APPUNTI CLASSI III / IV PROGRAMMAZIONE C++

```
// Somma.cpp: somma di due numeri
#include <iostream>
using namespace std;

int main()
{
    int a, b, s;
    cin >> a;
    cin >> b;
    s = a + b;
    cout << s;

    return 0;
}
```

Il programma che utilizza operazioni di input e output deve quindi contenere all'inizio un'apposita
dichiarazione (direttiva **include**) per includere in esso la libreria di I/O:
#include <iostream>

Un programma complesso può contenere e utilizzare un grande numero di nomi e identificatori
per i dati e le istruzioni. Risulta quindi comodo, per evitare ambiguità sull'uso degli identificatori,
raggruppare i nomi in contenitori, detti **namespace** (*spazio dei nomi*), predefiniti nel linguaggio
o definiti dai programmatori.
In particolare gli identificatori standard del linguaggio C++ sono contenuti nel *namespace* **std**
(abbreviazione di *standard*).

Scorciatoie per le parentesi graffe:

{	Alt + 123 (tastierino numerico)	Ctrl + Alt + Shift + [	Shift + AltGr + [
}	Alt + 125 (tastierino numerico)	Ctrl + Alt + Shift +]	Shift + AltGr +]

- **La funzione main:**

Una funzione, in generale, oltre a ricevere gli argomenti necessari all'elaborazione, restituisce (o, come si dice correntemente in informatica, *ritorna*) un valore come risultato dell'elaborazione, proprio come accade per le funzioni matematiche. La funzione *main* restituisce al sistema operativo un valore 0 (*Falso*) o diverso da 0 (*Vero*) per indicare, rispettivamente, l'assenza o la presenza di uno stato di errore al termine dell'esecuzione. Questo spiega l'istruzione finale del programma precedente: **return** `0;`
Il *main* ritorna il valore 0, cioè errore = Falso, come dire "non si sono verificati errori". Quindi il valore restituito dalla funzione *main* è un numero intero: il tipo di dato restituito è indicato prima della parola *main* con la specificazione **int**, che significa appunto numero intero.

La prima riga dopo *main* specifica quali sono le variabili utilizzate nel programma e di che tipo
sono (**int**, cioè numeri interi). **int** a, b, s;

Nelle righe successive vengono scritte le istruzioni da eseguire, nella stessa sequenza prevista dall'algoritmo:
cin >> a;
cin >> b;
s = a + b;
cout << s;
cin>> indica la lettura dei dati dalla tastiera (*input*), l'istruzione *s = a + b* specifica che il risultato del calcolo *a + b* deve essere assegnato alla variabile *s*, e infine **cout**<< ordina la visualizzazione del risultato di *output* (variabile *s*) sullo schermo del computer.
I simboli >> per *cin* e << per *cout* rappresentano le frecce di direzione (verso) del flusso dei dati dalle unità standard di I/O alle variabili in memoria centrale e viceversa.

L'istruzione *cout* << visualizza il messaggio o il valore lasciando il cursore sulla stessa riga del video; l'output di una successiva istruzione *cout* << verrà visualizzato subito dopo. Per portare il cursore all'inizio della riga successiva del video (cioè per andare a capo nella visualizzazione), occorre aggiungere alla fine dell'istruzione il carattere di ritorno a capo rappresentato nel linguaggio C++ con la parola **endl** (*end of line*); l'output successivo comparirà all'inizio di una nuova riga.
Per esempio:

```
cout << "Risultato del calcolo = " << area << endl;
```

- **TIPI DI VARIABILI**

Tipo	Descrizione	Numero di bit utilizzati
short int	numeri interi compresi tra –32768 e +32767	16
int	numeri interi compresi tra -2.147.483.648 e 2.147.483.647	32
long int	numeri interi compresi tra -2.147.483.648 e 2.147.483.647	32
float	numeri in virgola mobile (*floating point*) in singola precisione	32
double	numeri in virgola mobile (*floating point*) in doppia precisione	64
char	un carattere	8
bool	può assumere solo i valori *true* (vero) o *false* (falso)	8

```
int eta;
int anni;
float statura;
double AreaCerchio;
double AreaTriangolo;
char risposta;
bool trovato;
```

```
const double PIGRECO = 3.14;

const char OK = 's';

const int SCONTO = 20;
```

```
int contatore = 0;
char risposta = 's';
```
I valori di tipo carattere sono racchiusi tra apici.

