



UNIVERSITÀ
DI PARMA

C++ introduzione

Alberto Ferrari





C++

STRUTTURA DI UN PROGRAMMA

```
/*  
 * First C++ program that says hello (hello.cpp)  
 */  
#include <iostream> // Needed to perform IO operations  
using namespace std;  
  
int main() { // Program entry point  
    cout << "hello, world" << endl; // Say Hello  
    return 0; // Terminate main()  
}
```

- insiemi di definizioni di nomi (classi, funzioni, costanti)
- le librerie standard mettono i loro nomi nel namespace **std**
- per usare queste definizioni occorre
 - specificare nel codice il nome completo di namespace
 - `std::cin ...`
 - oppure specificare il namespace per il nome con la direttiva
 - `using std::cin;`
 - oppure includere tutto il namespace con la direttiva
 - `using namespace std;`

- **cin**: input da console >> è l'operatore di estrazione
- è possibile concatenare più operazioni di lettura
 - getline(cin, line): lettura intera riga

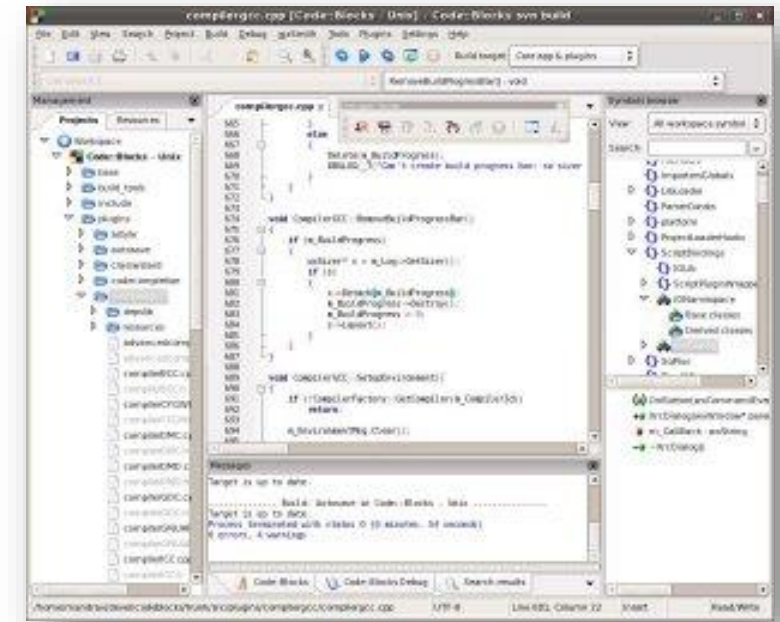
```
#include <iostream>
using namespace std;
int main() {
    string name;
    int age;
    cout << "Name and age?" << endl;
    cin >> name >> age;
    cout << "Hello, " << name << "." << endl;
    cout << "You're " << age << " years old." << endl;
    string fullName;
    cout << "Surname and Name: ";
    cin.ignore(); // ignores \n that cin >> str has lefted (if user pressed enter key)
    getline(cin, fullName);
    cout << "Bye " << fullName;
}
```

```
#include <iostream>
int main() {
    int integer1, integer2, sum;           // declaration
    std::cout << "Enter first integer\n";  // prompt
    std::cin >> integer1;                  // read an integer
    std::cout << "Enter second integer\n"; // prompt
    std::cin >> integer2;                  // read an integer
    sum = integer1 + integer2;            // assignment of sum
    std::cout << "Sum is " << sum << std::endl; // print sum
    return 0;                             // indicate that program ended successfully
}
```

```
#include <iostream>
using std::cout;      // program uses cout
using std::cin;       // program uses cin
using std::endl;     // program uses endl
int main() {
    int integer1, integer2, sum;           // declaration
    cout << "Enter first integer\n";      // prompt
    cin >> integer1;                       // read an integer
    cout << "Enter second integer\n";     // prompt
    cin >> integer2;                       // read an integer
    sum = integer1 + integer2;            // assignment of sum
    cout << "Sum is " << sum << endl;     // print sum
    return 0;                             // indicate that program ended successfully
}
```

```
#include <iostream>
using namespace std;
int main() {
    int integer1, integer2, sum;           // declaration
    cout << "Enter first integer\n";      // prompt
    cin >> integer1;                       // read an integer
    cout << "Enter second integer\n";     // prompt
    cin >> integer2;                       // read an integer
    sum = integer1 + integer2;            // assignment of sum
    cout << "Sum is " << sum << std::endl; // print sum
    return 0;                             // indicate that program ended successfully
}
```


