Facultad:	Ingeniería
Escuela:	Computación
Asignatura:	Programación con Estructuras de Datos

Tema: Estructura Cola.

Competencia

- Desarrolla sistemas de información informáticos mediante la integración de principios matemáticos, ciencia computacional y prácticas de ingeniería, considerando estándares de calidad y mejores prácticas validadas por la industria del software.

Materiales y Equipo

- Guía Número 4
- Computadora con programa Microsoft Visual C#.

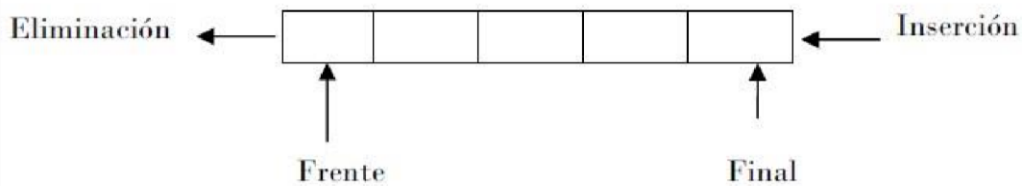
Introducción Teórica

ESTRUCTURA DINÁMICA COLA

Las colas son llamadas también estructuras **FIFO** (**first-in, first-out, primero en entrar, primero en salir**). Sus aplicaciones son numerosas: colas de las tareas realizadas por una impresora, acceso a almacenamiento en disco, sistema compartido y uso del CPU.

Como su definición lo dice, en una cola los elementos deben insertarse por la final de la cola y eliminarse por el frente, así se cumple la definición primero en entrar, primero en salir.

Gráficamente una cola puede representarse de la siguiente manera:



Al igual que en la representación de listas, el último elemento de la cola debe apuntar a NULL para indicar el final de la misma. Las colas también son estructuras dinámicas que pueden aumentar o disminuir en tiempo de ejecución, la única diferencia es que aquí se manejan dos punteros una indicando el frente o inicio de la cola, y el otro para representar el final de la cola.

Entre las funciones más comúnmente usadas sobre colas están: Inicializar, Insertar, Eliminar, Cola Vacía.

Procedimiento

Ejemplo 1

1. Cree un proyecto en Visual C# en modo consola (nómbrelo como usted quiera) y cree dos clases adicionales al proyecto: la clase **nodo** y la clase **Cola**

2. En la clase **nodo** codifique lo siguiente:

```
class nodo
{
    public int info; //atributo para dato almacenado
    public nodo sgte; //atributo para enlace a otros nodos
}
```

3. En la clase **Cola** tendremos el siguiente código (se presentan bloques por cada método):

```
class Cola
{
    public nodo primero; //inicio de la cola
    public nodo ultimo; //final de la cola

    public Cola()
    {
        primero = ultimo = null;
    }
```

Atributos y constructor

Método Encolar

```
public void Encolar (int valor)
{
    nodo aux = new nodo();
    aux.info = valor;
    if(primero == null)  //en caso de que la cola esté vacia
    {
        primero = ultimo= aux;  //ingresamos dato a la cola
        aux.sgte = null;         //hacemos que señale a null el nodo
    }
    else //si la cola no está vacia
    {
        ultimo.sgte = aux; //el último elemento va a señalar a aux
        aux.sgte = null;
        ultimo = aux; //metemos el nodo a la cola
    }
}
```

Método Desencolar

```
public void Desencolar() //desencola sin decirme valor
{
    if (primero == null) Console.WriteLine("Cola Vacia"); //si la cola está vacia enviamos mensaje
    else primero = primero.sgte; //cambiamos de posición el apuntador primero
}
```

Método Desencolar devolviendo el valor

```
public int DesencolarValor() //desencola y me muestra valor desencolado
{
    int valor = 0;
    if(primero ==null) Console.WriteLine("Cola vacia");
    else
    {
        valor = primero.info;
        primero = primero.sgte;
    }
    return valor;
}
```

```
    public void Mostrar()
    {
        if (primero == null) Console.WriteLine("Cola vacia");
        else
        {
            nodo puntero;
            puntero = primero;

            do
            {
                Console.Write("{0}\t", puntero.info);
                puntero = puntero.sgte;
            }
            while (puntero != null);
        }
        Console.WriteLine("\n");
    }
} //fin de la clase Cola
```

