Programmazione 2
e Laboratorio di Programmazione

Corso di Laurea in

Informatica

Università degli Studi di Napoli "Parthenope"

Anno Accademico 2023-2024

Prof. Luigi Catuogno

1

Informazioni sul corso

Docente	Luigi Catuogno **luigi.catuogno@uniparthenope.it**
Orario	Lun: 9:00-11:00 Mer: 11:00-13:00
Sede	Centro Direzionale Napoli **Aula Magna**
Ricevimento	Mer: 14:00-16:00 (previo appuntamento) Ufficio docente oppure Team: **cxxa3bo**

2

Libri di testo

Introduzione al linguaggio – costrutti e tecniche di base

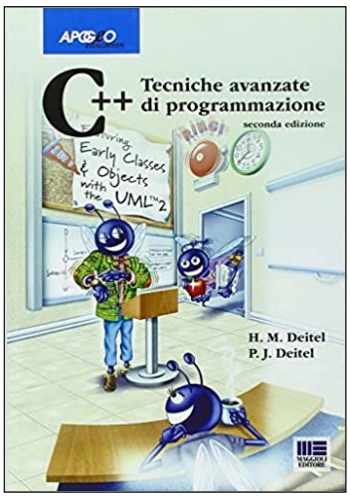
[FdP] H. M. Deitel, P. J. Deitel
C++ Fondamenti di programmazione
II ed. (2014) Maggioli Editore (Apogeo Education)
ISBN: 978-88-387-8571-9

3

Libri di testo

Tecniche avanzate e strutture dati elementari

[TAP] H. M. Deitel, P. J. Deitel
C++ Tecniche avanzate di programmazione
II ed. (2011) Maggioli Editore (Apogeo Education)
ISBN: 978-88-387-8572-6



4

Risorse on-line

Team del corso

Programmazione 2 AA 2023-24 - Prof. Catuogno
Comunicazioni, incontri e avvisi per il corso
Codice: **ftomzjx**

Piattaforma e-learning

Programmazione II e Laboratorio di Programmazione II - A.A. 2023-24
Materiale didattico, manualistica, esercitazioni.
URL: https://elearning.uniparthenope.it/course/view.php?id=2386

5

intermezzo

Il generatore di numeri *pseudocasuali*

6

Altre funzioni: Generatori di numeri casuali

- Il corredo di librerie standard del C/C++ fornisce un nutrito insieme di funzioni di uso generale che soddisfano le più frequenti esigenze della programmazione
- E' frequente la necessità di avere numeri *apparentemente* casuali (*pseudo casuali)*. La libreria fornisce diversi algoritmi per questo scopo.
- Tali algoritmi sono noti come PRNG (*pseudo random number generator*).

7

Altre funzioni: Generatori di numeri casuali

- Una classe PRNG di maggior successo (ormai quasi in disuso) è quello che utilizza espressioni *non lineari additive modulari «con feedback»* del tipo:

$$X_{k+1} = a \times X_k \; mod \; n$$

  - Dove n e a sono costanti scelte in fase di implementazione in modo da garantire che la sequenza di valori prodotti a partire da un certo punto $X_k$si ripeta dopo un periodo «accettabilmente» lungo. (Perciò *«pseudo»* casuali).

8

Altre funzioni: Generatori di numeri casuali

- Per utilizzare un PNRG occorre innanzitutto «posizionarsi» nella sequenza.
  - All'inizio *(una tantum*), l'utente inserisce un «seme» *(seed)* che altro non è che il valore scelto per X_k utilizzando una funzione di *inizializzazione*
    $$X_k \leftarrow input;$$
  - Successivamente, l'utente utilizza una funzione di *estrazione*, ogni volta che ha bisogno di un numero casuale (non c'è bisogno di inizializzare il PRNG ogni volta).
    - e.g. l'i-esimo valore estratto dopo l'inizializzazione ($i > 0$) è:
      $$X_{k+i} = aX_{k+i-1} \; mod \; n$$

9

Altre funzioni: Generatori di numeri casuali

- Il PRNG che utilizzeremo fornisce due funzioni di libreria:
  - `void` **`srand`***`(unsigned int seed)`* per l'inizializzazione
  - `int` **`rand`***`(void)`* per l'estrazione.

- Il modificatore di tipo *unsigned*, posto prima di un tipo numerico indica che i valori assunti dalle variabili sono rappresentati «senza segno».
- Il tipo *void* rappresenta i dati provenienti da un «insieme vuoto». Una funzione dichiarata come *void* non restituisce alcun valore.
- Se tra i parametri formali è presente la sola indicazione *void*, allora significa che la funzioni non prende alcun parametro.

10

Esempio: lancio dei dadi

Si scriva un programma C++ che simuli 15 lanci di un dado e visualizzi di volta in volta il valore estratto. Il programma chieda inizialmente un valore *seed* all'utente.

11

Esempio: lancio dei dadi

```
 1  #include <iostream>
 2  #include<iomanip>
 3  #include <cstdlib>
 4  using namespace std;
 5
 6  int main()
 7  {
 8          int dado;
 9          unsigned int seed;
10          cout << "inserisci il seed:";
11          cin >> seed;
12          srand(seed);
13
…  …
```

12

Esempio: lancio dei dadi

```
…
14          for (int i=1;i<=10;i++){
15                  dado=rand()%6+1;
16                  cout << "lancio "<<setw(3) << i << ":" <<setw(3)<< dado<< endl;
17          }
18  }
```

13

Esercizio: l'azzardo del *pari&dispari*

Si scriva un programma C++ che simuli una partita di 5 round a *pari&dispari* con due dadi.

L'utente dispone di una somma iniziale di 10 euro

1) a ogni round, il programma chiede all'utente di inserire la posta che intende scommettere (non superiore alla somma di cui dispone al momento)
2) Il suo pronostico (0=pari, 1=dispari)

Il programma lancia due dadi e somma i due numeri estratti.

1) Se l'utente ha indovinato, il suo *portafogli* è incrementato del valore della posta (in caso contrario è decrementato dello stesso valore)
2) Si procede a giocare un altro round a meno che: a) non si è già raggiunto il numero massimo di 5 round giocati; b) il giocatore non ha finito tutti i suoi soldi.

Al termine, il programma visualizza la cifra accumulata dall'utente.

14

Esercizio: l'azzardo del *pari&dispari*

```
Inserisci il seed: 3456
Hai 10 euro. Inserisci la posta: 11
Hai 10 euro. Inserisci la posta: 0
Hai 10 euro. Inserisci la posta: 10
Inserisci il tuo pronostico: 1
Numeri estratti: 3 e 1 hai perso.
La partita finisce qui. Hai 0 euro.
```

15

Esercizio: l'azzardo del *pari&dispari*

```
Inserisci il seed: 5678
Hai 10 euro. Inserisci la posta: 1
Inserisci il tuo pronostico: 1
Numeri estratti: 5 e 2 hai vinto.
Hai 11 euro. Inserisci la posta: 1
…(avanti così per 3 volte)…
Hai 9 euro. Inserisci la posta: 5
Inserisci il tuo pronostico: 1
Numeri estratti: 3 e 4 hai vinto.
La partita finisce qui. Hai 14 euro.
```

16