6.1 Definizione

Una variabile di tipo `char` consente di memorizzare un singolo carattere. Molto spesso, però, è comodo poter trattare come una sola unità un insieme di caratteri alfanumerici, detto *stringa*; a questo scopo si possono utilizzare gli array di `char`. La linea di codice

```
char a[10];
```

dichiara un vettore costituito da dieci caratteri.

```
char frase[] = "Analisi, requisiti ";
```

dichiara invece l'array monodimensionale di caratteri `frase`, il cui numero di elementi è determinato dalla quantità di caratteri presenti tra doppi apici più uno, il carattere *null* (`\0`) che chiude la stinga (Figura 6.1).

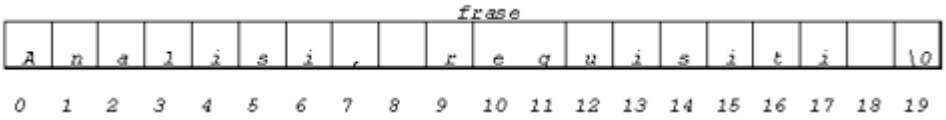

Figura 6.1 Contenuto dell'array `frase[]`.

Il carattere `\0` è il primo del codice ASCII, corrisponde alla notazione binaria 00000000 e non ha niente a che vedere con il carattere `0` che corrisponde a 00110000.

È importante osservare la differenza tra le due inizializzazioni:

```
char d = 'r';
     char b[] = "r";
```

La prima assegna alla variabile `d` di tipo `char` il valore `r`, la seconda assegna all'array `b[]` la sequenza di caratteri `r` e `\0`; in quest'ultimo caso si tratta effettivamente di una stringa. Naturalmente, quando si desidera far riferimento a un carattere si deve inserirlo tra apici singoli: per esempio

```
b[2] = 't';
```

assegna al terzo elemento dell'array `b` il carattere `t`.

Il carattere terminatore `\0` ci permette di trattare le stringhe senza conoscere a priori la dimensione.

Il programma del Listato 6.1 consente di verificare la corrispondenza tra ogni carattere presente in una stringa e il suo equivalente valore all'interno del codice ASCII, espresso nei sistema decimale e ottale.

```
/* Visualizzazione caratteri di una stringa */

#include <stdio.h>

char frase[] = "Analisi, requisiti ";

main()
{
int i=0;
while(frase[i]!='\0') {
  printf("%c = %d = %o \n", frase[i], frase[i], frase[i]);
  i++;
}
}
```

Listato 6.1 Visualizzazione di differenti rappresentazioni di caratteri

Il ciclo `while` permette di fare la scansione, uno a uno, dei caratteri della stringa. Viene controllato se il carattere in esame è `\0`, nel qual caso non ci sono più caratteri da esaminare e l'iterazione ha termine. L'istruzione `printf` visualizza a ogni ciclo un elemento dell'array, in tre formati differenti:

```
printf("%c = %d = %o \n", frase[i], frase[i], frase[i]);
```

Il primo formato, specificato da `%c`, indica il carattere ASCII stesso, il secondo e il terzo sono i suoi corrispondenti codici espressi nel sistema decimale (`%d`) e ottale (`%o`). Questo gioco di corrispondenze tra caratteri e numeri interi, definite dal codice ASCII, è sempre valido e offre grande libertà al programmatore.

L'esecuzione del programma dà il risultato:

```
A = 65  = 101
     n = 110 = 156
     a = 97  = 141
     l = 108 = 154
     i = 105 = 151
     s = 115 = 163
     i = 105 = 151
     , = 44  = 54
       = 32  = 40
     r = 114 = 162
     e = 101 = 145
     q = 113 = 161
     u = 117 = 165
     i = 105 = 151
     s = 115 = 163
     i = 105 = 151
     t = 116 = 164
     i = 105 = 151
       = 32  = 40
```

Comunque, se si desidera la visualizzazione dell'intera stringa, è possibile usare l'istruzione `printf` tramite la specifica del formato `%s`:

```
printf("%s", frase);
```

Tale istruzione, se inserita nel Listato 6.1 , restituirebbe:

```
Analisi, requisiti
```

In questo caso è l'istruzione `printf` stessa che provvede a stampare carattere per carattere la stringa e a bloccarsi nel momento in cui identifica il carattere `\0`.