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 novo 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 di 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 I'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". E' 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;
```

E' chiaro a questa 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 mortifi-

care il Pascal o l'Algol inserendo all'interno di un programma volutamente dei salti del tipo GOTO ETICHETTA, niente paura: tanto l'Algol quanto il Pascal (e vedremo… il BASAL), dispongono di questa 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 quanta 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 possono starci altre compound (trad. blocchi: insieme di istruzioni racchiuse da BEGIN e END) anche nidificate l'una dentro l'altra sui 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
FOR I=1 TO 100          20 FOR I=1 TO 100
T=T+1                   30 T=T+1: NEXT
END.                    40 END
```

2)

```
BASAL                   BASIC
BEGIN
FOR I=1 TO 100          20 FOR I=I TO 100
BEGIN
PRINT I                 40 PRINT I
  T=T+1                 50 T=T+1
  Q=Q+I                 60 Q=Q+I
END                     70 NEXT
END.                    80 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.

E' 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 "A=";A              20 INPUT"A=";A
  INPUT "B=";B              30 INPUT"B=";B
  X=A                       40 X=A
  GOSUB *FACT*              50 GOSUB 140
  A=X                       60 A=X
  X=B                       70 X=B
  GOSUB *FACT*              80 GOSUB 140
  B=X                       90 B=X
  PRINT"A!+B!=";A+B         100 PRINT"A!+B!=";A+B
END.                        110 END
*FACT*.
BEGIN
FOR I=X-1 TO 2 STEP-1       140 FOR I=X-1 TO 2 STEP-1)
  X=X*I-(X=0)               150 X-X*I-(X=0):NEXT
END                         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.
<BOOLEAN>	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 <BOOLEAN> 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 IF A>0 THEN 40
   BEGIN               30 GOTO 90
     PRINT S           40 PRINT S
     D=D+R             50 D=D+R
   END                 60 GOTO 120
  ELSE
   BEGIN
     A=A+1             90 A=A+1
     D=-4              100 D=-4
   END
END.                   120 END
```

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

solo in questa 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. E’ obbligatorio racchiudere l'insieme dei casi in un blocco BEGIN-END. Esempio:

```
BEGIN
CASE A OF
    BEGIN
      3←D=D*F          40 IF A=3 THEN D=D*F:GOTO 110
      2←BEGIN          50 IF A<>2 THEN 90
          D=D*F        60 D=D*F
            PRINT Q    70 PRINT Q
        END            80 GOTO 110
      OT←C=C*F         90 C=C*F
END                    110 END
END.
```

4**) REPEAT**
“
“
UNTIL <BOOLEAN>

E’ 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
    A=A+3              30 A=A+3
    D=D-3              40 D-D-3
  UNTIL A+D<12         50 IF NOT(A+D<12)THEN 30
END.                   60 END
```

5) **WHILE <BOOLEAN> 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 IF NOT(P<>A-Z)THEN 70
   BEGIN               40 D=D*F
     D=D*F             50 A=A-E
     A=A-E             60 GOTO 20
   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    40 RESTORE: DATA 2,4,5,-1,0:
   BEGIN                  FOR II=1 TO 5:READ J
     S=SD-J            40 S=SD-J
     PRINT J+3;J-3     50 PRINT J+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. E’ 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 GOSUB 90
    GOSUB *ORNELLA*    60 GOTO 30
    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) E’ individuata l'istruzione da tradurre (le linee non contenenti parole-chiave del BASAL non sono modificate).
2) E’ 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) E’ 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$(2)="FOR I=1 TO 10"; A$(3)="H=H+I"; A$ (4)="END". La fase iniziale della 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 (line ramo else)", quindi: A$(2)=A$(2) + "GOTO" + STR$ ((J+1)*10) dove, in questo 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
```

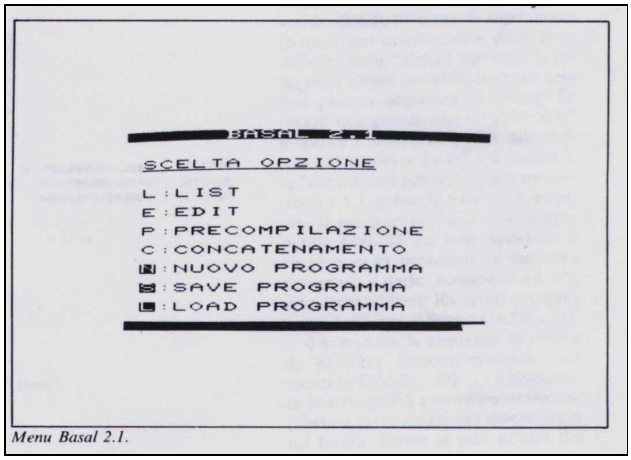
Che è per l'appunto il programma BASIC corrispondente al programma BASAL da cui siamo partiti. Tutto qui.

