El taller 1 se hará en la última media hora

Vamos a implementar en C++ con header (.h) y .cpp el tipo PILA y el tipo COLA usando el tipo lista doblemente enlazadas con centinelas

Suponga que el tope de una pila será el primer elemento de la lista que la implementa y el frente de una cola será el primer elemento de la lista que la implementa

- Crear una carpeta vacía llamada **prueba_pilaYcola**.
- Colocar en la carpeta **prueba_pilaYcola,** los archivos: **listaDoble.h y listaDoble.cpp** que están en MODULO 7 CONTENIDOS->laboratorio 5
- Ejecutar VSCODE y abrir esa carpeta (open folder)

- Crear los archivos vacíos: **pila.h, pila.cpp, cola.h y cola.cpp**
- Crear un archivo llamado **prueba.cpp** donde harán la prueba de los tipos Pila y Cola,
- cambiar el tasks.json : **"${file}", por "${workspaceFolder}*.cpp",** si quiere ejecutar el programa con VScode. Si no utilice el comando en terminal:
  - g++ -o prueba prueba.cpp Lista_con_centinela.cpp

Ejercicio final: concatenar dos listas y devolver la lista concatenada (nueva)

TALLER 1

RECUERDEN SUBIR A MI CARPETA DRIVE COMPARTIDA “TALLER 1” EL CODIGO DE LA FUNCION QUE SE LES PIDE. PARA SUBIRLA, GUARDENLA EN UN ARCHIVO DE TEXTO CON NOMBRE:

APELLIDO_NOMBRE_CEDULA.cpp

Y SUBEN EL ARCHIVO AL VINCULO DRIVE:

https://drive.google.com/drive/folders/1PvACuvwnytsDEgV1PPqlAYb5KUfIs_Mx?usp=sharing

este vinculo esta en MODULO 7 CONTENIDOS -> PLAN DE CLASES

5 puntos si corre la función haciendo una prueba, si no 0 puntos

Taller 1:

Dada definición de lista doblemente enlazada con centinelas (crear carpeta ListaDoble y bajarse listaDoble.h y listaDoble.cpp de modulo 7 laboratorio 5 a esa carpeta)

Hacer y agregar (al .h y .cpp) una función llamada obtener_pos, que devuelva el elemento (entero) de la lista en una posición dada. Su protocolo es el siguiente. Se supone que la posición es válida:

int obtener_pos(const Lista_dob_enl_cen & lista, int pos)

Para probar su función haga un programa, llamado prueba.cpp, que cree una lista vacía, le agregue 3 elementos, imprima la lista y utilice la función para obtener el elemento en la posición 0, luego en la posición 2 y luego en la posición 1 y los imprima.

Recuerde que para correr su programa con VSCODE debe:

En el archivo tasks.json reemplazar "${file}", por "${workspaceFolder}*.cpp",

Si no, puede generar el ejecutable por terminal con el comando:

g++ -o prueba prueba.cpp listaDoble.cpp

Comencemos por implementar el tipo Pila. El archivo pila.h contendrá:

```
#include "listaDoble.h"

// El tipo Pila
typedef  Lista_dob_enl Pila;

//Verificar si la pila v está vacía
// no es necesaria
// bool esVacia(Pila&);
// Devuelve una pila vacía
Pila& crearPilaVacia() ;
// Empila data en la pila
void empilar(Pila &pila , int data);
// Desempila el tope
// La pila no puede estar vacía
void desempilar(Pila &pila);
// Devuelve el tope de la Pila v
// la pila no debe estar vacía
int tope(const Pila &pila);
// imprime la pila
void imprimir_pila(Pila &pila);
// devuelve el numero de elementos de
la pila
int num_elem_pila(const Pila & pila);
```

Ahora el archivo pila.cpp

Pensemos cómo sería cada función de Pila utilizando Lista_con_centinela

El comienzo del archivo contendrá:

```
#include "pila.h"
```

Ahora creemos el archivo pila.cpp. Supondremos que el tope es el primero de la lista

```
#include "pila.h"

Pila& crearPilaVacia() {
    return crearListaVacia();
}
// Empila data en la pila v
void empilar(Pila &pila , int data){
    insertar(pila, data, 0);
}
// Desempila el tope
// La pila no puede estar vacía
void desempilar(Pila &pila){
    eliminar(pila,0);
}
```

```
// Devuelve el tope de la Pila v
// la pila no debe estar vacía
int tope(const Pila &pila){
    return pila.centinela->proximo->data;
}
//Verificar si la pila v está vacía
//bool esPilaVacia(const Pila &pila) {
    return esListaVacia(pila); }
void imprimir_pila(Pila &pila){
    imprimir(pila);
}
int num_elem_pila(const Pila & pila){
    return pila.num_elem;
}
```

Ahora implementemos el tipo Cola

Recuerden no escribir los momentarios....

Ahora creemos el archivo cola.h. Supondremos que el frente es el primero de la lista

```
#include "lista_simple_cent.h“

typedef Lista_con_centinela Cola;

//crear cola vacia
Cola& crearColaVacia() ;

// devolver el frente
int frente(const Cola& cola);

/// eliminar el frente
// supone cola no vacía
void desencolar(Cola& cola);

// encolar elem
void encolar(Cola& cola, int elem);

// devolver el numero de elementos
int num_elem_cola(const Cola& cola);

// imprimir cola
void imprimir_cola(const  Cola&);

// note que aqui no defino esVacia() porque es
// la misma que esVacia() de Lista_con_centinela
// bool esVacia(const Cola &);
```

Ahora creemos el archivo cola.cpp

Pensemos cómo sería cada función de Cola utilizando Lista_con_centinela

El comienzo del archivo contendrá:

#include “cola.h"

Ahora creemos el archivo cola.cpp. Supondremos que el frente es el primero de la lista

```
#include "cola.h"
// devuelve cola vacia
Cola& crearColaVacia() {
    return crearListaVacia();
}
// el frente es el primero de la list vaciaa
// se supone que la cola no esta vacia
int frente(const Cola& cola){
    return cola.centinela->proximo->data;
}
// elimina el frente de la cola
// se supone que la cola no esta vacia
void desencolar(Cola& cola){
    eliminar(cola, 0);}
// inserta elem de último en la lista
void encolar(Cola& cola, int elem){
    insertar(cola, elem, cola.num_elem);
}
// devuelve el num. de elementos
int num_elem_cola(const Cola& cola){
    return cola.num_elem;
}
// imprime cola
void imprimir_cola(const Cola& cola){
    imprimir(cola);
}
// bool esVacia(const Cola& cola) no se define
```