



## EJERCICIOS RESUELTOS EN CLASE DE TEORÍA

1. **[Factorial.cpp]** Realizar un programa que tomando un número entero por teclado, devuelva su factorial. Se deberá crear una función factorial(x) la cual se usará desde el programa principal.
2. **[Potencia.cpp]** Escribir un programa que contenga la función potencia(x, y) implementada por nosotros y que calcule el valor de x elevado a y mediante multiplicaciones, sabiendo que y siempre es un valor entero. No se puede utilizar la función pow.
3. **[Numeroe.cpp]** El valor de  $e^x$  se puede aproximar por la suma:

$$e^x = \sum_{i=0}^n \frac{x^i}{i!}$$

Escribe un programa que le pida al usuario el valor de  $x$  y el valor de  $n$  y muestre por pantalla el valor de la aproximación de  $e^x$  para el  $x$  y el  $n$  introducidos. La función principal deberá llamar a una función **aproximación (x, n)** que devuelve el valor buscado. Compara el resultado de la aproximación calculada con el resultado obtenido usando la función  $\exp(x)$  de la librería matemática y muestra la diferencia de los tipos de cálculo por pantalla

4. **[Euclides.cpp]** El máximo común divisor (mcd) de dos números P y Q es el mayor entero D que divide a ambos. Un algoritmo muy conocido para calcularlo es el de Euclides. Éste utiliza dos variables, que contienen inicialmente a cada uno de los números, y trata de hacer que su contenido sea el mismo. Para ello, irá restando la menor a la mayor hasta que ambas contengan el mismo valor. En dicho momento, el valor obtenido en cualquiera de ellas es el máximo común divisor de los dos números iniciales.

Por ejemplo, si  $P = 18$  y  $Q = 12$ , el algoritmo hará que P y Q vayan tomando los siguientes valores:

Inicialmente  $P = 18$  y  $Q = 12$  ( $P > Q \Rightarrow P := P - Q$ )

Después  $P = 6$  y  $Q = 12$  ( $Q > P \Rightarrow Q := Q - P$ )

Después  $P = 6$  y  $Q = 6$  ( $P = Q \Rightarrow$  El mcd es 6)

5. **[MCM.cpp]** Realizar un programa que calcule el mínimo común múltiplo de dos números introducidos por teclado. Recuerda que:

$$MCM(a, b) = \frac{a * b}{mcd(a, b)}$$

Usa la función m.c.d implementada antes.

6. **(P2) [Combinatorios.cpp]** Escribir un programa que calcule el combinatorio de un número. Divide el programa en funciones (usa una función que calcule el factorial de un número dado).

$$C_m^n = C_{m,n} = C(m, n) = \binom{m}{n} = \frac{m!}{n!(m-n)!}$$

## EJERCICIOS PARA RESOLVER

7. **[PotenciaV2.cpp]** Escribir un programa que contenga la función **potencia(x, y)** implementada por nosotros y que calcule el valor de x elevado a y mediante multiplicaciones, siendo x e y números enteros. Hacer uso de ella en el programa principal para mostrar tanto  $x^y$  como  $y^x$ . No se puede utilizar la función pow.
8. **[EsPrimo.cpp]** Escribir una función que devuelva si un número que se le pasa como parámetro es primo o no. Úsala para el programa que, dado un número introducido por teclado, nos devuelva todos los números primos inferiores a él.



9. **[Division.cpp]** Escribe una función que, dados dos enteros positivos  $x$  e  $y$ , calcule la división entera y el resto de la división utilizando únicamente restas.

10. **[Racionales.cpp]** Diseñar un programa que pida por pantalla los cuatro valores que componen el numerador y denominador de dos números racionales y construir un subprograma que reciba esos valores y devuelva la suma en forma de número en coma flotante.

A continuación haz un segundo subprograma que realice el mismo cálculo que el del otro subprograma pero “devuelva” el numerador y el denominador del resultado.

11. **[VolumenEsfera.cpp]** Diseñar un programa que pida el valor del radio de una esfera desde el programa principal. Éste valor debe ser pasado a una función **volumen ( r )** para que calcule su volumen. El mensaje que muestra el valor del volumen por pantalla se hará desde el programa principal.

12. **(Op1) [AreasVolumenes.cpp]** Amplia el programa del ejercicio “VolumenEsferas.cpp” de manera que calcule las áreas y volúmenes de varias figuras geométricas. El programa deberá mostrar en pantalla el siguiente menú:

- 1- Área y volumen de una esfera
- 2- Área y volumen de un cubo
- 3- Área y volumen de un cilindro
- 4- Salir

y deberá repetirse hasta que se elija la opción 4. Al pulsar cada una de las opciones se deberá calcular tanto el área como el volumen en diferentes funciones y llamarlas desde la función principal que es la que contiene el menú.

13. **(P4) [NormalizaVector.cpp]** Escribir un programa que, dado un vector por sus componentes ( $x_1$ ,  $y_1$ ) normalice su valor. Se deberá crear una función que devolverá el valor de las dos componentes del vector normalizadas. A esta función se la llamará desde la función principal.

14. **[Matematico.cpp]** Escribir un pequeño programa matemático que pueda realizar los siguientes cálculos:

- 1- Cálculo del perímetro de un círculo
- 2- Cálculo del área de un círculo
- 3- Cálculo del volumen de un cilindro.
- 4- Cálculo del volumen de una esfera
- 5- Cálculo del volumen de un cono

Para ello el programa presentará por pantalla un menú donde se podrá elegir una de estas opciones o se deberá elegir la opción de salir del programa. Hacer primero el diseño descendente de este problema.

