

Estructura del Informe

Análisis del Problema

1.1 Descripción del Problema

El sistema busca resolver dos objetivos principales:

- **Gestión de información mediante una pila:** Administrar inserciones, eliminaciones y consultas sobre una colección de datos (en este caso, "platos") usando una estructura LIFO (Last In, First Out).
- **Resolución de un problema matemático básico:** Calcular el factorial de un número entero no negativo. Para ello, se utiliza una pila de enteros que permite almacenar y procesar los factores de manera ordenada.

1.2 Requerimientos del Sistema

Requerimientos

- **Ingreso de elementos en la pila:** Permitir insertar nuevos datos (platos) o números para el cálculo.
- **Eliminación de elementos:** Eliminar el elemento en la parte superior de la pila.
- **Consulta del elemento superior:** Visualizar el elemento que se encuentra en la parte superior de la pila.
- **Verificación de estado:** Determinar si la pila está vacía.
- **Visualización completa:** Mostrar todos los elementos almacenados.
- **Cálculo de factorial:** Permitir al usuario ingresar un número y calcular su factorial utilizando la pila.
- **Menú interactivo:** Contar con un menú que permita acceder a todas estas funcionalidades de manera intuitiva.

1.3 Estructuras de Datos Propuestas

- **Pila mediante Nodos (para platos):** Se utiliza una pila implementada con nodos enlazados. Cada nodo contiene un dato (el nombre de un plato) y un puntero al siguiente nodo.
- **Pila de Enteros (para el factorial):** Se utiliza una pila similar para almacenar los números enteros (factores) que se usarán para calcular el factorial.

1.4 Justificación de la Elección

- **Adecuación al Problema:** La estructura de pila es idónea para operaciones de tipo LIFO, lo que la hace perfecta para gestionar datos donde la inserción y eliminación deben ocurrir en un extremo específico (la "parte superior").
- **Eficiencia:** Las operaciones en una pila (push, pop, peek) se realizan en tiempo constante, lo cual es fundamental para aplicaciones que requieran respuesta rápida.
- **Flexibilidad:** La implementación con nodos enlazados permite un manejo dinámico de la memoria, evitando límites fijos en la cantidad de datos que se pueden manejar.

Diseño de la Solución

2.1 Descripción de las Estructuras de Datos y Operaciones

- **Estructura de Pila (nodos):** Cada nodo contiene el dato y un puntero que apunta al siguiente nodo. Se mantiene un puntero denominado "tope" para indicar el último elemento agregado.
- **Operaciones Definidas:**
 - *Ingreso (push):* Agregar un nuevo nodo en la parte superior.
 - *Eliminación (pop):* Remover el nodo en la parte superior.
 - *Consulta (peek):* Mostrar el elemento en la parte superior sin eliminarlo.
 - *Verificación:* Comprobar si la pila está vacía.
 - *Visualización completa y vaciado:* Listar todos los elementos o eliminar todos de la pila.

2.2 Algoritmos Principales

Pseudocódigo para Agregar un Proceso (Insertar Elemento)

Algoritmo AgregarProceso(Dato):

Crear un nuevo nodo

nodo.dato ← Dato

nodo.siguiente ← tope

tope ← nodo

Incrementar el contador de elementos (tamaño)

FinAlgoritmo

Pseudocódigo para Cambiar el Estado del Proceso (Ejemplo: Marcar un elemento como procesado o realizar pop)

Algoritmo CambiarEstadoProceso:

Si tope es NULL entonces

Imprimir "La pila está vacía"

Sino

Nodo temporal ← tope

Actualizar tope a tope.siguiente

Liberar memoria del nodo temporal

Decrementar el contador de elementos (tamaño)

FinSi

FinAlgoritmo

2.3 Diagramas de Flujo

Inicio

|

▼

¿Usuario ingresa dato? —► [Sí] —► Crear nodo con el dato

|

▼

[No]

|

▼

Asignar nodo.siguiiente = tope

|

▼

Fin

|

▼

Actualizar tope = nodo

|

▼

Fin de inserción

2.4 Justificación del Diseño

- **Ventajas y Eficiencia:**
 - La pila permite realizar operaciones de inserción y eliminación en tiempo $O(1)$, lo que garantiza rapidez.
 - El uso de nodos enlazados permite una asignación dinámica de memoria, evitando límites fijos.
- **Claridad y Mantenimiento:**
 - La modularidad permite separar la lógica de gestión de platos y la resolución del problema matemático, facilitando pruebas y futuras modificaciones.

