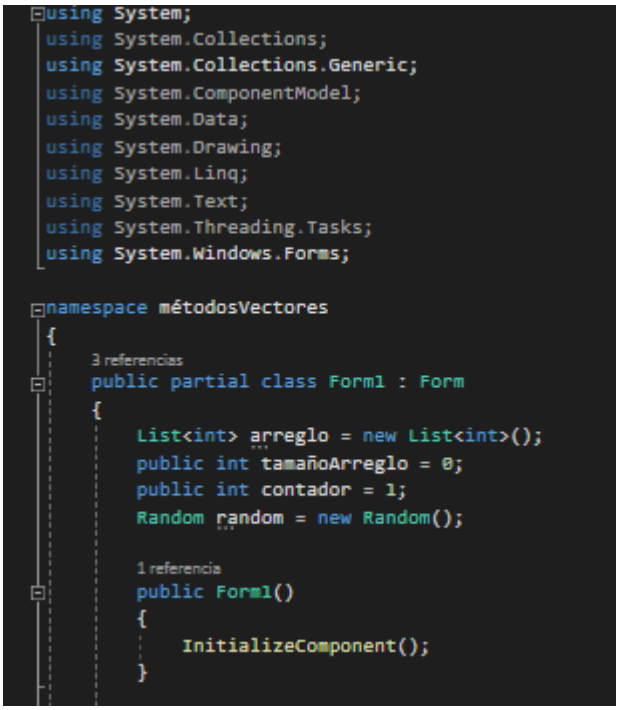
MANUAL TÉCNICO DE LA APLICACIÓN

DANIEL BRAND TABORDA

2020

UNIVERSIDAD DE ANTIOQUIA



```
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace métodosVectores
{
    3 referencias
    public partial class Form1 : Form
    {
        List<int> arreglo = new List<int>();
        public int tamañoArreglo = 0;
        public int contador = 1;
        Random random = new Random();

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

Al iniciar podemos ver las librerías usadas para el programa.

Nombre del aplicativo ´métodosVectores´

Variables globales a las que haremos referencia más adelante.

```
List<int> arreglo = new List<int>();
public int tamañoArreglo = 0;
public int contador = 1;
Random random = new Random();
```

´arreglo´ es un objeto de tipo arrayList en el cual vamos a guardar los valores que ingrese el usuario.

‘tamañoArreglo’ lo definimos para que el usuario establezca el límite de números en el arreglo.

‘contador’ lo usaremos para ingresar agregar los valores manualmente dentro del arreglo

´random´ lo usaremos para generar números aleatorios cuando sea necesario.

```
1 referencia
private void button1_Click(object sender, EventArgs e) //botón para confirmar el tamaño del arreglo
{


    switch(textBox1.Text)
        {
            case "":
                MessageBox.Show("Ingrese un valor","Aletar",MessageBoxButtons.OK,MessageBoxIcon.Warning);
                break;
            default:
                button1.Visible = false;
                tamañoArreglo = Int32.Parse(textBox1.Text);

                var result = MessageBox.Show("Desea llenar su arreglo de forma manual?", "Atención", MessageBoxButtons.YesNo,MessageBoxIcon.Information);
                if (result == DialogResult.No)
                {
                //si la respuesta es no se habilita el botón de llenar aleatorio
                llenarAleatorio.Visible = true;

                }
                else
                {
                    //si la respuesta es Si habilitamos las herramientas para llenar manualmente.
                    completar.Visible = true;
                    textBoxLlenarManual.Visible = true;
                    llenarManual.Visible = true;
                }
            break;
    }



}
```

Si lo que se encuentra en el primer textBox es un valor nulo el programa desplegará un mensaje

Si lo que se encuentra es un valor válido entonces el primer botón se va a desaparecer, la variable global tamaño adquiere el valor del primer textbox

Desplegamos un mensaje y guardamos el valor de la decisión en la variable result, el resto del procedimiento se aprecia en la imagen.

```
private void button2_Click(object sender, EventArgs e) //botón para llenar aleatoriamente el arreglo.
{
    int numeroAleatorio;

    for (int i = 0; i < tamañoArreglo; i++)
    {
        numeroAleatorio = random.Next(0, 100);
        arreglo.Insert(i,numeroAleatorio);
        listBox1.Items.Add("    "+(i+1)+"                "+numeroAleatorio);
    }
    //volvemos invisible el botón llenarAleatorio para evitar inconvenientes con el usuario
    llenarAleatorio.Visible = false;
    indice.Visible = true;
    numero.Visible = true;
    posicion1.Visible = true;
    posicion2.Visible = true;

}
```

En el caso de que el botón llenarAleatorio se active entonces cuando el programa registre el evento click de este botón llenará el arreglo de forma aleatoria y mostrará estos valores en el listbox

Cuando se de click sobre él se volverá invisible y los textbox faltantes serán visibles.

