2.5 Variabili carattere

Sino a questo momento abbiamo lavorato con variabili di tipo intero. Introduciamo ora variabili di tipo carattere, che scopriremo avere in comune con le prime molto di più di quanto potremmo supporre.

Le variabili di tipo carattere assumono valori alfanumerici che comprendono le lettere dell'alfabeto minuscole e maiuscole, le cifre decimali, la punteggiatura e altri simboli. Scrivendo

```
char x, y, z;
```

la parola chiave `char` specifica che gli identificatori `x`, `y` e `z` che la seguono si riferiscono a variabili di tipo carattere. La definizione fa sì che venga riservato uno spazio in memoria sufficiente per contenere un carattere alfanumerico. La dimensione può variare rispetto all'implementazione; molte versioni riservano per i `char` uno spazio di un byte, il che permette di fare riferimento a 256 configurazioni di bit distinti e quindi individuare ogni carattere del codice in uso sulla macchina. I codici di più ampio utilizzo sono l'ASCII e l'EBCDIC. Per assegnare un valore costante a una variabile `char` lo si deve racchiudere tra apici singoli:

```
x = 'A';
y = ';';
z = '&';
```

Come si può facilmente dedurre dal codice ASCII riportato in Appendice D, si hanno le seguenti corrispondenze.

Carattere	Decimale	Esadecimale	Binario
A	65	41	01000001
;	59	3B	00111011
&	38	26	00100110

A ogni carattere presente nel codice corrisponde una rappresentazione numerica univoca, per cui è possibile confrontare due simboli non solamente con uguaglianze e diseguaglianze, ma anche per mezzo di tutti gli altri operatori relazionali.

"A" (65) è maggiore di ";" (59) che a sua volta è maggiore di "&" (38). Osservando il codice ASCII possiamo vedere che le lettere alfabetiche maiuscole sono in ordine crescente da A (65) a Z (90), le minuscole vanno da a (98) a z (122) e le cifre decimali da 0 (48) a 9 (57). Dunque ha perfettamente senso l'istruzione condizionale

```
if(x=='A')
  printf("Si tratta di una A");
```

ma anche

```
if(x>='A' && x<='Z')
  printf("Si tratta di una lettera maiuscola");
```

Per poter visualizzare dei `char` con una `printf` si deve come al solito indicarne il formato; per esempio:

```
printf("%c %c %c", x, y, z);
```

Come abbiamo già osservato, i simboli di percentuale tra i doppi apici definiscono il formato di stampa delle corrispondenti variabili; questa volta le `c` (*character*) specificano che si tratta di caratteri. L'esecuzione dell'istruzione restituisce

```
A ; &
```

mentre l'istruzione

```
printf("%c%c%c", x, y, z);
```

restituisce

```
A;&
```

Naturalmente si possono stampare valori di tipi diversi all'interno della stessa `printf`, indicandone il formato nell'ordine corretto. Se `n` e `m` sono variabili `int` si può scrivere

```
x = 'a'; y = 'b';
```

```
n = 100; m = 4320;
printf("%c = %d   %c = %d", x, n, y ,m);
```

che restituisce

```
a = 100   b = 4320
```

Per mezzo dell'istruzione `scanf` si possono poi inserire caratteri a tempo di esecuzione:

```
scanf("%c", &x)
```

Il formato `%c` indica che si tratta di un carattere, così come `%d` indicherebbe che si tratta di un intero da visualizzare in formato decimale .

Esistono altre due funzioni standard di input/output, cui si può far riferimento per mezzo di `<stdio.h>`, che permettono di leggere e scrivere caratteri: `getchar` e `putchar`.

Se `x` è una variabile di tipo carattere,

```
x = getchar();
```

bloccherà il programma in attesa di un carattere introdotto da tastiera. Si noti che la presenza delle parentesi tonde è necessaria anche se dentro non vi è racchiuso alcun argomento.

Per visualizzare un carattere abbiamo invece la funzione

```
putchar(x);
```

Vediamo una semplice applicazione di queste due funzioni. I due programmi del Listato 2.3 hanno funzionamenti identici: i caratteri dovranno essere digitati uno dietro l'altro e successivamente dovrà essere battuto un Invio.

> ✓ **NOTA**
>
> Se il programma dovesse prevedere l'immissione di più valori in tempi diversi, l'inserimento di un carattere potrebbe costituire un problema, dato che la digitazione del tasto di Invio da tastiera corrisponde a un carattere accettato da `scanf("%c",…)`. In tali casi verrà utilizzata un'opportuna ulteriore lettura di un carattere in una variabile ausiliaria tramite un'istruzione del tipo `scanf("%c", pausa)` o `pausa=getchar()`.

```
#include <stdio.h>

main()
{
char x, y, z;
printf("digita tre carat.: ");
scanf("%c%c%c", &x, &y, &z);


printf("Hai digitato: ");
printf("%c%c%c\n", x, y, z);


}
```

```
#include <stdio.h>

main()
{
char x, y, z;
printf("digita tre carat.: ");
x = getchar();
y = getchar();
z = getchar();
printf("Hai digitato: ");
putchar(x);
putchar(y);
putchar(z);
putchar('\n');
 }
```

Listato 2.3 Due esempi di input/output di caratteri