MENU DE CADENA DE RESTAURANTES

Integrantes:
Andrés Oto
Mathías Borja
David Pilatuña

27 DE ENERO DE 2026

1. Introducción

Este documento describe el desarrollo, funcionamiento y posibles modificaciones del proyecto CRUD Cadenas de Cadena de Restaurantes desarrollado en **C++**. El sistema permite gestionar cadena de restaurantes mediante las operaciones básicas: Crear, Leer, Actualizar y Eliminar, facilitando la administración de productos o platillos de manera ordenada.

El proyecto fue diseñado con fines académicos, aplicando conceptos fundamentales de programación estructurada, uso de estructuras, vectores y control de flujo.

---

2. Objetivos

Objetivo General

Desarrollar un sistema CRUD en C++ que permita administrar una cadena de restaurante de manera eficiente mediante consola.

Objetivos Específicos

- Implementar estructuras para representar los datos de la cadena de restaurante.
- Aplicar las operaciones CRUD (Crear, Leer, Actualizar, Eliminar).
- Utilizar vectores para el almacenamiento dinámico de datos.
- Manejar entradas y salidas por consola.
- Facilitar futuras modificaciones y mejoras al sistema.

---

3. Alcance del Proyecto

El sistema permite:

- Registrar nuevos cadena de restaurantes.
- Mostrar toda la cadena de restaurantes registrados.
- Modificar información de una cadena de restaurante existente.
- Eliminar cadena de restaurantes mediante su identificador.

---

4. Tecnologías Utilizadas

- Lenguaje de programación: **C++**
- Entorno de desarrollo: Dev-C++
- Librerías estándar:
  - `<iostream>`
  - `<vector>`
  - `<string>`

---

5. Estructura del Sistema

5.1 Estructura Cadena Restaurante

La estructura `Cadena Restaurante` representa cada elemento del sistema:

- **id**: Identificador único del cadena de restaurante.
- **nombre**: Nombre del platillo o producto.
- **descripcion**: Descripción del cadena de restaurante.
- **categoria**: Tipo de cadena de restaurante (bebida, comida, postre, etc.).
- **precio**: Costo del cadena de restaurante.

Esta estructura incluye una función para mostrar la información por pantalla.

---

6. Funcionalidades del CRUD

6.1 Crear (Create)

Permite registrar una nueva cadena de restaurante solicitando los datos al usuario y almacenándolos en un vector.

```
Menu agregarMenu() {
    Menu nuevoMenu;
    cout << "Ingrese el ID del menú: ";
    cin >> nuevoMenu.id;

    for (const auto& menu : listaMenus) {
        if (nuevoMenu.id == menu.id) {
            cout << "\nEl menú ya existe (ID duplicado).\n";
            return {};
        }
    }
    cin.ignore();
    cout << "Ingrese el nombre del menú: ";
    getline(cin, nuevoMenu.nombre);
    cout << "Ingrese la descripción del menú: ";
    getline(cin, nuevoMenu.descripcion);
    cout << "Ingrese la categoría del menú: ";
    getline(cin, nuevoMenu.categoria);
    cout << "Ingrese el precio del menú: ";
    cin >> nuevoMenu.precio;
    cout<<"\nMenu Registrado Con Éxito.\n";
    return nuevoMenu;
}
```

6.2 Leer (Read)

Muestra toda la cadena de restaurantes registrados en el sistema recorriendo el vector.

```
void leerMenu() {
    if (listaMenus.empty()) {
        cout << "\nNo hay menús registrados\n";
    } else {
        cout << "\n*=*=*=*=*=*=*=*=*=* LISTA DE MENÚS REGISTRADOS *=*=*=*=*=*=*=*=*=*\n";
        for (const auto& m : listaMenus) {
            cout <<"ID: "<< m.id << "\nNombre: " << m.nombre << "\nDescripción: "
                 << m.descripcion << "\nCategoria: " << m.categoria
                 << "\nPrecio: $" << m.precio << endl;
        }
    }
}
```