BASAL 2.1: note al programma

Buona parte del listato basic presentato nelle pagine precedenti funge da sistema operativo per il BASAL. E' infatti possibile registrare programmi su nastro (o disco), 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 posta 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 (3,5) si dovrà digitare A%=MX (3|5). E veniamo alle abbreviazioni concesse: la prima è il punto interrogativo che sta per PRINT. Tutte le altre si attivano con il primo carattere delle statement seguito da uno spazio bianco. Così per scrivere CASE G OF si potrà facoltativamente digitare C G OF prima del return di linea.



Menu Basal 2.1.

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-destra], o [CRSR-sinistra] 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 menù 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 questa punta 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 sui 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. II 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 questa 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 questa punta 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, e necessario ogni volta visualizzare anche un GOTO 11410 che, con un secondo [RETURN] nel buffer, permette appunto di continuare.

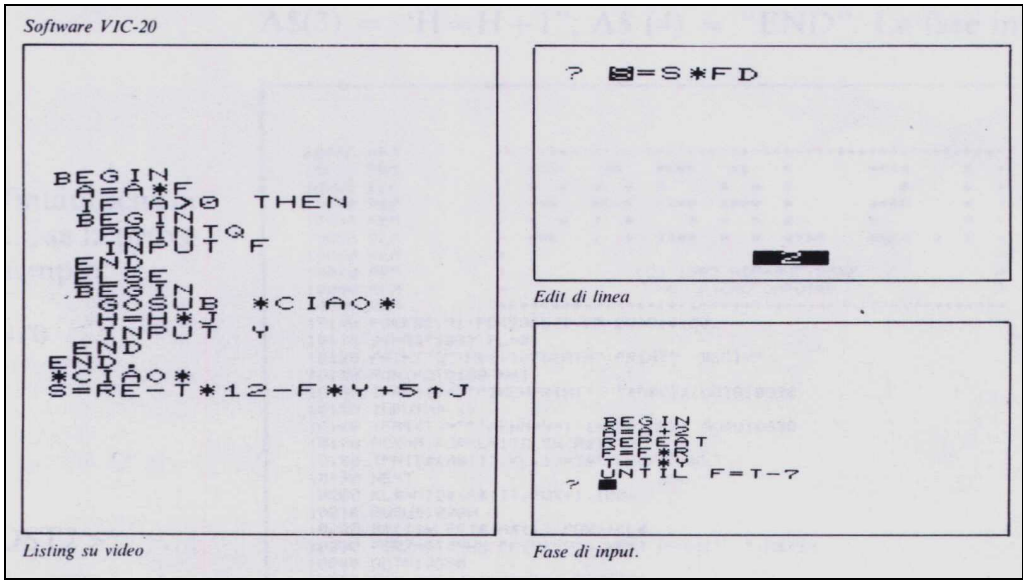
Chi è interessato ai dettagli, faccia riferimento alla routine a partire dalla linea 11190.

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

[(*)] Il top del VIC20 da 16Kb è all'indirizzo 24576 (96*256): con questa istruzione viene così creato un buffer di 1 Kb [8 Kb nella versione per C-64, il cui top è all'indirizzo 40960 (160*256)].

[(**)] Indicata come {WH} nel listato



Conclusione

Vi raccomandiamo a questo punto, se avete intenzione di scrivere qualche programmino in BASAL, di "studiarvi" (si fa per dire!) attentamente quanta 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...

Esempio

Il programma-esempio qui listato simula il lancio di due dadi e la loro relativa visualizzazione sullo schermo. E' 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 "{SC}"
  FOR I=2;12 DO
   BEGIN
     CD$ = ""
     PRINT "{SC}{CD}{CD}{CD}{CD}";
     PRINT "    -------   ------- "
     FOR J=1 TO I
        CD$=CD$+"{CR}"
     A%=INT(RND(1)*6+1)
     CASE A% OF
      BEGIN
       1←A$="323"
       2←A$="232"
       3←A$="222"
       4←A$="121"
       5←A$="121"
       6←A$="111"
      END
      WHILE A$<>"" DO
      BEGIN
       B$=LEFT$(A$,1)
       A$=RIGHT$(A$,LEN(A$)-1)
       CASE VAL(B$) OF
        BEGIN
         1←PRINT CD$;" | o   o |"
         2←PRINT CD$;" |   o   |"
         3←PRINT CD$;" |       |"
        END
     END
  END
  PRINT "{SC}{CD}{CD}{CD}{CD}{CD}{CD}{CD}{CD}{CD}";
  PRINT "    -------   ------- "
  REPEAT
    Z$=INKEY$
  UNTIL Z$<>""
  UNTIL Z$="*"
END.
```

Programma esempio "Dadi"

