Variabili e Tipi

Proprio come in matematica, le variabili sono nomi associati a valori arbitrari, non noti o comunque variabili. Questi valori possono essere di vario tipo: numeri interi, razionali, booleani (vero o falso) od anche caratteri. Il C richiede esplicitamente il tipo di una variabile quando la si dichiara, questo per facilitare la compilazione, e si dirà quindi che il C è un linguaggio tipizzato. Tecnicamente, la variabile rappresenta uno spazio in memoria che contiene dei dati che possono essere modificati; il tipo di dato serve appunto per specificare che porzione di memoria *allocare* a quel dato, ad esempio, una variabile di tipo intero (`int`) occuperà meno di un tipo carattere (`char`), perché il numero di caratteri (e.g. quelli della tastiera) è molto minore rispetto alla quantità di numeri interi rappresentabili (e.g. tipicamente un intero può rappresentare al massimo 4294967295 numeri diversi, mentre i caratteri ASCII sono solo 128). Bisogna ricordare che le dimensioni dei vari tipi sono dipendenti dalla piattaforma su cui si compila il codice: una macchina a 32 bit o una a 64 bit.
Il primo esempio che facciamo chiarirà le idee:

```c
#include <stdio.h>
int main() {

  int x; // è una dichiarazione: il compilatore allocerà un tot di byte in
   memoria per contenere "x";
  x=4; //inizializzazione della variabile x, il compilatore metterà il valore
    4 nella porzione di memoria precedentemente riservata a x
  x=x+10; //partendo da destra, viene aggiunto 10 alla x (che vale 4) e l'
   operatore assegnamento (=) assegna 4+10, cioè il valore 14 alla porzione
   di memoria x (la nostra variabile)

  printf("x vale: %d\n",x);//legge il contenuto della variabile x, cioè 14

  return 0;
}
```

L'otput del programma sarà dunque:

```
$ x vale:  14
```

Per sapere le dimensioni (in byte) dei vari tipi in C possiamo scrivere un programma del genere:

```c
#include <stdio.h>
int main() {

  char cr;//carattere ad es: "a" o "1"
  int a; /*solito intero */ short b; //intero short
  double c; /* intero doppio */ float f; //virgola mobile
  int vett[10]; //vettore di dieci interi
  char* ptr; //puntatore (*) ad un char (un puntatore è sempre e solo un
   indirizzo di memoria quindi 32 o 64 bit)

  // sizeof() restituisce la dimensione di un "oggetto", anche di un vettore
  printf("char = %lu byte\n", sizeof(cr) );
    printf("int = %lu byte\n", sizeof(a) );
    printf("short (int) = %lu byte\n", sizeof(b) );
    printf("double (int) = %lu byte\n", sizeof(c) );
    printf("float (int) = %lu byte\n", sizeof(f) );
  printf("char* (puntatore) = %lu byte\n", sizeof(ptr) );
  printf("int vett[10], cioè 10*int = %lu byte\n", sizeof(vett) );

  return 0; }
```

Compilato con gcc su un computer a 64 bit sarà (ma non necessariamente):

```
     $ char = 1 byte
int = 4 byte
short (int) = 2 byte
double (int) = 8 byte
float (int) = 4 byte
char* (puntatore) = 8 byte
int vett[10], cioè 10*int = 40 byte
```

Esercizio

Si propone infine un esercizo, la cui soluzione fatta da Elena:

> "Scrivere un programma che, data una somma di euro da pagare, stampi quali e quanti tagli di banconote (e monete) sono necessari per pagare quella somma. Si consideri il taglio da 50 come il massimo."

```c
#include <stdio.h>
int main() {

  int tot;
  // tot=157;
  scanf("%d", &tot); //chiediamo all'utente di inserire un valore da tastiera

  int banc50;
  banc50=tot/50;
  int resto;
  resto=tot%50;
  printf("le banconote da 50 sono:%d e il resto:%d\n",banc50,resto);

  int banc20;
  banc20=resto/20;

  resto=resto%20;

  int banc10;
  banc10=resto/10;
  resto=resto%10;

  printf("le banconote da 20 sono:%d\n",banc20);
  printf("le banconote da 10 sono:%d\n",banc10);

  int banc5;
  banc5=resto/5;
  resto=resto%5;

  int mon2;
  mon2=resto/2;
  resto=resto%2;

  int mon1;
  mon1=resto/1;
  resto=resto%1;
  printf("le banconote da 5 sono:%d\n",banc5);
  printf("le monete da due sono:%d\n",mon2);
  printf("le monete da uno sono:%d\n",mon1);

return 0; }
```