



Universidad Nacional de Ingeniería

.- Facultad de Electrotecnia y Computación -.

Ingeniería en Computación

Algoritmización y estructuras de datos

Laboratorio No. 3

Alumno:

Marcel Enmanuel Díaz Largaespada (2020-1384U)

Docente:

Adilson González López

Fecha de entrega:

Lunes 23 de agosto del 2021

Índice

1. Introducción.....	2
2. Diseño del <i>form</i>	3
3. Código Fuente.....	4
3.1. Declaraciones y métodos iniciales.	4
3.2. Métodos de ordenamiento.	5
3.3. Métodos de interacción con la interfaz.	8
4. Ejecución del programa	10
5. Conclusiones	13

1. Introducción

Para el ejercicio de este laboratorio se emplearon nuevamente los forms con los que hemos venido trabajando en los laboratorios pasados, con la particularidad de que se emplearon nuevas funciones y se exploraron nuevas herramientas de diseño que el mismo IDE de Visual Studio nos proporciona.

En esta ocasión, hicimos uso de diferentes algoritmos para reordenar los elementos de un arreglo. En el siguiente documento se mostrará el código según ha sido estructurado, junto con breves explicaciones de cada una de las funciones del mismo.

2. Diseño del *form*

En este espacio se aprovecharon las funcionalidades del IDE de Visual Studio y se exploraron sus múltiples opciones de diseño para crear una ventana visualmente agradable.

The image shows a Windows application window titled "Laboratorio" with a light blue background. The window contains the following elements:

- A label "Ingrese un número" above a yellow text input box.
- A label "Seleccione el método de ordenamiento" above a white dropdown menu with a downward arrow.
- A light blue rectangular area labeled "lbNumero" below the yellow input box.
- A larger light blue rectangular area below the dropdown menu.
- A button labeled "Limpiar" positioned between the two light blue areas.
- A status label "0 elementos ingresados" at the bottom left of the window.

3. Código Fuente

3.1. Declaraciones y métodos iniciales.

Partimos declarando un conjunto de variables globales que estaremos empleando a lo largo del código:

```
public Form1()
{
    InitializeComponent();
}

int N, aux, k, i;
int[] Numeros;
```

La siguiente función nos servirá para imprimir en el cuadrado de texto que llamamos **txtOrden** el orden de los números según el método de reordenamiento que escogimos.

```
void Imprimir ()
{
    for (k = 0; k < N; k++)
    {
        txtOrden.Text = txtOrden.Text + Numeros[k] + " ";
    }

    txtOrden.Text = txtOrden.Text + "\r\n";
}
```

3.2. Métodos de ordenamiento.

En seguida, se diseñan los métodos que llamaremos posteriormente en el código para reordenar los elementos del arreglo:

```
public void BurbujaMenor()
{
    i = 0;
    N = 0;

    foreach(int Elemento in lbNumero.Items)
    {
        Array.Resize(ref Numeros, N + 1);
        Numeros[N] = Elemento;
        N += 1;
    }

    txtOrden.Clear();

    for(i = 1; i < N; i++)
    {
        for (int j = N - 1; j > 0; j--)
        {
            if (Numeros[j - 1] > Numeros[j]){
                aux = Numeros[j - 1];
                Numeros[j - 1] = Numeros[j];
                Numeros[j] = aux;
            }
            Imprimir();
        }
    }
}
```

Método de la burbuja menor

```
public void BurbujaMayor()
{
    N = 0;
    foreach (int Elemento in lbNumero.Items)
    {
        Array.Resize(ref Numeros, N + 1);
        Numeros[N] = Elemento;
        N += 1;
    }
    txtOrden.Clear();

    for (i = N - 1; i > 0; i--)
    {
```

```

        for (int j = 0; j < i; j++)
        {
            if (Numeros[j] > Numeros[j + 1])
            {
                aux = Numeros[j];
                Numeros[j] = Numeros[j + 1];
                Numeros[j + 1] = aux;
            }
            Imprimir();
        }
    }
}