```
30 PRINT "{SC}"
40 RESTORE: DATA 2,12 :FOR II=1 TO 2:READ I
60 CD$ = ""
70 PRINT "{SC}{CD}{CD}{CD}{CD}";
80 PRINT "    -------   ------- "
90 FOR J=1 TO I
100 CD$=CD$+"{CR}":NEXT
110 A%=INT(RND(1)*6+1)
140 IF A%=1 THEN A$="323":GOTO   210
150 IF A%=2 THEN A$="232":GOTO   210
160 IF A%=3 THEN A$="222":GOTO   210
170 IF A%=4 THEN A$="121":GOTO   210
180 IF A%=5 THEN A$="121":GOTO   210
190 IF A%=6 THEN A$="111":GOTO   210
210 IF NOT( A$<>"" ) THEN  320
230 B$=LEFT$(A$,1)
240 A$=RIGHT$(A$,LEN(A$)-1)
270 IF VAL(B$)=1 THEN PRINT CD$;" | o   o |":GOTO
310
280 IF VAL(B$)=2 THEN PRINT CD$;" |   o   |":GOTO
310
290 IF VAL(B$)=3 THEN PRINT CD$;" |       |":GOTO
310
310 GOTO 210
320 NEXT
330 PRINT "{SC}{CD}{CD}{CD}{CD}{CD}{CD}{CD}{CD}{CD}";
340 PRINT "    -------   ------- "
360 GET Z$
370 IF NOT( Z$<>"") THEN 360
380 IF NOT( Z$="*") THEN 30
390 END
```

Traduzione Basic del programma "Dadi"

I due dadi sono visualizzati 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 assegniamo 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 casuale compreso tra 1 e 6, con il primo CASE... OF ... è assegnata la sequenza di strisce da disegnare. Successivamente, (col secondo CASE ... OF ...), è analizzata la stringa A$ e conseguentemente plottate 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!?!

Listato del programma

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 sui disco un file sequenziale atto ad ospitare il programma per future riletture.

