Universidad Naciona Autónoma de México

Facultad de Ingeniería

Estructuras de Datos y Algoritmos I

Actividad: Pilas

Sánchez Hernández Marco Antonio

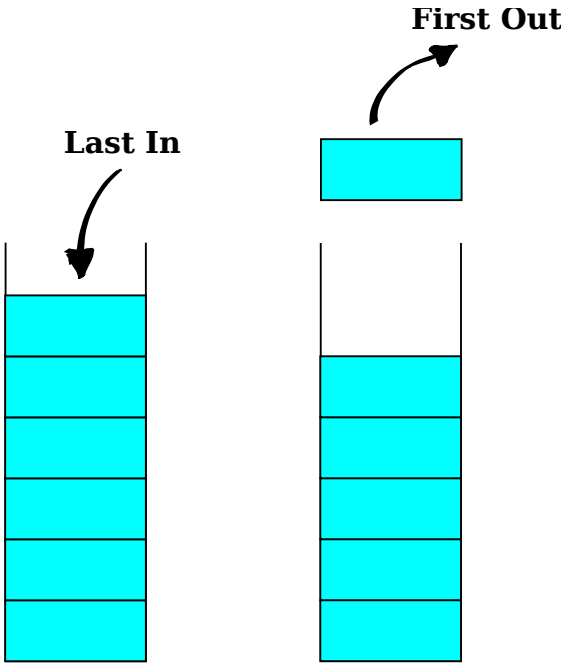
Fecha: 11/junio/2021

¿Qué es una pila?

Una pila es un tipo de estructura de datos lineal,[1] propuesta por primera vez por el computólogo alemán Mr. F. L. Bauer, razón por la que en 1988 recibió el *IEEE Computer Society Pioneer Award*. Los elementos y las operaciones en una pila se rigen bajo el principio, el útilmo que entra es el primero que sale (***LIFO***, por sus siglas en inglés) **[Figura 1.1]**, esto significa que, una vez se inserten varios elementos en una pila, solo se podrá acceder y ejecutar operaciones sobre el último elemento ingresaso (**tope**).

Figura 1.1

El último elemento que entra, es el primero que sale.



Nota. La imagen representa un contenedor con paredes, con el fin de ejemplificar gráficamente una pila, por lo que no es posible accerder mas que al elemento superior.

Operaciones una pila

Push: esta operación permite insertar un nuevo elemento en una pila, cada que un nuevo elemento es añadido, la posición del **tope** se incrementa en uno. Si una pila se encuentra llena, y

1 Los elementos pertenecientes a esta clase de estructura de datos son almacenados linealmente uno después de otro.

se trata de insertar un nuevo elemento, entonces se generará una condición conocida en inglés como ***overflow***.

Pop: esta operación permite eliminar el último elemento que fue añadido a una pila, cada que un elemento es eliminado, la posición del **tope** decrece en uno. Si una pila se encuentra vacía, y se trata de ejecutrar esta operación, resultará en una condición conocida en inglés como ***underflow***.

Peek/Top: en una pila no vacía, regresará el valor del elemento **tope**. Si una pila está vacía, la operación **peek/top** generará la condición ***underflow***.

Empty: esta operación verifica si una pila se encuentra vacía o no, regresará como valor verdadero si la pila se encuentra vacía.

Algunos autores añaden una quinta operación nombrada ***Peep***, la cual permite acceder a la información almacenada en la *enésima* posición de una pila, sin embargo, esta no será tomada en cuenta en este archivo, ya que no todas las fuentes consultadas la consideran una operación válida sobre una pila.

