UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

CS1103
Programación Orientada a Objetos 2

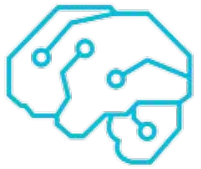
Unidad 1: Ejercicios

Carlos Arias
Marvin Abisrror
Rubén Rivas

Ejercicio #1

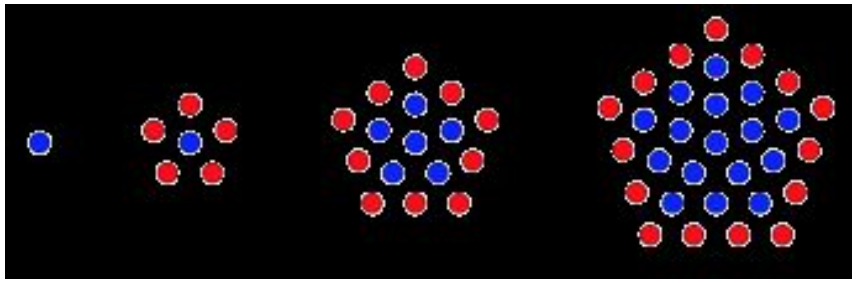
Crear una función que tome un vector y encontrar el entero que aparezca en una cantidad impar de veces:

```
buscar_impar({1, 1, 2, -2, 5, 2, 4, 4, -1, -2, 5}) --> -1
buscar_impar({20, 1, 1, 2, 2, 3, 3, 5, 5, 4, 20, 4, 5}) --> 5
buscar_impar({10}) --> 10
```



Ejercicio #2

Escribir una función que tome un número entero positivo y que calcule cuantos puntos existe en un pentágono alrededor del punto central en la n iteración.



```
calcular_pentagono(1) --> 1
calcular_pentagono(2) --> 6
calcular_pentagono(3) --> 16
calcular_pentagono(8) --> 141
```

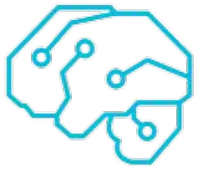
Ejercicio #3

Implementar 2 funciones de ordenamiento quick en su forma **recursiva** y en su forma **iterativa** utilizando arreglos dinámicos.

Para generar el arreglo leer un archivo de texto (ejemplo: **`in.txt`**) y el resultado generarlo en un archivo de texto (ejemplo: **`out.txt`**).

Basado en la versión recursiva generar una clase **`quicksort_t`** que sobrecargue el operador **`<<`** para recibir el archivo input y **`>>`** para decir el archivo output.

```
quicksort_t qs;
qs << "in.txt";
qs >> “out.txt”;
```



Ejercicio #4

Basado en la versión anterior implemente una versión genérica que usando template.

¿La versión usando templates soporta cualquier tipo?

¿Qué ocurre si el tipo es una estructura personalizada (class o struct)? ¿Qué limitaciones encuentra?

Ejercicio #5

Basado en el ejemplo de class (diapositiva 22) de la presentacion de teoria, crear una clase Integer e implementar las siguientes operaciones matemáticas adicionales:

- Suma acumulativa (+=)
- Resta (-) y resta acumulativa (-=)
- Multiplicacion (*) y resta acumulativa (*=)
- Division
- Potencia
- comparaciones lógicas (>, >=, <, <=, !=, ==)

Adicionalmente generar una clase Number que usando template acepte cualquier tipo numérico.

Ejercicio #6

Basado en **`std::vector`** crea una clase que se denominará **`utec::vector`**, esta versión inicial será limitada a que acepte solo enteros y deberá contar con los siguientes métodos:

- 3 constructores básicos (default, copia y operador asignación) y constructor con parámetro size del tipo size_t.
- **`destructor`**
- **`push_back`**
- **`pop_back`**
- **`insert`**, debido a que no se va esta version no va contar con iteradores, utilizar subindices.
- **`erase`**, similar condicion que insert.
- **`operador +`** que genere un nuevo vector que incluya los elementos del primero y el segundo vector.