# 2 Operatori matematici e costrutto `if`

Questa dispensa propone esercizi sulla scrittura di algoritmi, in linguaggio C, utili alla comprensione delle operazioni tra numeri e del costrutto condizionale `if`. Si introducono anche le due funzioni principali della libreria `stdio.h`: `scanf` e `printf`.

Si assume una conoscenza di base sul linguaggio C, tale da permettere al lettore di comprendere il significato del seguente frammento di codice.

```
#include <stdio.h>

void main()
{
    /* corpo del programma */

    getchar();
}
```

Ai fini del corso, è ininfluente la scelta di dichiarare il main come

```
  int main(int argc, char *argv[]) {
    ...

    getchar();
  }
```

oppure come

```
  void main() {
    ...

    getchar();
  }
```

Tuttavia, si richiede che il lettore comprenda la differenza tra le due alternative.

L'istruzione `getchar()` non fa parte della soluzione. Si tratta di un'istruzione bloccante per mettere l'elaboratore in attesa di un carattere da tastiera. Senza questa istruzione, o istruzioni equivalenti (e.g., la `system("PAUSE")`), l'esecuzione del programma termina immediatamente senza permettere all'utente di visualizzare l'output a video.

## 2.1 Operazioni matematiche

Si assume che il lettore familiarizzi con i tipi di dato numerici previsiti dal C (e.g., `int`, `float`) e con i rispettivi specificatori di formato (e.g. `"%d"`, `"%d"`).

Inoltre, le operazioni matematiche essenziali necessarie alla comprensione degli esercizi proposti in questa sezione sono:

```
#include <stdio.h>

void main()
{
    printf("Addizione: 1+2 = %d\n", 1+2);

    printf("Moltiplicazione: 1*2 = %d\n", 1*2);

    printf("Sottrazione: 1-2 = %d\n", 1-2);

    printf("Divisione: 8/3 = %d (%f)\n", 8/3, 8.0/3.0);

    printf("Resto della divisione intera: 8/3 = %d", 8 % 3);

    getchar();
}
```

Prima di procedere oltre, il lettore deve aver compreso il significato di questo frammento di codice (e.g., provando a compilarlo e ad eseguirlo).

### 2.1.1 Esercizi

**Esercizio 2.1**

Scrivere un programma che esegua la differenza di due numeri inseriti da tastiera.

**Esercizio 2.2**

Scrivere un programma che riceve in ingresso un prezzo ed uno sconto da applicare, e restituisce il prezzo scontato e il risparmio ottenuto.

**Esercizio 2.3**

Scrivere un programma che prende in ingresso un tempo espresso in secondi e ne restituisce l'equivalente nel formato ore, minuti, secondi.

**Esercizio 2.4**

Scrivere un programma che prende in ingresso un prezzo in euro e restituisce il numero minimo di banconote utilizzando solo pezzi da 50, 20 e 5 euro. Indicare anche la moneta rimanente.

**Esercizio 2.5**

Scrivere un programma che prenda in ingresso un carattere (e.g., `a`, `b`, `c`) ed un numero intero positivo $n$. Il programma dovrà stampare a video il carattere inserito, formattato come carattere e come intero. Poi dovrà stampare a video il carattere $n$ posizioni successive a quello inserito. Ad esempio, se il carattere inserito è la lettera `a` ed $n = 3$, il programma dovrà stampare a video il carattere `d`.

## 2.2 Costrutto `if` e condizioni

Il costrutto `if` codifica un ramo condizionale. Il linguaggio C segue la seguente sintassi:

```
if (condizione)
  statement
[else statement]
```

ove le parentesi quadre indicano che la parte **`else`** `statement` è opzionale. Come per tutti gli altri costrutti in C, se uno `statement` è una sola istruzione terminata da punto e virgola, non serve altro. Se invece uno `statment` è composto da più istruzioni terminate da punto e virgola, sarà necessario racchiuderlo tra parentesi graffe.

```
if (condizione) {
  istruzione1;
  istruzione2;
  ...
}
```

La `condizione` è un espressione booleana, ovvero un'istruzione che, quando valutata, risulta sempre in un valore pari a zero (0, falso) o uno (1, vero). Per comporre espressioni booleane complesse si utilizzano i seguenti operatori:

**Operatori relazionali** valutano relazioni binarie tra i due operandi:

- `<` minore di
- `<=` minore di o uguale uguale a
- `>` maggiore di
- `>=` maggiore di o uguale a
- `==` uguale a
- `!=` non uguale a (diverso)

**Operatori booleani** valutano condizioni di verità tra i due operandi

- `&&` AND (congiunzione logica)
- `||` OR (disgiunzione logica)

**Attenzione:** si osservi che in C, l'operazione di assegnamento `a = 3` è diversa dall'operazione di confronto `a == 3`. La prima è sempre valutata come vera (1, uno), mentre la seconda, ovviamente, dipende dal valore memorizzato in `a`. Perciò:

```c
#include <stdio.h>

void main()
{
    int a;

    a = 3;

    if (a = 4) //assegnamento
        printf("Questo ramo viene sempre eseguito.\n");
    else
        printf("Questo ramo NON viene mai eseguito.\n");

    if (a == 4) //confronto
        printf("La variabile 'a' contiene il valore %d", a);
    else
        printf("Questo ramo NON viene mai eseguito.\n");

    getchar();
}
```

### 2.2.1 Esercizi

**Esercizio 2.6**

Scrivere un programma che calcoli la distanza tra due punti, $a$ e $b$, su un retta.

**Esercizio 2.7**

Scrivere un programma che calcoli la distanza tra due punti, $a$ e $b$, su un retta. Potete utilizzare la funzione `abs()` della libreria `math.h`, che calcola il valore assoluto di un numero intero.

```
printf("abs(1-2) = %d", abs(1-2));
//output: abs(1-2) = 1
```

**Esercizio 2.8**

Scrivere un programma che legga da tastiera un numero intero che rappresenta un anno (e.g., 2012) e che determini poi se tale anno è bisestile o meno. Si può assumere che il numero intero letto da tastiera sia sempre valido (e.g., di 4 cifre, positivo).

Un anno è bisestile se è multiplo di 4 ma non di 100, oppure se è multiplo di 400.