```
11090 INPUT"{SC}NOME FILE"; LJ$:
LJ$="@0:"+LJ$+",S,W"
11095 OPEN 2,8,2,LJ$

11140 CLR: INPUT"{SC}NOME FILE";LJ$: LJ$=LJ$+",S,R"
11145 OPEN 2,8,2,LJ$
```

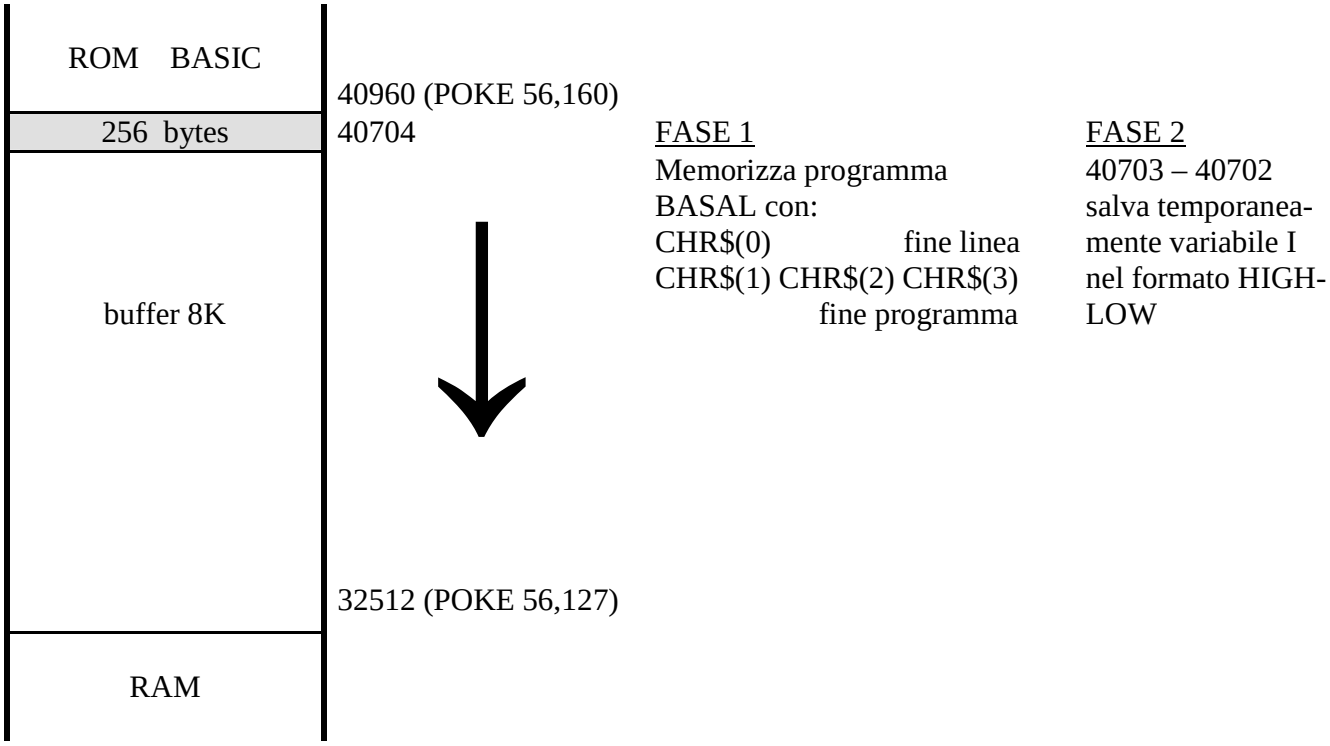
```
10000 REM     ++++++++++++++++++++++++++++++++++++++++++++++++++
10010 REM     + ***    **   ****   **   *        ****     *    +
10020 REM     + *  *  *  *  *     *  *  *          *     *    +
10030 REM     + ***   ****  ****  ****  *        ****     *    +
10040 REM     + *  *  *  *     *  *  *  *        *        *    +
10050 REM     + ***   *  *  ****  *  *  ****     ****  *  *    +
10060 REM     +                                                +
10070 REM     +            (C) 1983 CAIO-SOFTWARE              +
10080 REM     +              MC  MICROCOMPUTER                 +
10090 REM     ++++++++++++++++++++++++++++++++++++++++++++++++++
10100 POKE 56, 127: POKE 53281, 1: GOTO 10430: REM 56,91 E 36879,29 PER VIC 16K
10110 DIM A$(180): FL=0
10120 PRINT "{SC}": A$(1)="BEGIN": PRINT " BEGIN"
10130 FOR I=2 TO 180: N=I
10140    IF A$(I)<>"" THEN PRINT "  "+A$(I): GOTO 10240
10150    INPUT A$(I)
10160    IF A$(I)="" THEN N=N-1: GOTO 10630
10170    PO%=0: FOR KL=1 TO LEN(A$(I))
10180       IF MID$(A$(I),KL,1)="←" THEN PO%=KL
10190    NEXT
10200    KL$=MID$(A$(I),PO%+1,100)
10210    GOSUB 10320
10220    A$(I)=LEFT$(A$(I),PO%)+KL$
10230    FOR Z=0 TO HH%: PRINT "{CU}"; : NEXT: PRINT " "; A$(I)
10240    IF A$(I)<>"E" THEN 10280
10250    A$(I)="": L=I-1: GOSUB 10840: PRINT "{SC}"
10260    FOR K=1 TO N-1: PRINT "  "; A$(K): NEXT
10270    GOTO 10150
10280 NEXT I: GOTO 10540
10290 REM "***************************
10300 REM "  ROUTINE ABBREVIAZIONI   *
10310 REM "***************************
10320 HH%=(LEN(KL$)+2)/40: REM 22 PER VIC
10330 IF LEFT$(KL$,2)="C " THEN KL$="CASE"+RIGHT$(KL$,LEN(KL$)-1)
10340 IF LEFT$(KL$,2)="G " THEN KL$="GOSUB"+RIGHT$(KL$,LEN(KL$)-1)
