

Universidad Mariano Gálvez de Guatemala

Facultad de ingeniería en sistemas

Campus Villa nueva, Guatemala

Curso: Programación I

Docente: Inge. Carlos Arias



Actividad: Laboratorio 9

Nombre: Stephanie Cristina Sabán Cárcamo

Carnet: 5090- 23-11167

Sección: "A"

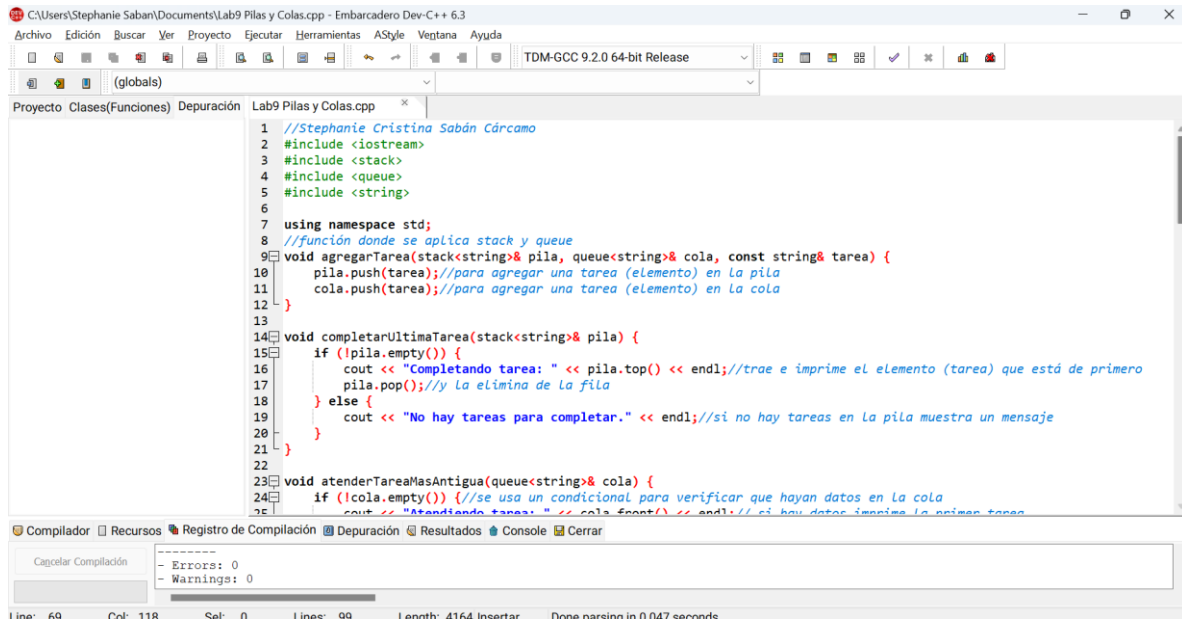
Fecha: 15/05/24

## Introducción

Una pila es una estructura de datos lineal que sigue el principio "Último en Entrar, Primero en Salir" (LIFO), donde `push(item)` agrega un elemento a la parte superior, `pop()` elimina y devuelve el elemento en la parte superior, y `top()` devuelve el elemento en la parte superior sin eliminarlo; la librería `stack` en C++ proporciona una implementación. Una fila es una estructura de datos lineal que sigue el principio "Primero en Entrar, Primero en Salir" (FIFO), donde `enqueue(item)` agrega un elemento al final, `dequeue()` elimina y devuelve el elemento al frente, y `front()` devuelve el elemento al frente sin eliminarlo; la librería `queue` en C++ proporciona una implementación.

Estas estructuras son fundamentales en la programación y se utilizan en diversas aplicaciones, como la gestión de tareas (donde las tareas se agregan a una fila y se procesan en el orden en que llegaron), la administración de búferes (donde los datos se almacenan temporalmente en una fila o pila), la evaluación de expresiones (donde los operandos se colocan en una pila), y el recorrido de estructuras de datos (como árboles, donde se utiliza una pila o fila para realizar un recorrido en profundidad o en anchura).

## Código Comentado Primer Programa



```
1 //Stephanie Cristina Sabán Cárcamo
2 #include <iostream>
3 #include <stack>
4 #include <queue>
5 #include <string>
6
7 using namespace std;
8 //función donde se aplica stack y queue
9 void agregarTarea(stack<string>& pila, queue<string>& cola, const string& tarea) {
10     pila.push(tarea); //para agregar una tarea (elemento) en la pila
11     cola.push(tarea); //para agregar una tarea (elemento) en la cola
12 }
13
14 void completarUltimaTarea(stack<string>& pila) {
15     if (!pila.empty()) {
16         cout << "Completando tarea: " << pila.top() << endl; //trae e imprime el elemento (tarea) que está de primero
17         pila.pop(); //y la elimina de la fila
18     } else {
19         cout << "No hay tareas para completar." << endl; //si no hay tareas en la pila muestra un mensaje
20     }
21 }
22
23 void atenderTareaMasAntigua(queue<string>& cola) {
24     if (!cola.empty()) //se usa un condicional para verificar que hayan datos en la cola
25         cout << "Atendiendo tarea: " << cola.front() << endl; //si hay datos imprime la primer tarea
```

El código utiliza las estructuras de datos stack (pila) y queue (cola) de la biblioteca estándar de C++

Se incluyen los encabezados necesarios: “<iostream>”, “<stack>”, “<queue>” y “<string>”. Luego la función agregarTarea toma una cadena “tarea” y la agrega a la pila “pila” y a la cola “cola” utilizando los métodos “push” y “push”, respectivamente.

La función completarUltimaTarea maneja la pila “pila”

Si la pila no está vacía (!pila.empty()), imprime "Completando tarea: " seguido del elemento en la cima de la pila (pila.top()) y luego lo elimina de la pila (pila.pop()).

Si la pila está vacía, imprime "No hay tareas para completar."

La función atenderTareaMasAntigua maneja la cola cola:

Si la cola no está vacía (!cola.empty()), imprime "Atendiendo tarea: " seguido del elemento al frente de la cola (cola.front()). Sin embargo, la línea está incompleta y no se muestra el resto de la implementación.

```

21 }
22
23 void atenderTareaMasAntigua(queue<string>& cola) {
24     if (!cola.empty()) {//se usa un condicional para verificar que hayan datos en la cola
25         cout << "Atendiendo tarea: " << cola.front() << endl;// si hay datos imprime la primer tarea
26         cola.pop();//luego de imprimirla la borra de la cola
27     } else {
28         cout << "No hay tareas para atender." << endl;//si la cola esta vacia imprime el mensaje
29     }
30 }
31
32 void mostrarTareas(const stack<string>& pila, const queue<string>& cola) {
33     stack<string> tempPila = pila;//crea una pila temporal
34     queue<string> tempCola = cola;//crea una cola temporal
35
36     cout << "Tareas en la pila (ultima primero):" << endl;
37     while (!tempPila.empty()) //siempre que la pila no este vacia va imprimir las tareas de forma (last in first out)
38         cout << tempPila.top() << endl;
39         tempPila.pop();
40     }
41
42     cout << "\nTareas en la cola (primera primero):" << endl;
43     while (!tempCola.empty()) //siempre que la pila no este vacia va imprimir las tareas de forma (first in first out)
44         cout << tempCola.front() << endl;
45         tempCola.pop();
46     }
47 }
48
49 int main() {

