EJERCICIOS DE LA PRIMERA ENTREGA

1. Desarrolla una función dentro de la clase que cuente el número de ocurrencias de un elemento en una lista ligada. Por ejemplo, si la lista es L1 = {1, 2, 3, 4, 5, 4, 12}, al recibir como entrada 4 debe regresar 2

Explicación de la solución: Para solucionar este ejercicio, se debe de pedir el número el cual se quiere conocer cuantas veces se encuentra en la lista, por lo que al saber “cuanto” quiere decir que necesitaremos contar, por lo que haremos uso de un contador, para ir contando el número de ocurrencias de dicho número,

Entonces, se solicita el número, se recorre la lista, y cada vez que el valor del nodo que se encuentre en ese momento visitado sea igual al valor que ingreso el usuario, se incrementará en 1 el valor del número de ocurrencias, al final se imprimirá en pantalla el número de ocurrencias de dicho número.

Una captura de pantalla de una computadora

Descripción generada automáticamente

1. Escribe una función dentro de la clase que obtenga el enésimo nodo en una lista ligada. La función recibe de entrada un índice entero y regresa el nodo correspondiente a dicha posición.

NOTA: En este ejercicio asumí que el primer elemento tiene la posición 0, como si se tratará de un arreglo, pero en este caso es una lista ligada simple dinámica

Explicación de la solución del problema: Se declara e inicializa una variable que con el valor de 0, esta variable simulara el índice de la lista, por lo cual se inicializa con el valor de 0, ya que los espacios de un arreglo se empiezan a contar desde esa posición.

Se va recorriendo la lista y conforme se vaya pasando al siguiente nodo se incrementa el valor del índice, esto se hace hasta llegar al valor que haya ingresado como posición el usuario, y es en ese momento donde se imprime el valor del nodo, después de ese momento se rompe el ciclo para que ya no se hagan recorridos y ejecuciones innecesarias.

Captura de pantalla de computadora

Descripción generada automáticamente

1. Ejercicios sobre nodos: Recrear en memoria mediante el uso de la clase Nodo, el siguiente diagrama

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Explicación de la solución del problema:

Para este ejercicio se utilizó la implementación de la clase Nodo, está clase tiene un constructor al que hay que pasarle el valor que guardará, e inicializa el valor del nodo siguiente como nullptr, también tiene métodos setters y getters, métodos setter para establecer valores (métodos de escritura) y para retornar valores (métodos de lectura).

La solución fue la siguiente, crear 7 nodos, cada uno con el valor de los nodos que están en la imagen de arriba, y se hizo los enlaces como se muestran arriba, los enlaces quedaron de la siguiente manera:

nodo1->setSig(nodo2);

nodo2->setSig(nodo3);

nodo3->setSig(nodo4);

nodo4->setSig(nodo1);

nodo5->setSig(nodo2);

nodo6->setSig(nodo3);

nodo7->setSig(nodo2);

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para saber que la implementación es correcta se imprime en pantalla el mensaje de “Todo bien” para indicar que los enlaces fueron correctos

Una captura de pantalla de una computadora

Descripción generada automáticamente

1. Realiza en el main una función que dadas dos listas: l y j, traslada a la lista l todos los enteros almacenados en la lista j, de modo que los elementos trasladados se agregan al final de la lista l en el orden en el que aparecen en la lista y está última queda vacía. Escribe un programa que pruebe tu función y muestre el contenido final de todas las listas. Emplea únicamente los métodos de la interfaz Lista. Usa listas simplemente ligadas. Por ejemplo si l = {1, 2, 3, 4, 5} y j = {6, 7, 8, 9}. Las listas resultantes son: l = {1, 2, 3, 4, 5, 6, 7, 8, 9} y j = {}. La función debe tener la siguiente forma:

Void traslada(ListaLigada &l1, ListaLigada &l2)

{

//Algoritmo

}

Explicación de la solución del ejercicio: Para este ejercicio solo debemos de tener las dos listas con elementos, debemos de hacer un algoritmo para que mientras la lista 2 tenga elementos, obtenga el valor de la cabeza de la lista 2, este valor se debe de introducir al final de la lista 1, finalmente se debe de borrar el nodo cabeza de la lista 2, y el nuevo valor de cabeza de la segunda lista debe de ser el nodo posterior al que era cabeza, es decir la nueva cabeza será cabeza = cabeza.getSiguiente(), todo este proceso se hace mientras la segunda lista tenga elementos, al final de la ejecución del algoritmo la primera lista contendrá todos los elementos de ambas listas, mientras que la segunda lista estará vacía.

Captura de pantalla de un celular

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Agrega un método a la clase ListaEnlazadaSimple que invierta los elementos de la lista, únicamente a partir de una posición P, dada modificando sus enlaces. El método debe tener la siguiente forma:

Void ListaLigada::invierte\_desde\_p(Nodo \*posicion)

{

}

Explicación de la solución del problema dada una lista, se pide al usuario que ingrese una posición, por lo cual se utiliza un contador que va simulando las posiciones de la lista hasta llegar al nodo que cumpla con la posición que indico el usuario, a partir de esa posición se invierten los elementos de la lista, por lo que el nodo que se encuentre en la posición n-1 debe de estar conectado a la cola de la lista, los demás elementos deben estar invertidos de manera normal.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente