Conversioni fra Tipi di Dati

Luca Abeni

Tipi di Dato e Conversioni

- In C, ogni variabile è caratterizzata da un tipo
  - I valori che la variabile può assumere dipendono dal tipo
- È possibile mescolare variabili di tipo diverso nelle espressioni aritmetiche

```
double res;
int a = 3;
float b = 2.1;

res = a * b;
```

- In altri linguaggi, questo sarebbe un errore...
- Come viene calcolato il risultato???
- Il programma deve eseguire conversioni implicite fra vari tipi...

Conversioni Implicite

- `res = a * b;`
  - Se `a` è un intero e `b` un numero a virgola mobile...
  - Prima di tutto converte il valore di `a` da complemento a 2 a virgola mobile...
  - ...Poi fa il prodotto
  - Se `res` è a precisione doppia, il risultato va convertito prima di fare l'assegnamento
- In alcuni casi, risultati "strani":
  - `res = 1.0 + 5/2;`
  - 5 e 2 sono numeri interi...
  - ...Quindi "/" è una divisione intera: $5/2 = 2$ (con resto 1)
  - 2 è convertitio in virgola mobile e sommato a `1.0`

Conversioni Implicite - Quando

- Conversioni implicite avvengono:
  - In caso di assegnamento (conversione nel tipo della variabile a sinistra di "=")
  - Nelle espressioni
  - Nell'invocazione di funzioni (conversione nei tipi dei parametri formali)
- I casi di assegnamento e invocazione di funzione sono semplici
  - Il tipo della variabile oggetto dell'assegnamento o dei parametri formali specificano chiaramente verso che tipo convertire
- Le conversioni nelle espressioni possono essere meno chiare...

Conversioni Implicite - Assegnamento

```
double d;
int n;

d = 5;
n = 2.5
```

- Per assegnare 5 (valore intero) alla variabile d, il valore intero viene convertito in virgola mobile
  - Qui, non si ha perdita di informazione ($5 \rightarrow 5.0$)
- Per assegnare 2.5 (valore in virgola mobile) alla variabile n, il valore intero viene convertito in intero
  - Perdita di informazione! ($2.5 \rightarrow 2$)
  - Il valore viene troncato

Conversioni Implicite - Invocazione di Funzioni

```
int f(double a, double b);

...

res = f(1, 2);
```

- Per invocare `f()`, 1 e 2 devono essere convertiti da interi a `double`
- In generale: quando una funzione viene invocata il tipo dei parametri attuali viene convertito nel tipo dei parametri formali
- Come fa il compilatore a conoscere il tipo dei parametri formali?
  - Il prototipo della funzione deve essere stato dichiarato!
  - Una funzione deve essere dichiarata prima di poter essere invocata!

Conversioni Implicite - Valutazione di Espressioni

- Un'espressione viene valutata seguendo la priorità degli operatori
  - Prima moltiplicazioni e divisioni, poi somme...
  - A parità di priorità, si valuta da sinistra a destra
  - Parentesi per cambiare le precedenze
- Se un'operazione ha operandi dello stesso tipo, no problem...
- ...Altrimenti, uno dei due operandi deve essere convertito per avere operandi dello stesso tipo
  - Gerarchia dei tipi di dato
  - Promozione da tipo inferiore a tipo superiore nella gerarchia

Gerarchia dei Tipi in C

- Tipi di dato organizzati in gerarchia da "più semplice" a "più complesso"
  - `long double`
  - `double`
  - `float`
  - `unsigned long int`
  - `long int`
  - `unsigned int`
  - `int`
- Operazione tra due tipi diversi: promosso quello "più in basso"
  - *int* + *double*: il primo operando è convertito a `double`
  - *float* / *unsigned int*: il secondo operando è convertito a `float`

Ancora su Valutazione di Espressioni

- Attenzione!!! La promozione avviene solo se gli operandi sono di tipo diverso
- La conversione è fatta in base ai tipi degli operandi, non in base al risultato!
  - Eesempio: `5 / 2` non comporta conversioni
- Se due operandi sono dello stesso tipo, il risultato ha il tipo degli operandi
  - Rischio di overflow (esempio: *int* + *int* non viene convertito a `long int` in caso di overflow)
  - Se gli operandi sono interi, divisione intera
  - ...

Espressioni con Variabili Intere

- Variabili di tipo `char` e `short int` sono sempre convertite in `int` quando sono usate in un'espressione
- Variabili di tipo `unsigned short int` sono convertite in `int` se possibile (se il valore è rappresentabile come `int`), in `unsigned int` altrimenti
- Nota: queste promozioni di interi avvengono sempre, indipendentemente dagli altri operandi

Problemi con Conversioni Implicite

- Come visto, talvolta le conversioni implicite causano perdita d'informazione...
- In altri casi, si possono verificare problemi maggiori

```
char c;

...
c = 782;
```

- Il tipo `char` è rappresentato su 8 bit
  - Può codificare solo fino a $2^8 = 256$ valori
- Il valore 782 non è codificabile
- L'assegnamento `c = 782` memorizza in `c` un valore diverso da 782

Problemi con Conversioni da Interi a Virgola Mobile

- I numeri in virgola moblie possono rappresentare valori molto grandi...C
- ...Ma hanno dei "buchi" nei valori rappresentabili
- Ci sono valori interi (`long int`) non rappresentabili come `float`

```
long int i = 2147483600;
float f;

f = i;
printf("i=%ld   f=%f\n", i, f);
```

- 2147483600 è codificabile come `long int`, ma non è codificabile senza approssimazioni come `float`
- `f` assume il valore codificabile più vicino a 2147483600, ma c'e' un errore di approssimazione!

Conversioni Esplicite (Cast)

- Per evitare i problemi dovuti alle conversioni implicite, è possibile richiedere esplicitamente la conversione di un valore
- Cast: (*tipo*) *valore* $\rightarrow$ converte *valore* in *tipo*
  - Esempio: `(double)1` codifica il valore `1` come numero in virgola mobile a precisione doppia
- Se `d` è una variabie di tipo `double`, `d = 5 / 2` assegna a `d` il valore 2
  - Perché?
- `d = (double)5 / 2` assegna a `d` il valore 2.5
- In caso di incertezza sulle conversioni implicite, meglio aggiungere qualche cast!

L'Operatore di Cast

- Il nome di un tipo di dato fra parentesi rappresenta l'operatore di cast (conversione a tipo) verso il tipo specificato
  - Esempio: `(int)` $\rightarrow$ operatore di cast a intero
  - `(int)d` converte il valore della variabile `d` in intero
- Gli operatori di cast hanno la precedenza su tutti gli altri
  - `i = 2 + (int)d`: prima converte il valore di `d` in intero, poi effettua la somma (fra due interi)
  - `(int)2.5 * 4` viene valutato a 8