORH=1TOLEN(A$(I))
11480 IFMID$(A$(I),H,1)<>"!"THEN11500
11490 A$(I)=LEFT$(A$(I),H-1)+";"+MID$(A$(I),H+1,100)
11500 NEXT:NEXT
11510 FORI=2TON
11520 IFLEFT$(A$(I),2)="IF"THEN12410
11530 IFLEFT$(A$(I),3)="FOR"THEN12610
11540 IFLEFT$(A$(I),2)="DO"THEN12150
11550 IFLEFT$(A$(I),5)="UNTIL"THEN12060
11560 IFLEFT$(A$(I),5)="WHILE"THEN12270
11570 NEXT
11580 FORI=2TON
11590 IFLEFT$(A$(I),4)="CASE"THEN11800
11600 NEXT
11610 FORI=2TON
11620 IFLEFT$(A$(I),1)="#"THEN12740
11630 NEXT
11640 FORI=2TON
11650 IFLEFT$(A$(I),2)="IF"ORLEFT$(A$(I),2)="DO"THEN12780
11660 IFLEFT$(A$(I),1)="@"THENA$(I)=RIGHT$(A$(I),LEN(A$(I))-1):GOTO12780
11670 IFLEFT$(A$(I),1)="#"THENA$(I)=RIGHT$(A$(I),LEN(A$(I))-1):GOTO11680
11680 NEXT
11690 PRINT"□":FORI=1TON
11700 IFA$(I)="BEGIN"ORA$(I)="END"THENA$(I)=""
11710 IFA$(I)="END,"THENA$(I)="END"
11720 A$(I)=STR$(I*10)+A$(I)
11730 IFLEN(A$(I))>4THENPRINTA$(I)
11740 NEXT
11750 IFPEEK(197)=64THEN11750
11760 RETURN
11770 REM ************************
11780 REM * PREC. CASE..OF..... *
11790 REM ************************
11800 H=4
11810 H=H+1
11820 IFMID$(A$(I),H,2)<>"OF"THEN11810
11830 B$=MID$(A$(I),5,H-5)
11840 GOSUB12880:A$(I)="":A$(I+1)="":A$(J)=""
11850 J$=STR$((J+1)*10)
11860 FORT=I+2TOJ-1
11870 H=1:IFA$(T)="OT+BEGIN"THENK=1:D=I:I=T:GOSUB12890:I=D:A$(T)="":A$(J)="":GOT
O11990
11880 H=H+1
11890 IFMID$(A$(T),H,1)<>"+"THEN11880
11900 H=H-1:K=LEN(A$(T))-H-1
11910 L$=LEFT$(A$(T),H):R$=RIGHT$(A$(T),K)
11920 IFL$="OT"THENA$(T)=R$:GOTO11950
11930 IFR$="BEGIN"THEN11960
11940 A$(T)="IF"+B$+"="+L$+"THEN"+R$+":GOTO "+J$
11950 NEXT:GOTO11600
11960 D=I:I=T:K=1:GOSUB12890:I=D
11970 A$(T)="IF"+B$+"<>"+L$+"THEN "+STR$((J+1)*10)
11980 A$(J)="@GOTO"+J$
11990 T=J
12000 NEXT
12010 I=T+1
12020 GOTO11600
12030 REM ************************
12040 REM * PREC. REPEAT..UNTIL *
12050 REM ************************
12060 J=I
12070 J=J-1
12080 IFA$(J)<>"REPEAT"THEN12070
12090 A$(J)=""
12100 A$(I)="IFNOT("+RIGHT$(A$(I),LEN(A$(I))-5)+")THEN"+STR$((J+1)*10)
12110 GOTO11570
12120 REM ************************
12130 REM * PREC. SALTI INCOND. *
12140 REM ************************
12150 H=0
12160 H=H+1:IFMID$(A$(I),H,1)<>"#"THEN12160
12170 H=H-1:K=LEN(A$(I))-H
12180 B$=RIGHT$(A$(I),K)
12190 M=1
12200 M=M+1:IFLEFT$(A$(M),K)<>B$THEN12200
12210 A$(I)=LEFT$(A$(I),H)+STR$((M+1)*10)
12220 IFLEFT$(A$(I),4)="GOTO"THENA$(M)=A$(M)+"%"
12230 GOTO11570
12240 REM ************************
12250 REM * PREC. WHILE...DO... *
12260 REM ************************
12270 B$=MID$(A$(I),6,LEN(A$(I))-7)
