

**UNIVERSIDAD MARIANO GÁLVEZ DE GUATEMALA**

**SEDE SOLOLÁ, SOLOLÁ, INGENIERÍA EN SISTEMAS**



**TERCER CICLO**

**Catedra:** Programación I.

**Catedrático:** Inge. Neftalí Cristian García López

**Tema:**

Tarea semana 9

**Estudiante:**

Selvin Geovany Tzunún Balán.

Carnet: 2290-24-44.

Pascual Evelio Yaxon Tziquin

Carnet: 2290-24-26037

## Pilas Estáticas: Código Comentado:

```
area semana 9.cpp  X
Tarea semana 9

1 // Tarea semana 9.cpp : Este archivo contiene la función "main". La ejecución del programa comienza y termina ahí.
2 // Pilas Estáticas
3
4 #include <iostream> // Se incluye la biblioteca para realizar operaciones de entrada y salida (cout, cin, etc.)
5
6 using namespace std; // Se usa el espacio de nombres estándar para evitar escribir std:: antes de cout, endl, etc.
7
8 // Definición de la clase PilaEstática
9 class PilaEstática {
10 private:
11     static const int MAX = 10; // Tamaño máximo de la pila (capacidad fija de 10 elementos)
12     int arr[MAX]; // Arreglo que almacena los elementos de la pila
13     int top; // Índice del último elemento insertado en la pila
14
15 public:
16     // Constructor: inicializa la pila vacía con top en -1
17     PilaEstática() : top(-1) {}
18
19     // Método para insertar (empujar) un valor en la pila
20     bool push(int valor) {
21         if (top >= MAX - 1) { // Verifica si la pila está llena
22             cout << "Pila esta llena (overflow). " << endl; // Muestra mensaje de error si hay desbordamiento
23             return false; // Retorna falso indicando que no se pudo insertar
24         }
25         arr[++top] = valor; // Incrementa top y luego inserta el valor en esa posición
26         return true; // Retorna verdadero indicando que se insertó con éxito
27     }
28
29     // Método para eliminar (sacar) el valor de la cima de la pila
30     int pop() {
31         if (top < 0) { // Verifica si la pila está vacía
32             cout << "Pila esta vacia (underflow). " << endl; // Muestra mensaje de error si hay subdesbordamiento
33             return -1; // Retorna un valor de error
34         }
35         return arr[top--]; // Devuelve el valor en la cima y luego decreuenta top
36     }
37
38     // Método para ver el valor en la cima de la pila sin eliminarlo
39     int peek() const {
40         if (top < 0) { // Verifica si la pila está vacía
41             cout << "Pila vacia. " << endl; // Muestra mensaje si no hay elementos
42             return -1; // Retorna un valor de error
43         }
44         return arr[top]; // Retorna el valor en la cima
45     }
46 };
47
48 // Función principal del programa
49 int main() {
50     PilaEstática pila; // Se crea un objeto de tipo PilaEstática
51
52     pila.push(100); // Inserta 100 en la pila
53     pila.push(200); // Inserta 200 en la pila
54     pila.push(300); // Inserta 300 en la pila
55
56     cout << "Elemento de la cima: " << pila.peek() << endl; // Muestra el elemento en la cima (300)
57     cout << "Elemento eliminado: " << pila.pop() << endl; // Elimina y muestra el elemento en la cima (300)
58     cout << "Elemento eliminado: " << pila.pop() << endl; // Elimina y muestra el siguiente elemento (200)
59
60     return 0; // Fin del programa
61 }
62
63
```

## Compilación

```
Consola de depuración de Microsoft Visual Studio
Elemento de la cima: 300
Elemento eliminado: 300
Elemento eliminado: 200

E:\Tareas\Programacion trabajo 2\Tarea semana 9\x64\debug\Tarea semana 9.exe (proceso 7480) se cerró con el código 0 (0x0).
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. . .
```