10350 IF LEFT$(KL$,2)="I " THEN KL$="INPUT"+RIGHT$(KL$,LEN(KL$)-1)
10360 IF KL$="B" OR RIGHT$(KL$, 2)="\B" THEN KL$=KL$+"EGIN"
10370 IF LEFT$(KL$,2)="U " THEN KL$="UNTIL"+RIGHT$(KL$,LEN(KL$)-1)
10380 IF LEFT$(KL$,2)="W " THEN KL$="WHILE"+RIGHT$(KL$,LEN(KL$)-1)
10390 IF LEFT$(KL$,2)="IF" AND RIGHT$(KL$,1)="T" THEN KL$=KL$+"HEN"
10400 IF KL$="R" THEN KL$="REPEAT"
10410 IF LEFTS(KL$,1)="?" THEN KL$="PRINT"+RIGHT$(KL$,LEN(KL$)-1)
10420 RETURN
10427 REM "***************************
10428 REM "  ROUTINE ABBREVIAZIONI   *
10429 REM "***************************
10430 PRINT "{SC}{CD}{RV}     BASAL. 2.1      {BL}"
10440 PRINT "{CD} SCELTA OPZIONE"
10450 PRINT " --------------    "
10460 PRINT "{CD} L LIST SU VIDEO   "
10470 PRINT "{CD} E EDIT DI LINEA   "
10480 PRINT "{CD} P PRECOMPILAZI0NE "
10490 PRINT "{CD} C CONCATENAMENTO  "
10500 PRINT "{CD} {RV}N{RO} NUOVO PROGRAMMA "
10510 PRINT "{CD} {RV}S{RO} SAVE PROGRAMMA  "
10520 PRINT "{CD} {RV}L{RO} LOAD PROGRAMMA  "
10530 PRINT "{CD}{RV} (C) CAIO SOFTWARE       {BL}";
10540 GET Z$: IF Z$="" THEN 10540
10550 IF Z$="E" THEN GOSUB 10820: GOTO 10430
10560 IF Z$="C" THEN 10120
10570 IF Z$="n" THEN CLR: GOTO 10110
10580 IF Z$="P" THEN PRINT "{SC}": GOSUB 11460: GOTO 11230
10590 IF Z$="s" THEN PRINT "{SC}": GOTO 11090
10600 IF Z$="|" THEN GH=80: OPEN 1,4: CMD 1:GOTO 10640
10610 IF Z$="l" THEN PRINT "{SC}": GOTO 11140
10620 IF Z$<>"L" THEN 10430
10630 GH=40: REM 22 PER VIC20
10640 PRINT "{SC}": H=0: FOR I=1 TO N
10650 IF RIGHT$(A$(I-1),5)="BEGIN" THEN H=H+1-(GH>30)
10660 IF A$(I)="END" OR A$(I)="END." THEN H=H-1+(GH>30)
10670 IF LEN(A$(I))+H<=GH THEN 10740
10680 B$=A$(I)
10690 IF B$="" THEN PRINT : GOTO 10750
10700 FOR I=1 TO H: PRINT " "; : NEXT
10710 PRINT LEFT$(B$,GH-H)
10720 B$=MID$(B$,GH+1-H,100)
10730 GOTO 10690
10740 PRINT SPC(H);A$(I)
10750 IF PEEK(653)<>0 THEN 10750
```

```
10760 NEXT
10770 GET Z$: IF Z$="" THEN 10770
10780 CLOSE 1: GOTO 10550
10790 REM *************************
10800 REM * ROUTINE EDIT DI LINEA  *
10810 REM *************************
10820 I=2
10830 L=N
10840 T=I: Q=0
10850 T=T-1
10860 IF T>L THEN T=T-1
10870 IF T<1 THEN T=T+1
10880 PRINT "{SC}{CD}{CR}{CR}{RO}"; A$(T)
10890 PRINT "{HM}"; TAB(244); TAB(229); "{RO}"; T; "{CL} "
10900 GET Z$: IF Z$="" THEN 10900
10910 IF Z$=CHR$(145) THEN 10850
10920 IF Z$="I" THEN 11030
10930 IF Z$="D" THEN 11060
10940 IF Z$="C" THEN 10120
10950 IF Z$=CHR$(17) THEN T=T+1: GOTO 10860
10960 IF Z$=CHR$(13) THEN RETURN
10970 IF Z$ <> CHR$(29) THEN 10900
10980 PRINT "{SC}{CD}{CR}{CR}"; A$(T)
10990 PRINT "{HM}"; TAB(244); TAB(229); "{RV}"; T; "{CL} {HM}"
11000 INPUT KL$: TY=I: I=T: GOSUB 10320: A$(T)=KL$: I=TY: IF Q>1 THEN Q=Q-1: T=T+1: GOTO 10980
11010 Q=0: GOTO 10880
11020 RETURN
11030 Q=Q+1: FOR W=N TO T STEP -1
11040    A$(W+1)=A$(W)
11050 NEXT: A$(T)="": N=N+1: L=L+1: I=I+1: GOTO 10880
11060 FOR W=T TO L
11070   A$(W)=A$(W+1)
11080 NEXT: N=N-1: L=L-1: I=I-1: GOTO 10880
11090 INPUT"{SC}NOME FILE"; LJ$: LJ$="@0:"+LJ$+",S,W": REM NASTRO OPEN 2,1,1,"PROG"
11095 OPEN 2,8,2,LJ$
11100 PRINT #2, N
11110 FOR I=1 TO N
11120    PRINT #2, A$(I)
11130 NEXT: CLOSE 2: GOTO 10430
11140 CLR: INPUT"{SC}NOME FILE";LJ$: LJ$=LJ$+",S,R": REM NASTRO OPEN 2,1,0,"PROG"
11145 OPEN 2,8,2,LJ$
11150 INPUT #2, N: DIM A$(180)
11160 FOR I=1 TO N
11170    INPUT #2, A$(I)
11180 NEXT: CLOSE 2: G0TO 10430
11190 REM *******************************************
11200 REM * QUESTA ROUTINE  TRASFERISCE AUTOMATICAMENTE *
11210 REM * IL PROGRAMMA GENERATO NELLA MEMORIA DEL VIC *
11220 REM *******************************************
11230 S=40704: Z=0: PRINT "{SC}": FOR I=1 TO N: REM 24320 PER VIC 16K
11240    FOR K=1 TO LEN(A$(I))
11250       PO=ASC(MID$(A$(I), K, 1))
11260       IF PO=34 THEN Z=1-Z
11270       IF (PO=32 OR PO=160) AND Z=0 THEN 11290
11280       S=S-1: POKE S,PO
11290    NEXT
11300    S=S-1: POKE S,0
11310 NEXT: POKE 56,S/256-2: POKE S-1,1: POKE S-2,2: POKE S-3,3
11320 I=40704: REM 24320 PER VIC 16K
11330 I=I-1
11340 IF PEEK(I)<>0 THEN A$=A$+CHR$(PEEK(I)): GOTO 11330
11350 IF PEEK(I-1)=1 AND PEEK(I-2)=2 AND PEEK(I-3)=3 THEN PRINT "{SC}{CD}{CD}{CD}"; A$: PRINT "{BL}RUN{HM}":
GOTO 11400
11360 POKE 40703,I/256: REM 24319 PER VIC 16K
11370 POKE 40702,(I/256-PEEK(40703))*256: REM 24318 PER VIC 16K
11380 PRINT "{SC}{CD}{CD}{CD}{WH}"; A$: A$=""
11390 PRINT "GOTO 11410{HM}"
11400 POKE 198, 10: POKE 631, 13: POKE 632, 13: END
11410 I=PEEK(40703)*256+PEEK(40702): GOTO 11330: REM 24318 E 24319 PER VIC 16K
11420 REM *********************************
11430 REM * TRADUZIONE DA BASAL A BASIC DEL *
11440 REM * PROGRAMMA CONTENUTO IN A$(N)    *
11450 REM *********************************
11460 FOR I=2 TO N
