Laboratorio di programmazione

Lezione II

Tatiana Zolo

zolo@cs.unipr.it

IL PROGRAMMA C++

Istruzioni (espressioni terminate da “;”)

- istruzioni di dichiarazione (es. `int x = 0;`);
- istruzioni di assegnamento (es. `m = (x + y + z) / 3.0;`);
- istruzioni di output (es. `cout << "La media e' " << m;`).

IL PROGRAMMA C++

Istruzioni (espressioni terminate da ";")

- istruzioni di dichiarazione (es. `int x = 0;`);
- istruzioni di assegnamento (es. `m = (x + y + z) / 3.0;`);
- istruzioni di output (es. `cout << "La media e' " << m;`).

Funzioni: ogni programma per essere eseguito deve contenere una funzione main() e l'esecuzione comincia con la prima istruzione di main(). Ogni funzione è composta da 4 parti:

- tipo di ritorno;
- nome della funzione;
- lista dei parametri;
- corpo della funzione.

Le prime tre parti insieme rappresentano il prototipo della funzione.

IL PROGRAMMA C++

Istruzioni (espressioni terminate da ";")

- istruzioni di dichiarazione (es. `int x = 0;`);
- istruzioni di assegnamento (es. `m = (x + y + z) / 3.0;`);
- istruzioni di output (es. `cout << "La media e' " << m;`).

Funzioni: ogni programma per essere eseguito deve contenere una funzione main() e l'esecuzione comincia con la prima istruzione di main(). Ogni funzione è composta da 4 parti:

- tipo di ritorno;
- nome della funzione;
- lista dei parametri;
- corpo della funzione.

Le prime tre parti insieme rappresentano il prototipo della funzione.

- Istruzione `return` e `return 0;`.

IL PROGRAMMA C++

Istruzioni (espressioni terminate da ";")

- istruzioni di dichiarazione (es. `int x = 0;`);
- istruzioni di assegnamento (es. `m = (x + y + z) / 3.0;`);
- istruzioni di output (es. `cout << "La media e' " << m;`).

Funzioni: ogni programma per essere eseguito deve contenere una funzione main() e l'esecuzione comincia con la prima istruzione di main(). Ogni funzione è composta da 4 parti:

- tipo di ritorno;
- nome della funzione;
- lista dei parametri;
- corpo della funzione.

Le prime tre parti insieme rappresentano il prototipo della funzione.

- Istruzione `return` e `return 0;`.
- es. `media.cpp`; `prova.cpp` (notazione scientifica).

TIPI DI DATI FONDAMENTALI

- char: singoli caratteri e piccoli interi (modificatori signed e *unsigned*);
- int: valori interi di differenti dimensioni (signed, *unsigned*, *lon*g e *short*);
- float, double: valori in virgola mobile (*long* ai double);
- bool: vero o falso.

La dimensione del dato contenibile nella variabile dipende dallo spazio riservato in memoria per ogni tipo di dichiarazione (es. `tipi.cpp`, operatore `sizeof()`).

OPERATORI DI UGUAGLIANZA, RELAZIONALI E LOGICI

Il risultato di questi operatori è di tipo bool (es. `boolean.cpp`):

OPERATORE	FUNZIONE	USO
!	NOT logico	`!E`
<	minore di	`E1 < E2`
<=	minore o uguale a	`E1 <= E2`
>	maggiore di	`E1 > E2`
>=	maggiore o uguale a	`E1 >= E2`
==	uguaglianza	`E1 == E2`
!=	disuguaglianza	`E1 != E2`
&&	AND logico	`E1 && E2`
||	OR logico	`E1 || E2`

IF E WHILE

- if

```
if (condizione)
  istruzione;
```

L'istruzione viene eseguita se la condizione è vera (es. `voto_giudizio1.cpp`).

- while

```
while (condizione)
  istruzione;
```

L'istruzione viene eseguita finché la condizione è vera (es. `mcm1.cpp`).

ESERCIZI

- if-else: problema di conversione Euro –> Lire e Lire –> Euro (chiedere all'utente quale conversione vuole fare). `Euro_Lire2.cpp`.

ESERCIZI

- if-else: problema di conversione Euro –> Lire e Lire –> Euro (chiedere all'utente quale conversione vuole fare). `Euro_Lire2.cpp`.
- while: stampa tutti i caratteri dal codice ASCII 32 al codice 126. `char.cpp`.

ESERCIZI

- if-else: problema di conversione Euro –> Lire e Lire –> Euro (chiedere all'utente quale conversione vuole fare). `Euro_Lire2.cpp`.
- while: stampa tutti i caratteri dal codice ASCII 32 al codice 126. `char.cpp`.
- if-else: conversione di un carattere da minuscolo a maiuscolo o viceversa. `minusc_maiusc.cpp`.

ESERCIZI

- if-else: problema di conversione Euro –> Lire e Lire –> Euro (chiedere all'utente quale conversione vuole fare). `Euro_Lire2.cpp`.
- while: stampa tutti i caratteri dal codice ASCII 32 al codice 126. `char.cpp`.
- if-else: conversione di un carattere da minuscolo a maiuscolo o viceversa. `minusc_maiusc.cpp`.
- while: leggere da standard input una sequenza di numeri interi terminata da un intero negativo e calcolarne la media (escludendo il numero negativo). Stampare quindi il risultato su standard output. `media_n.cpp`.

ESERCIZI

- if-else: problema di conversione Euro –> Lire e Lire –> Euro (chiedere all'utente quale conversione vuole fare). `Euro_Lire2.cpp`.
- while: stampa tutti i caratteri dal codice ASCII 32 al codice 126. `char.cpp`.
- if-else: conversione di un carattere da minuscolo a maiuscolo o viceversa. `minusc_maiusc.cpp`.
- while: leggere da standard input una sequenza di numeri interi terminata da un intero negativo e calcolarne la media (escludendo il numero negativo). Stampare quindi il risultato su standard output. `media_n.cpp`.
- while: trasformare il programma del voto e del giudizio in modo che si ripeta fino a che l'utente non inserisce un numero negativo.