12280 A$(I)="IFNOT("+B$+")THEN"
12290 IFA$(I+1)="BEGIN"THEN12330
12300 A$(I)=A$(I)+STR$((I+2)*10)
12310 A$(I+1)="@"+A$(I+1)+":GOTO"+STR$(I*10)
12320 GOTO11570
12330 GOSUB12880
12340 A$(I)=A$(I)+STR$((J+1)*10)
12350 A$(I+1)=""
12360 A$(J)="@GOTO"+STR$(I*10)
12370 GOTO11570
12380 REM ************************
12390 REM * PRECOMPILAZIONE IF *
12400 REM ************************
12410 IFA$(I+1)="BEGIN"THEN12490
12420 A$(I)=A$(I)+A$(I+1)
12430 A$(I+1)=""
12440 IFA$(I+2)<>"ELSE"THENA$(I)="#"+A$(I):GOTO11570
12450 D=I:I=I+2:GOSUB12880:I=D
12460 A$(I)=A$(I)+":GOTO"+STR$((J+1)*10)
12470 A$(I+2)=""
12480 I=I+2:GOTO11570
12490 A$(I)=A$(I)+STR$((I+2)*10)
12500 GOSUB12880
12510 A$(I+1)="@GOTO"+STR$((J+1)*10)
12520 A$(J)=""
12530 IFA$(J+1)<>"ELSE"THEN11570
12540 H=J:D=I:I=J+1
12550 GOSUB12880:I=D
12560 A$(H)="@GOTO"+STR$((J+1)*10)
12570 A$(H+1)="":GOTO11570
12580 REM ************************
12590 REM * PRECOMPILAZIONE FOR *
12600 REM ************************
12610 H=0
12620 IFRIGHT$(A$(I),2)="DO"THEN12980
12630 IFA$(I+1)="BEGIN"THEN12670
12640 IFLEFT$(A$(I+1),3)="FOR"THENI=I+1:H=H+1:GOTO12620
12650 A$(I+1)=A$(I+1)+":NEXT"
12660 J=I+1:GOTO12700
12670 GOSUB12880
12680 A$(I+1)=""
12690 A$(J)="NEXT"
12700 IFH=0THEN11570,
12710 FORT=1TOH
12720 A$(J)=A$(J)+":NEXT"
12730 NEXT:GOTO11570
12740 IFRIGHT$(A$(I),1)="%"THENA$(I)="":GOTO11630
12750 A$(I)=""
12760 IFA$(I+1)<>"BEGIN"THENA$(I+1)=A$(I+1)+":RETURN":GOTO11630
12770 GOSUB12880:A$(I+1)="":A$(J)="RETURN":GOTO11630
12780 L=LEN(A$(I))-1
12790 L=L+1
12800 IFMID$(A$(I),L,1)<>" "THEN12790
12810 S=VAL(RIGHT$(A$(I),LEN(A$(I))-L))-10
12820 S=S+10:IFA$(S/10)=""ORA$(S/10)="END"ORA$(S/10)="BEGIN"THEN12820
12830 A$(I)=LEFT$(A$(I),L)+STR$(S):GOTO11680
12840 REM ************************
12850 REM * RICERCA  DELL'  END *
12860 REM * RELATIVO A UN BEGIN *
12870 REM ************************
12880 K=0
12890 J=I
12900 J=J+1
12910 IFRIGHT$(A$(J),5)="BEGIN"THENK=K+1:GOTO12900
12920 IFA$(J)="END"THENK=K-1:KK=1
12930 IFK<>0THEN12900
12940 RETURN
12950 REM ************************
12960 REM * PREC. FOR I=...,...,..DO *
12970 REM ************************
12980 A$=A$+"RESTORE:DATA":T=0
12990 K=K+1:IFMID$(A$(I),K,1)<>"="THEN12990
13000 FORR=K+1TOLEN(A$(I))-2
13010 IFMID$(A$(I),R,1)=";"THENA$=A$+",":T=T+1:GOTO13030
13020 A$=A$+MID$(A$(I),R,1)
13030 NEXT
13040 B$=MID$(A$(I),4,K-4)
13050 A$(I)=A$+":FORII=1TO"+STR$(T+1)+":READ"+B$
13060 A$="":GOTO12620
```

caso, J vale appunto B. Anche la stringa 4 è annullata; il nostro programma è diventato:

