Logotipo, nombre de la empresa

Descripción generada automáticamenteLista circular 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 unir el ultimo dato con el primero, en esta lista dejará de existir el valor nulo que tanto nos podría ayudar y además deberemos comunicarla con un valor pasado

**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 circular | | | |
| Código fuente: | Estimado | Real | Error |
| (min) | (min) |
| Tiempo de Diseño | 20 | 5 | 75% |
| Tiempo de Codificación. | 30 | 10 | 67% |
| Tiempo estimado de fabricación | 50 | 15 | 70% |
| Total, de líneas de código nuevas. | 0 | 0 | #¡DIV/0! |
| Total, de líneas de código reusadas. | 120 | 335 | 179% |
| Total, de líneas de código del programa. | 120 | 335 | 179% |
| Cantidad de errores de compilación. | 10 | 0 | 100% |
| Cantidad de errores de ejecución. | 10 | 0 | 100% |

**Código fuente:**

//intento de lista circular

/\*

Se busca que el ultimo valor este conectado con el primero y viceversa

como sabremos cuando parar de mostrar valores? facil, contando los que tenemos o hemos puesto

tambien guardando en una variable el ultimo valor puesto, no espera, pero y si se repiten? no entonces no

la funcion de contar debe de saber cuando parar Para asi poderle mandar un valor que nos diga cuando deterla

como sabe cuando parar?

el anterior a inicio es el final

entonces, deberia agregar uno valor antes de inicio

\*/

#include <iostream>

using namespace std;

/\*dudas

\*/

class nodo{

public:

int x;

nodo();

nodo(int x);

nodo \*sig;

/\*nodo\* pas;

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

pas = pas->sig;

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

};

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){

//se tiene que unir el ultimo con el primero

//ya podemos poner cuantos valores queramos, solono se si queden bien puestos jsjsjs

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

i = new nodo(x);

i->sig = i;

}else { //sino auxiliar va a tener un nodo

a= new nodo(x);

a->sig = i;

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

while(p->sig !=i){//mientras el siguiente no sea igual a nuestro incio

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){

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

a=busc(x);

if(i && a){ //si i no es nulo

cout << "Valor de a: "<< a<<endl;

if(i == i->sig){ //si solo hay un valor eliminaremos todos, para evitarnos complejos

lista::delall();

}else if(a == i){ //si el valor a buscar es igual al incio

p=i;

cout << i <<endl;

/\*vamos a usar p como un auxiliar en este caso,

ya que no estamos ligando el final con el incio, tenemos que conocer el final para poder ligarlo al inicio

\*/

while(p->sig != i){

p = p->sig;

}

i=i->sig;

p->sig = i;

cout << i <<endl;

cout << i->sig <<endl;

delete a;

}else{

cout <<"Entre al else"<<endl;

p=i;

// int cont =0;

// int lim =lista::cont();

//para que tengamos una variable antes de nuestra psisision(si) actual

/\*

cubrimos antes si solo hay un elementos, pero que tal si solo hay dos (o mas)?

lo que haciamos aqui era que mientras p siguiente no fuera igual a el valor que buscabamos

p -> a = busq

y lo que haciamos aqui era solo

este eliminando uno despues del que si queremos eliminar

\*/

while(p->sig != a){ //mientras p siguiente no sea igual a el valor que buscamos

//se cicla

/\*if(cont != lim){

break;

}

cont++;\*/

// cout <<"Entre al while psig: "<< p->sig << " a: " << a <<endl;

p = p->sig;

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

// cout <<"sali en pasao"<<endl;

// cout <<"psig: "<< p->sig << " a: " << a<<" asig: "<< a->sig <<endl;

p->sig = a->sig; //cambiabamos la variable

// cout <<"psig: "<< p->sig << " a: " << a<<" asig: "<< a->sig <<endl;

delete a;

}

// delete a;

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

}else{ //si si es nulo

cout << "No hay nada que eliminar/no existe este valor"<<endl;

}

}

void lista::delall( ){

if(i==NULL){

cout<<"Elementos eliminados"<<endl;

}else if(i == i->sig){

delete i;

}else{

a=i;

int cont =1;

int lim =lista::cont();

//cout << "lim "<<lim <<endl;

while (cont != lim){

// cout << "cont "<<cont <<endl;

i=i->sig;

delete a;

a=i;

cont++;

}

}

cout << "sale, no hay de queso namas de papa"<<endl;

i=NULL;

a=NULL;

}

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

a=i; //auxiliar es igual al incio

int cont =0;

int lim =lista::cont();

while(cont != lim){ //minetras auxiliar no sea nulo se repetira

// cout << "entre en el while cont "<< cont << " ax: "<< a->x << " x: " << x <<endl;

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;

cont++;

}

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(){ //si logro que los cuente bien, podré poner esa sentencia como una constante para todos los ciclos de repeticion

//ya lo si funciona

//si i en nulo no hay nada

// si i sig es igual a i = 1

// si i sig no es igual a i, hay que contar

int cont =1;

nodo\* p = NULL;

a=i; //para no alterar el incio

//cout << "ola entre correctamente"<<endl;

//a nunca sera nulo, es el problema, pero, lo unimos a inicio, entonces, cuando el siguiente valor sea el de inicio, debe parar

if(i==NULL){

return 0;

}else if(i == i->sig){

return 1;

}else{

//cout << "ola soi igual a: "<< a->x <<" a: " << a <<" asig: "<< a->sig <<" i: "<< i <<endl;

while(a->sig != i){ //mientras que a siguiente no sea igual a i

// cout << "ola soi igual a: "<< a->x <<" a: " << a <<" asig: "<< a->sig <<" i: "<< i <<endl;

cont++;

a=a->sig;

//cout << "ola estoy en el while y es la vez "<< cont <<endl;

}

// cout << "ola sali vivo"<<endl;

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

int lim =lista::cont();

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

if(i==NULL){

cout <<"No hay valores ingresados"<<endl;

}else if(i == i->sig){ //cuando solo haya uno

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

}else{ // cuando hay mas de uno

a=i;

p=i;

int cont =0;

while(cont != lim){

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;

// cout << "Entra? despues de eliminar"<<endl;

//a=a->sig; // asi puedes hacer que vaya en orden primero el primero

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

a=a->sig; //en este orden puedes hacer que vaya del otro lado, primero el ultimo

cont++;

// mientras no sea igual a nuestro limite de numeros

}

}

}

lista::~lista(){

delall();

}

int main(){

//variables main

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

// int contaux = 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<<contaux<< endl;

cout<<"Opcion 1"<<endl;

cout<<"Valor a agregar:";

cin >> x;

//contaux++;

ob1.add(x);

//cout<<contaux<< endl;

break;

}

case 2:

{

//cout<<contaux<< endl;

cout<<"Opcion 2"<<endl;

cout<<"Valor a eliminar:";

cin >> x;

ob1.del(x);

//contaux--;

//cout<<contaux<< endl;

break;

}

case 3:

{

//cout<<contaux<< endl;

cout<<"Opcion 3"<<endl;

//contaux=0;

ob1.delall();

//cout<<contaux<< endl;

break;

}

case 4:

{

cout<<"Opcion 4"<<endl;

cout<<"Valor a buscar: ";

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;

}