- Integrated Development Environment
- Code::Blocks
 - open source, cross platform, free C, C++ and Fortran IDE
 - www.codeblocks.org
 - https://www3.ntu.edu.sg/home/ehchua/programming/howto/CodeBlocks_HowTo.html
- GCC, the GNU Compiler Collection
 - gcc.gnu.org
 - licenza copyleft (GNU GPL)
 - [Minimalist GNU for Windows](http://www.gnu.org/software/gcc/windows/)



- Cannot Compile any C/C++ Program after Installing CodeBlocks. Check:
 - You downloaded the CodeBlocks with "MinGW GNU C/C++ Compiler" (e.g., "codeblocks-10.05mingw-setup.exe").
 - Goto "Settings" menu ⇒ "Compiler..." ⇒ Select tab "Toolchain Executables" ⇒ Check the "Compiler's Installation Directory". It shall be set to the "MinGW" sub-directory of the CodeBlocks installation directory, e.g., "c:\Program Files\codeblocks\MinGW" suppose that CodeBlocks is installed in "c:\Program Files\codeblocks".
- Cannot Build or Run Program - Build/Run Buttons and Menu-Items are Grey and Not Selectable
 - A previous program is still running. You need to terminate the program by closing the output console window.
- Error: undefined reference to `WinMain@16'
 - Check that you have a main() function in your function. Check your spelling of main!

https://www3.ntu.edu.sg/home/ehchua/programming/howto/CodeBlocks_HowTo.html

- <https://www.geany.org/>
- *Geany is a small and lightweight Integrated Development Environment. It was developed to provide a small and fast IDE, which has only a few dependencies from other packages.*
- *Geany is known to run under Linux, FreeBSD, NetBSD, OpenBSD, MacOS X, AIX v5.3, Solaris Express and Windows.*
- *The code is licensed under the terms of the GNU General Public Licence.*



- linguaggio di programmazione basato sul paradigma orientato agli oggetti
- 1983 Bjarne Stroustrup (Bell Labs)
 - evoluzione del linguaggio C (*C with classes*)
- struttura di C++
 - nucleo del linguaggio
 - libreria standard STL (Standard Template Library)
- versioni:
 - **C++98**, C++03, C++11, **C++14**, C++17, (C++20)



<https://gcc.gnu.org/>

- GCC (GNU Compiler Collection, in origine GNU C Compiler) è un compilatore multi-target creato inizialmente da Richard Stallman, come parte del progetto GNU
- le versioni recenti sono incluse nelle principali distribuzioni del sistema operativo GNU/Linux e di molti altri sistemi
- nato inizialmente come un compilatore per il linguaggio C, dispone oggi di vari front end per altri linguaggi (Java, C++, Objective C ...)

Il progetto GNU è un progetto collaborativo lanciato nel 1983 da Richard Stallman per creare GNU: un sistema operativo Unix-like completo, utilizzabile esclusivamente utilizzando software libero. Il nome GNU è l'acronimo ricorsivo di "GNU's Not Unix".



