Esercitazione Programmazione Procedurale con Laboratorio

Tipi e Conversioni di Tipo

Esercizio 1 Tipi delle espressioni

Dare il valore del risultato e il tipo delle seguente espressioni, specificare anche il tipo degli operandi all'interno dell'espressione. Utilizzare anche le slide sui letterali (*literals*):

- $2 + 3$
- $2.2 + 3.3$
- $2 + 3.3$
- $1/2$
- $1.0/2.0$
- $'A'/5$ (il valore ASCII di $'A'$ è 65)
- $1/2.0$
- $1ul + 3l$
- $1u + 3l$ (supponiamo che sia un *long* che un *unsigned int* occupino 4 byte)
- $5ul + 6ll$
- $3f * 5.6L$
- $-1u + 1.2$ (attenzione al primo operando che viene convertito a *unsigned int*; supponiamo che un *unsigned int* occupi 4 byte)
- $1.2 + 1.4l$
- $1ll * 2.0f$

Soluzione dell'esercizio 1

- 5, *int*, gli operandi sono entrambi di tipo *int*
- 5.5, *double*, gli operandi sono entrambi di tipo *double*
- 5.3, *double*, il primo operando ha tipo *int*, il secondo ha tipo *double*
- 0, *int*, gli operandi sono entrambi di tipo *int*
- 0.5, *double*, gli operandi sono entrambi di tipo *double*
- 13, *int*, gli operandi sono entrambi di tipo *int* (i letterali carattere hanno tipo *int*)
- 0.5, *double*, il primo operando ha tipo*int*, il secondo ha tipo *double*
- 4, *unsigned long*, il primo operando ha tipo *unsigned long*, il secondo ha tipo *long*

- 4, *unsigned long int*, il primo operando ha tipo *unsigned int*, il secondo ha tipo *long*. Applicare seconda regola di conversione (slide 12 conversione di tipo): un *long* non è in grado di rappresentare tutti i valori di *unsigned int*, quindi vengono tutti e due convertiti a *unsigned long*.
- 11, *long long*, il primo operando ha tipo *unsigned long*, il secondo ha tipo *long long*.
- 16.8, *long double*, il primo operando ha tipo *float*, il secondo ha tipo *long double*.
- 4294967296.2, *double*, il primo operando ha tipo *int* ma dato che è seguito da *u* viene convertito ad *unsigned int*: seguire la slide 20 (conversioni), nella quale si dice di sommare al numero in questione (cioè -1) $UINT\_MAX + 1$, quindi il valore sarà $-1 + UINT\_MAX + 1 = UINT\_MAX = 4294967295$. Il secondo operando ha tipo *double*.
- 2.6, *long double*, il primo operando ha tipo *double*, il secondo ha tipo *long double*.
- 2.0, *float*, il primo operando ha tipo *long long int*, il secondo ha tipo *float*.

Esercizio 2 Uso di conversione esplicita (cast)

Modificare il seguente programma in modo che il risultato stampato sia 3.4 invece di 3.

```c
#include <stdio.h>

int main() {

  int sum = 17, count = 5;
  double mean;

  mean = sum / count;
  printf("Value of mean : %f\n", mean );

  return 0;
}
```

Soluzione dell'esercizio 2

```c
#include <stdio.h>

int main() {
  int sum = 17, count = 5;
  double mean;

  mean = (double) sum / count;
  printf("Value of mean : %f\n", mean );

  return 0;
}
```

Anche *(double) sum / (double) count)* è corretto, anche se basta convertire uno solo dei due operandi a *double*, in modo che l'altro venga implicitamente convertito a *double* anch'esso. Notere invece che *(double) (sum / count)* è errato, dato che si converte a *double* il risultato di *sum / count*: in questo caso si esegue la divisione tra due *int* e poi si converte esplicitamente il risultato a *double*, ottenendo 3.0

Esercizio 3 Ancora conversioni

Rispondere alle domande sotto forma di commenti. Il comando *if (cond)* esegue il comando seguente se la condizione *cond* è vera. Supporre che il tipo *int* occupi 4 byte. Provare a scrivere ed eseguire nel proprio ambiente di programmazione.

```c
#include <stdio.h>
#include <limits.h>
#include <float.h>

int main() {
  unsigned int pippo = UINT_MAX;     // Che valore ha pippo?
  short pluto = (short) ULLONG_MAX;        // Che valore ha pluto?
  unsigned int paperino = INT_MIN;  // Che valore ha paperino?
  short paperoga = (short) FLT_MAX;  // Che valore ha paperoga?
  unsigned short gastone = 5;
  gastone = gastone - 2;  //Che regola e' stata applicata qui? Conversioni?
```

```
12
13    unsigned int paperone = gastone;
14    int amelia = -2;
15    long long archimede= 20LL;
16    long macchianera= 56L;
17
18    if (gastone > amelia)
19      printf("Hello1\n");  // Viene stampato? Perche'?
20
21    if (paperone > amelia)
22      printf("Hello2\n"); // Viene stampato? Perche'?
23
24    archimede = archimede + paperone;      //Qual e' il tipo dell'espressione a destra dell'uguale?
25    macchianera = macchianera + paperone;      //Qual e' il tipo dell'espressione a destra dell'uguale?
26
27    printf("Value of pippo %u\n", pippo);
28    printf("Value of pluto %d\n", pluto);
29    printf("Value of paperino %u\n", paperino);
30    printf("Value of paperoga %d\n", paperoga);
31    printf("Value of gastone %u\n", gastone);
32    printf("Valore di archimede %lld\n", archimede);
33    printf("Valore di macchianera %ld\n", macchianera);
34  }
```

Soluzione dell'esercizio 3

- *unsigned int pippo = UINT_MAX*; Il valore di *pippo* è il massimo valore esprimibile per il suo tipo, cioè *UINT_MAX*, $4,294,967,295$ se *unsigned int* è rappresentato con 4 byte.

