Corso sul linguaggio C++

Modulo LC

1.1 - Introduzione alla programmazione

Prerequisiti

- Saper progettare un algoritmo
- Saper descrivere un algoritmo mediante linguaggio di progetto (Notazione Lineare Strutturato o diagrammi di flusso)
- Saper usare un sistema operativo
- Compilazione, link, esecuzione di programmi
- Conoscere il concetto di espressione

Introduzione

In questa lezione si vedranno alcuni concetti fondamentali per la programmazione in C++.

Con gli strumenti descritti si potranno scrivere e provare semplici programmi.

Ovviamente per esercitarsi è necessario che sul pc sia presente un compilatore C++ di qualche tipo (Microsoft, Borland, Dev C++, ecc).

Informazioni generali

Per poter programmare occorre conoscere:

- l'algoritmo risolutivo del problema
- la struttura del programma
- gli elementi del programma

N.B.

- I caratteri **grassetto** indicano parole chiave del linguaggio, mentre i caratteri *corsivo* indicano elementi che dovranno essere specificati dal programmatore
- Il C++ è **case sensitive** ossia distingue tra maiuscolo e minuscolo, quindi bisogna fare attenzione durante la scrittura delle istruzioni

Concetti fondamentali

Un' **istruzione** è una frase scritta in un determinato linguaggio di programmazione che indica alla macchina un singolo compito da svolgere.

Questo compito sarà eseguito quando tutta la sequenza di istruzioni sarà conosciuta dalla macchina.

La sequenza di istruzioni costituisce il **programma**.

Eseguire un programma significa svolgere tutte le sue istruzioni in sequenza, partendo dalla prima.

Concetti fondamentali

I dati su cui il programma opera sono contenuti in:

- variabili
- costanti
- espressioni

Il dato contenuto in una variabile *può essere modificato* al contrario di quello contenuto in una costante.

Variabili

La dichiarazione di una **variabile** segue la sintassi:

tipo ident;

dove

- *tipo* indica il tipo della variabile, che può essere:
  - **int** – interi
  - **bool** – tipo logico o booleano
  - **char** – caratteri
  - **float** – numeri in virgola mobile semplice precisione
  - **double** – numeri in virgola mobile doppia precisione
- *ident* è il nome (identificatore) che il programmatore ha assegnato alla variabile
- Esempi:
  - **int** somma**;** **//** Dichiara la variabile di nome somma di tipo **int**
  - **int** var1**,** var2**;** **//** Dichiara due variabili, ognuna di tipo **int**

Costanti

La dichiarazione di una **costante** segue la sintassi:

#define *ident espressione;*

oppure

const *tipo ident* = *espressione;*

dove

- *ident* è il nome (identificatore) che il programmatore ha assegnato alla costante
- *espressione* indica il valore da assegnare
- *tipo* indica il tipo della variabile (elencati in precedenza)
- Esempi:
  - **const int** somma = 40**;** **//** Dichiara la costante di nome *somma* come **int**
  - **#define** n 50; // Dichiara la costante *n* pari a 50

Espressioni

All'interno delle istruzioni possono figurare **espressioni**

Una espressione può essere:

- una costante
- una variabile
- una combinazione di espressioni mediante operatori
- Esempi:
  - 5 è un'**espressione costante intera**
  - 3+6 è un'**espressione numerica**
  - X+5 è un'**espressione variabile**
  - 'a' è un'**espressione costante carattere.**

Successivamente si vedranno in dettaglio i vari tipi di operatori

Struttura di un programma in C++

```
#include <cstdlib>
#include <iostream>
using namespace std;

int main(int argc, char *argv[]))
{
    dichiarazione variabili e costanti
    blocco istruzioni
    system ("PAUSE");
    return EXIT_SUCCESS;
}
```

Le frasi in *corsivo* indicano un elemento generico che sarà specificato successivamente nella fase di codifica del programma