```
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)//controlar no ingresar letras al arreglo
{
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
    {
        e.Handled = true;
    }
}
```

Evento keyPress para evitar ingresar cosas que no sean números enteros a un textBox, este evento se le puso a todos los textbox para evitar inconvenientes con los usuarios (revisar los eventos de los textBox)

```
private void llenarManual_Click(object sender, EventArgs e)//botón para llenar el arreglo de forma manual
{
    int valor = int.Parse(textBox1.Text);
    if (textBoxLlenarManual.Text.Equals(""))
    {
        //si el usuario activa el botón sin llenar el textbox
        var result = MessageBox.Show("Si deja el espacio en blanco se llenará con un 0 ¿Desea dejarlo en blanco?", "Alerta", MessageBoxButtons.YesNo,MessageBoxIcon.Warning);
        if (result == DialogResult.Yes)
        {
            if (contador == 1)
            {
                listBox1.Items.Add("      " + contador + "              " + "0");
                arreglo.Insert(0, 0);
            }
            else
            {
                listBox1.Items.Add("      " + contador + "              " + "0");
                arreglo.Insert(contador - 1, 0);
            }

        }
        else
        {
            //disminuir en uno a contador para que no se alteren las posiciones del vector si se da click en el botón
            contador--;
        }

    }
```

Si se activa la parte de llenar manualmente lo primero que puede pasar es que el usuario de click en el botón agregar sin haber escrito nada dentro del textbox correspondiente, para este caso saldrá el mensaje que muestra la imagen, también si el primer elemento que se va agregar está vacío hubo que controlar esa excepción, si el usuario decide no hacerlo se le resta uno a contador para no se alteren las posiciones (cuando se clickea el botón contador aumenta en +1 por defecto)

```
else
{
    //agregamos elementos al listbox y al arreglo
    listBox1.Items.Add("      " + contador + "              " + textBoxLlenarManual.Text);
    arreglo.Insert(contador - 1, int.Parse(textBoxLlenarManual.Text));
}

if (contador == tamañoArreglo)
{
    llenarManual.Visible = false;
    completar.Visible = false;
    indice.Visible = true;
    numero.Visible = true;
    posicion1.Visible = true;
    posicion2.Visible = true;
}
contador++;
```

Si el usuario ingresa un valor válido insertamos el valor de ese textbox en la posición contador – 1. Luego cuando contador sea igual al tamaño del arreglo desactivamos el botón de llenar manual y el botón completar. Y aumentamos contador en +1.

```
private void button2_Click_1(object sender, EventArgs e) //botón usado para corroborar que los elementos del arreglo coinciden los los del listbox
{
    foreach (var item in arreglo)
    {
        MessageBox.Show("" + item);
    }
}
```

Para corroborar que los elementos del listBox coinciden con los del arreglo debe de cambiar la propiedad visible de este botón a ‘true’.

```
private void button3_Click(object sender, EventArgs e) //botón para cambiar un elemento del arreglo dado un indice y un valor
{
    if(numero.Text.Equals("") || indice.Text.Equals(""))
    {
        MessageBox.Show("Ingrese valores","Alerta",MessageBoxButtons.OK,MessageBoxIcon.Warning);
    }
    else
```

Primero controlamos que lo que esté dentro de índice y numero no sea nulo, si este es el caso saldrá el mensaje mostrado.

```
else
{
    int indiceCambiar = int.Parse(indice.Text);
    int numeroCambiar = int.Parse(numero.Text);
    int i = 1;

    if (indiceCambiar > tamañoArreglo)
    {
        MessageBox.Show("Êste número no se encuentra en el arreglo","Alerta", MessageBoxButtons.OK);
    }
    else
    {
        arreglo.RemoveAt(indiceCambiar - 1);//eliminamos el elemento que se encuentra en el índice dado
        var result = MessageBox.Show("Desea realizar el cambio? Si oprime 'Si' el cambio se realizará, si oprime 'No' se le asignará un valor aleatorio", "Atencion", MessageBoxButtons.YesNoCancel);

        if (result == DialogResult.Yes)
        {
            listBox1.Items.Clear();
            listBox1.Items.Add("Posición     Valor");
            arreglo.Insert(indiceCambiar - 1, numeroCambiar);
            foreach (var item in arreglo)//reescribimos los elementos del listbox con el elemento nuevo del arreglo
            {
                listBox1.Items.Add("    " + i + "              " + item);
                i++;
            }
        }
        else if (result == DialogResult.No)
        {
            listBox1.Items.Clear();
            listBox1.Items.Add("Posición     Valor");
            int numeroAleatorio = random.Next(0, 100); // si la respuesta es no reescribimos el arreglo pero con un numero aleatorio en el índice dado
            arreglo.Insert(indiceCambiar - 1, numeroAleatorio);
            foreach (var item in arreglo)
            {
                listBox1.Items.Add("    " + i + "              " + item);
                i++;
            }
        }
    }
}
```

Convertimos los valores de índice y de numero a enteros y declaramos la variable i en 1. Si todo sale bien se elimina el número en el índice ingresado con el método arreglo.removeAt(index) y determinamos qué es lo que el usuario va a hacer mediante el messageBox, el resto de la explicación se ve en la imagen.