11470    FOR H=1 TO LEN(A$(I))
11480       IF MID$(A$(I),H,1)<>"|" THEN 11500
11490       A$(I)=LEFT$(A$(I),H-1)+","+MID$(A$(I),H+1,100)
11500 NEXT: NEXT
11510 FOR I=2 TO N
```

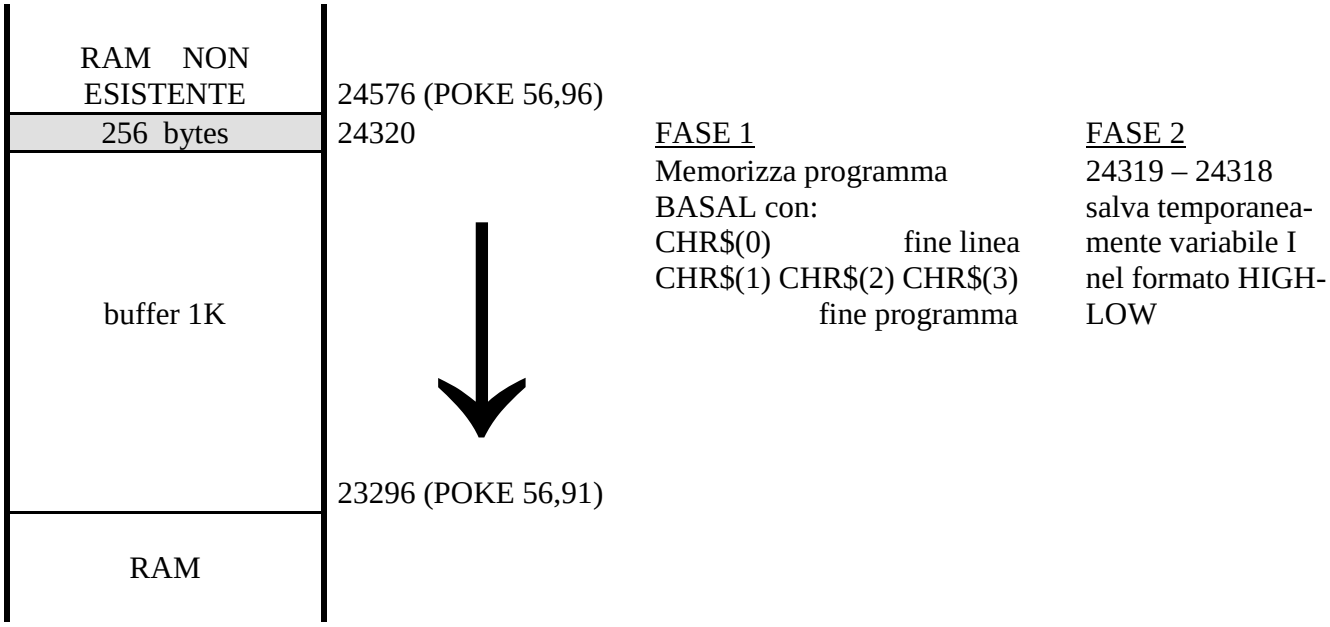
```
11520    IF LEFT$(A$(I),2)="IF" THEN 12410
11530    IF LEFT$(A$(I),3)="FOR" THEN 12610
11540    IF LEFT$(A$(I),2)="GO" THEN 12150
11550    IF LEFT$(A$(I),5)="UNTIL" THEN 12060
11560    IF LEFT$(A$(I),5)="WHILE" THEN 12270
11570 NEXT
11580 FOR I=2 TO N
11590    IF LEFT$(A$(I),4)="CASE" THEN 11800
11600 NEXT
11610 FOR I=2 TO N
11620    IF LEFT$(A$(I),1)="*" THEN 12740
11630 NEXT
11640 FOR I=2 TO N
11650    IF LEFT$(A$(I),2)="IF" OR LEFTS(A$(I),2)="GO" THEN 12780
11660    IF LEFT$(A$(I),1)="@" THEN A$(I)=RIGHT$(A$(I),LEN(A$(I))-1): GOTO 12780
11670    IF LEFT$(A$(I),1)="#" THEN A$(I)=RIGHT$(A$(I),LEN(A$(I))-1): GOTO 11680
11680 NEXT
11690 PRINT "{SC}": FOR I=1 TO N
11700    IF A$(I)="BEGIN" OR A$(I)="END" THEN A$(I)=""
11710    IF A$(I)="END." THEN A$(I)="END"
11720    A$(I)=STR$(I*10)+A$(I)
11730    IF LEN(A$(I))>4 THEN PRINT A$(I)
11740 NEXT
11750 IF PEEK(197)=64 THEN 11750
11760 RETURN
11770 REM ************************
11780 REM *  PREC. CASE..OF      *
11790 REM ************************
11800 H=4
11810 H=H+1
11820 IF MID$(A$(I),H,2)<>"OF" THEN 11810
11830 B$=MID$(A$(I),5,H-5)
11840 GOSUB 12880: A$(I)="": A$(I+1)="": A$(J)=""
11850 J$=STR$((J+1)*10)
11860 FOR T=I+2 TO J-1
11870    H=1: IF A$(T)="OT←BEGIN" THEN K=1: D=I: I=T: GOSUB 12990: I=D: A$(T)="": A$(J)="": GOTO 11990
11880    H=H+1
11890    IF MID$(A$(T),H,1)<>"←" THEN 11880
11900    H=H-1: K=LEN(A$(T))-H-1
11910    L$=LEFT$(A$(T),H): R$=RIGHT$(A$(T), K)
11920    IF L$="OT" THEN A$(T)=R$: GOTO 11950
11930    IF R$="BEGIN" THEN 11960
11940    A$(T)="IF"+B$+"="+L$+ "THEN"+R$+":GOTO "+J$
11950 NEXT: GOTO 11600
11960 D=I: I=T: K=1: GOSUB 12890: 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 GOTO 11600
12030 REM ************************
12040 REM * PREC. REPEAT. .UNTIL *
12050 REM ************************
12060 J=I
12070 J=J-1
12080 IF A$(J)<>"REPEAT" THEN 12070
12090 A$(J)=""
12100 A$(I)="IF NOT("+RIGHT$(A$(I),LEN(A$(I))-5)+") THEN "+STR$((J+1)*10)
12110 GOTO 11570
12120 REM *************************
12130 REM * PREC. SALTI INCOND.   *
12140 REM *************************
12150 H=0
12160 H=H+1: IF MID$(A$(I),H,1)<>"*" THEN 12160
12170 H=H-1: K=LEN(A$(I))-H
12180 B$=RIGHT$(A$(I), K)
12190 M=1
12200 M=M+1: IF LEFT$(A$(M),K)<>B$ THEN 12200
12210 A$(I)=LEFT$(A$(I),H)+STR$((M+1)*10)
12220 IF LEFT$(A$(I),4)="GOTO" THEN A$(M)=A$(M)+"%"
12230 GOTO 11570
12240 REM *********************
12250 REM * PREC. WHILE...DO.. *
12260 REM *********************
12270 B$=MID$(A$(I),6,LEN(A$(I))-7)
12280 A$(I)="IF NOT("+B$+ ") THEN"
12290 IF A$(I+1)="BEGIN" THEN 12330
12300 A$(I)=A$(I)+STR$((I+2)*10)
```