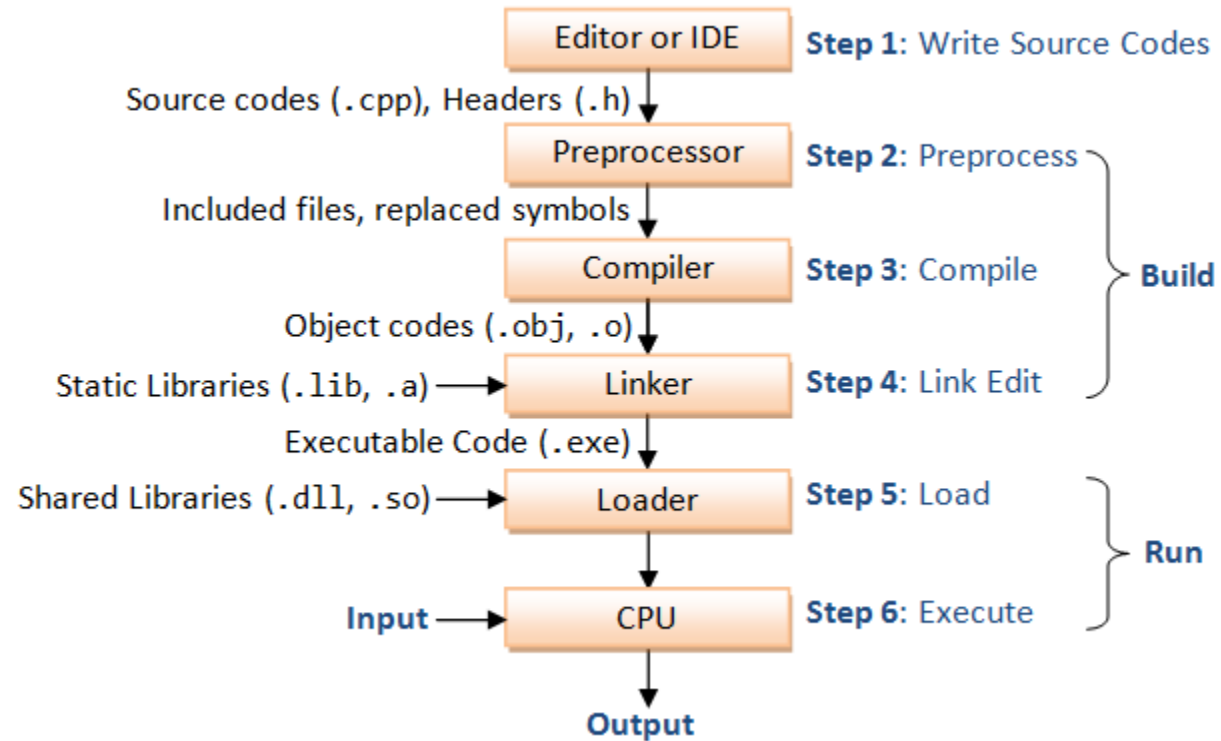
- o comando per visualizzare informazioni e versione del compilatore

```
alb@alb-UNIPR:~$ gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/6/lto-wrapper.exe
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion=6.2.0 --with-tune=generic --enable-checking=release --build=x86_64-linux-gnu --host=x86_64-linux-gnu --target=x86_64-linux-gnu
Thread model: posix
gcc version 6.2.0 20161018 (Ubuntu 6.2.0-7ubuntu11)
alb@alb-UNIPR:~$
```

```
Microsoft Windows [Versione 10.0.16299.192]
(c) 2017 Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\alber>gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=C:/Program Files (x86)/CodeBlocks/MinGW/bin/./libexec/gcc/mingw32/5.1.0/lto-wrapper.exe
Target: mingw32
Configured with: ../../../../src/gcc-5.1.0/configure --build=mingw32 --enable-languages=ada,c,c++,fortran,lto,objc,obj-c++ --enable-libgomp --enable-lto --enable-graphite --enable-libstdcxx-debug --enable-threads=posix --enable-version-specific-c-runtime-libs --enable-fully-dynamic-string --enable-libstdcxx-threads --enable-libstdcxx-time --with-gnu-ld --disable-werror --disable-nls --disable-win32-registry --disable-symvers --enable-cxx-flags='-fno-function-sections -fno-data-sections -DWINPTHREAD_STATIC' --prefix=/mingw32tdm --with-local-prefix=/mingw32tdm --with-pkgversion=tdm-1 --enable-sjlj-exceptions --with-bugurl=http://tdm-gcc.tdragon.net/bugs
Thread model: posix
gcc version 5.1.0 (tdm-1)

C:\Users\alber>
```



- gcc processa file di input attraverso 4 passi:
 - ***preprocessing***
 - ***compilation***
 - traduzione del codice sorgente ricevuto dal preprocessore in codice assembly
 - ***assembly***
 - creazione del codice oggetto
 - ***linking***
 - combinazione delle funzioni definite in altri file sorgenti o definite in librerie con la funzione main() per creare il file eseguibile

https://www3.ntu.edu.sg/home/ehchua/programming/cpp/gcc_make.html

- rimozione dei commenti
- interpretazioni di speciali direttive per il preprocessore denotate da "#" (normalmente all'inizio del codice)
 - #include - include il contenuto di un determinato file
 - es. #include<cmath>
 - #define
 - macro e costanti
 - #define ELEMENTS 100
 - sostituisce in tutto il programma 100 alla parola ELEMENTS (per convenzione tutto maiuscolo)
- esempio di utilizzo - comando cpp (c pre processing)
 - cpp test.cpp testPre.cpp