I dati di tipo alfanumerico formati da più caratteri si chiamano **stringhe** e possono essere considerati come una sequenza di dati di tipo *char*. Per facilitare la rappresentazione di questi dati, che sono di uso comune nella programmazione, il linguaggio C++ mette a disposizione il tipo **string**.

Per utilizzare questo tipo occorre includere nel programma la libreria **string** con la dichiarazione:
#include <string>

```
string nome;
string siglaprov = "TO";
```
I valori alfanumerici formati da più caratteri devono essere racchiusi tra virgolette.

- **Commenti**

```
// calcolo
s = a + b;
int eta;    // età di una persona
```
Il doppio // serve per inserire i commenti

- **Incremento variabili**

Nel linguaggio C++ si può anche scrivere: pag_stampate++;
per indicare, con l'**operatore ++**, un incremento unitario del valore di una variabile.

Esiste anche l'**operatore – –** per il decremento unitario: giorni_a_scadenza--;

Operatore	Utilizzo	Esempio	Istruzione equivalente
++	Incremento unitario	x++;	x = x + 1;
––	Decremento unitario	x––;	x = x – 1;
+=	Incremento	x += y;	x = x + y;
–=	Decremento	x –= y;	x = x – y;
*=	Moltiplicazione	x *= y;	x = x * y;
/=	Divisione	x /= y;	x = x / y;
%=	Resto della divisione tra interi	x %=y;	x = x % y;

Nelle istruzioni di assegnazione e nelle espressioni di calcolo possono poi comparire le **funzioni**, ossia strumenti predefiniti (*built-in*) del linguaggio che, ricevendo un valore, restituiscono un valore calcolato.
Tra le più usate, possono essere citate le seguenti:
. **pow(x, y)**, per indicare il calcolo della potenza di *x* con esponente *y* (*x* e *y* devono essere di tipo *double*, così come il risultato ottenuto dal calcolo)
. **sqrt(x)**, per indicare il calcolo della radice quadrata del numero x
. **ceil(x)**, per indicare il valore di un numero arrotondato all'intero superiore
. **floor(x)**, per ottenere un numero senza parte decimale.

Pertanto i programmi che usano queste funzioni devono contenere all'inizio la direttiva:
#include <cmath>
per l'inclusione della libreria **cmath** con i prototipi delle funzioni matematiche.

- **Operatori di confronto**

Nel programma C++, gli **operatori di confronto** si indicano con i simboli seguenti:

==	uguale
>	maggiore
<	minore
<=	minore o uguale
>=	maggiore o uguale
!=	diverso

- **I connettivi logici**

I **connettivi logici** che possono essere utilizzati in C++ sono:
&& indica l'operazione di congiunzione *(And)*
|| indica l'operazione di disgiunzione *(Or)*
! indica la negazione *(Not)*

- **Selezione o struttura di alternativa**

```
if (condizione) {
  istruzioni-a;
}
else {
  istruzioni-b;
}
```

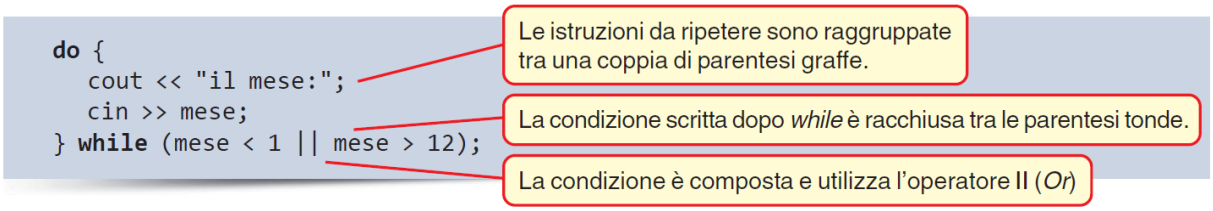
La condizione è racchiusa tra due parentesi tonde.

Se la condizione è vera, viene eseguita la sequenza *istruzioni-a*, altrimenti viene eseguita la sequenza *istruzioni-b*.