6.3 Actualizar (Update)

Permite modificar los datos de una cadena de restaurante existente utilizando el ID como referencia.

```
void actualizar() {
    vector<Menu> menus = cargarMenu(); // cargar desde archivo
    int idBuscar;
    bool encontrado = false;

    cout << "\nIngrese el ID del menú a actualizar: ";
    cin >> idBuscar;
    cin.ignore();

    for (Menu &m : menus) {
        if (m.id == idBuscar) {
            cout << "\n=== MENÚ ENCONTRADO ===\n";
            cout << "ID: " << m.id << endl;
            cout << "Nombre actual: " << m.nombre << endl;
            cout << "Descripción actual: " << m.descripcion << endl;
            cout << "Categoría actual: " << m.categoria << endl;
            cout << "Precio actual: $" << m.precio << endl;

            cout << "\nIngrese el nuevo nombre: ";
            getline(cin, m.nombre);

            cout << "Ingrese la nueva descripción: ";
            getline(cin, m.descripcion);

            cout << "Ingrese la nueva categoría: ";
            getline(cin, m.categoria);

            cout << "Ingrese el nuevo precio: ";
            cin >> m.precio;

            encontrado = true;
            break;
        }
    }

    if (!encontrado) {
        cout << "\nNo se encontró un menú con ese ID\n";
        return;
    }

    // Reescribir archivo con datos actualizados
    ofstream archivo(ruta);
    if (!archivo) {
        cerr << "\nError al abrir el archivo para actualizar.\n";
        return;
    }

    for (const auto& m : menus) {
        archivo << m.id << "|"
                << m.nombre << "|"
                << m.descripcion << "|"
                << m.categoria << "|"
                << m.precio << "\n";
    }

    cout << "\nMENÚ ACTUALIZADO CORRECTAMENTE\n";
```

6.4 Eliminar (Delete)

Elimina una cadena de restaurante del sistema mediante su ID, actualizando el vector.

```
void eliminarMenu() {
    vector<Menu> menus = cargarMenu();
    int idEliminar;
    bool encontrado = false;

    cout << "\nIngrese el ID del menú a eliminar: ";
    cin >> idEliminar;

    auto it = remove_if(menus.begin(), menus.end(),
                        [idEliminar](const Menu& m) {
                            return m.id == idEliminar;
                        });

    if (it != menus.end()) {
        menus.erase(it, menus.end());
        encontrado = true;
    }

    if (!encontrado) {
        cout << "\nNo se encontró un menú con ese ID\n";
        return;
    }

    ofstream archivo(ruta);
    if (!archivo) {
        cerr << "\nError al abrir el archivo para eliminar\n";
        return;
    }

    for (const auto& m : menus) {
        archivo << m.id << "|"
                << m.nombre << "|"
                << m.descripcion << "|"
                << m.categoria << "|"
                << m.precio << "\n";
    }

    listaMenus = menus;
    cout << "\nMENÚ ELIMINADO CORRECTAMENTE\n";
}
```

---

7. Flujo del Programa

1. Mostrar cadena de restaurante principal.
2. Solicitar opción al usuario.
3. Ejecutar la operación seleccionada.
4. Retornar a la cadena de restaurante principal hasta que el usuario decida salir.

---

8. Validaciones Implementadas

- Verificación de ID existente.
- Control de opciones inválidas de la cadena de restaurante.
- Prevención de accesos fuera de rango del vector.

---

9. Posibles Modificaciones y Mejoras

9.1 Modificaciones Simples

- Cambiar nombres de variables.
- Ajustar categorías de la cadena de restaurante.
- Modificar el formato de impresión.
- Cambiar el tipo de dato del precio

9.2 Mejoras Intermedias

- Implementar búsqueda por nombre o categoría.
- Ordenar cadena de restaurantes por precio o nombre.

9.3 Mejoras Avanzadas

- Implementar clases y programación orientada a objetos.
- Conectar con una base de datos.
- Implementar roles de usuario

---

10. Link del Repositorio

https://github.com/andrxuy/CRUD-CADENA-DE-RESTAURANTES.git

---

11. Conclusión

El proyecto CRUD de Cadenas de CadenaRestaurantes en C++ cumple con los objetivos planteados, permitiendo aplicar conceptos fundamentales de programación. Además, sirve como base para proyectos más complejos, ya que puede ser ampliado con nuevas funcionalidades y mejoras.

