

UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE ELECTROTECNIA Y COMPUTACIÓN

---

# Algoritmización y Estructuras de Datos

---

*Profesor:*

Adilson G. López

Grupo: 2M1 - CO

Desarrolladores:

Gabriel A. Ortiz — 2020 - 0325U

Marcel E. Díaz — 2020 - 1384U

# Índice

|  |           |
|--|-----------|
| <b>1. Introducción</b>                       | <b>2</b>  |
| <b>2. Código Fuente</b>                      | <b>3</b>  |
| 2.1. Variables globales . . . . .            | 3         |
| 2.2. Variables globales . . . . .            | 4         |
| 2.3. Eventos . . . . .                       | 5         |
| 2.3.1. Evento de cargar formulario . . . . . | 5         |
| 2.3.2. Evento insertar . . . . .             | 5         |
| 2.3.3. Evento eliminar . . . . .             | 5         |
| 2.3.4. Evento buscar . . . . .               | 6         |
| 2.3.5. Evento establecer . . . . .           | 6         |
| 2.3.6. Evento mostrar . . . . .              | 7         |
| 2.3.7. Evento modificar . . . . .            | 7         |
| 2.4. Métodos . . . . .                       | 8         |
| 2.4.1. Método para limpiar . . . . .         | 8         |
| 2.4.2. Método para buscar . . . . .          | 8         |
| 2.4.3. Método para insertar . . . . .        | 9         |
| 2.4.4. Método para eliminar . . . . .        | 10        |
| 2.4.5. Método para actualizar . . . . .      | 10        |
| <b>3. Conclusión</b>                         | <b>11</b> |

