# Capítulo II Estructura de Datos Pilas.

## Contenido:

- Objetivo.
- Introducción.
- Definición de Pilas
- Ejemplo de Pilas en la vida Real
- Representación Gráfica de una Pila
- Estructura de Datos de una Pila.
- Operaciones Básicas de una Pila.
- Explicación de las operaciones Básicas de una Pila.
- Representación de las Pilas.
- Aplicaciones de las Pilas.
- Conclusiones
- Tarea Práctica
- Orientación para el Laboratorio.

## Objetivo.

- Que los estudiantes aprendan el trabajo con las pilas.

## Introducción

En este capítulo se estudian la estructura de datos conocida con el nombre de Pilas, en el mismo se verá que son, cómo se representan, sus operaciones básicas, y algunas aplicaciones de las mismas en la informática. El contenido está organizado de lo más simple a lo más complejo. Se utiliza el lenguaje de programación C++, utilizando el entorno de desarrollo visual Studio .NET. y también SINJAI.

## Definición de Pilas.

Una pila es una estructura de datos en la que los elementos se añaden y se eliminan siempre por un único extremo llamado comúnmente el tope de la pila, también conocido cima de la pila. Trabaja bajo el principio de “El último que entra es el primero en salir”; es decir, siempre van a

entrar o se van a insertar elementos por el tope de la pila y además, siempre va a salir el elemento que se encuentra en el tope, el último que entró. Se utilizan para operaciones intermedias donde es necesario alma33cenar y retirar la información a corto plazo.

## Ejemplos de Pilas en la vida real.

Las pilas las vemos en la vida real, por ejemplo:

Una Pila de platos; los platos una vez que se van lavando y secando se van poniendo uno encima de otro, si necesita un plato, deberá tomar el primero de la pila, es decir, el último que se puedo en ella, no así el que está abajo ó el primero que se puso en la pila.

Una Pila de libros; lo mismo, podemos apilar libros e irlos colocando uno encima de otros, para poder acceder al último libro o el que está en el fondo de la pila es necesario sacar primero los anteriores, pues sino toda la pila de libros se puede caer.

## Representación gráfica y elementos de las pilas

Para simplificar la exposición y explicación de una pila, se utiliza una pila con datos de tipo carácter. Y se representa gráficamente la pila de la siguiente forma:

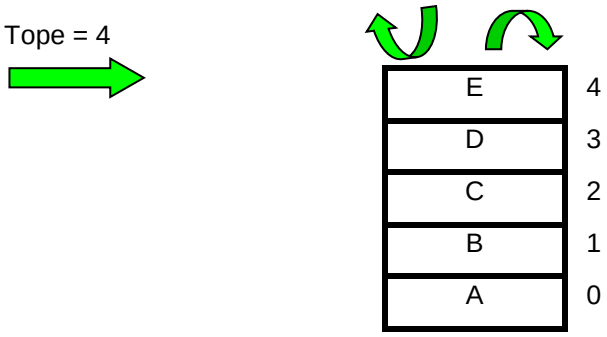



Fig. 1

En la figura anterior se tiene una Pila, los elementos que la componen son A, B, C, D, E. Las flechas indican el extremo de la Pila, llamado tope o cima cuyo valor para este caso es 4 y por donde van a entrar y salir los elementos.

Por lo que en definitiva, una pila tiene un conjunto de elementos y un tope para indicar cuál es el tope ó cima de la pila, quien es el último de ella o que entró en dicha pila.

## Estructura de Datos de una Pila.

Como se ha visto en el apartado anterior, las pilas son una colección de elementos y tienen un tope; para la representación de esta colección de elementos en la computadora se puede utilizar un arreglo, y utilizar otra variable que se le puede llamar tope para tener en ella la información de la última posición en el arreglo que está ocupada. Quedando la definición de la estructura de Datos de la siguiente forma:

Constante MáximaPila=5
pila es una estructura cuya información contiene un arreglo y una variable llamada tope.

Para el trabajo con ella, se puede utilizar una variable p, que será de tipo pila.

**Utilizando C++, quedaría de la siguiente forma:**

```
#include <iostream>  //libreria estándar de C
using namespace std;  // ya no hay que poner std:: delante de las instrucciones
const int maxpila=5;

struct stack{
       int tope;
       int items[maxpila];
       };

       struct stack p;
```

## Operaciones Básicas de las Pilas.

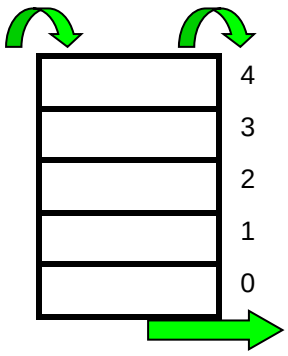
Pero trabajar con una pila en cualquier lenguaje de programación implica que se deberá operar con ella, por lo que primero se debe conocer las operaciones básicas que se realizan sobre ella, que a continuación se muestran:

Insertar: Se encarga de insertar ó introducir nuevos elementos a la Pila.
Llena: Verifica si la pila está llena, en cuyo caso, no se puede insertar en la pila
Vacía: Verifica si la pila está vacía, en cuyo caso, no se puede eliminar de la pila
Eliminar: Elimina de la pila último elementos que entró.

