**Aplicaciones**

1. **Aplicaciones de Listas**

Debido a la naturaleza estudiada de las listas, como lo es, el uso de memoria dinámica, es decir, ampliar y liberar memoria deliberadamente en ejecución, su aplicación está inmersa en diversas prácticas, como pilas y colas, y programas como listas de alumnos, listas de clientes, listas de espera, listas de distribución de correo y muchas más.

1. **Implementación de Lista Doble en un Listado de Estudiantes**

Estudiante.h

#pragma once

#include <iostream>

**class** **Estudiante**{

**private:**

std::string id;

std::string primer\_nombre;

std::string segundo\_nombre;

std::string primer\_apellido;

std::string segundo\_apellido;

**public:**

Estudiante();

Estudiante(std::string, std::string, std::string, std::string, std::string);

**void** **set\_primer\_nombre**(std::string);

std::string get\_primer\_nombre();

**void** **set\_segundo\_nombre**(std::string);

std::string get\_segundo\_nombre();

**void** **set\_primer\_apellido**(std::string);

std::string get\_primer\_apellido();

**void** **set\_segundo\_apellido**(std::string);

std::string get\_segundo\_apellido();

**void** **set\_id**(std::string);

std::string get\_id();

std::string getter\_general(**int**);

**void** **to\_string**();

};

Estudiante.cpp

#include "Estudiante.h"

#include <iomanip>

Estudiante::Estudiante(){

}

Estudiante::Estudiante(std::string p\_nombre, std::string s\_nombre, std::string p\_apellido, std::string s\_apellido, std::string \_ID){

primer\_nombre = p\_nombre;

segundo\_nombre = s\_nombre;

primer\_apellido = p\_apellido;

segundo\_apellido = s\_apellido;

id = \_ID;

}

**void** Estudiante::set\_primer\_nombre(std::string p\_nombre){

primer\_nombre = p\_nombre;

}

std::string Estudiante::get\_primer\_nombre(){

**return** primer\_nombre;

}

**void** Estudiante::set\_segundo\_nombre(std::string s\_nombre){

segundo\_nombre = s\_nombre;

}

std::string Estudiante::get\_segundo\_nombre(){

**return** segundo\_nombre;

}

**void** Estudiante::set\_primer\_apellido(std::string p\_apellido){

primer\_apellido = p\_apellido;

}

std::string Estudiante::get\_primer\_apellido(){

**return** primer\_apellido;

}

**void** Estudiante::set\_segundo\_apellido(std::string s\_apellido){

segundo\_apellido = s\_apellido;

}

std::string Estudiante::get\_segundo\_apellido(){

**return** segundo\_apellido;

}

**void** Estudiante::set\_id(std::string \_id){

id = \_id;

}

std::string Estudiante::get\_id(){

**return** id;

}

std::string Estudiante::getter\_general(**int** opcion){

**if**(opcion == **1**){

get\_primer\_nombre();

}

**else** **if**(opcion == **2**){

get\_segundo\_nombre();

}

**else** **if**(opcion == **3**){

get\_primer\_apellido();

}

**else** **if**(opcion == **4**){

get\_segundo\_apellido();

}

**else** **if**(opcion == **5**){

get\_id();

}

}

**void** Estudiante::to\_string(){

std::cout << std::setw(**10**) << id.c\_str() << " " <<

std::setw(**20**) << primer\_nombre.c\_str() << " " <<

std::setw(**20**) << segundo\_nombre.c\_str() << " " <<

std::setw(**20**) << primer\_apellido.c\_str() << " " <<

std::setw(**20**) << segundo\_apellido.c\_str() << std::endl;

}

Nodo.h

#pragma once

#include "Estudiante.h"

**class** **Nodo**

{

**public:**

Nodo(Estudiante\* \_valor, Nodo\* \_siguiente, Nodo\* \_anterior);

Nodo();

~Nodo();

Estudiante\* **get\_valor**();

