Corso sul linguaggio C++

Modulo LC

1.3 - Strutture di controllo

Prerequisiti

- Tecnica top-down
- Programmazione elementare in C++

Introduzione

In molti casi le sole istruzioni semplici (lettura, stampa e assegnamento) non sono sufficienti a risolvere certi problemi.

In questo caso c'è la necessità di introdurre strutture che consentono di alterare il flusso di esecuzione del programma.

Questi costrutti sintattici prendono il nome di **strutture di controllo** e possiamo classificarle in:

- **Sequenza**
- **Selezione**
- **Iterazione**

Informazioni generali

Attraverso l'uso delle strutture di controllo sarà possibile realizzare programmi anche molto complessi.

Le strutture di controllo sono strumenti sintattici che consentono di alterare l'esecuzione sequenziale dei programmi.

E' sempre indispensabile che la progettazione dei programmi segua la tecnica **top-down**, che prevede la scomposizione di un problema in sottoproblemi, via via più semplici, da risolvere in un secondo momento

La sequenza

Tutte le istruzioni presenti nella parte eseguibile del programma si dice che costituiscono una **SEQUENZA**, nel senso che **costituiscono un blocco che può essere considerato come un'unica istruzione**

In questo caso la sintassi è la seguente:

```
{
    istruzione;
    istruzione;
    ……
}
```

Questa struttura di controllo prende il nome di **SEQUENZA** ed è caratterizzata dalla coppia di parentesi graffe aperta "{" e chiusa "}".

ATTIVITA': scrivere una applicazione C++ che calcoli la somma di due numeri letti da tastiera.

La sequenza

La sequenza è sempre presente nel **main**()

```
/* PROGRAMMA: calcolo della somma di 2 valori
   interi */
/* AUTORE: Mario Rossi */
/* DATA: 15/02/2002 */
/* COMPILATORE: Visual C++ */
/* FILE: somma.cpp */
#include <iostream>
#define RISULTATO "La somma e':  "
int main()
{  int a,b,c;
    cout<<"Primo valore: ";      cin>>a;
    cout<<"Secondo valore: ";    cin>>b;
    c=a+b;
    cout<<RISULTATO<<c;
    system("Pause");
    return 0;
}
```

La sequenza

```
/* PROGRAMMA: calcolo della somma di 2 valori interi */
/* AUTORE: Mario Rossi */
/* DATA: 15/02/2002 */
/* COMPILATORE: Visual C++ */
/* FILE: somma.cpp */
#include <iostream>
#define RISULTATO "La somma e':  "
int main()
{  int a,b,c;
     cout<<"Primo valore: ";        cin>>a;
     cout<<"Secondo valore: ";      cin>>b;
     c=a+b;
     cout<<RISULTATO<<c;
     system("Pause");
     return 0;
}
```

Notare i **commenti** posti all'inizio del file. Anche se non sono obbligatori, vanno sempre inseriti perché hanno lo scopo di documentare il lavoro prodotto e facilitare eventuali modifiche future.

La sequenza

```
/* PROGRAMMA: calcolo della somma di 2 valori interi */
/* AUTORE: Mario Rossi */
/* DATA: 15/02/2002 */
/* COMPILATORE: Visual C++ */
/* FILE: somma.cpp */
#include <iostream>
#define RISULTATO "La somma e':  "
int main()
{  int a,b,c;
     cout<<"Primo valore: ";        cin>>a;
     cout<<"Secondo valore: ";      cin>>b;
     c=a+b;
     cout<<RISULTATO<<c;
     system("Pause");
     return 0;
}
```

Notare l'uso delle costanti stringa. La costante può essere dichiarata esternamente al **main**()

La costante può essere utilizzata ad esempio nelle istruzioni di stampa.

La selezione doppia

Quando si vuole che il programma segua un percorso diverso a seconda del valore di una espressione logica (vero o falso) si utilizza la **selezione doppia** che ha la sintassi seguente:

if (*espressione-logica)*
*istruzione**;***
else *istruzione***;**

Se *espressione-logica* risulta **VERA** (diversa da 0) viene eseguita la prima *istruzione*, altrimenti viene eseguita la seconda *istruzione*

ATTIVITA': scrivere una applicazione C++ che verifichi se un numero è pari o dispari

La selezione doppia

```
#include <iostream>
#define NUMERO "\nIl numero "
#define PARI     "pari\n"
using namespace std;
int main()
{  int numero;
     cout<<"Immetti il numero: ";
     cin>>numero;
     cout<<"\n"<<NUMERO;
     if (numero % 2)   cout<<"non e' ";
       else  cout<<"e' ";
     cout<<PARI;
     system("Pause");
     return 0;
}
```

L'espressione logica risulta **VERA** se *numero% 2* è diverso da zero (ossia *numero* contiene un valore dispari)

Operatore ternario

Un caso particolare di selezione doppia si può rappresentare con **l'operatore ternario** "?:".

L'operatore ternario ha la sintassi:

(espress-logica) **?** *istruzione1* **:** *istruzione2***;**

Esso calcola *espress-logica* e se risulta **VERA** esegue *istruzione1* altrimenti esegue *istruzione2*.

Costituisce un modo **abbreviato** per effettuare una selezione doppia.

Operatore ternario

ATTIVITA': scrivere tramite l'operatore ternario le seguenti s.d.c. selettive:

1) if (primo > secondo) a=primo-secondo; **else** a=secondo-primo;

(primo>secondo) ? a=primo-secondo **:** a=secondo-primo;

2) **if** (x>0) modulo_x=x; **else** modulo_x=-x;

modulo_x=(x>0)? x : (-x);

3) **if** (y==z) x=u*x+a; **else** x=u*x+b;

x=(u*x+(y==z)? a:b)

La selezione semplice

In alcuni casi, la struttura selettiva si può utilizzare nella forma seguente (**selezione semplice**):

if (*espressione-logica)*
*istruzione***;**

Se *espressione-logica* risulta **VERA** viene eseguita *istruzione*

ATTIVITA': scrivere una applicazione C++ che calcoli il minimo fra tre numeri letti da tastiera.

La selezione semplice

```
#include <iostream>
#include <stdlib.h>
using namespace std;
#define MINIMO "Il minimo e': "
int main()
{   int x,y,z, min;
      cout<<"Primo valore: ";        cin>>x;
      cout<<"Secondo valore: ";      cin>>y;
      cout<<"Terzo valore: ";        cin>>z;
      if (x<y)   min=x;
      else       min=y;
      if (z<min)
        min=z;
      cout<<MINIMO<<min<<endl;
      system("PAUSE");
      return 0;
}
```

Esempio di selezione semplice

La selezione multipla

Quando la scelta deve essere fatta tra molti valori si usa quest'altra struttura di controllo:

```
switch (espressione-intera)
{  case valore-1:
        istruzione-1;break;
   case valore-2:
        istruzione-2; break;
        ……..
        default: istruzione;
}
```

Questa struttura di controllo prende il nome di **SELEZIONE MULTIPLA**

La selezione multipla serve a selezionare un dato valore tra diversi possibili. Se ***espressione_intera*** vale *valore-1* viene eseguita *istruzione1*, se vale *valore-2* viene eseguita *istruzione2* e così via.Se nessun valore corrisponde al valore di *espressione-intera*, la parola **default** consente di trattare le situazioni di mancata corrispondenza.

ATTIVITA': scrivere una applicazione C++ che stampi il nome del mese di cui viene immesso il relativo valore numerico

La selezione multipla

```
switch (mese)
  { case 1:  cout<<"Gennaio\n"; break;
    case 2:  cout<<"Febbraio\n"; break;
    case 3:  cout<<"Marzo\n"; break;
    case 4:  cout<<"Aprile\n"; break;
    case 5:  cout<<"Maggio\n"; break;
    case 6:  cout<<"Giugno\n"; break;
    case 7:  cout<<"Luglio\n"; break;
    case 8:  cout<<"Agosto\n"; break;
    case 9:  cout<<"Settembre\n"; break;
    case 10: cout<<"Ottobre\n"; break;
    case 11: cout<<"Novembre\n"; break;
    case 12: cout<<"Dicembre\n"; break;
    default: cout<<"Non so!\n";      }
}
```

Quando viene trovata una corrispondenza di valori, viene eseguita l'istruzione

Dopo eseguita l'istruzione corrispondente al valore ***mese,*** l'istruzione **break** fa saltare il controllo alla fine dell'istruzione **switch**.

La parola **default** serve a prevedere il caso in cui nessuno dei valori elencati sia corrispondente al valore contenuto nella variabile *mese*

L'iterazione

Quando si deve eseguire ripetutamente un blocco di istruzioni si usano le strutture di controllo iterative.

Le strutture di controllo **iterative** servono ad eseguire i **cicli**.

Un ciclo è la ripetizione di un dato blocco di istruzioni. Nelle strutture di controllo iterative è spesso necessario utilizzare una variabile detta **contatore** per tenere traccia del numero di volte che il ciclo viene effettuato

Iterazione predefinita

Un primo tipo di costrutto iterativo è il seguente:

for (*contatore=inizio;* *contatore<=fine;* *contatore++*)
*istruzione***;**

Condizione di ripetizione

Questa struttura di controllo esegue ripetutamente *istruzione* per ciascuno dei valori che *contatore* assume, partendo da *inizio* fino a *fine*. L'istruzione *contatore*++ serve ad incrementare il valore di *contatore*.