Método Mostrar

4. En el Main o programa principal escribimos el siguiente código

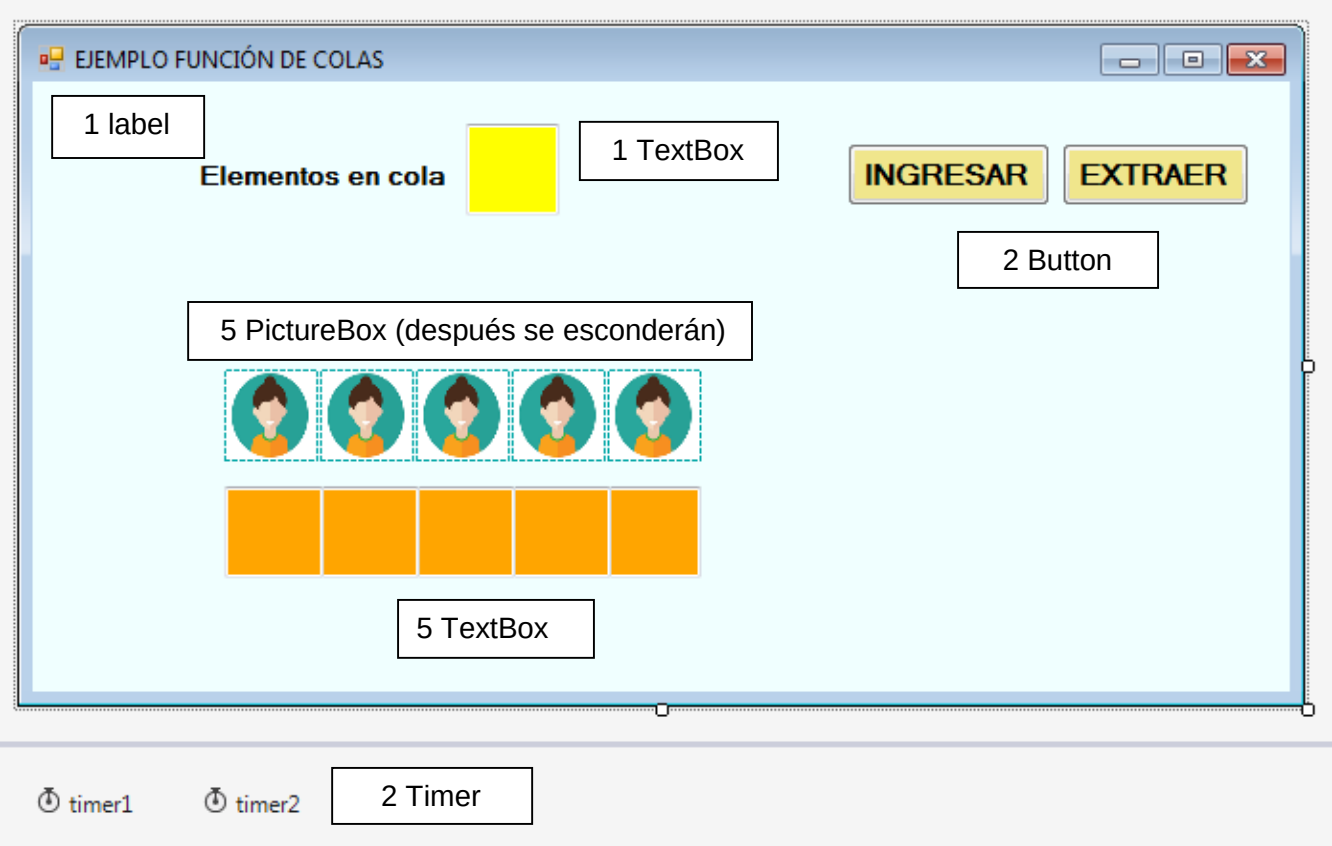
```
static void Main(string[] args)
{
    Cola objcola = new Cola();
    Console.WriteLine("Colocando 5 elementos en la cola");

    objcola.Encolar(3);
    objcola.Encolar(27);
    objcola.Encolar(5);
    objcola.Encolar(22);
    objcola.Encolar(23);
    objcola.Mostrar();

    Console.WriteLine("Retirando dos elementos en cola");
    objcola.Desencolar();
    objcola.Mostrar();
    objcola.Desencolar();
    objcola.Mostrar();

    Console.WriteLine("Se va a retirar un nodo más, con el valor de {0}", objcola.DesencolarValor());

    objcola.Mostrar();
    Console.ReadLine();
}
```

