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

3

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!
```

4

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 */

5

Hello.cpp

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

Il C++ utilizza un preprocessore del tutto analogo a quello del compilatore C, con leggerissime differenze.

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

6

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.

7

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:

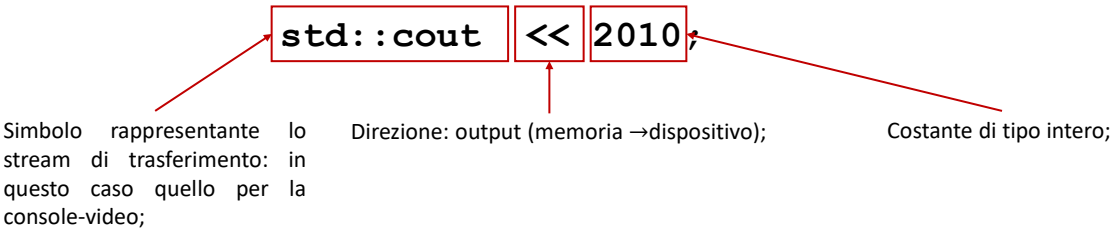
- **`cout`** per comunicare con il video, attivo solo in output
- **`cin`** per comunicare con la tastiera, attivo solo in input

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

8

Input/Output da *console*

Per visualizzare il valore di una costante sullo schermo:



9

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*).

10

Input/Output da *console*

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

11

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  }
```

12

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  }
```

13

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 }
```

14

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.
```

15

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
:-*
```

16

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
```

17

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 *(focaccout)*
- Visualizzi la storiella mostrata in seguito tenendo conto di alcune condizioni che si verificano in base al valore dell'input;

18

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

19

intermezzo

Impiego di funzioni matematiche

20

Esempio: impiego di funzioni matematiche

- Si scriva il programma C++ che: dati l'intero n e il numero reale k (in doppia precisione), restituisca il risultato della seguente formula:

$$ris = \sum_{i=1}^{n} \left(\frac{1}{k}\right)^{i}$$

- Per il calcolo della potenza utilizziamo la ***funzione di libreria*** `pow()`

21

Esempio: impiego di funzioni matematiche

PrimoUsoPow.cpp

```
1  #include<iostream>
2  #include<cmath>
3  using std::cin;
4  using std::cout;
5
6  int main()
7  {
…    …
```

L'inclusione di questo header, rende disponibili le funzioni della libreria matematica in dotazione al compilatore.

22

Esempio: impiego di funzioni matematiche

- La libreria matematica fornisce una raccolta di funzioni matematiche quali: elevamento a potenza, funzioni trigonometriche e un po' di costanti matematiche come π , *e*, etc.
- Elevamento a potenza: la funzione pow()

```
double pow(double base, double exp);
```

23

Esempio: impiego di funzioni matematiche

- La libreria matematica fornisce una raccolta di funzioni matematiche quali: elevamento a potenza, funzioni trigonometriche e un po' di costanti matematiche come π , *e*, etc.
- Elevamento a potenza: la funzione pow()

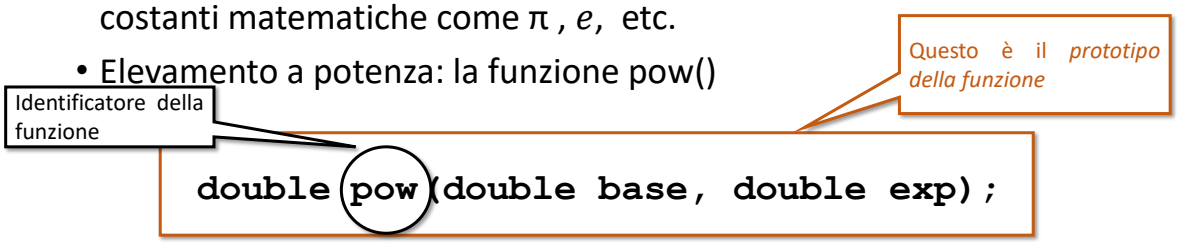
Questo è il *prototipo della funzione*