---

12. Anexos

- **PRESENTACIÓN DE DIAPOSITIVAS:**

https://www.canva.com/design/DAG_d_rGUYg/bEQh5Jj5iwu7DwC5nmkmpw/edit?utm_content=DAG_d_rGUYg&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

- **CÓDIGO:**

CÓDIGO .CPP CON ARCHIVOS

```
/* PROYECTO SEGUNDO BIMESTRE "MENUS EN CADENA DE RESTAURANTES"
VERSIÓN DE ARCHIVOS
Andrés Oto
Matías Borja
David Pilatuña*/

#include<iostream>
#include<vector>
#include<string>
#include<algorithm>
#include<fstream>
#include <locale>
using namespace std;
//estructura

struct Menu {
   int id;
   string nombre;
   string descripcion;
   string categoria;
   double precio;
};

vector<Menu> listaMenus;
string ruta = "menus.txt";

vector<Menu> cargarMenu() {
   vector<Menu> menus;
   ifstream archivo(ruta);
   if (!archivo) {
      cerr << "Archivo no encontrado. Se iniciará una nueva lista.\n";
      return menus;
   }

   string linea;
   while (getline(archivo, linea)) {
      Menu m;
      int pos = 0;
      int campo = 0;
      string token;

      while ((pos = linea.find('|')) != string::npos) {
         token = linea.substr(0, pos);
         if (campo == 0) {
            m.id = stoi(token);
         } else if (campo == 1) {
            m.nombre = token;
```

```
            } else if (campo == 2) {
                m.descripcion = token;
            } else if (campo == 3) {
                m.categoria = token;
            }
            linea.erase(0, pos + 1);
            campo++;
        }
        m.precio = stod(linea);

        menus.push_back(m);
    }
    return menus;
}

Menu agregarMenu() {
    Menu nuevoMenu;
    cout << "Ingrese el ID del menú: ";
    cin >> nuevoMenu.id;

    for (const auto& menu : listaMenus) {
        if (nuevoMenu.id == menu.id) {
            cout << "\nEl menú ya existe (ID duplicado).\n";
            return {};
        }
    }
    cin.ignore();
    cout << "Ingrese el nombre del menú: ";
    getline(cin, nuevoMenu.nombre);
    cout << "Ingrese la descripción del menú: ";
    getline(cin, nuevoMenu.descripcion);
    cout << "Ingrese la categoría del menú: ";
    getline(cin, nuevoMenu.categoria);
    cout << "Ingrese el precio del menú: ";
    cin >> nuevoMenu.precio;
          cout<<"\nMenu Registrado Con Éxito.\n";
    return nuevoMenu;
}

void guardarMenu(){
    ofstream archivo(ruta);
    if (!archivo) {
        cerr << "\nError al abrir el archivo para guardar.\n";
        return;
    }
    for (const auto& m : listaMenus) {
        archivo << m.id << "|"
                << m.nombre << "|"
                << m.descripcion << "|"
                << m.categoria << "|"
```

```
            << m.precio << "\n";
    }
}

void buscarMenu() {
    int id;
    cout << "Ingrese el ID del menú a buscar: ";
    cin >> id;
    ifstream archivo(ruta);
    if (!archivo) {
        cerr << "\nNo se pudo abrir el archivo menus.txt\n";
        return;
    }
    string linea;
    while (getline(archivo, linea)) {
        Menu m;
        int pos = 0;
        int campo = 0;
        string token;

        while ((pos = linea.find('|')) != string::npos) {
            token = linea.substr(0, pos);
            if (campo == 0) {
                m.id = stoi(token);
            } else if (campo == 1) {
                m.nombre = token;
            } else if (campo == 2) {
                m.descripcion = token;
            } else if (campo == 3) {
                m.categoria = token;
            }
            linea.erase(0, pos + 1);
            campo++;
        }
        m.precio = stod(linea);
        if (m.id == id) {
            cout     <<     "\n==================     MENÚ     ENCONTRADO
=============== \n";
            cout << "ID: " << m.id << endl;
            cout << "Nombre: " << m.nombre << endl;
            cout << "Descripción: " << m.descripcion << endl;
            cout << "Categoría: " << m.categoria << endl;
            cout << "Precio: $" << m.precio<< endl;
            return;
        }
    }
    cout << "\nMenú no encontrado\n";
}

void leerMenu() {
```

```
    if (listaMenus.empty()) {
        cout << "\nNo hay menús registrados\n";
    } else {
        cout << "\n*=*=*=*=*=*=*=*=*=* LISTA DE MENÚS REGISTRADOS
*=*=*=*=*=*=*=*=*=*=*\n";
        for (const auto& m : listaMenus) {
            cout <<"ID: "<< m.id << "\nNombre: " << m.nombre << "\nDescripción:
"
                << m.descripcion << "\nCategoria: " << m.categoria
