

PROGETTO 7

Scheda progetto	
Nome progetto	Maggiori – minori
Problema	Scrittura di tutti i numeri naturali compresi tra due estremi definiti
Algoritmo creato	Determinazione di tutti i numeri naturali compresi tra due estremi
Teoria di Pascal	L'iterazione enumerativa

Nel quinto progetto di questa guida avevamo iniziato a parlare di iterazione, in inglese loop, una struttura della programmazione utilizzata per ripetere una serie di istruzioni in base al verificarsi o meno di una condizione. La struttura repeat...until è oggi divenuta obsoleta, tanto che nessun linguaggio moderno la utilizza, mentre il ciclo while è ancora molto utilizzato e per la condizione a posteriori si usa un altro tipo di costrutto non presente in Pascal. Il terzo di tipo di iterazione che avevamo accennato è quello di tipo enumerativo, che si usa quando viene stabilito fin dall'inizio il numero di volte per cui sarà ripetuto il blocco di istruzioni.

Analisi del problema

Questa volta dobbiamo ricevere due numeri, stabilire il maggiore e il minore, quindi stampare su schermo tutti gli interi in mezzo a questi due valori. Visto che il nostro programma è solo dimostrativo e non abbiamo nessuna necessità di stampare migliaia di numeri, lavoreremo con variabili di tipo **shortint**. Ecco la nostra tabella.

A	B	Stampa...
0	2	0 1 2
1	3	1 2 3
-2	+5	-2 -1 0 1 2 3 4 5
7	10	7 8 9 10
0	0	0
1	1	1
1	0	0 1
0	1	0 1

Notiamo che non ci sono casi evidenti che dovremo prevedere. Ovviamente dovremo stare attenti ad utilizzare solamente valori shortint o il nostro programma avrà un crash. Ma come essere sicuri che tutto ciò succeda? Qui si pone per la prima volta nei nostri progetti uno dei problemi fondamentali di tutta la programmazione, sia questa in Pascal o in un qualunque altro linguaggio, ossia la **gestione degli errori logici**, chiamata in altri linguaggi più moderni **gestione delle eccezioni**. Ogni linguaggio ne ha una propria con una sua sintassi, ma noi non ci cureremo di questo dettaglio.

La gestione degli errori è una buonissima tecnica di programmazione e, nei programmi evoluti, è meglio utilizzarla (di conseguenza è necessario imparare la sua sintassi), ma noi stiamo creando programmi test, che utilizzeremo solo noi. Pertanto non dobbiamo considerare gli utenti del software dei perfetti dementi, ma saremo noi, in fase di test, a comportarci in modo perfetto, seguendo le nostre regole. Cerchiamo di fare ordine e di capire meglio con un esempio.

Quando creiamo un software da distribuire al pubblico dobbiamo prevedere ogni caso di possibile errore causato dal programma (ad esempio una divisione per zero) e gestire il tutto con apposite sintassi (spesso basta una semplice struttura condizionale). Bisogna sicuramente avvertire l'utente con messaggi appositi prima che commetta l'errore, ma comunque non possiamo lasciare nulla al fato: bisogna pensare che l'utente non sa cosa passa nella testa del programmatore (e spesso non si pone nemmeno questo problema), mentre il programmatore, se si vuole definire veramente tale, **deve** essere in grado di calcolare ogni cosa al millesimo e deve poter prevedere tutti gli errori commessi dall'utente. Questi progetti, tuttavia, servono solo ad illustrare le basi del Pascal, sarebbe inutile ogni volta scrivere listati di centinaia di righe solo per prevedere una potenza zero alla zero: per questo dicevamo prima che ci limiteremo ad essere diligenti. Se il nostro progetto non prevede casi appositi **volutamente**, noi non dobbiamo fare dei danni. In questo caso dovremo imparare ad utilizzare un'iterazione, quindi perché perdere tempo a gestire eccezioni per overflow? Limitiamoci semplicemente, in fase di test, a non usare valori appositi. Cercheremo di seguire spesso questa tecnica, ma è bene ricordare che, quando in futuro ognuno creerà da sé i propri programmi per gli altri, dovrà schematizzare tutti i possibili errori (in un'industria si parla di **CP, critical points**, ossia punti critici).

Ragioniamo ancora un po': davvero non esiste alcun caso da prevedere?

Leggiamo il problema postoci:

"Determinazione di tutti i numeri naturali compresi tra due estremi"

Non è specificato se il primo deve essere maggiore del secondo o viceversa.

Quindi, effettivamente, il nostro programma deve essere in grado di prevedere quando un numero è minore di un altro. Potrebbe essere spontaneo pensare che, in uno dei due casi, effettueremo uno scambio fra le due variabili, oppure si potrebbe anche dire che questo caso sia inutile da prevedere, esattamente come quello precedente. In effetti ciò è vero, ma almeno una selezione sarà inserita nel programma, in modo da non perdere l'abitudine al suo utilizzo e per evidenziare il fatto che la programmazione è composta spesso di grandi annodamenti. Andiamo ora a vedere l'algoritmo.