## Explicación de las Operaciones Básicas de las Pilas.

En este apartado se explican las operaciones básicas de las pilas, cómo es que funcionan y cómo se hacen, para después entender su programación.

Vacía(EMPTY):
La operación debe informar cuando una pila está vacía, es decir, cuando no tiene ningún elemento. Gráficamente se puede representar una Pila vacía como se muestra en la figura siguiente; figura Nro2:




Tope = -1

*Fig.2*

En este caso no existe ningún elemento en la Pila, y por lo tanto el tope de la Pila es -1.

***LLena():***

La Pila está llena cuando están ocupadas todas las casillas, en el ejemplo anterior cuando la pila tenga cinco elementos que es el valor que le dimos a la constante maxpila, ; por lo que si la pila tiene 5 elementos entonces estará llena.

En la siguiente figura; fig3; se muestra la representación gráfica de una Pila llena.

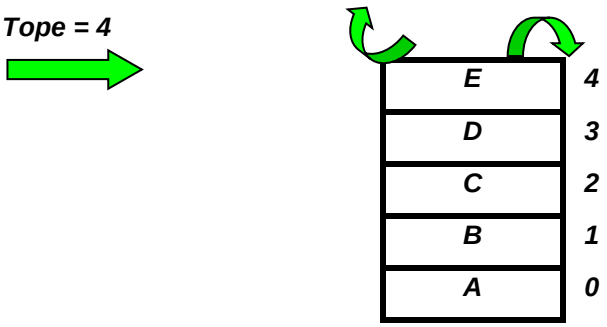



*Fig3*

La pila estará llena cuando el tope sea igual a MAXPILA-1

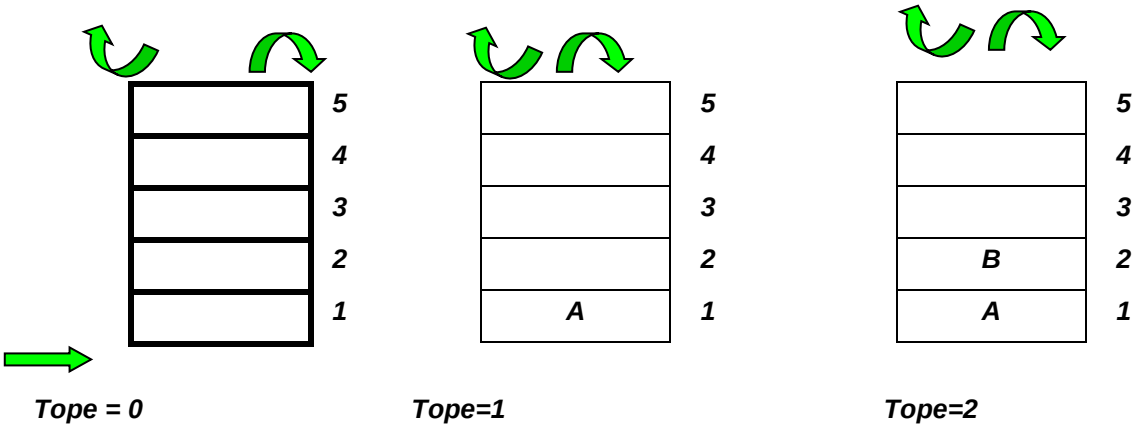
***Insertar(PUSH):***

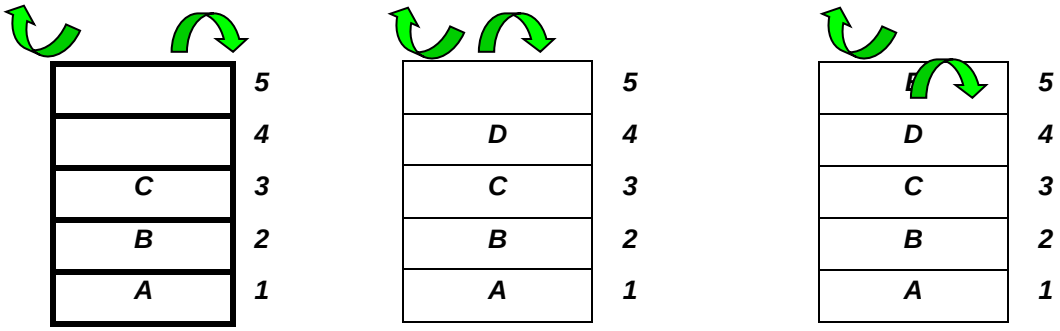
Para insertar elementos en una pila, se debe incrementar el tope y en la nueva posición ó casilla asignar el nuevo valor que se quiere insertar.

