Hello World

Hello World

Il **primo programma in C++**
il più semplice in assoluto è il famoso “Hello World!” (Ciao Mondo) che ha soltanto lo scopo di stampare a video la scritta per spiegare la sintassi basilare del linguaggio.

```
#include <iostream>  //libreria
using namespace std; // pazio dei nomi

int main (){ //inizio del main
  //corpo del main
  cout<< "Hello World!";
  return 0;
} //fine del main
```

Elementi Fondamentali:

- `#include` è una direttiva del preprocessore, un comando, che permette di richiamare le librerie standard del C++.

Senza librerie un programma non avrebbe a disposizione i comandi per eseguire anche le operazioni più semplici, come leggere un file o stampare a video una scritta.

- La funzione principale in un qualsiasi programma in C++ è il `main()` che, in questo caso, non ha parametri, ma può ricevere anche degli input da riga di comando. Il main è indispensabile ed unico e deve esserci sempre;
- Le **parentesi graffe** `{}` servono, invece, per delimitare blocchi di istruzioni, o come vengono abitualmente chiamate “statement”, che sono eseguite in ordine, da quella più in alto, giù fino all’ultima;
- Il **punto e virgola** `;`, invece, serve per “chiudere” un’istruzione, per far capire che dopo quel simbolo ne inizia una nuova.
- `cout`, adibita a stampare a video tutto quello che gli viene passato come argomento e che fa parte della libreria `<iostream>`, senza la cui inclusione, non avrebbe senso.

All’interno di un ftprogramma C++ possono essere inseriti dei **commenti**, basti sapere, per adesso, che esistono due modi:

- `//` – Tutto quello che sta a destra sulla medesima riga viene considerato commento e viene ignorato (ai fini dell’interpretazione del linguaggio) dal compilatore;
- `/* … */` – Tutto quello che è compreso tra i due asterischi viene considerato commento; questa forma viene utilizzata per commenti su più righe.

Variabili

Le variabili vengono definite da un **tipo** e da un **nome**.

esempio:

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

int main (){
  int x;  //int (valore intero) indica il tipo,
          //mentre x indica il nome della variabile
}
```

Il nome per identificare una variabile (o una funzione o una costante) viene comunemente riferito come **identificatore**. → nome = identificatore

Un **identificatore** è costituito da una o più lettere, cifre o caratteri e deve iniziare con una lettera o il carattere di sottolineatura (underscore “_”);

il C++ è **case-sensitive**, quindi si fa distinzione tra lettere maiuscole e lettere minuscole:

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

int main (){
  //sono due variabili diverse
  int x;
  int X;
}
```

Il **tipo** della variabile indica quale tipo di valori può assumere il contenuto della variabile stessa, si può ben capire che un tipo intero conterà soltanto dei numeri, mentre il tipo carattere conterrà solamente lettere dell’alfabeto, numeri e simboli;

esempio:

```
#include <stdlib.h>

int main (){
  int valore = 10;
  char carattere = 'c';
}
```

Tutte le variabili, prima di essere utilizzate, **devono essere dichiarate**, cioè deve essere detto al compilatore il tipo della variabile ed il suo nome (es. int x)

esempio:

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

int main (){
  x = 5; //errore!! -> la variabile x non e' stata dichiarata!
```

```
  int x;
  x = 5; //OK
}
```

Successivamente **la variabile deve essere inizializzata**, cioè le deve essere assegnato un valore, operazione che generalmente viene fatta contemporaneamente alla dichiarazione.

```
// solo dichiarazione int x; // inizializzazione x = 10;
```

```
// dichiarazione ed inizializzazione int y = 15;
```

Variabili - tipi

Nella tabella seguente sono mostrati i vari tipi (principali), la parola chiave C++ che li identifica (`char` per carattere, `int` per intero, etc.), la tipologia di informazioni che rappresentano ed il numero di byte necessari per la loro rappresentazione in C++:

Tipi di dichiarazione	Rappresentazione	Numero di byte
`int`	Numero intero	2 (16 bit)
`char`	Carattere	1 (8 bit)
`short`	Numero intero “corto”	2 (16 bit)
`long`	Numero intero “lungo”	4 (32 bit)
`float`	Numero reale	4 (32 bit)
`double`	Numero reale “lungo”	8 (64 bit)
`bool`	Confronto	1 (8 bit)

int

Il tipo **int** permette di rappresentare numeri interi. Possiamo dichiarare un `int` con due varianti: **short e long**, anche se in realtà un tipo `int` è già di per se’ un tipo `short`, mentre il la keyword `long` permette di estendere (utilizzando due byte in

più) il range dei valori che il tipo int può assumere, consentendoci di lavorare con grandi numeri. (attenzione ad usare operazione di divisione `/`).

I numeri interi, rappresentati da `int`, sono quelli “senza la virgola” o parti frazionate.