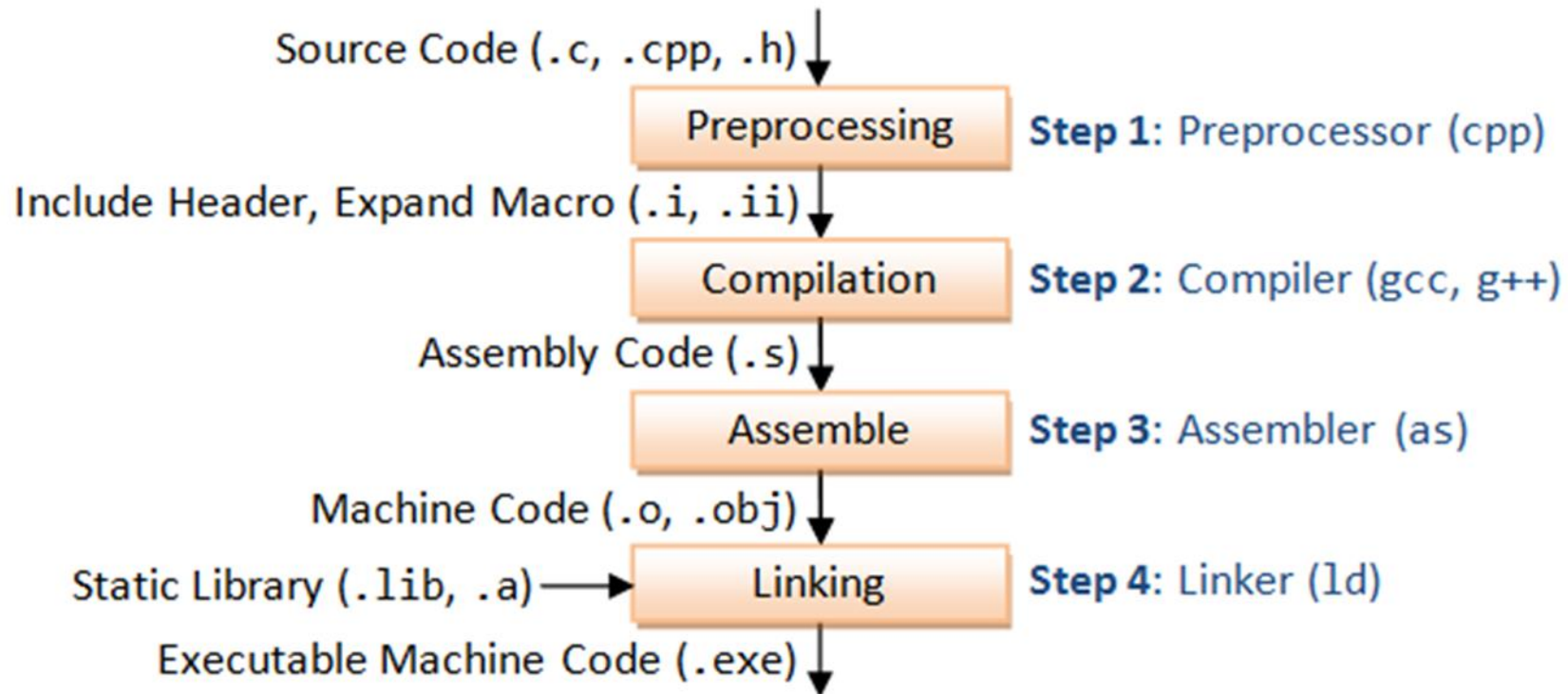
- converte il sorgente in linguaggio macchina
 - oggetti
 - progetti multi file
 - riferimenti/indirizzi non completi
- risultato: file oggetto (.o, .obj etc.)
 - è già linguaggio macchina

- unisce i file oggetto
 - risolve indirizzi/riferimenti
 - associa eventuali librerie esterne
- risultato: eseguibile
 - .exe, .com, nessuna estensione etc.

