L'AmigaBASIC

Esaurito l'argomento AmigaDOS, a partire da questo numero Amighevole tratterà del linguaggio fornito con l'Amiga, l'AmigaBasic. Come avremo modo di scoprire, non si tratta solo di un linguaggetto di programmazione «giusto per fare qualcosa», ma di un vero e proprio strumento per cominciare a sfruttare le caratteristiche della macchina. Come per magia, questo Basic non è nemmeno tanto «Basic», non nel senso che è difficile da usare, ma intendendo con questo che permette uno stile di programmazione più consono ai canoni informatici degli ultimi 20 anni... Programmazione strutturata, procedure con passaggio dei parametri per nome e per valore, variabili locali e globali, gestione delle interruzioni da mouse... E tante altre feature che avremo modo di scoprire nel corso di questa serie di articoli.

di Andrea de Prisco

Progammazione pulita

Se da una parte il bello del... pollo è il pollo, dall'altra diremo che il bello di AmigaBasic è che... non si tratta di Basic. Spieghiamoci meglio: il linguaggio di programmazione Basic ha di brutto che sono necessari i numeri di linea, non esistono forme di strutturazione, le variabili sono sempre e comunque globali a tutto il programma, niente procedure definibili dall'utente, niente ricorsione niente... tant'altro. Ovvero il linguaggio Basic è proprio brutto. Non per niente ha molte affinità col Fortran, altro esemplare paleoinformatico purtroppo non ancora in via di estinzione. Per quanto qualcuno potrà non essere troppo d'accordo col sottoscritto resta universalmente riconosciuto che un linguaggio più si avvicina al linguaggio macchina più è difficile da usare, mentre più è ad alto livello, più si avvicina al modo di ragionare «umano» ovvero senza GOTO, numeri linea ecc. ecc.

Così dopo l'assembler, Fortran e Cobol, come progresso insegna, qualcuno ben pensò di dare vita ai cosiddetti linguaggi di programmazione algoritmici, capostipite l'Algol, pensati proprio per semplificare la scrittura dei programmi, una volta deciso l'algoritmo da implementare.

Col Basic un penoso passo indietro... senza contare che da linguaggetto per insegnare a programmare (male) qual era stato ideato, a causa di chissà quale altro folle è diventato il linguaggio di programmazione più diffuso al mondo. Così chi ha imparato col Basic «per definizione» non sa programmare e quando si trova davanti un nuovo linguaggio, più serio, non è più in grado di cavarsela.

L'AmigaBasic in tal senso si trova a metà strada tra l'orribile Basic e un più informatico Pascal o Algol. Se sei proprio scemo puoi usarlo come un Basic normale, se sei intelligente puoi divertirti a sfruttare appieno la programmazione strutturata messa a disposizione dal super interprete che supporta l'AmigaBasic. Inutile dirvi che nel corso di questi articoli non faremo mai un solo riferimento al lato brutto, ma solo a quello buono. Nei nostri esempi non useremo mai goto, né numeri linea... almeno finché l'AmigaBasic non ce lo imporrà: in questo senso sarà una sorta di banco di prova per il linguaggio.

Descrizione esterna

Prima di entrare nel merito (a proposito, non ve l'ho ancora detto, questo mese parleremo di costrutti iterativi, condizionali, più altre cose) diamo uno sguardo alle caratteristiche generali dell'AmigaBasic. Come già preannunciato, si tratta di un linguaggio di programmazione interpretato la cui caratteristica principale è quella di poter sfruttare praticamente tutte le risorse della macchina: grafica, suono, mouse, finestre, sprite, ecc.

Manca purtroppo la possibilità di multitasking all'interno dell'interprete, leggi: se vuoi lanciare contemporaneamente due programmi Basic occorre caricare due volte l'interprete e mandare in esecuzione un programma su ogni interprete... memoria permettendo.

