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

2

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**

3

Libri di testo

Introduzione al linguaggio – costrutti e tecniche di base

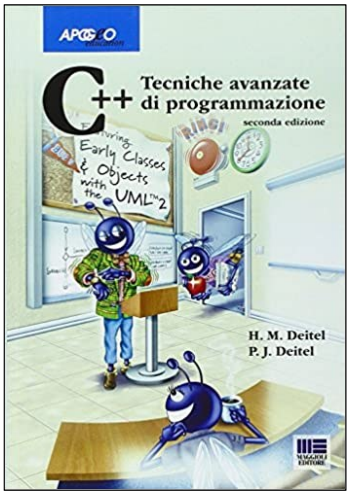
[FdP] H. M. Deitel, P. J. Deitel
C++ Fondamenti di programmazione
II ed. (2014) Maggioli Editore (Apogeo Education)
ISBN: 978-88-387-8571-9

4

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



5

Altro materiale didattico

Materiale di integrazione e approfondimento degli argomenti del corso

[MD] Slide e appunti del docente distribuiti durante il corso

[Misc] Articoli, link e riferimenti bibliografici su temi di approfondimento, curiosità e materiale «ludico».

6

Manualistica

Documentazione degli strumenti utilizzati, di utile consultazione

[GCC] Richard M. Stallman and the GCC Developer Community
Using the GNU Compiler Collection
GCC Version 13.2.0 (2023), https://gcc.gnu.org/onlinedocs/gcc-13.2.0/gcc/

[libc++] Paolo Carlini, et al.
The GNU C++ Library Manual
(2020), https://gcc.gnu.org/onlinedocs/libstdc++/manual/index.html

[C::B] Code::Blocks Team
Code::Blocks Users manual
Version 2.1.11 (2023), https://www.codeblocks.org

Altri riferimenti saranno forniti volta per volta…

7

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

8

Dal C al C++

9

Input/Output da *console*

10

Hello.cpp

```
1  // Programma che visualizza una riga di testo.
2  #include <iostream>

3  int main()
4  {
5       std::cout << "Salve, mondo!" << std::endl;
6       return 0;
7  }
```

```
Salve, mondo!
```

11

Hello.cpp

```
1  // Programma che visualizza una riga di testo.
2  #include <iostream>
```

Il C++ introduce il commento su linea singola, che inizia con // e termina con la fine della riga corrente

E' ancora possibile utilizzare i commenti «alla C» delimitati dai terminatori /* e */

12

Hello.cpp

```
1  // Programma che visualizza una riga di testo.
2  #include <iostream>
```

Il C++ utilizza lo stesso preprocessore del compilatore C, introducendo diverse innovazioni.

Il file header **`iostream`** definisce gli *stream di input/output* per la console (tastiera+schermo).

13

Input/Output da *console*

In C++, il flusso dei dati in transito tra le periferiche (o tra i *file*) e la memoria del calcolatore (*e.g.* le variabili) è gestito tramite delle entità denominate ***stream***;

I dati possono fluire in due direzioni:

- I dati che transitano attraverso lo stream dalla memoria al dispositivo/file di destinazione sono dati in ***output***;
- Viceversa, quelli che fluiscono nella direzione inversa sono dati in ***input***;

Uno stream può consentire il transito dei dati in una sola delle due direzioni o in entrambe.

14

Input/Output da *console*

Nel file header iostream sono definiti, tra gli altri, due strutture dati che implementano gli stream destinati all'I/O da console:

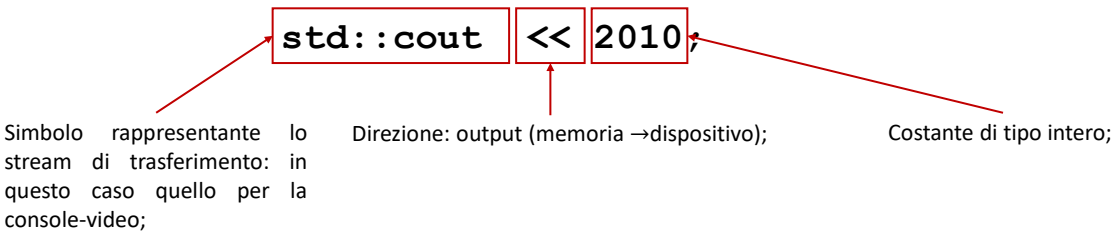
- **`cout`** per comunicare con il video, attivo solo in output
- **`cin`** per comunicare con la tastiera, attivo solo in input

per trasmettere dati attraverso questi due stream si utilizzano rispettivamente gli operatori di flusso **<<**(output) e **>>**(input);

15

Input/Output da *console*

Per visualizzare il valore di una costante sullo schermo:



16

Hello.cpp

```
5    std::cout << "Salve, mondo!" << std::endl;
```

Il prefisso **`std::`** davanti a **`cout`** è necessario quando si utilizzano identificatori «inseriti» nel programma mediante l'inclusione dell'header **`iostream`**.

La notazione **`std::out`** indica che l'istruzione si riferisce al nome cout definito nel *namespace* **`std`** definito in **`iostream`**.

Un namespace è un meccanismo di C++ per riferirsi in maniera esplicita ai nomi (*e.g.* variabili) definite in un dato ambito di visibilità (*scope*).

17

Input/Output da *console*

Per ricevere un valore immesso da tastiera (e riporlo in una variabile)

```
int anni;
std::cin >> anni ;
```

Simbolo rappresentante lo stream di trasferimento: in questo caso quello dalla console-tastiera;

Direzione: input (dispositivo →memoria);

Variabile di tipo intero;

18

InserisciNum.cpp

```
1  #include<iostream>

2  int main()
3  {
4        int num=0;
5        std::cout << "Inserisci un numero intero: ";
6        std::cin >> num;
7        std::cout << "Hai inserito "<<num<<std::endl;
8        return 0;
9  }
```

19

InserisciNum.cpp

Indicare ogni volta esplicitamente il namespace di un simbolo «importato» può essere noioso. È possibile rendere visibile un nome nel namespace corrente mediante la direttiva **`using`**

Lo utilizziamo per omettere il prefisso **`std::`**

```
1  #include<iostream>

2  int main()
3  {
4        int num=0;
5        std::cout << "Inserisci un numero intero: ";
6        std::cin >> num;
7        std::cout << "Hai inserito "<<num<<std::endl;
8        return 0;
9  }
```

20

InserisciNum.cpp

D'ora in poi, per i simboli **cout**, **cin** e **endl** L'indicazione del namespace **std::** sarà «sottintesa».

```
1  #include<iostream>
2  using std::cout;
3  using std::cin;
4  using std::endl;
5  int main()
6  {
7        int num=0;
8        cout << "Inserisci un numero intero: ";
9        cin >> num;
10       cout << "Hai inserito " << num << endl;
11       return 0;
12 }
```

21

InserisciNum.cpp

```
1  #include<iostream>
2  using std::cout;
3  using std::cin;
4  using std::endl;
5  int main()
6  {
7        int num=0;
8        cout << "Inserisci
9        cin >> num;
10       cout << "Hai inseri
11       return 0;
12 }
```

Promemoria: editing e compilazione

```
$ nano InserisciNum.cpp
```

…comporre il file salvare con **Ctrl+O**…

```
$ g++ InserisciNum.cpp
$ ./a.out
```

Il nome **a.out** è quello che il compilatore dà, per convenzione, al prodotto della compilazione. E' possibile cambiarlo utilizzando lo switch **–o** sulla linea di comando:

```
$ g++ InserisciNum.cpp –o InserisciNum
$ ./InserisciNum
Inserisci un numero intero:
…
```

22

Input/Output da *console*

Per visualizzare il contenuto di una variabile sullo schermo:

```
int anni=18;
cout  << anni ;
```

```
18
```

```
cout  << "Eta' " << anni << endl << «Compiuti.";
```