15. **(P3) [Distancia.cpp]** Escribir un programa que calcule la distancia de dos puntos del plano, dados por sus coordenadas  $(x_1, y_1)$  y  $(x_2, y_2)$ . Se deberá crear una función **distancia (x1, y1, x2, y2)** que devolverá el valor de la distancia. A esta función se la llamará desde la función principal. Usa una función para leer las coordenadas de un punto y otra que calcula la distancia entre dos puntos.

16. **[Gravedad.cpp]** Escribir un programa que calcule la fuerza de atracción gravitatoria entre dos masas  $m_1$  y  $m_2$  situadas en el plano, con coordenadas  $(x_1, y_1)$  y  $(x_2, y_2)$ . Se deberá crear una función **fgrav (m1, x1, y1, m2, x2, y2)** que devolverá el valor de la fuerza. A esta función se la llamará desde la función principal.

La función **fgrav** deberá llamar a la función **distancia (x1, y1, x2, y2)** del ejercicio “Distancia.cpp”



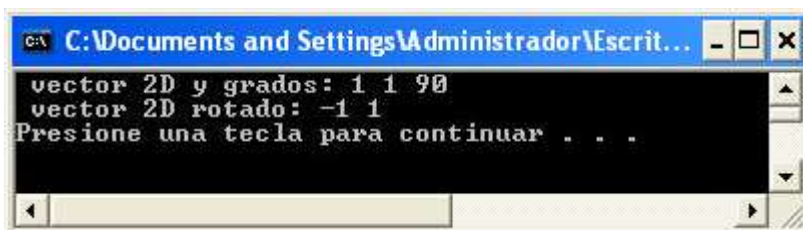


24. [Encriptar.cpp] Escribe una función capaz de *encriptar* un código de 4 caracteres. Para ello recibirá, además de los 4 caracteres, un número entero que deberá ser sumado al código ASCII de cada carácter para encriptarlo. Escribe también la función complementaria (*desencriptar*) y comprueba el comportamiento de ambas (ver figura).



```
C:\Documents and Settings\Administrador\Escri...
Dame codigo de 4 letras y clave: abcd 3
encriptado: defg
desencriptado: abcd
Presione una tecla para continuar . . .
```

25. Escribe una función que permita al usuario manejar vectores 2D (situados en el origen de coordenadas). Las funciones a implementar son:
- Convertir un vector 2D de coordenadas polares (módulo, ángulo) a cartesianas (x, y) y viceversa.
  - Suma (int x1, int y1, int x2, int y2, int &xres, int &yres) → suma dos vectores y devuelve el resultado sobre <xres, yres>.
  - Resta: resta dos vectores.
  - Producto escalar  $\leftarrow x1 \cdot x2 + y1 \cdot y2$ .
  - Girar (int x, int y, int angulo, int &xres, &yres) → devuelve sobre <xres, yres> el vector <x,y> girado <angulo> grados en sentido antihorario (ver figura).



```
C:\Documents and Settings\Administrador\Escri...
vector 2D y grados: 1 1 90
vector 2D rotado: -1 1
Presione una tecla para continuar . . .
```



26. (P1) Los siguientes programas son incorrectos o no dan el resultado esperado. Indica el por qué y escríbelos de la manera correcta.

```
-----
#include <iostream>
#include <stdlib.h>
using namespace std;

void es_par(void)
{
    if((valor_leido % 2) == 0)
        cout<< "El valor "<<valor_leido<<" es par\n";
    else
        cout<< "El valor "<<valor_leido<<" es impar\n";
}

int main(void)
{
    int valor_leido;

    cout << "Introduce un valor";
    cin >> valor_leido;
    es_par();
    system("PAUSE");
    return 0;
}
```

```
-----
#include <iostream>
#include <stdlib.h>
using namespace std;

void pide_caracter(char letra)
{
    do{
        cout <<"Introduce una letra minuscula\n";
        cin >> letra;
    } while((int(letra)<int('a')) || ((int(letra)> int('z')));
}

int main(void)
{
    char mi_letra = '?';
    pide_caracter(mi_letra);
    cout << "La letra leida es" <<mi_letra<< endl;
    system("PAUSE");
    return 0;
}
```



27. Esta práctica consiste en realizar un programa, correctamente modulado utilizando funciones, para emitir las facturas de los clientes de una **compañía de alquiler de automóviles sin conductor**.

Se leerá de teclado, nombre, edad, fecha de emisión del carnet de conducir, y número de tarjeta de crédito, Tipo de vehículo (A, B o C) y número de días de alquiler.

Se implementarán las siguientes funciones:

1. (\*) Comprobar Cliente

Si el cliente es menor de 25 años se le cobrará un suplemento de 30€

2. (\*) Comprobar Carnet

Se comprobará si el cliente posee más de cinco años de carnet, en caso contrario se mostrará un mensaje por pantalla informando de ello y se le cobrará un suplemento de 60€.

3. (\*) Calcular alquiler.

El precio del alquiler de un coche

- de tipo A es 40€ por día
- de tipo B es 30€ y
- de tipo C es 20€.

Si el vehículo es de tipo A se incluye un seguro a todo riesgo que se mostrará en la factura de 20€.

Si el vehículo es de tipo B se incluye una cobertura parcial (con una franquicia de 300€) por daños por choque y por robo que se mostrará en la factura y el alquiler se incrementa en 10€.

Independientemente del tipo de vehículo todos incluyen 1050 Km gratis.

4. Calcular descuentos

Si el número de días de alquiler es superior a una semana se descontará 50€.

Si el importe es superior a 200€ se le aplicará un descuento del 3%

5. Imprimir factura por pantalla mostrando los datos del cliente, del vehículo, el número de días de alquiler, los seguros que incluye y el importe final.

6. Cierre del día

Al final de cada sesión se mostrará el importe total facturado entre todos los vehículos que se han alquilado.