Capítulo II Estructuras de Datos Secuenciales.

Contenido

1. Introducción.
2. Que son las Listas Lineales Secuenciales.
3. Que es la Colocación Secuencial.
4. Pilas.
5. Aplicaciones de las Pilas.
6. Colas.
7. Colas Circulares.
8. Decolas o Bicolas.
9. Aplicaciones de las Colas.
10. Representación de Pilas y Colas en Object Pascal.
11. Estudio Individual.
12. Laboratorios
13. Orientación Seminario.
14. Clase Práctica.

Introducción

En este capítulo se estudian las listas secuenciales y su colocación secuencial. Entre ellas se tienen las Pilas y las Colas, que son estudiadas profundamente en este capítulo. Viéndose además las diversas variantes que se pueden tener de cada una de ellas, tales como las colas circulares, las decolas o bicolas. Ambas estructuras de Datos, son estudiadas aplicando el enfoque orientado a objetos, utilizando como lenguaje de programación Object Pascal y Delphi como Entorno de Desarrollo.

Que son las Listas Secuenciales

Las listas secuenciales son Estructuras de Datos donde cada elemento de la lista se encuentra uno a continuación del otro. Por ejemplo una lista de estudiantes, una Lista de Empleados, una lista de deportistas y así sucesivamente.

Que es la Colocación Secuencial.

La colocación Secuencial se refiere al hecho de utilizar arreglos o vectores para el almacenamiento de la lista secuencial. Así por ejemplo se puede tener una lista secuencial de estudiantes, que se encuentra almacenada en un arreglo, lo mismo para una Lista de Empleados, etc.

Pilas.

Definición

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 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 almacenar y retirar la información a corto plazo.

Ejemplos de Pilas en la Vida Real.

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

Ejemplo1: 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, no el que está abajo.

Ejemplo2: Una Pila de libros; 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:

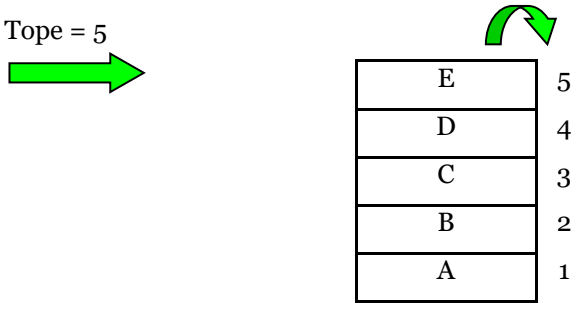


Fig. 1

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 5 y por donde van a entrar y salir los elementos

Operaciones Básicas de las Pilas.

Pero trabajar con una pila en cualquier lenguaje de programación se tiene que operar con ella, por lo que primero se deben conocer sus operaciones básicas que son:

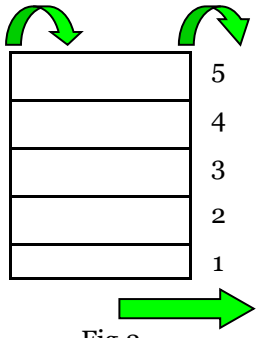
- Insertar (Push): Se encarga de insertar ó introducir nuevos elementos a la Pila
- Lllena (Full): Verifica si la pila está llena, en cuyo caso, no se puede insertar en la pila
- Vacía(Empty): Verifica si la pila está vacía, en cuyo caso, no se puede eliminar de la pila
- Eliminar(Pop): Elimina de la pila último elementos que entró.

Explicación de las Operaciones Básicas de las Pilas.

En este se explican las operaciones básicas de las pilas, anteriormente enunciadas.

Vacía (EMPTY):

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



Tope = 0 Fig.2

En este caso no existe ningún elemento en la Pila, y por lo tanto el tope de la Pila es 0. A continuación se muestra la función en object pascal que puede realizar esto:

```
Function Vacía: Bolean;
begin
if  Tope = 0 then Vacía: = True
else
Vacía = False;
End;
```

Llena (FULL):

La Pila está llena cuando están ocupadas todas las casillas hasta maxpila; en el caso nuestro se le asignó a maxpila valor 5 por lo que si la pila tiene 5 elementos entonces se encuentra llena. En la siguiente figura; fig3; se muestra la representación gráfica de una Pila llena.

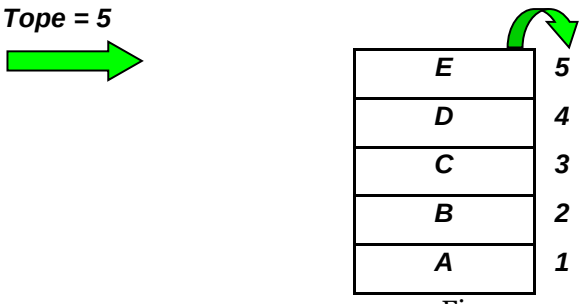


Fig3

En este caso estará llena cuando tope sea igual a maxpila, que en este caso es 5. A continuación se muestra la función en object pascal que puede realizar esta operación:

```
Function Llena: Boolean;
Begin
if  Tope = MaxPila then Llena:=True
else
Llena = False;
End;
```

Insertar (PUSH):

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

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.

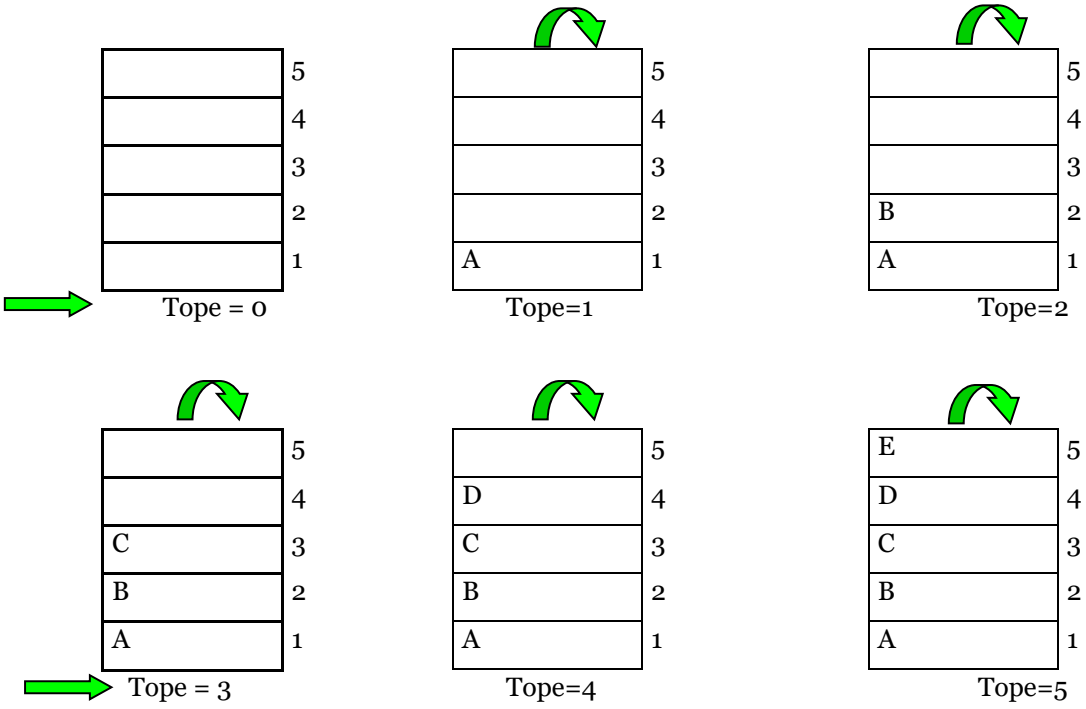


Fig4.

