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

Esercizi svolti

2

Input/Output da *console*

3

Esercizio: *a raccontare le favole*

- Si scriva un programma che:
  - Chieda all'utente di immettere:
  - L'anno corrente *(oggi)*
  - L'anno di nascita di Cappuccetto Rosso *(nata_cr)*
  - Il numero di focaccine che C.R. porta nel cestino *(focaccin)*
  - Il numero di focaccine che il Lupo sottrae a cappuccetto rosso *(focacciout)*
- Visualizzi la storiella mostrata in seguito tenendo conto di alcune condizioni che si verificano in base al valore dell'input;

4

Esercizio: *a raccontare le favole*

C'era una AAAA di nome Cappuccetto Rosso che si recava dalla sua nonnina al di là del bosco per portarle BBBB focaccine calde calde. Durante il tragitto, il Lupo Cattivo rubò a Cappuccetto Rosso ben CCCC focaccine.
Giunta infine dalla nonna, cappuccetto le porse il cestino e la nonna disse:
a) «grazie nipotina mia per queste DDDD focaccine» se nel cesto ci sono ancora focaccine
b) «grazie nipotina mia per avermi fatto visita!» se nel cesto non c'è più alcuna focaccina.

- Al posto di AAAA il programma deve scrivere «bambina» se l'età di C.R. è inferiore ai 12 anni, «ragazza» se è superiore ai 12 anni ma inferiore ai 20 e «donna» negli altri casi;
- Al posto di BBBB e CCCC vanno sostituiti corrispondenti valori ottenuti in input
- Al posto di DDDD deve essere visualizzato il numero di focaccine residue

5

Esercizio: *a raccontare le favole*

```
 1  int main()
 2  {
 3          int oggi, nata_cr, focaccin, focaccout, anni_cr, residuo;
 4
 5          cout << "Immetti anno corrente: ";
 6          cin >> oggi;
 7          cout << "Anno di nascita di C.R.: ";
 8          cin >> nata_cr;
 9          cout << "Focaccine in input: ";
10          cin >> focaccin;
11          cout << "Focaccine in output: ";
12          cin >> focaccout;
13          cout <<endl<<endl<<endl;;
 …  …
```

6

Esercizio: *a raccontare le favole*

```
…  …
14         anni_cr=oggi-nata_cr;
15         residuo=focaccin-focaccout;
…  …
```

7

Esercizio: *a raccontare le favole*

Al punto AAAA, occorre effettuare una «catena» di confronti con l'età di C.R. Il confronto che ha esito positivo, conduce alla visualizzazione della parola corrispondente.

```
16         cout << "C'era una ";
17         if (anni_cr <= 10)
18               cout << "bambina ";
19         else
20               if (anni_cr <=20)
21                     cout << "ragazza ";
22                     else
23                           cout << "donna ";
24
25
26         cout <<"di nome Cappuccetto Rosso che si recava… "<<endl;
```

8

Esercizio: *a raccontare le favole*

Al punto BBBB, occorre scrivere il valore delle «focaccine in input». Non ci sono controlli di alcun tipo;

```
27        cout << "per portarle " << focaccin << " focaccine..." << endl;
```

Al punto CCCC, analogamente:

```
28  cout << "il Lupo Cattivo le rubò " << focaccout << " focaccine." <<
    endl;
```

9

Esercizio: *a raccontare le favole*

Al punto DDDD, occorre scegliere quale messaggio visualizzare, in base al fatto che le focaccine residue siano o meno in quantità maggiore di zero.

```
29        cout << "La nonna disse grazie nipotina mia per ";
30        if (residuo > 0) {
31              cout << "queste "<< residuo;
32              cout << "focaccine!"<<endl;
33        }
34        else
35              cout << "avermi fatto visita!"<<endl;
36  }
```

10

I/O formattato: manipolatori di flusso

11

Esercizio: manipolatori di stream

Si modifichi il programma **`PiGrecoQuarti_v1.cpp`** in modo che oltre che il numero di iterazioni, chieda all'utente anche il numero di cifre decimali da visualizzare del risultato.

```
*** Calcolo di Pi greco ***
    Inserisci il max numero di iterazioni:3000000
    Inserisci il numero di cifre decimali:12
Pi=3.141592986923
```

