Tarea 5

Ejercicio 1

Desarrolle una clase ordenamiento, la cual contenga métodos y atributos para ordenar un conjunto finito de elementos. En dicha clase se deberán usar plantillas para crear componentes genéricos, además, se deberá tener lo siguiente (como mínimo):

• Un método genérico para realizar el ordenamiento bubble sort.

• Un método genérico para realizar el ordenamiento merge sort.

• Un método genérico para para realizar el ordenamiento selection sort.

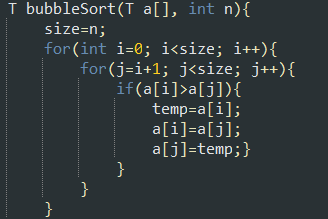
• Un método genérico para realizar el ordenamiento quicksort.

• Métodos para acceder y cambiar elementos dentro de la clase (setters y

getters)

Bubble Sort

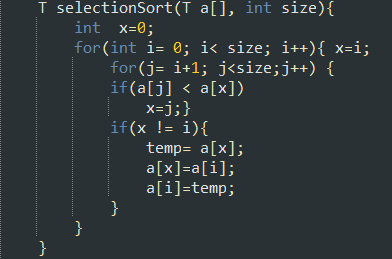
Dada una colección de elementos compara los dos primeros elementos de la colección y los intercambia en función de su orden en caso de ser necesario. El mismo proceso se repite hasta llegar al último elemento de la colección, como resultado se tiene que en la última posición de la colección estará el elemento con mayor valor.



Si el numero que se encuentra en la posición i es mayor al de la posición j, estos se intercambian, para ello utlizo una variable temporal que guarde el valor original de de i, mientras se intercambian los valores

Selection Sort

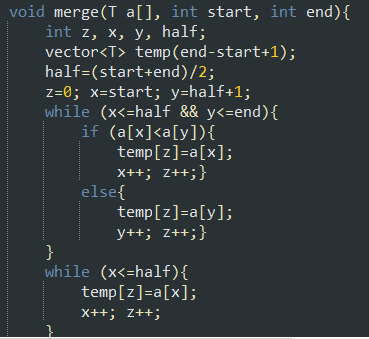
Consiste en encontrar el elemento menor del arreglo x, posteriormente, el segundo más pequeño nuevamente denominado x, para comparar el valor de cada arreglo según el índice más pequeño de cada iteración de i, así sucesivamente hasta el último elemento menos pequeño



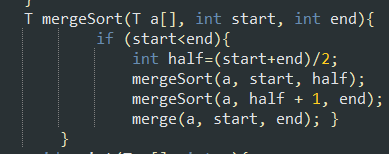
Si el numero que se encuentra en la posición j es menor al de la posición x, estos se intercambian, para ello utlizo una variable temporal que guarde el valor original de j, mientras se intercambian los valores

Merge sort

Establezco un método auxiliar que permita dividir el arreglo en 2, si se cuenta con dos listados ordenados, es posible mezclarlos de manera tal que el resultado de la mezcla es la concatenación ordenada de los 2 listados anteriores.



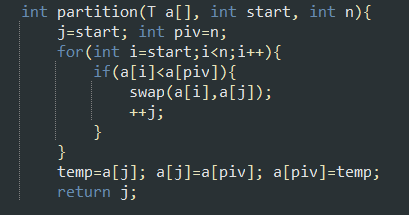
Aquí las se comparan los valores de las mitades, del inicio hasta la mitad y de la mitad hasta el inicio, en caso de que en cada iteración haya un elemento menor en una posición mas adelante del elemento con el que se compara, hay un swap entre ellos, de lo contrario permanecen igual.



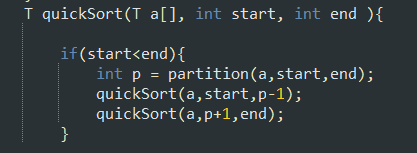
El método final de merge sort utiliza el método auxiliar que primero organiza y pega la primera mitad, posteriormente la segunda mitad y finalmente une ambas mitades del arreglo.

Quick Sort

Primero, tenemos que elegir un elemento de la lista de elementos a ordenar, al que se le llama pivote, posteriormente se organizan los demás elementos de la lista a cada lado del pivote, de manera que a un lado queden todos los menores que él, y al otro los mayores. En este momento, el pivote ocupa exactamente el lugar que le corresponde en el orden final. Repetimos este proceso para cada sublista mientras éstas contengan más de un elemento.



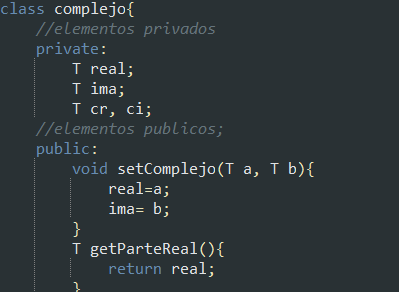
Primero utilizamos un método auxiliar que divida al elemento y compare cada elemento según el pivote, si ambos elementos se encuentran en el lado equivocado de la lista, estos intercambian posición.