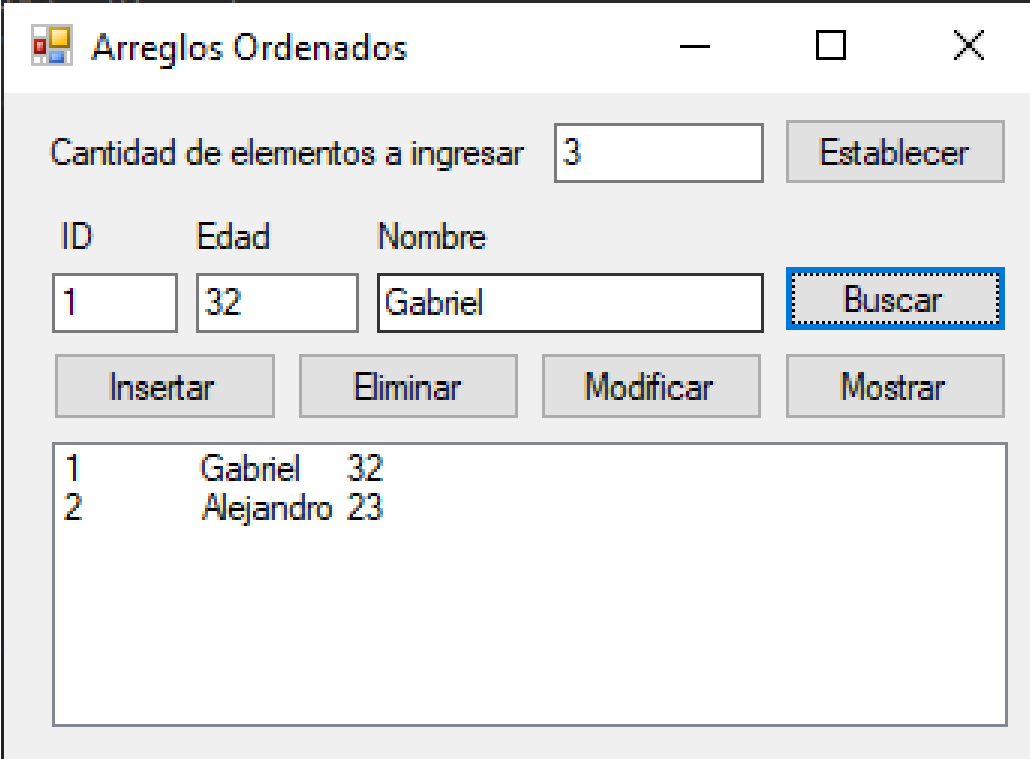
## 1. Introducción

Los arreglos ordenados poseen distintos métodos para efectuar su ordenamiento, los tres más comunes es el algoritmo de burbuja mayor, el algoritmo de burbuja menor y el algoritmo de burbuja con señal. Estos algoritmos tratan de distintas maneras el ordenamiento de números en un arreglo.

El lenguaje utilizado para este laboratorio es C#, lenguaje creado por Microsoft. Utilizamos en su misma medida .NET Framework con el IDE de Visual Studio 2019. Con el fin de aprender las herramientas que posee tanto el lenguaje con el entorno de desarrollo a la vez que se aprenden distintos algoritmos de manejo de datos.

## 2. Código Fuente

### 2.1. Variables globales




The screenshot shows a Java Swing window titled "Arreglos Ordenados". The window contains a form with the following elements:

- A label "Cantidad de elementos a ingresar" followed by a text input field containing the value "3" and an "Establecer" button.
- Three labels: "ID", "Edad", and "Nombre".
- Three text input fields: the first contains "1", the second contains "32", and the third contains "Gabriel".
- A "Buscar" button, which is highlighted with a blue dashed border.
- Four buttons: "Insertar", "Eliminar", "Modificar", and "Mostrar".
- A list box at the bottom containing two entries:

|   |           |    |
|---|-----------|----|
| 1 | Gabriel   | 32 |
| 2 | Alejandro | 23 |


## 2.2. Variables globales



```
1 int n = 0, size, i, pos, x, x2;  
2 int[] Age, Id;  
3 string[] Names;
```


## 2.3. Eventos

### 2.3.1. Evento de cargar formulario




```
1 private void RA2_Load(object sender, EventArgs e)
2 {
3     btnUpdate.Enabled = false;
4 }
```

### 2.3.2. Evento insertar



```
1 private void BtnInsert_Click(object sender, EventArgs e)
2 {
3     x = int.Parse(txtId.Text);
4     Insert(x);
5     Clean();
6 }
7 }
```

### 2.3.3. Evento eliminar



```
1 private void BtnDelete_Click(object sender, EventArgs e)
2 {
3     x = int.Parse(txtId.Text);
4     Delete(x);
5     Clean();
6 }
```

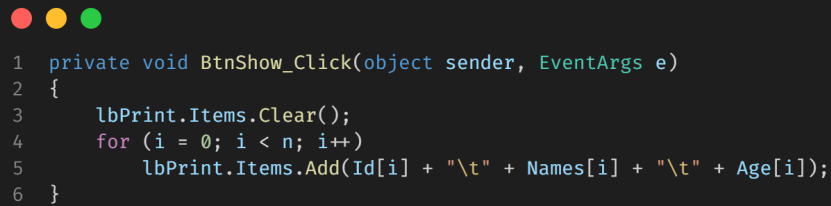
#### 2.3.4. Evento buscar

```
1 private void BtnSearch_Click(object sender, EventArgs e)
2 {
3     if (n > 0)
4     {
5         x2 = int.Parse(txtId.Text);
6         pos = Search(x2);
7         if (pos ≤ -1)
8             MessageBox.Show(x2 + " no esta registrado");
9         else
10        {
11            txtAge.Text = Age[pos].ToString();
12            txtName.Text = Names[pos];
13            btnUpdate.Enabled = true;
14        }
15    }
16    else
17        MessageBox.Show("No hay registros");
18 }
```

