

Universidad Mariano Gálvez de Guatemala

Campus Villa Nueva, Guatemala

Ingeniería en Sistemas

Ing. Carlos Arias

Curso: Programación I

Laboratorio 9

Sección: “A”

Carné: 5090-23-1407

Nombre: Angelyn Judith Díaz Zeceña

Introducción

Como podemos ver, en este laboratorio, se llevó a cabo la utilización de estructuras de pilas y colas en C++, demostrándolo por medio de un listado de opciones en el que un estudiante decide agregar tareas, completar tareas, atender tareas, mostrarlas y seleccionar si ya no desea agregar más datos, pero que más adelante podrá observar a detalle.

Contenido

```
Laboratorio9 AdministradorDeTareas
1 //Angelyn Judith Díaz Zeceña 5090-23-1407
2
3 #include <iostream> //Aqui se utiliza para la entrada y salida de datos
4 #include <stack> //esta es una biblioteca que sirve para crear y operar con una pila
5 #include <queue> //Esta es una biblioteca para el uso de estructuras de cola
6 #include <string>
7
8 class AdministradorDeTareas {
9 private:
10     // Aqui se utiliza una pila para manejar las tareas
11     std::stack<std::string> pilaDeTareas;
12     // Aqui se utiliza una Cola para manejar las tareas
13     std::queue<std::string> colaDeTareas;
14
15 public:
16     // Función para agregar una tarea a la pila y la cola
17     void agregarTarea(const std::string& tarea) {
18         pilaDeTareas.push(tarea);
19         colaDeTareas.push(tarea);
20     }
21
22     // Aqui se elimina la pila de la funcion
23     void completarUltimaTarea() {
24         if (!pilaDeTareas.empty()) {
25             std::string ultimaTarea = pilaDeTareas.top();
26             pilaDeTareas.pop();
27             std::cout << "Tarea completada y eliminada: " << ultimaTarea << std::endl;
28         }
29         else {
30             std::cout << "No hay tareas para completar." << std::endl;
31         }
32     }
33
34     // Función para atender la tarea más antigua
35     void atenderTareaMasAntigua() {
36         if (!colaDeTareas.empty()) {
37             std::string tareaMasAntigua = colaDeTareas.front();
38             colaDeTareas.pop();
39             std::cout << "Tarea atendida y eliminada: " << tareaMasAntigua << std::endl;
40         }
41         else {
42             std::cout << "No hay tareas para atender." << std::endl;
43         }
44     }
45 }
```

Como podemos ver en esta captura de pantalla, se utilizaron diferentes tipos de bibliotecas, como “string” para manejar las cadenas de texto, como también los tipos de clases privadas y públicas. Con la utilización de pila y cola, y gracias a la ayuda de un void, y un ciclo, permitiendo que la clase pueda agregar tareas y completar la última tarea agregada (usando una pila) y atender la tarea más antigua (usando una cola).

```

// Función para mostrar todas las tareas en la cola
void mostrarTodasLasTareas() {
    if (colaDeTareas.empty()) {
        std::cout << "No hay tareas para mostrar." << std::endl;
        return;
    }

    std::cout << "Tareas actuales:" << std::endl;
    std::queue<std::string> colaTemp = colaDeTareas; // Copia de la cola para no modificar la original
    while (!colaTemp.empty()) {
        std::cout << "- " << colaTemp.front() << std::endl;
        colaTemp.pop();
    }
}

};

int main() {
    std::cout << "-----Hola personita-----" << std::endl;

    AdministradorDeTareas adminTareas;
    int opcion;
    std::string tarea;

    // Aqui se utiliza un Bucle principal para el menu
    do {

        std::cout << "Menu:" << std::endl;
        std::cout << "1. Agregar una tarea" << std::endl;
        std::cout << "2. Completar la ultima tarea" << std::endl;
        std::cout << "3. Atender la tarea mas antigua" << std::endl;
        std::cout << "4. Mostrar todas las tareas" << std::endl;
        std::cout << "5. Salir" << std::endl;
        std::cout << "Ingrese su opcion: ";
        std::cin >> opcion;

```

Como podemos ver en esta otra captura de pantalla, se utiliza un `main`, y gracias a la ayuda de un `cout`, el usuario es saludando, mientras que se declara un objeto llamado “`adminTareas`” y “`AdministradorDeTareas`” para gestionar las tareas. Declarando variables y gracias a la ayuda de un bucle `do-while`, que implementa un menú interactivo presentando 5 opciones.

```

switch (opcion) {
    case 1:
        std::cout << "Ingrese la descripcion de la tarea: ";
        std::getline(std::cin, tarea);
        adminTareas.agregarTarea(tarea); // Aqui Agrega la tarea
        break;
    case 2:
        adminTareas.completarUltimaTarea(); // Aqui Completa la última tarea
        break;
    case 3:
        adminTareas.atenderTareaMasAntigua(); // Aqui Atiende la tarea más antigua
        break;
    case 4:
        adminTareas.mostrarTodasLasTareas(); // Aqui muestra todas las tareas
        break;
    case 5:
        std::cout << "Saliendo del programa." << std::endl;
        break;
    default:
        std::cout << "Opcion invalida. Por favor, intente de nuevo." << std::endl;
}
} while (opcion != 5); // Aqui se utiliza un ciclo para repetir el menú hasta que el usuario elija salir

return 0; // Aqui muestra que el programa ya ha finalizado