Questa struttura prende il nome di **ITERAZIONE PREDEFINITA** e si usa quando il programmatore conosce a priori il numero di volte che il ciclo deve essere eseguito

L'**ITERAZIONE PREDEFINITA** esegue il ciclo fintantochè la condizione al centro risulta **VERA**. Quando diviene **FALSA**, ossia quando *contatore>fine* il ciclo termina

Iterazione predefinita

Un altro tipo di iterazione predefinita è

for (*contatore=inizio; contatore>=fine; contatore- -*)
istruzione;

Condizione di ripetizione

Questa è analoga alla precedente, ma si usa quando il valore *inizio* è maggiore del valore *fine*. Per indicare che il contatore questa volta deve indietreggiare, si usa l'istruzione *contatore - -*.

L'**ITERAZIONE PREDEFINITA** esegue il ciclo fintantochè la condizione al centro risulta **VERA**. Quando diviene **FALSA**, ossia quando *contatore<fine*, il ciclo termina

ATTIVITA': scrivere una applicazione C++ che calcoli la somma di N numeri interi letti da input.

Iterazione predefinita

```
# include <iostream>
#include <cstdlib>
# define RISPOSTA "La somma e': "
#define  NUMERO   "Immetti valore: "
using namespace std;
int main()
{  int somma=0,  valori,  cont,  valore;
   cout<<"Numero di valori da sommare: ";   cin>>valori;
   for (cont=0; cont<valori; cont++)
   { cout<<NUMERO;
     cin>>valore;
     somma+=valore;
   }
   cout<<RISPOSTA<<somma<<"\n";
   system("Pause");
   return 0;
}
```

contatore

La variabile *cont* è assunta come **contatore** del numero di cicli. Il ciclo si ripete quando *cont<valori.* L'incremento del contatore è *cont++*

La variabile *somma* è usata come accumulatore

Iterazione precondizionata

Quando **NON** si conosce a priori il numero di volte che un ciclo deve essere ripetuto, si usa la seguente:

while (*espress-logica*)
*istruzione***;**

Questa struttura di controllo esegue il ciclo fintantochè *espressione-logica* risulta **VERA**. Esce quando diventa **FALSA**.

Ovviamente, se *espressione-logica* è falsa già all'inizio, la while NON SARA' ESEGUITA NEMMENO UNA VOLTA

Questa struttura prende il nome di **ITERAZIONE PRECONDIZIONATA** poiché la condizione da verificare si trova prima del blocco di istruzioni da ripetere

ATTIVITA': scrivere una applicazione che dato un intero n da input, ne calcola il divisore più grande.

Iterazione precondizionata

```
#include <iostream>
using namespace std;
int main()
{  int n, i, maxdiv;
   cout<<"Valore di n: ";
   cin>>n;
   i=n-1;
   while (i>1)
   {  if (n%i==0)
         break;
      i--;
    }
   maxdiv=(i==0)? 1:i;
   cout<<"Il massimo divisore di "<<n<<" e' "<<maxdiv<<endl;
   system("Pause");
   return(0);
}
```

Operatore ternario

Iterazione postcondizionata

In altri casi, e sempre quando NON si conosce a priori il numero di volte che un ciclo deve essere ripetuto, si usa la seguente:

do
{ *istruzione;*
} **while** (*espress-logica*);

Questa struttura di controllo esegue il ciclo fino a quando *espressione-logica* risulta **VERA**. Esce quando diventa **FALSA**.

Ovviamente, anche se *espressione logica* è falsa già all'inizio, la **do-while** VERRA' ESEGUITA ALMENO UNA VOLTA.

Questa struttura prende il nome di **ITERAZIONE POSTCONDIZIONATA** poiché la condizione da verificare si trova dopo il blocco di istruzioni da ripetere

ATTIVITA': scrivere una applicazione che legga da input una serie di stringhe terminata dalla stringa nulla e che stampi la lunghezza massima di esse.

Iterazione postcondizionata

```
#include <iostream>
#include <string>
#include <cstdlib>
using namespace std;
int main()
{   string s;
    int maxlen=0;
    do { cout<<"Immetti stringa (Ctrl-Z per finire): ";
         cin>>s;
         if ((int)s.size()>maxlen)
              maxlen = s.size();
    } while (cin);
   cout<<"La lunghezza massima delle stringhe e' "<<maxlen<<endl;
   system("Pause");
   return 0;
}
```

La fine dell'input avviene premendo Ctrl-Z.

Argomenti

- La sequenza
- La selezione doppia
- Operatore ternario
- La selezione semplice
- La selezione multipla
- L'iterazione
- Iterazione predefinita
- Iterazione precondizionata
- Iterazione postcondizionata

Altre fonti di informazione

- J. Purdum, C – ed. Jackson
- Romagnoli-Ventura, C/C++ - Ed. Petrini
- A. Lorenzi et alii – Il linguaggio C++ - Ed. ATLAS
- A.Bellini-A.Guidi, Conoscere il C – ed. McGraw Hill