5. Ejecute el programa y observe su funcionamiento.

Ejemplo 2

Si observamos el ejercicio anterior podemos aplicar correctamente una clase cola y su funcionamiento en entorno consola. El ejemplo 2 se enfoca en mostrar una simulación de ese funcionamiento en un entorno gráfico (valiéndonos de las herramientas que ya nos ofrece C#). **Cabe aclarar que este ejemplo NO implementa clases de cola, sino que simplemente muestra cómo debe ser su simulación gráfica.**

1. Cree un formulario o ventana con características similares a las descritas en la imagen (se incluyen todas las herramientas que necesitará)



Consideraciones iniciales:

a. Una vez colocados todas las herramientas requeridas se debe dar el formato visual necesario para que la simulación se muestre como esperamos, no se preocupe si deja de ver los PictureBox. A continuación se ofrece una tabla con las modificaciones que se realizaron (SÓLO SE DETALLAN LOS QUE NECESITAN CAMBIOS ESPECÍFICOS):

Herramienta	Ubicación	Tamaño
TextBox1	x= 300 y= 215	49,49
TextBox2	x= 250 y=215	
TextBox3	x=200 y=215	
TextBox4	x=150 y=215	
TextBox5	x=100 y=215	
Form1		672,356

Herramienta	Ubicación	Tamaño
PictureBox1	x= -50 y=150	49,49
PictureBox2		
PictureBox3		
PictureBox4		
PictureBox5		

Herramienta	Intervalo y enabled
Timer1	Intervalo = 10
Timer2	Enabled = false

2. Entrar al código y comenzar a escribir lo siguiente:

```
public partial class Cola : Form
{
    int total = 0; //cantidad de elementos en cola
    Random numeros = new Random(); //los valores serán random

    public Cola()
    {
        InitializeComponent();
        txttotal.Text = Convert.ToString(total);
        //en textbox superior se mostrará el conteo de total
    }
```

3. Activamos el evento click del botón para ingresar:

```
private void btningresar_Click(object sender, EventArgs e)
{
    if( total ==5)
    { MessageBox.Show("La cola está llena"); }
```

```
        else
        {
            total++;  //incrementamos cantidad de elementos en cola
            txttotal.Text = Convert.ToString(total);

            switch(total) //dependiendo de cuantos números hay en cola
            {
                case 1:
                    txtn1.Text = Convert.ToString(numeros.Next(1,99));
                    txtn1.Visible = true;
                    break;
                case 2:
                    txtn2.Text = Convert.ToString(numeros.Next(1, 99));
                    txtn2.Visible = true;
                    break;
                case 3:
                    txtn3.Text = Convert.ToString(numeros.Next(1, 99));
                    txtn3.Visible = true;
                    break;
                case 4:
                    txtn4.Text = Convert.ToString(numeros.Next(1, 99));
                    txtn4.Visible = true;
                    break;
                case 5:
                    txtn5.Text = Convert.ToString(numeros.Next(1, 99));
                    txtn5.Visible = true;
                    break;
                default:
                    MessageBox.Show("Error en la cola");
                    break;
            }
            timer1.Enabled = true;
        }
    }
```