- **Struttura di ripetizione**

```
do {
  istruzioni;
} while (condizione);
```

La sequenza di istruzioni compresa tra *do* e *while* viene ripetuta tante volte, mentre la condizione scritta dopo *while* si mantiene vera: in altre parole la ripetizione termina quando la condizione scritta dopo *while* diventa falsa.



Solo quando il mese digitato dall'utente è compreso tra 1 e 12 la condizione diventa falsa e l'esecuzione del programma procede con l'istruzione successiva a *while*.

- **Ripetizione precondizionale**

```
while (condizione) {
   istruzioni;
}
```

Mentre la *condizione* si mantiene vera, viene eseguita la sequenza di istruzioni (racchiuse tra le parentesi graffe).
Per esempio se si vuole far entrare da tastiera un elenco di numeri, segnalando la fine dell'elenco con il numero 0, si può utilizzare il seguente frammento di programma C++.

```
int numero;

cout << "Inserire un numero (0=fine): ";
cin >> numero;
while (numero != 0) {
   ................;
   ................;
   cout << "Inserire un numero (0=fine): ";
   cin >> numero;
}
```

Se, invece, si vuole inserire un elenco di nomi, usando il carattere * per segnalare la fine dell'elenco, si può scrivere:

```
string nome;

cout << "Nome (*=fine): ";
cin >> nome;
while (nome != "*") {
   .............;
   .............;
   cout << "Nome (*=fine): ";
   cin >> nome;
}
```

Per controllare la fine dell'inserimento, il nome acquisito da tastiera viene confrontato con il carattere *: essendo *nome* una stringa, il carattere * è racchiuso tra una coppia di virgolette. In alternativa, considerando l'asterisco * come singolo carattere, si può scrivere:

```
while (nome[0] != '*' ) {

}
```

Poiché la stringa può essere considerata come una sequenza di caratteri, il confronto del carattere *, che segnala la fine dell'inserimento, deve essere fatto con il primo carattere della stringa (carattere della posizione 0 della stringa) indicato con *nome[0]*. In questo caso il carattere * è racchiuso tra una coppia di apici.

Struttura ripetizione con contatore

```
for (int i=min; i<=max; i++) {
  istruzioni;
}
```

Le istruzioni da eseguire in modo iterato possono essere una o più istruzioni. In presenza di più istruzioni è necessario racchiuderle tra una coppia di parentesi graffe.
Le istruzioni vengono ripetute tante volte, quante ne occorrono per portare il valore del contatore *i* dal valore iniziale (*min*) al valore finale (*max*), incrementandolo di 1 ad ogni ripetizione.
La dichiarazione del tipo di contatore può essere posta direttamente all'interno della struttura *for*. Per esempio, il seguente frammento di programma rappresenta un'iterazione per ottenere il doppio di 20 numeri inseriti da tastiera:

```
for (int i=1; i<=20; i++) {
   cin >> numero;
   cout << numero*2 << endl;
}
```

La struttura di ripetizione con contatore è una struttura derivata dalla struttura fondamentale di ripetizione, nel senso che il controllo della ripetizione delle istruzioni mediante un contatore può essere realizzato in modo equivalente usando una struttura di ripetizione postcondizionale *do … while* o una struttura precondizionale *while*, come si vede dal seguente esempio.

Queste tre strutture sono funzionalmente equivalenti:
struttura **for**

```
for(int i=1;i<=n;i++) {
  cout << "xxxxxxxxx" << endl;
}
```

struttura **do … while**

```
int i;
i = 1;
do {
  cout << "xxxxxxxxx" << endl;
  i++;
} while (i<=n);
```

struttura **while**

```
 int i;
 i = 1;
 while (i<=n) {
   cout << "xxxxxxxxx" << endl;
   i++;
 }
```

La struttura *for*, nel caso di ripetizione con un numero prefissato di volte, risulta più compatta e più semplice da rappresentare rispetto alla struttura *while*, in quanto dentro il *for* sono rappresentati l'assegnazione del valore iniziale del contatore e l'incremento del contatore stesso.
Si noti che, nel secondo esempio con la struttura di ripetizione postcondizionale, le istruzioni vengono eseguite almeno 1 volta , anche nel caso in cui il valore iniziale di *i* sia maggiore del valore finale.

Per esercitarsi online : https://www.onlinegdb.com/online_c++_compiler
http://cpp.sh/