1.4 Input e output

Perché il programma per il calcolo dell'area del rettangolo sia più generale ed effettivamente "utile", l'utente deve poter immettere i valori della base e dell'altezza, mediante l'istruzione di input:

```
scanf("%d", &base);
```

L'esecuzione di questa istruzione fa sì che il sistema attenda in input un dato da parte dell'utente. Analogamente a quello che accadeva in `printf`, `%d` indica un valore intero in formato decimale che verrà assegnato alla variabile `base`. Si presti attenzione al fatto che in una istruzione `scanf` il simbolo `&` (*ampersand*) deve precedere immediatamente il nome della variabile; `&base` sta a indicare l'indirizzo di memoria in cui si trova la variabile `base`. L'istruzione `scanf("%d", &base);` può allora essere così interpretata: "leggi un dato intero e collocalo nella posizione di memoria il cui indirizzo è `&base`".

Durante l'esecuzione di un programma può essere richiesta all'utente l'immissione di più informazioni, perciò è opportuno visualizzare delle frasi esplicative; a tale scopo facciamo precedere le istruzioni `scanf` da appropriate visualizzazioni in output sul video, tramite istruzioni `printf`:

```
printf("Valore della base: ");
scanf("%d", &base);
```

L'argomento di `printf` è semplicemente una costante, quindi deve essere racchiuso tra doppi apici. Quello che apparirà all'utente in fase di esecuzione del programma sarà:

```
Valore della base:
```

In questo istante l'istruzione `scanf` attende l'immissione di un valore; se l'utente digita 15 seguito da Invio:

```
Valore della base: 15<Invio>
```

questo dato verrà assegnato alla variabile `base`. Analogamente, possiamo modificare il programma per l'immissione dell'altezza e magari aggiungere un'intestazione che spieghi all'utente cosa fa il programma, come nel Listato 1.5.

```
/* Calcolo area rettangolo */

#include <stdio.h>

int base, altezza, area;

main()
{
  printf("AREA RETTANGOLO\n\n");

  printf("Valore base: ");
  scanf("%d", &base);
  printf("Valore altezza: ");
  scanf("%d", &altezza);

  area = base*altezza;

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

Listato 1.5 Immissione di valori

Vediamo il risultato dell'esecuzione del programma nell'ipotesi che l'utente inserisca i valori 10 e 13:

```
AREA RETTANGOLO

Valore base: 10
Valore altezza: 13
Base: 10
Altezza: 13
Area: 130
```

Per lasciare una linea vuota si deve inserire un ulteriore `\n` nell'istruzione `printf` all'interno di doppi apici: `printf("AREA RETTANGOLO\n\n")`. Il primo `\n` fa andare il cursore a linea nuova dopo la visualizzazione di

AREA RETTANGOLO, il secondo lo fa scorrere di un ulteriore linea. Il ragionamento è valido in generale: se si desidera saltare un'altra riga basta aggiungere un \n e se si vuole lasciare una linea prima della visualizzazione si fa precedere \n ad AREA RETTANGOLO:

```
printf("\nAREA RETTANGOLO\n\n\n");
```

Inoltre è possibile inserire il salto in qualsiasi posizione all'interno dei doppi apici, come nel seguente esempio:

```
printf("AREA \nRET\nTAN\nGOLO");
```

che provoca in fase di esecuzione la visualizzazione:

```
AREA
RET
TAN
GOLO
```

Si possono stampare più variabili con una sola printf, indicando prima tra doppi apici i formati in cui si desiderano le visualizzazioni e successivamente i nomi delle rispettive variabili. L'istruzione

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

inserita alla fine del programma precedente stamperebbe, se i dati immessi dall'utente fossero ancora 10 e 13:

```
10 13 130
```

Nell'istruzione il primo %d specifica il formato della variabile base, il secondo %d quello di altezza e il terzo quello di area. Per raffinare ulteriormente l'uscita di printf, si possono naturalmente inserire degli a-capo a piacere:

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

che hanno come effetto

```
10
13
130
```

Se, per esempio, si desidera una linea vuota tra il valore della variabile base e quello di altezza e due linee vuote prima del valore della variabile area, è sufficiente inserire i \n nella descrizione dei formati, esattamente dove si vuole che avvenga il salto a riga nuova:

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

All'interno dei doppi apici si possono scrivere i commenti che devono essere stampati. Per esempio, se la visualizzazione della terza variabile deve essere preceduta da Area:, l'istruzione diventa la seguente:

```
printf("%d\n%d\nArea: %d", base, altezza, area);
```

che darà in uscita

```
10
13
Area: 130
```

Analogamente si può procedere con le altre variabili:

```
printf("Base: %d\nAltezza: %d\nArea: %d", base, altezza, area);
```

Si tratta dunque di inserire i vari messaggi che devono apparire sul video tra doppi apici, prima o dopo i simboli che descrivono i formati degli oggetti da visualizzare.

Così come \n effettua un salto a linea nuova, la sequenza \t provoca l'avanzamento del cursore di uno spazio di tabulazione:

```
printf("Base: %d\tAltezza: %d\tArea: %d", base, altezza, area);
```

produce come uscita

```
Base: 10          Altezza: 13       Area: 130
```

Esistono altre sequenze di caratteri con funzioni speciali, dette *sequenze di escape*. Riassumiamo quelle più usate, invitando il lettore a provarle nelle `printf`.

`\n`	va a linea nuova
`\t`	salta di una tabulazione
`\b`	ritorna un carattere indietro (*backspace*)
`\a`	suona il campanello della macchina
`\\`	stampa il carattere `\`
`\"`	stampa il carattere `"`

Le ultime due sequenze meritano un commento. Normalmente i doppi apici chiudono la descrizione del formato di una `printf`, perciò se si desidera visualizzare il carattere `"` lo si deve far precedere da `\`; una considerazione analoga vale per lo stesso carattere `\`.

È possibile inserire nella `printf`, al posto delle variabili, delle espressioni, di tipo specificato dal formato:

```
printf("Area: %d", 10*13);
```

Il `%d` ci indica che il risultato dell'espressione è di tipo intero; l'istruzione stamperà 130. Un'espressione può naturalmente contenere delle variabili:

```
printf("Area: %d", base*altezza);
```

Si può definire all'interno di una istruzione `printf` anche il numero di caratteri riservati per la visualizzazione di un valore, nel seguente modo:

```
printf("%5d%5d%5d", base, altezza, area);
```

Il `%5d` indica che verrà riservato un campo di cinque caratteri per la visualizzazione del corrispondente valore, che sarà sistemato a cominciare dall'estrema destra di ogni campo:

Se vengono inseriti degli spazi o altri caratteri nel formato, oltre alle descrizioni `%5d`, essi appariranno nelle posizioni corrispondenti. Inserendo poi un carattere `-` dopo il simbolo di percentuale e prima della lunghezza del campo il valore viene sistemato a cominciare dall'estrema sinistra della maschera. L'istruzione

```
printf("%-5d%-5d%5d", base, altezza, area);
```

visualizza dunque