## 1.2 Variabili e assegnamenti

Supponiamo di voler calcolare l'area di un rettangolo di base 3 e altezza 7; osserviamo nel Listato 1.3 il programma che risolve il problema.

```
/* Calcolo area rettangolo */

#include <stdio.h>

main()
{
  int base;
  int altezza;
  int area;

  base = 3;
  altezza = 7;
  area = base*altezza;

  printf("%d\n", area);
}
```

Listato 1.3 Uso di variabili

Per rendere evidente la funzione espletata dal programma abbiamo inserito un commento:

```
/* Calcolo area rettangolo */
```

I commenti possono estendersi su più linee e apparire in qualsiasi parte del programma; devono essere preceduti da `/*` e seguiti da `*/`: tutto ciò che appare nelle zone così racchiuse non viene preso in considerazione dal compilatore e non ha nessuna influenza sul funzionamento del programma. Un altro modo per inserire un commento è farlo precedere da `//`, ma in questo caso deve terminare a fine linea:

```
// Calcolo area rettangolo
```

Dopo il `main()` e la parentesi graffa aperta sono presenti le *dichiarazioni* delle variabili (intere) necessarie:

```
int base;
int altezza;
int area;
```

La parola chiave `int` specifica che l'identificatore che lo segue si riferisce a una variabile numerica di tipo intero; dunque `base`, `altezza` e `area` sono variabili di questo tipo. Anche le dichiarazioni – così come le altre istruzioni – devono terminare con un punto e virgola. Nel nostro esempio, alla dichiarazione della variabile corrisponde anche la sua *definizione*, la quale fa sì che le venga riservato uno spazio in memoria centrale. Il *nome* di una variabile la identifica, il suo *tipo* ne definisce la dimensione e l'insieme delle operazioni che si possono effettuare su di essa. La dimensione può variare rispetto all'implementazione; alcune versioni del C riservano agli `int` uno spazio di quattro byte, il che permette di poter lavorare su interi che vanno da –2147483648 a +2147483647; altre versioni riservano due byte (gli interi permessi vanno da –32768 a +32767) . Tra le operazioni fra `int` consentite vi sono: somma, sottrazione, prodotto e divisione, che corrispondono rispettivamente agli operatori `+`,`-`,`*`,`/`.

L'istruzione:

```
base = 3;
```

assegna alla variabile `base` il valore 3; inserisce cioè il valore (3) che segue l'operatore = nello spazio di memoria riservato alla variabile (`base`). Effetto analogo avrà `altezza = 7`. L'assegnamento è dunque realizzato mediante l'operatore =.

L'istruzione

```
area = base*altezza;
```

assegna alla variabile `area` il prodotto dei valori di `base` e `altezza`, mentre l'ultima istruzione,

```
printf("%d\n", area);
```

visualizza 21, il valore della variabile `area`. Tra i doppi apici, il simbolo di percentuale `%` specifica che il carattere che lo segue definisce il formato di stampa della variabile `area`; `d` (*decimal*) indica che si desidera la visualizzazione di un intero nel sistema decimale. Invece `\n` provoca come abbiamo già visto un salto a linea nuova dopo la visualizzazione.

In generale la struttura di un programma C prevede che le variabili possano essere dichiarate sia dopo `main()` e la parentesi graffa aperta, e anteriormente alle istruzioni operative come nell'esempio visto, sia prima di `main()`. La struttura generale risulta quindi la seguente:

```
inclusione librerie
dichiarazioni di variabili
main
{
  dichiarazioni di variabili
  istruzione 1
  istruzione 2
  istruzione 3
  ...
  istruzione N
}
```

Si tenga presente che nella sintassi il punto e virgola fa parte dell'istruzione stessa.

