

Universidad de Guanajuato División de Ingenierías  
Campus Irapuato Salamanca (DICIS)

Algoritmos y estructura de datos  
Carlos Hugo García Capulín

Tarea No. 9  
Reporte Estructura pila

Jair Chávez Islas  
28/Octubre/2021

# Problema

Una Pila es un tipo de dato abstracto del tipo LIFO (Last-In-First-Out), es decir, el último dato en entrar es el primero en salir, su finalidad es para almacenar datos que estaban en procesamiento pero que por algún motivo se debe suspender, y se deben almacenar para posteriormente continuar con su procesamiento.

La forma más simple de implementar esta estructura es considerando que los datos estarán siendo agrupados en forma de un arreglo: de tal forma que al ir ingresando datos a la pila el primero en ingresar queda al fondo de la pila (dato D1), los siguientes se van acomodando encima del anterior (datos D2 y D3), de tal forma que el último en entrar queda en la parte superior (dato D4), al sacar un elemento de la pila el dato que se regresa es el último que entró (dato D4).



Continuando con el programa que comenzamos a hacer en clase con el maestro, precisamente comenzamos a hacer el de pila, y ya tenemos la mayor parte del programa solo nos faltan las funciones IsEmpty y Pop, la función pop es la que sacará de la pila el último dato agregado y la función IsEmpty es la que nos indicará cuando la pila esté vacía, estas funciones son las que haremos para este reporte.

Teniendo el ejemplo de que hagamos una pila de 5 datos que sean los siguientes 10,20,30,40,50

Quedarían impresos de la siguiente manera:

50  
40  
30  
20  
10

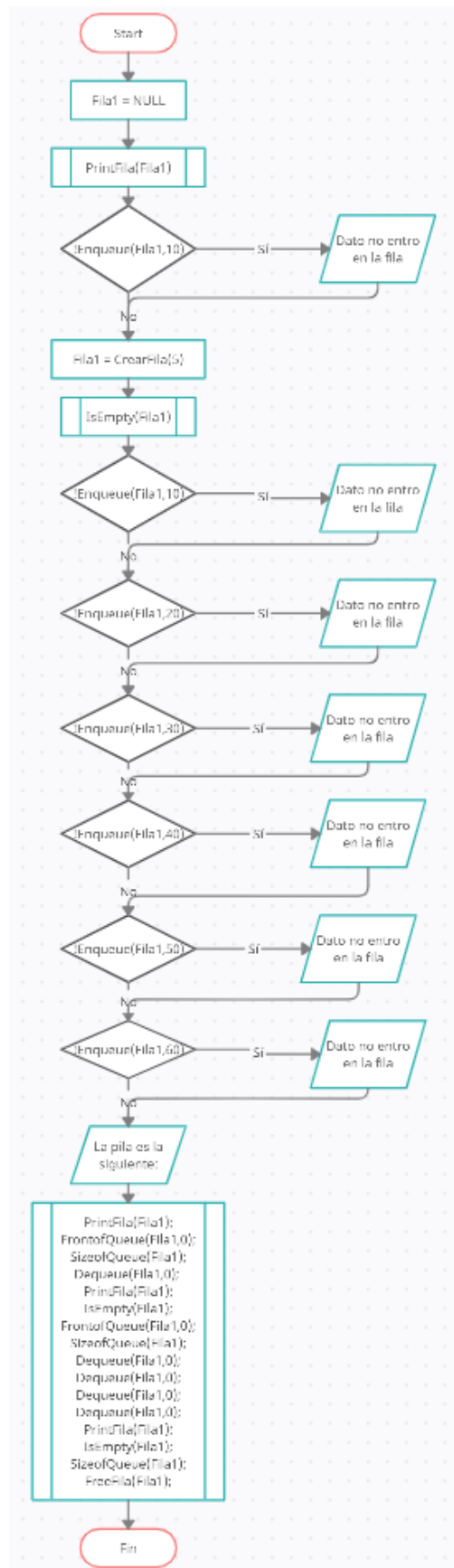
Al eliminar un dato con la función pop, se iría el 50 ya que es el último elemento que ingresamos, entonces al volver a imprimir los datos quedaría de la siguiente manera