```
double pow(double base, double exp);
```

24

Esempio: impiego di funzioni matematiche

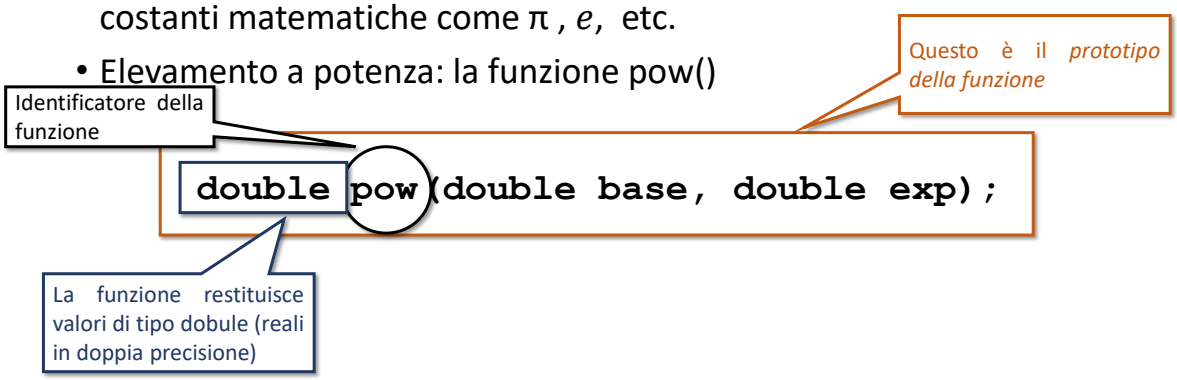
- La libreria matematica fornisce una raccolta di funzioni matematiche quali: elevamento a potenza, funzioni trigonometriche e un po' di costanti matematiche come π , *e*, etc.
- Elevamento a potenza: la funzione pow()



25

Esempio: impiego di funzioni matematiche

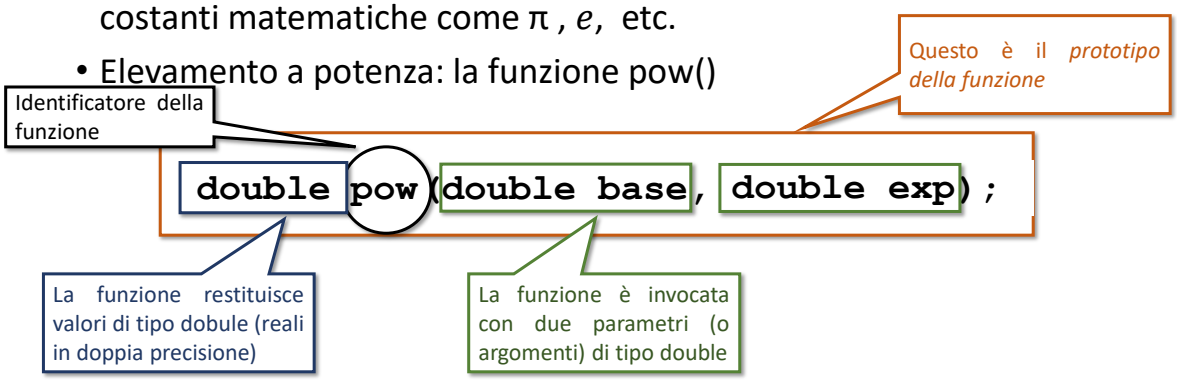
- La libreria matematica fornisce una raccolta di funzioni matematiche quali: elevamento a potenza, funzioni trigonometriche e un po' di costanti matematiche come π , *e*, etc.
- Elevamento a potenza: la funzione pow()



26

Esempio: impiego di funzioni matematiche

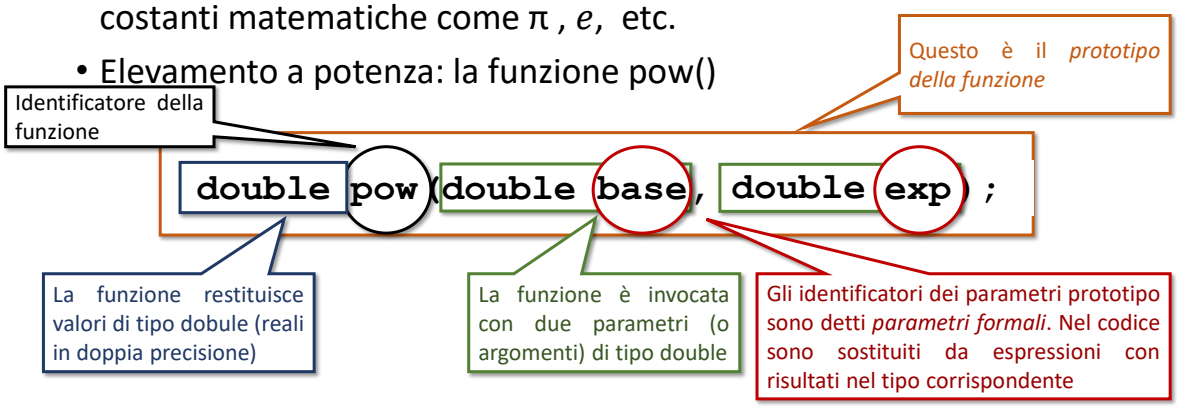
- La libreria matematica fornisce una raccolta di funzioni matematiche quali: elevamento a potenza, funzioni trigonometriche e un po' di costanti matematiche come π , *e*, etc.
- Elevamento a potenza: la funzione pow()



27

Esempio: impiego di funzioni matematiche

- La libreria matematica fornisce una raccolta di funzioni matematiche quali: elevamento a potenza, funzioni trigonometriche e un po' di costanti matematiche come π , *e*, etc.
- Elevamento a potenza: la funzione pow()



28

Esempio: impiego di funzioni matematiche

PrimoUsoPow.cpp

```
 1  #include <iostream>
 2  #include <cmath>
 3  using std::cin;
 4  using std::cout;
 5  using std::endl;
 6  int main()
 7  {
 8          int n;
 9          double k,ris=0;
10
11          cout << "inserisci n:";
12          cin >> n;
13          cout << "inserisci k:";
14          cin >> k;
15          for (int i=1;i<=n;i++){
16                  ris=ris+pow((1/k),i);
17
18          cout << "ris="<< ris <<endl;
19  }
```

29

Esempio: impiego di funzioni matematiche

PrimoUsoPow.cpp

```
 1  #include <iostream>
 2  #include <cmath>
 3  using std::cin;
 4  using std::cout;
 5  using std::endl;
 6  int main()
 7  {
 8          int n;
 9          double k,ris=0;
10
11          cout << "inserisci n:";
12          cin >> n;
13          cout << "inserisci k:";
14          cin >> k;
15          for (int i=1;i<=n;i++){
16                  ris=ris+pow((1/k),i);
…   …
```

Il for del C++ consente di dichiarare la variabile di controllo direttamente nell'intestazione del ciclo.

La variabile così dichiarata può essere usata soltanto nei «confini» del suo ciclo for

30

Alcune funzioni matematiche…

Funzione	Risultato
`double pow (double a, double b);`	a^b
`double sqrt (double x);`	$\sqrt{x}\ (x \geq 0)$
`double sin (double x);`	$\sin x$
`double cos (double x);`	$\cos x$
`double tan (double x);`	$\tan x$
`double exp (double x);`	e^x
`double log (double x);`	$\ln x\ (x \geq 0)$
`double log10 (double x);`	$\log_{10} x\ (x \geq 0)$
`double asin (double x);`	$\sin^{-1} x\ \ (x \in [-\pi/2, \pi/2])$
… `acos(), atan(), sinh(), cosh(), tanh(), asinh()`….	

31

Alcune costanti…

Costante	Tipo	Valore
`M_PI`		π
`M_E`		e
`M_SQRT2`		$\sqrt{2}$
`M_LN2`		$\ln 2$
`M_LN10`		$\ln 10$
`M_LOG2E`		$\log_2 e$

32

I/O formattato: manipolatori di flusso

33

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.

34

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  }
```

35

Esempio: output standard di numeri reali

Valore immesso dall'utente per k

```
*** Calcolo di Pi greco ***
    Inserisci il max numero di iterazioni:3000000
Pi=3.14159
```

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

36

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.

37

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.

38

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`**

39

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;
```

40

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
```

41

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
```

42

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;
 …  …
```


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  }
```

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,

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

$g = 9.81\ m/s^2$

45