```
BEGIN
 IF A>0 THEN B=B+1:GOTO 90
 BEGIN
  H=H+3
  A=A-1
 END
END.
```

Ci siamo quasi: non resta che togliere tutti i BEGIN e gli END; sostituire a "END." la stringa END e aggiungere il numero di linea (dato dall'indice di stringa moltiplicato per 10) a tutte le stringhe non nulle, ottenendo:

```
20 IF A>0 THEN B=B+1:GOTO 90
60 H=H+3
70 A=A-1
90 END
```

Che è per per l'appunto il programma BASIC corrispondente al programma BASAL da cui siamo partiti. Tutto qui.

BASAL 2.1: note al programma

Buona parte del listato basic presentato nelle pagine precedenti funge da sistema operativo per il BASAL. È infatti possibile registrare programmi su nastro, rileggerli, eseguire l'edit di linea, list su stampante, su video e usare molte abbreviazioni nella fase di input. Facendo partire l'esecuzione del programma appare il menu: si accede alle varie opzioni schiacciando le lettere indicate; dove la lettera è in campo inverso vuol dire che per evitare pressioni accidentali, bisogna schiacciarla insieme allo Shift. Per il list su stampante basta premere il tasto commodore e la lettera "L".

Per input-are un programma BASAL basta premere da menu Shift "N" che sta per Nuovo-Programma. Essendo obbligatorio il BEGIN iniziale, esso viene posto automaticamente in memoria e quindi richiesta la seconda linea. Dato che il VIC non accetta in input alfanumerico stringhe contenenti virgole, nel caso sia necessario inserire linee con questo carattere, basterà sostituirlo con il carattere "!" che si ottiene digitando Shift "-" (meno).

Ad esempio: per input-are la linea A%=MX (315) si dovrà digitare A%=MX (315). E veniamo alle abbreviazioni concesse: la prima è il punto interrogativo che sta per PRINT. Tutte le altre si attivano con il primo carattere dello statement seguito da uno spazio bianco. Così per scrivere CASE G OF si potrà facoltativamente digitare C G OF prima del return di linea.

Le parole chiave che si possono abbreviare sono: CASE, GOSUB, INPUT, BEGIN, UNTIL, WHILE, REPEAT, THEN. Per BEGIN e REPEAT, che non sono seguite mai da altro, si può omettere lo spazio e digitare rispettivamente B e Return o R e Return.

Finita l'operazione di Input, premere nuovamente Return e, dopo il list, qualsiasi tasto, ad eccezione di quelli indicati nel menu, per tornare al menu. Digitando "L" si ha il listing su video del programma BASAL. Essendo lo scroll molto veloce, è possibile arrestarlo momentaneamente tenendo premuto o lo Shift o il tasto Commodore o CTRL.

Chiaramente è possibile usare anche lo Shift Lock che permette di arrestare lo scroll senza tenere impegnato alcun dito. Il byte della memoria del VIC che "sente" la pressione di questi tre tasti è il 653 (vedere linea 10750).

L'edit avviene una linea per volta (sul tipo delle programmabili) e per attivarlo basta digitare "E", seguita da Return solo se si è in fase di input. Con i tasti CRSR-su e CRSR-giù si scorre il programma avanti e indietro; con CRSR-destra e CRSR-sinistra ci si posiziona sui caratteri da correggere. Digitando invece "I" o "D" prima di CRSR-des, o sin. si possono inserire o cancellare delle linee (ad ogni pressione).

Per comprendere meglio come funziona l'edit facciamo un esempio: digitate questo mini programma:

```
BEGIN
 PRINT Q
 I=I+1
END.
```

e dopo essere tornati al menu decidiamo di sostituire alla Q una T e di togliere la linea d'assegnamento.

Premere "E" per andare in edit, una volta CRSR-giù per posizionarsi sulla linea 2 e tramite CRSR--destra sulla Q.

A questo punto digitiamo "T" per la sostituzione, return per reinserire la linea corretta, una volta CRSR-giù per posizionarsi sulla linea 3 e il tasto "D" per togliere la linea con l'assegnamento. Per uscire dall'ambiente edit premere return.

Se si vogliono aggiungere altre linee di coda al programma, tornati al menu basterà premere "C" che sta per concatenamento. Da menu, "P" sta per precompilazione e serve appunto per precompilare il programmi BASAL e far partire la loro esecuzione.

Prima però di essere inserito automaticamente in memoria il nuovo programma viene listato sul video e per procedere basta toccare qualsiasi tasto.

Fra la fase di precompilazione e il fatidico "RUN" che fa partire l'esecuzione, vi è una fase alquanto delicata. Il programma BASIC codice oggetto si trova memorizzato all'interno dell'array A$(N) (lo stesso in cui stava parcheggiato il programma BASAL): bisogna trasferirlo dall'array, alla memoria vera e propria del VIC.

Per risolvere questo problema, tutto il contenuto delle stringhe è dapprima copiato in una zona protetta dalla memoria (abbassando il top con POKE 56,91) e successivamente ogni linea di programma, prelevata con delle PEEK, viene visualizzata sullo schermo.

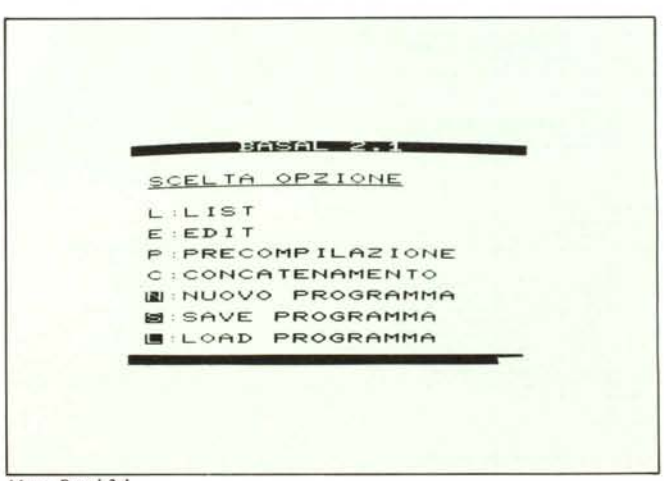
A questo punto un CHR$(13), corrispondente al [RETURN] da tastiera, è forzato all'interno del buffer, che scaricandosi sul video provoca l'inserimento automatico in memoria della linea visualizzata.

Essendo inoltre l'esecuzione arrestata ad ogni inserimento di linea, è necessario ogni volta visualizzare anche un GOTO 11410 che, con un secondo [RETURN] nel buffer, permette appunto di continuare.

Chi è interessato ai dettagli, faccia riferimento alla routine a partire dalla linea 11190.

Tutto ciò, per motivi puramente estetici, avviene utilizzando caratteri dello stesso colore dello sfondo, sicché chi sta davanti allo schermo non si accorge di cosa sta succedendo.

Togliendo di contro la "E" in campo inverso (importante: solo



Menu Basal 2.1.

```
BEGIN
  A=A*F
  IF A>0 THEN
    BEGIN
      PRINT"
      INPUT F
    END
  ELSE
    BEGIN
      GOSUB *CIAO*
      SHIFT*J
      INPUT Y
    END
END.
*CIAO*
S=RE-T*12-F*Y+5↑J
```

Listing su video

```
? S=S*FD
```

Edit di linea

```
BEGIN
A=A*F
REPEAT
PRINT"
IF A>0 THEN
UNTIL F=T-7
?
```

Fase di input.

```
100 REM "■ = CTRL BLK"
110 REM "■ = CTRL WHT"
120 REM "■ = CTRL RED"
130 REM "■ = CTRL CYN"
140 REM "■ = CTRL PUR"
150 REM "■ = CTRL GRN"
160 REM "■ = CTRL BLU"
170 REM "■ = CTRL YEL"
180 REM "■ = RVS ON  "
190 REM "■ = RVS OFF "
200 REM "■ = CRSR GIU"
210 REM "■ = CRSR DES"
220 REM "■ = CRSR SU "
230 REM "■ = CRSR SIN"
240 REM "■ = HOME    "
250 REM "■ = CLR     "
260 REM "■ = INST    "
```

Simboli di controllo ottenibili tra virgolette e tasti da premere.

```
11090 INPUT"■■NOME FILE ";LJ$:LJ$=LJ$+",S,W"
11095 OPEN2,8,2,LJ$
11140 CLR:INPUT"■■NOME FILE ";LJ$:LJ$=LJ$+",P"-11089
11145 OPEN2,8,2,LJ$
```

Linee da aggiungere (alcune) o sostituire (altre) al programma per usare il BASAL 2.1 in congiunzione col disco VIC-1540. In questo caso, digitando da menu shift L o shift S, sarà obbligatorio identificare con un nome il programma BASAL da registrare. Ciò è necessario per creare sul disco un file sequenziale atto ad ospitare il programma per future riletture.

quella!) nella linea 11380, vedrete apparire in sequenza le linee tradotte nell'attimo in cui finiscono in memoria.

Conclusione

Vi raccomandiamo a questo punto, se avete intenzione di scrivere qualche programmino in BASAL, di "studiarvi" (si fa per dire!) attentamente quanto detto in merito al linguaggio, ricordandovi che va tutto liscio se e solo se è usato correttamente.

Se avete messo qualche BEGIN o END in più o in meno, se cercate di usare istruzioni in modo poco corretto o con sintassi errata non sperate di riuscire ad ottenere il voluto.

E ricordate che le prime volte, per qualsiasi cosa, è sempre un casino... MC

Il programma-esempio qui listato simula il lancio di due dadi e la loro relativa visualizzazione sullo schermo. È stato scelto questo tema per la possibilità di inserire, quasi forzatamente, diverse istruzioni BASAL inesistenti in BASIC. Tenteremo di chiarire ulteriormente ogni dubbio proponendo un esempio di applicazione pratica di programmazione strutturata. Le promesse sono mantenute: è stato risolto l'algoritmo senza l'ausilio di alcun salto condizionato o incondizionato.