Supongamos la siguiente situación:

Se quieren insertar en una Pila los elementos A, B, C, D, E, F

Para comenzar asumiremos que la Pila está vacía y veremos y analizaremos la siguiente figura, Fig.4.






***Tope = 3*** ***Tope=4*** ***Tope=5***

***Fig4***

a) De inicio la pila está Vacía, el tope vale -1 y no hay elementos en la Pila
b) Se inserta A en la Pila, por lo que, el tope vale 0(se incrementa), en la posición 0 se encuentra el último elemento que entró.
c) Se inserta B en la Pila, por lo que, el tope vale 1(se incrementa), en la posición 1 se encuentra el último elemento que entró.
d) Se inserta C en la Pila, por lo que, el tope vale 2(se incrementa), en la posición 2 se encuentra el último elemento que entró.
e) Se inserta D en la Pila, por lo que, el tope vale 3(se incrementa), en la posición 3 se encuentra el último elemento que entró.
f) Se inserta E en la Pila, por lo que, el tope vale 4(se incrementa), en la posición 4 se encuentra el último elemento que entró.
g) Se inserta F en la Pila, pero el tope vale MaxPila-1, ya no caben más elementos, por lo que no se puede insertar

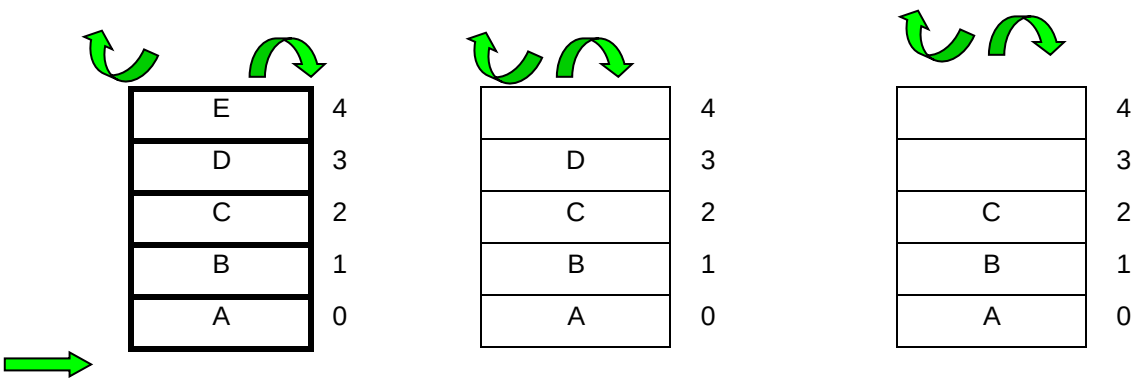
***Eliminar(POP):***

Para Eliminar en la pila, se debe decrementar el tope.

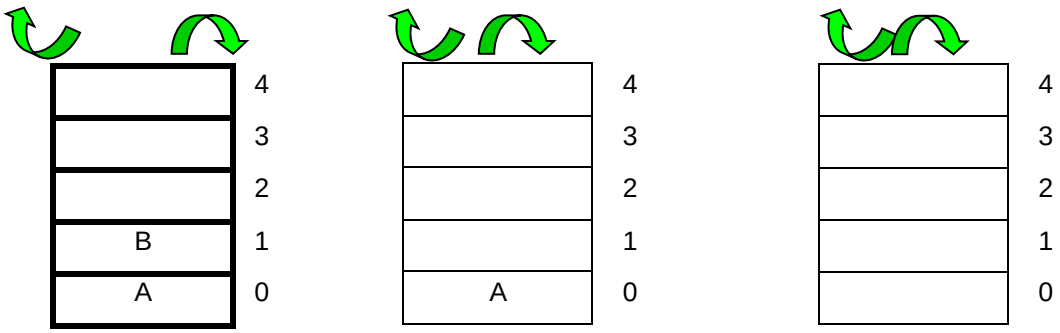
Supongamos la siguiente situación:

Se quieren eliminar en una Pila todos los elementos.

Para comenzar asumiremos que la Pila está llena y veremos y analizaremos la siguiente figura, Fig.5.




