HOMEWORKS

Esercizio 1

Si modifichi il classico programma *Hello World*:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
   cout << "Hello World !" << endl;

   system("PAUSE");
   return EXIT_SUCCESS;
}
```

in modo che il programma stampi N ripetizioni della stringa “HelloWorld” (su righe diverse), con N inserito dall’utente.

Suggerimenti:
• Usare cin >> per acquisire N.
• Usare un ciclo WHILE con una variabile contatore per gestire le ripetizioni.
• Attenzione a usare cout << con **“\n”** oppure **endl** per il ritorno a capo.

Esercizio 2

Calcolare l’indice di massa corporea (BMI) a partire da altezza (h) e peso (m) immessi dall’utente, specificando il tipo di situazione (sottopeso, normale, ecc…). Formula: BMI = m / h^2 (m in kg, h in metri).
Si faccia riferimento alla seguente tabella:

Classe	Sottopeso	Normale	Sovrappeso	Obeso	Obeso grave
BMI	< 18.5	18.5 - 25	25 - 30	30 - 40	> 40

Suggerimenti:
• Chiedere all’utente altezza e peso con cout << + cin >> con variabili **float**
• Usare una sequenza di IF per determinare la classe in base al BMI
• Fare attenzione a gestire un input errato da parte dell’utente
(altezza e/o peso uguali a zero o negativi)

Esercizio 3

Si chieda all’utente di inserire due numeri interi (a e b), tali che:
a > b > 0. Se ne calcoli la “strana somma” secondo la formula:

strana somma = a + (b) + (b – 1) + (b – 2) + … + 0

Si visualizzi a schermo il risultato dell’operazione.

Suggerimenti:
• Chiedere all’utente di inserire i due numeri usando cout <<, cin >>
• Fare attenzione a gestire l’input (vincoli sui numeri **a** e **b**)

- Usare un ciclo WHILE per calcolare la “strana somma”, decrementando opportunamente **b**

Esercizio 4

Calcolare minimo, massimo e media di una sequenza di numeri in virgola mobile (**float**) immessi dall’utente. L’utente immette un numero alla volta; 0 termina l’immissione e fa si che il programma stampi i risultati (ignorando lo 0 finale).

Suggerimenti:

- Dichiarare e inizializzare alcune *variabili di stato* per mantenere minimo, massimo, somma e numero di valori finora immessi dall’utente
- Utilizzare un ciclo WHILE per leggere i numeri e aggiornare le variabili di stato
- Fare attenzione a gestire correttamente ingresso e uscita dal ciclo!
- Fare attenzione a gestire il caso particolare con sequenza di numeri vuota

Esercizio 5

Scrivere un programma che proponga all'utente di inserire un intervallo di temperature espresse in gradi Celsius. Se dopo l'inserimento di entrambi gli estremi una temperatura è inferiore a -273,15°C, il programma termina. Successivamente, stampare a terminale la conversione dell'intervallo in gradi Fahrenheit.
Nota: Fahrenheit = 32+9/5 * Celsius.

Esercizio 6

Scrivete un programma che, dopo aver richiesto in input un numero intero positivo *n*, stampi la corrispondente tabellina, moltiplicando *n*
per i numeri interi da 1 a 10, come indicato nel seguente esempio di esecuzione.

```
1 x 9 = 9
2 x 9 = 18
3 x 9 = 27
4 x 9 = 36
5 x 9 = 45
6 x 9 = 54
7 x 9 = 63
8 x 9 = 72
9 x 9 = 81
10 x 9 = 90
```

Esercizio 7

Disegnare sullo schermo un rettangolo, utilizzando un carattere a piacere (ad es. '#'), dopo averne chiesto all'utente altezza (h) e lunghezza (l).
Ad esempio, per h = 4 e l = 6, ci si aspetta il seguente output:

```
******
******
******
******
```

Suggerimenti:

- Chiedere all'utente altezza e lunghezza
- Fare attenzione a gestire l'input (altezza e/o lunghezza negativi)
- Per il disegno, usare due cicli WHILE annidati, uno esterno che scandisce le righe, uno interno che scandisce le colonne (o viceversa)

Esercizio 8

Estendere l'esercizio precedente in modo da supportare i seguenti tipi di figura (l'utente seleziona il tipo ed immette i parametri della figura all'avvio del programma)

Quadrato (parametro: lato)

```
***
***
***
```

Rettangolo (parametri: base, altezza)

```
****
****
****
```

Triangolo Rettangolo (parametri: base, altezza)