```

La función `atenderTareaMasAntigua` verifica si la cola `cola` no está vacía utilizando el método `empty()`. Si no está vacía, imprime "Atendiendo tarea: " seguido del elemento al frente de la cola con `front()`, luego elimina ese elemento con `pop()`. Si la cola está vacía, imprime "No hay tareas para atender.". La función `mostrarTareas` crea copias temporales `tempPila` y `tempCola` de la pila `pila` y la cola `cola`, respectivamente. Imprime "Tareas en la pila (última primero):" y luego imprime los elementos de `tempPila` utilizando `top()` y `pop()` en un bucle `while`. Luego, imprime "Tareas en la cola (primera primero):" e imprime los elementos de `tempCola` utilizando `front()` y `pop()` en otro bucle `while`, siguiendo los principios LIFO (Last-In-First-Out) para la pila y FIFO (First-In-First-Out) para la cola.

```

48
49 int main() {
50     stack<string> pila;
51     queue<string> cola;
52     int opcion;
53     string tarea;
54
55     do //utilizamos un do while para hacer el menu y que no se cierre hasta que nosotros lo digamos.
56     {
57         cout<<"Stephanie Cristina Saban Carcamo"<<endl;
58         cout<<"Menu de actividades pendientes de oficios en la casa:"<<endl;
59         cout << "1. Agregar tarea por hacer" << endl;
60         cout << "2. Completar ultima tarea" << endl;
61         cout << "3. Atender tarea mas antigua" << endl;
62         cout << "4. Mostrar todas las tareas en cola y en pila" << endl;
63         cout << "5. Salir" << endl;
64         cout << "Selecciona una opcion: ";
65         cin >> opcion;
66         cin.ignore();
67         //utilizamos un switch para tener case y poder hacer llamado de las funciones de una manera más ordenada.
68         switch (opcion) {
69             case 1:
70                 do//utilizo un do while para no estar ingresando una y otra vez la opcion 1 para ingresar las tareas
71                 {
72                     cout << "Introduce la tarea o escribe 'terminar': ";
73                     getline(cin, tarea);
74                     agregarTarea(pila, cola, tarea);
75                     }while(tarea!= "terminar");
76                     break;
77             case 2:
78                 completarUltimaTarea(pila);

```

Instrumento de Compilación | Depuración | Resultados | Console  
 Sel: 0    Lines: 99    Length: 4164 Insertar    Done parsing in 0.047 seconds

```

69 do{//utilizo un do while para no estar ingresando una y otra vez la opcion 1 para ingresar las tareas
70     cout << "Introduce la tarea o escribe 'terminar': ";
71     getline(cin, tarea);
72     agregarTarea(pila, cola, tarea);
73     }while(tarea!= "terminar");
74     break;
75     case 2:
76         completarUltimaTarea(pila);
77         system ("pause");//para esperar un rato en pantalla y ver lo que imprime
78         system ("cls");//para limpiar lo que hay en pantalla y no se sature.
79         break;
80     case 3:
81         atenderTareaMasAntigua(cola);//realiza la tarea más antigua y la va borrando
82         system ("pause");
83         system ("cls");
84         break;
85     case 4:
86         mostrarTareas(pila, cola);//muestra todas las tareas que hay por hacer o que aun no se han atendido.
87         system ("pause");
88         system ("cls");
89         break;
90     case 5:
91         cout << "Saliendo del programa." << endl;//ultima opcion para salir del programa
92         break;
93     default:
94         cout << "Opcion invalida." << endl;//si pone una opcion que no esta en el menu sale este mensaje
95     }
96     } while (opcion != 5);
97

```

Este código presenta un menú con diferentes opciones utilizando un bucle do-while y una estructura switch. En el caso 1, permite ingresar tareas a una pila y una cola mediante la función agregarTarea. En el caso 2, completa la última tarea agregada a la pila con completarUltimaTarea y luego pausa la ejecución. En el caso 3, atiende la tarea más antigua de la cola con atenderTareaMasAntigua y pausa. En el caso 4, muestra todas las tareas pendientes en la pila y la cola con mostrarTareas y pausa. En el caso 5, imprime "Saliendo del programa" para salir del bucle. Si se ingresa una opción inválida, muestra "Opción inválida". El bucle while externo se repite hasta que la opción sea 5, permitiendo así al usuario interactuar con el menú de tareas.