Tope = 4=MaxPila-1 Tope=3 Tope=2




Tope = 1 Tope=0 Tope=-1

**Fig5**

a) De inicio la pila está llena y el tope es 4, es igual a MaxPila-1
b) Se elimina, y sale de la Pila el último que entró, en este caso E y se decrementa el tope, que valdrá ahora 3
c) Se quiere volver a eliminar, entonces debe salir el último que se encuentra en la Pila que es D, y se decrementa el tope que vale ahora 2.
d) Se quiere volver a eliminar, entonces debe salir el último que se encuentra en la Pila que es C, y se decrementa el tope que vale ahora 1
e) Se quiere volver a eliminar, entonces debe salir el último que se encuentra en la Pila que es B, y se decrementa el tope que vale ahora 0
f) Se quiere volver a eliminar, entonces debe salir el último que se encuentra en la Pila que es A, y se decrementa el tope que vale ahora -1
g) Se quiere volver a eliminar, entonces debe salir el último que se encuentra, pero la Pila está vacía, no hay nada que eliminar porque el tope vale -1.

## Representación de las Pilas

Hasta ahora ya conoce que son las pilas y las operaciones básicas que se pueden realizar sobre ellas, pero como se pueden representar en la computadora?. Anteriormente se habló también de un arreglo, al ser un conjunto de elementos, pero también se pueden representar utilizando listas dinámicas, aspecto que se estudiará más adelante. En este capítulo se estudia utilizando arreglos. Por lo que a modo de conclusión las Pilas se pueden representar utilizando:

- Arreglos
- Listas enlazadas dinámicas

## Representación de Pilas utilizando arreglos

Para utilizar un arreglo será necesario definirlo, definir el tamaño del mismo y definir el tope, en este caso la estructura de datos será la misma que hemos declarado anteriormente.

Estructura de Datos

Constante MáximaPila=5
UN registro con la siguiente información:
Información : arreglo de un tipo determinado de dato
Tope máximo de la Pila

En C++

```
#include <iostream>  //libreria estándar de C
using namespace std;  // ya no hay que poner std:: delante de las
instrucciones
const int maxpila=5;
```

```
struct stack{
int tope;
int items[maxpila];
};

struct stack p;
```

Si analizamos la definición anterior nos daremos cuenta que existe un arreglo llamado Información donde vamos a ir insertando los elementos; estaremos limitados del espacio asignado en memoria es este caso específico a cinco elementos, que se expresa en la constante maxpila, aunque este valor puede ser cambiado en dependencia de la cantidad de elementos que se quiera trabajar con la pila. Se tiene una variable llamada Tope que nos indicará donde se encuentra el último elemento en la pila, es decir el último que entró.

*Operación Vacía*

Comenzaremos implementando la operación de Vacía. Podemos decir que la pila estará vacía cuando no exista ningún elemento en la misma, veamos la siguiente figura, fig6. :

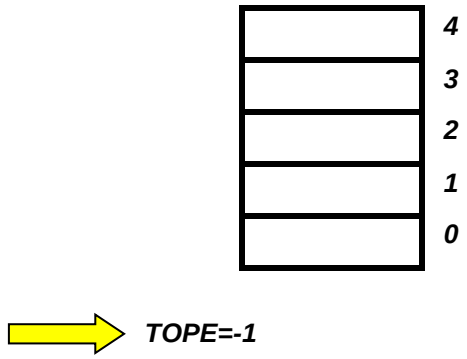



***Fig6.***

- La pila estará vacía cuando el Tope es igual a -1 o según como se trabaja el arreglo en el lenguaje de programación.
- La operación podemos implementarla como una función que devuelva verdadero si la pila está vacía, falso si no lo está.

| SEUDOCODIGO | C++ |
|---|---|
| Función Vacía ( Pila )<br>**Si Tope_de_Pila = -1 entonces**<br>**Vacía = Verdadero**<br>**Si no Vacía = falso**<br>Terminar | `int vacia()`<br>`{`<br>`if (p.tope==-1)`<br>`return 1;`<br>`else`<br>`return 0;`<br>`}` |

*Operación Llena*

En este apartado se muestra la operación llena, recuerde que estará llena cuando el tope sea igual a máxpila-1. En el ejemplo es 4, porque a maxpila le asignamos el valor de 5, vea la figura, fig7

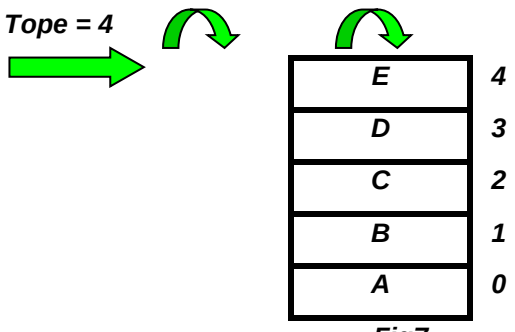



***Fig7.***

- La pila estará llena cuando el tope sea igual a MAXPILA
- La operación podemos implementarla como una función que devuelva verdadero cuando la pila está llena y falso cuando no lo está.

***A continuación se muestra este algoritmo***

| *SEUDOCODIGO* | *C++* |
|---|---|
| *Función Llena ( Pila )*<br>*Iniciar*<br>***Si Tope_ de _ Pila = Máxima _ Pila-1 entonces***<br>***Llena = Verdadero***<br>***Si no***<br>***Llena = Falso***<br>***Fin Si***<br>*Termina* | `int llena()`<br>`{`<br>`  if (p.tope==maxpila-1)`<br>`  return 1;`<br>`  else`<br>`  return 0;`<br>`}` |

*Operación Insertar*

Para insertar en la pila se debe verificar primero que no esté llena, ya que en ese caso no cabrían más elementos en la Pila. En el caso de que no esté llena, entonces se puede insertar, se incrementa el tope y luego en la nueva posición se asigna el nuevo valor.

A continuación se muestra este algoritmo

| *SEUDOCODIGO* | *C++* |
|---|---|
| *Procedimiento Insertar ( Pila, X )*<br>*Iniciar*<br>***Si LLena (Pila) entonces Error, la Pila está llena***<br>***Sino***<br>***Incrementar (Tope_ de_ Pila)***<br>***Pila(tope)= X***<br>***Fin Si***<br>*Terminar* | `void insertar(int x)`<br>`{`<br>`if (llena())`<br>`  {cout<<"La pila esta llena";`<br>`  cout<<endl;`<br>`}`<br>`  else`<br>`  {`<br>`  p.tope ++;`<br>`  p.items [p.tope]=x;`<br>`}`<br>`}` |

*Operación Eliminar*

Para eliminar en la pila debemos verificar primero que no esté vacía pues en ese caso no se podrá eliminar ningún elemento.
La operación podemos implementarla como una función donde pasaremos como parámetro la pila y devolverá el elemento que se ha eliminado.

La operación podemos implementarla como un procedimiento donde se le pasará como parámetro el valor del elemento y la pila donde vamos a insertar.

A continuación se expone el algoritmo:

| *SEUDOCODIGO* | *C++* |
|---|---|
| *Función Eliminar ( Pila )*<br>*Iniciar*<br>***Si vacía ( Pila ) entonces Error, la pila está vacía***<br>***Si no***<br>***Eliminar = Pila ( tope )***<br>***Decremento (Tope _ de _Pila)***<br>***Fin Si***<br>*Terminar* | `int eliminar()`<br>`{`<br>`if (vacia())`<br>`{cout<<"La pila esta vacia";`<br>`cout<<endl;`<br>`}`<br>`else`<br>`{`<br>`return p.items[p.tope];`<br>`p.tope--;`<br>`}`<br><br>`}` |

## Aplicaciones de las Pilas

Las pilas son utilizadas en diversas aplicaciones tales como son:

- Llamadas a subprogramas
- Recursividad
- Tratamiento de expresiones aritméticas
- Ordenamiento

A continuación explicaremos cada uno de los casos mencionados anteriormente.

- *Llamadas a subprogramas*

Cuando se tiene un programa que llama a otro subprograma internamente se usan pilas para guardar el estado de las variables del programa en el momento que se hace la llamada. Así en el momento que termina el subprograma se recuperan los valores almacenados en la pila. Además de las variables debe guardarse la dirección del programa en el punto en el cual fue interrumpido porque esa es a la posición que se regresa el control del proceso.

- *Operaciones Matemáticas.*

- *Ordenamiento*

## Ejercicios Prácticos Resueltos de Pilas

***1. Realice una procedimiento que permita Invertir una pila***

Para poder invertir una pila (es decir el orden de los elementos de la pila) tendríamos que recorrer la pila, eliminar uno a uno el orden de los elemento e ir insertándolos en otra pila, de esta forma obtengo los elementos de la pila invertidos, luego asignar la pila auxiliar a la anterior pila.