Explicación de la figura 4.

1. De inicio la pila está Vacía, el tope vale 0 y no hay elementos en la Pila
2. Se inserta A en la Pila, por lo que, el tope vale 1(se incrementa), en la posición 1 se encuentra el último elemento que entró.
3. Se inserta B en la Pila, por lo que, el tope vale 2(se incrementa), en la posición 2 se encuentra el último elemento que entró.
4. Se inserta C en la Pila, por lo que, el tope vale 3(se incrementa), en la posición 2 se encuentra el último elemento que entró.
5. Se inserta D en la Pila, por lo que, el tope vale 4(se incrementa), en la posición 2 se encuentra el último elemento que entró.
6. Se inserta E en la Pila, por lo que, el tope vale 5(se incrementa), en la posición 2 se encuentra el último elemento que entró.
7. Se inserta F en la Pila, por lo que, el tope vale 5, es igual a MaxPila, ya no caben más elementos, por lo que no se puede insertar

A continuación se muestra la función en Object Pascal que puede realizar esta operación:

```
Procedure Insert( elem: tipo-elem )
Begin
If  LLena then showmessage('Pila Llena')
Else
Begin
Inc(Tope);
Items[Tope]:= elem;
End;
End;
```

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.

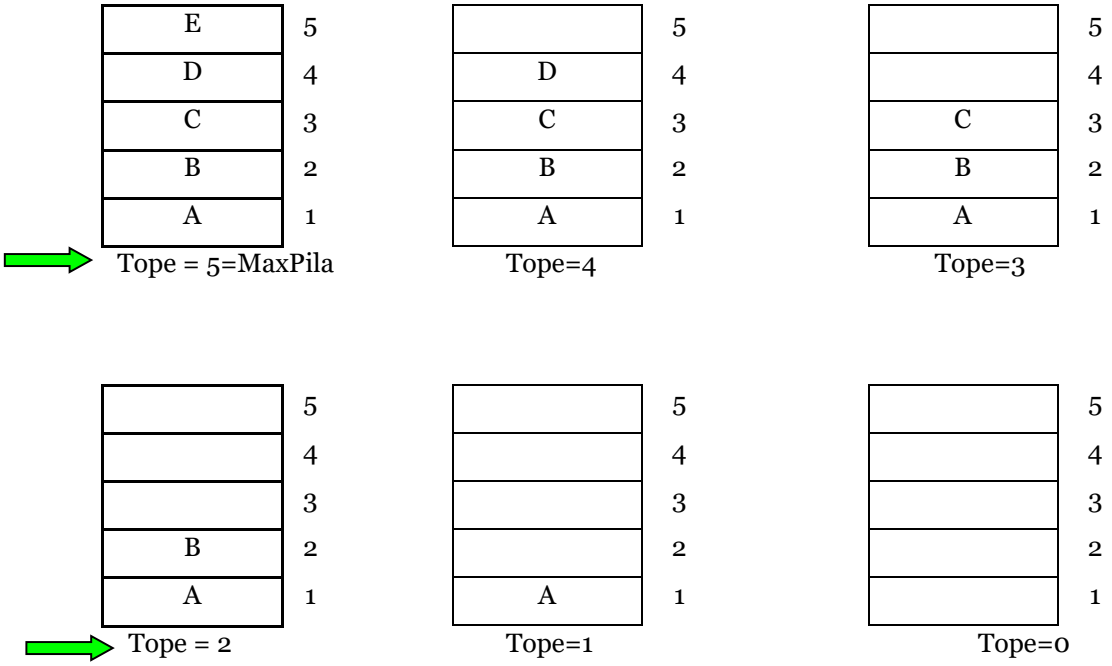


Fig5.

a) De inicio la pila está llena y el tope es 5, es igual a MaxPila.
b) Se elimina, y sale de la Pila el último que entró, en este caso E y se disminuye el valor del tope, que valdrá ahora 4.
c) Se quiere volver a eliminar, entonces debe salir el último que se encuentra en la Pila que es D, y se disminuye el valor del tope que vale ahora 3.
d) Se quiere volver a eliminar, entonces debe salir el último que se encuentra en la Pila que es C, y se disminuye el valor del tope que vale ahora 2.
e) Se quiere volver a eliminar, entonces debe salir el último que se encuentra en la Pila que es B, y se disminuye el valor del tope que vale ahora 1.

f) Se quiere volver a eliminar, entonces debe salir el último que se encuentra en la Pila que es A, y se disminuye el valor del tope que vale ahora 0.
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 0.

A continuación se muestra la función en Object Pascal que puede realizar esta operación:

```
Function Eliminar: tipoinfo;
begin
    if  vacia  then showmessage ('Error, la pila está vacía)
    else
    begin
    Eliminar:= items[Tope];
    dec(tope);
    end;
End;
```

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?.

Ya se mencionó también que son listas secuenciales, y que se pueden representar utilizando un arreglo, al ser un conjunto de elementos, pero también se pueden representar utilizando listas dinámicas, aspecto que usted hasta ahora no ha estudiado, pero que verá y analizará en próximos capítulo. En este capítulo usted estudiará la representación utilizando arreglos. Por lo que a modo de conclusión las Pilas se pueden representar utilizando:

- Arreglos
- Listas enlazadas dinámicas.

Representación Básica de una Pila, utilizando Object Pascal y Delphi como entorno de desarrollo.

Vamos a ver un ejemplo donde se utilice una pila, este caso es bastante simple y básico. Se crea la unit Pilas, y en la sección Interface se define la clase TPila, con sus atributos tope y el arreglo que le llamamos ítems, y sus atributos activos, que son las operaciones básicas.
A continuación se muestra la sección interface con la definición de la clase TPila.

```
Unit Pilas;        {Se pone de nombre a la unidad Pilas }

Interface

uses dialogs;

const maxpila= 10;       {Se define que va a tener como máximo 10 elementos la pila}

type

    tipoinfo = string;     {Este será el tipo de datos que va a tener la información de la Pila }

  TPila = object
   Private
   tope : 0..maxpila;
   items: array [1..maxpila] of tipoinfo;
```

```
Public
   constructor init;
   function vacia: boolean;
   function llena:boolean;
   procedure insertar(x: tipoinfo);
   function eliminar: tipoinfo;
   end;
```

Implementación de las Operaciones Básicas de una Pila utilizando object Pascal y Delphi como Entorno de Desarrollo.