                << "\nPrecio: $" << m.precio << endl;
        }
    }
}

//LO DE ABAJO DESDE LA 159 HASTA LA 189, SIGUE SIENDO EN
CONSOLA
void actualizar() {
    vector<Menu> menus = cargarMenu(); // cargar desde archivo
    int idBuscar;
    bool encontrado = false;

    cout << "\nIngrese el ID del menú a actualizar: ";
    cin >> idBuscar;
    cin.ignore();

    for (Menu &m : menus) {
        if (m.id == idBuscar) {
            cout << "\n=== MENÚ ENCONTRADO ===\n";
            cout << "ID: " << m.id << endl;
            cout << "Nombre actual: " << m.nombre << endl;
            cout << "Descripción actual: " << m.descripcion << endl;
            cout << "Categoría actual: " << m.categoria << endl;
            cout << "Precio actual: $" << m.precio << endl;

            cout << "\nIngrese el nuevo nombre: ";
            getline(cin, m.nombre);

            cout << "Ingrese la nueva descripción: ";
            getline(cin, m.descripcion);

            cout << "Ingrese la nueva categoría: ";
            getline(cin, m.categoria);

            cout << "Ingrese el nuevo precio: ";
            cin >> m.precio;

            encontrado = true;
            break;
        }
    }
```

```
    if (!encontrado) {
        cout << "\nNo se encontró un menú con ese ID\n";
        return;
    }

    // Reescribir archivo con datos actualizados
    ofstream archivo(ruta);
    if (!archivo) {
        cerr << "\nError al abrir el archivo para actualizar.\n";
        return;
    }

    for (const auto& m : menus) {
        archivo << m.id << "|"
                << m.nombre << "|"
                << m.descripcion << "|"
                << m.categoria << "|"
                << m.precio << "\n";
    }

    cout << "\nMENÚ ACTUALIZADO CORRECTAMENTE\n";

    // sincroniza la lista en memoria
    listaMenus = menus;
}
void eliminarMenu() {
    vector<Menu> menus = cargarMenu();
    int idEliminar;
    bool encontrado = false;

    cout << "\nIngrese el ID del menú a eliminar: ";
    cin >> idEliminar;

    auto it = remove_if(menus.begin(), menus.end(),
                        [idEliminar](const Menu& m) {
                            return m.id == idEliminar;
                        });

    if (it != menus.end()) {
        menus.erase(it, menus.end());
        encontrado = true;
    }

    if (!encontrado) {
        cout << "\nNo se encontró un menú con ese ID\n";
        return;
    }

    ofstream archivo(ruta);
```

```
    if (!archivo) {
        cerr << "\nError al abrir el archivo para eliminar\n";
        return;
    }

    for (const auto& m : menus) {
        archivo << m.id << "|"
                << m.nombre << "|"
                << m.descripcion << "|"
                << m.categoria << "|"
                << m.precio << "\n";
    }

    listaMenus = menus;
    cout << "\nMENÚ ELIMINADO CORRECTAMENTE\n";
}

//SOLO LA OPCION (case: 1) está con archivos, tomar en cuenta
int main() {
    setlocale(LC_ALL, "");

    listaMenus = cargarMenu();

    int opcion=-1;

    while (true) {
        cout << "\n*=*=*=*=*=*=*=*=*=* APP REGISTRO MENÚS *=*=*=*=*=*=*=*=*=*\n";
        cout << "1. Registrar nuevo menú.\n";
        cout << "2. Buscar un menú.\n";
                cout << "3. Actualizar Menú." << endl;
                cout << "4. Eliminar Menú."<<endl
        cout << "5. Mostrar todos los menús.\n";
        cout << "0. Salir.\n";
        cout << "Seleccione una opción: ";
        cin >> opcion;

        switch (opcion) {
            case 1: {
                Menu nuevo = agregarMenu();
                if (nuevo.id != 0) {
                    listaMenus.push_back(nuevo);
                    guardarMenu();
                }
            }
                break;
                        case 2:
                                buscarMenu();
                                break;
                        case 3:
```

```
                        actualizar();
                        break;
                    case 4:
                       eliminarMenu();
                       break;
                    case 5:
                        leerMenu();
                        break;
            case 0:
               cout << "\nSALIENDO DEL SISTEMA...\n";
               return 0;

            default:
               cout << "\nOPCIÓN INVÁLIDA. DÍGITE UNA OPCIÓN CORRECTA\n";
         }
      }
      return 0;
}
```