- g++ test.cpp
 - compila il file test.cpp e genera il file eseguibile
 - a.exe (Windows)
 - a.out (Linux)
- **Windows**
 - g++ -o test.exe test.c
 - con l'opzione -o si specifica il file di output
 - per eseguire il programma
 - test
- **linux**
 - g++ -o test test.c
 - con l'opzione -o si specifica il file di output
 - per eseguire il programma
 - ./test

- g++ -c test.c
 - l'opzione -c effettua la compilazione ma non il linking
 - viene generato il file oggetto test.o

- ***warning***
 - messaggi di avvertimento (attenzione!)
 - non interrompono la compilazione
 - avvisano della (possibile) presenza di irregolarità nel codice
 - inibire i messaggi di warning (sconsigliato)
 - opzione -w
 - g++ -w test.cpp
 - settare al massimo il livello di warning (consigliato)
 - opzione -wall
 - g++ -wall test.cpp
- ***error***
 - interrompono la compilazione
 - indicano errori che devono essere corretti



- step 1: ***preprocessing***
 - processa le direttive al compilatore (iniziano con #)
 - esempio #include e #define
 - le direttive devono essere processate prima della compilazione
- step 2: ***compile***
 - compilazione e generazione del codice oggetto (.obj, .o)
- step 3: ***link***
 - collega il codice compilato con altro codice compilato e librerie (.lib, .a) e produce il codice eseguibile (.exe)
- step 4: ***load***
 - carica il codice eseguibile in memoria
- step 5: ***run***
 - esegue il codice

C++

tipi di dato e variabili

<https://cppguide.readthedocs.io/>

- *Python* tipizzazione dinamica
- *C++* tipizzazione statica
 - le comuni variabili *non* sono *riferimenti*, ma *contenitori* di dati
- necessaria *dichiarazione* del tipo di dato
 - possibile *type inference* (`auto`)
- tipi principali: `int`, `float`, `double`, `bool`, `string`, `char`

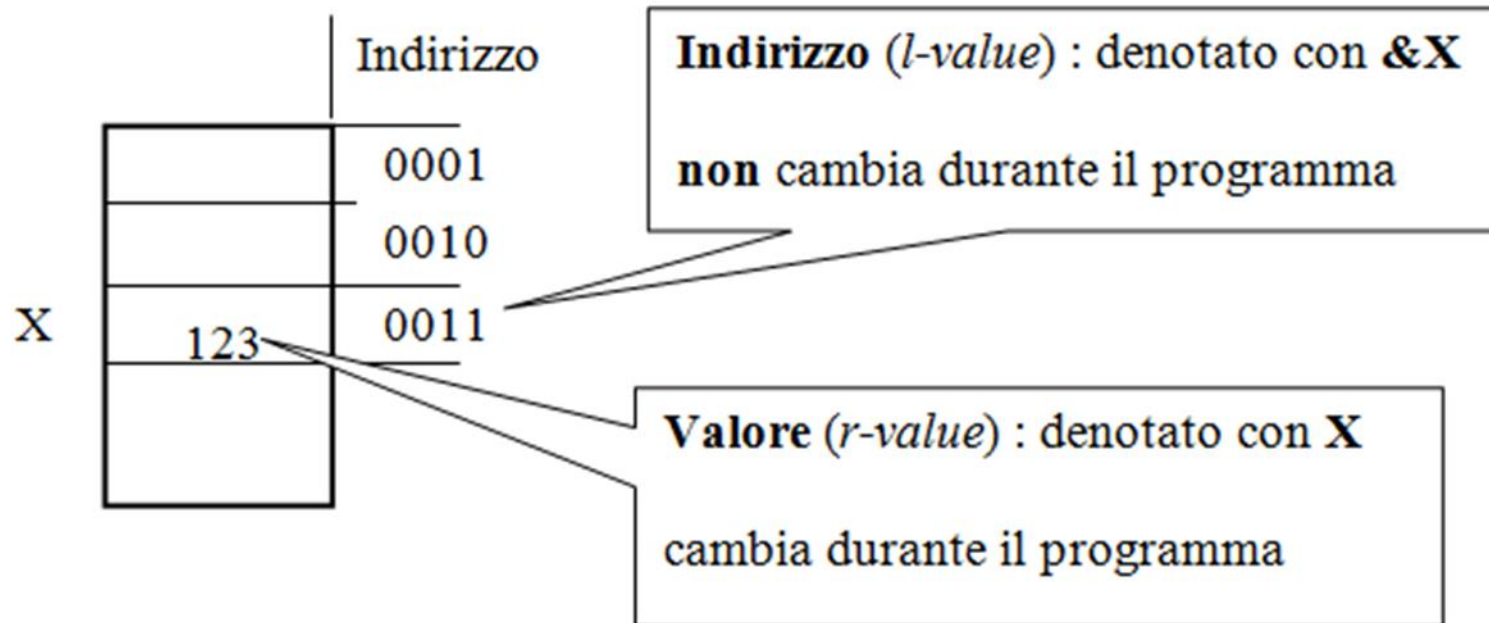
```
int x = 10;
double h = 3.7;
string s = "hello";

auto y = 5;           // type inference
auto k = 2.2;
auto t = "hola"s;    // compiler settings: -std=c++14
```

