7.9 void

Abbiamo trattato il passaggio di parametri e la restituzione di un valore da parte di una funzione. Prendiamo ora in esame funzioni che non restituiscono alcun valore, e funzioni che non hanno parametri. In entrambi i casi il C mette a disposizione un "tipo" speciale detto `void`.
Tipico esempio di funzioni che non restituiscono alcun valore è quello delle funzioni il cui scopo è la visualizzazione di un messaggio o, più in generale, la produzione di un'uscita su uno dei dispositivi periferici. Queste funzioni sono talvolta conosciute con il curioso nome di funzioni "lavandino" (*sink*, in inglese) poiché prendono dati che riversano in una qualche uscita, senza ritornare niente al chiamante. Un esempio di funzione "lavandino" è la funzione `stampa_bin` (Listato 7.7).

```
#include <stdio.h>
#define DIM_INT 16

void stampa_bin ( int );

main()
{
  char resp[2];
  int  num;

  resp[0] = 's';

  while( resp[0] == 's' ) {
    printf("\nInserisci un intero positivo: ");
    scanf("%d", &num);
    printf("La sua rappresentazione binaria è: ");

    stampa_bin( num );

    printf("\nVuoi continuare? (s/n): ");
    scanf("%s",resp);
  }
}

void stampa_bin( int v )
{
  int i, j;
  char a[DIM_INT];

  if (v == 0)
    printf("%d", v);
  else  {
    for( i=0; v != 0; i++)   {
      a[i] = v % 2;
      v /= 2;
    }
```

```
    for(j = i-1 ; j >= 0; j--)
      printf("%d", a[j]);
  }
}
```

Listato 7.7 esempio di funzione “lavandino”

Un esempio di chiamata della funzione è:

```
Inserisci un intero positivo: 13
La sua rappresentazione binaria è: 1101
Vuoi continuare? (s/n): s
Inserisci un intero positivo: 64
La sua rappresentazione binaria è: 1000000
Vuoi continuare? (s/n): n
```

La funzione `stampa_bin` divide ripetutamente per 2 il decimale `v` e memorizza i resti delle divisioni intere nel vettore `a[]`, che poi legge a ritroso per visualizzare l’equivalente binario del decimale `v`. Come il lettore avrà osservato, sia nella dichiarazione

```
void stampa_bin( int );
```

sia nella definizione

```
void stampa_bin( int v )
{
...
}
```

si usa lo specificatore di tipo `void` per indicare l’assenza di un valore di ritorno. Viceversa, quando per una funzione non è specificato il tipo `void` per il valore di ritorno, nel blocco istruzioni della funzione è logico che sia presente per lo meno una istruzione `return`.
Il tipo `void` è usato anche per le funzioni che non assumono alcun parametro. Un esempio di funzione che non ha parametri e che non restituisce alcun valore è rappresentato da `mess_err` (Listato 7.8).

```
#include <stdio.h>

void mess_err( void );

main()
{
  int a, b, c;

  printf("Inserire dividendo:");
  scanf("%d", &a);
  printf("Inserire divisore:");
  scanf("%d", &b);
  if (b != 0) {
    c = a/b;
    printf("%d diviso %d = %d\n", a, b, c);
  }
else
  mess_err();
}

void mess_err( void )
{
  int i;
```

```
  char c;

  for (i = 0; i <= 20; i++) printf("\n");
  printf("                    ERRORE! DENOMINATORE NULLO");
  printf("\n Premere un tasto per continuare\n");
  scanf("%c%c", &c, &c);
}
```

Listato 7.8 Funzioni senza parametri

La funzione `mess_err` non prende parametri e non restituisce alcun valore; essa ha il solo compito di visualizzare un messaggio di errore nel caso di inserimento di un denominatore nullo. In C una funzione che non prende parametri può essere anche designata semplicemente dalle sole parentesi tonde, senza usare la parola chiave `void`. Per esempio:

```
void mess_err();
```

e

```
mess_err();
```

sono dichiarazioni valide. Nel secondo caso, però, `mess_err` viene considerata una funzione il cui eventuale valore di ritorno è di tipo `int`, anche se in realtà non ha nessun valore di ritorno. Non è forse questo il caso anche della funzione `main`? Abbiamo continuato a definirla con:

```
main()
{
...
}
```

a indicare il fatto che non ritorna nessun valore – è quindi di tipo `void` – e che non assume parametri. Una equivalente (e forse anche più corretta) definizione di `main` potrebbe essere:

```
void main( void )
{
...
}
```

Il fatto è che prima dello standard ANSI il C non prevedeva la parola chiave `void`, e oggi, per motivi di compatibilità, sono ammesse le due notazioni, con e senza `void`. Attualmente lo standard stabilisce che `main` sia implicitamente definita come funzione `void`, mentre in passato veniva comunemente definita di tipo `int`. Per evidenti motivi di leggibilità si consiglia caldamente di far uso di `void` tutte le volte che è necessario, soprattutto al fine di indicare che la funzione non ritorna nessun valore ■.