```

Método de la burbuja mayor

```

public void BurbujaConSeñal()
{
    N = 0;
    bool Band;

    foreach (int Elemento in lbNumero.Items)
    {
        Array.Resize(ref Numeros, N + 1);
        Numeros[N] = Elemento;
        N += 1;
    }

    txtOrden.Clear();
    i = 0;
    Band = false;

    while (i < N - 1 && Band == false)
    {
        Band = true;

        for (int j = 0; j < N - 1; j++)
        {
            if (Numeros[j] > Numeros[j + 1])
            {
                aux = Numeros[j];
                Numeros[j] = Numeros[j + 1];
                Numeros[j + 1] = aux;
                Band = false;
            }
        }
        Imprimir();
    }
}

```

```
        i += 1;  
    }  
}
```

Método de la burbuja con señal

3.3. Métodos de interacción con la interfaz.

En la función siguiente, precisamente, en la función de carga del *form* con el que estamos trabajando, se le añaden tres elementos a la caja de opciones que llamamos **cbMetodo**. Estas opciones nos servirán para llamar a las funciones previamente escritas:

```
private void Form1_Load(object sender, EventArgs e)
{
    cbMetodo.Items.Add("Burbuja Menor");
    cbMetodo.Items.Add("Burbuja Mayor");
    cbMetodo.Items.Add("Burbuja con señal");
    cbMetodo.DropDownStyle = ComboBoxStyle.DropDownList;
}
```

En este bloque de código indicamos a qué función deberá de llamar según la opción seleccionada:

```
private void cbMetodo_SelectedIndexChanged(object sender, EventArgs e)
{
    if (cbMetodo.SelectedIndex == 0)
    {
        BurbujaMenor();
    }
    if (cbMetodo.SelectedIndex == 1)
    {
        BurbujaMayor();
    }
    if (cbMetodo.SelectedIndex == 2)
    {
        BurbujaConSeñal();
    }
}
```

En el bloque siguiente, le damos una funcionalidad a la caja de texto que nombramos **txtNumero**. Dicha caja de texto capturará, luego de un enter, el número entero ingresado y lo mostrará en la listbox llamada **lbNumero**. El label que indica la cantidad de elementos ingresados se modificará correspondientemente.

```
private void txtNumero_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == 13)
    {
        int Num = int.Parse(txtNumero.Text);
        lbNumero.Items.Add(Num);
        lblEtiqueta.Text = Convert.ToString(lbNumero.Items.Count + "
elementos ingresados");
        txtNumero.Clear();
    }
}
```

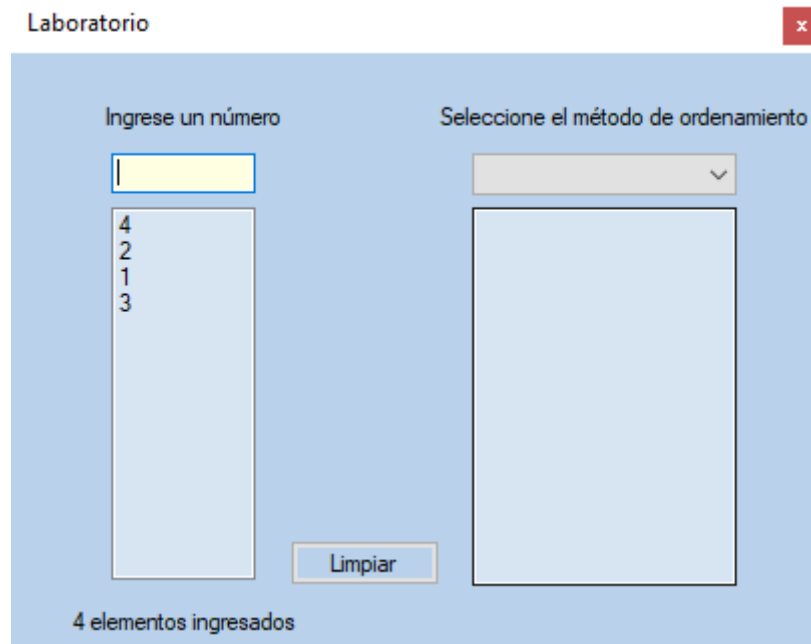
```
        txtNumero.Focus();  
    }  
}
```