Solución Final

3.1 Código

```
#include <iostream>

#include <string>

#include <locale>

using namespace std;

struct Nodo {

    string plato;

    Nodo* siguiente;

};

class Pila {

private:

    Nodo* tope;

    int tamano;

public:

    Pila() {

        tope = NULL;

        tamano = 0;

    }

    void ingresoP(string plato) {

        Nodo* nuevoNodo = new Nodo();

        nuevoNodo->plato = plato;

        nuevoNodo->siguiente = tope;

        tope = nuevoNodo;

        tamano++;

        cout << "El plato " << plato << " se ha insertado." << endl;
```

```
}
```

```
void EliminarP() {
```

```
    if (tope == NULL) {
```

```
        cout << "La pila de platos está vacía." << endl;
```

```
    } else {
```

```
        Nodo* temp = tope;
```

```
        cout << "El plato " << tope->plato << " se ha eliminado." << endl;
```

```
        tope = tope->siguiente;
```

```
        delete temp;
```

```
        tamano--;
```

```
    }
```

```
}
```

```
string UltimoE() {
```

```
    if (tope != NULL) {
```

```
        return tope->plato;
```

```
    } else {
```

```
        return "Pila vacía.";
```

```
    }
```

```
}
```

```
bool Vacio() {
```

```
    return tope == NULL;
```

```
}
```

```

void Mostrarp() {

    if (Vacio()) {

        cout << "La pila de platos está vacía." << endl;

    } else {

        cout << "Lista de platos:" << endl;

        Nodo* temp = tope;

        while (temp != NULL) {

            cout << temp->plato << " ";

            temp = temp->siguiente;

        }

        cout << endl;

    }

}

```

```

int TamanoPila() {

    return tamano;

}

```

```

void VaciarPila() {

    while (!Vacio()) {

        EliminarP();

    }

    cout << "Todos los elementos han sido eliminados." << endl;

}

};

```

```

struct NodoInt {

```

```

int valor;

NodoInt* siguiente;

};

class PilaInt {

private:

    NodoInt* tope;

public:

    PilaInt() {

        tope = NULL;

    }

    void push(int valor) {

        NodoInt* nuevo = new NodoInt();

        nuevo->valor = valor;

        nuevo->siguiente = tope;

        tope = nuevo;

    }

    int pop() {

        if (tope == NULL) {

            cout << "La pila de enteros está vacía." << endl;

            return 1; // Valor por defecto en caso de error

        } else {

            int val = tope->valor;

            NodoInt* temp = tope;

            tope = tope->siguiente;

            delete temp;

```

```

        return val;

    }

}

bool empty() {

    return tope == NULL;

}

};

void resolverFactorial() {

    int n;

    cout << "\n--- Resolver Problema Matemático: Calcular Factorial ---" << endl;

    cout << "Problema: Calcular el factorial de un número entero n (n!). " << endl;

    cout << "Definición:  $n! = n * (n-1) * (n-2) * \dots * 1$ , siendo  $0! = 1$ ." << endl;

    cout << "Ingrese un número entero no negativo: ";

    cin >> n;

    // Verificación básica e interpretación del input.

    if (n < 0) {

        cout << "\nInterpretación: El número ingresado (" << n << ") es negativo." << endl;

        cout << "El factorial está definido solo para enteros no negativos." << endl;

        return;

    }

    cout << "\nInterpretación: Ha ingresado n = " << n << ". Procederemos a calcular " << n << "!"
    << endl;

    cout << "Paso 1: La pila se utilizará para almacenar los números del 1 hasta " << n << ". " <<
    endl;

```



```

Pila<int> pila;

for (int i = 1; i <= n; i++) {

    pila.push(i);

    cout << "Interpretación: Se agrega el factor " << i << " a la pila." << endl;

}

long long resultado = 1;

cout << "\nPaso 2: Se extraen los factores de la pila para multiplicarlos:" << endl;

while (!pila.empty()) {

    int factor = pila.pop();

    cout << "Interpretación: Se extrae el factor " << factor << " de la pila." << endl;

    resultado *= factor;

    cout << "Interpretación: Resultado intermedio = " << resultado << endl;

}

cout << "\nInterpretación: El factorial de " << n << " (n!) es: " << resultado << endl;

}

int main() {

    setlocale(LC_CTYPE, "Spanish");

    Pila pilaPlatos;

    int opcion;

    string plato;

    do {

```

```
cout << "\n=== MENÚ PRINCIPAL ===" << endl;

cout << "1. Insertar plato en la pila" << endl;

cout << "2. Eliminar plato de la pila" << endl;

cout << "3. Ver plato superior" << endl;

cout << "4. Verificar si la pila de platos está vacía" << endl;

cout << "5. Mostrar todos los platos" << endl;

cout << "6. Mostrar tamaño de la pila de platos" << endl;

cout << "7. Eliminar todos los platos de la pila" << endl;

cout << "8. Resolver problema matemático: Calcular factorial" << endl;

cout << "9. Salir" << endl;

cout << "Ingrese una opción: ";

cin >> opcion;

switch (opcion) {

    case 1:

        cout << "Ingrese el nombre del plato: ";

        cin >> plato;

        pilaPlatos.ingresoP(plato);

        break;

    case 2:

        pilaPlatos.EliminarP();

        break;

    case 3:

        cout << "Plato superior: " << pilaPlatos.UltimoE() << endl;

        break;

    case 4:
```

```
        cout << (pilaPlatos.Vacio() ? "La pila de platos está vacía." : "La pila de platos tiene  
elementos.") << endl;
```

```
        break;
```

```
    case 5:
```

```
        pilaPlatos.Mostrarp();
```

```
        break;
```

```
    case 6:
```

```
        cout << "Tamaño de la pila de platos: " << pilaPlatos.TamanoPila() << endl;
```

```
        break;
```

```
    case 7:
```

```
        pilaPlatos.VaciarPila();
```

```
        break;
```

```
    case 8:
```

```
        resolverFactorial();
```

```
        break;
```

```
    case 9:
```

```
        cout << "Saliendo del programa..." << endl;
```

```
        break;
```

```
    default:
```

```
        cout << "Opción no válida. Intente nuevamente." << endl;
```