Creazione dell'algoritmo

Ecco l'algoritmo:

1. Leggi A
2. Leggi B
3. Se $A > B$
 - a. Scambia A e B
 - b. Scrivi tutti i numeri compresi tra A e B
4. Altrimenti
 - a. Scrivi tutti i numeri compresi tra A e B

Solo una piccola annotazione: i passaggi 3B e 4° sono esattamente identici. Vedremo in progetti futuri come questo possa essere un vantaggio per noi, in quando potremo risparmiare di scrivere del codice utilizzando quella che viene chiamata una **funzione** o, in altri casi, una **procedura**. Non preoccupiamocene ora: questa nota era solo per sviluppare la curiosità e il sesto senso del lettore. Nella programmazione osservare è molto importante.

Creazione del programma in Pascal

Ecco il listato del programma:

```
program mag_min;
uses crt;
var a,b,z,i:shortint;
begin
writeln ('Inserisci due numeri interi poco elevati:  ');
readln (a,b);
if (a>b)
then begin
z:=a;
a:=b;
b:=z;
for i:=a to b do
begin
write (i, ' ');
end;
end
else begin
for i:=a to b do
begin
write (i, ' ');
end;
end;
repeat until keypressed;
end.
```

Analisi del programma

Cosa fa questo programma? Semplice, legge due variabili intere non elevate (termine generico in effetti ma più che sufficiente nel nostro campo), le scambia qualora la prima sia maggiore della seconda, e scrive tutti i valori compresi tra le variabili, estremi inclusi. Per realizzare il tutto viene utilizzata una selezione e una coppia di iterazioni identiche. Questa iterazione, detta enumerativa, verrà analizzata nel prossimo paragrafo.

TEORIA: Iterazione enumerativa

Il ciclo enumerativo ha un funzionamento molto semplice: viene presa una variabile intera (nel nostro caso la *i*) e viene utilizzata come contatore per il numero di esecuzioni. Questa variabile subirà ad ogni iterazione un cambiamento identico, ossia verrà incrementata di 1 (grazie all'utilizzo della parola *to*). Ecco la dimostrazione

```
for i:=1 to 5 ...
```

La variabile *i* è inizializzata ad 1, e ad ogni ciclo viene incrementata. Quando raggiunge il valore 5, il ciclo viene eseguito un'ultima volta e poi si prosegue con il resto del listato. E' anche possibile eseguire un conteggio all'indietro, sostituendo alla parola *to* l'altra **downto**. Ecco l'esempio

```
for i:=5 downto 1 ...
```

Il resto dell'iterazione è composto dalla parola chiave *do*, che indica l'inizio del blocco di istruzioni e gli eventuali delimitatori di blocco, ossia la coppia *begin...end*.

CURIOSITA': Il ciclo enumerativo nei nuovi linguaggi

Al giorno d'oggi la struttura enumerativa del Pascal non è più presente in nessun linguaggio. Tuttavia il ciclo *for* esiste ancora. Questo paragrafo è solo una curiosità, che introduce il lettore ad un eventuale studio di altri linguaggi. Questo listato funziona in JavaScript.

```
for (i=0 ; i<5 ; i++)
{
    istruzioni;
}
```

Analizziamo brevemente il blocco: sembra arabo in effetti, ma possiamo osservare uno zero e un 5, il che ci induce a pensare che siamo di fronte ad un ciclo che cresce e notiamo un ++, che ci indica un aumento... Abbiamo ancora le idee confuse, e solo un po' di teoria ci può chiarire le idee:

Ecco lo schema dell'iterazione

```
for (inizializzazione; condizione; cambiamento)
    istruzioni;
```

Con **inizializzazione** intendiamo l'assegnamento di un valore al contatore;

Con **condizione** intendiamo una condizione che deve essere vera perché avvenga il ciclo; notiamo quindi che il ciclo può anche non essere mai eseguito, come nel caso

```
for (i=0; i<0; i++)
```

In Pascal il ciclo enumerativo viene eseguito almeno una volta, negli altri linguaggi no.

Con **cambiamento** indichiamo la variazione che subirà il contatore ad ogni iterazione. Il simbolo ++ è un operatore moderno chiamato **operatore di incremento** ed equivale a scrivere *i = i+1;*

In pratica, la variabile viene incrementata.

Proposte

1. Scrivere la tabellina di un numero inserito da tastiera.

2. Scrivere i numeri pari minori di n
3. Scrivere i primi n multipli di m

Riepilogo

Termini riservati nuovi	
for	Ciclo enumerativo
to	Ciclo crescente
downto	Ciclo decrescente
do	Esegui

Progetto creato da Squall1988

<http://www.pslife.net>

© PiEsseLife - 2003