Universidad Mariano Gálvez de Guatemala
Carrera de Ingeniería en Sistemas – 5090
Curso: Programación I
Docente: Ing. Carlos Alejandro Arias

Laboratorio 9
Pilas y Colas

Pablo Javier Roldán Vásquez
Carné: 5090-23-13164
Fecha 15/05/2024

Introducción

En el ámbito de la programación, el dominio de las estructuras de datos básicas es fundamental para construir soluciones eficientes y escalables. En este laboratorio, nos enfocaremos en comprender y aplicar el uso de dos estructuras de datos fundamentales: pilas y colas, en el contexto de la programación en C++. Estas estructuras proporcionan un marco sólido para organizar y manipular datos de manera ordenada y eficiente. A través de la implementación de un programa que gestione una lista de tareas, se aplicarán las operaciones fundamentales de estas estructuras: push y pop para pilas, y enqueue y dequeue para colas.

Se declaran las bibliotecas y se crea la clase pila la cual eliminara el valor más reciente, utilizando push para agregar un valor y pop para eliminar el valor

```cpp
#include <iostream>
#include <stack>
#include <queue>
#include <vector> // Necesitas incluir vector

using namespace std;

// Clase Pila para manejar una pila de enteros
class Pila {
public:
    void push(int valor) { stack_.push(valor); }  // Agrega un valor a la pila
    int pop() {  // remueve y retorna el valor en la parte superior de la pila
        int valor = stack_.top();
        stack_.pop();
        return valor;
    }
    bool empty() const { return stack_.empty(); } // Verifica si la pila está vacía
private:
    stack<int> stack_;  // Objeto de tipo stack para almacenar los enteros
};
```

Se declara la clase cola la cual eliminará el valor más antiguo, es decir el primero que se ingresó en la lista, utilizando queue para agregar un valor a la cola y dequeue para eliminar el valor más antiguo.

También se crea la clase tarea que representa una tarea con el número que le dará el usuario.

```cpp
// Clase Cola para manejar una cola de enteros
class Cola {
public:
    void enqueue(int valor) { queue_.push(valor); } // Agrega un valor a la cola
    int dequeue() { // Remueve y retorna el valor en la parte frontal de la cola
        int valor = queue_.front();
        queue_.pop();
        return valor;
    }
    bool empty() const { return queue_.empty(); } // Verifica si la cola está vacía
private:
    queue<int> queue_; // Objeto de tipo queue para almacenar los enteros
};

// Clase Tarea para representar una tarea con un número
class Tarea {
public:
    int numero; // Número de la tarea
    Tarea(int num) : numero(num) {} // Constructor que inicializa la tarea con un número dado
};
```

Se crean instancias para la clase pila, la clase cola y un vector que almacenará las tareas, también se crea el menú con el que el usuario decidirá la acción que desea realizar.

```
int main() {
    Pila pila; // Crear una instancia de la clase Pila
    Cola cola; // Crear una instancia de la clase Cola
    vector<Tarea> tareas; // Crear un vector para almacenar las tareas
    system("color 8e");
    int opcion;
    do {
        system("cls");
        cout << "----------Menu-------------" << endl;
        cout << "1. Agregar nueva tarea" << endl;
        cout << "2. Completar la ultima tarea" << endl;
        cout << "3. Atender la tarea mas antigua" << endl;
        cout << "4. Mostrar todas las tareas" << endl;
        cout << "5. Salir" << endl;
        cout << "--------------------------" << endl;
        cin >> opcion;
        system("cls");
```

La primera opción es agregar una nueva tarea la cual agrega la tarea a la cola y a la pila, luego la almacena en el vector y le asignará el número que ingresó el usuario.

```
    switch (opcion) {
        case 1: {
            int nuevoNumero;
            cout << "Ingrese el numero de la nueva tarea: ";
            cin >> nuevoNumero;  // Leer el número de la nueva tarea
            tareas.push_back(Tarea(nuevoNumero)); // Agregar la tarea al vector
            cola.enqueue(nuevoNumero);  // Agregar la tarea a la cola
            pila.push(nuevoNumero);     // Agregar la tarea a la pila
            break;
            system("pause");
            }
```

La segunda opción muestra un mensaje de que la tarea más reciente se ha completado y luego la elimina, si no hay ninguna tarea almacenada en el vector mostrará un mensaje al no encontrar ninguna tarea.

```
case 2:
    if (!pila.empty()) {
        cout << "Ultima tarea completada: " << pila.pop() << "\n"; // Completa y muestra la última tarea
        tareas.pop_back(); // Remueve la última tarea del vector
    } else {
        cout << "No hay tareas pendientes.\n";
    }
    system("pause");
    break;
```

La tercera opción busca por medio de un bucle la tarea más antigua de la lista y la elimina luego de haber mostrado un mensaje de que se atendió la tarea más antigua. Si no encuentra ninguna tarea en la lista mostrará un mensaje de que no hay tareas en la cola

```
case 3:
    if (!cola.empty()) {
    int numeroTarea = cola.dequeue();
        cout << "Atendiendo la tarea mas antigua: " << numeroTarea << "\n";
    for (auto it = tareas.begin(); it!= tareas.end(); ++it) { // Busca y eliminaa la tarea correspondiente del vector tareas
    if ((*it).numero == numeroTarea) {
        tareas.erase(it); // Eliminar la tarea encontrada
    break; // Sale del bucle una vez encontrada y eliminada la tarea
            }
        }
    } else {
        cout << "No hay tareas en cola.\n";
    }
    system("pause");
    break;
```

En la cuarta opción muestra en pantalla las tareas existentes y la quinta opción es para salir del programa.

```
            case 4:
                cout << "Todas las tareas:\n";
                for (const auto& tarea : tareas) {
                    cout << tarea.numero << "\n"; // Muestra todas las tareas
                }
                system("pause");
                break;
            case 5:
                cout << "Saliendo...\n";
                break;
            default:
                cout << "Opcion invalida.\n";
        }
    } while (opcion != 5);

    return 0;
```

Conclusión

Este laboratorio ha proporcionado la oportunidad valiosa para profundizar la comprensión sobre el uso de estructuras de datos básicas en la programación en C++. Al implementar un programa para gestionar una lista de tareas utilizando pilas y colas, Se han practicado las operaciones fundamentales asociadas con estas estructuras, como push, pop, enqueue y dequeue. Esta experiencia práctica no solo ha permitido consolidar los conocimientos teóricos, sino también ha brindado una visión clara de cómo estas estructuras pueden ser aplicadas de manera efectiva para resolver problemas en el mundo real.

Referencias

Github: https://github.com/PabloRoldan2/Laboratorio-9.git