Tipi di dato, memoria e conversioni

Alessandra Giordani
agiordani@disi.unitn.it
Lunedì 7 maggio 2011
http://disi.unitn.it/~agiordani/

Il linguaggio C è

- *esplicitamente tipato*: occorre esplicitamente associare ad ogni variabile il tipo di dato che essa rappresenta
- *staticamente tipato*: il tipo di una variabile non puo cambiare durante il suo ciclo di vita;
- *fortemente/debolmente tipato*: non è possibile mischiare tipo di dato diversi nella stessa espressione, o se è possibile è richiesto che le conversioni siano esplicite (<op> casting). Però, in alcuni casi questa conversione esplicita non è necessaria.

Tipi di dato fondamentali

- int e' il tipo di dato che consente di rappresentare numeri interi all'interno di un programma.
  - Su un architettura a 32 bit, un int è rappresentato in complemento a 2 utilizzando 4 byte;
- float permette di rappresentare numeri in virgola mobile
  - usando 4 byte con una precisione di circa 7 cifre decimali;
- double permette di rappresentare numeri decimali in virgola mobile in doppia precisione
  - usando 8 byte con una precisione di circa 15 cifre decimali;
- char permette di rappresentare caratteri alfabetici.
  - usando un byte. La sequenza di bit che rappresenta un carattere puo anche essere interpretata come un intero senza segno

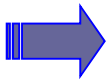
I tipi di dati scalari

- I tipi aritmetici, i tipi enumerativi ed i puntatori vengono detti tipi scalari, poiché i valori che li compongono sono distribuiti su una *scala lineare*, su cui si può stabilire una relazione di ordine totale
- La dichiarazione fornisce al compilatore le informazioni relative al numero di byte da allocare e alle modalità di interpretazione di tali byte
- Le parole chiave char, int, float, double, ed enum descrivono i tipi base; short, long, signed, unsigned sono i qualificatori che modificano i tipi base

Qualificatori short/long

- Al tipo int possono essere assegnate dimensioni diverse su architetture distinte (tipicamente 4 o 8 byte)
- Il tipo int rappresenta il formato “naturale” per il calcolatore, ossia il numero di bit che la CPU manipola normalmente in una singola istruzione
- Supponiamo che int corrisponda a celle di memoria di 4 byte:
  - Il tipo short int corrisponde generalmente a 2 byte
  - Il tipo long int a 4/8 byte
- Nelle dichiarazioni di interi short/long la parola int può essere omessa

```
short int j;
long int k;
```



```
short j;
long k;
```

Qualificatori unsigned/signed

- Si possono individuare casi in cui una variabile può assumere solo valori positivi (ad es., i contatori)
- Il bit più significativo non viene interpretato come bit di segno
- Esempio: una variabile short int può contenere i numeri interi compresi fra –32768 e 32767, mentre una variabile dichiarata unsigned short int può contenere valori da 0 a 65535 (2^{16}-1)

unsigned (int) p;

- Lo specificatore signed consente di definire esplicitamente una variabile che può assumere valori sia positivi che negativi
- Normalmente signed è superfluo, perché i numeri interi sono con segno *per default*

I tipi interi

Tipo	Byte	Rango
int	4	da -2^{31} a $2^{31}-1$
short int	2	da -2^{15} a $2^{15}-1$
long int	4	da -2^{31} a $2^{31}-1$
	8	da -2^{63} a $2^{63}-1$
unsigned int	4	da 0 a $2^{32}-1$
unsigned short int	2	da 0 a $2^{16}-1$
unsigned long int	4	da 0 a $2^{32}-1$
signed char	1	da -2^{7} a $2^{7}-1$
unsigned char	1	da 0 a $2^{8}-1$

Dimensione e rango dei valori dei tipi interi sulla macchina di riferimento

Operatore sizeof()

```
#include <stdio.h>
int main()
{
    printf("Size of (in bytes):\n");
    printf("int: %d\n", sizeof(int));
    printf("short int: %d\n", sizeof(short int));
    printf("long int: %d\n", sizeof(long int));
    printf("char: %d\n", sizeof(char));
    printf("float: %d\n", sizeof(float));
    printf("double: %d\n", sizeof(double));
    printf("long double: %d\n", sizeof(long double));
    return 0;
}
```

Le combinazioni di tipi

- Nelle espressioni, il C ammette la combinazione di tipi aritmetici:

  num=3*2.1;

  l'espressione è la combinazione di un int ed un double; inoltre num potrebbe essere di qualunque tipo scalare, eccetto un puntatore
- Per associare un significato alle espressioni contenenti dati di tipi diversi, il C effettua automaticamente un insieme di *conversioni implicite*:

  3.0+1/2

  verrebbe valutata 3.0 anziché 3.5, dato che la divisione viene effettuata in aritmetica intera

Conversioni implicite

- Le conversioni implicite vengono effettuate in quattro circostanze:
  - Conversioni di assegnamento — nelle istruzioni di assegnamento, il valore dell'espressione a destra viene convertito nel tipo della variabile di sinistra
  - Conversioni ad ampiezza intera — quando un char od uno short int appaiono in un'espressione vengono convertiti in int; unsigned char ed unsigned short vengono convertiti in int, se int può rappresentare il loro valore, altrimenti sono convertiti in unsigned int
  - In un espressione aritmetica, gli oggetti sono convertiti per adeguarsi alle regole di conversione dell'operatore
  - Può essere necessario convertire gli argomenti di funzione

Conversioni implicite di assegnamento

- Per le conversioni di assegnamento, sia j un int e si consideri…

  j=2.6;

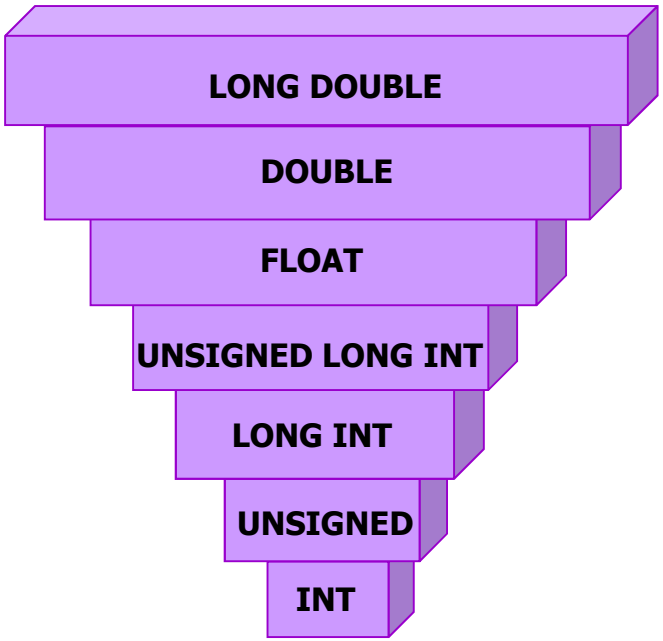
  Prima di assegnare la costante di tipo double, il compilatore la converte in int, per cui j assume il valore intero 2 (agisce per troncamento, non per arrotondamento)

- La conversione ad ampiezza intera o *promozione ad intero*, avviene generalmente in modo trasparente

  int ch='a'+32

Conversioni implicite - gerarchia

- L'analisi di un'espressione da parte del compilatore ne comporta la suddivisione in sottoespressioni; gli operatori binari impongono operandi dello stesso tipo: l'operando il cui tipo è "gerarchicamente inferiore" viene convertito al tipo superiore:



Esempio**:** La somma fra un int e un double (1+2.5) viene valutata come (1.0+2.5)

Le conversioni di tipo esplicite: cast

- In C, è possibile convertire esplicitamente un valore in un tipo diverso effettuando un cast
- Per realizzare una conversione di tipo esplicita di un'espressione, si pone tra parentesi tonde, prima dell'espressione, il tipo in cui si desidera convertire il risultato

```
#include<stdio.h>

int main()
{
    int j, k;
    float f;

    printf("Inserire due valori j,k. Calcorero j/k con e senza casting\n");
    scanf("%d %d", &j, &k);
    f=j/k;
    printf("valore di f=%d/%d senza casting= %f\n",j,k,f);
    f=(float)j/k; //conversione esplicita
    printf("valore di f=%d/%d con casting= %f\n",j,k,f);
    f=(j*1.0)/k; //conversione implicita
    printf("valore di f=%d/%d con 1.0= %f\n",j,k,f);
    return 0;
}
```

La combinazione di floating-point

- L'uso congiunto di float, double e long double nella stessa espressione fa sì che il compilatore, dopo aver diviso l'espressione in sottoespressioni, ampli l'oggetto più corto di ogni coppia associata ad un operatore binario
- In molte architetture, i calcoli effettuati sui float sono molto più veloci che quelli relativi a double e long double…
  - I tipi di numeri più ampi dovrebbero essere impiegati solo quando occorre una grande precisione o occorre memorizzare numeri molto grandi
- Possono esserci problemi quando si effettuano conversioni da un tipo più ampio ad uno meno ampio
  - Perdita di precisione
  - Overflow

Le tipologie di costanti intere – 1

- Oltre alle costanti decimali, il C permette la definizione di costanti ottali ed esadecimali
- Le costanti ottali vengono definite anteponendo al valore ottale la cifra 0
- Le costanti esadecimali vengono definite anteponendo la cifra 0 e x o X

Decimale	Ottale	Esadecimale
3	03	0x3
8	010	0X8
15	017	0xF
16	020	0x10
21	025	0x15
–87	–0127	–0x57
187	0273	0xBB
255	0377	0Xff

Le tipologie di costanti intere – 2

- Esempio: Leggere un numero esadecimale da terminale e stampare gli equivalenti ottale e decimale

```
/* Stampa gli equivalenti ottale e decimale
 * di una costante esadecimale
 */
#include<stdio.h>

int main()
{
    int num; char a;

    printf("Digitare una costante esadecimale: ");
    scanf("%x", &num);
    printf("L'equivalente decimale di %x e' %d\n", num, num);
    printf("L'equivalente ottale di %x e' %o\n", num, num);
    return 0;
}
```

Le costanti floating-point

- Le costanti floating–point sono, per default, di tipo double
- Lo standard ANSI consente tuttavia di dichiarare esplicitamente il tipo della costante, mediante l'uso dei suffissi f/F o l/L, per costanti float e long double, rispettivamente

```
#define PI  3.14159

float area_of_circle(float radius)
{
    float area;

    area = PI * radius * radius;
    return area;
}
```

Conversione da stringa ad intero

```c
#include <stdio.h>
#include <stdlib.h>

main()
{
     char anno_nascita[5], anno_corrente[5];
     int anni;

      printf("Inserire l'anno di nascita: ");
     scanf("%s", anno_nascita);
     printf("Inserire l'anno corrente: ");
     scanf("%s", anno_corrente);

     /* atoi() converte una stringa in un intero */
     anni = atoi(anno_corrente) - atoi(anno_nascita);
     printf("Eta': %d\n", anni);
     exit(0);
}
```

Esercizio per la prova pratica

- Scrivere un main che
  - Legga una frase e un carattere
  - Conti quante volte il carattere appare nella stringa
  - Non sia case sensitive (non tenga conto della differenza tra maiuscole e minuscole)
- Esempi
  - “pratica” ‘a’ → 3 occorrenze
  - “Prova pratica” ‘p’ → 2 occorrenze