L'interprete, non l'abbiamo ancora detto, è lungo circa 90 K (100 la release 1.2) ed è stato scritto dalla Microsoft Corporation. Ciò implica che è abbastanza compatibile con altre versioni per altre macchine, non ultimo il

1

```
10 IF A=0 THEN 40
20 PRINT "E' DIVERSO DA ZERO"
30 GOTO 50
40 PRINT "E' UGUALE A ZERO"
50 END
```

2

```
100 IF A=0 THEN PRINT "E' UGUALE A ZERO":GOTO 120
110 PRINT "E' DIVERSO DA ZERO"
120 END
```

3

```
200 IF A>B THEN 220
210 T=A:A=B:B=T
220 RETURN
```

Figura 1: Esempio di programma scritto in Basic standard.

Figura 2: Con la possibilità di inserire istruzioni di seguito al THEN, la leggibilità migliora molto.

Figura 3: Subroutine usata dal programma di figura 4.

```
100 A=X:B=Y
110 GOSUB 200
120 B=Z
130 GOSUB 200
140 B=T
150 GOSUB 200
160 END
```

Figura 4: Eseguendo queste linee Basic, A conterrà il maggiore dei valori contenuti in X, Y, Z e T.

```
100 T=0
110 FOR I=1 TO 100
120 T=T+I
130 PRINT I,T
140 NEXT
```

Figura 5: Esempio di utilizzo del costrutto FOR-NEXT.

```
IF Exp THEN LC1 ELSE LC2        (a)

IF Exp THEN
  LC1                           (b)
ELSE
  LC2
END IF
```

Figura 6: Sintassi del costrutto IF-THEN-ELSE. (a) ad una sola linea, (b) multilinea.

BasicA della grande famiglia MS-Dos.

Si tratta di un Basic abbastanza veloce, specialmente quando si tratta di manipolare oggetti grafici grazie al coprocessore interno alla macchina: certo un bel compilatore non guasterebbe affatto, soprattutto considerato che in questo modo non occorre portarsi dietro l'interprete ma otterremmo direttamente codice eseguibile dal processore. Come dire: multitasking a più non posso.

Tra le cose non troppo positive annoveriamo un penoso editor, col quale scrivere un programma, a causa di una diffusa lentezza operativa, diventa abbastanza scocciante. Grazie però (come sempre) al multitasking di Amiga possiamo caricare un qualsiasi altro editor a nostra scelta (magari anche un word processor) e scrivere su questo i nostri programmi. Ovviamente prima di dare RUN al programma occorrerà salvare il testo dal WP e caricarlo dall'AmigaBasic. Il tutto è sì molto più lungo, ma con la lentezza dell'editor del Basic si può anche pareggiare.

Come abbiamo più volte ripetuto, da Basic è possibile l'interazione col mouse per dare comandi, manipolare finestre, scegliere dai menu a discesa definibili dall'utente. Oltre all'interazione, per così dire, normale è possibile l'interazione (ancora per così dire) forzata. Tramite il meccanismo delle interruzioni, è possibile indicare in quale punto del programma bisogna saltare in seguito alla pressione dei tasti del mouse, indipendentemente da quello che stava facendo il programma in quel momento. Molto bello.

Per quanto riguarda l'aritmetica, segnaliamo la possibilità di definire variabili in singola o doppia precisione, nonché interi lunghi e corti. Oltre, naturalmente, alle variabili di tipo stringa.

Per concludere, la possibilità di creare e manipolare file di tipo sequenziale e ad accesso casuale, ai quali dedicheremo un intero articolo.

Basic in generale

Giusto per completezza, qualora vi fosse qualche lettore che non ha mai programmato in tale linguaggio, daremo qui di seguito qualche altra informazione riguardo al Basic. Quello brutto, non l'AmigaBasic.

```
IF A=0 THEN PRINT "Uguale a 0" ELSE PRINT "Diverso da 0"
                                        (a)

IF A=0 THEN
  PRINT "Uguale a 0"
ELSE                                    (b)
  PRINT "Diverso da 0"
ENDIF
```

Figura 7: Programma di figura 1 scritto in AmigaBasic. (a) con sintassi unilinea e (b) multilinea del costrutto IF.

Niente strutturazione ma semplici istruzioni tutte etichettate dal cosiddetto numero linea. Possiamo utilizzare variabili numeriche, variabili di tipo stringa, nonché le matrici formate da elementi di questo tipo. Troviamo istruzioni per eseguire salti condizionati (IF-THEN) o incondizionati (GOTO), subroutine (GOSUB), più le interazioni FOR-NEXT per eseguire un pezzo di codice un certo numero di volte e/o per determinati valori di un indice.

