

Sommario

Modulo 2 - Comandi di selezione	2
Comando IF-THEN-ELSE	2
Conclusione	5

Cosa impareremo in questo modulo:

- Comando IF-THEN-ELSE
- Comando Select

Modulo 2 - Comandi di selezione

A volte abbiamo bisogno di eseguire comandi diversi nel codice a seconda di diverse condizioni. Un esempio potrebbe essere quello di consentire il pagamento di una fattura solo se l'utente ha sufficiente disponibilità economica nel suo conto corrente.

Per realizzare controlli di questo tipo, i linguaggi di programmazione forniscono comandi di selezione. Nel caso del nostro pseudo linguaggio, vedremo il comando "IF-THEN-ELSE" e il comando Select. Questi due comandi hanno sintassi molto simili in quasi tutti i linguaggi di programmazione.

Comando IF-THEN-ELSE

Il comando IF-THEN-ELSE viene utilizzato quando si vogliono eseguire uno o più comandi solo al verificarsi di una condizione. Si può eventualmente eseguire un diverso insieme di comandi se ciò non avviene.

La struttura di questo comando è la seguente:

```
IF(<espressione logica / confronto / valore logico>)  
THEN  
    {Operazioni}  
ENDIF;
```

Osservate che il comando IF ha un solo punto e virgola dopo la parola ENDIF, anche se ogni operazione nel blocco tra THEN e ENDIF deve essere delimitata da un punto e virgola, come in qualsiasi altra parte del codice. Un altro punto da notare in termini di sintassi è che la condizione, data da un'espressione logica o un confronto di espressione, deve essere inserita in parentesi.

L'espressione all'interno delle parentesi restituirà un valore logico ed esso è il valore che verrà utilizzato nel processo decisionale del comando SE.

Nella struttura del comando che è stato presentato, l'insieme di operazioni verrà eseguito solo se il risultato dell'espressione è un valore VERO.

Un esempio di utilizzo di questa struttura può essere osservato nel seguente frammento di codice:

```
IF(eta >= 18)  
THEN  
    WRITE "Hai più di 18 anni!";  
    accesso ← VERO;  
ENDIF;
```

Il comando IF ha una struttura più completa, come si può vedere di seguito:

```
IF(<espressione logica / confronto / valore logico>)  
THEN  
    {Insieme di operazioni 1}  
ELSE  
    {Insieme di operazioni 2}  
ENDIF;
```

In questo tipo di sintassi dei comandi, l'insieme delle operazioni 1 viene eseguito se il risultato dell'espressione è vero e l'insieme delle operazioni 2 viene eseguito se il risultato è falso.

Usando lo stesso esempio di "età >= 18", si potrebbe avere un codice come questo:

```
IF(eta >= 18)  
THEN  
    WRITE "Hai più di 18 anni!";  
    accesso ← VERO;  
ELSE  
    WRITE "Hai meno di 18 anni!";  
    accesso ← FALSO;  
ENDIF;
```

Comando SELECT

Talvolta le condizioni che usiamo in un comando di decisione sono date da una serie di possibili valori per la stessa variabile. Questa situazione può essere gestita con una serie di comandi IF-THEN-ELSE.

```
IF(nota = 1)  
THEN  
    WRITE "E' stato immesso il valore 1";  
ELSE  
    IF(nota = 2)  
    THEN  
        WRITE "E' stato immesso il valore 2";  
    ELSE  
        IF(nota = 5)  
        ...  
        ENDIF  
    ...  
    ENDIF;  
ENDIF;
```

Anche se questo codice funziona, sarà molto lungo con molti livelli di comandi, il che renderà il codice difficile da leggere, comprendere e cambiare. Per evitare questo problema, è disponibile il comando SELECT, che ha la seguente struttura:

```

SELECT (variabile)
  CASE valore1:
    {Operazioni}
  ENDCASE;
  CASE valore2:
    {Operazioni}
  ENDCASE;
  ...
  DEFAULT:
    {Operazioni}
  ENDCASE;
ENDSELECT;

```

Questo comando valutato il valore della variabile (quella racchiusa tra parentesi) esegue il blocco di operazioni associate a un determinato valore, fra tutti quelli racchiusi fra le istruzioni CASE e ENDCASE. Se la variabile ha un valore diverso da tutti i casi previsti, sarà eseguito il codice all'interno DEFAULT e ENDCASE. Se non vi è da eseguire una sequenza di operazioni per un valore specifico, l'insieme di istruzioni tra CASE e ENDCASE può essere vuoto (CASO: seguito da ENDCASE;).

Utilizzando il comando SELECT nel precedente esempio, abbiamo un codice come questo:

```

...
SELECT (nota)
  CASE 1:
    WRITE "E' stato immesso il valore 1";
  ENDCASE;
  CASE 2:
    WRITE "E' stato immesso il valore 2";
  ENDCASE;
  CASE 5:
    WRITE "E' stato immesso il valore 5";
  ENDCASE;
  CASE 10:
    WRITE "E' stato immesso il valore 10";
  ENDCASE;
  DEFAULT:
    WRITE "Valore non valido!";
  ENDCASE;
ENDSELECT;

```

Con il comando SELECT, il codice ha una struttura molto più piccola e leggera rispetto all'esempio che utilizza il comando IF-THEN-ELSE. Pertanto, il comando SELECT è molto più adatto quando si lavora con una serie finita di possibili valori della stessa variabile.

È importante ricordare che l'istruzione CASE non accetta operatori logici, quindi non è possibile effettuare alcun confronto. Dobbiamo utilizzare solo valori predefiniti.

In alcuni linguaggi di programmazione utilizzando un comando simile a SELECT non è richiesto l'uso di un ENDCASE; legato alla istruzione CASE. Inoltre, lo stesso comportamento del comando può variare da

linguaggio a linguaggio. Nel caso del nostro pseudolinguaggio, è stato standardizzato per ogni caso l'uso di un ENDCASE;

Conclusione

In questa sezione abbiamo appreso l'uso dei comandi SELECT e IF-THEN-ELSE. Essi servono per controllare il flusso di esecuzione dell'algoritmo, nel quale la deviazione si verifica sulla realizzazione (o meno) di una determinata condizione. Impariamo quando e come usarli, e il layout di default. Se necessario, riesaminare il contenuto!