**void** **set\_valor**(Estudiante\* new\_valor);

Nodo\* **get\_siguiente**();

**void** **set\_siguiente**(Nodo\* new\_siguiente);

Nodo\* **get\_anterior**();

**void** **set\_anterior**(Nodo\* new\_anterior);

**protected:**

**private:**

Estudiante\* valor;

Nodo\* siguiente;

Nodo\* anterior;

};

Nodo.cpp

#include "Nodo.h"

Nodo::Nodo(Estudiante\* \_valor, Nodo\* \_siguiente, Nodo\* \_anterior)

{

valor = \_valor;

siguiente = \_siguiente;

anterior = \_anterior;

}

Nodo::Nodo()

{

}

Nodo::~Nodo()

{

}

Estudiante\* Nodo::get\_valor()

{

**return** valor;

}

**void** Nodo::set\_valor(Estudiante\* new\_valor)

{

valor = new\_valor;

}

Nodo\* Nodo::get\_siguiente()

{

**return** siguiente;

}

**void** Nodo::set\_siguiente(Nodo\* new\_siguiente)

{

siguiente = new\_siguiente;

}

Nodo\* Nodo::get\_anterior()

{

**return** anterior;

}

**void** Nodo::set\_anterior(Nodo\* new\_anterior)

{

anterior = new\_anterior;

}

ListaDoble.h

#pragma once

#include "Nodo.h"

**class** **ListaDoble**

{

**public:**

ListaDoble(Nodo\* \_cabeza, Nodo\* \_cola);

ListaDoble();

Nodo\* **get\_cabeza**();

**void** **set\_cabeza**(Nodo\* \_cabeza);

Nodo\* **get\_cola**();

**void** **set\_cola**(Nodo\* \_cola);

**void** **insertar\_cabeza**(Estudiante\* valor);

**void** **insertar\_cola**(Estudiante\* valor);

**void** **insertar\_entre**(std::string valor\_buscado, Estudiante\* valor, **int** opcion);

**void** **eliminar**(std::string valor);

Estudiante\* **buscar**(std::string valor\_buscado);

**void** **imprimir**();

**bool** **esta\_vacia**();

**protected:**

**private:**

Nodo\* cabeza;

Nodo\* cola;

};

ListaDoble.cpp

#include <iostream>

#include "ListaDoble.h"

#include <iomanip>

ListaDoble::ListaDoble(Nodo\* \_cabeza, Nodo\* \_cola){

cabeza = \_cabeza;

cola = \_cola;

}

ListaDoble::ListaDoble(){

cabeza = nullptr;

cola = nullptr;

}

Nodo\* ListaDoble::get\_cabeza(){

**return** cabeza;

}

**void** ListaDoble::set\_cabeza(Nodo\* nueva\_cabeza){

cabeza = nueva\_cabeza;

}

Nodo\* ListaDoble::get\_cola(){

**return** cola;

}

**void** ListaDoble::set\_cola(Nodo\* nueva\_cola){

cola = nueva\_cola;

}

**void** ListaDoble::insertar\_cabeza(Estudiante\* valor){

Nodo\* nuevo\_nodo;

**if**(esta\_vacia()){

nuevo\_nodo = **new** Nodo(valor, nullptr, nullptr);

cola = nuevo\_nodo;

}

**else**{

nuevo\_nodo = **new** Nodo(valor, cabeza, nullptr);

cabeza->set\_anterior(nuevo\_nodo);

}

cabeza = nuevo\_nodo;

}

**void** ListaDoble::insertar\_cola(Estudiante\* valor){

Nodo\* nuevo\_nodo;

**if**(esta\_vacia()){

nuevo\_nodo = **new** Nodo(valor, nullptr, nullptr);

cabeza = nuevo\_nodo;

}

**else**{

nuevo\_nodo = **new** Nodo(valor, nullptr, cola);

cola->set\_siguiente(nuevo\_nodo);

}

cola = nuevo\_nodo;