Se implementan las operaciones de la pila, sus atributos activos, en este caso, init, vacia, llena,insertar y eliminar. A continuación se muestra la implementación:

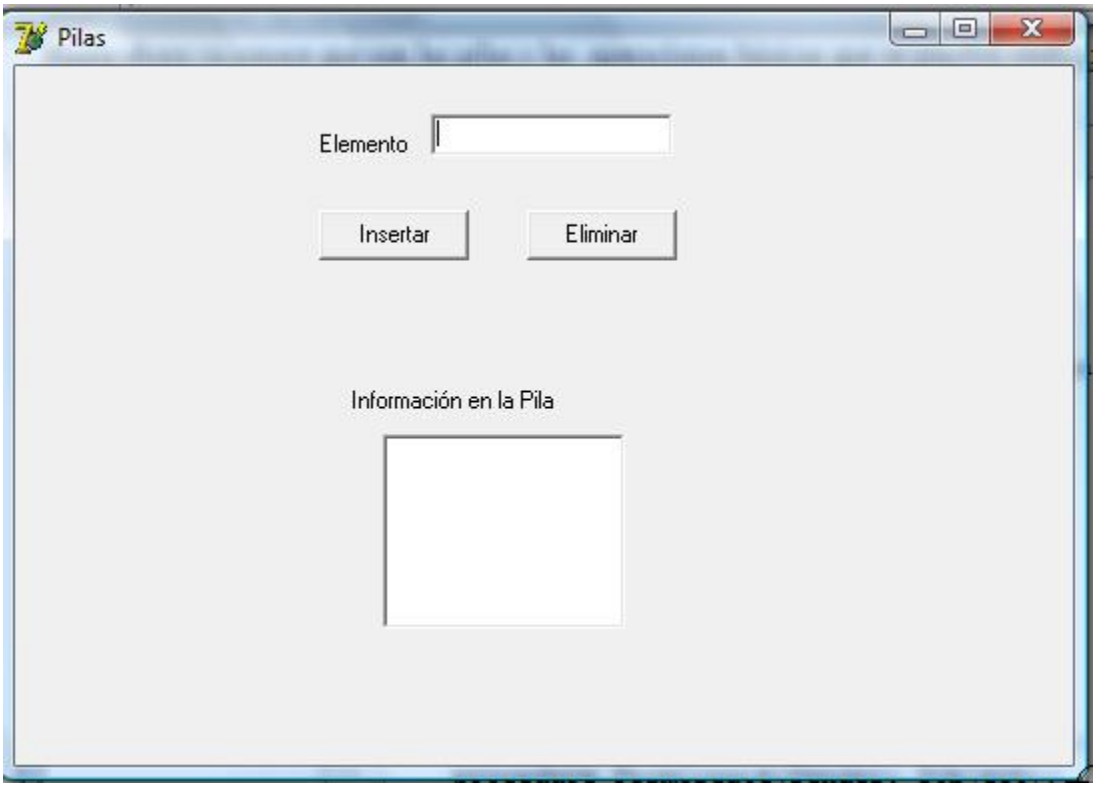
```
Implementation

   Constructor TPila.init ;
   begin
   tope:=0;
   end;


   Function TPila.vacia : boolean;
   begin
   vacia:= (tope=0);
   end;


   Function TPila.llena : boolean;
   begin
   llena:= (tope=maxpila);
   end;
```

```
Function TPila.eliminar : tipoinfo;
begin
if vacia then showmessage('Pila vacía')
else
begin
 eliminar:= items[tope];
 dec(tope);
end;
end;

Procedure TPila.insertar(x:tipoinfo);
begin
if llena then showmessage ('Pila llena')
else
begin
inc (tope);
items[tope]:=x;
end;
end;
```

Uso Básico de una Pila con Delphi y Object Pascal.

Para usar esta unidad, vamos a crear un ejemplo bastante simple.
Se crea la siguiente interfaz, insertando un control label (para el título), un control edit, para escribir y capturar el elemento que se quiere insertar en la Pila, dos botones(Insertar y eliminar), los cuales al hacer clic en ellos se invocará al método insertar y eliminar, respectivamente de la pila.

A continuación se muestra la Interfaz explicada:



En la unit del formulario, en la sección uses, se pone en uso la unit Pilas, como se muestra a continuación:

Interface

uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, **Pilas**, StdCtrls;

En la unit del formulario, en la sección **var**, se pone define una nueva variable, que en este caso le llamamos p que va a ser de tipo TPila, como se muestra a continuación:

var
Form1: TForm1;
P:TPila;

Se escribe el código siguiente asociado al evento oncreate del formulario:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
p.init;
end;
```

Se escribe el código siguiente asociado al botón insertar:

```
procedure TForm1.BtnInsertarClick(Sender: TObject);
begin
p.insertar(edit1.Text );
listboxPila.Items.Append(edit1.Text );
end;
```

Se escribe el código siguiente asociado al botón eliminar.

```
procedure TForm1.BtnEliminarClick(Sender: TObject);
var x:tipoinfo;
begin
x:=p.eliminar ;
listboxPila.Items.Delete(listbox1.Items.IndexOf(x));
end;
```

Se guarda el proyecto, se compila y corre el programa.

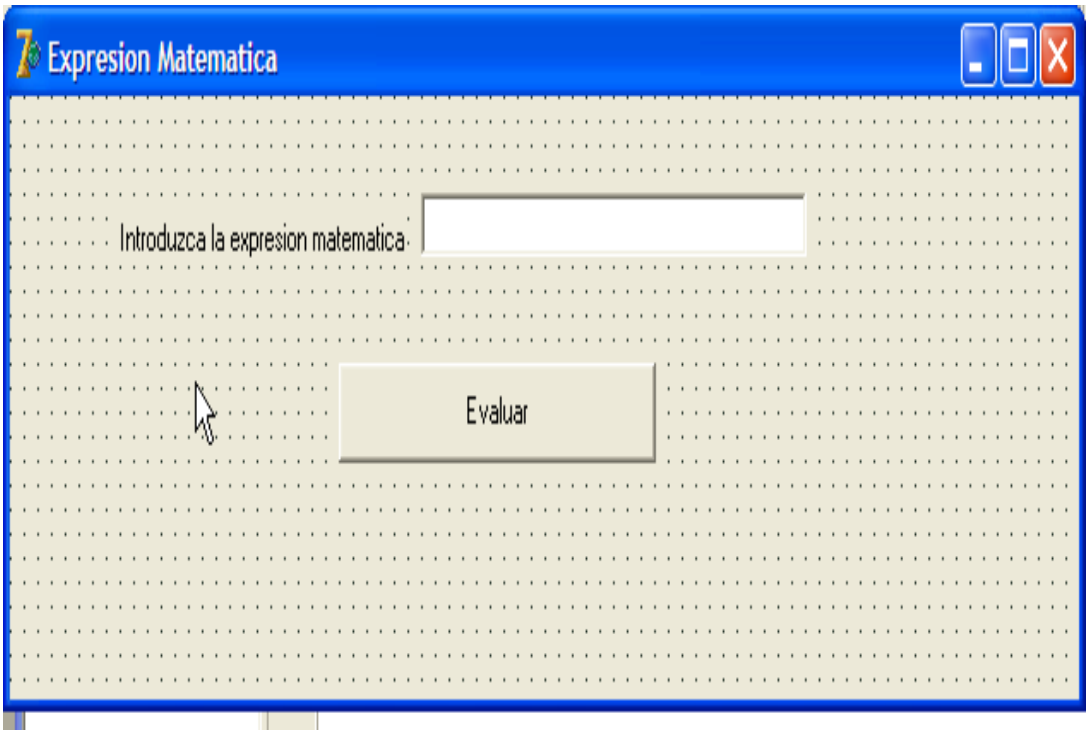
Aplicaciones de las Pilas

Las pilas son utilizadas en diversas aplicaciones tales como son:

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

Tratamiento de Expresiones Matemáticas.

Vamos a explicar esta aplicación. Para ello se crea la siguiente interfaz.



En la misma se han insertado los siguientes controles, un label, para indicar y especificar que debe escribir el usuario, en este caso es una expresión matemática.
Se insertó un control edit, para introducir la expresión matemática.
Se insertó un control button, para que una vez escrita la expresión matemática, al hacer clic en él, se evalúe la expresión, y se muestre un mensaje que diga si está correcta o no la expresión. Que será evaluada, en función del uso correcto o incorrecto de paréntesis.

En el evento onClic del botón Evaluar escribir el siguiente código:

```
procedure TForm1.BtnEvaluarClick(Sender: TObject);
begin
if EvaluarExp(edtexpresion.Text ) then showmessage('La expresión está escrita correctamente')
else showmessage('La expresión está escrita incorrectamente')
end;
```

En el código del formulario escribir el siguiente código, para evaluar la expresión, en este caso debe tomar en cuenta que cada paréntesis o llave o corchete que se abre debe ser cerrado, y que el último que se abrió es el primero que debe ser cerrado, es por ello que aquí se utiliza una Pila. Por los que en la sección uses de la unidad del formulario deber

```
Interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Pilas;

.....
.....