- si definisce variabile uno spazio identificato da un nome in cui è possibile scrivere, recuperare e manipolare dati nel corso del programma
- una variabile è caratterizzata da:
 - il suo valore: ***right value*** (*rvalue*)
 - il suo indirizzo: ***left value*** (*lvalue*)
 - lo spazio di memoria occupato
- la ***dichiarazione*** di una variabile associa un identificatore a un tipo e determina l'allocazione di un'area di memoria (***non assegna un valore alla variabile!***)
- è possibile inizializzare una variabile durante la dichiarazione
- per convenzione l'***identificatore*** (nome variabile) è scritto in minuscolo e dovrebbe essere mnemonico (***non può essere una parola riservata!***)

- una variabile può essere utilizzata (cioè è *visibile*) solo dopo la sua *definizione*
- le variabili definite in un *ambiente* (blocco o funzione) sono visibili in tutti gli ambienti in esso contenuti
- all'interno di uno stesso ambiente *non* è possibile definire *due variabili* con lo *stesso nome*
- in ambiente *distinti* è possibile definire variabili con lo stesso nome sia se sono dello stesso tipo sia se sono di tipo diverso
- se in un ambiente sono visibili più variabili con lo stesso nome, il nome si riferisce a quella la cui dichiarazione è «*più vicina*» al punto di utilizzo

- variabili globali (*permanenti*)
 - definite all'inizio del programma sono *visibili a tutte le funzioni* e il loro ciclo di vita termina con la terminazione del *programma*
- variabili locali (*temporanee*)
 - vengono *istanziate* al momento in cui vengono eseguite le prime istruzioni del *blocco* in cui sono contenute
 - le variabili vengono *distrusse* e la memoria ad esse associata viene rilasciata al *termine dell'esecuzione del blocco* in cui sono dichiarate



- è **necessario** che associare un tipo
 - ad ogni **variabile**
 - ad ogni **parametro** di funzione
 - ad ogni **valore restituito** da una funzione (tipo della funzione)
- i tipi di dato si dividono in 3 categorie
 - tipi di dato **scalari** (int, char, float, double, boolean ...)
 - tipi di dato **strutturati** definiti nelle librerie (string ...) o dall'utente (utilizzando struct o class)
 - **puntatori**
- il tipo specifica
 - la quantità di **memoria** che verrà allocata per la variabile (o risultato dell'espressione)
 - l'insieme dei **valori** che è possibile memorizzare in una variabile
 - la modalità di **interpretazione** di questi valori (schemi di bit - rappresentazione)
 - le **operazioni** che è possibile eseguire sui valori

- quando si dichiara una *variabile* è necessario **specificarne** il tipo in modo esplicito
 - **int** x; // non è specificato il **valore** che è **indefinito**
 - **double** h = 3.7;
 - **string** s = "hello";
- oppure utilizzare la parola chiave **auto** per indicare al compilatore di *dedurre* il tipo dall'inizializzatore (**type inference**)
 - **auto** y = 5; // type inference: C++11
 - **auto** k = 2.2;
 - **auto** s = string{"hello"}; // C++14: **auto** s = "hello"s;
- quando si dichiara una *funzione* è necessario **specificare** il tipo di **ciascun argomento** e del **valore restituito** (*void* se la funzione non restituisce alcun valore)
 - **double** media(**int** a, **int** b);