```
int x = 7;
int y = 3;
int z;
z = x / y; // z vale 2, cioè la parte intera della divisione tra 7 e 3
```

---

char

Il tipo `char` può contenere qualsiasi carattere definito secondo lo standard ASCII, quindi qualsiasi lettera (maiuscola o minuscola), cifra (da 0 a 9) e simbolo previsto dalla codifica.

È molto importante ricordare che un `char` può contenere uno ed un solo carattere. (più avanti vedremo come memorizzare degli insieme di caratteri, ossia stringhe).

Per **dichiarare una variabile char**, ad esempio inizializzandola con la lettera ‘`r`‘, basta scrivere:

```
char a = 'r';
```

---

float e double

I tipi `float` e `double` sono i cosiddetti numeri in virgola mobile, che rappresentano l’insieme dei numeri reali: con essi possiamo rappresentare numeri molto piccoli o numeri molto grandi, positivi e negativi e naturalmente con e senza decimali.

La differenza tra i due sta nel numero di bit riservati alla rappresentazione dei numeri, che si va a riflettere sul range di numeri e sul numero di cifre dopo la virgola che possiamo memorizzare. Quindi se abbiamo bisogno di particolare accuratezza, utilizziamo il tipo `double`.

```
double x = 7.0;
double y = 2.0;
double z;
z = x / y; // z vale 3.5
```

---

bool

i tipi `bool` sono valori che possono essere `true` oppure `false`.

```
bool x = true;
bool y = false;
```

```
#include <iostream>
using namespace std;
int main() {
    bool a = true;
    bool b = false;
    int c = a;
    bool d = 4;
    bool e = 1;
    bool f = c;
    if (a || b) cout << " OK";
    if (a && b) cout << " Impossibile";
    if (a == b) cout << " Impossibile";
    if (a != b) cout << " OK";
    if (a == true) cout << " OK";
    if (a) cout << " OK";
    cout << " " << a << b << c << d << e << f;
}
```

Operatori

Gli **operatori** si suddividono in:

- Operatori aritmetici;
- Operatori di confronto;
- Operatori logici;

Operatori aritmetici

Comprendono somma, sottrazione, moltiplicazione, divisione intera, divisione con modulo ecc. (in ordine `+`, `-`, `*`, `/`, `%`).

Operazioni con gli int	Simbolo	Esempio
Addizione	+	4 + 27 = 31
Sottrazione	–	76 – 23 = 53
Moltiplicazione	*	4 * 7 = 28

Operazioni con gli int	Simbolo	Esempio
Divisione intera	/	10 / 3 = 3 (3 è il n di volte divisibili senza resto)
Divisione con modulo	%	11 / 6 = 5 (5 è il resto della divisione)

Operazioni con i double	Simbolo	Esempio
Addizione	+	2.5 + 14.3 = 16.8
Sottrazione	–	43.8 – 12.7 = 31.1
Moltiplicazione	*	7.5 * 3.0 = 22.5
Divisione	/	5.0 / 2.0 = 2.5

Esistono poi degli operatori ai quali bisogna porre particolare attenzione, questi sono l'operatore di incremento (++) e decremento (--) post-fisso e suffisso.

esempio:

```
int x = 2;
x++; //equivale a scrivere x = x + 1;
```

Inoltre esistono metodi alternativi (e più veloci) rispetto alle banali (ma chiare) operazioni:

```
int x = 2;
int y = 3;
x += y;  //equivale a scrivere x = x + y;
```

Questa forma risulta essere più concisa, e per questo più facile da usare, ma bisogna porre attenzione nel suo uso perché potrebbe indurre in errori dovuti alla poca chiarezza del codice:

```
int y = 4;
y += 2; // j adesso vale 6

int x = 3; x *= y + 3;
// x adesso vale 27
// questo perché equivale a x=x*(y+3) e non x=(x*y)+3
```

Operatori di confronto

Operatori che permettono di verificare determinate condizioni, come ad esempio l'uguaglianza(`==`), la disuguaglianza(`!=`) oppure `>` , `<` , `>=` , `<=` ;

Simbolo	Significato	Utilizzo
==	uguale a	`a == b`
! =	diverso da	`a != b`
>	maggiore	`a > b`
>=	maggiore o uguale	`a >= b`
<	minore	`a < b`
<=	minore o uguale	`a <= b`

esempi:

a == b VERO ← → a è uguale a b.
FALSO altrimenti.

a > b VERO ← → a è maggiore strettamente di b.
FALSO altrimenti.

Operatori logici

Da utilizzare con le istruzioni condizionali ed iterative, come ad esempio l'operatore e/allo stesso tempo (`&&`), o/oppure(`||`)

Simbolo	Significato	Utilizzo
&&	AND logico	`a && b`
||	OR logico	`a || b`

esempi:

a && b VERO ← → a è vero e b è vero.
FALSO altrimenti.

a || b VERO ← → a è vero e/o b è vero.
FALSO altrimenti

Input - Output

In questa lezione esaminiamo gli strumenti che il **linguaggio C++**
ci offre per operare su input e output. In particolare vediamo come effettuare

semplici stampe a video a partire dalle informazioni inserite con la tastiera.

Per fare questo dobbiamo includere il file **<iostream>**
che mette a disposizione alcune funzioni predefinite per eseguire la lettura da un dispositivo di input (es. tastiera) o scrittura su un dispositivo di output (es. video);

i costrutti cui parleremo sono:

1. **cout <<**
2. **cin >>**

cout

L'istruzione per stampare a video più usata è la **cout**, che ci permette di controllare ciò che viene stampato, nel senso che permette di decidere cosa stampare ed in quale forma. La struttura di cout è la seguente:

```
cout << /*lista informazioni da stampare*/
```

Stampa sullo **stdout** (il video o la console in questo caso) la lista di argomenti conformemente alla stringa di formato specificata.

esempio:

```
int x = 10;
cout << "Il numero è " << x << endl;
/* l'output a video è "Il numero è 10" */
cout << "24.3";
/* l'output a video è 24.3 */
cout << "IVA = 20%" << endl;
/* l'output a video è, "IVA = 20%" */
```

Le **sequenze di escape** servono per rappresentare quei caratteri "speciali" presenti nella codifica ASCII e che non stampano nulla a video, ma permetto di introdurre all'interno di ciò che verrà stampato eventuali spaziature.

Tipo di opzione	Descrizione
`\n`	Ritorno a capo
`\t`	Tabulazione orizzontale
`\b`	Tabulazione verticale

Tipo di opzione	Descrizione
`\a`	Torna indietro di uno spazio
`\f`	Salto pagina

scanf

L'istruzione **cin** serve per leggere dallo **stdin** (generalmente la tastiera) una sequenza di caratteri (lettere o cifre) che verranno memorizzate all'interno di opportune variabili. Cin è, quindi, definita come segue:

```
cin >> /*lista informazioni da leggere*/
```

esempio:

```
#include <stdio.h>

int main(){
  int i;
  cin >> i;
  cout << i << endl;
}
```

Questo semplice programma sopra esposto serve solamente per leggere un numero da tastiera e ristamparlo a video;

Esercizi

Esercizio 1

realizzare un programma che chiede all'utente 2 valori interi e ne restituisce la somma.

> Esempio Output:
> inserisci primo valore: 3
> inserisci secondo valore: 5
> la somma è: 8

SOLUZIONE:

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

int main() {
    // dichiarazione variabili intere
    int x;
    int y;

    // lettura variabili
    cout << "inserisci primo valore: ";
    cin >> x;
    cout << "inserisci secondo valore: ";
    cin >> y;

    // operazioni
    int somma = x+y;

    // stampa somma
    cout << "la somma è: " << somma;

    return 0;
}
```

Esercizio 2

realizzare un programma che chiede all'utente un valore intero e ne restituisce il valore successivo (+1) e quello precedente (-1).

> Esempio Output:
> inserisci valore: 4
> valore: 4
> valore + 1: 5
> valore - 1: 3

SOLUZIONE:

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

int main(){
    // dichiarazione variabili
    int x;

    // lettura variabili
    cout << "inserisci valore: ";
    cin >> x;
```

```
    // operazioni
    int somma = x+1;
    int sottrazione = x-1;

    //stampa
    cout << "valore: " << x << endl;
    cout << "valore + 1: " << somma << endl;
    cout << "valore - 1: " << sottrazione << endl;

    return 0;
}
```

Esercizio 3

realizzare un programma che chiede all'utente 2 valori reali e ne restituisce la media.

> Esempio Output:
> inserisci primo valore: 6
> inserisci secondo valore: 5
> la media è: 5.5

SOLUZIONE:

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

int main(){
    // dichiarazione variabili
    float x;
    float y;

    // lettura variabili
    cout << "inserisci primo valore: ";
    cin >> x;
    cout << "inserisci secondo valore: ";
    cin >> y;

    float media = (x+y)/2;

    // operazioni
    cout << "la media è: " << media;

    return 0;
}
```