In figura 1 è mostrato un segmento di programma Basic-standard che stampa la frase «uguale a zero» o «diverso da zero» a seconda del valore contenuto nella variabile A. Si noti come l'uso di salti e controsalti renda particolarmente ardua la comprensione anche di un così semplice programmino. Un lieve miglioramento c'è stato quando si pensò di aggiungere la possibilità di indicare dopo il THEN dell'IF invece che il numero linea di una nuova istruzione, l'istruzione o le istruzioni da eseguire. Il risultato è mostrato in figura 2, è evidente il miglioramento di comprensibilità dello stesso algoritmo.

La possibilità di definire subroutine, proprio come in linguaggio macchina e nulla di più, permette di risparmiare codice quando alcune porzioni di questo sono ripetute più volte in uno stesso programma. Definendo una subroutine a partire da una certa linea, terminante con l'istruzione RETURN, possiamo invocarla ogni volta che ci serve, semplicemente con l'istruzione GOSUB. In figura 3 è mostrata una semplicissima subroutine che testando il valore di due variabili A e B restituisce i medesimi valori ponendo in A il più grande e in B il più piccolo: la variabile T è adoperata come temporanea per permettere lo scambio di contenuti. In figura 4 una banale applicazione di tale routine: in X, Y, Z e T ci sono quattro valori qualsiasi, dopo l'esecuzione di questo programmino A conterrà il più grande dei quattro.

Per concludere la nostra mini carrellata sul Basic, in figura 5 è mostrato un utilizzo del ciclo FOR-NEXT. Le istruzioni contenute tra il FOR e il NEXT (linee 120-130) sono eseguite

```
8   IF Exp1 THEN
      LC1
    ELSEIF Exp2 THEN
      LC2
    ELSEIF Exp3 THEN
      LC3
      .
      .
      .
    ELSE
      LCn+1
    ENDIF

10  WHILE Exp
      LC
    WHEND
```

```
9   FOR Var=Start TO Stop STEP Step
      LC
    NEXT

11  A=X:B=Y
    GOSUB OrdinaAeB
    B=Z
    GOSUB OrdinaAeB
    B=T
    GOSUB OrdinaAeB
    END

    OrdinaAeB:
      IF A>B THEN T=A:A=B:B=T
      RETURN
```

Figura 8: Sintassi della estensione ELSEIF.

Figura 9: Sintassi del costrutto FOR-NEXT in AmigaBasic.

Figura 10: Sintassi del costrutto WHILE-WEND dell'AmigaBasic.

Figura 11: Programma di figura 3 e 4 in versione AmigaBasic.

«per tutti i valori di I compresi tra 1 e 100» (libera traduzione di FOR I=1 TO 100). Se non è indicato alcuno step (passo di incremento) è assunto pari a 1, è possibile indicare step negativi e, generalmente, anche frazionari. Per la cronaca, il programmino in figura 5 stampa i primi 100 numeri e la somma di tutti i numeri compresi tra 1 e il numero testé stampato.

L'AmigaBasic

La più grande differenza esistente tra il Basic standard e l'AmigaBasic, è certamente il fatto che quest'ultimo non necessita della numerazione delle linee di programma. A causa di ciò, vengono messi a disposizione del programmatore altri strumenti per esprimere ugualmente, anzi meglio, qualsiasi algoritmo senza la necessità di numerare le linee. Prima però di iniziare, occorre avvertire che sebbene spariscono i numerini, ahimé, resta ben radicato anche nell'AmigaBasic il concetto di linea. Alcune cose, ad esempio, possono esser fatte solo messe sulla stessa linea. Per altre è necessario adoperare linee diverse (ovvero separate da un return) dunque giudichiamo la sintassi ancora un po' vecchia maniera (fra l'altro potrebbero anche sorgere problemi trasferendo programmi con altre macchine).

Ad esempio il Pascal utilizza come separatore tra le istruzioni il «punto e virgola» (che è un carattere visibile) mentre i vari return e/o spazi servono solo per visualizzare il listato con una certa forma e sono completamente ignorati dal compilatore.