}

**void** ListaDoble::insertar\_entre(std::string valor\_buscado, Estudiante\* valor, **int** opcion){

Nodo\* aux = cabeza;

**while**(aux!=nullptr){

**if**(valor\_buscado == aux->get\_valor()->getter\_general(opcion)){

Nodo\* nuevo = **new** Nodo(valor,aux->get\_siguiente(), aux);

**if**(aux != cola && aux->get\_siguiente()!= nullptr){

aux->get\_siguiente()->set\_anterior(nuevo);

}**else**{

cola = nuevo;

}

aux->set\_siguiente(nuevo);

**break**;

}

aux = aux->get\_siguiente();

}

}

**void** ListaDoble::imprimir(){

Nodo\* aux = cabeza;

std::cout << std::setw(**10**) << "ID" << " " <<

std::setw(**20**) << "Primer Nombre" << " " <<

std::setw(**20**) << "Segundo Nombre" << " " <<

std::setw(**20**) << "Primer Apellido" << " " <<

std::setw(**20**) << "Segundo Apellido" << std::endl;

**while**(aux!=nullptr){

aux->get\_valor()->to\_string();

aux = aux->get\_siguiente();

}

}

**void** ListaDoble::eliminar(std::string valor){

Nodo\* aux = cabeza;

Nodo\* temp;

**while**(aux!=nullptr){

**if**( valor == aux->get\_valor()->get\_id() ){

**if**(cabeza == cola){

cabeza = nullptr;

cola = nullptr;

}

**else**{

**if**(aux->get\_anterior() == nullptr){

temp = aux->get\_siguiente();

aux->set\_siguiente(nullptr);

cabeza = temp;

cabeza->set\_anterior(nullptr);

}

**else** **if**(aux->get\_siguiente() == nullptr){

temp = aux->get\_anterior();

aux->set\_anterior(nullptr);

cola = temp;

cola->set\_siguiente(nullptr);

}

**else**{

aux->get\_anterior()->set\_siguiente(aux->get\_siguiente());

aux->get\_siguiente()->set\_anterior(aux->get\_anterior());

}

}

**delete** aux;

**break**;

}

aux = aux->get\_siguiente();

}

**if**(aux == nullptr){

printf("ID de estudiante no encontrado.");

}

}

Estudiante\* ListaDoble::buscar(std::string valor\_buscado){

Nodo\* aux = cabeza;

**while**(aux!=nullptr){

**if**( valor\_buscado == aux->get\_valor()->get\_id() ){

**return** aux->get\_valor();

}

aux = aux->get\_siguiente();

}

**if**(aux == nullptr){

**return** nullptr;

}

}

**bool** ListaDoble::esta\_vacia(){

**return** (cabeza == nullptr);

}

Main.cpp

#include <iostream>

#include "Nodo.h"

#include "ListaDoble.h"

#include "windows.h"

#include "validaciones.h"

**void** **mostrar\_menu\_principal**(**int** opcion){

system("cls");

std::cout << "**\n**--------------LISTADO ESTUDIANTIL-----------------**\n**";

std::cout << "**\n**Seleccione una opcion:**\n\n**";

std::cout << (opcion == **1** ? "> " : " ") << "Anadir estudiantes**\n**";

std::cout << (opcion == **2** ? "> " : " ") << "Eliminar estudiantes**\n**";

std::cout << (opcion == **3** ? "> " : " ") << "Buscar estudiantes**\n**";

std::cout << (opcion == **4** ? "> " : " ") << "Mostrar estudiantes**\n**";

std::cout << (opcion == **5** ? "> " : " ") << "Salir**\n**";

}

**int** **main**()

{

Estudiante\* nuevo\_estudiante;

ListaDoble lista\_estudiantes;

**int** parametro;

**bool** hallado;

std::string dato;

std::string aparametro;

**int** opcion = **1**;

**char** tecla;

HANDLE consoleHandle = GetStdHandle(STD\_OUTPUT\_HANDLE);