4. A continuación programamos el evento click del botón extraer

```
private void btnextraer_Click(object sender, EventArgs e)
{
    if (total == 0)
    { MessageBox.Show("Cola vacia, no puede eliminar datos"); }
    else
    {   //en realidad pasamos la información de un cuadro a otro
        txtn1.Text = txtn2.Text;
        txtn2.Text = txtn3.Text;
        txtn3.Text = txtn4.Text;
        txtn4.Text = txtn5.Text;

        //descartamos los nodos que no usamos, los dejamos no visibles
        if (total == 1) txtn1.Visible = false;
        if (total == 2) txtn2.Visible = false;
        if (total == 3) txtn3.Visible = false;
        if (total == 4) txtn4.Visible = false;
        if (total == 5) txtn5.Visible = false;
        total--;
        txttotal.Text = Convert.ToString(total);
        timer2.Enabled = true;
    }
}
```

5. Programamos el evento Tick del primer timer

```
private void timer1_Tick(object sender, EventArgs e)
{
    btningresar.Enabled = false; //deshabilitamos el botón agregar
    if(total ==1)
    {   pbox1.Left = pbox1.Left + 5; //hacemos que se desplace de 5 en 5
        if (pbox1.Left >= 300) timer1.Enabled = false;  //detenemos el timer cuando llegue a 300
    }
    if (total == 2)
    {   pbox2.Left = pbox2.Left + 5;
        if (pbox2.Left >= 250) timer1.Enabled = false;  //detenemos el timer cuando llegue a 250
    }
    if (total == 3)
    {   pbox3.Left = pbox3.Left + 5;
        if (pbox3.Left >= 200) timer1.Enabled = false; //detenemos el timer cuando llegue a 200
    }
    if (total == 4)
    {   pbox4.Left = pbox4.Left + 5;
        if (pbox4.Left >= 150) timer1.Enabled = false; //detenemos el timer cuando llegue a 150
    }
```

```
    if (total == 5)
    {   pbox5.Left = pbox5.Left + 5;
        if (pbox5.Left >= 100) timer1.Enabled = false; //detenemos el timer cuando llegue a 100
    }
    if (timer1.Enabled == false) btningresar.Enabled = true; //timer apagado, enciende botón
}
```

