Introduzione alla Programmazione

claudio.lucchese@unive.it

Espressioni

Le espressioni sono combinazioni valide di costanti, variabili, operatori e chiamate di funzione.

Costanti numeriche: 10 (int) 10**u** (unsigned) 10**l** (long)
10.0 (double) 10.0**f** (float) 10.0**d** (double)

Operatori aritmetici: **() + - * / %**

Operatori relazionali e di uguaglianza: **< <= > >= == !=**

Operatori logici: **&& || !**

Per ogni operatore è definita la precedenza e l'associatività.

Es. di espressione: (-2 + (2*3-4*3+2)/7) / (2*1)

Operatori, Promozioni e Conversioni Implicite

Ciascun operatore ha un tipo di input e un tipo di output (dominio e codominio di una funzione):

- Le operazioni +,-,*,/ sono definite tra coppie di **unsigned, int, unsigned long, long, float e double e restituiscono un valore dello stesso tipo**.
- Nota: non sono definite tra tipi short! Questi vengono sempre trasformati in interi (***Promozione***)

Se i tipi in input non sono corretti/coerenti:

- https://en.cppreference.com/w/c/language/conversion
- se possibile i valori vengono convertiti al tipo più espressivo,
  altrimenti avremo errore a tempo di compilazione

Esempio:

- 10 % 3 *ha valore 1*
- 10.0 % 3.0 error: invalid operands to binary % (have 'double' and 'double')
- 10 / 3 *ha valore 3* ---> divisione intera
- 10 / 3.0 *ha valore 3.333* ---> 10 viene prima convertito in double

Casting: Conversioni Esplicite

Si può cambiare il tipo di un’espressione con l’operazione di casting. Es:

- (float) 10 — ha valore 10 float
- ((int) 10.0) % ((int) 3.0) — ha valore 1 intero
- (int) 10.0 — il 10 double viene trasformato in un 10 intero
- (int) 5.1 — ha valore 5 intero (troncamento)
- (int) 5.9 — ha valore 5 intero (troncamento)
- (float)((int) 5.9) — ha valore 5.0 float

Overflow

Le operazioni tra interi *unsigned* sono calcolate modulo 2^b *(resto della divisione)*

(b è il numero di bit)

```c
#include <stdio.h>

int main() {
  unsigned int i = 0;
  unsigned int j = i - 1u;

  unsigned char x = 255 + 1;

  unsigned char h = 255;
  int k = h + 1;

  return 0;
}
```

- Quanto vale j ?
- Quanto vale x ?
- Quanto vale k ?

Overflow

```
1  #include <stdio.h>
2
3  int main() {
4      unsigned int i = 0;
5      unsigned int j = i - 1u;
6
7      unsigned char x = 255 + 1;
8
9      unsigned char h = 255;
10     int k = h + 1;
11
12     return 0;
13 }
```

Le operazioni tra interi *unsigned* sono calcolate modulo 2^b *(resto della divisione)*

(b è il numero di bit)

Nel caso di interi *signed*, dipende dal compilatore.

- Quanto vale j ?
  - **4294967295** (-1 % 2^{32} = 2^{32} -1 = max uint)
- Quanto vale x ?
  - **0**, perché:
  - per realizzare l'assegnamento 256 viene trasformato in unsigned char: 256 % 2^8 = 0
- Quanto vale k ?
  - **256**
  - la somma viene eseguita tra interi e il risultato della somma viene coerentemente assegnato a un intero

Esercizi

1. Trovare il massimo tra 3 numeri.
2. Verificare se l'espressione (0.1+0.1+0.1)==0.3 è vera.
3. Date le lunghezze di tre segmenti, dire se questi possono formare un triangolo.
4. Date le lunghezze di tre segmenti, dire se questi possono formare un triangolo rettangolo.
5. Dato in input un numero positivo intero <=255, visualizzare la sua rappresentazione binaria
6. Data una variabile x, quanto valgono le seguenti espressioni: "x++", "++x", "x++ + ++x"?
7. Dati i coefficienti di un'equazione di secondo grado $ax^2+bx+c=0$, visualizzarne le soluzioni