CONSOLE\_CURSOR\_INFO cursorInfo;

GetConsoleCursorInfo(consoleHandle, &cursorInfo);

cursorInfo.bVisible = false;

SetConsoleCursorInfo(consoleHandle, &cursorInfo);

**do**{

mostrar\_menu\_principal(opcion);

tecla = \_getch();

**switch**(tecla){

**case** **72**: // Flecha arriba

**if** (opcion > **1**) opcion--;

**break**;

**case** **80**: // Flecha abajo

**if** (opcion < **5**) opcion++;

**break**;

**case** **13**: // Enter

**if**(opcion == **1**){

cursorInfo.bVisible = true;

SetConsoleCursorInfo(consoleHandle, &cursorInfo);

nuevo\_estudiante = **new** Estudiante();

dato = ingresar\_nombre("Ingrese el primer nombre del estudiante:");

nuevo\_estudiante->set\_primer\_nombre(dato);

printf("**\n**");

dato = ingresar\_nombre("Ingrese el segundo nombre del estudiante (si no tiene, escriba 'ninguno'):");

nuevo\_estudiante->set\_segundo\_nombre(dato);

printf("**\n**");

dato = ingresar\_nombre("Ingrese el primer apellido del estudiante:");

nuevo\_estudiante->set\_primer\_apellido(dato);

printf("**\n**");

dato = ingresar\_nombre("Ingrese el segundo apellido del estudiante (si no tiene, escriba 'ninguno'):");

nuevo\_estudiante->set\_segundo\_apellido(dato);

printf("**\n**");

nuevo\_estudiante->set\_id(generarEtiqueta());

lista\_estudiantes.insertar\_cola(nuevo\_estudiante);

printf("**\n**");

system("pause");

cursorInfo.bVisible = false;

SetConsoleCursorInfo(consoleHandle, &cursorInfo);

}

**else** **if**(opcion == **2**){

cursorInfo.bVisible = true;

SetConsoleCursorInfo(consoleHandle, &cursorInfo);

lista\_estudiantes.imprimir();

printf("Ingrese el ID del estudiante:**\n**");

std::cin >> dato;

lista\_estudiantes.eliminar(dato);

printf("**\n**");

system("pause");

cursorInfo.bVisible = false;

SetConsoleCursorInfo(consoleHandle, &cursorInfo);

}

**else** **if**(opcion == **3**){

cursorInfo.bVisible = true;

SetConsoleCursorInfo(consoleHandle, &cursorInfo);

printf("Ingrese el ID del estudiante:**\n**");

std::cin >> dato;

nuevo\_estudiante = lista\_estudiantes.buscar(dato);

**if**(nuevo\_estudiante != nullptr){

printf("Se encontro el estudiante.**\n**");

printf("Con la informacion: %s %s %s %s**\n**", nuevo\_estudiante->get\_primer\_nombre().c\_str(),

nuevo\_estudiante->get\_segundo\_nombre().c\_str(),

nuevo\_estudiante->get\_primer\_apellido().c\_str(),

nuevo\_estudiante->get\_segundo\_apellido().c\_str());

}**else**{

printf("No se hallo dicho estudiante.**\n**");

}

system("pause");

cursorInfo.bVisible = false;

SetConsoleCursorInfo(consoleHandle, &cursorInfo);

}

**else** **if**(opcion == **4**){

lista\_estudiantes.imprimir();

system("pause");

}

**else** **if**(opcion == **5**){

system("cls");

std::cout << "Saliendo del programa." << std::endl;

exit(**0**);

}

**break**;

**default:**

**break**;

}

}**while**(true);

printf("**\n**");

system("pause");

cursorInfo.bVisible = true;

SetConsoleCursorInfo(consoleHandle, &cursorInfo);

}

1. **Referencias**

Ceballos, F. (2020) C/C++ Curso de programación. 5ª. Edición.