Implementation

....
....
Function EvaluarExp (expresion:string): Boolean;
var i:integer;   p:TPila;  c:string;   eval:boolean;
begin
p.init;
eval:=true;
For i:=1 to length(Expresion) do
begin
if Expresion[i] in  ['(','{','['] then p.insertar(Expresion[i])else
if Expresion[i] in  [')','}',']'] then
if p.vacia then  Eval:=False     else
  begin
```

```
 c:=p.eliminar;
 if (expresion[i]=')') and (c<>'(') then Eval:=False
 else if ((expresion[i]='}') and (c<>'{')) then Eval:=False
 else if ((expresion[i]=']') and (c<>'[')) then Eval:=False
 end;
end; {del for}
if p.vacia and Eval then Evaluarexp:=True
else EvaluarExp:=False;
end;
```

Colas.

Definición de Colas.

Una Cola es una estructura de datos en la que los elementos se añaden siempre al final de la lista y los elementos salen siempre por el principio de la lista. Una cola es una estructura de datos que consta de una serie de elementos, tiene la característica de que el primero que entra es el primero en salir. Una cola crece al insertar datos nuevos en un extremo (el final de la cola) y se hace pequeña al eliminar los datos del otro extremo (el principio de la cola).

Ejemplos de Colas de la Vida Real.

Las colas abundan en la vida como por ejemplo:

- Una cola esperando en un consultorio médico para ser atendidos. Los pacientes en la medida que llegan así serán atendidos.
- Los autos esperando a pasar por un cruce de calles. El primer auto de la fila o cola es el primero que pasa cuando se enciende la luz verde en el semáforo.
- Las personas que esperan para usar un teléfono público en un punto entel. Según el orden en que llegan las personas y se incorporan a la cola, así será la asignación del teléfono una vez que ha sido dejado de usar o ha quedado libre.
- La cola en un cajero automático de un banco, esperando el turno para realizar sus transacciones.

Representación Gráfica y Elementos de una Cola.

Como bien se dijo anteriormente una cola es un conjunto de elementos, en el gráfico se muestran los elementos A, B, C, D y E, pero además se conoce quien debe ser atendido primero(variable primero) y quien debe ser atendido de último (variable ultimo). Ver figura 6 a continuación.

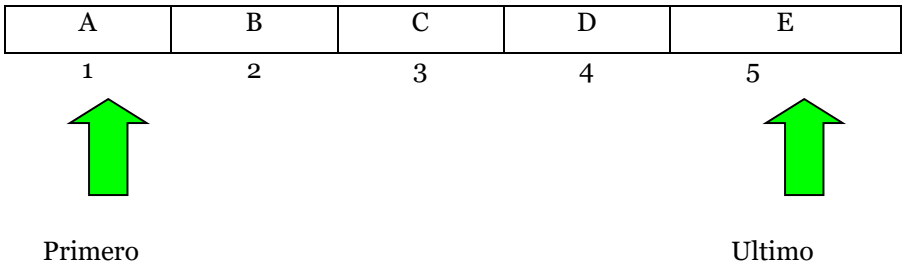


Fig6.

Operaciones Básicas de una Cola.

Trabajar con una cola en cualquier lenguaje de programación implica que se debe operar con ella, por lo que primero se tiene que conocer sus operaciones básicas que a continuación se exponen.

- Insertar: Se encarga de insertar ó introducir nuevos elementos a la Cola por detrás
- Llena: Verifica si la Cola está llena, en cuyo caso, no se puede insertar.
- Vacía: Verifica si la cola está vacía, en cuyo caso, no se puede eliminar.
- Elimina: Elimina de la cola el primer elementos que entró.

Explicación de las Operaciones Básicas de una cola.

Ya conoce como se representa una cola, y sus operaciones básicas en cualquier lenguaje de programación; ahora conocerá cómo es que funciona y su utilidad. Para así de esta forma pueda programarla en cualquier lenguaje de programación.

Operación Vacía.

Esta operación es la encargada de informar cuando la cola está vacía; cuando, que es cuando no tiene ningún elemento. Gráficamente se puede representar una cola vacía como se muestra en la figura Nro7, donde se tiene un espacio designado para contener sus elementos pero todavía no han ingresados elementos a ella.

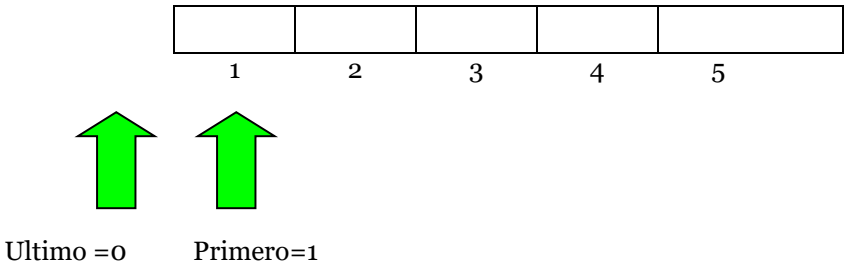


Fig.7

Se ha asignado a las variables primero valor 1 al conocer de antemano que primero siempre se encuentra en la posición 1, a último se le asignó valor 0, ya que es un valor que varía constantemente.

A continuación se muestra la función que puede realizar esto:

```
Function Vacía: Bolean;
begin
if  ultimo < primero  then Vacía: = True
else
Vacía = False;
End;
```

Operación Llena.

La Cola está llena cuando están ocupadas todas las casillas desde 1 hasta maxcola; en este caso se asume que maxcola vale 5, por lo que como máximo esta cola tendrá 5 elementos, y en cuyo caso estará llena. En la siguiente figura; Fig8 se muestra la cola llena gráficamente.

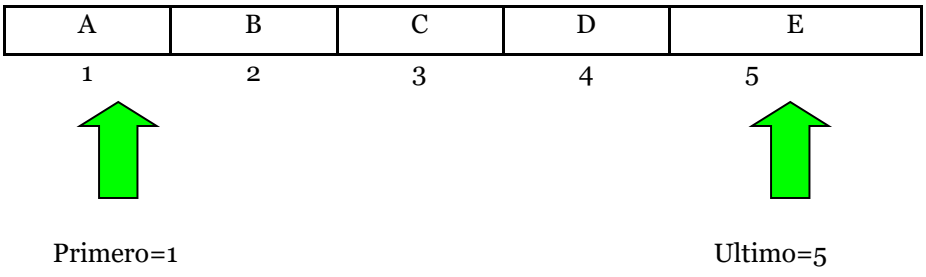


Fig8

En este caso estará llena cuando ultimo sea igual a maxcola que en este caso es 5. A continuación se muestra la función en object pascal que puede realizar esta operación:

```
Function Llena: Boolean;
Begin
if Ultimo = MaxCola then
Llena:=True
else
Llena = False;
End;
```

Operación Insertar

Es operación es la encargada de introducir nuevos elementos a la cola. Esto se hace incrementando el valor de la variable último que se asocia a la posición que ocupa en el arreglo, y luego en esta nueva posición de último se asigna el valor que quiere insertar. Para entender esto de mejor forma vea la fig9 y la explicación asociada.

a)Cola vacía					
	1	*2*	*3*	*4*	*5*
Último=0	*Primero=1*				
b)Se inserta A	*A*				
	1	*2*	*3*	*4*	*5*
	Primero=1	*=Ultimo*			
c)Se inserta B	*A*	*B*			
	1	*2*	*3*	*4*	*5*
	Primero=1	*Ultimo=2*			
d)Se inserta C	*A*	*B*	*C*		
	1	*2*	*3*	*4*	*5*
	Primero=1		*Último=3*		
e)Se inserta D	*A*	*B*	*C*	*D*	
	1	*2*	*3*	*4*	*5*
	Primero=1			*Último=4*	
f)Se inserta E	*A*	*B*	*C*	*D*	*E*
	Primero=1				*Último=5*
g)Se quiere Insertar F	*No se puede está llena la cola*				
	1	*2*	*3*	*4*	*5*

Fig.9

En la figura anterior, se muestra la inserción de 5 elementos en una cola. Estos son: A, B, C, D, E, F. De inicio (a) la cola está vacía y Primero=1, Ultimo=0. (b) Se inserta el elemento A y pasa a ser el primero y último de la cola. (c) en este momento se inserta C, que pasa a ser el último y está en la posición 3 valor de último también y A sigue siendo el primero en la cola. (d) se inserta D pasando a ser el último que toma a la vez valor 4. (e) se inserta a E, que pasa a ser el último de la cola y se llena la cola, después de esto no se pueden insertar más elementos pues está llena y ya ho hay espacio para más.

A continuación se muestra la función en Object Pascal que puede realizar esta operación:

```
Procedure Insert( elem: tipo-elem )
Begin
If  LLena then showmessage('Cola  Llena')
Else   Begin
Inc(ultimo);
Items[ultimo]:= elem;
End;
End;
```

Operación Eliminar:

Es la operación que se encarga de sacar elementos de la cola, si se aplica el principio de que el último que llega es último en salir, siempre debe salir el primero de la cola. Cuando se elimina todos los demás elementos se recorren una posición adelante. Vea la siguiente figura y su explicación para comprender con mayor detalle lo anterior.

La cola está llena	A	B	C	D	E
	1	2	3	4	5
	Primero=1				Último=5
Se Elimina y sale A	B	C	D	E	
	1	2	3	4	5
	Primero=1			Último= 4	
SeElimina y sale B	C	D	E		
	1	2	3	4	5
	Primero=1		Ultimo=3		
Se elimina y sale C	D	E			
	1	2	3	4	5
	Primero=1	Último=2			
Se Elimina y sale D	E				
	1	2	3	4	5
	Primero=1	Último=1			
Se Elimina y sale E					
	1	2	3	4	5
Último=0	Primero=1				
Se Elimina	No se puede eliminar la cola esta vacia				
	1	2	3	4	5

Fig.10

Para la explicación de esta operación, se tomó de inicio una cola llena de cinco elementos (a). Cuando se elimina (b), sale el primero, que es A, recorriéndose todos una posición hacia delante, estableciéndose B como primero. (c) al volver a eliminar sale C, tomando D su posición al recorrerse todos los demás elementos una posición anterior. (d) se vuelve a eliminar y sale D, tomando E su lugar, (e) se vuelve a Eliminar y sale E quedando la cola vacía, y último en cero, y primero en 1, si se quiere volver a eliminar (f) da un error, pues no se pueden eliminar elementos en la cola ya que está vacía.

A continuación se muestra la función en Object Pascal que puede realizar esta operación:

```
Function Eliminar: tipoinfo;
begin
    if  vacia  then showmessage ('Error, la cola  está vacía)
    else
    begin
    Eliminar:= items[pimero];
    For I:=1 to q.último-1 do
    q.info[i]:= q.info[i+1;
    dec(ultimo);
    end;
End;
```

Representación Básica de una cola utilizando object pascal y delphi como entorno de desarrollo.

Vamos a ver un ejemplo donde se utilice una cola, este caso es bastante simple y básico. Se crea la unit Colas, y en la sección Interface se define la clase TCola, con sus atributos primero, último y el arreglo que le llamamos ítems, y sus atributos activos, que son las operaciones básicas.

A continuación se muestra la sección interface con la definición de la clase TCola.

```
unit Colas;

interface

uses dialogs;

const maxcola = 10;

type

    tipoinfo=string;
```

```
TCola = object
Private
primero, ultimo: 0..maxcola;
items: array[1..maxcola] of tipoinfo;
Public
Constructor Init;
Function Vacia:boolean;
Function Llena:boolean;
Function Eliminar: Tipoinfo;
Procedure Insertar(x:tipoinfo);
End;
```

Implementación de las Operaciones Básicas de la Cola.

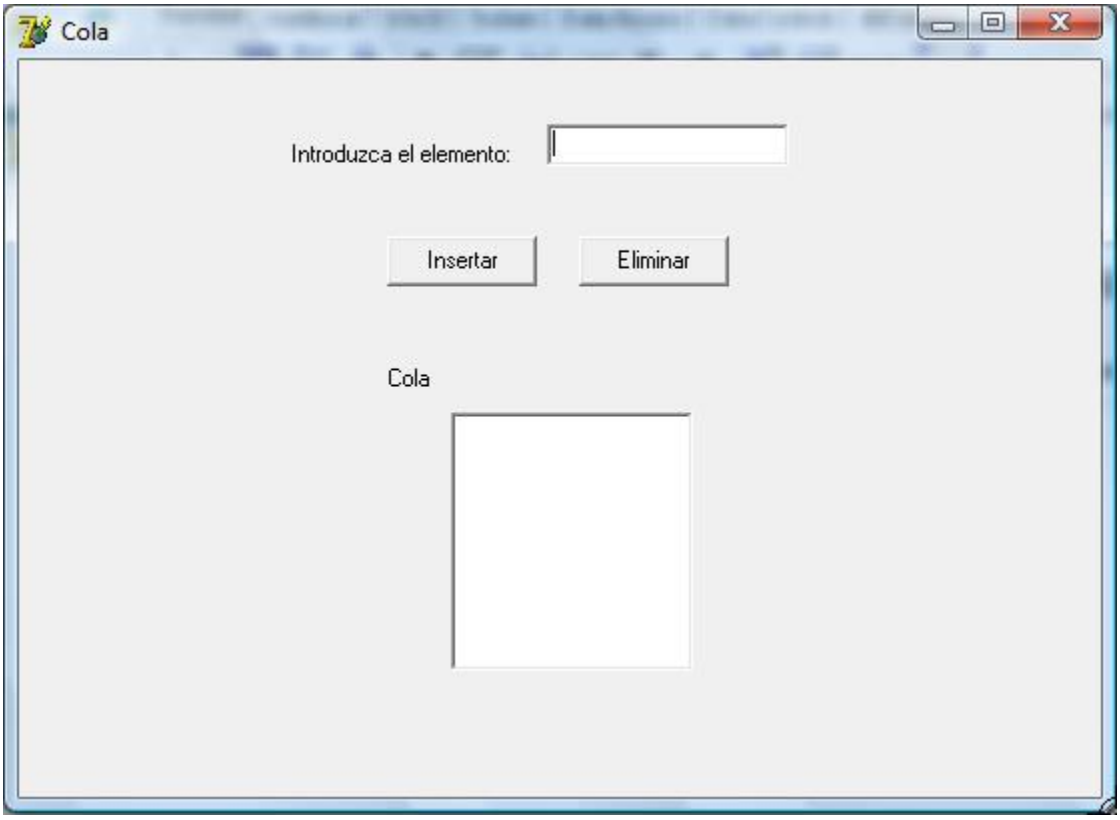
Se implementan las operaciones de la cola, sus atributos activos, en este caso, init, vacía, llena, insertar y eliminar. A continuación se muestra esta implementación:

```
Implementation

    Constructor TCola.Init;
    begin
            Primero:=1;
            Ultimo:=0;
    end;

    Function TCola.Vacia:boolean;
    begin
    Vacia:=(ultimo<primero)
    end;

    Function TCola.Llena:boolean;
    begin
    Llena:=(ultimo=maxcola);
    end;
```

```
Function TCola.Eliminar: Tipoinfo;
var i:integer;
begin
if vacia then showmessage('Cola Vacía')
else
      begin
         eliminar:=items[primero];
         for i:=1 to ultimo-1 do
         items[i]:=items[i+1];
         dec(ultimo);
      end;
end;


Procedure TCola.Insertar(x:tipoinfo);
begin
if llena then showmessage('Cola llena')
 else
 begin
 inc(ultimo);
 items[ultimo]:=x;
 end;
end;
```

Uso Básico de una Cola utilizando Object Pascal y Delphi como entorno de desarrollo.

Para usar esta unidad, vamos a crear un ejemplo bastante simple.
Se crea la siguiente interfaz, insertando un control label (para el título), un control edit, para escribir y capturar el elemento que se quiere insertar en la Cola, dos botones (Insertar y eliminar), los cuales al hacer clic en ellos se invocará al método insertar y eliminar, respectivamente de la Cola.



En la unit del formulario, en la sección uses, se pone en uso la unit Colas, como se muestra a continuación:

Interface

Uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, **Colas**;

En la unit del formulario, en la sección **var**, se pone define una nueva variable, que en este caso le llamamos q que va a ser de tipo TCola, como se muestra a continuación:

var

Form1: TForm1;
q: TCola;

Se escribe el código siguiente asociado al evento oncreate del formulario:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
q.Init;
end;
```

Se escribe el código siguiente asociado al botón insertar:

```
procedure TForm1.BtnInsertarClick(Sender: TObject);
begin
q.Insertar(edtcola.Text);
ListBox1.Items.Append(edtcola.Text);
end;
```

Se escribe el código siguiente asociado al botón Eliminar:

```
procedure TForm1.Button1Click(Sender: TObject);
var x:tipoinfo;
begin
x:=q.Eliminar;
listbox1.Items.Delete(listbox1.Items.IndexOf(x) );
end;
```

Colas Circulares.

Definición

Para hacer un uso más eficiente de la memoria disponible, se trata a la cola como a una estructura circular, determinando el algoritmo de colas circulares. La figura 11 muestra gráficamente una cola circular.

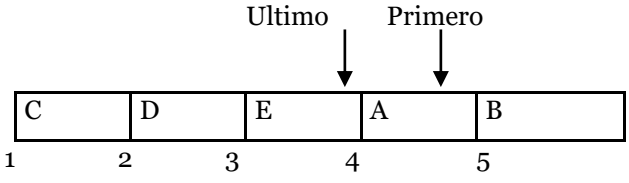


Fig.11

En la figura anterior se representa un cola, donde el puntero Primero nos dice donde se encuentra el primer elemento, en este caso en la posición 4; y el puntero Ultimo nos dice donde se encuentra el último elemento, en este caso en la posición 3.

Operaciones Básicas.

Las operaciones básicas serian las mismas vistas con anterioridad, Vacía, Llena, Insertar y Eliminar, pero su implementación difiere de las estudiadas.

Operación Vacía

Gráficamente se puede representar a la cola circular vacía como se muestra en la figura 12. Se Asume de que ultimo y primero valen cero

Ultimo = 0 Primero =0

Fig12

A continuación se muestra la función en object pascal que puede realizar esto:

```
Function Vacía: Boolean;
begin
if  Ultimo=0 and primero=0   then Vacía: = True
else
Vacía = False;
End;
```

Operación Llena

Gráficamente se puede representar a la cola circular llena como se muestra en la figura 13.

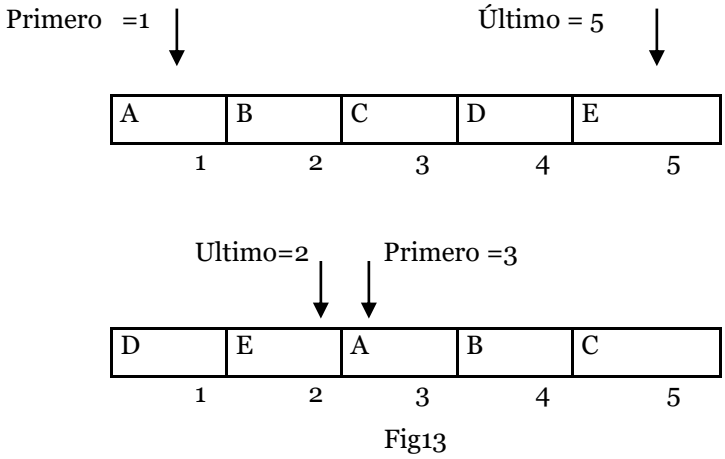


Fig13

A continuación se muestra la función en object pascal que puede realizar esto:

```
Function Llena: Boolean;
begin
if (Ultimo=MaxCola) and (Primero=1)) or
(Ultimo+1=Primero) then Llena: = True
else
Llena: = False;
End;
```

Operación de Insertar.

La inserción se explica en la siguiente figura:

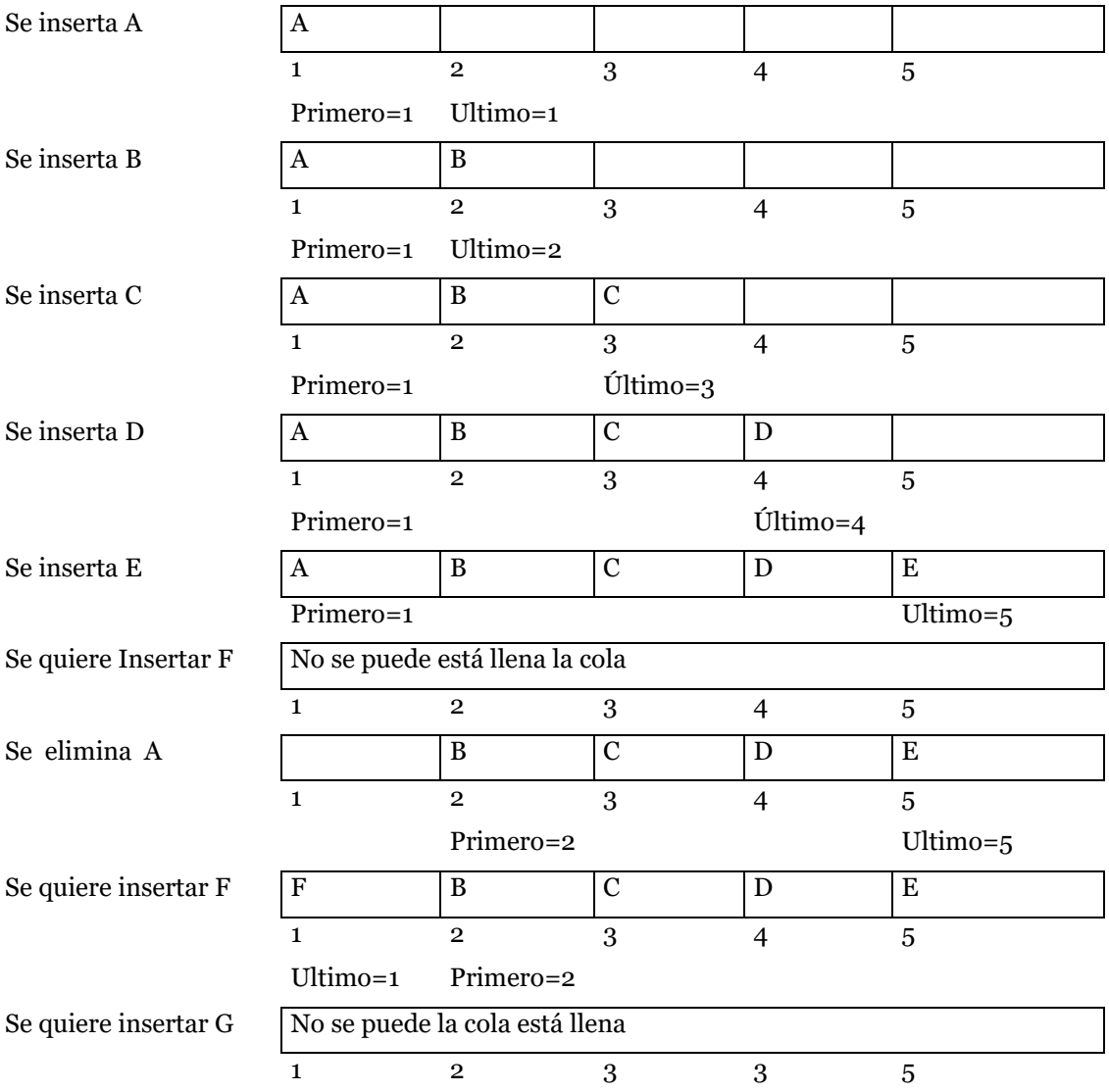


Fig14

A continuación se muestra la función en Object Pascal que puede realizar esta operación:

```
Procedure Insert( elem: tipo-elem )
Begin
If  LLena then showmessage('Cola  Llena')
Else
Begin
If ultimo=Maxcola then
      ultimo:=1
      Else    inc(último );
info[ultimo]:=x;
if primero=0 then Primero:=1
End;
```

Operación Eliminar

Gráficamente se puede representar la eliminación como se muestra en la siguiente figura.

La cola está llena	F	G	H	D	E
	1	2	3 Ultimo=3	4 Primero=4	5
Se elimina y sale D	F	G	H		E
	1	2	3 Ultimo=3	4	5 Primero=5
Se elimina y sale E	F	G	H		
	1 Primero=1		3 Ultimo=3	4	5
Se elimina y sale F		G	H		
	1	2 Primero=2	3 Último=3	4	5
Se elimina y sale G			H		
	1	2	3 Primero=3	4 Ultimo=3	5
Se elimina y sale H					
Primero = Ultimo=0	1	2	3	4	5
Se elimina	La cola está vacía no se puede Eliminar				

Fig15

A continuación se muestra la implementación en Object Pascal de la Función Eliminar.

```
Function Eliminar( elem: tipo-elem )
Begin
If  Vacia  then showmessage('Cola  Vacía)
Else
Begin
Eliminar:= info[primero];
If primero=ultimo then begin
        primero:=0
         ultimo=0
else
if primero=maxcola then primero:=1
else primero:=primero+1
end;
End;
```

Representación Básica de una cola circular utilizando object pascal y delphi como entorno de desarrollo.

Vamos a ver un ejemplo donde se utilice una cola circular, este caso es bastante simple y básico. Se crea la unit ColasC, y en la sección Interface se define la clase TColaC, con sus atributos primero, último y el arreglo que lo heredará de la clase Tcola, ya definida anteriormente.

A continuación se muestra la sección interface con la definición de la clase TColaC.

```
unit ColasC;

interface

uses Colas, Dialogs;

type
```

```
TColaC=object(TCola)
Constructor Init;
Function Vacia:boolean;
Function Llena:Boolean;
Function Eliminar: tipoinfo;
Procedure Insertar(x:tipoinfo);
end;
```

Implementación de las Operaciones Básicas de la Cola Circular.

Se implementan las operaciones de la cola circular, sus atributos activos, en este caso, init, vacía, llena, insertar y eliminar. A continuación se muestra esta implementación:

Implementation

```
Constructor TColaC.Init;
begin
primero:=0;
ultimo:=0;
end;

Function TColaC.Vacia:boolean;
begin
if (primero=0) and (ultimo=0) then
Vacia:=true
else
vacia:=False;
end;

Function TColaC.Llena:Boolean;
begin
if ((primero=1) and (ultimo=maxcola)) or (ultimo+1=primero) then
llena:=true
else
llena:=False;
end;
```

```
Function TColaC.Eliminar: tipoinfo;
begin
if vacia then Showmessage('Cola Vacía')
else
begin
Eliminar:=items[primero];
if primero=ultimo then init
else
begin
if primero=maxcola then primero:=1
else
inc(primero);
end;
end;
end;

Procedure TColaC.Insertar(x:tipoinfo);
begin
if llena then showmessage('Cola llena')
else
begin
if ultimo = maxcola then ultimo:=1
else
inc(ultimo);
items[ultimo]:=x;
if primero = 0 then primero:=1;
end;
end;
```

Decola ó Bicola.

Definición

Una Decola o Bicola es una cola, con la característica de que se pueden hacer inserciones y eliminaciones por cualquiera de los dos extremos de la cola, a continuación se representa gráficamente:

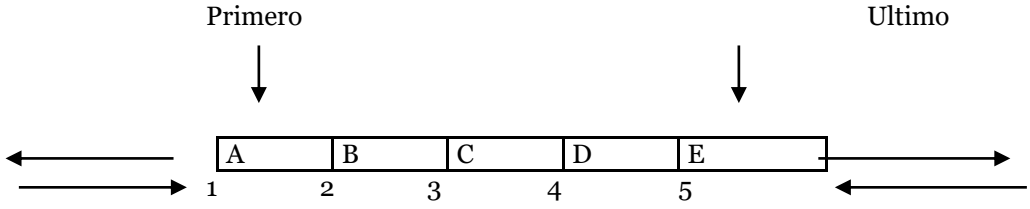


Fig.16

Las flechas en el extremo izquierdo significan que se puede eliminar e insertar por este extremo. Las flechas en el extremo derecho significan que se puede insertar y eliminar por este extremo.

Operaciones Básicas.

Al poderse insertar y eliminar por ambos extremos, las operaciones básicas sobre ellas quedarían de la siguiente forma:

- Insertar delante.
- Insertar por detrás.
- Eliminar por delante.
- Eliminar por atrás.

Insertar por Delante.

A continuación se muestra la implementación en object pascal del procedimiento Insertar por delante.

```
Procedure InsertarDelante(x:tipoinfo);
begin
If llena then write(ERROR,cola llena)
Else
Begin
Inc(ultimo);
For i:=último-1  dowto 1 do
info[I+1]:=info[I]
info[ultimo]:=x;
end;
end;
```

Eliminar por Atrás.