- **direttiva** al preprocessore
 - `#define PI_GRECO 3.141592`
 - il preprocessore sostituisce ogni occorrenza di `PI_GRECO` con `3.141592`
- **const**
 - `const int PI_GRECO 3.141592;`
 - il valore non può essere modificato nel corso del programma
 - `PI_GRECO = 2; // error: assignment of read-only variable 'PI_GRECO'`

```
#include <iostream>
int main() {
    int i; double d;
    float f; char c;
    bool b; int v[10];
    int *pi; double *pd;
    std::cout << "size of int:      " << sizeof i << " bytes " << std::endl;
    std::cout << "size of double:    " << sizeof d << " bytes " << std::endl;
    std::cout << "size of float:     " << sizeof f << " bytes " << std::endl;
    std::cout << "size of char:      " << sizeof c << " bytes " << std::endl;
    std::cout << "size of bool:      " << sizeof b << " bytes " << std::endl;
    std::cout << "size of int[10]:   " << sizeof v << " bytes " << std::endl;
    std::cout << "size of int*:      " << sizeof pi << " bytes " << std::endl;
    std::cout << "size of double*:   " << sizeof pd << " bytes " << std::endl;
    return 0;
}
```

```
size of int:      4 bytes
size of double:   8 bytes
size of float:    4 bytes
size of char:     1 bytes
size of bool:     1 bytes
size of int[10]: 40 bytes
size of int*:     4 bytes
size of double*:  4 bytes
```

<https://en.cppreference.com/w/cpp/language/types>

- operazioni su numeri:
 - + - * / %
 - ++ -- (*attenzione sono assegnamenti*)
 - attenzione: la **divisione tra interi** dà risultato intero (trunc)
 - assegnamento: = += -= ...
- confronti: > >= < <= != ==
 - **attenzione**: i confronti non si possono concatenare
- operazioni booleane (and, or, not): &&, ||, !
 - `cout << (3 < 5) << endl;` // output -> 1
 - `cout << (3 < 5 < 4) << endl;` // **output -> 1 (!!!)**
 - `cout << (3 < 5 && 5 < 4) << endl;` // output -> 0

- **string**: sequenza mutevole di byte (tipo char)
 - operazioni di confronto
 - concatenazione: +
- *apici doppi* per valori *string*, *singoli* per *char*

```
string sentence = "Lorem ipsum";  
sentence[6] = 'I';  
cout << sentence[6]; // 'I'  
  
int n = 5;  
string txt = to_string(n);  
int val = stoi(txt); // see also stod, stof...
```

```
#include <cstdlib>
#include <iostream>
#include <ctime>

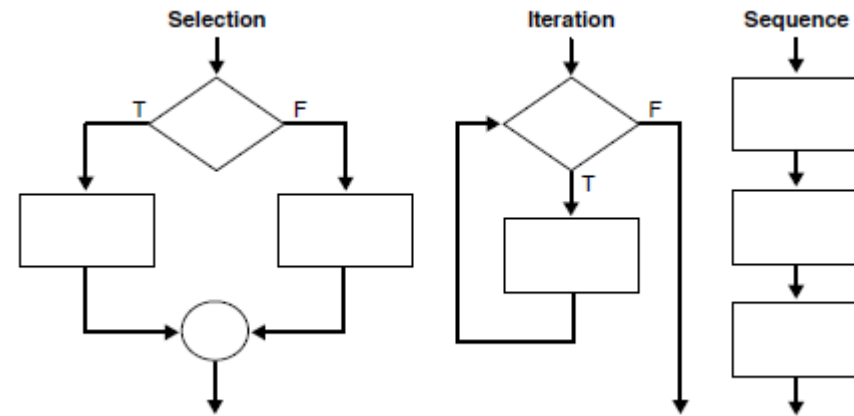
int main() {
    std::srand(std::time(nullptr)); // use current time as seed for random generator
    int random_variable = std::rand();
    std::cout << "Random value on [0 " << RAND_MAX << "]: " << random_variable << '\n';
    int range;
    std::cout << "Insert range ";
    std::cin >> range;
    std::cout << "Random value on [0" << range << "]: " << random_variable % range << '\n';
}
```

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

int main()
{
    double x = 10.25, result;
    result = sqrt(x);
    cout << "Square root of " << x << " is " << result << endl;

    return 0;
}
```

<http://www.cplusplus.com/reference/cmath/>



C++

strutture di controllo



