Esercizio 1 - Area del rettangolo

Completare il seguente programma C:

```c
int main (void) {
    const int b = 4;
    const int h = 7;
    int area;

    // ...
}
```

Il programma calcola l'area del rettangolo con base `b` e altezza `h`, e scrive il risultato nella variabile `area`.

Verificate il comportamento con il debugger, provando anche a cambiare i valori di `b` e `h`.

Esercizio 2 - Area del triangolo

Completare il seguente programma C:

```c
int main (void) {
    const int b = 1;
    const int h = 1;
    double area;

    // ...
}
```

Il programma calcola l'area del triangolo con base `b` e altezza `h`, e scrive il risultato nella variabile `area`.

Nota: fate attenzione alle precedenze degli operatori, e ai tipi presi in considerazione!

Esercizio 3 - Distanza tra due punti

Completare il seguente programma C:

```c
int main (void) {
    const double x1 = 0.5;
    const double y1 = 1.4;
    const double x2 = 0.7;
    const double y2 = 2.0;
    double square_dist;

    // ...
}
```

Il programma calcola il quadrato della distanza euclidea tra il punto `(x1, y1)` ed il punto `(x2, y2)`, scrivendo il risultato nella variabile `square_dist`. Si ricorda che:

$$D^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2$$

Attenzione: nel linguaggio C, non esiste un operatore per l'elevamento a potenza (l'operatore ^ esiste, ma fa un'altra cosa).

Esercizio 4 - Maiuscolo

Completare il seguente programma C:

```c
int main (void) {
    const char lower = 'a';
    char upper;

    // ...
}
```

Il programma scrive nella variabile `upper` la versione maiuscola del carattere minuscolo contenuto in `lower`.

Ricordate che il C non distingue tra "carattere ASCII" e "numero intero a 8 bit senza segno": la differenza sta solo nel modo in cui il programmatore interpreta i dati. Per il linguaggio C, il valore `'a'` è perfettamente identico a 97.

Esercizio 5 - Sistema lineare

Completare il seguente programma C:

```c
int main (void) {
    const double a1 = 0.5;
    const double b1 = 1.4;
    const double c1 = 0.7;
    const double a2 = 2.0;
    const double b2 = 3.7;
    const double c2 = 1.9;
    double x;
    double y;

    // ...
}
```

Il programma scrive nelle variabili x e y la soluzione del sistema lineare:

$$\begin{cases} a_1 x + b_1 y = c_1 \\ a_2 x + b_2 y = c_2 \end{cases}$$

Si consiglia di usare il metodo di Cramer, e di creare delle variabili per i risultati intermedi (i

determinanti). Si può supporre che il sistema sia sempre determinato.

Esercizio 6

Utilizzando l'ambiente di sviluppo, scrivere, compilare ed eseguire il seguente programma in linguaggio C, verificando che non ci siano errori né warning in fase di compilazione.

```
#include <stdio.h>
int main (void)
{
   int x;
   printf("Inserisci un numero: ");
   scanf("%d", &x);
   if (x>0)
      printf("Il valore %d e' positivo\n", x);
    else
      printf("Il valore %d e' negativo o pari a 0\n", x);
    return 0;
}
```

Dopo averlo eseguito, esercitarsi con l'esecuzione passo a passo osservando il valore della variabile x tramite 'watch', provare con diversi valori: 10, -10, 0, 9, - 15.

Esercizio 7 - calcolo modulo

Scrivere un programma in linguaggio C che dovrà:
a) Definire una variabile di tipo intero e di nome x e assegnarne un valore, positivo o negativo (fare diverse prove).
b) Stabilire utilizzando la struttura elementare if-then(-else) (vedere sotto) se tale variabile contiene un valore negativo e, in questo caso, trasformarlo nel corrispondente valore positivo riassegnandone il valore alla variabile x
c) Inviare in output il valore finale, ovvero il modulo del valore acquisito tramite la funzione printf nel seguente modo:

```
printf("Il modulo e': %d\n", x);
```

Approfondimento

Esercizio 8. Per chi ha già programmato

Scrivere un programma che stabilisca se un numero è primo; in particolare, la soluzione dovrà:
a) Acquisire da input un valore intero da tastiera
b) Utilizzare le strutture elementari while-do o do-while per stabilire tramite un ciclo se il numero è primo o meno
c) A seconda dei casi, inviare in output un opportuno messaggio.

Nota1) Sforzarsi di realizzare un algoritmo “ben strutturato”, senza salti: in questo caso l’utilizzo di una variabile logica (in gergo “flag”, bandierina) può aiutare.
Nota2) Adottare sempre il metodo dei raffinamenti successivi: ad esempio, nella stesura iniziale considero come divisori tutti i numeri che precedono il numero dato. Successivamente analizzo nel dettaglio il problema, per vedere se posso limitare il numero di cicli.

Esercizio 9

Scrivere un programma che definisca 3 variabili intere chiamate operand1, operand2 e result, e:
a) Acquisisca da tastiera il valore di operand1 e operand2 tramite la funzione scanf
b) Ne calcoli, **tramite l'utilizzo di una funzione**, la somma e la salvi nella variabile result
c) Visualizzi a video il valore della variabile result utilizzando la funzione printf:

```
printf("La somma e': %d", result);
```

Consiglio: prima di dare il run, eseguire sempre il programma passo-passo, controllando le variabili con il watch. In particolare, verificare sempre che i valori introdotti da tastiera e i valori letti nelle variabili coincidano, e così per i valori in output (test di correttezza per gli specificatori di formato e altro…). Inoltre verificare se ci sono casi in cui l’operazione di somma sia errata (overflow).

Esercizio 10.

Scrivere un programma C in grado di risolvere un’equazione di primo grado espressa nella forma ax+b=0; in particolare il programma dovrà:
a) Definire due variabili intere (int), a e b per memorizzare i coefficienti dell’equazione
b) Definire una variabile intera chiamata x in cui memorizzare il risultato dell’equazione
c) Acquisire da tastiera (o assegnare nel programma) il valore dei coefficienti a e b
d) Calcolare il valore di x e visualizzarlo a video

Approfondimento: cosa succede nel caso in cui il valore di a è uguale a 0?