ICOM-2022 Práctica 3

1. Realizar un programa para imprimir la siguiente secuencia, siendo M y N valores enteros ingresados por el usuario:

```
00   01   02   ...   0N
10   11   12   ...   1N
...
M0   M1   M3   ...   MN
```

2. Realizar un programa que recorra todos los números naturales desde 1 hasta 1000 y calcule e imprima la suma de todos los múltiplos de 2, 3, 4, y 5 en distintos acumuladores. (Acum2 acumula la suma de todos los múltiplos de 2, Acum3 acumula la suma de todos los múltiplos de 3, etc.).

3. Pi. Realice un programa que aproxime Pi utilizando la siguiente serie numérica propuesta por Leibniz,

$$\frac{\pi}{4} = \sum_1^{\infty} f(n) = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \ldots$$

4. Realizar un programa que sume los primeros N (ingresado por el usuario) números naturales. El programa debe imprimir el resultado de tal suma. (no aplicar n * (n+1) / 2). Encapsule el cálculo de la suma en una función **`int suma(int N);`** (implementar esta función antes de la función **`main`**).

5. La sucesión:

   $X_0 = 1$

   …

   $X_{i+1} = (x_i + a/x_i)/2$

   Converge a la raíz cuadrada de ***a***, si ***a*** es un número real positivo. Escriba un programa que pida al usuario el valor de ***a***, e imprima su raíz cuadrada utilizando la sucesión anterior. (Compare con el resultado de la función **sqrt(a)**). ¿Cómo va a determinar la convergencia? ¿Cuántos pasos necesita el algoritmo para converger? ¿Cómo depende eso del número **a**? Encapsule el cálculo de la sucesión en una función **`double miSqrt(double a);`** (implementar esta función antes de la función **`main`**).

6. A partir del desarrollo en serie de Taylor de e^x y especializándolo en ***x=1*** calcular el valor de ***e***. Intentar minimizar la cantidad de operaciones a realizar en cada término. Encapsule el cálculo de la serie en una función **`double calculaE();`** (implementar esta función antes de la función **`main`**).

7. Diseñe e implemente un programa que solicite al usuario un número (en base 10) y una nueva base (en el intervalo [2,16]), e imprima la representación de ese número usando la base ingresada. Para coeficientes entre 10 y 15 utilice los caracteres entre 'A' y 'F'.

   Ejemplo:

284 en base 16:	11C
284 en base 4:	10130
284 en base 8:	434
284 en base 13:	18B

8. Suponga la serie:

$$S = \sum_{1}^{0xFFFFFF} X_1 + X_2 + X_3$$

donde X_1, X_2 y X_3 son constantes e iguales a:

$X_1 = 1.126$

$X_2 = -1.125$

$X_3 = -0.001$

¿Cuánto es el resultado esperado? Haga un programa para verificarlo. Realice 2 versiones, una en donde las variables son del tipo **`float`** y otra en donde son del tipo **`double`**. Justifique los resultados encontrados.

9. Se desea definir un UDT para representar/manipular colores en formato RGB (rojo-verde-azul), para ello, parte de la definición es algo como:

```
struct RGB {
    unsigned char red;
    unsigned char green;
    unsigned char blue;
    // ...
};
```

   a. Implemente la función interna (método) **`int compositeColor();`** que retorne el entero que represente el **`color`** dado según la representación del ejercicio 10 de la práctica 2.

   b. Implemente el método **`bool setFromComposite(int color);`** que dado el color compuesto **`color`** (que almacena los colores según ejercicio 10 de la práctica 2) imponga a los atributos **`red`**, **`green`** y **`blue`** los colores correspondientes. El método debe retornar **`true`** si se pudo imponer los valores correctamente o **`false`** si el **`color`** dado tiene una representación incorrecta (por ejemplo un valor no nulo en los bits 24-31).

   c. Implemente el método **`bool setFromComponents(int R, int G, int B);`** que imponga a los atributos **`red`**, **`green`** y **`blue`** los colores correspondientes. El método debe retornar **`true`** si se pudo imponer los valores correctamente o **`false`** si alguna componente tiene un valor incorrecto.

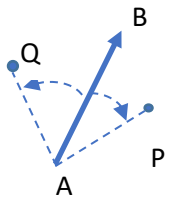
   d. Agregue un método **`void print();`** que imprima las componentes de la entidad.

   e. Implemente un programa que ejercite y pruebe los métodos anteriores.

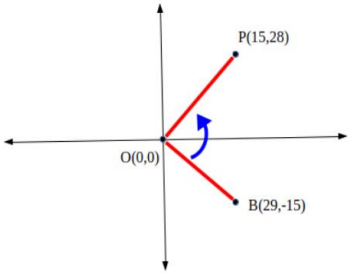
10. El código de control o número verificador es un mecanismo de detección de errores utilizado para verificar la corrección o integridad de un dato (por ejemplo el último dígito de un número de CUIT). Existe una gran cantidad de algoritmos, con distintos atributos, que calculan un número verificador. Crear un programa que lea un string e imprima su número verificador calculado de la siguiente manera: realizar la suma de las vocales multiplicadas por 4 más las consonantes multiplicadas por 2. Además las letras están pesadas según su posición, para esto hay que multiplicarlas por el valor de la posición (el primer carácter multiplica por 1, el segundo por 2, etc.). Suponga que las palabras solo pueden contener letras minúsculas. Por ejemplo, "balseiro" tiene que devolver 12228. Hint: utilizar for(char c : s) {…}

11. **Ejercicio progresivo[1] (corresponde al Parcial 1 2018, Problema 3, punto a) Segmentos y Polígonos en 2D.**

Dirección de un punto con respecto a un segmento orientado. Se dice que un punto P está a la derecha de un segmento orientado, si estando parado sobre el punto inicial de un segmento, y mirando en la dirección del punto final, debemos rotar la mirada en el sentido de las agujas del reloj para mirar a P. En forma análoga, un punto Q está a la izquierda de un segmento orientado, debemos rotar la mirada en el sentido contrario a las agujas del reloj para mirar a Q.



El producto cruz[2] posee una propiedad interesante que es utilizada para determinar la dirección de un punto respecto de un segmento orientado: *"El producto cruz entre 2 puntos es positivo (componente z hacia arriba) sí y sólo si el ángulo de esos puntos en el origen de coordenadas es en contra del movimiento de las agujas del reloj, y será negativo (componente z hacia abajo) si el ángulo de esos puntos en el origen es en el sentido de giro de las agujas del reloj, en el caso que sean colineales, es cero (componente z nula)"*.



Se solicita hacer una función que determine si un punto está o no a la derecha de un segmento ordenado. Prototipo (Respetarlo):

```
// retorna 1 si 'point' está a la derecha del segmento, -1 si está a
// la izquierda y 0 si son colineales.
int isOnTheRight( P2D_t point, P2D_t startSeg, P2D_t endSeg);
```

El UDT **`P2D_t`** está definido como:

```
struct P2D_t {
    double X;
    double Y;
};
```

[1] *Los ejercicios marcados de esta forma irán cobrando mayor complejidad con cada práctica, hasta llegar a tener la correspondiente a un ejercicio de parcial*

[2] https://es.wikipedia.org/wiki/Producto_vectorial