***Utilizaremos los procedimientos dados de operaciones de pilas.***

*INV _ Pila ( PILA)*
*Iniciar*

***Si Vacía(Pila) entonces error***
***Si no***
***Mientras no vacía(pila) hacer***
***X= eliminar pila(pila)***
***Insertar( aux , x)***
***Fin Hacer***
***Fin mientras***
***Hacer pila = aux***

*Terminar*

***2 - Escriba un algoritmo que permita eliminar los elementos repetidos en una pila(estos se encuentran de forma sucesiva)***

*Eliminar_Repetidos_ Pila ( PILA)*
*Iniciar*
***Si Vacía(pila) entonces error***
***Si no***
***Mientras no vacía(pila) hacer***
X= Eliminar Pila (pila)
Incrementar( i )
Arreglo(i)=x
verificar(x)
Si no verificar entonces
Insertar (aux,x)
Fin Hacer
Fin Mientras
***Hacer Pila = Aux***

*Terminar*

***3. - Escriba un algoritmo que permita eliminar un determinado elemento en una Pila***

*Eliminar_ Detelemento_ Pila ( PILA, Y)*
*Variables X: carácter*
*Aux : Pila*
*Iniciar*

Si Vacía(pila) entonces error
Si no
Mientras no vacía(pila) hacer
X= Eliminar Pila (pila)
Si X<> Y entonces
Insertar (Aux, X)
Fin Hacer
Fin Mientras
Mientras no vacía(Aux) hacer
X= Eliminar Pila (Aux)
Insertar (Pila, X)
Fin Hacer

Fin Mientras

*Terminar*

## Ejercicios A Resolver de Pilas

**1. Ponga ejemplos de la vida real donde se maneja el concepto de una pila**

1.

2.

**2. Realice el procedimiento que invierta un Pila en C++.**

**3. Realice el procedimiento que permita eliminar los elementos repetidos en una pila en C++**

4. **Escriba un algoritmo que permita eliminar un determinado elemento en una Pila en C++**

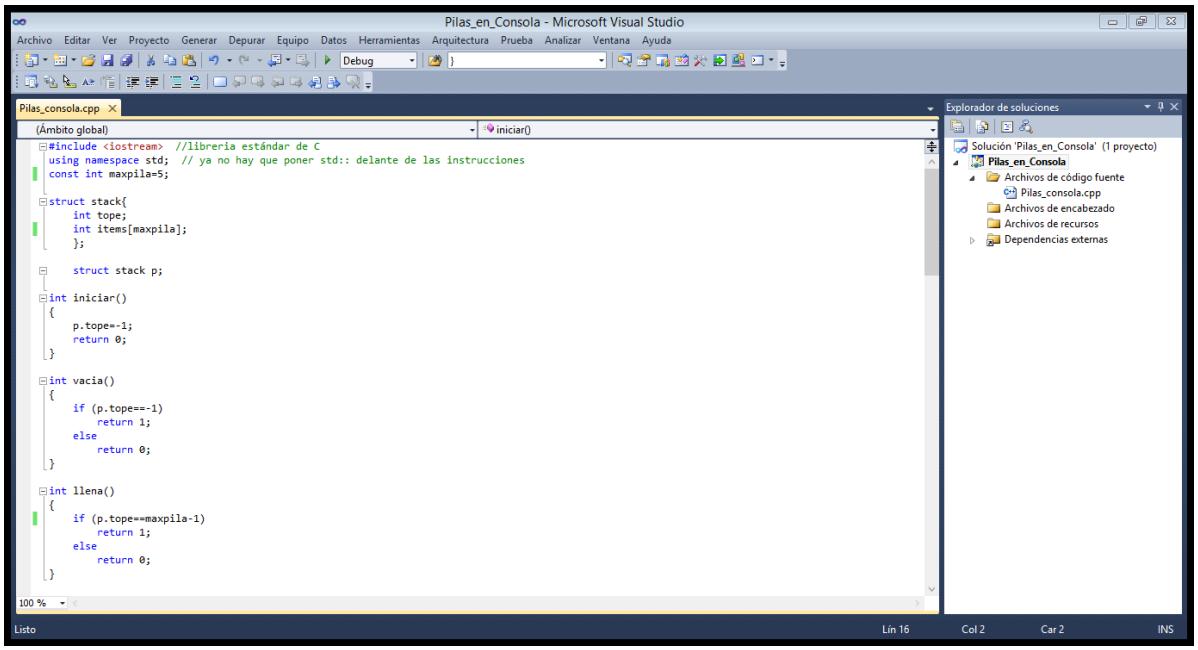
5. ***Escriba un programa principal en C++, que las opciones sean los ejercicios anteriores.***