Più valori possono essere «accodati». Sono trasferiti in ordine da sinistra verso destra

```
Eta' 18
Compiuti.
```

23

Input/Output da *console*

Per visualizzare il valore di una costante sullo schermo:

```
cout  << 2010 ;
cout  << "Ciao";
cout  << "Ciao" << "!";
```

```
2010
Ciao
Ciao!
```

Più valori possono essere «accodati». Sono trasferiti in ordine da sinistra verso destra

```
cout  << "Ciao" << endl <<":-*";
```

```
Ciao
:-*
```

24

Input/Output da *console*

Per ricevere un valore immesso da tastiera (e riporlo in una variabile)

```
int anni;
cin >> anni;
```
Supponiamo che qui l'utente immetta il valore 21…

```
cout << "Eta' " << anni;
```

```
Eta' 21
```

25

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;

26

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

27

I/O formattato: manipolatori di flusso

28

Esempio: output standard di numeri reali

- Si scriva il programma C++ che calcoli il valore di π (in doppia precisione) sviluppando la seguente serie:

$$\frac{\pi}{4} = \sum_{i=0}^{k} \frac{(-1)^i}{2i+1}$$

  - Il programma chiede in input il numero intero k e termina visualizzando il valore calcolato.

29

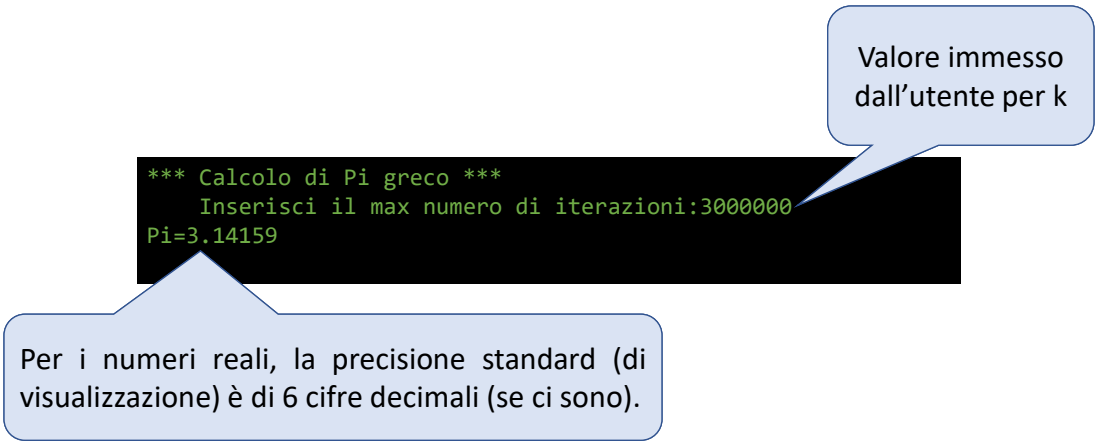
Esempio: output standard di numeri reali

PiGrecoQuarti_v1.cpp

```
 1  #include<iostream>
 2  using std::cin;
 3  using std::cout;
 4  using std::endl;
 5  int main()
 6  {
 7          int k;
 8          double pi=0.0,num=1;
 9          cout<<"*** Calcolo di Pi greco ***"<<endl;
10          cout<<"    Inserisci il max numero di iterazioni:";
11          cin >> k;
12          for (int i=0;i<=k;i++){
13                  pi+=num/(2*i+1);
14                  num*=-1;
15          }
16          pi=pi*4.0;
17          cout << "Pi="<<pi<<endl;
18  }
```

30

Esempio: output standard di numeri reali



31

I/O formattato: manipolatori di stream

Per i numeri reali, la precisione standard (di visualizzazione) è di 6 cifre decimali (se ci sono)

```
double cvar=4.14392439, dvar=1.8;
cout  << "cvar=" << cvar << endl;
cout  << "dvar=" << dvar << endl;
```

```
cvar=4.143924
dvar=1.8
```

Attenzione! il *troncamento* alla sesta cifra ha effetto solo sulla visualizzazione e non sul valore della variabile.