- *short pluto* = (*short*) *ULLONG_MAX*;. Un *unsigned long long* è rappresentato con 8 byte ed il suo valore massimo è *ULLONG_MAX* che ha valore $8,446,744,073,709,551,615$. Dato che il suo valore è assegnato ad una variabile di tipo *short int* (che occupa 2 byte in memoria), la slide da utilizzare è "Conversions to signed integer types" (slide 22, conversioni di tipo), che ci dice che, in caso di overflow (come qui), i bit più significativi vengono eliminati (da 8 a 2 byte, vengono eliminati 48 bit), ed i rimanenti bit vengono vengono interpretati nel nuovo tipo (qui appunto *short int*). Il valore di 16 bit tutti settati ad 1, in complemento a 2, è quindi -1: *pluto* vale -1.

- *unsigned int paperino* = *INT_MIN*;. Se un *int* è rappresentato con 4 byte, *INT_MIN* è il valore $-2,147,483,648$. Quando assegno ad un tipo integer senza segno (*unsigned*), devo utilizzare la slide "Conversions to usigned integer types" (slide 20, conversioni di tipo). In questo caso sommo a quel valore *Utype_MAX* + 1 (in questo caso *UINT_MAX*+1) tante volte quanto è necessario per essere rappresentabile nel nuovo tipo *unsigned int* (*UINT_MAX* è $4,294,967,295$). Quindi, $-2,147,483,648 + 4,294,967,295 + 1 = 2,147,483,648$.

- *short paperoga* = (*short*)*FLT_MAX*;. La slide "Floats and integer" (slide 21, conversioni di tipo) ci dice che, una volta scartata la parte decimale, se quello che rimane è fuori il range dei valori rappresentabili, il risultato non è definito. *FLT_MAX* vale +3.4E+38. Provate a controllare quanto vale *paperoga* nel vostro ambiente di programmazione.

- *gastone* = *gastone* − 2;. La regola applicata è di "Integer promotion" (slide 8, conversioni di tipo), dato che *unsigned short* ha un rank di conversione più basso di *int*. La costante letterale *2* ha invece tipo *int*. Dato che il risultato di tipo *int* viene assegnato alla variabile *gastone* (di tipo *unsigned short*), viene eseguita una conversione implicita da *int* a *unsigned short*, che però non pone problemi dato che il valore risultato 3 è rappresentabile nell'intervallo del nuovo tipo *unsigned short* (vedi ancora slide "Conversions to usigned integer types").

- Per valutare l'espressione a riga 19, prima si applica la regola di Integer Promotion (slide 8) come richiesto dalla seconda regola in slide 12. Dato che *gastone* ha tipo *unsigned short*: viene promosso a *int*, dato che il suo valore 3 è rappresentabile con un *int*. A questo punto abbiamo un operando di tipo *int* e uno di tipo *int* (il valore di *amelia*): quindi abbiamo che $3 > -2$ e *Hello1* viene stampato.

- Per valutare l'espressione a riga 22, non si applica la regola di Integer Promotion, dato che *paperone* ha già grado di conversione uguale a *int* (ha tipo *unsigned int*). *amelia* ha tipo *int*. Quindi, in accordo alla prima parte della seconda regola in slide "What does it happen?" (slide 12, conversioni di tipo), *int* viene convertito a *unsigned int*, il tipo di *paperone*. -2 viene quindi convertito a *unsigned int*, diventando così il valore $4,294,967,294$. Quindi 3 (*paperone*) non è maggiore di tale valore. *Hello2* non viene stampato.

- *archimede = archimede + paperone*;. Non c'è bisogno di fare Integer Promotion, dato che sia *archimede* sia *paperone* hanno grado di conversione uguale o maggiore o di *int*. Per la seconda parte della seconda regola in slide 12, *paperone* (tipo *unsigned int*) è convertito al tipo di *archimede* (*long long*). Infatti, il tipo *long long* è in grado di rappresentare tutti i valori del tipo *unsigned int*, dato che il suo range va da $-9,223,372,036,854,775,808$ a $9,223,372,036,854,775,807$, mentre *unsigned int* da 0 a $4,294,967,295$.

- *macchianera = macchianera+paperone*;. Come al punto precedente (stessa motivazione), anche qui non c'è bisogno di applicare la regola di Integer Promotion. Seguire sempre la seconda parte della seconda regola in slide 12. In questo caso però, il tipo di *macchianera* (*long*) non è in grado di rappresentare tutti i valori del tipo di *paperone* (*unsigned int*). Infatti un tipo *long* rappresenta da $-2,147,483,648$ a $2,147,483,647$, mentre *unsigned int* da 0 a $4,294,967,295$ (*long* non è in grado di rappresentare i valori tra $2,147,483,647$ e $4,294,967,295$). Per questo motivo, gli operandi vengono tutti e due convertiti a *unsigned long*.