**Orientación para el Laboratorio Nro1.**

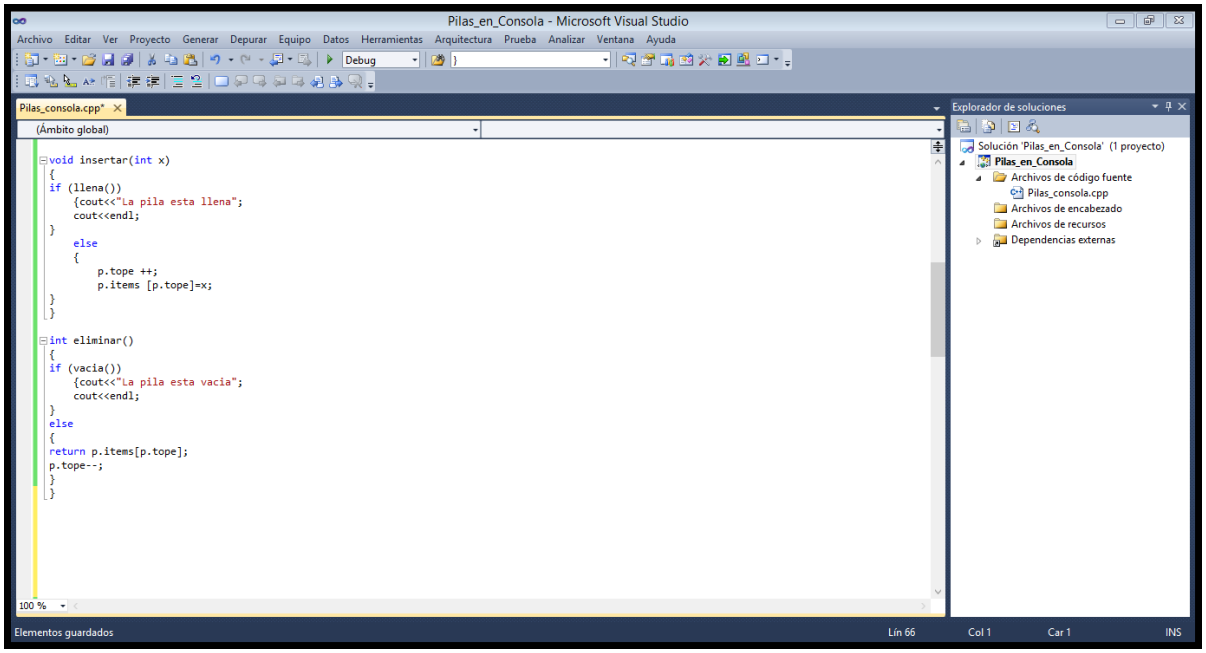
Utilizando el compilador Zinjal o visual Studio .NET C++, Implementar una pila como la estudiada en clases. Para tal efecto puede utilizar lo siguiente:

**Utilizando Visual Studio .NET y C++**

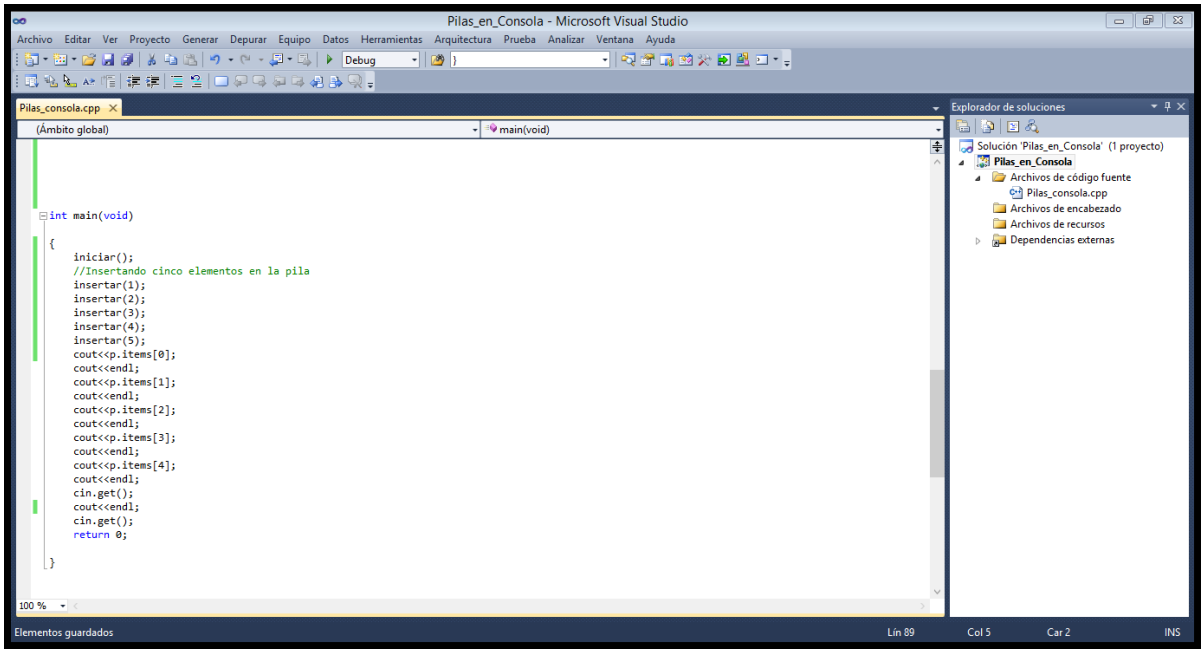
1.- En esta imagen se muestran las operaciones, vacía, llena e iniciar.