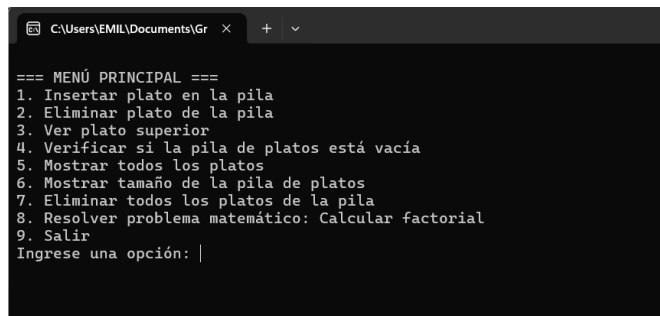
```
    }
```

```
    } while (opcion != 9);
```

```
    return 0;
```

```
}
```

3.2 Capturas de Pantalla



```
C:\Users\EMIL\Documents\Gr x + v

=== MENÚ PRINCIPAL ===
1. Insertar plato en la pila
2. Eliminar plato de la pila
3. Ver plato superior
4. Verificar si la pila de platos está vacía
5. Mostrar todos los platos
6. Mostrar tamaño de la pila de platos
7. Eliminar todos los platos de la pila
8. Resolver problema matemático: Calcular factorial
9. Salir
Ingrese una opción: |
```

```
=== MENÚ PRINCIPAL ===
1. Insertar plato en la pila
2. Eliminar plato de la pila
3. Ver plato superior
4. Verificar si la pila de platos está vacía
5. Mostrar todos los platos
6. Mostrar tamaño de la pila de platos
7. Eliminar todos los platos de la pila
8. Resolver problema matemático: Calcular factorial
9. Salir
Ingrese una opción: 1
Ingrese el nombre del plato: te
El plato te se ha insertado.
```

```
=== MENÚ PRINCIPAL ===
1. Insertar plato en la pila
2. Eliminar plato de la pila
3. Ver plato superior
4. Verificar si la pila de platos está vacía
5. Mostrar todos los platos
6. Mostrar tamaño de la pila de platos
7. Eliminar todos los platos de la pila
8. Resolver problema matemático: Calcular factorial
9. Salir
Ingrese una opción: 2
El plato te se ha eliminado.
```

=== MENÚ PRINCIPAL ===

1. Insertar plato en la pila
2. Eliminar plato de la pila
3. Ver plato superior
4. Verificar si la pila de platos está vacía
5. Mostrar todos los platos
6. Mostrar tamaño de la pila de platos
7. Eliminar todos los platos de la pila
8. Resolver problema matemático: Calcular factorial
9. Salir

Ingrese una opción: 8

--- Resolver Problema Matemático: Calcular Factorial ---

Problema: Calcular el factorial de un número entero n ($n!$).

Definición: $n! = n * (n-1) * (n-2) * \dots * 1$, siendo $0! = 1$.

Ingrese un número entero no negativo: 5

Interpretación: Ha ingresado $n = 5$. Procederemos a calcular $5!$

Paso 1: La pila se utilizará para almacenar los números del 1 hasta 5.

Interpretación: Se agrega el factor 1 a la pila.

Interpretación: Se agrega el factor 2 a la pila.

Interpretación: Se agrega el factor 3 a la pila.

Interpretación: Se agrega el factor 4 a la pila.

Interpretación: Se agrega el factor 5 a la pila.

Paso 2: Se extraen los factores de la pila para multiplicarlos:

Interpretación: Se extrae el factor 5 de la pila.

Interpretación: Resultado intermedio = 5

Interpretación: Se extrae el factor 4 de la pila.

Interpretación: Resultado intermedio = 20

Interpretación: Se extrae el factor 3 de la pila.

Interpretación: Resultado intermedio = 60

Interpretación: Se extrae el factor 2 de la pila.

Interpretación: Resultado intermedio = 120

Interpretación: Se extrae el factor 1 de la pila.

Interpretación: Resultado intermedio = 120

Interpretación: El factorial de 5 ($n!$) es: 120