Generazione di numeri pseudocasuali

Docente
Mario Perna
prof.perna.mario@darzo.net

A.S.
2025/2026
Materia
Informatica

Introduzione

I numeri casuali vengono usati in simulazioni, giochi, estrazioni e test.

- In C++ si utilizzano le funzioni **rand()** e **srand()** ed appartengono alla libreria **<cstdlib>**.
- Questi numeri sono **pseudo-casuali**: sembrano casuali, ma sono calcolati con formule matematiche.

La funzione rand()

- rand() restituisce un numero intero tra **0** e **RAND_MAX** (valore costante definito dal sistema).
- Esempio di utilizzo:

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

using namespace std;
int main() {
   int n = rand();
   cout << n << endl;
}
```

Se eseguito più volte, produrrà **sempre la stessa sequenza** di numeri.

Precisazioni

In C++, può capitare che pur non includendo la libreria <cstdlib>, la funzione rand() continui ad assolvere al suo dovere. Ma come è possibile ?....

Questo perché molti compilatori C++ (come GCC, Clang o MSVC) **includono implicitamente** alcuni header della libreria C.

Questa è **una conseguenza non standard** e **dipende dall'implementazione del compilatore**.

Inizializzare il generatore con srand()

- Per cambiare la sequenza di numeri si usa **srand()**, che imposta il **seme (seed)** del generatore.
- In genere si usa **l'orario attuale** come seme, tramite **time(0)** contenuto all'interno della libreria time()

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

using namespace std;
int main() {
   srand(time(0));
   int n = rand();
        cout << n << endl;
}
```

Perché serve srand(time(0)) ?

- Senza quell'istruzione, il generatore partirà **sempre dallo stesso seme predefinito (di solito 1)**. Questo significa che il programma genererà **sempre la stessa sequenza di numeri**, ogni volta che viene eseguito.
- Se invece utilizziamo **time(0)**, il seme cambierà continuamente, quindi anche la sequenza sarà **diversa a ogni esecuzione**.

Limitare l’intervallo

- rand() genera valori fino a RAND_MAX (**costante** definita nella libreria <cstdlib>), ma spesso servono numeri in un intervallo più piccolo.
- Si usa l’operatore modulo **%** per limitare l’intervallo:

```
int n = rand() % 10 + 1;  // genera numeri tra 1 e 10 compresi
```