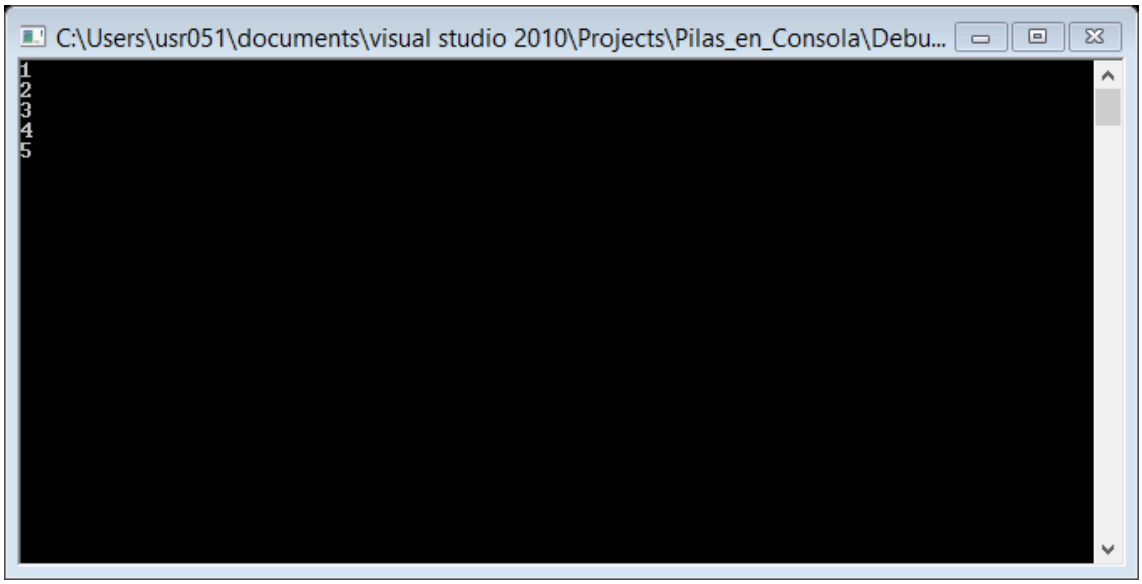

2.- En esta imagen se muestran las operaciones, vacía, llena e iniciar.



3.- Programa principal




4.- Resultado al correr la aplicación




5.- A este programa, deberá realizar los siguientes cambios:

a. Creará un menú principal, que permita obtener por teclado los datos que van ingresando
b. Permitirá también la opción de eliminar y mostrar en cualquier momento la pila y el valor de tope.

**Utilizando Zinjal**

Para este caso, realice los cambios necesarios que permitan la entrada por teclado de los valores de la pila. Deberá insertar elementos en la pila, mostrarlos y también eliminarlos.

```
#include <iostream>
#include<stdlib.h>
using namespace std;
const int maxpila=5;

struct stack{
        int tope;
        int items[maxpila];
};

struct stack p;

int iniciar()
{
        p.tope=-1;
        return 0;
}

int vacia()
{
        if (p.tope==-1)
                return 1;
        else
                return 0;
}

int llena()
{
        if (p.tope==maxpila-1)
                return 1;
        else
                return 0;
}
```

```
void insertar(int x)
{
        if (llena())
        {cout<<"La pila esta llena";
        cout<<endl;
        }
        else
        {
                p.tope ++;
                p.items [p.tope]=x;
        }
}

int eliminar()
{
        if (vacia())
        {cout<<"La pila esta vacia";
        cout<<endl;
        }
        else
        {
                return (p.items[p.tope]);
                p.tope--;
        }
}

int main(int argc, char *argv[]) {

        system("cls"); //Limpia la pantalla con ms dos.
        iniciar();
        //Insertando cinco elementos en la pila
        insertar(1);
        insertar(2);
        insertar(3);
        insertar(4);
        insertar(5);
        //Se muestra la pila
        cout<<p.items[0];
        cout<<endl;
```

```
        cout<<p.items[1];
        cout<<endl;
        cout<<p.items[2];
        cout<<endl;
        cout<<p.items[3];
        cout<<endl;
        cout<<p.items[4];
        cout<<endl;
        cin.get();
        cout<<endl;
        cin.get();
        return 0;
        system("pause");

        return 0;
}
```