Corporacion Univeritaria Minuto de Dios

UNIMINUTO

Asignatura:

Estructura de Datos

NRC :7233

Tema

Lista en C++

Presentado:

Michael Daniel Murillo LópezID:534830

DOCENTE:

Segundo Fidel Puerto Garavito

Colombia Ciudad De Bogotá
D.C septiembre 27 de 2017

Se Necesita crear una Agenda en donde se necesita los siguientes requisitos
La agenda debe agregar Nombre, Apellido, Correo Electrónico, Teléfono, Celular
La agenda debe modificar un dato especifico o todos.
La agenda debe eliminar un contacto especifico.
La agenda debe ver un contacto especifico o todos.
El archivo se debe guardar en un documento de texto. Los Datos deben ser los que se piden en el momento de agregar.

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
/**
 *
 * @author Michael Daniel Murillo López
 * Estrutura de Datos
 * ID:534830 Corporación
 * Universitaria Minuto de Dios - UNIMINUTO
 * NRC:7273
 */
#include <stdlib.h>
```

```
#include <iostream>
#include <fstream>
#include <string.h>
using namespace std;
//Prototipo de funciones.
void Anadir();
void Eliminar();
void Editar();
void Ver();
void VerTodo();
void Informacion();
//Definicion de estructura.
struct informacion {
    char cedula [25];
    char nombre[25];
    char apellido[25];
    char correo[100];
    char direccion[100];
    char telefono[25];
    char celular[25];
}
persona[25];
//Declaracion de variables.
int opcion;
int registro = 0;
char sn[10];
int main() {
    system("Color 2F");
   cout « "________________________________________________
« endl « endl;
    cout « " __^__                                   __^__" « endl;
    cout « "( ___ )---------------------------------( ___ )" « endl;
    cout « " |   |                                 | |" « endl;
    cout « " | * |        Agenda en                 | * |" « endl;
    cout « " | * |             C++                 | * |" « endl;
    cout « " |___|                                  |___|" « endl;
     cout « "(_____)-------------------------------(_____)" « endl «
endl;
    cout « "          1. Anadir un contacto." « endl « endl;
    cout « "          2. Eliminar un contacto." « endl « endl;
    cout « "          3. Editar un contacto." « endl « endl;
    cout « "          4. Ver informacion de un contacto." « endl « endl;
    cout « "          5. Ver todos los contactos." « endl « endl;
    cout « "          6. Ver registro en Agenda.txt" « endl « endl;
    cout « "          7. Informacion « " « endl « endl;
```

```
    cout « "          8. Salir." « endl « endl;
  cout « "_____
« endl « endl;
    cout « "Numero de operacion a realizar: ";
    cin » opcion;
    switch (opcion) {
        case 1:
            Anadir();
            break;
        case 2:
            Eliminar();
            break;
        case 3:
            Editar();
            break;
        case 4:
            Ver();
            break;
        case 5:
            VerTodo();
            break;
        case 6:
            system("Agenda.txt");
            main();
            break;
        case 7:
            informacion();
            break;
        case 8:
            exit(0);
            break;
        default:
          cout « "Operacion incorrecta. Escriba nuevamente la operacion
a realizar";
            main();
            break;
    }
    return 0;
}
//Fin main.
^
```

Con las listas tendremos un pequeño repertorio de operaciones básicas que se pueden realizar:

- Añadir o insertar elementos.

- Buscar o localizar elementos.
- Borrar elementos.
- Moverse a través de una lista, anterior, siguiente, primero.

Cada una de estas operaciones tendrá varios casos especiales, por ejemplo, no será lo mismo insertar un nodo en una lista vacía, o al principio de una lista no vacía, o la final, o en una posición intermedia.