Implementación de una pila

El presente código escrito en lenguaje C muestra la implementación de una pila y las cuatro operaciones expuestas anteriormente, por conveniencia, el código fue dividio en dos archivo, *main.c* y *stack.h*:

Código del archivo *main.c*

```
#include <stdio.h>
#include <stdlib.h>
#include “stack.h”

int main() {

   int x;
   unsigned short opcion;
   initstack(); //Se inicializa una pila P.

   do {

      opcion = 0;
```

```
      printf(“\n\n\t\t***Implementaci\u00F3n de una Pila en lenguaje C***\n\n”);
      printf(“1. Push\n”);
      printf(“2. Pop\n”);
      printf(“3. Empty\n”);
      printf(“4. Peek\n”);
      printf(“5. Salir\n\n”);
      scanf(“%hu”, &opcion);

      switch(opcion) {

         case 1:
            printf(“Ingrese un n\u00FAmero entero: ”);
            scanf(“%d”, &x);
            push(x);
         break;

         case 2:
            x = pop();
            printf(“%d ha sido eliminado\n\n”, x);
         break;

         case 3:
            /*Si no existe algún elemento en la pila se mostrará este mensaje*/
            if (emptystack())
               printf(“La pila est\u00E1 vac\u00EDa\n\n”);

            else {
               /*Si la pila está llena, se muestra este mensaje.*/
               if (P.tope == TAMANIO-1) {
                  printf(“La pila est\u00E1 llena\n\n”);
                  break;
               }
                /*Si la pila contiene elementos, pero no está llena, se muestra este mensaje,
acompañado del número de elementos que hay en la pila.*/
               printf(“La pila contiene %d elementos\n\n”, (P.tope+1));
            }
         break;

         case 4:
            x = peek();
```

```
            printf(“Elemento tope: %d\n\n”, x);
         break;

         case 5:
            return 0;

         default:
            printf(“ERROR: la opci\u00F3n ingresada no es v\u00E1lida\n\n”);
         break;

      }

   } while(opcion != 5);

   return 0;
}
```

Código del archivo *stack.h*

```
#include <stdio.h>
#include <stdlib.h>

/*Definición de constantes.*/
#define TAMANIO 10 //Define el tamaño del arreglo utilizado para una pila.
#define VERDADERO 1
#define FALSO 0

/*Definición del tipo de dato pila a partir de una estructura.*/
struct pila {
   int elemento[TAMANIO];
   int tope; //Define el índice del último elemento.
} P;

/*Prototipo de la funcion initstack.*/
void initstack();

/*Prototipo de la funcion emptystack.*/
int emptystack();

/*Prototipo de la funcion push.*/
void push();
```

```
/*Prototipo de la funcion pop.*/
int pop();

/*Prototipo de la funcion peek.*/
int peek();

/*Definición de la función initstack, la cual inicializa una pila.*/
void initstack() {
   P.tope = -1; //Asigna el valor -1 a tope para indicar que la pila está vacía.
}

/*Definición de la función emptystack, la cual implementa la operación empty.*/
int emptystack() {

   if (P.tope == -1)
      return VERDADERO;

   else
      return FALSO;

}

/*Definición de la función push, la cual implementa la operación push.*/
void push(int element) {

/*Si se intenta añadir un nuevo elemento cuando la lista está llena se muestra la condición
Stack Overflow y termina el programa.*/
   if (P.tope == TAMANIO-1) {
      printf("\nStack Overflow\n");
      exit(1);
   }

   P.tope++;
   P.elemento[P.tope] = element;

}

/*Definición de la función pop, la cual implementa una pop.*/
int pop() {
```

```
   int x; //Variable local.

   /*Se comprueba si la pila contiene elementos o no.*/
   if (emptystack()) {
      printf("\nStack Underflow\n");
      exit(1);
   }

   x = P.elemento[P.tope];
   P.tope--;

   return x;
}

/*Definición de la función peek, la cual implementa una peek.*/
int peek() {

   /*Se comprueba si la pila contiene elementos o no.*/
   if (emptystack()) {
      printf("\nStack Underflow\n");
      exit(1);
   }

   return P.elemento[P.tope];
}
```

Referencias

GeekforGeeks. (2019, 15 de octubre). *Stack Data Structure*. https://www.geeksforgeeks.org/stack-data-structure/

Kumar A. (2020). *A practial Approach to Data Structure and Algorithm with Programming in C*. Arcler Press. https://www-bibliotex-com.pbidi.unam.mx:2443/pdfreader/practical-approach-to-data-structure-algorithm-programming-in-c

Kumar A. y Kumar V. (2019). *Data Structures in C Programming*. Arcler Press. https://bibliotex-ipublishcentral-com.pbidi.unam.mx:2443/pdfreader/data-structures-c-programming