Logotipo

Descripción generada automáticamente

# LISTA CIRCULAR DOBLE

# 

**Fecha:** 10/05/2023

**Materia:** Estructura de datos

**Profesor/Profesora:** Ismael López Buenrostro

**Nombre:** Luis Rigoberto Ramírez Llamas

**Registro:** 21100298

**Carrera:** Drllo. Software

**Grupo:** 4E1

**Descripción del programa**

Diagrama

Descripción generada automáticamenteSe pretende crear una lista enlazada circular doble genérica, la cual permita un nodo similar al contenido mostrado en la siguiente figura.

Tabla

Descripción generada automáticamente**Lista de requerimientos**

Se desea crear un programa que emplee dicha lista circular doble y que permita realizar las siguientes opciones.

Tabla

Descripción generada automáticamenteAdemás del funcionamiento de cada uno de los puntos anteriores se revisará:

a)El programa deberá contener por lo menos dos clases, una clase nodo y una clase lista.

b)Cada nodo deberá almacenar la información dé una persona(nombre, apellido, edad y registro)

c)TODAS las impresiones de mensajes o textos deberán ser realizadas desde el main.

d)No deberá haber código referente en el main, pues este solo contendrá el uso del objeto lista.

Evite:

a)Uso de variables globales.

b)Variables no inicializadas.

Notas:

a)El programa es 100% orientado a objetos.

b)La comunicación entre clases, realícese a través de métodos y sus parámetros.

**Contenido del PSP**

|  |  |  |  |
| --- | --- | --- | --- |
| Código fuente: | Estimado  (min) | Real  (min) | Error |
| Tiempo de Diseño | 120 | 130 | 8% |
| Tiempo de Codificación. | 180 | 200 | 11% |
| Tiempo estimado de fabricación | 300 | 330 | 10% |
| Total, de líneas de código nuevas. | 450 | 401 | 11% |
| Total, de líneas de código reusadas. | 0 | 0 | 0% |
| Total, de líneas de código del programa. | 450 | 401 | 11% |
| Cantidad de errores de compilación. | 30 | 10 | 66% |
| Cantidad de errores de ejecución. | 15 | 3 | 80% |

**Código fuente**

**LCD.cpp (main)**

#include <iostream>

#pragma once

#include "ListaCircularDoble.h"

#include "Nodo.h"

using namespace std;

int main()

{

string respuesta;

char R = NULL;

int dato = NULL;

ListaCircularDoble lis;

Persona persona;

while (R != 'S')

{

cout << "¿Que deceas hacer?" << endl;

cout << "\n[ A ]\tAgregar datos\n[ B ]\tBuscar\n[ C ]\tEliminar\n[ D ]\tModificar\n[ E ]\tContar nodos\n[ F ]\tMostrar todo\n[ G ]\tIterador\n[ S ]\tSalir" << endl;

cin >> R;

switch (R)

{

case('A'): //Agregar

{

lis.Agregar(persona);

break;

}

case('B'): //Buscar

{

cout << lis.Buscar(persona) << endl;

break;

}

case('C'): //Eliminar

{

respuesta = lis.Eliminar(persona);

cout << respuesta << endl;

break;

}

case('D'): //Modificar

{

lis.Modificar(persona);

break;

}

case('E'): //Contar nodos

{

int Cantidad = NULL;

Cantidad = lis.Contar();

cout << "La cantidad de Nodos actuales son:" << Cantidad << endl;

break;

}

case('F'): //Mostar todo

{

respuesta = lis.Mostrar();

cout << respuesta << endl;

break;

}

case('G'): //Iterador

{

lis.Iterador();

break;

}

default:

break;

}

};

}

**ListaCircularDoble.cpp**

#include "ListaCircularDoble.h"

#include "Nodo.h"

#include <iostream>

#include <string>

#include <string.h>

using namespace std;

void ListaCircularDoble::Agregar(Persona persona)

{

cout << "Dame los datos de la persona" << endl;

cout << "Nombre:\t" << endl;

fgets(persona.nombre, 50, stdin);

fgets(persona.nombre, 50, stdin);

cout << "Apellido:\t" << endl;

fgets(persona.apellido, 50, stdin);

cout << "Edad:\t";

cin >> persona.edad;

cout << "Registro:\t";

cin >> persona.registro;

if (inicio == NULL)

{

inicio = new Nodo;

inicio->persona = persona;

inicio->prev = inicio;

inicio->next = inicio;

iterador = inicio;

ultimo = inicio;

}

else

{

Nodo\* aux = inicio;

while (aux != ultimo)

{

aux = aux->next;

}

aux->next = new Nodo;

aux->next->persona = persona;

aux->next->next = inicio;

aux->next->prev = aux;

ultimo = aux->next;

inicio->prev = ultimo;

}

}

string ListaCircularDoble::Buscar(Persona persona)

{

string resultado = "";

cout << "¿Deseas buscar por nombre o registro?\n[ N ]\tNombre\n[ R ]\tRegistro\n" << endl;

cin >> R;

Nodo\* aux = inicio;

switch (R)

{

case('N'):

{

cout << "Dame el nombre a buscar" << endl;

cin >> fgets(persona.nombre, 50, stdin);

break;

}

case('R'):

{

cout << "Dame el registro a buscar" << endl;

cin >> persona.registro;

break;

}

}

if (inicio == NULL)

{

return "No hay datos para la busqueda";

}

do {

if (strcmp(aux->persona.nombre, persona.nombre) == 1 || aux->persona.registro == persona.registro)

{

cout << "Persona encontrada\n" << endl;

return resultado += "\nNombre:\t\t" + string(aux->persona.nombre) + "\nApellido:\t" + string(aux->persona.apellido) + "\n\nEdad:\t\t" + to\_string(aux->persona.edad) + "\n\nRegistro:\t" + to\_string(aux->persona.registro) + "\n\n";

}

aux = aux->next;

} while (aux != inicio);

if(resultado == "")

return "La persona no esta registrada";

}

string ListaCircularDoble::Eliminar(Persona persona)

{

Nodo\* aux = inicio;

if (inicio == NULL)

{

return "No hay datos para eliminar";

}

cout << "¿Deseas eliminar por nombre o registro?\n[ N ]\tNombre\n[ R ]\tRegistro\n" << endl;

cin >> R;

switch (R)

{

case('N'):

{

cout << "Dame el nombre a buscar" << endl;

cin >> fgets(persona.nombre, 50, stdin);

break;

}

case('R'):

{

cout << "Dame el registro a buscar" << endl;

cin >> persona.registro;

break;

}

}

do {

if (strcmp(aux->persona.nombre, persona.nombre) == 1 || aux->persona.registro == persona.registro)

{

aux->prev->next = aux->next;

aux->next->prev = aux->prev;

if (inicio == ultimo)

{

inicio = NULL;

ultimo = NULL;

iterador = NULL;

aux->next = NULL;

}

else {

if (aux == inicio) {

inicio = aux->next;

}

if (aux == ultimo) {

ultimo = aux->prev;

}

if (aux == iterador) {

iterador = aux->next;

}

}

}

aux = aux->next;

} while (aux != inicio);

return "Se ha eliminado todos los casos";

};

void ListaCircularDoble::Modificar(Persona persona)

{

Nodo\* aux = inicio;

if (inicio == NULL)

{

cout << "No hay datos para eliminar" << endl;

return;

}

cout << "¿Deseas modificar por nombre o registro?\n[ N ]\tNombre\n[ R ]\tRegistro\n" << endl;

cin >> R;

switch (R)

{

case('N'):

{

cout << "Dame el nombre a modificar" << endl;

cin >> fgets(persona.nombre, 50, stdin);

break;

}

case('R'):

{

cout << "Dame el registro a modificar" << endl;

cin >> persona.registro;

break;

}

}

do {

if (strcmp(aux->persona.nombre, persona.nombre) == 1 || aux->persona.registro == persona.registro)

{

cout << "Dame los datos de la persona" << endl;

cout << "Nombre:\t" << endl;

fgets(persona.nombre, 50, stdin);

fgets(persona.nombre, 50, stdin);

cout << "Apellido:\t" << endl;

fgets(persona.apellido, 50, stdin);

cout << "Edad:\t";

cin >> persona.edad;

cout << "Registro:\t";

cin >> persona.registro;

aux->persona = persona;

return;

}

aux = aux->next;

} while (aux != inicio);

return;

};

int ListaCircularDoble::Contar()

{

Nodo\* aux = inicio;

int cantidad = 0;

if (inicio == NULL)

{

return cantidad;

}

do

{

cantidad++;

aux = aux->next;

} while (aux != inicio);

return cantidad;