Struttura di un programma in C++

```
#include <cstdlib>
#include <iostream>
using namespace std;

int main(int argc, char *argv[]))
{
    dichiarazione variabili e costanti
    blocco istruzioni
    system ("PAUSE");
    return EXIT_SUCCESS;
}
```

Questo è il programma principale detto **main program** all'interno del quale scriveremo le istruzioni da eseguire

Struttura di un programma in C++

```
#include <cstdlib>
#include <iostream>
using namespace std;

int main(int argc, char *argv[]))
{
    dichiarazione variabili e costanti
    blocco istruzioni
    system ("PAUSE");
    return EXIT_SUCCESS;
}
```

Questo è il *blocco delle dichiarazioni*. In questa parte si scrivono alcuni elementi che serviranno in seguito al programma.

Struttura di un programma in C++

```
#include <cstdlib>
#include <iostream>
using namespace std;

int main(int argc, char *argv[]))
{
    dichiarazione variabili e costanti
    blocco istruzioni
    system ("PAUSE");
    return EXIT_SUCCESS;
}
```

In questa parte si scrivono in sequenza, le istruzioni che dovranno essere eseguite.

Le istruzioni semplici

Le istruzioni semplici sono:

- lettura
- stampa
- calcolo e assegnazione

Un programma formato solo da questi 3 tipi di istruzioni viene sempre eseguito in sequenza dall'inizio alla fine.

Lettura

L'istruzione di lettura serve a leggere dalla tastiera il valore di una variabile. La sintassi è:

cin>>*ident*;

dove

- *ident* indica il nome della variabile da acquisire

- Esempio:
  - **cin**>>nlati;

Legge da tastiera un valore intero e lo pone nella variabile *nlati*

Stampa

L'istruzione di stampa serve a stampare a video il valore di una variabile. La sintassi è:

cout<<*espressione***;**

dove

- *espressione* può essere una costante, una variabile o una espressione qualunque.

- Esempio:
  - **cout**<<nlati;
  - **cout**<<"La somma è "<<somma;

Stampa a video il contenuto della variabile *nlati*

Stampa il messaggio *"La somma è"* seguito dal valore della variabile *somma*

Stampa

Si possono ottenere in stampa effetti speciali mediante le **sequenze di escape** da abbinare all'istruzione **cout**. Alcune sono indicate nella tabella seguente.

Carattere di escape	Effetto
\n	A capo
\a	Segnalazione acustica
\t	Tabulazione orizzontale
\b	Una battuta indietro
\r	Ritorno a capo sulla stessa riga
\0	Carattere di fine stringa
\f	Salto pagine
\\	Stampa il carattere '\'

Stampa

Esempi di sequenze di escape

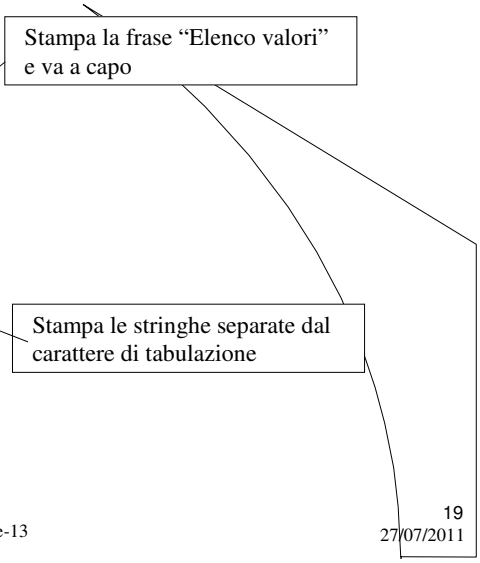
```
cout<<"Emetto un beep (\\a) - premere un tasto \a";
cout<<"Emetto un carattere di carriage return (\\r) - premere un tasto\r";
cout<<"Questo e' l'uso del tabulatore (\\t)\n";
cout<<"Oggi\te'\til\t19\tfebbraio\t1993 - premere un tasto";
cout<<"Questo e' il carattere \101 di codice ottale \\101 - premere un tasto";
cout<<"Questo e' il carattere \x41 avente codice hex \\x41\n";
```

Stampa

Ad esempio:

cout<<"Elenco valori\n";

cout<<"Nome\tTel.";



Stampa formattata

L'istruzione **cout** consente anche la **stampa formattata**, ossia con determinate caratteristiche di visualizzazione dei dati.

