

Sommario

Strutture di iterazione2

WHILE-ENDWHILE.....2

REPEAT-UNTIL3

FOR-DO3

Conclusione4

Impareremo in questo modulo:

- Costrutto WHILE – DO
- Costrutto REPEAT – UNTIL
- Costrutto FOR – DO

Strutture di iterazione

Le strutture di iterazione sono utili quando si vuole fare una operazione ripetitiva, senza dover scrivere lo stesso codice più volte. Questo diventa ancora più importante quando non sappiamo al momento della scrittura del codice, quante volte verrà ripetuto questo comando. Per conoscere i comandi di questo tipo, presenti in tutti i linguaggi di programmazione, impareremo i comandi WHILE- DO, REPEAT-UNTIL e FOR-DO.

WHILE - ENDWHILE

Il comando WHILE- DO ripete una serie di operazioni, mentre una data condizione è vera. La sua struttura è la seguente:

```
WHILE (<espressione logica/confronto>
DO
    (operazioni);
ENDWHILE;
```

In questo comando, l'intera serie di operazioni viene eseguita fino a quando l'espressione logica o di confronto assumono valore vero. Una cosa importante da osservare è che, se l'espressione logica o di confronto è falsa sin dall'inizio, non viene eseguita alcuna operazione.

Nel programma che segue un esempio di utilizzo di questo comando:

```
1   Programma: Conto alla rovescia;
2   Variabili
3       var contatore : intero;
4   Inizio
5       contatore ← 10;
6   WHILE (contatore >=0)
7   DO
8       WRITE contatore, "...";
9       contatore ← contatore - 1;
10  ENDWHILE;
11  Fine.
```

Questo codice mostra alla riga 5 l'assegnazione del valore di 10 alla variabile contatore. Dopo questa riga il comando WHILE in 6 controlla se questo valore è maggiore o uguale a zero, poiché è vero, viene eseguito il

codice delle righe da 7 a 10. Questo frammento di codice stampa 10 sullo schermo, seguito da puntini (sulla riga 8) e decrementa (sottrae 1) il valore del contatore di 1 (riga 9). Fatto questo si ritorna alla riga 6, si effettua nuovamente il controllo e il ciclo si ripete. Questa ripetizione è chiamata ciclo. Il codice si fermerà solo quando si stamperà il valore "0 ..." e il contatore verrà impostato al valore di -1. A questo punto, poiché la condizione alla riga 6 è falsa, l'esecuzione passerà alla riga 10, che è ENDWHILE ed il programma terminerà incontrando il comando "Fine".

Spesso, eseguire un programma nella mente, controllando quale codice viene eseguito e che valori assumono le variabili, è molto importante per trovare eventuali problemi.

REPEAT-UNTIL

Il comando REPEAT-UNTIL è molto simile al comando WHILE-DO, ma la condizione è verificata alla fine del ciclo.

```
REPEAT  
    (operazioni);  
UNTIL (<espressione logica/confronto>;
```

In questo caso, la differenza principale è che l'insieme di operazioni viene eseguito almeno una volta, anche se la condizione è inizialmente falsa (cosa che non avviene con il comando WHILE-DO).

Bisogna stare molto attenti a questa differenza, che può influenzare la logica del programma, quando si arriva ad un valore di soglia.

FOR-DO

L'iterazione FOR-DO è un po' diversa dalle altre strutture, in quanto si crea una variabile utilizzata come accumulatore e si controlla il suo valore. La sintassi del comando è la seguente:

```
FOR <variabile> ← <valore_iniziale> TO <valore_finale>  
DO  
    (operazioni);  
ENDFOR;
```

Questo comando pone la <variabile> al <valore_iniziale> ed esegue le operazioni definite nel blocco tra DO e ENDFOR. Al termine della prima iterazione, il valore della <variabile> viene incrementato di 1 (uno) e il processo ripetuto, fino al momento in cui il valore assunto dalla <variabile> è maggiore di <valore_finale>.

Così il codice qui sotto stamperà sullo schermo il seguente risultato: "1 ... 2 ... 3 ..." perché il ciclo verrà eseguito per $i = 1$, $i = 2$ e $i = 3$. Dopo la terza iterazione, quando i viene incrementato di 1, ed assume valore 4, la condizione di far variare i tra 1 e 3 non è più vera, e quindi, le operazioni all'interno del ciclo non sono più eseguite.

```
FOR i ← 1 TO 3  
DO  
    Write i, "...";  
ENDFOR;
```

Alcune possibili varianti per il comando sono:

```
FOR <variabile> ← <valore_iniziale> DECREASE_TO <valore_finale>  
DO  
    (operazioni);  
ENDFOR;
```

Il comando precedente si utilizza quando si vuole passare da un valore iniziale maggiore ad un valore finale inferiore, come un "conto alla rovescia". Un'altra variante è quella di cambiare la dimensione dell'incremento/decremento della <variabile> ad un altro valore definito da <passo>, come illustrato di seguito:

```
FOR <variabile> ← <valore_iniziale> [TO/DECREASE_TO] <valore_finale> STEP  
<passo>  
DO  
    (operazioni);  
ENDFOR;
```

Un esempio di codice potrebbe essere:

```
FOR i ← 1 TO 10 STEP 2  
DO  
    WRITE i, "...";  
ENDFOR;
```

Stampa tutti i numeri dispari fino a 10 (1, 3, 5, 7 e 9) perché il passo del ciclo è 2, e quindi la variabile i, che controlla il loop sarà aumentata di 2 ad ogni iterazione. Quando il campo <passo> non è definito nel ciclo, si assume che il passo è 1.

Conclusione

In questo modulo abbiamo appreso le strutture di selezione WHILE-DO, REPEAT-UNTIL e FOR-DO. Esse servono per implementare ripetizioni di istruzioni nel flusso di esecuzione dell'algoritmo, continuando i

LOGICA DI PROGRAMMAZIONE - MODULO 3

loop fintantoché una condizione è vera. Questa condizione si chiama condizione di arresto. Impariamo quando e come usarli, e il layout di default. Se necessario, riesaminare il contenuto!