Logotipo, nombre de la empresa

Descripción generada automáticamenteLista doble

Nombre: Oscar Alejandro Penilla Skakievich

Fecha: 25/10/22

Grupo:5D1

Registro: 20300701

**Descripción del programa:**

Se usaran nodos, estos crearan listas en las cuales sus variables y métodos se almacenarán en espacios dinámicos de memoria, en este caso lo único que debemos hacer es almacenar números enteros dentro de una de estas y la excepción es que buscaremos como comunicar el valor pasado con uno actual, cosa que solo podíamos ver en el valor siguiente

**Requerimientos:**

El programa debe de lograr hacer estas funciones

* Agregar. Añadir un elemento
* Eliminar uno. Eliminar un elemento
* Eliminar todos. Eliminar todos los elementos
* Buscar. buscar un elemento por su valor
* Contar. contar los elementos
* Obtener  o mostrar. muestra en donde inician los elementos y en donde terminan

**PSP:**

|  |  |  |  |
| --- | --- | --- | --- |
| Contenido del PSP – Lista doble | | | |
| Código fuente: | Estimado | Real | Error |
| (min) | (min) |
| Tiempo de Diseño | 30 | 5 | 83% |
| Tiempo de Codificación. | 120 | 10 | 92% |
| Tiempo estimado de fabricación | 150 | 15 | 90% |
| Total, de líneas de código nuevas. | 40 | 6 | 85% |
| Total, de líneas de código reusadas. | 120 | 221 | 84% |
| Total, de líneas de código del programa. | 160 | 227 | 42% |
| Cantidad de errores de compilación. | 30 | 0 | 100% |
| Cantidad de errores de ejecución. | 20 | 0 | 100% |

**Código fuente:**

#include <iostream>

using namespace std;

/\*dudas

Como era para poner las sentencias en el codigo? jeje

como se cambia la posicion de un nodo? para cuando es eliminado

\*/

class nodo{

public:

int x;

nodo();

nodo(int x);

nodo \*sig;

};

nodo::nodo(){

x=0;

sig=NULL;

}

nodo::nodo(int x){

this->x=x;

sig=NULL;

}

class lista{

private:

nodo \*i,\*a; //"a" es auxiliar e "i" de inicio

public:

lista();

void add(int x); //anadir un elemento

void del(int x);//eliminar un elemento

void delall();//eliminar todos los elementos

nodo\* busc(int x);//buscar un elelemento por su valor

int cont();//contrar los elementos

// void impr(); //se deberia imprimir el objeto? no no se puede imprimir, esto solo sirve para guardar info

void obte();//muestra en donde inician los elementos y en donde terminan solo vamos a mostrar

~lista();

};

lista::lista(){

a=NULL;

i=NULL;

}

void lista::add(int x){

if(i==NULL){ //si no hay nada se crea un nodo al inicio

i = new nodo(x);

}else //sino auxiliar va a tener un nodo

a= new nodo(x);

nodo \*p=i; //se hace un puntero del tipo nodo hacia inicio

while(p->sig !=NULL){//mientras el siguiente no sea nulo

p=p->sig; //nuestro puntero lo asignamos hacia el siguiente

//y este siguiente tomara el lugar de auxiliar que antes ya era una nueva varibale

}

p->sig=a; //en el momento en el que el siguiente tenga algo seguira hasta que encuentre algo vacio

}

void lista::del(int x){

/\*

hay diferentes casos

cuando se elimina el primero

cuando se elimina uno cualquiera

cuando se elimina un espacio nulo \v (palomita bonita)

puedo llamar a buscar para conocer el valor que quiero eliminar

\*/

nodo\* p = NULL; //creamos un puntero para saber la posicion anterior

if(i){

a=busc(x); //nos va a retornar un puntero de tipo nodo(posicion de memoria) para saber que variable eliminar

if(a == i){ // si mi valor esta en la primera posicion que va a hacer?

