Universidad Autónoma de Baja California
Facultad de Ciencias Químicas e Ingeniería
Algoritmos y Estructuras de Datos
Ciclo 2020-1

Práctica 2
"Estructuras LIFO Estáticas"

Profesor(a): Thelma Violeta Ocegueda Miramontes

Grupo: 552

Alumnos:	Matrícula:
• Cota Robledo Benjamín	1225836
• Rodríguez Muñoz José Luis	1260368

Tijuana, Baja California, a 11 de febrero de 2020

CÓDIGO EN C

```c
#include <stdio.h> // Librería principal
#include <conio.h> // Uso del getch
#define N 10 // Se define N y se usara para el tamaño del arreglo

typedef struct // typedef es una palabra cuya funcion es darle un nombre a un tipo de dato, en este caso una estructura cuyo nombre tiene Pilas
{
    int datos[N]; // El tamaño del arreglo de los datos
    int tope, top; // Tope se refiere a la primera pila mientras que top a la segunda pila
} Pilas; // El nombre de la estructura

/* Funciones prototipo */
void push(Pilas * pilas, int, int); // Funcion que mete un dato dentro de la pila
void pop(Pilas * pilas, int); // Funcion que saca un dato que se encuentra al tope de la pila
void mostrarPilas(Pilas * pilas); // Muestra ambas pilas

int main()
{
    Pilas pilas; // Se declara una variabla llamada pilas de tipo Pilas
    pilas.tope = - 1; // El tope de la primera pila empieza en -1
    pilas.top = N; // El tope de la segunda pila empieza en N para que tome el otro extremo ya que ambas pilas trabajan en el mismo vector
    char opcion; // Opcion para el menú
    int pila, dato; // pila es para elegir con que pila estaremos trabajando, dato es el dato a introducir en la funcion push

    do // Hace mientras que que ...
    {
        //system("cls"); // Limpia pantalla
        printf("\n\t Estructuras LIFO Est%cticas", 160); // Despliega en pantalla el menu
        printf("\n\n 1) Push");
        printf("\n 2) Pop");
        printf("\n 3) Mostrar pilas");
        printf("\n 4) Salir");
        printf("\n\n\t Introduce una opci%cn: ", 162);
        opcion = getchar(); // Pide que se introduzca un caracter al usuario

        switch(opcion) // Tomando la opción que se introdujo por el usuario
        {
        case '1': // Opción 1 realizará push a la pila que se eliga
            //system("cls"); // Limpia pantalla
            printf("\n 1) Primer pila");
            printf("\n 2) Segunda pila");
            printf("\n\n\t Elige una pila: ");
            scanf("%d", &pila); // Se pedira al usuario ingresar con que pila quiere trabajar
            //system("cls");
            printf("\n Dato que desea ingresar: ");
```

```
            scanf("%d", &dato); // Se pedira al usuario que dato desea
introducir a la pila
            push(&pilas, dato, pila); // La función push recibe 3 parámetros,
la estructura, el dato a introducir y que pila se quiere trabajar
            printf("\n\n\t Push!");
            getch(); // Pide teclazo para continuar
            break; // Se cierra el case

        case '2': // Opción 2 realizara pop a la pila que se eliga
            //system("cls"); // Limpia pantalla
            printf("\n 1) Primer pila");
            printf("\n 2) Segunda pila");
            printf("\n\n\t Elige una pila: ");
            scanf("%d", &pila); // Se pedira al usuario que pila quiere
realizar la función pop
            pop(&pilas, pila); // La función pop tiene 2 parámetros, la
estructura y la pila que se quiere trabajar
            getch(); // Pide teclazo para continuar
            break; // Se cierra el case

        case '3': // Opción 3 muestra el contenido de ambas pilas
            //system("cls"); // Limpia pantalla
            mostrarPilas(&pilas); // La función solo recibe como parámetro la
estructura
            getch(); // Pide teclazo para continuar
            break; // Se cierra el case
        }
    } while(opcion != '4' && opcion != EOF); // Opción sea diferente de 4 o
que opción sea diferente al fin de archivo
      {
          //system("cls"); // Limpia pantalla
          printf("\n\t Cierre exitoso ...");
      }
}

void push(Pilas * pilas, int dato, int opcion) // Función push que recibe la
estructura, el dato a introducir y la pila que se quiere trabajar
{
    if(opcion == 1) // Si opción es 1 se trabaja con la primera pila
    {
        if(pilas -> tope != N - 1 && pilas -> tope != pilas -> top) // Si el
tope de la primera pila es N-1 y el tope es diferente al tope de la segunda
pila entonces la pila no está llena y se puede introducir el dato
        {
            pilas -> tope = pilas -> tope + 1; // Se le aumenta 1 al tope
            pilas -> datos[pilas -> tope] = dato; // El tope tomara el valor
introducido
        }
        else // De lo contrario la pila está llena
        {
            printf("\n Pila llena ...");
        }
    }
    else // Checando si se introdujo otra opción ...
        if(opcion == 2) // Si opción es 2 se trabaja con la segunda pila
        {
```

```
            if(pilas -> top != 0 && pilas -> top != pilas -> tope + 1) // Si
el tope de la segunda pila es diferente de 0 y es diferente al tope de la
primera pila + 1 entonces
            {
                pilas -> top = pilas -> top - 1; // Se le resta 1 al tope
                pilas -> datos[pilas -> top] = dato; // El tope tomara el
valor introducido
            }
            else // De lo contrario la pila está llena
            {
                printf("\n Pila llena ...");
            }
        }
        else // Si se introdujo una opción que no existe se despliega en
pantalla el siguiente mensaje y regresara al menú
        {
            printf("\n Opci%cn inv%clida ...", 162, 160);
        }
}

void pop(Pilas * pilas, int opcion) // La función pop recibe la estructura y
la opción que define con que pila se va a trabajar
{
    if(opcion == 1) // Si opción es 1 entonces se trabajara con la primer
pila
    {
        if(pilas -> tope != - 1) // Si el tope de la pila es diferente de -1
entonces la pila no está vacia
        {
            printf("\n\n\t Pop: %d", pilas -> datos[pilas -> tope]); // Se
saca el dato que se encuentra al tope y se despliega en pantalla
            pilas -> tope = pilas -> tope - 1; // Y se le resta -1 al tope
        }
        else // De lo contrario la pila está vacía
        {
            printf("\n Pila vac%ca ...", 161);
        }
    }
    else // Checando si se introdujo otra opción ...
        if(opcion == 2) // Si opción es 2 entonces se trabajara con la
segunda pila
        {
            if(pilas -> top != N) // Si el tope es diferente de N entonces la
pila no está vacía
            {
                printf("\n\n\t Pop: %d", pilas -> datos[pilas -> top]); // Se
saca el dato que se encuentra al tope y se despliega en pantalla
                pilas -> top = pilas -> top + 1; // Y se le suma +1 al tope
            }
            else // De lo contrario la pila está vacía
            {
                printf("\n Pila vac%ca ...", 161);
            }
        }
        else // Si se introdujo una opción que no existe se despliega en
pantalla el siguiente mensaje y regresara al menú
        {
```

```
            printf("\n Opci%cn inv%clida ...", 162, 160);
        }
}

void mostrarPilas(Pilas * pilas) // La función mostrarPilas solo recibe la estructura a mostrar
{
    int i; // Una variable i que funcionara como índice para ambas pilas
    printf("\n\t Pila 1"); // Se despliega en pantalla que se mostrara la pila 1 primero

    for(i = 0; i != pilas -> top && i <= pilas -> tope; i++) // Ciclo for en donde se inicializa i como 0, mientras que i sea diferente del tope de la segunda pila y sea menor o igual al tope de la primera pila, se le sumara 1 a i finalizando el ciclo
    {
        printf("\n [ %d ] ", pilas -> datos[i]); // Se imprime en pantalla el dato que se encuentra en esa posición de i
    }
    printf("\n\n\t Pila 2"); // Se despliega en pantalla que se mostrara la pila 2

    for(i = N - 1; i != pilas -> tope && i >= pilas -> top; i--) // Ciclo for en donde se inicializa i como N - 1, mientras que i sea diferente al tope de la primera pila e sea mayor o igual al tope de la segunda pila, se le restara 1 a i finalizando el ciclo
    {
        printf("\n [ %d ] ", pilas -> datos[i]); // Se imprime en pantalla el dato que se encuentra en esa posición de i
    }
}
```

CONCLUSIONES

Cota Robledo Benjamín: En esta práctica utilice como primera vez una sola estructura para trabajar dos pilas usando el mismo arreglo, al principio no sabía cómo atacar el problema hasta que encontré que cada tope puede tomarse como una pila independiente, se me hizo interesante ya que se puede representar como el tener una sola memoria y se tienen dos procesos a la vez independientes, si uno come todo el espacio de memoria entonces el otro no podrá trabajar o más bien no podrá utilizar todos los recursos de la memoria.

Rodríguez Muñoz José Luis: Este ejercicio/práctica reforcé los conceptos y además la implementación en código de las estructuras LIFO estáticas, aplicando las operaciones básicas de las pilas, pero ahora, utilizando dos pilas con una sola estructura, esto quiere decir que se utilizó el mismo vector para contener los datos de las dos pilas. Es interesante ya que en la primera práctica solo se hizo con una pila, y al momento de pensar con mi compañero el cómo podía hacerse con dos pilas fue un poco complicado, pero se logró el objetivo o bien, competencia de la práctica.