### Funcionamiento del programa

```

C:\Users\Stephanie Saban\Do
Stephanie Cristina Saban Carcamo
Menu de actividades pendientes de oficios en la casa:
1. Agregar tarea por hacer
2. Completar ultima tarea
3. Atender tarea mas antigua
4. Mostrar todas las tareas en cola y en pila
5. Salir
Selecciona una opcion: 1
Introduce la tarea o escribe 'terminar': Lavar
Introduce la tarea o escribe 'terminar': Trapear
Introduce la tarea o escribe 'terminar': Barrer
Introduce la tarea o escribe 'terminar': terminar

```

Después de ingresar las tareas, escribir “terminar” para regresar al menú

```
C:\Users\Stephanie Saban\Do x + v
Stephanie Cristina Saban Carcamo
Menu de actividades pendientes de oficios en la casa:
1. Agregar tarea por hacer
2. Completar ultima tarea
3. Atender tarea mas antigua
4. Mostrar todas las tareas en cola y en pila
5. Salir
Selecciona una opcion: 1
Introduce la tarea o escribe 'terminar': Lavar
Introduce la tarea o escribe 'terminar': Trapear
Introduce la tarea o escribe 'terminar': Barrer
Introduce la tarea o escribe 'terminar': terminar
Stephanie Cristina Saban Carcamo
Menu de actividades pendientes de oficios en la casa:
1. Agregar tarea por hacer
2. Completar ultima tarea
3. Atender tarea mas antigua
4. Mostrar todas las tareas en cola y en pila
5. Salir
Selecciona una opcion: 2
```

```
Stephanie Cristina Saban Carcamo
Menu de actividades pendientes de oficios en la casa:
1. Agregar tarea por hacer
2. Completar ultima tarea
3. Atender tarea mas antigua
4. Mostrar todas las tareas en cola y en pila
5. Salir
Selecciona una opcion: 2
Completando tarea: terminar
Presione una tecla para continuar . . . |
```

```
C:\Users\Stephanie Saban\Do x + v
Stephanie Cristina Saban Carcamo
Menu de actividades pendientes de oficios en la casa:
1. Agregar tarea por hacer
2. Completar ultima tarea
3. Atender tarea mas antigua
4. Mostrar todas las tareas en cola y en pila
5. Salir
Selecciona una opcion: 3
Atendiendo tarea: Lavar
Presione una tecla para continuar . . . |
```

```
C:\Users\Stephanie Saban\Do x + v
Stephanie Cristina Saban Carcamo
Menu de actividades pendientes de oficios en la casa:
1. Agregar tarea por hacer
2. Completar ultima tarea
3. Atender tarea mas antigua
4. Mostrar todas las tareas en cola y en pila
5. Salir
Selecciona una opcion: 4
Tareas en la pila (ultima primero):
Terminar
Barrer
Trapear
Lavar

Tareas en la cola (primera primero):
Trapear
Barrer
Terminar
terminar
Presione una tecla para continuar . . . |
```

Muestra todas las tareas pendientes en la cola y en la fila en su respectivo orden.

### Conclusión

Las pilas y colas son estructuras de datos fundamentales en programación que siguen los principios LIFO (Último en Entrar, Primero en Salir) y FIFO (Primero en Entrar, Primero en Salir), respectivamente. En C++, las operaciones típicas de las pilas incluyen push (agregar un elemento), pop (eliminar el elemento superior) y top (obtener el elemento superior sin eliminarlo); mientras que en las colas se utilizan enqueue (agregar un elemento al final), dequeue (eliminar el elemento del frente) y front (obtener el elemento del frente sin eliminarlo). Este código demuestra la implementación de estas estructuras utilizando las librerías stack y queue de la biblioteca estándar de C++, junto con un menú que permite al usuario ingresar tareas, completar y atender tareas, y mostrar las tareas pendientes tanto en la pila como en la cola, ilustrando así el uso práctico de estas estructuras de datos en un programa.

### LINK DEL REPOSITORIO CON EL PROGRAMA

<https://github.com/StephanieSabann/Lab9-Stephanie-Sab-n.git>