Le dichiarazioni delle variabili dello stesso tipo possono essere scritte in sequenza separate da una virgola; per esempio, nel Listato 1.3 avremmo potuto scrivere:

```
int   base, altezza, area;
```

Dopo la dichiarazione di tipo sono specificati gli *identificatori* di variabile, che possono essere in numero qualsiasi, separati da virgola e chiusi da un punto e virgola. In generale, quindi, la dichiarazione di variabili ha la forma:

```
tipo  lista di identificatori;
```

Esistono inoltre regole da rispettare nella costruzione degli identificatori, che devono iniziare con una lettera o con un carattere di sottolineatura `_` e possono contenere lettere, cifre e `_`. La lunghezza può essere qualsiasi ma caratteri significativi sono spesso i primi 255 (247 secondo lo standard), anche se nelle versioni del C meno recenti questo limite scende a 32 o anche a 8 caratteri. Le lettere maiuscole sono considerate diverse dalle corrispondenti minuscole. Esempi di identificatori validi sono: `nome1`, `cognome2`, `cognome_nome`, `alberoBinario`, `volume`, `VOLUME`, `a`, `b`, `c`, `x`, `y`; al contrario non sono corretti: `12nome`, `cognome-nome`, `vero?` e `padre&figli`. Teniamo a ribadire che `volume` e `VOLUME` sono differenti. Oltre a rispettare le regole precedentemente enunciate, un identificatore non può essere una parola chiave del linguaggio (vedi Appendice B per l'elenco delle parole chiave), né può essere uguale a un nome di funzione.

Allo scopo di rendere più chiaro il risultato dell'esempio precedente, si possono visualizzare i valori delle variabili `base` e `altezza`:

```
printf("%d ", base);
printf("%d ", altezza);
printf("%d", area);
```

Nelle prime due istruzioni `printf` si è inserito all'interno dei doppi apici, di seguito all'indicazione del formato di stampa `%d`, uno spazio, in modo che venga riportato in fase di visualizzazione dopo il valore della base e dell'altezza, così da ottenere:

```
3 7 21
```

e non `3721`. Se si vuole far precedere la visualizzazione dei valori da un testo di descrizione, è sufficiente inserirlo prima del simbolo di percentuale:

```
printf("Base: %d ", base);
printf("Altezza: %d ", altezza);
```

```
printf("Area: %d", area);
```

Quello che viene prodotto in esecuzione è

```
Base: 3 Altezza: 7 Area: 21
```

Per fare in modo che a ogni visualizzazione corrisponda un salto riga si deve inserire `\n` prima della chiusura dei doppi apici:

```
printf("Base: %d\n", base);
printf("Altezza: %d\n", altezza);
printf("Area: %d\n", area);
```

In questo caso in esecuzione si otterrebbe

```
Base: 3
Altezza: 7
Area: 21
```

Mentre `int` è una parola chiave del C e fa parte integrante del linguaggio, `base`, `altezza` e `area` sono identificatori di variabili scelti a nostra discrezione. Lo stesso effetto avremmo ottenuto utilizzando al loro posto altri nomi generici, quali `x`, `y` e `z`.

La forma grafica data al programma è del tutto opzionale; una volta rispettata la sequenzialità e la sintassi, la scrittura del codice è libera. In particolare, più istruzioni possono essere scritte sulla stessa linea, come nell'esempio seguente:

```
#include <stdio.h>
main() {int x,y,z; x = 3; y = 7;
z= x*y; printf("Base: %d\n", x); printf("Altezza: %d\n", y);
printf("Area: %d\n", z);}
```

Questo programma, però, è notevolmente meno leggibile del precedente.

✓ **NOTA**

Lo stile facilita il riconoscimento delle varie unità di programma e riduce il tempo per modificare, ampliare e correggere gli errori. Se ciò è vero in generale, lo è particolarmente per questo linguaggio poiché, come si avrà modo di vedere, il C spinge il programmatore alla sintesi, all'utilizzo di costrutti estremamente asciutti, essenziali. Non importa quale stile si decida di utilizzare, importante è seguirlo con coerenza.

In generale è bene dare alle variabili nomi significativi, in modo che si possa facilmente ricostruire l'uso che si è fatto di una certa variabile, qualora si debba intervenire a distanza di tempo sullo stesso programma.