}

string ListaCircularDoble::Mostrar()

{

Nodo\* aux = inicio;

string resultado = "";

if (inicio == NULL)

{

resultado = "No se han agregado datos para mostrar";

}

else

{

do

{

resultado += "\nNombre:\t\t" + string(aux->persona.nombre) + "\nApellido:\t" + string(aux->persona.apellido) + "\n\nEdad:\t\t" + to\_string(aux->persona.edad) + "\n\nRegistro:\t" + to\_string(aux->persona.registro) + "\n\n";

aux = aux->next;

} while (aux != inicio);

}

return resultado;

};

void ListaCircularDoble::Iterador()

{

if (inicio == NULL)

{

cout << "No sean agregado datos para que camine el iterador" << endl;

return;

}

char R = NULL;

while (R != 's')

{

cout << "¿Que deceas hacer?" << endl;

cout << "\n\t[ a ]\tMostar iterador\n\t[ b ]\tInicio\n\t[ c ]\tUltimo\n\t[ d ]\tSiguiente\n\t[ e ]\tPrevio\n\t[ f ]\tDireccion del iterador\n\t[ s ]\tSalir del sub menu" << endl;

cin >> R;

switch (R)

{

case('a'): //Mostar iterador

if (iterador != NULL)

cout << "El iterador contiene:" << endl;

cout << "\nNombre:\t" << iterador->persona.nombre << "\nApellido:\t" << iterador->persona.apellido << "\n\nEdad:\t" << iterador->persona.edad << "\n\nRegistro:\t" << iterador->persona.registro << "\n\n" << endl;

break;

case('b'): //Inicio

{

iterador = inicio;

cout << "Movido al inicio y mostrando:" << endl;

cout << "\nNombre:\t" << iterador->persona.nombre << "\nApellido:\t" << iterador->persona.apellido << "\n\nEdad:\t" << iterador->persona.edad << "\n\nRegistro:\t" << iterador->persona.registro << "\n\n" << endl;

break;

}

case('c'): //Ultimo

{

iterador = ultimo;

cout << "Movido al ultimo y mostrando:" << endl;

cout << "\nNombre:\t" << iterador->persona.nombre << "\nApellido:\t" << iterador->persona.apellido << "\n\nEdad:\t" << iterador->persona.edad << "\n\nRegistro:\t" << iterador->persona.registro << "\n\n" << endl;

break;

}

case('d'): //Siguiente

{

cout << "Movido al siguiente y mostrando:" << endl;

if (iterador == ultimo)

iterador = inicio;

else

iterador = iterador->next;

cout << "\nNombre:\t" << iterador->persona.nombre << "\nApellido:\t" << iterador->persona.apellido << "\n\nEdad:\t" << iterador->persona.edad << "\n\nRegistro:\t" << iterador->persona.registro << "\n\n" << endl;

break;

}

case('e'): //Previo

{

cout << "Movido al previo y mostrando:" << endl;

if (iterador == inicio)

iterador = ultimo;

else

iterador = iterador->prev;

cout << "\nNombre:\t" << iterador->persona.nombre << "\nApellido:\t" << iterador->persona.apellido << "\n\nEdad:\t" << iterador->persona.edad << "\n\nRegistro:\t" << iterador->persona.registro << "\n\n" << endl;

break;

}

case('f'): //Direccion del iteradorS

{

cout << "El iterador se ubica en la direccion" << iterador << endl;

break;

}

default:

break;

}

}

return;

}

**ListaCircularDoble.h**

#pragma once

#include <iostream>

#include "Nodo.h"

class ListaCircularDoble

{

Nodo\* iterador;

Nodo\* inicio;

Nodo\* ultimo;

public:

char R;

string resultado = "";

void Agregar(Persona persona);

string Eliminar(Persona persona);

string Buscar(Persona persona);

void Modificar(Persona persona);

int Contar();

string Mostrar();

void Iterador();

ListaCircularDoble()

{

inicio = NULL;

iterador = NULL;

ultimo = NULL;

}

};

**Nodo.cpp**

#include "Nodo.h"

**Nodo.h**

#pragma once

#include <iostream>

#include <string>

using namespace std;

struct Persona

{

char nombre[50];

char apellido[50];

int edad = NULL;

int registro = NULL;

};

class Nodo

{

public:

Persona persona;

Nodo\* next;

Nodo\* prev;

Nodo():next(NULL), prev(NULL) {};

};