```
12310 A$(I+1)="@"+A$(I+1)+":GOTO"+STR$(I*10)
12320 GOTO 11570
12330 GOSUB 12880
12340 A$(I)=A$(I)+STR$((J+1)*10)
12350 A$(J+1)=""
12360 A$(J)="@GOTO"+STR$(I*10)
12370 GOTO 11570
12380 REM **********************
12390 REM * PRECOMPILAZIONE IF *
12400 REM **********************
12410 IF A$(I+1)="BEGIN" THEN 12490
12420 A$(I)=A$(I)+A$(I+1)
12430 A$(I+1)=""
12440 IF A$(I+2)<>"ELSE" THEN A$(I)="#"+A$(I): GOTO 11570
12450 D=I:I=I+2: GOSUB 12890: I=D
12460 A$(I)=A$(I) + ":GOTO"+STR$((J+1)*10)
12470 A$(I+2)=""
12480 I=I+2: G0TO 11570
12490 A$(I)=A$(I)+STR$((I+2)*10)
12500 GOSUB 12880
12510 A$(I+1)="@GOTO"+STR$((J+1)*10)
12520 A$(J)=""
12530 IF A$(J+1)<>"ELSE" THEN 11570
12540 H=J: D=I: I=J+1
12550 GOSUB 12880: I=D
12560 A$(H)="@GOTO"+STR$((J+1)*10)
12570 A$(H+1)="": GOTO 11570
12580 REM ************************
12590 REM * PRECOMPILAZIONE FOR  *
12600 REM ************************
12610 H=0
12620 IF RIGHT$(A$(I),2)="DO" THEN 12980
12630 IF A$(I+1)="BEGIN" THEN 12670
12640 IF LEFT$(A$(I+1),3)="FOR" THEN I=I+1: H=H+1: GOTO 12620
12650 A$(I+1)=A$(I+1)+":NEXT"
12660 J=I+1: GOTO 12700
12670 GOSUB 12880
12680 A$(I+1)=""
12690 A$(J)="NEXT"
12700 IF H=0 THEN 11570
12710 FOR T=1 TO H
12720    A$(J)=A$(J)+":NEXT"
12730 NEXT: GOTO 11570
12740 IF RIGHT$(A$(I),1)="%" THEN A$(I)="": GOTO 11630
12750 A$(I)=""
12760 IF A$(I+1)<>"BEGIN" THEN A$(I+1)=A$(I+1)+":RETURN": GOTO 11630
12770 GOSUB 12880: A$(I+1)="": A$(J)="RETURN": GOTO 11630
12780 L=LEN(A$(I))-1
12790 L=L-1
12800 IF MID$(A$(I),L,1)<>" " THEN 12790
12810 S=VAL(RIGHT$(A$(I),LEN(A$(I))-L))-10
12820 S=S+10: IF A$(S/10)="" OR A$(S/10)="END" OR A$(S/10)="BEGIN" THEN 12820
12830 A$(I)=LEFT$(A$(I),L)+STR$(S): GOTO 11680
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 IF RIGHT$(A$(J), 5)="BEGIN" THEN K=K+1: GOTO 12900
12920 IF A$(J)="END" THEN K=K-1: KK=1
12930 IF K<>0 THEN 12900
12940 RETURN
12950 REM ***************************
12960 REM *  PREC. FOR I=.,;..;..DO  *
12970 REM ***************************
12980 K=0: A$="RESTORE: DATA": T=0
12990 K=K+1: IF MID$(A$(I), K, 1) <> "=" THEN 12990
13000 FOR R=K+1 TO LEN(A$(I))-2
13010    IF MID$(A$(I),R,1)=";" THEN A$=A$+",": T=T+1: GOTO 13030
13020    A$=A$+MID$(A$(I),R,1)
13030 NEXT
13040 B$=MID$(A$(I),4,K-4)
13050 A$(I)=A$+":FOR II=1 TO"+STR$(T+1)+":READ"+B$
13060 A$="": GOTO 12620
```

COMMODORE 64



VIC20 – 16K



- FASE 1 → da array A$(.) a buffer
- FASE 2 → ba buffer a memoria per l'esecuzione