A continuación se muestra la implementación en object pascal del procedimiento Eliminar por detrás.

```
Functión EliminarAtrás: Tipoinfo;
Begin
If  vacía(Q) then write(Error, cola vacía)
Else
begin
Eliminar = info[ultimo];
Dec(Ultimo);
End;
End;
```

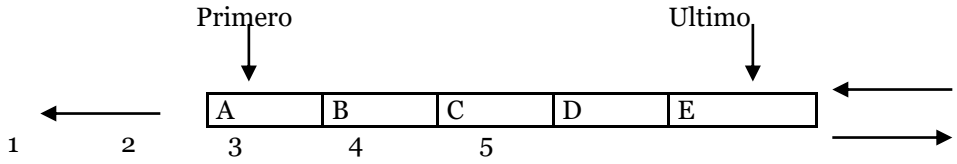
Variantes de las DECOLAS:

- Decola con Entrada Restringida.
- Decola con Salida Restringida.

Decola con Entrada Restringida.

Las entradas o inserciones solo pueden hacerse por el final de la cola, ya no por delante. Las eliminaciones se permiten por cualquiera de los dos extremos.

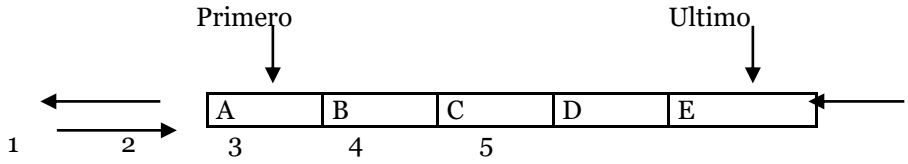
En la siguiente figura se representa gráficamente este caso:



Decola con Salida Restringida.

Las salidas se realizan solamente por un extremo; por el primero. Se permite las entradas por cualquier extremo de la cola, tanto por delante como por atrás.

En la siguiente figura se representa gráficamente este caso:



Aplicación de Colas en la Computadora.

El concepto de la cola ligado a computación puede verse en las colas de impresión. Cuando hay una sola impresora para atender a varios usuarios, puede suceder que alguno de ellos soliciten los servicios de impresión al mismo tiempo o mientras el dispositivo este ocupado. En estos casos se forma una cola con los trabajos que esperan para ser impresos. Los mismos se irán imprimiendo en el orden en el cual fueron introducidos en la cola.

Otro caso de aplicaciones de colas en computación, es el que se presenta en los sistemas de tiempo compartido. Varios usuarios comparten ciertos recursos, como CPU y memoria de la computadora. Los recursos se asignan a los procesos que están en cola de espera. Suponiendo que todos tienen una misma prioridad, en el orden en el cual fueron introducidos en la cola.

Representación de Pilas y Colas en Object Pascal.

```
Unit ListSecuencialOO;
Interface

  Const    MaxElem= 10;
  Type

  TypeInfo = string;

  TListSec = Object
  Items: Array [ 1..MaxMem ] of TypeInfo;
  Last: 0..MaxElem;
  Constructor Init;
  Function Empty: Boolean;
  Function Full: Boolean;
  Function Insert (Item: TypeInfo): Boolean;
  Function GetLast: TypeInfo;
  End;

  TStackSec = Object ( TListSec )
  Function Delete: TypeInfo;
  End;
  TQueueSec = Object ( TListSec )
  First: 0..MaxElem;
  Constructor Init;
  Function Empty: Boolean;
  Function Delete: TypeInfo;
  Function GetFirst:TypeInfo;
  End;

{ MÉTODOS de TListSec}
Constructor TListSec.Init;
Begin
Last:=0;
End;

Function TListSec.Full: Boolean;
Var T: TDir;
Begin
Full:= (Last=MaxElem)
End;

Function      TListSec.Insert(Elem:TypeInfo):
Begin
If Full Then Insert:= False
                Else Begin
                    Insert:=true;
                    Inc ( Last );
                    Items [ Last ]:= Item;
                    End;
End;

Function TListSec.GetLast: TypeInfo;
Begin
If Empty Then GetLast:=" "
Else
GetLast:= Items [ Last ];
End;
{ MÉTODOS de TStackSec}

FunctionTStackSec.Delete: TypeInfo;
Begin
If Empty Then Delete:="
Else Begin
    Delete:= Items [ Last ];
    Dec ( Last );
    End;
End;
```

```
Function TListSec.Empty: Boolean;
Begin
Empty:= (Last = 0)
End;
{MÉTODOS de TQueueSec}
Constructor TQueueSec.Init;
Begin
Last:= 0;  First:=1;
End;
Function TQueueSec.Delete: TypeInfo;
  Begin
  If Empty Then Delete:=' '
        Else Begin
              Delete:= Items[First ];
              For i:=1 to Last-1 do
              Items [i]:= Items[i+1];
             Dec( Last );
             End;
 End;
```

```
Function TQueueSec.Empty : Boolean;
Begin
Empty:= (First > Last);
End;

Function TQueueSec.GetFirst: TypeInfo;
Var Temp: TDir;
Begin
If Empty Then GetFirst:="
Else
 GetFirst:= Ítems[First];
End;
```

TAREA o ESTUDIO INDEPENDIENTE.

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

1. _____

2. _____

2. Ponga ejemplos de la vida real donde se maneja el concepto de una cola.

1. _____

2. _____

3.- Implementar las clases estudiadas en este tema.

LABORATORIOS

- **Laboratorio Grupo1:**
  Implementar la aplicación "Evaluar una Expresión Matemática", utilizando Object Pascal y Delphi como entorno de desarrollo.
- **Laboratorio Grupo2:**
  Implementar la clase cola circular, y una interfaz básica para trabajar con ella de forma básica.

SEMINARIO.

- Para el todo lo estudiado en este tema:
  - Haga lo mismo utilizando Visual Studio. Net, utilizando tanto Visual Basic, C y C#.
  - Esto incluye, pilas, aplicaciones de pilas, colas, colas circulares, aplicaciones de colas, lista secuecial, herencias y conceptos de la programación OO.

CLASE PRÁCTICA.

Implemente las siguientes situaciones, utilizando Object pascal y Delphi como Entorno de Desarrollo. Complete en los lugares dejados para ello, la solución.

Ejercicio Nro1:

Se desea hacer un procedimiento que cree una pila con 10 nombres de personas tecleados por el usuario.

Ejercicio Nro2:

En una gasolinera se quiere llevar el control de los autos que arriban a la cola de espera por el servicio. De cada uno de ellos se recogen los siguientes datos:
Carnet del Propietario
Apellidos y nombre del Propietario.
Marca del Auto.
Número de Placa.
La atención a los mismos se realiza por orden de llegada y para el procesamiento de la información la misma es almacenada en una cola que admite como máximo 20 autos.
A medida que los autos van saliendo de la cola por haber realizado el servicio, se introduce la cantidad de gasolina por litros despachada y el monto de la compra realizada es almacenada secuencialmente durante todo día.

Resuelva las siguientes situaciones.

- Defina las variables, clases necesarias.
- Insertar un auto en la cola.
- Sacar un auto serviciado de la cola y almacenar su información en la lista secuencial.

- Listar la información de los autos serviciados en un día según un tipo de propiedad.
- Determinar la cantidad de autos atendidos en el día y el importe total de todos los servicios.