Il Pascal (continua)

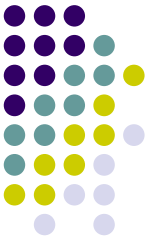
Università degli Studi di Salerno

Corso di Laurea in Scienze della Comunicazione

Informatica generale (matr. Dispari)

Docente: Angela Peduto

A.A. 2005/2006



Selezione multipla

- Nel caso di dover far una scelta tra molte opzioni, ad esempio se vogliamo chiedere ad una persona quali sono i piatti che preferisce tra quelli nel menu di un ristorante, si usa il comando:

case selettore **of**
caso1 : istruzione;
caso2 : istruzione;
caso3 : istruzione;
...
else: istruzione;
end;

Selezione multipla (2)

- Con il costrutto **case**, il valore del selettore viene confrontato con il valore di ogni singolo caso, quando viene trovato un caso che eguaglia il selettore l'istruzione che segue viene eseguita, poi il controllo passa alla prima istruzione dopo il costrutto case. Se nessun caso soddisfa il selettore, viene eseguita l'ultima istruzione, quella individuata dalla parola **else**. Occorre fare attenzione che il costrutto case termina con un **end**.

```
PROGRAM SceltaMenu;
USES crt;
VAR a:INTEGER;
BEGIN
    CLRSCR;
    WRITELN (' Il Menu di oggi:');
    WRITELN ('1. Linguine agli scampi');
    WRITELN('2. Tagliatelle alla bolognese');
    WRITELN('3. Insalatona 5 colori');
    WRITELN('Digita 1, 2 o 3 a seconda del piatto che preferisci');
    READLN(a);
    CASE a OF
      1:BEGIN
          WRITELN('Ottima scelta gli scampi sono freschissimi!');
      END;
      2:BEGIN
          WRITELN('Ottima scelta le tagliatelle alla bolognese sono il piatto della casa!');
      END;
      3:BEGIN
          WRITELN('Ottima scelta! Per stare leggeri ma gustare un buon piatto non c''e'' nulla
          do meglio della nostra insalatona');
      END;
      ELSE
      BEGIN
          WRITELN(' Spiacente ma il piatto da Lei scelto non e'' nel menu'');
      END;
    END;
    readln;
END.
```

```
ROGRAM numeri_casuali;
USES crt;
VAR a,b,c:INTEGER;
BEGIN
    CLRSCR;
    RANDOMIZE; (*Inizializza,ossia azzera il generatore di numeri casuali *)
    c:=RANDOM(10); (*generare un numero a caso compreso tra 0 e il numero indicato in
    parentesi. Il massimo consentito è 255 *)
    CASE c OF
    7:BEGIN
            WRITELN(' Il numero e'' ',c);
            END;
    9:BEGIN
            WRITELN(' Il numero e'' ',c);
            END;
    3:BEGIN
            WRITELN(' Il numero e'' ',c);
            END;
    ELSE
            BEGIN
            WRITELN(' Il numero uscito e'',c,' diverso da 3 7 e 9');
            END;
    END;
    readln;
END.
```


Iterazione enumerativa

- Serve per eseguire un operazione (una o più istruzioni) n volte.
- in Pascal è indicata dal comando:
  FOR a=: **ni** **TO** **nf** **DO**(* Niente punto e virgola *)
  begin
    istruzione 1;
    istruzione 2;
    istruzione ..;
  END;
- Nel nostro caso **ni** sostituisce il numero intero dal quale si parte a enumerare e **nf** sostituisce il numero fino al quale si continua a enumerare (ni<nf).

Iterazione enumerativa (2)

FOR a=: ni **TO** nf **DO**(* Niente punto e virgola *)
begin
  istruzione 1;
  istruzione 2;
  istruzione ...;
END;

Iterazione enumerativa (3)

- E' possibile anche procedere a ritroso, contando alla rovescia da un numero n fino ad arrivare ad un numero t (cioè in questo caso n>t:
- FOR a:=n DOWNTO t DO

Esempio

```
PROGRAM ciclo_do;  {short DO demo program}
VAR                                  {by A Programmer}
  index : INTEGER;
BEGIN
  FOR index := 1 TO 10 DO
  BEGIN
    WRITELN(index);
  END;
  readln;
END.
```


Esempio: calcola la potenza

```
PROGRAM potenza;
VAR a,b,c,i:INTEGER;
BEGIN
   WRITELN('Inserisci un numero');
   READLN(a);
   WRITELN(' Inserisci esponente ');
   READLN(b);
   c:=a;
   FOR i:= 1 TO b-1 DO
      begin
         c:=a*c;
   END;
   WRITELN('Il risultato della potenza e'' ' ,c);
   READLN;
END.
```

Iterazione per vero

WHILE *condizione* **DO**
begin
istruzione 1;
istruzione 2;
istruzione...;
END;

- il blocco di istruzioni presente tra WHILE DO e END; viene ripetuto finché si verifica la condizione.
- Nel caso in cui la condizione non si verifica il blocco di istruzioni non viene mai eseguito

Iterazione per vero (2)

- Tra **begin** ed **end** vanno inserite le istruzioni che cambiano il valore di un qualche cosa affinché la condizione si avveri. Se vi è una sola istruzione, **begin** e **end** possono essere omessi.
- E' importante fare attenzione che ci sia all'interno del ciclo qualcosa che modifichi il valore della condizione, altrimenti il ciclo si ripete all'infinito e il programma non giungerà mai alle righe che seguono l'istruzione **end**.
- Questo è un comune errore che con un poco di attenzione si può evitare.

