Programmazione 2 e Lab. di programmazione 2

Corso di Laurea in Informatica - Anno Accademico 2022-23

Docenti
Prof. Angelo Ciaramella
[angelo.ciaramella@uniparthenope.it]
Prof. Luigi Catuogno
[luigi.catuogno@uniparthenope.it]
Tutor
Dott. Antonio Vanzanella
[antonio.vanzanella@studenti.uniparthenope.it]

1

Il Linguaggio C++

(per programmatori C)

2

Input/Output da console

seconda parte

3

I/O formattato: manipolatori di stream

I manipolatori di tipo *sticky* producono il loro effetto indefinitamente, fino a che non vengono nuovamente impostati o *annullati*.
Per esempio: **`fixed e setprecision`**

Gli altri (per esempio **`endl`**) funzionano *solo una volta.* Vanno quindi inviati ogni volta che ce n'è bisogno.

4

I/O formattato: manipolatori di stream

Una volta manipolato lo stream **`cout`** nel modo seguente:

```
cout  << fixed << setprecision(4);
```

Una volta manipolato lo stream **`cout`** nel modo seguente:

```
cout  << setprecision(6) << defaultfloat;
```

5

Esempio: visualizzare dati in tabelle

Dati due array: un'array di stringhe contenente un elenco di nomi di valute e un array di reali contenenti i rispettivi cambi contro Euro, desideriamo visualizzare in una tabella su ciascuna riga il nome della divisa e il suo corrispondente importo in Euro.

6

Esempio: visualizzare dati in tabelle

Valute_v1.cpp

```
 1  #include<iostream>
 2  using std::cout;
 3  using std::endl;
 4
 5  const char *valute[6]={"Dollaro US","Dollaro Can","LST","Franco Svizzero","Yen","Corona
 6  Svedese"};
 7  const float cambi[6]={0.95,0.69,0.89,1.01,144.66,11.32};
 8  int main()
 9  {
10          cout << " *** Listino Cambi 7/3/2023 ***" << endl;
11
12          for (int i=0;i<6;i++){
13                  cout <<valute[i] <<" | "<<cambi[i]<<endl;
14          }
15  }
```

7

Esempio: visualizzare dati in tabelle

```
 *** Listino Cambi 7/3/2023 ***
Dollaro US | 0.95
Dollaro Can | 0.69
LST | 0.89
Franco Svizzero | 1.01
Yen | 144.66
Corona Svedese | 11.32
```

Per visualizzare i dati in output in una tabella, è necessario che:

- Ciascun dato sia visualizzato in uno spazio (campo) di dimensione prefissata;
- Sia possibile decidere l'allineamento dei dati all'interno del campo

8

I/O formattato: campi e allineamenti

Per visualizzare i dati in output in una tabella, è necessario che:
- Ciascun dato sia visualizzato in uno spazio (campo) di dimensione prefissata;
- Sia possibile decidere l'allineamento dei dati all'interno del campo

Desideriamo che le divise siano rappresentate in un campo di larghezza sufficiente a contenere il nome più lungo e uguale per tutte le righe;

Ciascun nome deve essere allineato al lato sinistro del campo.

Analogamente, gli importi di cambio contro l'Euro, devono essere rappresentati in un campo appositamente dimensionato e allineati a destra.

9

I/O formattato: campi e allineamenti

Per visualizzare i dati in output in una tabella, è necessario che:
- Ciascun dato sia visualizzato in uno spazio (campo) di dimensione prefissata;
- Sia possibile decidere l'allineamento dei dati all'interno del campo

Per questo scopo, si utilizzano i manipolatori **`setw()`**, **`left`** e **`right`**

Tutti questi manipolatori sono visibili nel namepace **`std`**

10

I/O formattato: campi e allineamenti

Per esempio nella seguente espressione:

```
cout << setw(5) << right << X << endl;
```

Il valore della variabile **X** viene rappresentato all'interno di un'area larga cinque caratteri:

- Se **X** è più corta di 5 caratteri, il compilatore visualizza il contenuto della variabile allineandolo al limite destro del campo e riempendo con uno *spazio* la parte di campo non utilizzata a sinistra.
- Se **X** è più lunga di 5 caratteri, nulla accade, il compilatore impiegherà il numero di caratteri necessario a visualizzare la **X**