Chiariremo meglio quanto detto illustrandovi il primo costrutto: l'IF-THEN-ELSE. Esistono due forme: ad una o più linee (!). In figura 6 è mostrata la sintassi: per EXP si intende una espressione logica, LC sta per lista comandi, un insieme più o meno esteso di istruzioni Basic, quindi LC1 ed LC2 sono due liste comandi. Se l'espressione logica è vera sarà eseguita LC1, se è falsa sarà eseguita LC2. A proposito di liste comandi, ricordiamo che se questi ultimi (in generale) sono posti su linee diverse occorre separarli da un return, se si trovano sulla stessa linea sono separati dai «due punti». In figura 6a e 6b, occorre rispettare gli accapo quindi battere return solo quando la sintassi lo impone. La differenza tra le due sintassi è che nel primo caso LC1 ed LC2 più l'espressione logica e le parole chiave non possono superare la lunghezza di una linea Basic, nel secondo caso LC1 ed LC2 possono essere lunghe quanto vogliamo (fino ad esaurimento della memoria). Ancora, in tutti e due i casi sia l'ELSE che LC2 sono facoltativi, ovvero non deve esser eseguito alcunché quando la condizione logica dà come risultato «falso».

Passiamo agli esempi. In figura 7a è mostrato il programmino di figura 1 e 2 in versione AmigaBasic secondo la sintassi unilinea di figura 6a. Come vedete dei numeri linea se ne può fare completamente a meno. Oltre al fatto che se ne guadagna in chiarezza: traducendo alla lettera quella linea, non da AmigaBasic ma dall'inglese parlato all'italiano leggiamo «se A è uguale a 0 allora stampa uguale a zero altrimenti stampa diverso da zero» che è esattamente quello che vogliamo. Dall'algoritmo (in questo caso banale, però...) al programma senza cambiare una virgola.

Per quanto riguarda la sintassi multilinea (6b), lo stesso programmino è mostrato in figura 7b. Si noti che la parola chiave END IF (attenzione, ci vuole lo spazio in mezzo) è obbligatoria dato che serve per delimitare le istruzioni da eseguire o da saltare a seconda del valore dell'espressione logica.

ELSEIF

Esiste una variante del comando IF ideata per concatenare tra loro vari comandi condizionali. Premettiamo che se ne sconsiglia l'uso non tanto per problemi tecnici quanto per non degradare lo stile di programmazione che il comando IF-THEN-ELSE ci permette di ottenere. Dal momento che la LC2 della clausola ELSE può benissimo contenere altri IF, non si capisce perché abbiano voluto aggiungere altra carne al fuoco che può solo creare un po' di confusione e nient'altro. Comunque, dal momento che per alcuni versi il costrutto potrebbe anche essere accettabile, diamo qui di seguito sintassi e semantica informale.

Immaginiamo di avere una certa sequenza di liste di comandi, ognuna riferente una determinata condizione logica. Bisogna eseguire la prima lista di comandi per la quale la corrispondente condizione logica è vera, e saltare tutte le altre. Indipendentemente dal fatto che altre condizioni logiche seguenti possano essere ugualmente verificate. Se nessuna condizione logica è verificata, eseguiremo una lista comandi «di riserva».

Se vi siete messi le mani nei capelli è quanto di meglio potevate fare. La situazione è così rara, infatti, che non si capisce perché abbiano pensato ad un apposito costrutto.

In figura 8 abbiamo «rappresentato» la sintassi. EXP1...EXPn sono le n espressioni logiche associate alle liste di comandi LC1...LCn. LCn+1 è la lista di comandi da eseguire se nessuna espressione dà come risultato vero.

Iterazioni

Per effettuare iterazioni in AmigaBasic, oltre al già citato (nel Basic standard) ciclo FOR-NEXT (sintassi in figura 9), esiste un comodo WHILE-WEND che permette di ripetere una determinata lista di comandi fino a quando una condizione logica non dà risultato falso. Ovviamente, all'interno del ciclo ci saranno delle istruzioni che prima o poi provocheranno tale evento. A meno che non siamo intenzionati a cicli infiniti: in questo caso sarà sufficiente impostare una condizione logica sempre vera, ad esempio la costante 1 (in AmigaBasic a valori diversi da zero è associato il «vero», allo zero il «falso»).