```

Como podemos ver en esta captura, se presenta 5 casos contenidos en un “switch” y llamando a los métodos para que, al momento de seleccionar una opción del menú, pueda ser ejecutada de la mejor manera, y en dado caso, se utiliza un “default”, para que si la opción no coincide con ninguno de los casos anteriores, imprima un mensaje de "Opción inválida" y solicita al usuario que intente de nuevo. Juntamente con la ayuda de un while y un return, indicando que el programa ya ha finalizado.

```
C:\Users\Angy\OneDrive\Doc x + v
-----Hola personita-----
-----
Menu:
1. Agregar una tarea
2. Completar la ultima tarea
3. Atender la tarea mas antigua
4. Mostrar todas las tareas
5. Salir
Ingrese su opcion: 1
Ingrese la descripcion de la tarea: matematicas
-----
Menu:
1. Agregar una tarea
2. Completar la ultima tarea
3. Atender la tarea mas antigua
4. Mostrar todas las tareas
5. Salir
Ingrese su opcion: 1
Ingrese la descripcion de la tarea: calculo
-----
Menu:
1. Agregar una tarea
2. Completar la ultima tarea
3. Atender la tarea mas antigua
4. Mostrar todas las tareas
5. Salir
Ingrese su opcion: 1
Ingrese la descripcion de la tarea: proceso
-----
```

Como podemos ver en esta captura de pantalla, se inicia saludando al usuario y solicitándole que coloque alguna opción. Ingresando la opción 1 para la descripción o agregación de tarea, en este caso, decidí agregar 3: matemáticas, calculo y proceso y colocando un separador para que se vea de manera más adecuada.

```
C:\Users\Angy\OneDrive\Doc  X + v
-----
Menu:
1. Agregar una tarea
2. Completar la ultima tarea
3. Atender la tarea mas antigua
4. Mostrar todas las tareas
5. Salir
Ingrese su opcion: 2
Tarea completada y eliminada: proceso
-----
Menu:
1. Agregar una tarea
2. Completar la ultima tarea
3. Atender la tarea mas antigua
4. Mostrar todas las tareas
5. Salir
Ingrese su opcion: 3
Tarea atendida y eliminada: matematicas
-----
Menu:
1. Agregar una tarea
2. Completar la ultima tarea
3. Atender la tarea mas antigua
4. Mostrar todas las tareas
5. Salir
Ingrese su opcion: 4
Tareas actuales:
- calculo
- proceso
-----
```

En esta otra captura de pantalla, podemos ver que se ingresó la opción dos y tres, funcionando de manera exitosa, completando, atendiendo, y eliminando tareas. Luego, seleccionando la opción 4 para mostrar las tareas restantes por hacer.

```
Consola de depuración de Mi  X + v
4. Mostrar todas las tareas
5. Salir
Ingrese su opcion: 3
Tarea atendida y eliminada: matematicas
-----
Menu:
1. Agregar una tarea
2. Completar la ultima tarea
3. Atender la tarea mas antigua
4. Mostrar todas las tareas
5. Salir
Ingrese su opcion: 4
Tareas actuales:
- calculo
- proceso
-----
Menu:
1. Agregar una tarea
2. Completar la ultima tarea
3. Atender la tarea mas antigua
4. Mostrar todas las tareas
5. Salir
Ingrese su opcion: 5
Saliendo del programa.

C:\Users\Angy\OneDrive\Documentos\TERCER SEMESTRE\Laboratorio9\x64\Debug\Laboratorio9.exe (proceso 20272) se cerró con e
l código 0.
Para cerrar automáticamente la consola cuando se detiene la depuración, habilite Herramientas ->Opciones ->Depuración ->
Cerrar la consola automáticamente al detenerse la depuración.
Presione cualquier tecla para cerrar esta ventana. . .
```

Y por último, podemos ver como en esta captura, existe la opción 5, ya que el usuario puede seguir ingresando datos hasta donde decida que, ya es suficiente y colocar la opción 5 para finalizar el ingreso de datos y así salirse exitosamente del programa.

Conclusión

En conclusión, este fue un laboratorio bastante interesante de realizar, ya que pude descubrir la utilización de nuevas bibliotecas, agregar mi toque decorativo al finalizar un menú y a poner en practica lo que es la utilización de una pila y una cola dentro de C++, ya que, es algo que muchas veces se utiliza, pero muchas veces no le prestamos la debida atención que se merece y a lo útil que se vuelve con el manejo correcto dentro del programa.

Referencias

<https://github.com/Ashe122/LABORATORIO-9.git>