



UNIVERSITÀ
degli STUDI
di CATANIA

Stringhe e IO di base in C++

Corso di programmazione I

Corso di Laurea Triennale in Informatica

Prof. Giovanni Maria Farinella

Web: <http://www.dmi.unict.it/farinella>

Email: gfarinella@dmi.unict.it

Dipartimento di Matematica e Informatica

Libreria standard C++ per IO

Documentazione: <https://en.cppreference.com/w/cpp/io>

Libreria di IO basata sui flussi (**stream**).

Un flusso è una astrazione che rappresenta **un canale di comunicazione** per l'interazione delle applicazioni con l'ambiente in cui opera (SO e hardware).

Le applicazioni usano gli stream per scambiare dati attraverso la rete, inviare output a video, ricevere input da tastiera, etc.

La parte relativa alla gestione dei dispositivi è modellata mediante **astrazioni** (classi template e polimorfismo, in modo da il **codice per la gestione dello Input/Output sia indipendente** dal device).

Gli stream si possono **collegare** di volta in volta ad eventuali **periferiche** come tastiera o video, oppure si usano per leggere/scrivere files.

Flussi standard o predefiniti

- Standard output (oggetto ~~std::cout~~), generalmente associato al video.
- Standard input (oggetto ~~std::cin~~), generalmente associato alla tastiera.
- Standard error (oggetto ~~std::cerr~~), generalmente associato al video.

Usare i canali di IO predefiniti

Stampare un messaggio sullo standard output

```
#include <iostream>
using namespace std;
cout << "Insert a number between 1 and 10:" << endl;
```

~~std::cout~~

- “`<<`” **Operatore di inserimento**. Spinge i dati alla sua destra nel canale standard output;
- È necessario includere lo header `iostream`;
- `cout, cin, cerr` sono oggetti ed hanno *scope globale*: il programmatore può usari all'interno del suo programma senza doversi preoccupare di **inizializzare** alcunchè;

Usare i canali di IO predefiniti

Stampare un messaggio sullo standard output

```
#include <iostream>
using namespace std;
cout << "Insert a number between 1 and 10:" << endl;
```

- **L'operatore di inserimento** `<<` viene usato per **inviare allo standard output** l'argomento che sta alla sua destra.
- `endl` è una sorta di *manipolatore* di IO, inserisce un ritorno a capo, equivalente al carattere di escape '`\n`'.

NB: La clausola "using namespace std" permette di risolvere i nomi (classi, funzioni, costanti) definiti nel namespace std senza dover usare l'operatore risolutore di scope (ovvero std::cout).

Usare i canali di IO predefiniti

Ricevere dati dallo standard input

```
#include <iostream>
using namespace std;
int x;
cin >> x;
```

- L'oggetto `cin` ha scope globale e viene usato per prelevare dati dallo standard input;
- **l'operatore di estrazione `>>`** viene usato per leggere i dati dallo standard input.

Esempi svolti

A8_01_input_output.cpp

Output formattato

Con C++ per formattare lo output si usano i **manipolatori**.

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

Manipolatori di base

- **setprecision(n)**, controlla il numero di cifre (*n*) da stampare per un valore numerico inviato ad uno stream di output.
- **fixed**. Se usato prima di **setprecision()**, il numero *n* passato come argomento di **setprecision()** rappresenta il numero di cifre decimali da stampare (continua...)

Manipolatori di base

- **fixed** (...continua) Se fixed non settato, allora setprecision() usato per indicare il numero di cifre totali da stampare.
- **setw(n)**. Controlla l'ampiezza, ovvero il numero totale di caratteri (NB: eventuale padding!), del prossimo dato da stampare sullo stream.
- **scientific** stampa i numeri in virgola mobile in notazione scientifica

NB: Usare defaultfloat per resettare il comportamento dello standard output per la stampa dei numeri a quello originale.

NB: per usare il manipolatore `defaultfloat`, che è stato introdotto a partire dal C++11, bisogna indicare al compilatore che si stanno usando caratteristiche di quella versione di linguaggio.