### Pila Dinámica, Código Comentado:

```
Tarea Semana 9 Progra.cpp - [Arquivo global]
1  #include <iostream>
2  using namespace std;
3  struct Nodo { //Definición de la estructura Nodo que forma la base de la pila
4      int dato; //Almacena el valor del elemento en la pila
5      Nodo* siguiente; // Puntero al siguiente nodo en la pila
6  };
7
8  class PilaDinamica { //Clase que implementa una pila usando memoria dinámica
9  private: //Indica que es de clase privada, que solo pocos tienen acceso a ella
10     Nodo* cima; //Apunta al nodo en la cima de la pila
11
12 public: //Contrario a la privada, es una clase pública, casi todos tienen acceso a ella
13     // Constructor - inicializa una pila vacía
14     PilaDinamica() : cima(nullptr) {} //Esto nos indica que no hay nodos inicialmente
15
16     ~PilaDinamica() { // Destructor - libera toda la memoria de los nodos al destruir la pila
17         while (cima != nullptr) { // Funciona mientras quedan nodos
18             pop(); //Esto nos ayuda a eliminar nodos uno por uno
19         }
20     }
21
22     void push(int valor) { // Método para insertar un elemento en la cima de la pila llamado "push"
23         Nodo* nuevo = new Nodo; //Crea un nuevo nodo
24         nuevo->dato = valor; //Asigna el valor al nodo
25         nuevo->siguiente = cima; //El nuevo nodo apunta a la antigua cima
26         cima = nuevo; //Actualiza la cima a este nuevo nodo
27     }
28
29     int pop() { // Método para eliminar y devolver el elemento de la cima llamada "pop"
```

```
Tarea Semana 9 Progra.cpp - x
Tarea Semana 9 Progra (Arreglo global)
10
11 if (cima == nullptr) { //Encargada de verificar si la pila está vacía
12     cout << "Pila vacía (underflow)." << endl; //Tira este mensaje en caso de pila vacía
13     return -1; //Retorna el valor de error
14 }
15 Node* temp = cima; //Esta guarda referencia al nodo a eliminar
16 int valor = temp->dato; //Guarda el valor antes de eliminar
17 cima = cima->siguiente; //La nueva cima es el siguiente nodo
18 delete temp; //Libera la memoria del nodo removido
19 return valor; //Devuelve el valor guardado
20
21
22 //Método para observar el elemento en la cima sin eliminarla (peek/top)
23 int peek()const { //El const nos indica que no modifica la pila
24     if (cima == nullptr) { //Verifica si la pila está vacía
25         cout << "Pila vacía." << endl; //Mensaje en caso de que la pila este vacía
26         return -1; //Retorna valor de error
27     }
28     return cima->dato; //Devuelve el dato en la cima
29 }
30
31
32 //Este int main es la función principal para demostrar el uso de la pila
33 int main() {
34     PilaDinamica pila; //Crea una instancia para la pila
35     pila.push(300); //Inserta 300 en la pila
36     pila.push(200); //Insertar 200 en la pila
37     pila.push(300); //Insertar 300 en la pila
38     cout << "Elemento en la cima:" << pila.peek() << endl; //Debería mostrar 300
39     cout << "Elemento eliminado:" << pila.pop() << endl; //Debería mostrar 300
40     cout << "Elemento eliminado:" << pila.pop() << endl; //Debería mostrar 200
41     return 0; //Retorna un valor de 0 XO
42 }
```

Compilación:

```
Console de depuración de Mi - x
Elemento en la cima:300
Elemento eliminado:300
Elemento eliminado:200

C:\Users\pasev\source\repos\Tarea Semana 9 Progra\x64\Debug\tarea Semana 9 Progra.exe (proceso 1568) se cerró con el código 0 (0x0).
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. . . .
```

Link del Repositorio:

<https://github.com/E-lev10/Tarea-Semana-9-progra.git>