In figura 10 troviamo la sintassi di tale costrutto. Si valuta l'espressione logica, se vera si esegue la lista di comandi compresa tra il WHILE e il WEND. Se dà esito falso il controllo passa all'istruzione dopo il WEND. Eseguita la lista di comandi (l'espressione era verificata) si torna a valutare EXP comportandoci come prima. Tutto qui.

Label e Subroutine

Dal momento che è possibile evitare la numerazione delle linee, per riferire ad una subroutine si utilizzano le label (o etichette che dir si voglia). Una label è formata da un certo numero di caratteri alfanumerici (il primo deve essere una lettera) e deve terminare con i «duepunti» onde poterle distinguere dai nomi di variabile. Nel riferimento fatto dal comando GOSUB i due punti spariscono, indicando solo il nome. In figura 11 abbiamo riportato la traduzione in AmigaBasic del programma di figura 3 e 4. Fa ancora abbastanza pena... quando parleremo delle procedure e del passaggio dei parametri sarà tutt'altra cosa (!).

```
'programma principale

true=1                          ' costante "vero"
INPUT a                         ' "a" e' il numero
IF a < 0 THEN
  PRINT "e' negativo"
ELSE
  IF a MOD 2 THEN               ' divisibile per 2
    PRINT "e' dispari"
    GOSUB NumeroPrimo           ' e' primo ?
  ELSE
    PRINT "e' pari"
    GOSUB Fattoriale            ' e' un fattoriale ?
  END IF
END IF
END                             ' fine programma


NumeroPrimo:                    ' subroutine

  r=INT(SQR(a))
  Primo=true
  FOR i=2 TO r
    Primo=Primo AND SGN(a MOD i)
  NEXT
  IF Primo THEN PRINT "e' primo"
  RETURN



Fattoriale:                     ' subroutine

  fatt=true
  count=2
  aa=a
  WHILE fatt
    aa=aa/count
    IF aa=1 THEN PRINT "e' il fattoriale di";count
    fatt=(aa-INT(aa)=0)
    count=count+1
  WEND
  RETURN
```

Figura 12: Programma CHECK NUM (vedi testo).

Esempio finale

Più che finale diremmo stupido, ma dato che stupido non si può scrivere nel titoletto di un nuovo paragrafo, preferiamo dirvelo a quattr'occhi.

Scherzi a parte, in figura 12 abbiamo scritto per voi un programmino che utilizza un po' tutte le istruzioni che abbiamo trattato questo mese. Ovviamente, da questo la stupidità di cui sopra, non ha nessuna utilità pratica, tranne quella di svelarvi alcune proprietà (stupide) del numero che darete in ingresso. Queste proprietà sono: negativo, pari, dispari, primo, fattoriale di qualcos'altro.

Come avrete notato il programma è composto da una parte principale e da due subroutine che servono rispettivamente per sapere se il numero in questione è primo o fattoriale. La scrittura «scalata» del testo del programma, è un accorgimento che consigliamo vivamente di adottare per aumentare la leggibilità del listato. Ogni volta che iniziamo una nuova lista di comandi (relativa ad una determinata istruzione) sfasiamo la scrittura di due caratteri. Terminata la lista ci riallineamo col precedente testo. Da notare che l'editor dell'AmigaBasic supporta questo genere di input nelle linee, dato che ogni accapo fa andare il cursore proprio sotto l'inizio della linea precedente. Dando uno sguardo al listato del programma principale possiamo subito individuare i vari ELSE ed ENDIF a quale comando si riferiscono: basta trovare il primo IF allineato con questi. Semplice no?

Per quanto riguarda il lato tecnico, e non quello estetico, del programma vi rimandiamo al manuale dell'AmigaBasic per le varie funzioni adoperate e non commentate in questo testo. L'unica difficoltà potrebbe venir fuori nella comprensione della quart'ultima linea:

fatt = (aa – INT (aa) = 0)

Non si tratta di un vero e proprio trucco: è semplicemente una espressione logica che vale 1 (vero) se il valore della variabile aa è intero, 0 (falso) se contiene anche una parte decimale. Così il WHILE di tre linee prima continua ad eseguire la lista di comandi fino a che aa non diventa un numero frazionario. Fine.