Ad esempio, con il compilatore GNU GCC:

```
$ g++ -std=c++11 A8_02_format.cpp
```

Esempi svolti

A8_02_format.cpp

Stringhe e oggetto string

Una stringa è una **sequenza di caratteri** che termina con un carattere speciale ('\0') che rappresenta la fine della stringa

Stampa di un letterale stringa

```
cout << "Hello World" << endl;
```

In C++ si ha a disposizione la **classe string**, che permette di manipolare le stringhe agevolmente.

- Ad esempio, non ci si deve occupare del carattere che rappresenta la fine della stringa.

Stringhe e oggetto string

Oggetti string e IO

```
#include <string> // Header necessario!
string name="Pippo";
string your_name;
cout << "My name is " << name << endl;
cout << "Please write your name: " << endl;
cin >> your_name;
cout << "Your name is" << your_name << endl;
```

NB: L'oggetto string viene inizializzato automaticamente con la string vuota "".



Stringhe e oggetto string

Concatenare stringhe..

```
#include <string> // Header necessario!
string name="Pippo";
string your_name="Marco";
string all_names = name + "and" + your_name;
cout << "My name and your name is " << \
all_names << endl;
```

..ma NON due letterali consecutivamente

//**Errore di compilazione!**

```
string all_names="Pippo" + " and " + " Marco ";
```

Stringhe e oggetto string

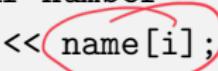
Lunghezza di una stringa

```
#include <string> // Header necessario!
string name="Pippo";
cout << "The length of the my name is " << \
name.length() << " chars!";
```



Individuare una stringa

```
string name="Pippo";
int i = 3;
cout << "The char number " << (i+1) << \
of my name is " << name[i];
//Il primo indice è zero!!
```

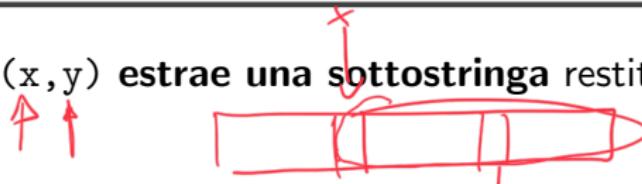


Stringhe e oggetto string

Sottostringhe

```
string name="Pippo";  
cout << "The name without the first letter: " \  
<< name << " " << name.substr(1,name.length()-2);
```

La funzione `substr(x,y)` **estrae una sottostringa** restituendo un oggetto string:



- il primo argomento è l'indice del primo carattere della sottostringa.
- il secondo argomento è la lunghezza della sottostringa da estrarre. Se non specificato, saranno considerati tutti i rimanenti caratteri fino alla fine della stringa originale.

Esempi svolti

A8_03_strings.cpp

stringstream

```
1 #include <sstream>
2 using namespace std;
3 stringstream ss; ←
4 ss << "Hello World" << endl; X
5 //oppure..
6 ss.str("Hello World!"); X
7 //inoltre..
8 cout << ss.str(); // estrazione del contenuto
```

La classe `stringstream` si può usare per immagazzinare dati (stringhe e numeri) in un buffer da usare successivamente.

`stringstream` si usa in entrambi i versi: estrazione (`>>`) e inserimento (`<<`);

stringstream

Comodo per **convertire i numeri nella loro rappresentazione in caratteri** .

```
1 #include <sstream>
2 using namespace std;
3
4 stringstream ss;
5 double g = 12345.6789;
6 ss << g << endl; // inserimento dei dati nel buffer
7 string my_number;
8 my_number = ss.str(); // stringa equivalente
```

12345.6789

stringstream

Comodo per **convertire stringhe in numeri**.

```
1 #include <sstream>
2 using namespace std;
3
4 stringstream ss;
5 double y;
6 ss << "123456.893"; //inserimento in ss
7 ss >> y; //estrazione in y
8 cout << "y=" << y << endl;
```

Esempi svolti

A8_04_stringstream.cpp

A8_04_stringstream2.cpp

FINE