Istruzione	Descrizione
cout.width(n);	Definisce l'ampiezza *n* del campo per le operazioni di input e output.
cout.fill('*');	Definisce il carattere di riempimento per gli spazi vuoti
cout.flags (**ios::left**); **cout.flags** (**ios::right**);	Allineamento a sinistra Allineamento a destra
cout.precision(5);	Imposta la precisione: 4 posti per la parte numerica e 1 per il punto decimale
cout.flags(**ios**::**fixed**);	Rappresentazione in virgola fissa
cout.flags (**ios**::**scientific**);	Rappresentazione in virgola mobile

Assegnazione

Questa istruzione consente di porre in una variabile il valore di una espressione. La sintassi è:

ident = *espressione*;

dove

- *espressione* può essere una costante, una variabile o una qualunque espressione che viene assegnata alla variabile di nome *ident*
- Esempi:
  - X = 1;
  - Nome = “Rossi”;
  - S = x + y;

Le librerie

Questo è un esempio di programma C++ che stampa i saluti.

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
    cout<<"Hello world - This is the first C++ program"<<'\n';
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Le righe che iniziano con il simbolo “#” si dicono **direttive**. La direttiva **#include <iostream>** serve ad includere la **libreria iostream**, necessaria per utilizzare le istruzioni di Input e Output (**cin** e **cout**)

Le **librerie** sono particolari file che contengono l’espansione di alcune istruzioni in modo che il programmatore possa limitarsi a fare solo riferimento ad esse.

Le direttive vengono esaminate da un modulo del compilatore detto **preprocessore** che viene attivato subito prima della compilazione

Uso di variabili

ATTIVITA':Scrivere un programma che calcoli la somma di due interi letti da input

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
  int a,b,c;
  cout<<"Primo valore: ";
  cin>>a;
  cout<<"Secondo valore: " ;
  cin>>b;
  c=a+b;
  cout<<"La somma e' "<<c<<endl;
  system("Pause");
  return EXIT_SUCCESS;
}
```

In questo caso si usano 3 variabili *a*,*b* e *c* intere. La sintassi della dichiarazione delle variabili è:

tipo lista_ident;

Per convenzione i nomi delle variabili si è soliti scriverli in caratteri minuscoli

tipo può essere: **int**, **float**, **char** e **double** a seconda del tipo di problema

Uso di variabili

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
  int a,b,c;
  cout<<"Primo valore: ";
  cin>>a;
  cout<<"Secondo valore: " ;
  cin>>b;
  c=a+b;
  cout<<"La somma e' "<<c<<endl;
  system("Pause");
  return EXIT_SUCCESS;
}
```

Il programma di esempio legge da tastiera i valori delle due variabili *a* e *b*.

Si **assegna** a c il valore della somma tra a e b.

La sintassi per l'assegnamento è:

ident = *espressione*;

espressione può essere una qualunque espressione in senso matematico o algebrico.

I commenti

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
  int a,b,c; /* dichiarazione variabili */
  cout<<"Primo valore: ";
  cin>>a;
  cout<<"Secondo valore: " ;
  cin>>b;
  c=a+b; //commento su una riga
  cout<<"La somma e' "<<c<<endl;
  system("Pause");
  return EXIT_SUCCESS;
}
```

Questi sono esempi di **commenti**. Un commento rappresenta note che il programmatore scrive all'interno del programma e che non saranno elaborate durante l'esecuzione.

La sintassi per i commenti prevede:

Il simbolo "/*" per inizio commento

Il simbolo "*/" per fine commento

Un commento può occupare anche più di una riga.

Se un commento si trova solo su una riga può essere utilizzato il simbolo "//"

Istruzioni

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
  int a,b,c; /* dichiarazione variabili */
  cout<<"Primo valore: ";
  cin>>a;
  cout<<"Secondo valore: " ;
  cin>>b;
  c=a+b; //commento su una riga
  cout<<"La somma e' "<<c<<endl;
  system("Pause");
  return EXIT_SUCCESS;
}
```

Si noti che ogni istruzione termina con un simbolo di ";".