#### 2.3.5. Evento establecer

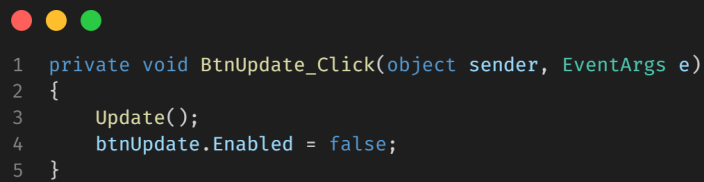
```
1 private void BtnSet_Click(object sender, EventArgs e)
2 {
3     size = int.Parse(txtCount.Text);
4     Id = new int[size];
5     Age = new int[size];
6     Names = new string[size];
7     n = 0;
8 }
```

### 2.3.6. Evento mostrar



```
1 private void BtnShow_Click(object sender, EventArgs e)
2 {
3     lbPrint.Items.Clear();
4     for (i = 0; i < n; i++)
5         lbPrint.Items.Add(Id[i] + "\t" + Names[i] + "\t" + Age[i]);
6 }
```

### 2.3.7. Evento modificar



```
1 private void BtnUpdate_Click(object sender, EventArgs e)
2 {
3     Update();
4     btnUpdate.Enabled = false;
5 }
```



## 2.4. Métodos

### 2.4.1. Método para limpiar

```
1 void Clean()  
2 {  
3     txtId.Clear();  
4     txtName.Clear();  
5     txtAge.Clear();  
6 }
```

### 2.4.2. Método para buscar

```
1 int Search(int x)  
2 {  
3     i = 0;  
4  
5     while (i < n && Id[i] < x)  
6         i++;  
7  
8     if (i ≥ n || Id[i] > x)  
9         pos = -i;  
10    else  
11        pos = i;  
12  
13    return pos;  
14 }
```

### 2.4.3. Método para insertar

```
1 void Insert(int x)
2 {
3     if(n > 0)
4     {
5         if (n ≤ size - 1)
6         {
7             pos = Search(x);
8             if (pos > 0)
9                 MessageBox.Show("El elemento ya existe");
10            else
11            {
12                pos -= 1;
13                for (i = n; i ≥ pos + 1; i--)
14                {
15                    Id[i] = Id[i - 1];
16                    Names[i] = Names[i - 1];
17                    Age[i] = Age[i - 1];
18                }
19
20                Id[pos] = int.Parse(txtId.Text);
21                Age[pos] = int.Parse(txtAge.Text);
22                Names[pos] = txtName.Text;
23                n++;
24                MessageBox.Show("Elemento insertado");
25            }
26        }
27    }
28    else
29    {
30        Id[n] = int.Parse(txtId.Text);
31        Age[n] = int.Parse(txtAge.Text);
32        Names[n] = txtName.Text;
33        n++;
34        MessageBox.Show("Elemento Insertado");
35    }
36 }
37
38 }
```

#### 2.4.4. Método para eliminar

```
1 void Delete(int x)
2 {
3     if (n > 0)
4     {
5         x = int.Parse(txtId.Text);
6         pos = Search(x);
7
8         if (pos ≤ -1)
9             MessageBox.Show(x + " no esta registrado");
10        else
11        {
12            for (i = pos; i < n - 1; i++)
13            {
14                Id[i] = Id[i + 1];
15                Names[i] = Names[i + 1];
16                Age[i] = Age[i + 1];
17            }
18            n--;
19            MessageBox.Show("La persona con id=" + x + " se ha eliminado");
20        }
21    }
22    else
23        MessageBox.Show("No hay registros");
24 }
```

#### 2.4.5. Método para actualizar

```
1 void Update()
2 {
3     x = int.Parse(txtId.Text);
4     Delete(x);
5     Insert(x);
6 }
```

### **3. Conclusión**

En la anterior práctica se vieron la manipulación de arreglos desordenados, donde únicamente se introducía el ítem en el último espacio del arreglo, sin embargo aquí cambia el proceso, debido a que se ordena según su número de id, ubicándolos de manera ascendente.