```
********
******
****
**
```

Suggerimenti: Attenzione al triangolo!
Usare una variabile di tipo **float** per mantenere la lunghezza della riga corrente e decrementarla di **(float)(base/altezza)** a ogni riga.

Esercizio 9

Realizzare un programma che legga tre valori interi (n1, n2, n3) compresi tra 1 e 100, estremi inclusi, e poi presenti a video il seguente menù di operazioni possibili:
A - somma tra n1, n2 e n3
B - prodotto tra n1 e n2
C - sottrazione tra n3 e n1
D - divisione tra n1 e n2 (risultato double).
X - uscita dal programma

Legge poi un carattere da tastiera: se il carattere è tra quelli indicati nel menù, si deve eseguire l'operazione richiesta, stampare i numeri utilizzati nell'operazione e il risultato e poi ripresentare il menù altrimenti il carattere deve essere ignorato e si deve ripresentare solo il menù.
Nel caso il carattere sia X, il programma termina.

Nota: nel caso D è necessaria la conversione di almeno uno dei due operandi.

Esercizio 10

Date le coordinate x e y di 3 punti (P1, P2, P3), verificare se i punti P2 e P3 sono equidistanti dal punto P1 (quindi se i punti P2 e P3 si trovano su una circonferenza con centro nel punto P1). La verifica viene fatta controllando se i quadrati delle distanze P1-P2 e P1-P3 sono uguali.

Input: coordinate x ed y dei 3 punti
Output: messaggio che indica se i punti sono equidistanti oppure no

Oss: i tipi di dato *float* o *double* sono approssimazioni finite dei numeri reali;
per confrontare due valori float o (double) è consigliabile impostare una verifica di appartenenza a un intervallo, invece che un semplice test di uguaglianza.
Es:
(distq12-distq13<TOL && distq12-distq13>-TOL)
dove TOL sarà una costante definita all'inizio del file come, es.: #define TOL 0.0001

Esercizio 11

Data una base ed un esponente, calcolare "base elevato ad esponente". Sia la base che l'esponente sono valori interi. Utilizzare la struttura di controllo "for"

Input: base, esponente
Output: base elevato ad esponente

Esercizio 12

Scrivere un programma che calcoli il massimo comun divisore (MCD) tra due numeri n1≥0 e n2≥0, usando l'algoritmo di Euclide.

Il principio di funzionamento dell'algoritmo è il seguente:
MCD(n1, 0) = n1, MCD(0, n2) = n2;
MCD(n1, n2) = n1, se n1 = n2;
MCD(n1, n2) = MCD(n1-n2, n2) se n1>n2;
MCD(n1, n2) = MCD(n1, n2-n1) se n2>n1

Input: numeri di cui calcolare il MCD
Output: MCD dei numeri.

Esercizio 13

Data una sequenza di interi (anche negativi), terminata dal valore 0, calcolare il massimo della sequenza.

Input: sequenza di interi (anche negativi)
Output: massimo della sequenza

Esercizio 14

Scrivere un programma che dato un carattere in ingresso, lo trasforma in un altro carattere, che si trova OFFSET posizioni più avanti nell'alfabeto.
(per esempio con OFFSET = 4, il carattere 'a' diventa 'e' oppure il carattere 'z' diventa 'd').

L'alfabeto considerato è:

ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz

Per esempio, con OFFSET = 4, la lettera 'X' diventa 'b'.

La complessità del programma sta nel fatto che nella codifica ASCII le sequenze 'A'..'Z' e 'a'..'z' (viene prima la sequenza di caratteri maiuscoli) non sono consecutive, ma tra loro c'è un altro insieme di caratteri, per cui occorre spezzare il programma in 2 "if".

Input: un carattere da convertire
Output: il carattere convertito

Si riporta porzione del codice ASCII:

032 0x20	056 0x38 8	080 0x50 P	104 0x68 h
033 0x21 !	057 0x39 9	081 0x51 Q	105 0x69 i
034 0x22 "	058 0x3a :	082 0x52 R	106 0x6a j
035 0x23 #	059 0x3b ;	083 0x53 S	107 0x6b k
036 0x24 $	060 0x3c <	084 0x54 T	108 0x6c l
037 0x25 %	061 0x3d =	085 0x55 U	109 0x6d m
038 0x26 &	062 0x3e >	086 0x56 V	110 0x6e n
039 0x27 '	063 0x3f ?	087 0x57 W	111 0x6f o
040 0x28 (	064 0x40 @	088 0x58 X	112 0x70 p
041 0x29)	065 0x41 A	089 0x59 Y	113 0x71 q
042 0x2a *	066 0x42 B	090 0x5a Z	114 0x72 r
043 0x2b +	067 0x43 C	091 0x5b [	115 0x73 s
044 0x2c ,	068 0x44 D	092 0x5c \	116 0x74 t
045 0x2d -	069 0x45 E	093 0x5d]	117 0x75 u
046 0x2e .	070 0x46 F	094 0x5e ^	118 0x76 v
047 0x2f /	071 0x47 G	095 0x5f _	119 0x77 w
048 0x30 0	072 0x48 H	096 0x60 `	120 0x78 x
049 0x31 1	073 0x49 I	097 0x61 a	121 0x79 y
050 0x32 2	074 0x4a J	098 0x62 b	122 0x7a z
051 0x33 3	075 0x4b K	099 0x63 c	123 0x7b {
052 0x34 4	076 0x4c L	100 0x64 d	124 0x7c |
053 0x35 5	077 0x4d M	101 0x65 e	125 0x7d }
054 0x36 6	078 0x4e N	102 0x66 f	126 0x7e ~
055 0x37 7	079 0x4f O	103 0x67 g	127 0x7f □

Esercizio 15
Scrivere un programma che dati 2 “insiemi” di numeri interi, calcoli l'intersezione di questi due insiemi. Gli insiemi sono memorizzati negli array 'ins1' (di lunghezza N) e 'ins2' (di lunghezza M), mentre l'insieme intersezione è contenuto nell'array 'ins_int'.

Esercizio 16
Scrivere un programma che prenda in input un numero in base 10, e lo converta in una base scelta dall'utente (la nuova base deve essere compresa tra 2 e 9).

Esercizio 17 (Privo di Quadrati)
Un numero intero positivo è privo di quadrati se non esiste un quadrato perfetto che lo divide (equivalentemente, se non esiste un quadrato di primo che lo divide). Scrivete un programma che legga un numero in input e determini se è privo di quadrati.

Esercizio 18 (Numeri perfetti)
Un numero è perfetto se è uguale alla somma dei suoi divisori propri.
Per esempio, 6 = 1+2+3 è perfetto.
Scrivete un programma che, dato un intero in input, scriva i numeri perfetti minori dell’intero dato.

Esercizio 19 (Primi Gemelli)
Due primi p e q sono gemelli se $p = q+2$.
Scrivete un programma che stampi i primi gemelli minori di un intero fornito dall'utente.