```
//Definicion de funciones.
void Anadir() {
    cout « "Escriba la siguiente informacion del contacto.";
    cout « "1. Cedula: ";
    cin » persona[registro].cedula;
    cout « "2. Nombre: ";
    cin » persona[registro].nombre;
    cout « "3. Apellido: ";
    cin » persona[registro].apellido;
    cout « "4. Fecha nacimiento: ";
    cin » persona[registro].correo;
    cout « "5. Direccion: ";
    cin » persona[registro].direccion;
    cout « "6. Telefono: ";
    cin » persona[registro].telefono;
    cout « "7. Celular: ";
    cin » persona[registro].celular;
    {
        ////// Entrada/Salida a archivo de texto "Agenda.txt"
        ofstream archivo;
        archivo.open("Agenda.txt", ios::app);
        archivo « "Cedula: " « persona[registro].cedula « endl;
        archivo « "Nombre: " « persona[registro].nombre « endl;
        archivo « "Apellido: " « persona[registro].apellido « endl;
       archivo « "Correo Electronico: " « persona[registro].correo « endl;
        archivo « "Direccion: " « persona[registro].direccion « endl;
        archivo « "Telefono: " « persona[registro].telefono « endl;
        archivo « "Celular: " « persona[registro].celular « endl;
        archivo.close();
        ////// Fin de Entrada/Salida a archivo de texto "Agenda.txt"
        registro++;
        cout « endl « "Contacto anadido." « endl;
        main();
    }
}
void Eliminar() {
```

```
    int opcionEliminar;
    cout « "Numero de registro a eliminar: ";
    cin » opcionEliminar;
    opcionEliminar–;
    if (opcionEliminar < registro) {
        cout « "1. Cedula: ";
        cout « persona[opcionEliminar].cedula;
        cout « "2. Nombre: ";
        cout « persona[opcionEliminar].nombre;
        cout « "3. Apellido: ";
        cout « persona[opcionEliminar].apellido;
        cout « "4. Correo Electronico ";
        cout « persona[opcionEliminar].correo;
        cout « "5. Direccion: ";
        cout « persona[opcionEliminar].direccion;
        cout « "6. Telefono: ";
        cout « persona[opcionEliminar].telefono;
        cout « "7. Celular: ";
        cout « persona[opcionEliminar].celular;
        do {
            cout « "Eliminar contacto de la agenda? [S/N]: ";
            cin » sn;
            if ((strcmp(sn, "s") == 0) || (strcmp(sn, "S") == 0)) {
                persona[opcionEliminar] = persona[registro];
                cout « endl « "Contacto eliminado." « endl;
                main();
            }
            if ((strcmp(sn, "n") == 0) || (strcmp(sn, "N") == 0)) {
                cout « "Contacto no eliminado.";
                main();
            }
        } while ((sn != "S") || (sn != "s") || (sn != "N") || (sn != "n"));
    }
    else {
        cout « "Numero de registro no existente.";
        +
        main();
    }
}
void Editar() {
    int opcionEditar;
    int informacionEditar;
    cout « "Numero de registro a editar: ";
    cin » opcionEditar;
    opcionEditar–;
```

```
if (opcionEditar < registro) {
    cout « "1. Cedula: ";
    cout « persona[opcionEditar].cedula;
    cout « "2. Nombre: ";
    cout « persona[opcionEditar].nombre;
    cout « "3. Apellido: ";
    cout « persona[opcionEditar].apellido;
    cout « "4. Correo Electronico ";
    cout « persona[opcionEditar].correo;
    cout « "5. Direccion: ";
    cout « persona[opcionEditar].direccion;
    cout « "6. Telefono: ";
    cout « persona[opcionEditar].telefono;
    cout « "7. Celular: ";
    cout « persona[opcionEditar].celular;
    cout « "8. Todo. ";
    cout « "9. Nada. ";
    cout « "Informacion a editar: ";
    cin » informacionEditar;
    switch (informacionEditar) {
        case 1:
            cout « "1. Cedula: ";
            cin » persona[opcionEditar].cedula;
            break;
        case 2:
            cout « "2. Nombre: ";
            cin » persona[opcionEditar].nombre;
            break;
        case 3:
            cout « "3. Apellido: ";
            cin » persona[opcionEditar].apellido;
            break;
        case 4:
            cout « "4. Correo Electronico ";
            cin » persona[opcionEditar].correo;
            break;
        case 5:
            cout « "5. Direccion: ";
            cin » persona[opcionEditar].direccion;
            break;
        case 6:
            cout « "6. Telefono: ";
            cin » persona[opcionEditar].telefono;
            break;
        case 7:
```