6. Finalmente se programa el evento Tick del segundo timer

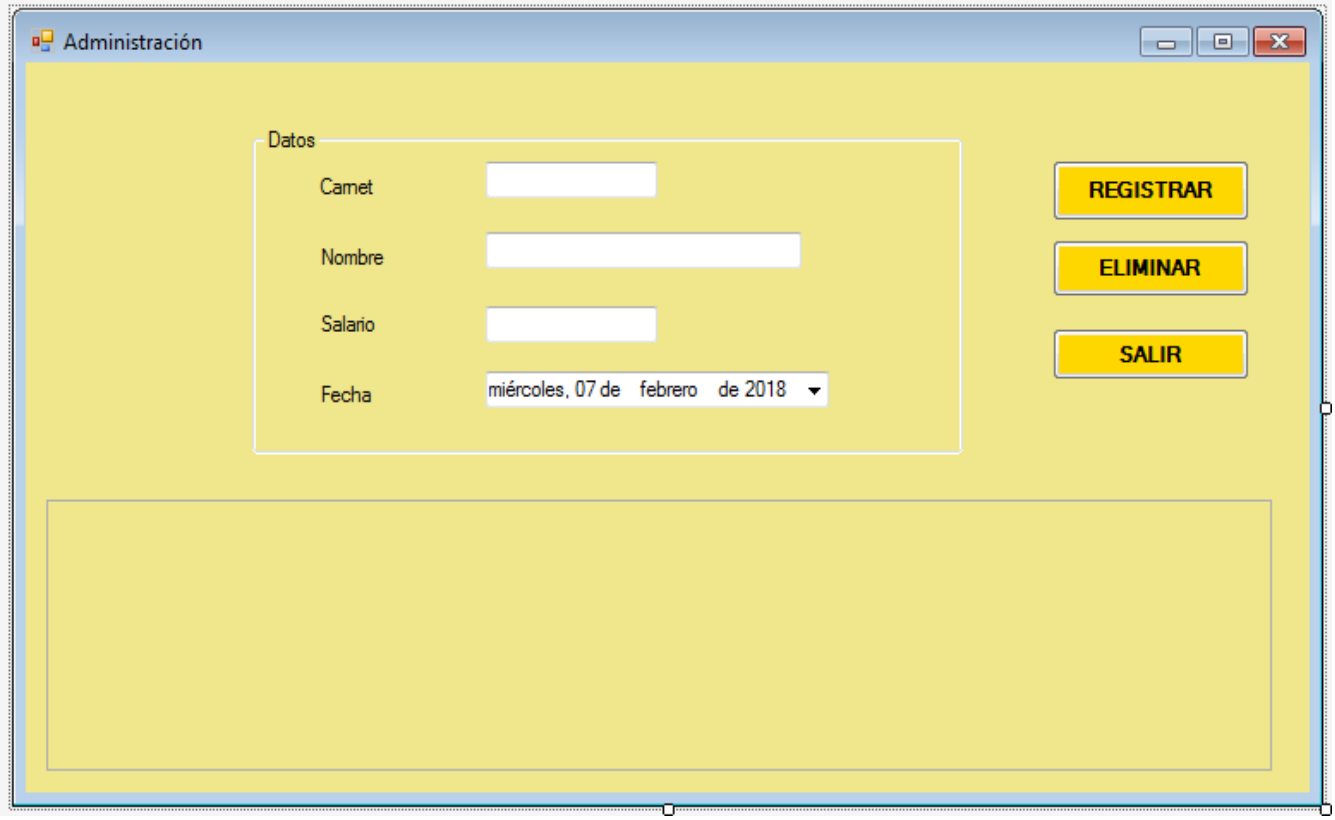
```
private void timer2_Tick(object sender, EventArgs e)
{
    btnextraer.Enabled = false;  //deshabilitamos extraer
    pbox1.Left = pbox1.Left + 5;  //movemos la imagen a la derecha
    if (pbox1.Left >= 600)        //al esconderse a la derecha
    {
        pbox1.Left = 300;  //regresamos el primer picturebox a la primera posición
        if (total == 4) pbox5.Left = -50;  //escondo el picturebox5
        if (total == 3) pbox4.Left = -50; //escondo el picturebox4
        if (total == 2) pbox3.Left = -50;  //escondo el picturebox3
        if (total == 1) pbox2.Left = -50;  //escondo el picturebox2
        if (total == 0) pbox1.Left = -50;  //escondo el picturebox1
        timer2.Enabled = false; //deshabilito timer
    }
    if (timer2.Enabled == false) btnextraer.Enabled = true;  //habilito botón de nuevo
}
```

Si observa el ejemplo anterior nos muestra cómo se mueve y visualiza una cola pero en realidad no utilizábamos clase cola o nodos asociados, tiene como objetivo solamente mostrar cómo se atienden los nodos. Un punto importante es que notemos que los timer aunque dan respuesta a una necesidad momentánea, no son la mejor alternativa.

Ejemplo 3

En el ejemplo 3 veremos el uso de la clase cola que ya nos ofrece C# (QUEUE).

1. Cree un nuevo proyecto de WindowsForm en C# que luzca similar a la siguiente imagen:
    - 3 TextBox
    - 3 Button
    - 1 DataGridView
    - 1 DateTimePicker
    - 1 GroupBox
    - 4 Label



2. Al proyecto le añadiremos una clase denominada **Empleados** con cuatro propiedades

```
class Empleados
{
    public string Carnet { get; set;}
    public string Nombre { get; set;}
    public decimal Salario { get; set; }
    public DateTime Fecha { get; set; }
}
```

3. En el código del formulario (F7) digitaremos antes del constructor, lo siguiente:

```
Queue<Empleados> Trabajadores = new Queue<Empleados>();
//creamos objeto de la clase cola, del tipo de la clase empleado (lo que almacena son objetos)
```