Iterazione per vero (3)

- Qualche esempio

```
...
x:=1;
 While x=5 Do
 x:=x+1;
...
```

- In questo esempio viene eseguito quanto scritto all'interno del ciclo affinchè non viene verificata la condizione, cioè ad x viene sommato 1 fino a che il suo valore non sia uguale a 5.

Esempio: somma di k numeri

```
Program sommak;
var K,x,somma: integer;
begin
    readln(K);
    somma:=0;
while (K>0) do
    begin;
          readln(x);
          somma:=somma+x;
          K:=K-1;
    end;
    writeln(somma);
   readln;
end.
```

Array

- Un *array* rappresenta una collezione di informazioni del medesimo tipo, ciascuna associata a un *indice* che appartiene a un intervallo di numeri interi (in Pascal è possibile comunque usare come indici anche gli altri tipi ordinali). La dichiarazione di un tipo *array* usa la seguente sintassi:

type identificatore = **array [** *indice_minimo .. indice_massimo* **] of** *tipo_base* **;**

- dove *indice_minimo* e *indice_massimo* sono gli estremi (inclusi) dell'intervallo di possibili indici, e *tipo_base* è il tipo degli elementi di cui è costituito l'array. Il *tipo_base* può essere sia uno dei tipi predefiniti del Pascal che un tipo definito dall'utente (sia atomico che strutturato). Il numero di elementi di un array è calcolabile come: *indice_massimo* - *indice_minimo* +1.

Array (2)

- Su una variabile di tipo array è possibile accedere ai singoli elementi usando la notazione:

  variabile **[** *espressione* **]**

- dove il risultato di *espressione* viene usato come indice per selezionare l'elemento corrispondente dell'array. Un elemento di un array può essere usato come una variabile appartenente al *tipo_base* dell'array.

Array - Esempio

```
Program Leggi3;
var A: array [0..2] of integer;
        i : integer;
begin
   i := 0;
   while i<3 do
     begin;
        read(A[i]);
        i:=i+1;
   end;
   end.
```


Iterazione per falso

- Esegue il ciclo racchiuso tra **REPEAT** e **UNTIL** se, e solo se, la condizione espressa da UNTIL è vera, in caso contrario (la proposizione è falsa) passa all'istruzione successiva.

  REPEAT
  istruzione 1;
  istruzione 2;
  istruzione ...;
  UNTIL *condizione*;
- Anche in questo caso occorre porre attenzione al fatto che la condizione di uscita deve diventare vera in qualche modo, altrimenti finiamo in un loop infinito perdendo il controllo del programma.

Repeat until - Esempio

```
PROGRAM ripetizione;
VAR a:INTEGER;
BEGIN
   REPEAT
        WRITELN(' Inserisci un numero maggiore di 10
        e minore di 30');
        READLN(a);
   UNTIL ((a>10) and (a<30));
   READLN;
END.
```


For, While, Repeat until

- Nel caso del ciclo **For** il numero di cicli eseguiti è noto, poiché il contattore parte da un elemento iniziale ed arriva fino all'elemento finale.
- Nel caso di **While** e di **Repeat Until** il numero di volte che viene ripetuto il ciclo generalmente non è noto a priori, in quanto dipende dal cambiamento che subisce la variabile che controlla la condizione.
- C'è da notare che le istruzioni del ciclo **Repeat Until** verranno eseguite almeno una volta, poiché la condizione viene verificata alla fine, dopo il codice del ciclo.
- Mentre la condizione di ingresso di un ciclo **while** essendo testata prima di eseguire il codice associato al ciclo, potrebbe se questa risulta falsa non fare nulla e riprendere dalle istruzioni che si trovano dopo il ciclo, è importante quindi verificare con attenzione se il flusso di esecuzione del programma entra almeno qualche volta nel codice del ciclo, controllando se la condizione di ingresso risulta vera almeno in qualche caso.

Ancora stringhe

- Per gestire sequenze di caratteri si usa il tipo *string* (una sequenza di caratteri viene detta *stringa* di caratteri), simile al tipo array. La dichiarazione di un tipo stringa usa la seguente sintassi:

  type *identificatore* **= string [** *lunghezza_massima* **];**
- dove *lunghezza_massima* rappresenta il massimo numero di caratteri della sequenza.
- La specifica della lunghezza massima può essere omessa insieme alle relative parentesi quadre; in questo caso si assume come lunghezza massima 255, che è il più alto valore specificabile.

Ancora stringhe (2)

- Con variabili di tipo stringa sono definite le seguenti operazioni:
  - assegnazione, attraverso il valore di un'altra variabile o di una costante letterale di tipo stringa
  - accesso a un singolo carattere, con una notazione simile a quella usata dagli array: il primo carattere ha indice 1, il secondo ha indice 2 e così via; quindi se x è una variabile di tipo stringa, x**[**1**]** rappresenta il primo carattere della stringa contenuta nella variabile
  - calcolo della lunghezza *effettiva* della stringa (in contrapposizione alla lunghezza massima) attraverso la funzione predefinita **length**
  - concatenazione di stringhe, attraverso l'operatore +