```
private void intercambioPosicion(int posicion1, int posicion2)
{
    int auxiliar;
    auxiliar = arreglo[posicion1];
    arreglo[posicion1] = arreglo[posicion2];
    arreglo[posicion2] = auxiliar;
}
```

Método que intercambia la posición de dos elementos del arreglo.

```
private void button4_Click(object sender, EventArgs e) //terminar de llenar el vector con ceros
{
    var result = MessageBox.Show("Completar el resto del vector con ¨0¨?", "Alerta", MessageBoxButtons.YesNo);
    if (result == DialogResult.Yes)
    {
        for (int i = contador - 1; i < tamañoArreglo; i++)
        {
            arreglo.Insert(i, 0);
            listBox1.Items.Add("      " + (i + 1) + "              " + 0);
        }
        //habilitamos los nuevos pasos cuando el arreglo esté lleno y deshabilitamos los que nos puedan causar problemas
        completar.Visible = false;
        llenarManual.Visible = false;
        indice.Visible = true;
        numero.Visible = true;
        textBoxLlenarManual.Visible = false;
        posicion1.Visible = true;
        posicion2.Visible = true;
    }

}
```

En este evento del botón completar apreciamos la importancia de la variable contador para poder llenar el arreglo desde la última posición ingresada y poder comenzar a generar ceros desde ese índice en adelante.

```
private void button4_Click_1(object sender, EventArgs e) // revertir el orden del arreglo
{

    int i = 1;
    arreglo.Reverse();
    listBox1.Items.Clear();
    listBox1.Items.Add("Posición     Valor");

    foreach (var item in arreglo)
    {
        listBox1.Items.Add("    " + i + "              " + item);
        i++;
    }
}
```

Botón para revertir el orden del arreglo. Se utilizó el método .Reverse() y se reescribió todo el listBox luego para cada elemento en arreglo se escribe en el listBox en con el nuevo orden.

```
private void button2_Click_2(object sender, EventArgs e) //botón para reiniciar el proceso
{
    //reestablecemos los valores iniciales
    button1.Visible = true;
    textBoxLlenarManual.Visible = false;
    indice.Visible = false;
    numero.Visible = false;
    llenarAleatorio.Visible = false;
    posicion1.Visible = false;
    posicion2.Visible = false;
    listBox1.Items.Clear();
    arreglo.Clear();
    indice.Text = "";
    numero.Text = "";
    posicion1.Text = "";
    posicion2.Text = "";
    listBox1.Items.Add("Posición     Valor");
}
```

El botón ´Limpiar´ cumple la función de reiniciar todo el proceso limpiando el listBox, el arreglo y reestableciendo la configuración inicial.

```
private void button4_Click_2(object sender, EventArgs e) // botón para intercambiar dos numeros
{
    if (posicion1.Text.Equals("") || posicion2.Text.Equals(""))
    {
        MessageBox.Show("Ingrese Valores","Alerta",MessageBoxButtons.OK);
    }
    else if (tamañoArreglo == 1)
    {
        MessageBox.Show("Deben de haber más valores para poder realizar este intercambio","Alerta",MessageBoxButtons.OK);
        posicion1.Visible = false;
        posicion2.Visible = false;
    }

    else
    {
        int posicioncambiar1 = int.Parse(posicion1.Text) - 1;
        int posicioncambiar2 = int.Parse(posicion2.Text) - 1;

        if (posicioncambiar1 + 1 > tamañoArreglo || posicioncambiar2 + 1 > tamañoArreglo)
        {
            MessageBox.Show("Este número no se encuentra dentro del arreglo.","Alerta",MessageBoxButtons.OK);
        }
        else
        {
            int i = 1;
            intercambioPosicion(posicioncambiar1, posicioncambiar2);
            listBox1.Items.Clear();
            listBox1.Items.Add("Posición     Valor");
            foreach (var item in arreglo)
            {
                listBox1.Items.Add("    " + i + "              " + item);
                i++;
            }
        }

    }
}
```

El botón ´Intercambiar´ lo primero que hace es ver si la posicion1 y la posicion2 no están vacíos o si el tamaño del vector no es 1 (lo que haría imposible realizar un intercambio). Si todo va bien hasta este punto el programa primero revisa que la posición 1 o la posición 2 (o ambas) no sean mayores al tamaño del arreglo (en este caso los índices pedidos no existen dentro del arreglo) y si todo va bien hasta este punto se invoca el método intercambioPosicion y se reescribe todo el listBox.