7.7 Chiamata di una funzione

Una funzione C viene invocata facendo riferimento al nome e passando a essa una lista di parametri conforme in tipo, numero e ordine alla lista dei parametri formali elencata nella definizione della funzione stessa (Listato 7.4).

```
#include <stdio.h>

double area(float, float, char);

main()
{
  float b, h;
  double a;
  char p;

  printf("Inserire poligono (Triangolo/Rettangolo): ");
  scanf("%c", &p);

  printf("\nInserire base: ");
  scanf("%f", &b);
  printf("\nInserire altezza  : ");
  scanf("%f", &h);

  a = area( b, h, p);

  printf("Il poligono (b = %f, h = %f) ha area %f\n", b, h, a);
}

double area(float base, float altezza, char poligono)
{
  switch (poligono) {
    case 'T':     return (base * altezza/2.0);
    case 'R':     return (base * altezza);
    default :     return -1;
  }
}
```

Listato 7.4 Chiamata di funzione

Il contenuto delle variabili di tipo `float b, h` e di tipo `char p` vengono passati alla funzione `area` per mezzo dell'istruzione

```
a = area( b, h, p);
```

Le variabili `b`, `h` e `p` sono dette *parametri attuali* poiché contengono i valori di ingresso in quella specifica chiamata di funzione con i quali si può calcolare l'area del poligono. Al posto di una variabile si può comunque passare una costante dello stesso tipo, per esempio:

```
a = area( b, h, 'T');
```

dove il valore `T` viene immesso nel parametro formale `poligono`. Non sono invece corrette le invocazioni:

```
a = area('T', b, h);
a = area( b, h);
a = area( b, h, 'T', x);
a = area( b, h, x);
```

dove `x` è una variabile `int`; il passaggio di parametri è errato o per ordine o per numero o per discordanza di tipo.

✓ **NOTA**

Per mezzo della chiamata di funzione si concretizza il concetto di “scatola nera”: il programma chiamante sfrutta i servizi di una funzione conoscendo soltanto il nome e l’interfaccia (tipo, ordine e numero dei parametri formali) e disinteressandosi dei dettagli implementativi.
Le funzioni offrono anche un altro vantaggio: possono essere invocate quante volte lo si desidera senza produrre duplicazione di codice. In pratica, a n chiamate, con $n \geq 1$, corrisponde sempre una sola definizione. Nel Listato 7.5, per esempio, la funzione `area` è chiamata due volte dalla funzione `main` per calcolare l’area del triangolo e del rettangolo, entrambi di base `b` e altezza `h`.

```
#include <stdio.h>

double area(float, float, char);

main()
{
  float b, h;
  double tri, ret;

  printf("Inserire base: ");
  scanf("%f", &b);
  printf("Inserire altezza: ");
  scanf("%f", &h);

  tri = area(b, h, 'T');

  ret = area(b, h, 'R');

  printf("Il triangolo (b = %f, h = %f) ha area %f\n", b, h, tri);
  printf("Il rettangolo (b = %f, h = %f) ha area %f\n", b, h, ret);
}

double area(float base, float altezza, char poligono)
{
  switch (poligono) {
    case 'T':     return (base * altezza/2.0);
    case 'R':     return (base * altezza);
    default :     return -1;
  }
}
```

Listato 7.5 Le funzioni come strumento di riutilizzo del codice