```
                cout « "7. Celular: ";
                cin » persona[opcionEditar].celular;
                break;
            case 8:
                cout « "1. Cedula: ";
                cin » persona[opcionEditar].cedula;
                cout « "2. Nombre: ";
                cin » persona[opcionEditar].nombre;
                cout « "3. Apellido: ";
                cin » persona[opcionEditar].apellido;
                cout « "4. Correo Electronico  ";
                cin » persona[opcionEditar].correo;
                cout « "5. Direccion: ";
                cin » persona[opcionEditar].direccion;
                cout « "6. Telefono: ";
                cin » persona[opcionEditar].telefono;
                cout « "7. Celular: ";
                cin » persona[opcionEditar].celular;
                break;
            case 9:
                main();
                break;
            default:
                    cout « "Operacion incorrecta.  Escriba nuevamente la
informacion a editar.";
                Editar();
                break;
        }
        main();
    }
    else {
        cout « "Numero de registro no existente.";
        main();
    }
}
void Ver() {
    int ver;
    cout « "Numero de registro a ver: ";
    cin » ver;
    ver–;
    if (ver < registro) {
        cout « "\n1. Cedula: ";
        cout « persona[ver].cedula;
        cout « "\n2. Nombre: ";
        cout « persona[ver].nombre;
```

```
        cout « "\n3. Apellido: ";
        cout « persona[ver].apellido;
        cout « "\n4. Correo Eletronico ";
        cout « persona[ver].correo;
        cout « "\n5. Direccion: ";
        cout « persona[ver].direccion;
        cout « "\n6. Telefono: ";
        cout « persona[ver].telefono;
        cout « "\n7. Celular: ";
        cout « persona[ver].celular;
        main();
    }
    else {
        cout « "Numero de registro no existente.";
        main();
    }
}
void VerTodo() {
    int verTodo;
    for (verTodo = 0; verTodo < registro; verTodo++) {
        cout « "\n1. Cedula: ";
        cout « persona[verTodo].cedula;
        cout « "\n2. Nombre: ";
        cout « persona[verTodo].nombre;
        cout « "\n3. Apellido: ";
        cout « persona[verTodo].apellido;
        cout « "\n4. Correo Electronico ";
        cout « persona[verTodo].correo;
        cout « "\n5. Direccion: ";
        cout « persona[verTodo].direccion;
        cout « "\n6. Telefono: ";
        cout « persona[verTodo].telefono;
        cout « "\n7. Celular: ";
        cout « persona[verTodo].celular;
        system("pause > nul");
    }
    main();
}
void informacion() {
    cout « "                              " « endl;
    cout « "Michael Daniel Murillo López" « endl;
    cout « "          Id :534830          " « endl;
    cout « "                              " « endl;
    cout « " Corporación Universitaria  " « endl;
    cout « "       Minuto de Dios         " « endl;
```

```
    cout « "                    " « endl;
    cout « "                    " « endl;
}
```

Una lista es un tipo de dato autor referenciado porque contienen un puntero o enlace (en inglés link, del mismo significado) a otro dato del mismo tipo. Las listas enlazadas permiten inserciones y eliminación de nodos en cualquier punto de la lista en tiempo constante (suponiendo que dicho punto está previamente identificado o localizado), pero no permiten un acceso aleatorio

Ventajas

Como muchas opciones en programación y desarrollo, no existe un único método correcto para resolver un problema. Una estructura de lista enlazada puede trabajar bien en un caso, pero causar problemas en otros. He aquí una lista con algunas de las ventajas más comunes que implican las estructuras de tipo lista. En general, teniendo una colección dinámica donde los elementos están siendo añadidos y eliminados frecuentemente e importa la localización de los nuevos elementos introducidos se incrementa el beneficio de las listas