11

I/O formattato: campi e allineamenti

Per esempio nella seguente espressione:

```
X=12;
cout << "X=" << setw(5) << right << X << endl;
```


```
cout << "X=" << setw(4) << left << X << endl;
```

12

Esempio: visualizzare una tabella *#2*

Valute_v2.cpp

```
1  #include<iostream>
2  using std::cout;
3  using std::endl;
4  using std::fixed;
5  using std::left;
6  using std::right;
7  #include<iomanip>
8  using std::setprecision;
9  using std::setw;
10
11 const char *valute[6]={"Dollaro US","Dollaro Can","LST","Franco Svizzero","Yen","Corona
12 Svedese"};
13 const float cambi[6]={0.95,0.69,0.89,1.01,144.66,11.32};
…  …
```

Dichiariamo l'uso dei vari manipolatori *as usual*

Esempio: visualizzare una tabella *#2*

Valute_v2.cpp

```
…  …
14 int main()
15 {
16         cout << " *** Listino Cambi 7/3/2023 ***" << endl;
17         cout << fixed << setprecision(2);
18         for (int i=0;i<6;i++){
19                 cout <<"|"<< left << setw(20) << valute[i]<<"|";
20                 cout << right << setw(8)<<cambi[i]<<"|"<<endl;
21         }
22 }
```

Impostiamo la precisione dei numeri rali (una volta per tutte)

Impostiamo allineamento e larghezza del campo prima di ogni singolo utilizzo.

Esempio: visualizzare una tabella *#2*

```
 *** Listino Cambi 7/3/2023 ***
|Dollaro US          |    0.95|
|Dollaro Can         |    0.69|
|LST                 |    0.89|
|Franco Svizzero     |    1.01|
|Yen                 |  144.66|
|Corona Svedese      |   11.32|
```

15

Esercizio: la caduta dei gravi *#2*

Modificare il programma CadutaDeiGravi.cpp in modo da formattare l'output in maniera più ordinata e gradevole.

```
 *** Caduta dei gravi ***
Inserisci l'altezza (h>0): 56.7
Inserisci delta (d>0): 0.5
+--t--+---altezza dal suolo----+
| 0.00|           56.7000007629|
| 0.50|           55.4737510681|
| 1.00|           51.7950019836|
| 1.50|           45.6637496948|
| 2.00|           37.0800018311|
| 2.50|           26.0437488556|
| 3.00|           12.5550003052|
| 3.50|           -3.3862533569|
```

16

Esercizio: la caduta dei gravi *#2*

Valute_v2.cpp

```
 1  #include<iostream>
 2  using std::cin;
 3  using std::cout;
 4  using std::endl;
 5  using std::fixed;
 6  using std::left;
 7  using std::right;
 8  #include<iomanip>
 9  using std::setprecision;
10  using std::setw;
11
12  int main()
13  {
…   …
```

17

Esercizio: la caduta dei gravi *#2*

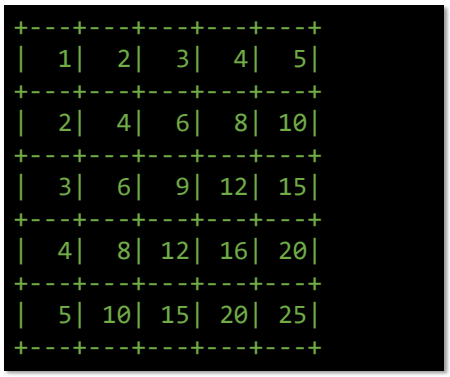
Valute_v2.cpp

```
14          const float g=9.81;
15          float x,h,t=0,d;
16          cout << "*** Caduta dei gravi ***" << endl;
17          cout<< "Inserisci l'altezza (h>0): ";
18          cin >> h;
19          cout<< "Inserisci delta (d>0): ";
20          cin >> d;
21          x=h;
22          cout <<"+--t--+---altezza dal suolo----+"<<endl;
23          cout << fixed << setprecision(2) << right;
24          while (x>0) {
25                  x=h-(g*t*t)/2;
26                  cout << "|"<< setprecision(2)<<setw(5) << t << "|";
27                  cout << setprecision(10)<<setw(24)<<x<<"|"<<endl;
28                  t=t+d;
29          }
30  }
```

