Capítulo III Estructura de Datos Colas.

Contenido:

1. Definición de Colas.
2. Ejemplo de colas en la vida real.
3. Representación gráfica y elementos en una cola.
4. Atributos de la clase Cola.
5. Operaciones Básicas de una Cola.
6. Colas Circulares.
7. Decolas o Bicolas.
8. Aplicaciones de las Colas.
9. Ejercicios prácticos
10. Orientación para el laboratorio.

Introducción

En este capítulo se estudian la Estructura de Datos Colas, Viéndose todos los aspectos básicos relacionados a ellas, así como también las diversas variantes que se pueden tener de cada una de ellas, tales como las colas circulares, las decolas o bicolas.

Definición de Colas.

Una Cola es una estructura de datos en la que los elementos se añaden siempre al final de la lista, 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, por ejemplo:

- Una cola esperando en un consultorio médico para ser atendidos. Los pacientes, serán atendidos, en la medida que llegan.
- 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 8 a continuación.

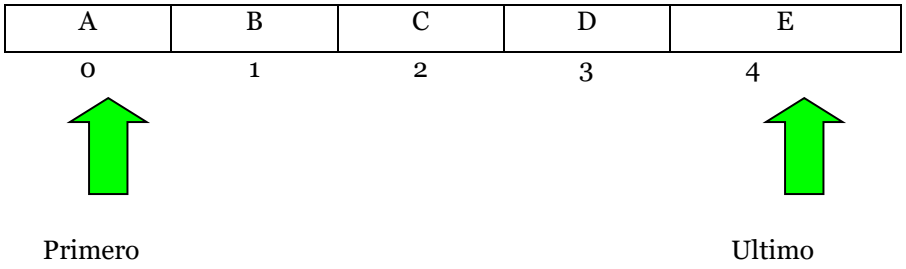


Fig8.

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 o después del último.
- 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 elemento que entró.

Explicación de las Operaciones Básicas de una cola.

Ya conoce como se representa una cola, y sus operaciones básicas; ahora conocerá cómo es que funciona y su utilidad.

Operación Vacía.

Esta operación es la encargada de informar cuando la cola está vacía; 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.

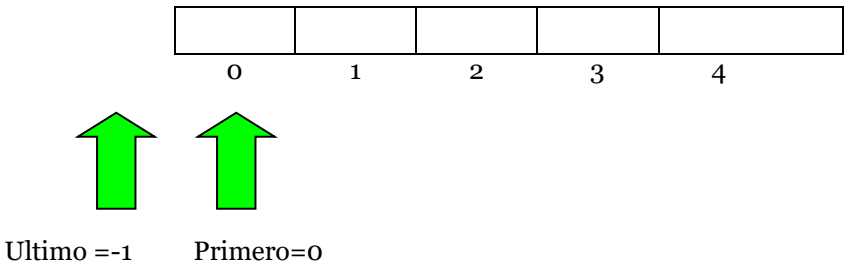


Fig.9

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

A continuación se muestra la función que puede realizar esto:

```
Función  vacia()

           Si  (ultimo < primero)
               Vacia = cierto;
           Sino
               Vacia = false;
```

```
int vacia(struct cola *col)
{
      if (col->ultimo<col->primero)
            return 1;
      else
            return 0;
}
```

Operación Llena.

La Cola está llena cuando están ocupadas todas las casillas desde 0 hasta maxcola-1; 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.

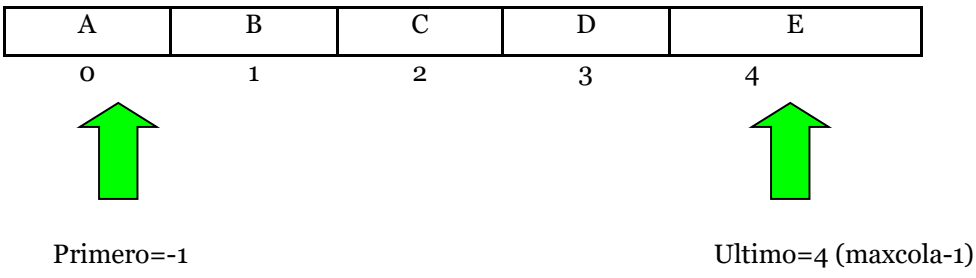


Fig10

La cola estará llena cuando último sea igual a maxcola-1 que en este caso es 4. A continuación se muestra la función que puede realizar esta operación:

```
Función llena()

            Si  (ultimo == maxcola-1 )
                Llena = cierto
            Sino
                Lena= fals0
```

```
int llena(struct cola *col)
{
       if (col->ultimo ==maxcola-1)
              return 1;
       else
              return 0;
}
```

Operación Insertar

Esta 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 se quiere insertar. Para entender esto de mejor forma vea la fig11 y la explicación asociada.

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

Fig.11

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

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

```
Proceso  insertar(valor)

            Si (llena())
                Mensaje ("Error, la cola está llena");
            sino

                incrementar ultimo
                datoscola[ultimo] = valor
```

```
void insertar(int x, struct cola *col)
{
       if (llena(col))
       {cout<<"La cola está llena";
       cout<<endl;
       }
       else
       {
              col->ultimo ++;
              col->items [col->ultimo]=x;
       }
}
```

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 12 y su explicación para comprender con mayor detalle lo anterior.

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

Fig.12

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 -1, y primero en 0, 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 que puede realizar esta operación:

```
Proceso o función  eliminar()

Declaro variable entera contador

            Si (vacia())
                Mensaje de Error("Error, la cola esta vacia");
            sino

                eliminar = datoscola[primero]
                Recorro todos los elementos una posición
                Decrementar ultimo
```

```
int eliminar(struct cola *col)
{
       int contador,valor;
       valor=0;
       if (vacia(col))
       {
              cout<<"La cola esta vacia";
              cout<<endl;
              return 0;
       }
       else
       {
              valor= col->items[col->primero];
              cout<<valor;
              for(contador=0; (contador <col->ultimo);
contador++);
              {
                     (col->items[contador])=(col->items[ contador
+1 ]);
              }
              col->ultimo--;
              return valor;
       }
}
```

Representación Básica de una cola utilizando C++ y Visual Studio .NET como entorno de desarrollo, la aplicación que se desarrolla es en consola.

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

struct cola{
       int primero,ultimo;
       int items[maxcola];
       };

       struct cola q;

int iniciar()
{
       q.primero=0;
       q.ultimo=-1;
       return 0;
}

int vacia()
{
       if (q.ultimo<q.primero)
              return 1;
       else
           return 0;
}

int llena()
{
       if (q.ultimo ==maxcola-1)
              return 1;
       else
           return 0;
}


void insertar(int x)
{
if (llena())
       {cout<<"La cola esta llena";
       cout<<endl;
}
       else
       {
              q.ultimo ++;
              q.items [q.ultimo]=x;
}
}

int eliminar()
{
       int i, valor;
```

```
    if (vacia())
       {cout<<"La cola esta vacia";
       cout<<endl;
       return 0;
}
else
{

    valor= q.items[q.primero];
    for(i=0;i<q.ultimo;i++)
           {
              q.items[i]=q.items[i+1];
        }
        q.ultimo--;
        return valor;
}
}


int main(void)

{
       iniciar();
       //Insertando cinco elementos en la cola
       insertar(1);
       insertar(2);
       insertar(3);
       insertar(4);
       insertar(5);
       cout<<q.items[0];
       cout<<endl;
    cout<<q.items[1];
       cout<<endl;
    cout<<q.items[2];
    cout<<endl;
    cout<<q.items[3];
    cout<<endl;
    cout<<q.items[4];
    cout<<endl;
    cin.get();
       cout<<"Eliminado elementos";
       cout<<eliminar();
       cout<<endl;
    cin.get();
       cout<<eliminar();
       cout<<endl;
    cin.get();
       cout<<eliminar();
       cout<<endl;
    cin.get();
       cout<<eliminar();
       cout<<endl;
    cin.get();
       cout<<eliminar();
       cout<<endl;
    cin.get();
       cout<<eliminar();
```

```
        cout<<endl;
    cin.get();
        return 0;

}
```

Representación Básica de una cola utilizando C y el compilador ZINJAL.

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


struct cola{
        int primero,ultimo;
        int items[maxcola];
};



int iniciar(struct cola *col)
{
int contador;
col->primero=0;
col->ultimo=-1;
for(contador=0; contador <maxcola; contador++);
col->items[contador]=0;

return 0;
}

int vacia(struct cola *col)
{
        if (col->ultimo<col->primero)
                return 1;
        else
                return 0;
```

```
}

int llena(struct cola *col)
{
        if (col->ultimo ==maxcola-1)
                return 1;
        else
                return 0;
}


void insertar(int x, struct cola *col)
{
        if (llena(col))
        {cout<<"La cola esta llena";
        cout<<endl;
        }
        else
        {
                col->ultimo ++;
                col->items [col->ultimo]=x;
        }
}

int eliminar(struct cola *col)
{
        int contador,valor;
        valor=0;
        if (vacia(col))
        {
                cout<<"La cola esta vacia";
                cout<<endl;
                return 0;
        }
        else
        {
                valor= col->items[col->primero];
```

```
                cout<<valor;
                for(contador=0; (contador <col->ultimo); contador++);
                {
                        (col->items[contador])=(col->items[ contador +1 ]);
                }
                col->ultimo--;
                return valor;
        }
}



int main(void)

{
        struct cola q;
        iniciar(&q);
        //Insertando cinco elementos en la cola
        insertar(10,&q);
        insertar(20,&q);
        insertar(30,&q);
        insertar(40,&q);
        insertar(50,&q);
        cout<<q.items[0];
        cout<<endl;
        cout<<q.items[1];
        cout<<endl;
        cout<<q.items[2];
        cout<<endl;
        cout<<q.items[3];
        cout<<endl;
        cout<<q.items[4];
        cout<<endl;
        cin.get();
```

```
    return 0;

}
```

Colas Circulares.

Definición

Para hacer un uso más eficiente de la memoria disponible, se trata a la cola como una estructura circular, determinando el algoritmo de colas circulares. La figura 13 muestra gráficamente una cola circular.

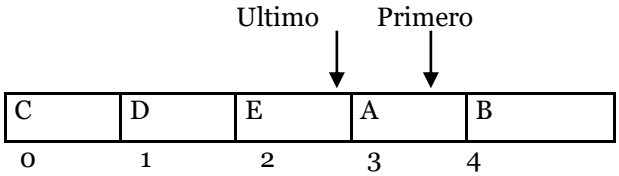


Fig.13

En la figura anterior se representa un cola, donde Primero nos dice donde se encuentra el primer elemento, en este caso en la posición 3; y Ultimo nos dice donde se encuentra el último elemento, en este caso en la posición 2.

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 14. Se Asume de que ultimo y primero valen de inicio -1, ya primero no se inicia en la posición 0 del arreglo, ya que primero en una cola circular estará en cualquier otra posición dentro del arreglo.

Ultimo = -1 Primero =-1

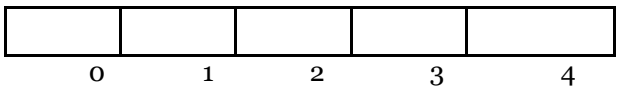


Fig14

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

Function Vacía

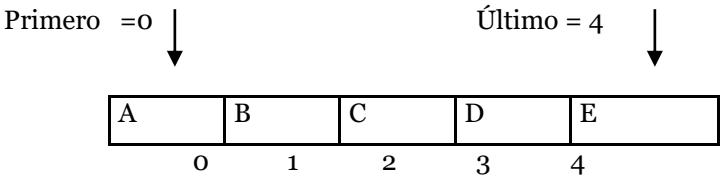
Si (Ultimo=-1 and primero=-1) entonces Vacía= Cierto

sino

Vacía = Falso

Operación Llena

Gráficamente se puede representar a la cola circular llena como se muestra en la figura 15.



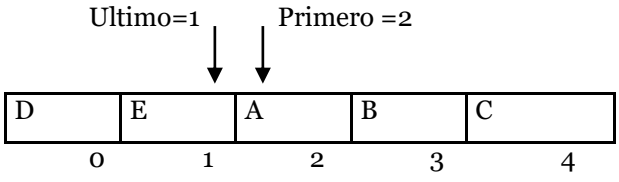


Fig15

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

Function Llena

Si (Ultimo=MaxCola-1) y (Primero=0)) ó (Ultimo+1=Primero) entonces Llena = cierto

sino

Llena = Falso

Operación de Insertar.

La inserción se explica en la siguiente figura (figura 16):

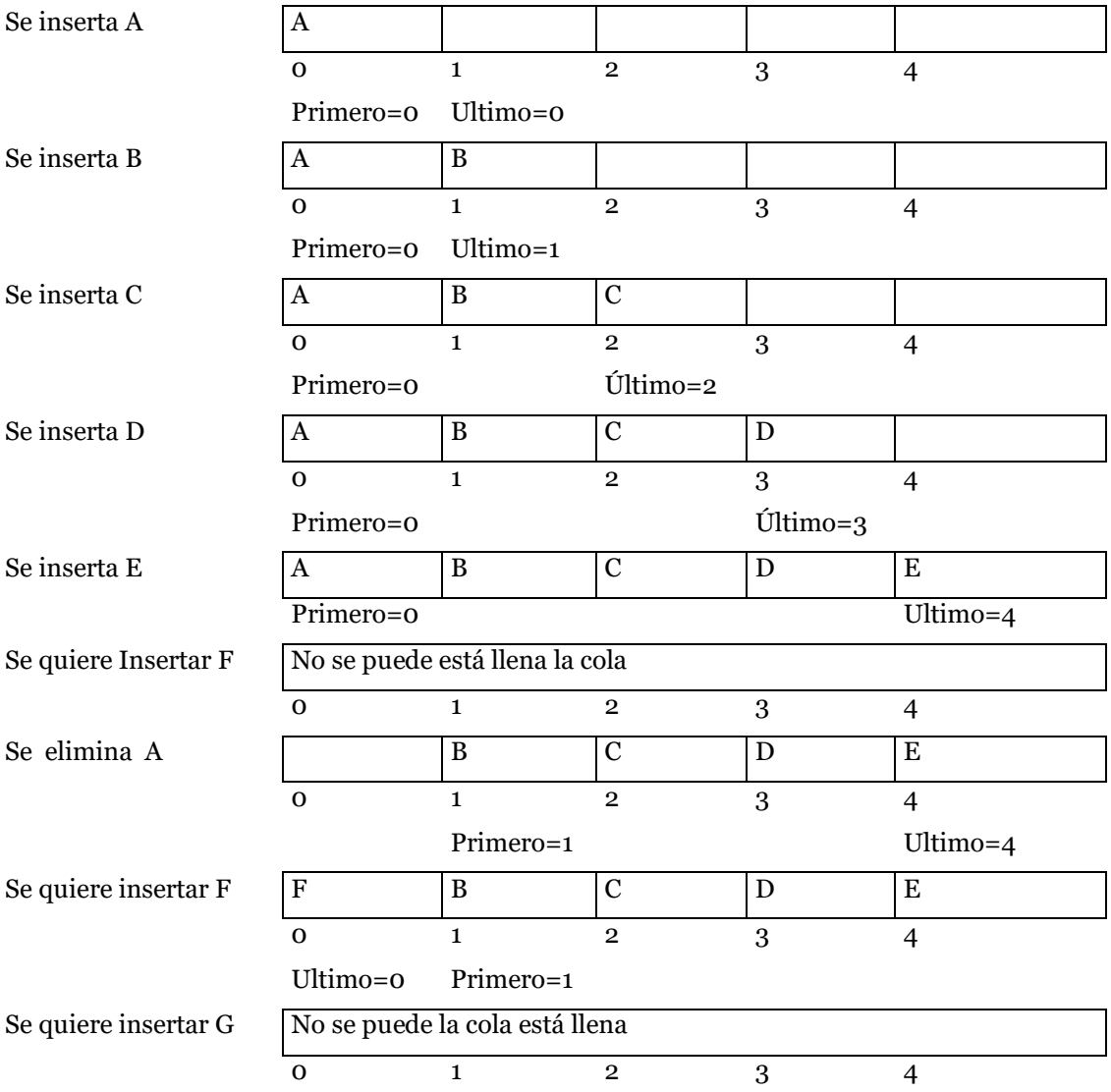


Fig16

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

Proceso Insertar(elem: tipo-elem)

```
si  LLena mostrar ('Cola  Llena')
sino

Si  ultimo=Maxcola-1 entonces
      ultimo=0
      sino    inc(último )
info[ultimo]=x;
si primero=-1 entonces Primero=0
```

Operación Eliminar

Gráficamente se puede representar la eliminación como se muestra en la siguiente figura (figura 17).

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

Fig17

A continuación se muestra la implementación en seudocodigo de la Función Eliminar.

```
Function Eliminar( elem: tipo-elem )
{
si Vacia entonces mostrar('Cola Vacía)
sino
{
Devolver info[primero];
si primero=ultimo entonces
{
        primero=-1
         ultimo=-1
}
Sino
Si primero=maxcola-1 entonces primero=0
Sino primero=primero+1
}
}
```

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:

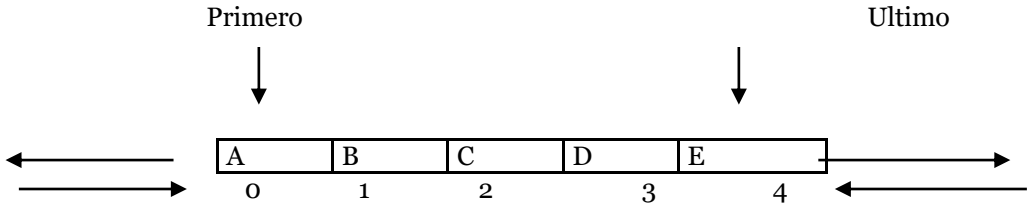


Fig.18

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.
- Vacía
- Llena

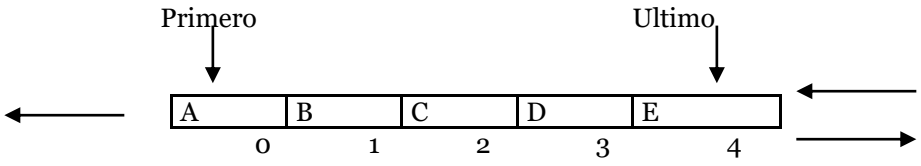
Variantes de las DECOLAS:

- 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.

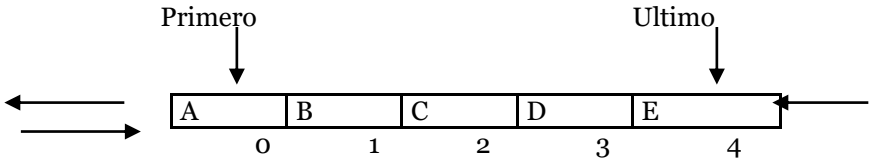
En la siguiente figura se representa gráficamente este caso:



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 se puede ver 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.

Prácticos a realizar.

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

1.

2.

2. Eliminar un determinado elemento de una cola.

3. Invertir los elementos en una cola.

4. Buscar si existe un determinado elemento en una cola.

5. Insertar un elemento en una cola, detrás de otro elemento específico.

LABORATORIO Nro. 2.

1.- Implementar una Cola, como la vista en clases y realice un menú principal, donde el usuario seleccione como opciones las operaciones básicas de la cola y pueda ingresar los diferentes valores a través del teclado.

2.- Implementar una cola circular, realice un menú principal

3.- Implementar una decola, realice un menú principal.