Logotipo, nombre de la empresa

Descripción generada automáticamentePila

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 vamos a tener menos opciones y veremos distintas maneras de eliminar los datos que tengamos, como este se llama pila, el ultimo dato ingresado por el usuario será el primero que eliminemos

**Requerimientos:**

El programa debe de lograr hacer estas funciones

* Agregar. Añadir un elemento
* Eliminar uno. Eliminar un elemento
* Eliminar todos. Eliminar todos los elementos
* Obtener  o mostrar. muestra en donde inician los elementos y en donde terminan

**PSP:**

|  |  |  |  |
| --- | --- | --- | --- |
| Contenido del PSP – Pila | | | |
| Código fuente: | Estimado | Real | Error |
| (min) | (min) |
| Tiempo de Diseño | 10 | 25 | 150% |
| Tiempo de Codificación. | 100 | 180 | 80% |
| Tiempo estimado de fabricación | 110 | 205 | 86% |
| Total, de líneas de código nuevas. | 20 | 25 | 25% |
| Total, de líneas de código reusadas. | 80 | 141 | 76% |
| Total, de líneas de código del programa. | 100 | 166 | 66% |
| Cantidad de errores de compilación. | 30 | 20 | 33% |
| Cantidad de errores de ejecución. | 20 | 50 | 150% |

**Código fuente:**

// pila

#include <iostream>

using namespace std;

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();//eliminar un elemento

void delall();//eliminar todos los elementos

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(){

nodo\* p = NULL;

if(i){ //si el primero es nulo solo no va a eliminar nada, si hay algo sigue

a=i; //auxiliar va a ser igual a nuestra i

//p=i;

if(a->sig == NULL){

cout<< "Solo hay un valor que eliminar "<<endl;

delall();

}else{

//cout << "eif"<<endl;

while(a){ //mientras a sea verdadero se repetira

// cout << "ewhile "<< a->sig << endl;

if(a->sig ==NULL) { //si es nulo el siguiente significa que estamos en la ultima posicion

// cout << "f"<<endl;

p=i;

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

p = p->sig; //para dejar un espacio

}

p->sig = a->sig;

delete a; //y eliminamos la variable ultima

break;

}else{ //si el siguiente lugar no es nulo, entonces pasamos al siguiente

// cout << "sig"<<endl;

a=a->sig;

}

}

}

}else{

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

}

}

void lista::obte(){

//cuando no hay nada que obtener

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

a=i;

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

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

a=a->sig;

}

}

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;

}

lista::~lista(){

delall();

}

int main(){

//variables main

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

nodo\* punter;

lista ob1;

do{

cout<<"Fila: ultimo que entra, ultimo que se va"<<endl;

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

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

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

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

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

cout<<"5.-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;

ob1.del();

break;

}

case 3:

{

cout<<"Opcion 3"<<endl;

ob1.delall();

break;

}

case 4:

{

cout<<"Opcion 4"<<endl;

ob1.obte();

break;

}

}

}while(op !=5);

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

return 0;

}