
Laboratorio di programmazione

Lezione III

Tatiana Zolo

`tatiana.zolo@libero.it`

IF-ELSE – OPERATORI

→ Espressione condizionale: $E1 ? E2 : E3$

$x = E1 ? E2 : E3;$

equivale a

```
if (E1)
    x = E2;
else
    x = E3;
```

→ es. `abs.cpp`.

IF-ELSE – OPERATORI

→ Espressione condizionale: $E1 ? E2 : E3$

$x = E1 ? E2 : E3;$

equivale a

```
if (E1)
    x = E2;
else
    x = E3;
```

→ es. `abs.cpp`.

→ Operatori di assegnamento composto

$a = a + b; \iff a += b;$

$a = a - b; \iff a -= b;$

$a = a * b; \iff a *= b;$

$a = a / b; \iff a /= b;$

IF-ELSE – OPERATORI

→ Espressione condizionale: $E1 ? E2 : E3$

$x = E1 ? E2 : E3;$

equivale a

```
if (E1)
    x = E2;
else
    x = E3;
```

→ es. `abs.cpp`.

→ Operatori di assegnamento composto

$a = a + b; \iff a += b;$

$a = a - b; \iff a -= b;$

$a = a * b; \iff a *= b;$

$a = a / b; \iff a /= b;$

→ es. `media_n.cpp`.

LIBRERIE

`#include <math.h>` $\Rightarrow$ C

`#include <cmath>` $\Rightarrow$ C++

Funzioni standard (argomento di tipo `double`):

`sqrt()`, `pow()`, `abs()`, `ceil()`, `floor()`,
`sin()`, `cos()`, `tan()`, `log()`, `sinh()`, `cosh()`, ...

es. `cubo.cpp`.

WHILE E DO-WHILE

- **while**: valuta la condizione ed esegue l'istruzione se la condizione è vera. Se la prima valutazione della condizione produce `false` l'istruzione non è mai eseguita.

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

- es. `voto_giudizio2.cpp`, `voto_giudizio3.cpp`.

WHILE E DO-WHILE

- **while**: valuta la condizione ed esegue l'istruzione se la condizione è vera. Se la prima valutazione della condizione produce `false` l'istruzione non è mai eseguita.

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

→ es. `voto_giudizio2.cpp`, `voto_giudizio3.cpp`.

- **do-while**: l'istruzione è eseguita prima che la condizione sia valutata (l'istruzione è eseguita almeno una volta).

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

→ es. `mcm2.cpp`.

IL CICLO FOR

```
for (inizializzazione; condizione; espressione)  
    istruzione;
```

- **inizializzazione (della variabile di controllo)**: istruzione di dichiarazione oppure un'espressione; può essere omessa.
- **condizione**: agisce come controllo del ciclo (il ciclo 'for' potrebbe non essere mai eseguito).
- **espressione**: viene valutata dopo ogni iterazione del ciclo, in genere usata per modificare le variabili inizializzate nell'inizializzazione e facenti parte della condizione.

Buona norma: non modificare la variabile di controllo dentro il corpo del ciclo.

es. `esempi_for.cpp`.

PRE-INCREMENTO E POST-INCREMENTO

→ `int a;
int i = 0;
a = ++i;`

`++i`, `++` precede la variabile: incrementa il valore della variabile *prima* della valutazione dell'espressione.

In memoria: $a \leftarrow 1, i \leftarrow 1$.

→ `int a;
int i = 0;
a = i++;`

`i++`, `++` segue la variabile: valuta l'intera espressione, *poi* incrementa il valore della variabile.

In memoria: $a \leftarrow 0, i \leftarrow 1$.

(Analogamente per `--i` e `i--`)

FOR E WHILE

→ `for (inizializzazione; condizione; espressione)
 istruzione;`

equivale a

```
inizializzazione;  
while (condizione) {  
    istruzione;  
    espressione;  
}
```

FOR E WHILE

→ `for (inizializzazione; condizione; espressione)
 istruzione;`

equivale a

```
inizializzazione;  
while (condizione) {  
    istruzione;  
    espressione;  
}
```

→ Istruzione **break**: interrompe il ciclo “while”, “do-while”, “for” o “switch” che la racchiude. Se il *break* è eseguito all’interno di un’istruzione di ciclo (o di switch) annidata, il ciclo (o switch) più esterno non è influenzato dalla terminazione del ciclo o switch più interno.

`es.mcm3.cpp, tavola_pitagorica.cpp.`

GET()

get(): estrae un singolo carattere dallo stream di input, *compreso un carattere bianco (e "l'andata a capo")*, e lo memorizza in un char.

Sintassi:

```
char c = cin.get();
```

oppure

```
char c;  
cin.get(c);
```

(vi è sotto il concetto di classe e metodo di una classe, programmazione object oriented).

es. `leggi_get.cpp`.

ESERCIZI

1. **funzioni standard**: calcola e stampa le radici di un'equazione di grado 2 quando il discriminante è positivo o nullo. I coefficienti dell'equazione sono inseriti da tastiera. `eqgrado2.cpp`.
2. **cicli**: calcola il fattoriale di un numero intero n , con n dato di input. Stampa quindi il risultato su standard output. `fattoriale.cpp`.
3. **cicli**: calcola se il numero intero maggiore di 1 in ingresso è primo oppure no. Nel secondo caso, stampa il più piccolo tra i suoi divisori. `primo.cpp`.
4. **for**: scrivere un programma che stampi un rettangolo la cui cornice sia costituita da caratteri asterisco e la parte interna da un carattere immesso dall'utente. Anche il numero di righe e di colonne viene deciso dall'utente. `rettangolo.cpp`.

ESERCIZI

5. **cicli e get()**: leggere da standard input una sequenza di caratteri (di lunghezza massima 32) terminata da spazio o da 'a capo' e determinare il numero di lettere 'a' presenti. Stampare quindi il risultato su standard output. `conta_a.cpp`.
6. **cicli**: leggere da standard input una sequenza di caratteri terminata da 'a capo' e determinare il numero di vocali minuscole presenti. Stampare quindi il risultato su standard output. `conta_vocali.cpp`.
7. **cicli**: calcola il massimo comune divisore tra due interi. `mcd.cpp`.