12

Esercizio: manipolatori di stream

PiGrecoQuarti_v2.cpp

```
 1  #include<iostream>
 2  using std::cin;
 3  using std::cout;
 4  using std::endl;
 5  using std::fixed;
 6  #include<iomanip>
 7  using std::setprecision;
 8
 9  int main()
10  {
11          int k,ncifre;
12          double pi=0.0,num=1;
13          cout<<"*** Calcolo di Pi greco ***"<<endl;
14          cout<<"    Inserisci il max numero di iterazioni:";
15          cin >> k;
16          cout<<"    Inserisci il numero di cifre decimali:";
17          cin>>ncifre;
 …  …
```

13

Esercizio: manipolatori di stream

PiGrecoQuarti_v2.cpp

```
 …  …
18          for (int i=0;i<=k;i++){
19                  pi+=num/(2*i+1);
20                  num*=-1;
21          }
22          pi=pi*4.0;
23          cout << fixed << setprecision(ncifre);
24          cout <<"Pi="<<pi<<endl;
25  }
```

14

Esercizio: la caduta dei gravi

Un peso viene lasciato cadere verticalmente da un'altezza *h* e desideriamo sapere a che altezza si trova ogni *d* secondi fino a che non tocca il suolo.

Si scriva un programma in C++ che:

1) chieda in input due numeri *h* e *d* in doppia precisione;
2) calcoli e visualizzi l'altezza a cui si trova, ogni d secondi,

$$x(t) = -\frac{1}{2}gt^2$$

$g = 9.81\ m/s^2$

15

Esercizio: la caduta dei gravi

CadutaDeiGravi.cpp

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

Il manipolatore *non parametrico* **`fixed`** è dichiarato in **`iostream`**;

Il manipolatore *parametrico* **`setprecision`** è dichiarato in **`iomanip`**;

Per utilizzare le funzioni matematiche - *e.g.* **`pow()`**

16

Esercizio: la caduta dei gravi

CadutaDeiGravi.cpp

```
12          const float g=9.81;
13          float x,h,t=0,d;
14          cout << "*** Caduta dei gravi ***" << endl;
15          cout<< "Inserisci l'altezza (h>0): ";
16          cin >> h;
17          cout<< "Inserisci delta (d>0): ";
18          cin >> d;
19
20          x=h;
…   …
```

Per memorizzare la costante di accelerazione di gravità utilizziamo una *variabile a sola lettura* con il modificatore **`const`**

17

Esercizio: la caduta dei gravi

Impostiamo la precisione fissa. Resterà semre attiva fino a diversa disposizione.

Per i tempi utilizziamo una precisione di 2 cifre. Una volta impostata resta sempre valida. Se occorre una precisione diversa…

```
21          cout << fixed;
22          while (x>0) {
23                  x=h-(g*pow(t,2))/2;
24                  cout << "t="<< setprecision(2)<< t << ": ";
25                  cout << setprecision(10) << x << endl;
26                  t=t+d;
27          }
28  }
```

… occorre impostarla. Da qui in poi, la precisione è 10 «fino a nuovo ordine».

18

Esercizio: la caduta dei gravi

Esempio:
Si immettano $h = 56.70$ e $d = 0.5$

```
*** Caduta dei gravi ***
Inserisci l'altezza (h>0): 56.7
Inserisci delta (d>0): 0.5
t=0.00: 56.7000007629
t=0.50: 55.4737510681
t=1.00: 51.7950019836
t=1.50: 45.6637496948
t=2.00: 37.0800018311
t=2.50: 26.0437488556
t=3.00: 12.5549983978
t=3.50: -3.3862519264
```

19

Esercizio: la caduta dei gravi

Esempio:
Si immettano $h = 56.70$ e $d = 0.5$

```
*** Caduta dei gravi ***
Inserisci l'altezza (h>0): 56.7
Inserisci delta (d>0): 0.5
t=0.00: 56.7000007629
t=0.50: 55.4737510681
t=1.00: 51.7950019836
t=1.50: 45.6637496948
t=2.00: 37.0800018311
t=2.50: 26.0437488556
t=3.00: 12.5549983978
t=3.50: -3.3862519264
```