```
BEGIN
  REPEAT
  PRINT"■"
  FOR I=2 TO 12 DO
  BEGIN
    PRINT"■■■■■   - - -     - - -     ■";
    FOR J=1 TO I
    PRINT"■";
    A%=RND(1)*6+1
    CASE A% OF
    BEGIN
      1 +A$="323"
      2 +A$="232"
      3 +A$="222"
      4 +A$="131"
      5 +A$="121"
      6 +A$="111"
    END
    WHILE A$<>"" DO
    BEGIN
      B$=LEFT$(A$,1)
      A$=RIGHT$(A$,LEN(A$)-1)
      CASE VAL(B$) OF
      BEGIN
        1 +PRINT"■■    ■■  ■■■■■■■■■■■■■■■■■";
        2 +PRINT"■■  ■  ■■  ■■■■■■■■■■■■■■■■■";
        3 +PRINT"■■      ■■  ■■■■■■■■■■■■■■■■■";
      END
    END
  END
  PRINT"■■■■■■■■■■■   - - -     - - -";
  REPEAT
  GET Z$
  UNTIL Z$<>""
  UNTIL Z$="*"
END.
```

```
30 PRINT"■"
40 RESTORE:DATA2,12:FOR I=1 TO 2:READ I
60 PRINT"■■■■■   - - -     - - -     ■";
70 FOR J=1 TO I
80 PRINT"■";:NEXT
90 A%=RND(1)*6+1
120 IF A%=1 THEN A$="323":GOTO190
130 IF A%=2 THEN A$="232":GOTO190
140 IF A%=3 THEN A$="222":GOTO190
150 IF A%=4 THEN A$="131":GOTO190
160 IF A%=5 THEN A$="121":GOTO190
170 IF A%=6 THEN A$="111":GOTO190
190 IF NOT(A$<>"") THEN 300
210 B$=LEFT$(A$,1)
220 A$=RIGHT$(A$,LEN(A$)-1)
250 IF VAL(B$)=1 THEN PRINT"■■    ■■  ■■■■■■■■■■■■■■■■■";:GOTO290
260 IF VAL(B$)=2 THEN PRINT"■■  ■  ■■  ■■■■■■■■■■■■■■■■■";:GOTO290
270 IF VAL(B$)=3 THEN PRINT"■■      ■■  ■■■■■■■■■■■■■■■■■";:GOTO290
290 GOTO190
300 NEXT
310 PRINT"■■■■■■■■■■■   - - -     - - -";
330 GET Z$
340 IF NOT(Z$<>"") THEN 330
350 IF NOT(Z$="*") THEN 30
360 END

READY.
```

Traduzione Basic programma "Dadi".

A sinistra programma esempio "Dadi".

I due dati sono visualizati l'uno a fianco all'altro, disegnandoli uno per volta. Dato che ogni faccia del dado è composta da pallini disposti più o meno regolarmente, sarà possibile schematizzare il loro disegno suddividendo idealmente ogni faccia in tre strisce orizzontali e, dopo aver definito delle "strisce standard", assegnare a ogni possibile numero una ben precisa sequenza di strisce. Se infatti assegnamo alla striscia "o o" il codice 1 (non il punto 1), alla striscia "o" il codice 2 e a " " il codice 3, la faccia 6 sarà generata dalla successione 1,1,1, la faccia 5 da 1,2,1, e così via. Il programma funziona esattamente così. Dopo aver spostato il cursore nella posizione in cui sarà disegnato il dado, assegnato ad A% un numero intero Random compreso tra 1 e 6, con il Primo CASE... OF ... è assegnata la sequenza di strisce da plottare. Successivamente, (col secondo CASE ... OF ...) è analizzata la stringa A$ e conseguentemente plottare le tre strisce per disegnare la tal faccia. Come si può notare dai due REPEAT inseriti, il tutto si ripete ogni qualvolta è premuto un qualsiasi tasto e il procedimento si arresta con la pressione del tasto asterisco. Semplice, no !?!