Práctica 2

Para cada ejercicio debes entregar toda la carpeta de la solución de Visual Studio. Antes de comprimir la carpeta debes borrar los ficheros innecesarios. Para ello debes ir al menú Build, y elegir **"Clean Solution"** (limpiar solución). En el caso contrario se perderá puntos.

La práctica se realizará por parejas, pero solo uno de los alumnos de cada grupo debe entregar la práctica en blackboard.

Es necesario escribir en un comentario los nombres de los autores al principio del fichero que contiene el programa principal. En el caso contrario se perderá puntos.

Si en algún ejercicio se ha indicado el orden o formato de entrada, debes respetar este orden/formato. En el caso contrario se perderá puntos.

1.-Creación de tu propia clase String

Desarrollar una clase llamada Mystr que hará las funciones de la clase String que c y c++ no tienen por defecto. Esta clase Mystr nos permitirá operar de manera sencilla con las cadenas de caracteres, de tal manera que no tendremos que preocuparnos de gestionar la memoria para trabajar con ellas (una vez completada esta clase).

Para ello, crearemos una Lista contigua de caracteres con esta clase. Las funciones a implementar son las siguientes:

1.1.-Constructores

Vacío, pasando otro objeto de tipo Mystr(constructor copia), o un array de char. En los dos últimos métodos se deberá reservar la memoria exacta para guardar ese array de char (incluyendo el '\0').

- Tener en cuenta que una cadena escrita en c++ como "Hola", realmente es un char [5] (De 5 posiciones, donde su última posición es '\0'. En este caso, la capacidad reservada será del doble de la cadena proporcionada.
- Para saber el tamaño una cadena de C, podemos usar la función strlen. Esta función cuenta el número de caracteres hasta llegar al '\0'.

1.2.-Sobrecarga de Operadores

Implementar la sobrecarga de los siguientes operadores **utilizando la función Compare del siguiente apartado**:

- inline bool operator==(const Mystr& other); //Usar la función Compare.
- inline bool operator!=(const Mystr& other); //Puede ser el opuesto a ==
- inline bool operator< (const Mystr& other); //Usar la función Compare (del siguiente apartado)
- inline bool operator> (const Mystr& other); //Usar la función Compare (del siguiente apartado)
- inline bool operator<=(const Mystr& other); //Puede ser el opuesto a >

- inline bool operator>=(const Mystr& other); //Puede ser el opuesto a <

Implementar las siguientes sobrecargas que darán una mayor utilidad a la clase:

- inline char& operator[] (int index); //Devuelve el caracter en el elemento “index”.
- inline Mystr operator+ (const Mystr & other); //Concatena una cadena a otra.

Puedes encontrar información de la sobrecarga de operadores en este link:
https://en.cppreference.com/w/cpp/language/operators

1.3.-Funciones

- unsigned int Length(); //Tamaño del array de caracteres. Se recomienda usar strlen en vez de almacenar el largo de la cadena en el objeto.
- unsigned int Capacity(); //La capacidad de la memoria reservada.
- int Replace(char find, char replaceBy); //Busca todos los caracteres iguales a “find” y los sustituye por replaceBy. Devuelve el número de caracteres remplazados.
- int Compare(const Mystr & other); //Devuelve 0 si ambas cadenas son iguales, 1 si la primera cadena es mayor que la de “other”, -1 si la de “other” es mayor que la primera. **-Se puede usar strcmp-**

  Ejemplo:

  Mystr(“Hola”).Compare(Mystr(“Adios”) Devuelve: -1.

  Mystr(“Hola”).Compare(Mystr(“Hola”) Devuelve: 0.

- int Remove(char find); //Busca todos los caracteres iguales a “find” y los borra. Devuelve el número de caracteres borrados. *Atender al decremento de la capacidad*

  Ejemplo: Mystr(“Ho-la, qu-e Tal”).Remove(‘-’) == Mystr(“Hola, que Tal”). En este caso se eliminan los guiones de tal manera que la igualdad se cumple.

- Mystr Right(unsigned int num); //Devuelve la subcadena de “num” caracteres empezando por el final.

  Ejemplo: Mystr(“LolitoFernandez”).Right(9) == Mystr(“Fernandez”). En este caso se recortan los 9 últimos caracteres de tal manera que la igualdad se cumple.

- Mystr Left(unsigned int num); //Devuelve la subadena comprendida entre el inicio y “num”.

  Ejemplo: Mystr(“LolitoFernandez”).Left(6) == Mystr(“Lolito”). En este caso se recortan los 6 primeros caracteres de tal manera que la igualdad se cumple.

- Mystr Substring (unsigned int initialIndex, unsigned int finalIndex); // Devuelve la subcadena partiendo de initialIndex hasta llegar a finalIndex.

  Ejemplo: Mystr(“>>>---(((><)))--->”).Substring(6,16) == Mystr(“(((><)))”). En este caso se recoge desde el carácter 6 hasta el 16, quedando 10 caracteres que cumplen la igualdad.