4. En el evento Click del botón de registro

```
private void btnRegistrar_Click(object sender, EventArgs e)
{
    Empleados empleado = new Empleados(); //creamos instancia de la clase empleado
    //capturamos los datos del empleado
    empleado.Carnet = txtcarnet.Text;
    empleado.Nombre = txtnombre.Text;
    empleado.Salario = Decimal.Parse(txtsalario.Text);
    empleado.Fecha = fecha.Value;
    Trabajadores.Enqueue(empleado);  //llamamos al método encolar para meter a la estructura
    dgvCola.DataSource = null; //iniciamos el datagridview con null
    dgvCola.DataSource = Trabajadores.ToArray(); //para pasarlo al dgv convertimos la cola en arreglo
    Limpiar(); //se limpian los textbox
    txtcarnet.Focus(); //se coloca el cursor sobre el primer textbox
}
```

5. En el evento Click del botón de Eliminar digitamos lo siguiente

```
private void btnEliminar_Click(object sender, EventArgs e)
{
    if (Trabajadores.Count != 0) //mientras haya trabajadores en la cola
    {
        Empleados empleado = new Empleados(); //instanciamos de la clase empleado
        /*este objeto se usa para poder recuperar los datos y mostrarlos en los textbox
         al momento de ser eliminados de la cola*/
        empleado = Trabajadores.Dequeue(); //llamamos al método desencolar
        //colocamos los datos en sus textbox
        txtcarnet.Text = empleado.Carnet;
        txtnombre.Text = empleado.Nombre;
        txtsalario.Text = empleado.Salario.ToString();
        fecha.Value = empleado.Fecha;
        //la estructura convertida en lista se le pasa al dgv (ahora tiene un empleado menos)
        dgvCola.DataSource = Trabajadores.ToList();
        MessageBox.Show("Se eliminó el registro en cola","AVISO");
        Limpiar();
    }
    else
    {
        MessageBox.Show("No hay empleados en cola","AVISO");
        Limpiar();
    }
    txtcarnet.Focus();
}
```

6. Codificamos el método Limpiar( )

```
public void Limpiar() //limpia los TextBox
{
    txtcarnet.Clear();
    txtnombre.Clear();
    txtsalario.Clear();
}
```

7. Codificamos el evento Click del botón de salida

```
private void btnSalir_Click(object sender, EventArgs e)
{
    Application.Exit(); //salir de la aplicación
}
```

Desarrollo de habilidades

Ejercicio 1

Modifique el ejercicio 3 de forma que incluya validaciones (campos vacíos, que el salario sea negativo o datos del tipo que no corresponde, entre otros). Utilice ErrorProvider para manejar estas validaciones.

Ejercicio 2

Realizar la simulación de una cola bancaria, el banco contará con 4 cajeros, cuando una persona entre al banco, deberá seleccionar la cola que tenga menos personas, el cajero podrá retirar las personas luego de realizar los trámites

Cajero 1	Cajero 2	Cajero 3	Cajero 4
1	1	1	1
1	1		1
1			

Ingrese una persona y ve que la cola con menos personas es la del cajero 3

Cajero 1	Cajero 2	Cajero 3	Cajero 4
1	1	1	1
1	1	1	1
1			

El cajero 1 despacha a una persona

Cajero 1	Cajero 2	Cajero 3	Cajero 4
1	1	1	1
1	1	1	1

Se puede realizar en modo consola o gráfico, cuando más de un cajero tenga el mismo número de elementos en la cola la persona puede escoger cualquier cajero.

Sugerencia: Le recomiendo utilizar más de un objeto cola.

Investigación Complementaria

Investigación 1

Documéntese sobre los hilos en C# (System.Threading.Tasks) y ¿Por qué serían más eficientes que los Timer en el Ejemplo 2?, ¿Cómo podría utilizarlos de forma que incluya lo realizado en los ejercicio 1 y 2?