```
a = input("First word? ")
b = input("Second word? ")

if a < b:
    print("The words are ordered")
elif a > b:
    print("The words are inverted")
else:
    print("The words are equal")
```

```
string a, b;
cin >> a >> b;
if (a < b) {
    cout << "The words are ordered" << endl;
} else if (a > b) {
    cout << "The words are inverted" << endl;
} else {
    cout << "The words are equal" << endl;
}
```



```
total = 0
count = 0

val = int(input("Val? (0 to finish) "))

while val != 0:
    total += val
    count += 1
    val = int(input("Val? (0 to finish) "))
)

if count != 0:
    print("The average is", total / count)
```

```
int val, total = 0, count = 0;
cout << "Val (0 to finish)? ";
cin >> val;
while (val != 0) {
    total += val;
    ++count;
    cout << "Val (0 to finish)? ";
    cin >> val;
}

if (count > 0) {
    cout << "Avg: " << total / float(count) << endl;
}
```

```
grocery = ["spam", "egg", "beans"]

grocery[0] = "sausage"      # replace an element

grocery.append("bacon")    # add an element to the end
grocery.pop()              # remove (and return) last element

grocery.insert(1, "bacon")  # other elements shift
removed = grocery.pop(1)    # remove (and return) element at
index

if "egg" in grocery:       # True, grocery contains "egg"
    grocery.remove("egg")  # remove an element by value
```



```
#include <vector>
// ...

vector<string> grocery = {"spam", "egg", "beans"};
cout << grocery[1] << endl;           // egg
cout << grocery.size() << endl;      // 3
grocery[0] = "sausage";              // replace an element

grocery.push_back("bacon")           // add an element to the end
grocery.pop_back()                   // remove last element

grocery.insert(begin(grocery) + 1, "bacon") // other elements shift
grocery.erase(begin(grocery) + 1)     // remove element at index
```





```
for (auto product : grocery) { // for each product in grocery
    cout << product << endl;
}

for (int i = 0; i < 5; ++i) { // for each i : 0 <= i < 5
    cout << i * i << endl;
}

int i = 0;
while (i < 5) { // equivalent while
    cout << i * i << endl;
    ++i;
}
```

algoritmi in python 3

esercizi



- 1.1 cerchio
 - chiedere all'utente il valore del raggio r di un cerchio
 - mostrare il valore dell'area e della circonferenza
 - se r è negativo, però, mostrare un messaggio d'errore
- 1.2 minore e maggiore
 - generare e stampare tre numeri interi casuali: a , b , c
 - ciascuno compreso tra 1 e 6
 - determinare e mostrare qual è il minore dei tre
*controllare prima di tutto se a è minore degli altri due
altrimenti controllare se b è minore di c
altrimenti ...*

```
3.141592653589793238462643383279
5028841971693993751058209749445923
07816406286208998628034825342117067
9821 48086 5132
823 06647 09384
46 09550 58223
17 25359 4081
2848 1117
4502 8410
2701 9385
21105 55964
46229 48954
9303 81964
4288 10975
66593 34461
284756 48233
78678 31652 71
2019091 456485 66
9234603 48610454326648
2133936 0726024914127
3724587 00660631558
817488 152092096
```



- 1.3 numero segreto
 - generare all'inizio del programma un numero “segreto” a caso tra 1 e 90
 - chiedere ripetutamente all'utente di immettere un numero, finché non indovina quello generato
 - ad ogni tentativo, dire se il numero immesso è maggiore o minore del numero segreto

○ 1.4 interesse composto

- dati di input: capitale iniziale, tasso d'interesse, durata investimento (anni)
- calcolare il capitale dopo ogni anno
- es. 100€ al 4.5%:

Anno 0:	100.00€
Anno 1:	104.50€
Anno 2:	109.20€
Anno 3:	114.12€ ...

○ 1.5 resistenze

- leggere, attraverso un ciclo, una sequenza di valori di resistenze elettriche
- la sequenza termina quando l'utente immette il valore 0
- alla fine, visualizzare la resistenza equivalente, sia nel caso di resistenze disposte in serie, che in parallelo

Formule utili:
 $R_s = \sum R_i$
 $1/R_p = \sum (1/R_i)$