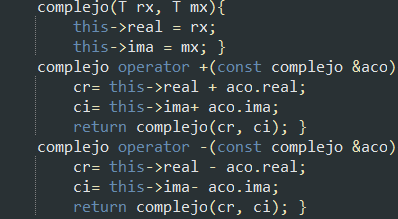
El método genérico final parte el arreglo y organiza cada sublista de acuerdo con el pivote.

Ejercicio 2

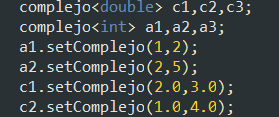
Desarrolle una clase complejo que permita operar con números de este tipo. Dicha clase, almacenará el valor real e imaginario de un elemento en atributos, así como proporcionará operadores sobrecargados para realizar las operaciones más habituales: Adicionalmente, se deberán sobrecargar los operadores de igualdad y desigualdad (== y !=) así como el operador de asignación (=).



Establecemos métodos genéricos, dado que la clase es completamente genérica, de tal manera que pueda aceptar más de un tipo de dato primitivo.



Sobrecargamos cada operación posible que se pueda realizar con números complejos, posteriormente.

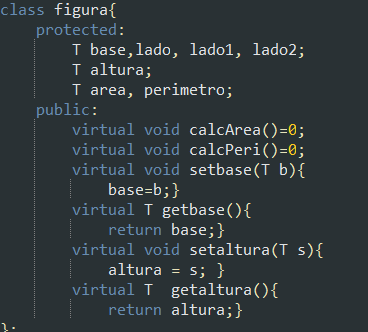


Implementación de mas de un tipo de dato, y em ambos se pueden realizar operaciones de tipo números complejo.

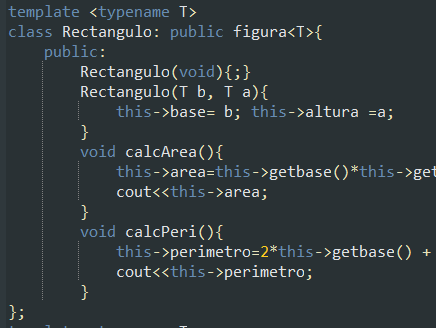
Ejercicio 3

Desarrolle una clase figurasGeométricas la cual deberá de contener los métodos necesarios para calcular el perímetro y área de un triángulo, cuadrado, rectángulo, rombo, romboide y trapecio. Para esta clase, deberá implementar todos los elementos utilizando plantillas de C++ y herencia. Dentro de la función main, muestre como podría ocupar la clase tomando en cuenta distintos escenarios asociados al uso de plantillas.

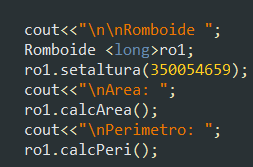
Primero establecemos una clase abastracta con los métodos y atributos principales que necesita cada figura



Cada tipo de figura, el cual se representa en forma de clase va a heredar de la clase figura, de modo que puedan ocupar sus atributos y métodos.



Se especifica el los métodos de calculo de área y perímetro para cada figura, en cada una de ellas se usan atributos genéricos, de modo que el calculo acepta diferentes tipos de valores.



En el main, se establecen figuras con distintos tipos de valores, y en cada una de ellas se calcula su área y perímetro.

Ejercicio 4

Crea una clase CuentaBancaria con atributos para el número de cuenta (int de

8 números), el id del cliente (int de 6 números), el saldo actual (float) y el interés

anual que se aplica por el manejo de la cuenta (int-porcentaje). Defina en la clase

los siguientes métodos:

• Constructor por defecto y constructor con id, saldo e interés.

• Setters y getters para el manejo de los atributos de la clase.

• GenerarId(): creara el id de una cuenta al momento de instanciar el objeto.

• actualizarSaldo(): actualizará el saldo de la cuenta aplicándole el interés

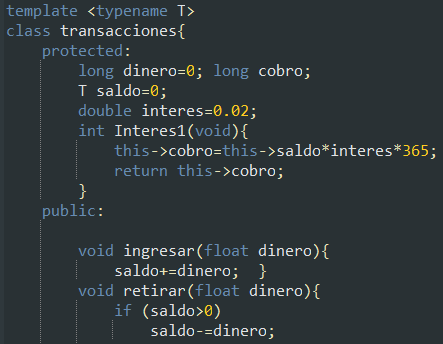
diario (interés dividido entre 365 aplicado al saldo actual).

• ingresar(float): permitirá ingresar una cantidad en la cuenta.

• retirar(float): permitirá sacar una cantidad de la cuenta (si hay saldo).

• MostrarInformacion(): Mostrara todos los datos de la cuenta.

Para este problema, establezco una clase que abarque las transacciones principales de la cuenta bancaria, el saldo de la cuenta bancara es de tipo genérico, de manera que se pueda ingresar cualquier valor numérico, pues así sucede con el dinero



Para cada método, se establecen operaciones estándar para la asignación de números de identificación, junto con la implementación de interés simple para los 365 días del año, con una tasa del 2% de interés, ya que me parece una tasa genérica para cuentas de ahorro.