Questa è una precisa regola sintattica del linguaggio.

La parola **endl** ha un effetto equivalente a '\n' (andare a capo)

Compilazione

Il programma scritto (**programma sorgente**) mediante un **editor** deve essere salvato su disco (con estensione **.c**) e sottoposto successivamente alla compilazione per ottenere un programma eseguibile.

Il compilatore è un programma che:

- **rileva eventuali errori** nella sintassi delle istruzioni, ossia nel modo in cui le istruzioni sono state scritte e segnala il punto in cui si sono verificati. Il programmatore dovrà correggere gli erorri e *compilare di nuovo il programma sorgente.*
- quando il programma risulta privo di errori sintattici, **effettua la sua traduzione** da **programma sorgente** a **programma oggetto (.obj)**, che risulta scritto in **linguaggio binario**.

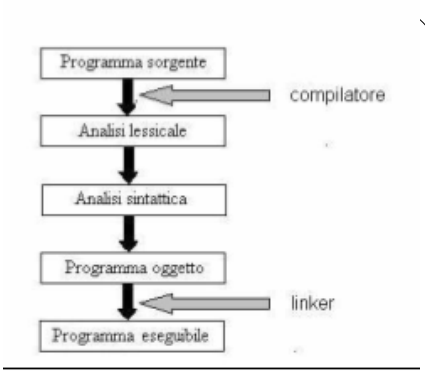
Link

Una volta ottenuto il programma oggetto **(.obj**) come effetto della compilazione, interviene il *linker* che lo collega alle librerie di sistema incluse (ad esempio **iostream**), generando il **programma eseguibile (.exe)**.

Questa fase, detta **fase di link** (da *to link* = collegare), completa la versione binaria (.obj) del programma con alcuni riferimenti mancanti. Il file .exe *è eseguibile anche all'esterno dell'ambiente di* sviluppo.

Attività di sviluppo software

L'attività di sviluppo software può essere schematizzata come indicato



Libreria **cmath**

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{ float r;
  cout<<"Immetti il raggio: ";
  cin>>r;
  cout<<"L'area e' "<<M_PI*r*r<<endl;
  system("Pause");
  return EXIT_SUCCESS;
}
```

Esistono **costanti predefinite** nelle varie librerie del C. Ad esempio, dovendo usare il valore di π, si deve includere la libreria **<cmath>**: **i**n questo modo è disponibile il valore di π con il nome simbolico **M_PI**

Il programma legge la misura del cerchio in una variabile **float** e calcola e stampa il valore dell'area anch'essa **float**

Libreria **cmath**

A titolo di esempio riportiamo le più comuni funzioni presenti in **cmath**

Funzione	*Effetto*
pow (b, e)	potenza b^e
sqrt(x)	radice quadrata di x
abs(x)	valore assoluto di x
ceil(x);	arrotonda x all'intero superiore
floor(x)	arrotonda x all'intero inferiore
srand(t)	inizializza il generatore di numeri casuali
rand()	estrae numero casuale

Argomenti

- Concetti fondamentali
- Variabili
- Costanti
- Espressioni
- Struttura di un programma in C++
- Le istruzioni semplici
- Lettura
- Stampa
- Stampa formattata
- Assegnazione
- Le librerie
- Uso di variabili
- I commenti
- Istruzioni
- Compilazione
- Link
- Attività di sviluppo software
- Libreria **cmath**

Altre fonti di informazione

- J. Purdum, C - ed. Jackson
- Romagnoli Ventura, C/C++ - Ed. Petrini
- A. Lorenzi et alii – Il linguaggio C++ - Ed. ATLAS