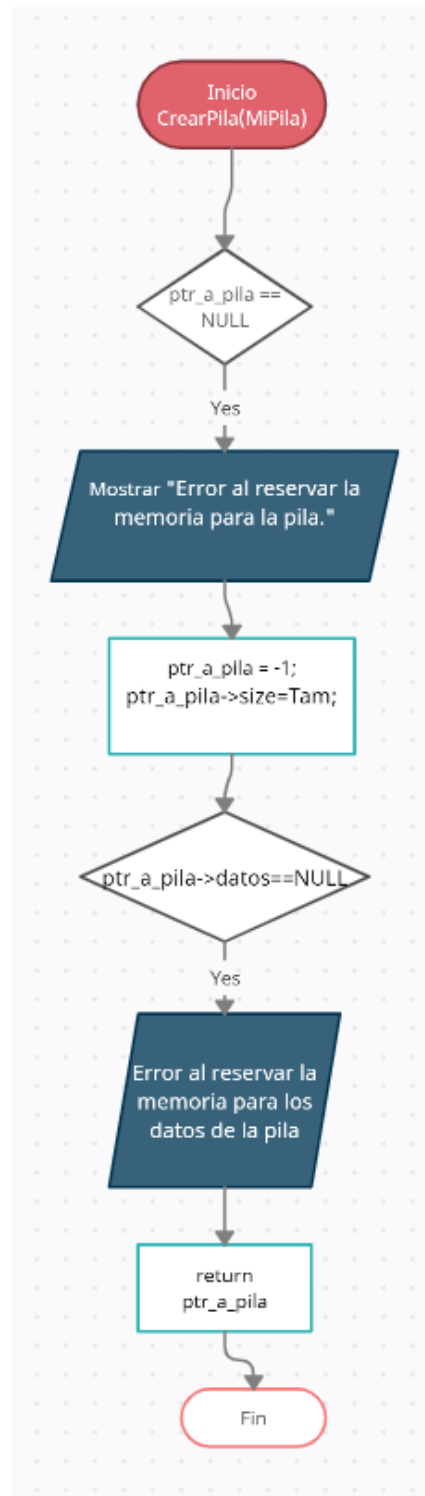
40  
30  
20  
10

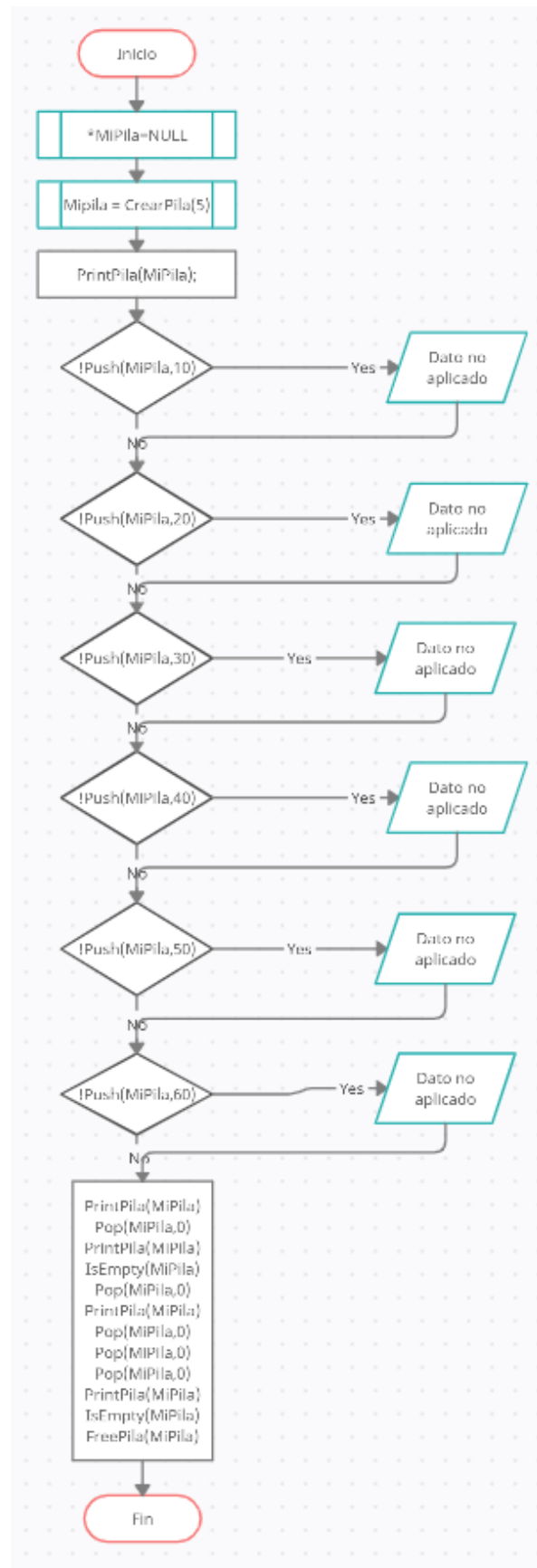
Pero aún tenemos nuestra pila con datos, por lo tanto, no está vacía, pero ya que hayamos eliminado con la función otros 4 datos, quedaría nuestra pila vacía así que deberíamos tener un mensaje para cuando eso pase.

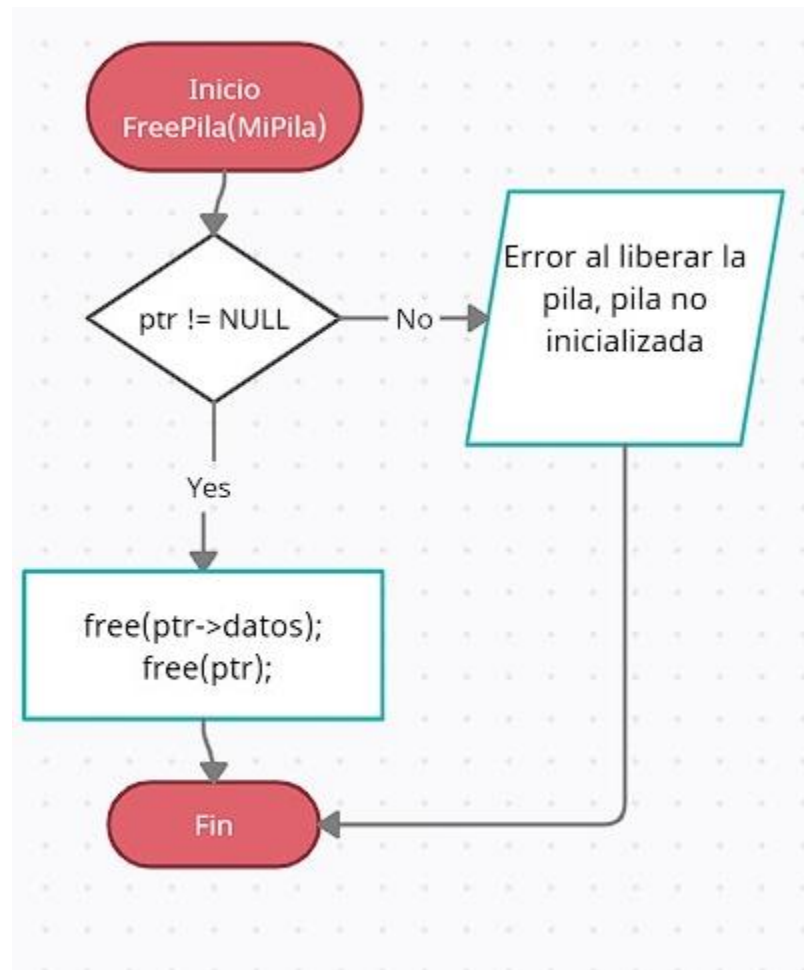
# Solución implementada

Diagrama del programa

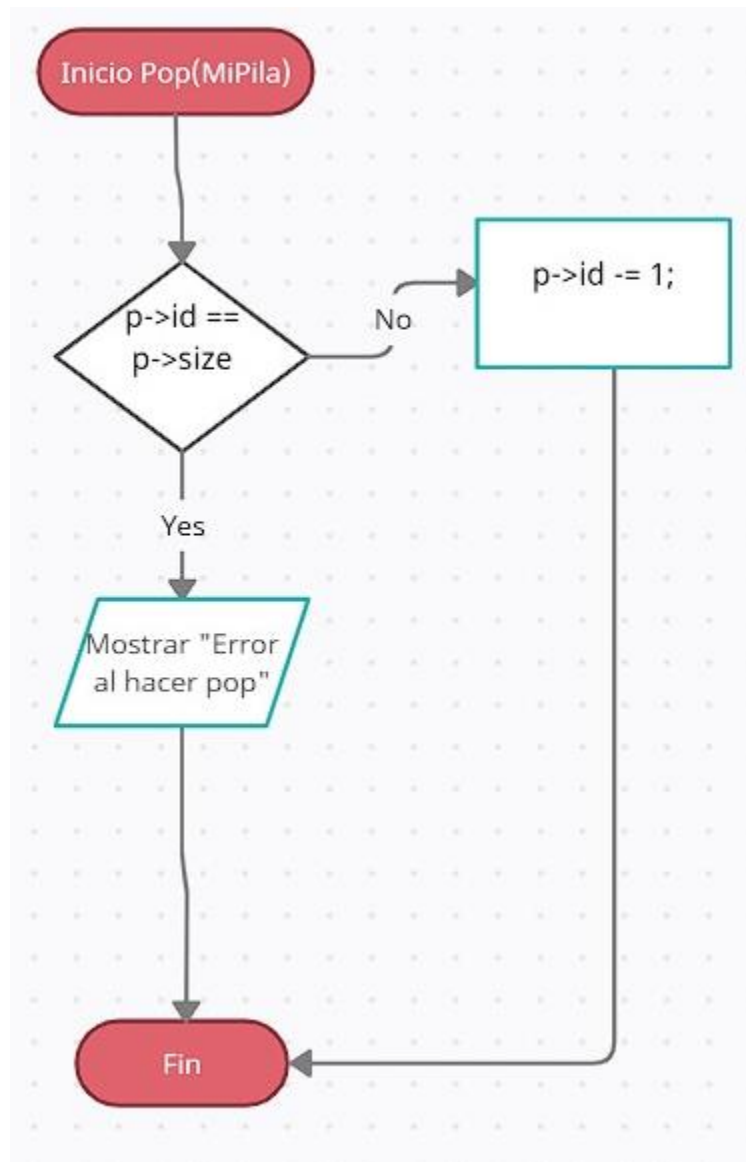


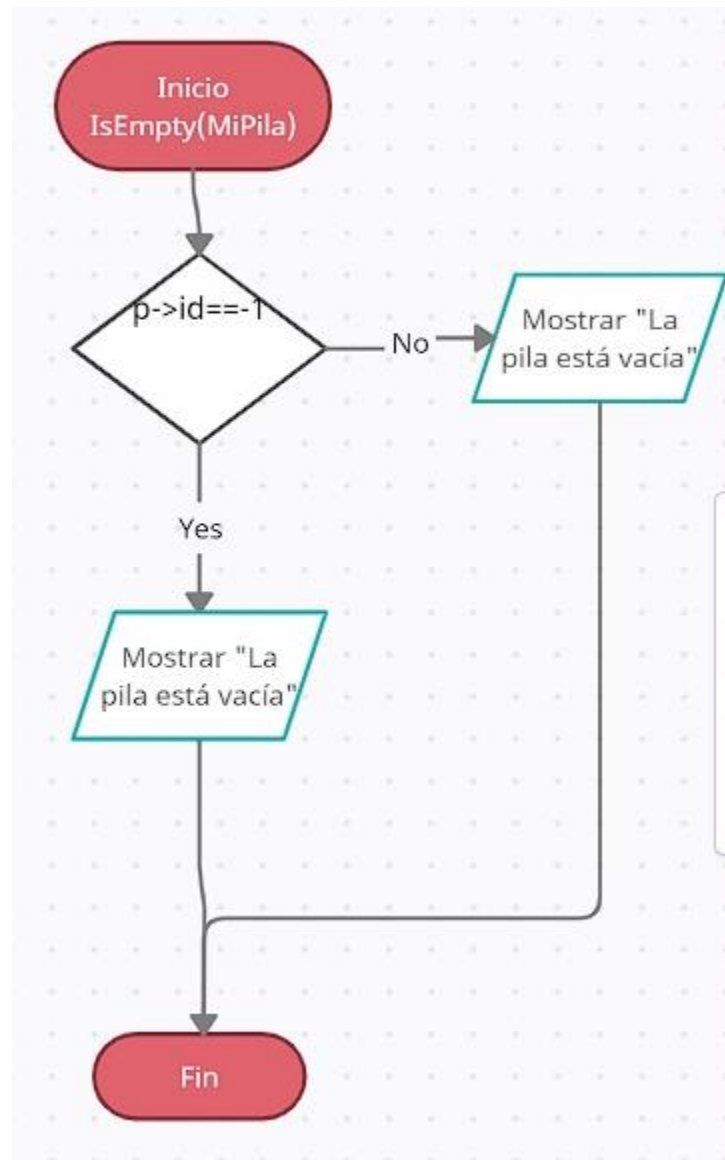












**Código comentado del programa**

```

1 //Agregamos las librerias necesarias para las funciones que necesitamos
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 //Declaramos una estructura
6 typedef struct
7 {
8     //Miembros de la estructura
9     int *datos;
10    int id;
11    unsigned int size;
12 }PILA; //Nombre de la estructura
13
14 //Aqui ponemos los prototipos de las funciones utilizadas
15 PILA* CrearPila(unsigned int Tam);
16 void FreePila(PILA *ptr);
17 char Push(PILA *ptr, int dato);
18 char IsFull(PILA *ptr);
19 void PrintPila(PILA *ptr);
20 char Pop(PILA *p, int*d);
21 char IsEmpty(PILA *p);
22
23 //Inicializamos la funcion principal
24 int main()
25 {
26     //Declaracion de variables de esta funcion
27     PILA *MiPila=NULL; //Declaramos la variable MiPila de tipo PILA
28
29     MiPila = CrearPila(5); //Guardamos en mi pila la funcion crearpila con 5 elementos
30     PrintPila(MiPila); //Imprimimos la pila
31
32     //Aplicamos la función push a mipila, con el valor 10
33     if(!Push(MiPila,10))
34     {
35         printf("\n Dato no aplicado");
36     }
37     //Aplicamos la función push a mipila, con el valor 20
38     if(!Push(MiPila,20))
39     {
40         printf("\n Dato no aplicado");
41     }
42     //Aplicamos la función push a mipila, con el valor 30
43     if(!Push(MiPila,30))
44     {
45         printf("\n Dato no aplicado");
46     }
47     //Aplicamos la función push a mipila, con el valor 40
48     if(!Push(MiPila,40))
49     {
50         printf("\n Dato no aplicado");
51     }

```

```

50     printf("\n Dato no aplicado");
51 }
52 //Aplicamos la función push a mipila, con el valor 50
53 if(!Push(MiPila,50))
54 {
55     printf("\n Dato no aplicado");
56 }
57 //Aplicamos la función push a mipila, con el valor 60
58 if(!Push(MiPila,60))
59 {
60     printf("\n Dato no aplicado");
61 }
62 //Imprimilos la pila con los valores a gregados
63 PrintPila(MiPila);
64 //Eliminamos un valor de la pila
65 Pop(MiPila,0);
66 //Volvemos a imprimir la pila para ver los cambios aplicados
67 PrintPila(MiPila);
68 //Comprobamos si la pila esta vacia
69 IsEmpty(MiPila);
70 //Eliminamos los valores restantes de la pila
71 Pop(MiPila,0);
72 Pop(MiPila,0);
73 Pop(MiPila,0);
74 Pop(MiPila,0);
75 //Imprimimos la pila una vez mas
76 PrintPila(MiPila);
77 //Comprobamos si la pila esta vacia
78 IsEmpty(MiPila);
79 //Liberamos la memoria de la fila
80 FreePila(MiPila);
81 //Se imprime un salto de linea
82 printf("\n");
83 return 0;//se retorna el valor de 0 para verificar que haya finalizado sin problemas
84 }
85
86 //Inicialziamos la funcion Pop
87 char Pop(PILA *p, int*d)
88 {
89     //Se declaran las variables locales de la funcion
90     char Flag;
91
92     //Si no se ha inicializado la pila, no se puede hacer pop
93     if(p->id == p->size)
94     {
95         printf("\nError al hacer pop, pila no inicializada");
96         Flag = 0;
97     }
98     else
99     {
100         //Se elimina el ultimo valor agregado

```

```

101         p->id -= 1;
102         Flag = 1;
103     }
104     printf("\ndespues de hacer el pop: ");
105     return Flag;
106 }
107
108 //Inicializamos la funcion IsEmpty
109 char IsEmpty(PILA *p)
110 {
111     //Comprobamos si la pila esta vacia
112     if(p->id == -1)
113     {
114         printf("\nLa pila esta vacia");
115     }
116     else
117     {
118         printf("\nla pila tiene datos aun");
119     }
120 }
121
122 void PrintPila(PILA *ptr)
123 {
124     //Se declaran las variables locales de la funcion
125     int k;
126
127     //Si no se ha creado la pila, no podemos imprimir
128     if(ptr==NULL)
129     {
130         printf("\n Error al imprimir: la pila no esta inicializada");
131         return;
132     }
133     //Se imprime valor a valor desde el primer dato introducido hasta el ultimo
134     for(k=ptr->id;k>=0;k--)
135         printf("\n%i",ptr->datos[k]);
136 }
137
138 //Se inicializa la funcion IsFull
139 char IsFull(PILA *ptr)
140 {
141     //Se verifica si se lleno la fila segun el numero de datos que recibe
142     if(ptr->id == (ptr->size-1))
143         return 1;
144     else
145         return 0;
146 }
147
148 //Se inicializa la funcion push
149 char Push(PILA *ptr, int dato)
150 {
151     //Se declaran las variables locales de la funcion

```

```

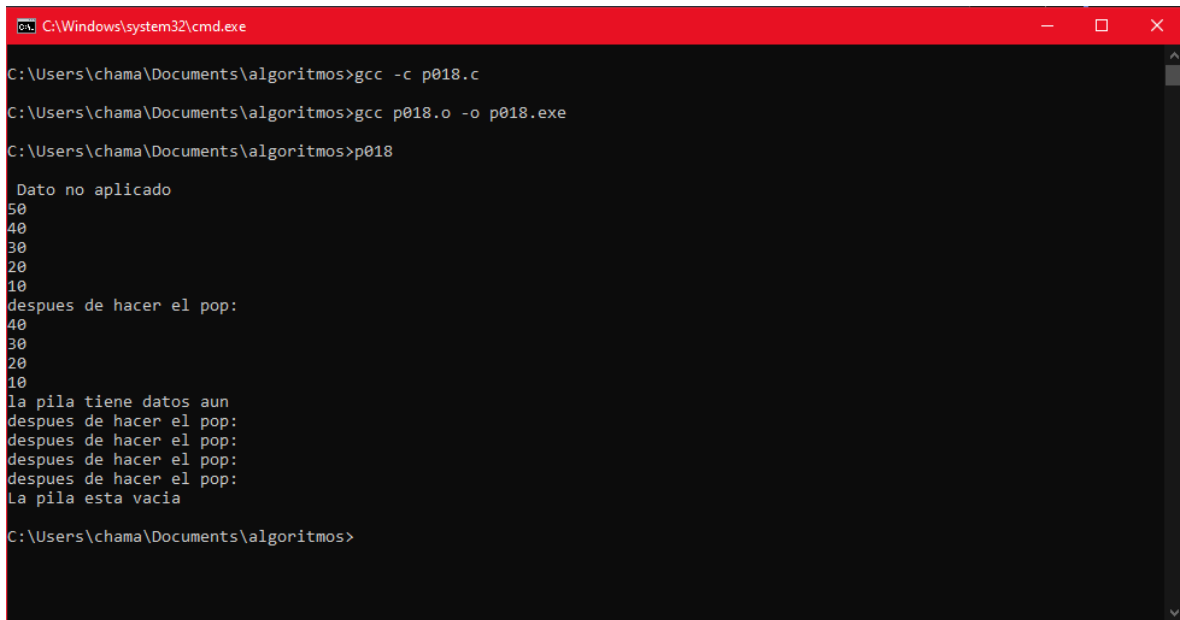
149 char Push(PILA *ptr, int dato)
150 {
151     //Se declaran las variables locales de la funcion
152     char Flag = 0;
153
154     //Si no se ha creado la pila, no podemos hacer push
155     if(ptr==NULL)
156         printf("\n Error: la pila no esta inicializada");
157     else if(!IsFull(ptr))
158     {
159         //Se agrega el dato que ingresamos en la funcion main
160         ptr -> id++;
161         ptr -> datos[ptr->id] = dato;
162         Flag = 1;
163     }
164     return Flag;
165 }
166
167 //Se inicializa la variable crearPila
168 PILA* CrearPila(unsigned int Tam)
169 {
170     //Declaracion de variables de esta funcion
171     //Se declara una variable ptr_a_pila de tipo PILA
172     PILA *ptr_a_pila;
173
174     //Se reserva memoria
175     ptr_a_pila = (PILA *)malloc(sizeof(PILA));
176
177     //Error al reservar la memoria
178     if(ptr_a_pila==NULL)
179     {
180         printf("Error al reservar la memoria para la pila.");
181         exit(0);
182     }
183
184     //Se define el tamaño de la pila al momento de crearla
185     ptr_a_pila->id=-1;
186     ptr_a_pila->size=Tam;
187     //Se reserva memoria
188     ptr_a_pila->datos=(int *)malloc(Tam*sizeof(int));
189
190
191     if(ptr_a_pila->datos==NULL)
192     {
193         //Error al reservar la memoria
194         printf("Error al reservar la memoria para los datos de la pila");
195         exit(0);
196     }
197     //Se retorna la variable
198     return ptr_a_pila;
199 }

```

```
199 }
200
201 //Se inicializa la funcion Freepila
202 void FreePila(PILA *ptr)
203 {
204     //Si ptr es distinto de null
205     if(ptr!=NULL)
206     {
207         //Se libera la memoria de la pila
208         free(ptr->datos);
209         free(ptr);
210     }
211     //Algun error al intentar liberar la memoria de la pila
212     else
213         printf("\n Error al liberar la pila, pila no inicializada");
214 }
```

# Pruebas y resultados

## Evidencia del programa



```
C:\Windows\system32\cmd.exe
C:\Users\chama\Documents\algoritmos>gcc -c p018.c
C:\Users\chama\Documents\algoritmos>gcc p018.o -o p018.exe
C:\Users\chama\Documents\algoritmos>p018

Dato no aplicado
50
40
30
20
10
despues de hacer el pop:
40
30
20
10
la pila tiene datos aun
despues de hacer el pop:
despues de hacer el pop:
despues de hacer el pop:
despues de hacer el pop:
La pila esta vacia
C:\Users\chama\Documents\algoritmos>
```

Primero utilicé la función `printpila` y me salieron los números del 50 al 10, hice `pop` una vez para verificar que funcionaba la función y si, se eliminó el ultimo valor agregado que era el 50 así que ahora tenemos del 40 al 10, y luego utilicé la función `IsEmpty` para verificar si aun habían valores en la pila y si, aun habían, entonces utilice la función `pop` 4 veces mas para eliminar todos los valores y una vez más utilice la función `IsEmpty` y me imprime un mensaje de que ahora la pila está vacía.

Y el resultado es justo como lo habíamos previsto en el planteamiento del problema.