32

I/O formattato: manipolatori di stream

Per i numeri reali, la precisione standard (di visualizzazione) è di 6 cifre decimali (se ci sono)

È possibile modificare questa scelta modificando la «configurazione» dello stream di output utilizzando appositi *manipolatori di stream*

Conosciamo già il manipolatore **`endl`** che produce il «ritorno a capo»

Con i manipolatori **`fixed`** e **`setprecision`**, possiamo intervenire sulla visualizzazione dei numeri reali.

33

I/O formattato: manipolatori di stream

I manipolatori non parametrici (che non richiedono la specificazione di alcun valore/parametro) come **`endl`** e **`fixed`** sono definiti nell'header **`iostream`**

Per utilizzare i *manipolatori* parametrici (come setprecision) occorre includere anche l'header **`iomanip`**

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

34

Esempio: manipolatori di stream

Esempio di utilizzo dei *manipolatori* **`fixed`** e **`setprecision`**

```
#include<iostream>
using std::cout
using std::endl
using std::fixed
#include<iomanip>
using std::setprecision;
```

35

Esempio: manipolatori di stream

Esempio di utilizzo dei *manipolatori* **`fixed`** e **`setprecision`**

```
double cvar=4.14392439, dvar=1.8;
cout  << fixed << setprecision(4);
cout  << "cvar=" << cvar << endl;
cout  << "dvar=" << dvar << endl;
```

```
cvar=4.1439
dvar=1.8000
```

36

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
```

37

Esercizio: manipolatori di stream

PiGrecoQuarti_v2.cpp

```
#include<iostream>
using std::cin;
using std::cout;
using std::endl;
using std::fixed;
#include<iomanip>
using std::setprecision;

int main()
{
        int k,ncifre;
        double pi=0.0,num=1;
        cout<<"*** Calcolo di Pi greco ***"<<endl;
        cout<<"    Inserisci il max numero di iterazioni:";
        cin >> k;
        cout<<"    Inserisci il numero di cifre decimali:";
        cin>>ncifre;
…
```

38

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  }
```


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.

I/O formattato: manipolatori di stream

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

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

Per riportare tutto alle condizioni di default occorre inviare:

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

41

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,

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

$g = 9.81\ m/s^2$

42

Esempio: visualizzare dati in tabelle

Dati due array: il primo contenente un elenco di nomi di valute e il secondo contenente i rispettivi cambi contro Euro (in singola precisione), desideriamo visualizzare in una tabella su ciascuna riga il nome della divisa e il suo corrispondente importo in Euro.

43

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 }
```

44

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
```

45

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
```

Ci piacerebbe che:

1. le divise siano rappresentate in un campo di larghezza sufficiente a contenere il nome più lungo e uguale per tutte le righe;
2. Ciascun nome sia allineato al lato sinistro del campo.
3. Analogamente, gli importi di cambio contro l'Euro, siano rappresentati in un campo appositamente dimensionato e allineati a destra.

46

I/O formattato: campi e allineamenti

Per visualizzare i dati in output in una vera 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(`**n**`)`** riserva uno spazio (*campo*) di n caratteri per la visualizzazione del dato che segue
- **`left`** e **`right`** determinano il lato da cui «*si comincia a scrivere*» nel campo. Lo spazio non utilizzato è occupato da un carattere «*di riempimento*» (lo spazio, per default)

47

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`**

48

I/O formattato: campi e allineamenti

Per esempio nella seguente espressione:

```
X=12; Y=3;
cout << "X=" << setw(5) << right << X << endl;
```


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

49

Esempio: visualizzare una tabella *#2*

Valute_v2.cpp

Dichiariamo l'uso dei vari manipolatori *as usual*

```
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};
…  …
```

50

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.

51

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|
```

52

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|
```

53

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

56

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.*

57

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.*

58

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

59