L'impatto avviene al tempo:

$$t = \frac{\sqrt{2gh}}{g} = 3.39994\ s$$

20

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|
```

21

Esercizio: la caduta dei gravi *#2*

CaduteDeiGravi2.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  {
 …  …
```

22

Esercizio: la caduta dei gravi *#2*

CaduteDeiGravi2.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  }
```

23

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|
+---+---+---+---+---+
```

24

Esercizio: la tavola pitagorica

Valute_v2.cpp

```
1  #include<iostream>
2  #include<iomanip>
3  using namespace std;
4
5  int main()
6  {
7          cout << "+---+---+---+---+---+" << endl;
8          cout <<right;
9          for(int i=1;i<=5;i++){
10                 cout<<"|";
11                 for(int j=1;j<=5;j++)
12                         cout << setw(3) << i*j << "|";
13                 cout <<endl<< "+---+---+---+---+---+" << endl;
14         }
15 }
```

25

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

Esempio: *l'importo 5609 è composto da 11 pezzi da 500, 1 pezzo da 100, 1 pezzo da 5, 2 pezzi da 2.*

26

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.*

27

Esercizio: *«Scusi, ha da cam*

Si scriva un programma C++ che chieda all'utente
in una variabile intera e restituisca il numero di ba
comporlo, scegliendo tra i seguenti tagli:

500, 200, 100, 50, 20,

Suggerimento: *E' opportuno che il programm
contenente i tagli e che confronti l'importo inse
dal maggiore al minore. Un ulteriore array conter
ciascun taglio.*

```
+-------+----+
|TAGLI  |   N|
+-------+----+
|500    |  11|
+-------+----+
|200    |   0|
+-------+----+
|100    |   1|
+-------+----+
|50     |   0|
+-------+----+
|20     |   0|
+-------+----+
|10     |   0|
+-------+----+
|5      |   1|
+-------+----+
|2      |   2|
+-------+----+
|1      |   0|
+-------+----+
```

28

Esercizio: *«Scusi, ha da cambiare?»*

```
1  #include<iostream>
2  #include<iomanip>
3  using namespace std;
4
5  int main()
6  {
7         int importo=0, residuo=0,i=0;
8         int tagli[9]={500,200,100,50,20,10,5,2,1};
9         int pezzi[9]={0};
10
11        cout <<"Inserisci l'importo (intero):"
12        cin >> importo;
13        residuo=importo;
```

Tutte le celle dell'array sono inizializzate a 0;

residuo è la variabile che a ogni passo contiene la somma ancora da cambiare. Inizialmente è pari a **importo**;

29

Esercizio: *«Scusi, ha da cambiare?»*

```
14        while (i<9)
15        {
16              if (residuo>=tagli[i]) {
17                    pezzi[i]=residuo/tagli[i];
18                    residuo=residuo%tagli[i];
19              }
20              i++;
21        }
...
```

Si prendono in considerazione tutti i tagli in ordine decrescente (da **tagli[0]** a **tagli[8]**)

Si, si poteva usare anche il **for**

30

Esercizio: *«Scusi, ha da cambiare?»*

```
14          while (i<9)
15          {
16                  if (residuo>=tagli[i]) {
17                        pezzi[i]=residuo/tagli[i];
18                        residuo=residuo%tagli[i];
19                  }
20                  i++;
21          }
 …
```

Se l'attuale ammontare del residuo è maggiore o uguale al taglio in esame (*i-esimo*) allora si passa a calcolare quante monete sono da erogare:

Numero di monete del taglio corrente (parte intera)

La parte eccedente (che è minore del taglio corrente) non può essere erogata, e quindi costituisce il nuovo residuo da ripartire nei passi seguenti, tra i tagli minori…

31

Esercizio: *«Scusi, ha da cambiare?»*

```
22          cout << "Scelta dei tagli: "<<endl;
23          cout << "+-------+----+"<< endl;
24          cout << "+TAGLI  +   N+"<< endl;
25          cout << "+-------+----+"<< endl;
26          for(int j=0;j<9;j++){
27                  cout << "|"<< setw(7) << left  << tagli[j];
28                  cout << "|"<< setw(4) << right << pezzi[j];
29                  cout << "|"<< endl;
30                  cout << "+-------+----+"<< endl;
31          }
32  }
```

32