18

Esercizio: la tavola pitagorica *5x5*

Scrivere un programma in C++ che calcoli e visualizzi la tavola pitagorica (5x5) formattata come in figura.



```
+---+---+---+---+---+
|  1|  2|  3|  4|  5|
+---+---+---+---+---+
|  2|  4|  6|  8| 10|
+---+---+---+---+---+
|  3|  6|  9| 12| 15|
+---+---+---+---+---+
|  4|  8| 12| 16| 20|
+---+---+---+---+---+
|  5| 10| 15| 20| 25|
+---+---+---+---+---+
```

19

intermezzo

Il generatore di numeri *pseudocasuali*

20

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*).

21

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).

22

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:
      $$X_{k+i} = aX_{k+i-1} \; mod \; n$$

23

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.

24

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.

25

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
…   …
```

26

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  }
```

27

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.

28

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.
```

29

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.
```

30

Esercizio: l'azzardo del *pari&dispari*

```
1  #include<iostream>
2  #include<cstdlib>
3  using namespace std;
4
5  int main()
6  {
7          unsigned int seed;
8          int d1,d2,soldi=10;
9          int posta, scelta;
10
11         cout << "Inserisci il seed: ";
12         cin >> seed;
13         srand(seed);
…  …
```

31

Esercizio: l'azzardo del *pari&dispari*

Condizioni affinchè si continui a giocare sono: che non siano già stati effettuati 5 lanci (i<5) e che il giocatore abbia ancora denaro per coprire le sue scommesse (soldi>=0)

```
14
15         for(int i=0;i<5 && soldi>=0;i++)
16         {
17                 do {
18                         cout << "Hai "<< soldi << "Euro. ";
19                         cout << "Inserisci la posta: ";
20                         cin >> posta;
21                 } while (posta>soldi || posta >= 0);
22
23                 cout << "Inserisci il tuo pronostico: ";
24                 cin >> scelta;
…                  …
```

32

Esercizio: l'azzardo del *pari&dispari*

Il programma rifiuta la posta se questa eccede le disponibilità del giocatore (posta>soldi) o se è minore o uguale a zero. In tal caso ripete indefinitamente al giocatore di ripeterne l'input.

```
…
15        for(int i=0;i<5 && soldi>=0;i++)
16        {
17              do {
18                    cout << "Hai "<< soldi << "
19                    cout << "Inserisci la posta:
20                    cin >> posta;
21              } while (posta>soldi || posta <= 0);
22
23              cout << "Inserisci il tuo pronostico: ";
24              cin >> scelta;
…               …
```

33

Esercizio: l'azzardo del *pari&dispari*

Lancio dei due dadi

Controlla il pronostico

```
25              d1=rand()%6+1;
26              d2=rand()%6+1;
27              cout << "Numeri estratti: "<<d1<<" e "<< d2;
28
29              if ((d1+d2)%2==scelta) {
30                    cout << "hai vinto."<<endl;
31                    soldi=soldi+posta;
32              } else {
33                    cout << " hai perso."<<endl;
34                    soldi=soldi-posta;
35              }
36        } // end for
37        cout << "La partita finisce qui. Hai "<<soldi<<" euro."<<endl;
38  }
```

34

Esercizio: *«Scusi, ha da cambiare?»*

Si scriva un programma C++ che che chieda all'utente di inserire un importo di danaro in una variabile intera e restituisca il numero di banconote e monete necessarie per comporlo, scegliendo tra i seguenti tagli:

500, 200, 100, 50, 20, 10, 5, 2, 1

Esempio: *l'importo 609 è composto da 1 pezzo da 500, 1 pezzo da 100, 1 pezzo da 5, 2 pezzi da 2.*

35

Esercizio: *«Scusi, ha da cambiare?»*

Si scriva un programma C++ che chieda all'utente di inserire un importo di danaro in una variabile intera e restituisca il numero di banconote e monete necessarie per comporlo, scegliendo tra i seguenti tagli:

500, 200, 100, 50, 20, 10, 5, 2, 1

Suggerimento: *E' opportuno che il programma dichiari un array di 9 interi contenente i tagli e che confronti l'importo inserito dall'utente con ciascun taglio dal maggiore al minore. Un ulteriore array conterrà il numero di pezzi utilizzato per ciascun taglio.*

36