- int TrimRight(); //Elimina los espacios en blanco que haya por la derecha del texto. Devuelve el número de espacios eliminados. *Atender al decremento de la capacidad*
- int TrimLeft(); //Elimina los espacios en blanco que haya por la izquierda del texto. Devuelve el número de espacios eliminados. *Atender al decremento de la capacidad*
- int Trim(); //Elimina los espacios en blanco en ambos lados del texto. Devuelve el número de espacios eliminados. *Atender al decremento de la capacidad*
- int ToUpper(); //Convierte todos los caracteres en letras mayúsculas. Devuelve el número de caracteres que han sido cambiados a mayúsculas. Se permite usar toupper de C.
- int ToLower(); //Convierte todos los caracteres en letras minúsculas. Devuelve el número de caracteres que han sido cambiados a minúsculas. Se permite usar tolower de C.
- bool StartsWith(const Mystr& other); //Devuelve true si la cadena empieza con "other".
- bool EndsWith(const Mystr& other); //Devuelve true si la cadena termina con "other".
- Mystr Concatenate(const Mystr& other); //Concatena la cadena añadiendo "other" de manera consecutiva. *Atender al incremento de la capacidad de la primera cadena*

  Ejemplo: Mystr("Hola, soy el botijo").Concatenate(Mystr(", regordete y con pitorro")) == Mystr("Hola, soy el botijo, regordete y con pitorro").

- Mystr Introduce(const Mystr& other, int unsigned int index); //Concatena la cadena introduciendola entre medias, empezando en la posición indicada por index. *Atender al incremento de la capacidad de la primera cadena*

  Ejemplo: Mystr("Cadenamolona").Introduce(Mystr(" no "), 6) == Mystr("Cadena no molona").

1.4.-Consideraciones

No se permite utilizar librerías que implementen la funcionalidad pedida. Sin embargo, si se permite la utilización de la librería <cstring> para ayudarnos con las operaciones de arrays y cadenas de caracteres.

Utilizar malloc, realloc y free dependiendo de las necesidades de la función utilizada.

El incremento de la capacidad será del doble de la capacidad actual. Al disminuir la capacidad, será la mitad, que ocurrirá cuando la cadena pase a ser la mitad de la mitad.

Ejemplo: Utilizando Mystr cad("Hola cadena"), la capacidad inicial será 24 (Con el \0). Si utilizamos cad.Concatenate("Escribiendo algo +"), la capacidad será 48. Si utilizamos con la primera cadena cad.Left(4), la capacidad de la nueva cadena pasará a ser 12 (Debido a que recorta de una cadena de capacidad 24 y el nuevo length es menor a la mitad de la mitad de este valor.

2.-Las pruebas de esta clase

Pruebas simples

Incluir en el main las siguientes operaciones:

- Creamos tres cadenas Mystr: “Kha\’zix”, “son las” y “Thresh”.
- Copiar una de ellas en otra cadena nueva usando el constructor copia, concatenad a su contenido: “ola que Ashe”.
- Recortar a esta última cadena por la izquierda 11 caracteres. Después 7 por la derecha.
- Concatenar a esta última cadena “Ketchup” y convertirlo todo a mayúsculas.
- Pasar la primera de todas las cadenas a minúsculas (La de “Kha\’zix”).
- Concatenar las tres primeras cadenas.
- Comparar esta última cadena resultante con la cadena en mayúsculas para ver cuál es mayor.
- Imprimir la cadena mayor, luego la menor.

Pruebas con lista doblemente enlazada

Creada una estructura que contenga:

- Puntero al nodo anterior en la lista.
- Un objeto de clase Mystr.
- Puntero al nodo siguiente de la lista.

Como en actividades de este tema, se propone crear un objeto que gestione una lista doblemente enlazada utilizando esta estructura como nodo.

Se añadirán a esta lista los siguientes nodos, con el siguiente contenido en sus cadenas (Espacios incluidos):

1. “ DOWN DOWN END_ ”
2. “ RULETA MOLA MAZO EHHH “
3. “ NOOO CONFUNDAS ESTA TAREA “
4. “ PEPE PAPA PIPI POPO PUPU “
5. “ UP__ UP__ UP__ “

El objetivo es recorrer esta lista doblemente enlazada en ambos sentidos de la siguiente manera:

1. Tras leer una cadena hay que usar la función Trim() e imprimir la cadena.
2. Se comprueba si esta misma cadena leída empieza por: UP, DOWN o END.
3. En función del resultado obtenido, la cadena siguiente a procesar (y las sucesivas hasta encontrar una cadena con UP, DOWN o END), será la posicionada antes (si encontramos UP) o la de después (si encontramos DOWN) y no se parará hasta encontrar otra etiqueta UP, DOWN o END.
4. Antes de pasar la siguiente cadena, se eliminarán 4 caracteres por la izquierda, sobreescribiendo a la actual cadena de la lista doblemente enlazada. Tras esto, repetimos el proceso desde el punto 1.
5. Al llegar a END, el programa termina.

El output del programa será:

```
DOWN    DOWN    END_

RULETA      MOLA    MAZOO EHHH
```

NOOO CONFUNDAS ESTA TAREA

PEPE PAPA PIPI POPO PUPU

UP__ UP__ UP__

PAPA PIPI POPO PUPU

CONFUNDAS ESTA TAREA

TA MOLA MAZOO EHHH

DOWN END_

MOLA MAZOO EHHH

UNDAS ESTA TAREA

PIPI POPO PUPU

UP__ UP__

POPO PUPU

S ESTA TAREA

MAZOO EHHH

END_