//Dos casos, si solo tiene un elemento o si existen mas de uno

i=i->sig; //i lo pasamos a la siguiente posicion

//y eliminamos a ya que esta en la primera posicion

}else{ // que pacas cuando se elimina un valor cualquiera

while(p->sig != a){ //para que tengamos una variable antes de nuestra posicon actual

p = p->sig;

} //fin de siguienten't, o sea, pasado

p->sig = a->sig;

}

delete a; //como al final siempre se va a eliminar a, solamente lo ponemos afuera de nuestras sentencias

}else{

cout << "No hay nada que eliminar"<<endl;

}

}

void lista::delall(){

if(i==NULL){

cout<<"Elementos eliminados"<<endl;

}

a=i;

while (i){

i=i->sig;

delete a;

a=i;

}

i=NULL;

a=NULL;

}

nodo\* lista::busc(int x){

/\*

Cuando lo que se busca no se encuentra

cuando se encuentra

no tengo la mas minima idea je je je

\*/

a=i; //auxiliar es igual al incio

while(a){ //minetras auxiliar no sea nulo se repetira

if(a->x == x){ // si el valor de x en a es igual a el valor de usuario

return a; //retornamos el valor de tipo nodo

a=a->sig; //y pasamos al siguiente

}

a=a->sig;

}

return NULL; //el while se repetira hasta que encuentre el valor, pero en el caso de que no este, simplemente dará un null

}

int lista::cont(){

int cont =0;

a=i; //para no alterar el incio

while(a){

a=a->sig;//para tomar el siguiente

cont++;//aumentamos el contador

}

return cont; //retornamos nuestro contador como haya quedado

}

void lista::obte(){

//cuando no hay nada que obtener

//cuando si hay algo, contando que deben aparecer todos unidos, sin espacios

a=i;

nodo\* p = NULL; //creamos un puntero para saber la posicion anterior

p=i;

//fin de siguienten't, o sea, pasado

while(a){ // mientras no sea nulo nuestra a

while(a != i && p->sig != a){ //para que tengamos una variable antes de nuestra posicon actual

p = p->sig;

}

cout <<"Valor anterior: " << p <<" Nodo: " << a <<", Valor del nodo: " <<a->x <<", A donde apunta "<< a->sig <<endl;

a=a->sig;

}

}

lista::~lista(){

delall();

}

int main(){

//variables main

int op=0, salida=0, x=0;

nodo\* punter;

lista ob1;

do{

cout<<"Elija una opcion"<<endl;

cout<<"1.-Agregar"<<endl;

cout<<"2.-Eliminar"<<endl;

cout<<"3.-Eliminar todos"<<endl;

cout<<"4.-Buscar"<<endl;

cout<<"5.-Contar"<<endl;

cout<<"6.-Obtener"<<endl;

cout<<"7.-Salir"<<endl;

//

cin>> op;

switch(op){

case 1:

{

cout<<"Opcion 1"<<endl;

cout<<"Valor a agregar:";

cin >> x;

ob1.add(x);

break;

}

case 2:

{

cout<<"Opcion 2"<<endl;

cout<<"Valor a eliminar:";

cin >> x;

ob1.del(x);

break;

}

case 3:

{

cout<<"Opcion 3"<<endl;

ob1.delall();

break;

}

case 4:

{

cout<<"Opcion 4"<<endl;

cin >> x;

if(ob1.busc(x)){

cout << "Elemento encontrado: ";

cout << ob1.busc(x) <<endl;

}else{

cout << "Elemento no encontrado"<<endl;

}

break;

}

case 5:

{

cout<<"Opcion 5"<<endl;

x = ob1.cont();

cout<<"Hay "<< x <<" elementos"<<endl;

break;

}

case 6:

{

cout<<"Opcion 6"<<endl;

ob1.obte();

break;

}

}

}while(op !=7);

ob1.delall(); //por si el usuario olivda eliminarlos, yo los elimino

return 0;

}