Finalmente, el botón **limpiar** nos servirá para eliminar todos los datos en pantalla y borrar del arreglo todos los elementos ingresados:

```
private void btnLimpiar_Click(object sender, EventArgs e)  
{  
    lbNumero.Items.Clear();  
    txtOrden.Text = string.Empty;  
    lblEtiqueta.Text = Convert.ToString(0 + " elementos ingresados")  
;  
}
```

4. Ejecución del programa

El código fue escrutado con detenimiento, con el fin de darnos una idea de cómo funcionaba cada una de sus partes. Luego, y después de habernos cerciorado de que no había ningún error, el resultado fue el siguiente:



Laboratorio

Ingrese un número

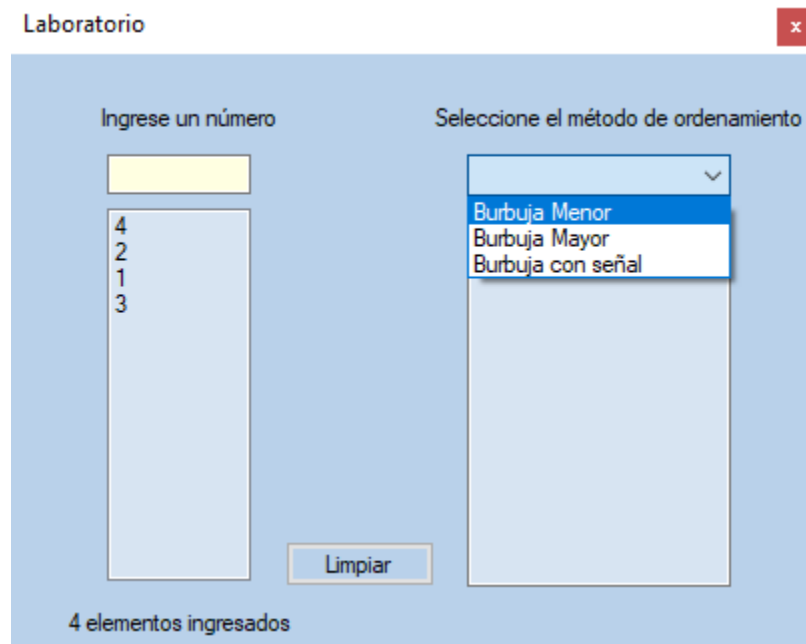
Seleccione el método de ordenamiento

4
2
1
3

Limpiar

4 elementos ingresados

Se ingresaron 4 elementos de forma arbitraria



Laboratorio

Ingrese un número

Seleccione el método de ordenamiento

Burbuja Menor
Burbuja Mayor
Burbuja con señal

4
2
1
3

Limpiar

4 elementos ingresados

Explorando las opciones del comboBox

Laboratorio



Ingrese un número

4
2
1
3

4 elementos ingresados

Seleccione el método de ordenamiento

Burbuja Menor

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

Limpiar

Ordenamiento por el método de burbuja menor

Laboratorio



Ingrese un número

4
2
1
3

4 elementos ingresados

Seleccione el método de ordenamiento

Burbuja Mayor

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

Limpiar

Ordenamiento por el método de burbuja mayor

Laboratorio ✕

Ingrese un número

Seleccione el método de ordenamiento

Burbuja con señal

4
2
1
3

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

Limpiar

4 elementos ingresados

Ordenamiento por el método de burbuja con señal

Laboratorio ✕

Ingrese un número

Seleccione el método de ordenamiento

Burbuja con señal

Limpiar

0 elementos ingresados

Limpiando toda la interfaz

5. Conclusiones

Se obtuvieron resultados satisfactorios con la realización de este laboratorio. Logramos conocer un poco más a fondo el excelente entorno de desarrollo de Visual Studio y empleamos nuevos algoritmos bastante sencillos para reordenar los elementos de un arreglo.

De esta manera, comenzamos a sentar las bases que nos permitirán contar con las herramientas de programas más complejos y completos, con diseños e interfaces mucho más vistosas, sencillas de crear y acordes a los estándares modernos.