**Manual técnico: Creador de Vectores**

**Lenguaje de programación:** C sharp ó C#

**Entorno de desarrollo:** Visual Studio 2019

**Objetivos de la aplicación**

**Objetivo general:**

Crear un vector a partir de un entrada generada por el usuario y desarrollar distintas operaciones con este vector.

**Objetivos especificos:**

* Capturar dato introducido por el usuario para generar vector = n
* Crear vector de tamaño n
* Cambiar un valor del vector creado
* Invertir el vector
* Intercambiar dos posiciones de un vector

using System;

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;

using Microsoft.VisualBasic; //Libreria utilizada para hacer uso de los InputBox

Usa la librería Microsoft.VisualBasic para utilizar los InputBox, que serán de vital importancia en la creación de la aplicación

namespace Practica\_4\_Vectores

La variable n se usa de modo general para utilizar los métodos de la clase. Sea crea una variable de tipo vector entero y una variable de tipo Random, las cuales también será generales para usar en distintos botones y métodos.

Los métodos usados en el TextBox llamado VectorSizw, con los eventos Click, Leave y Keypress permiten generar un efecto de valor predefinido en el TextBox y valida si el valor introducido es numérico, ya que se necesita que se cumpla está condición para poder trabajar.

//Método que permite establecer una condición que genera un valor por defecto en el TextBox y con el evento click lo hace desaparecer para introducir valores

private void VectorSize\_Click(object sender, EventArgs e)

{

if (VectorSize.Text == "0")

{

VectorSize.Text = "";

}

}

//Método que permite establecer una condición que genera un valor por defecto en el TextBox y con el evento leave lo hace aparecer de nuevo en en TextBox

private void VectorSize\_Leave(object sender, EventArgs e)

{

if (VectorSize.Text == "")

{

VectorSize.Text = "0";

}

}

//Método para validar si lo que se introduce en el TextBox es un valor que se permite para los objetivos de la aplicación

private void VectorSize\_KeyPress(object sender, KeyPressEventArgs e)

{

Validar.SoloNumeros(e); //Usa el método SoloNumeros de la clase Validar

}

El primer botón permite crear el vector con las condiciones subsecuentes, en caso de que se desee llenar o no el vector o si en algún punto del proceso de creación no se desea seguir llenando el vector.

//Este botón permite la creación del vector, siguiente unos parámetros determinados

private void button1\_Click(object sender, EventArgs e)

{

// Se declara una variable booleana que permitirá validar si los valores de los InputBox son númericos

// Se convierte lo tecleado dentro del TextBox en un valor entero de 32 bits

bool test;

n = Convert.ToInt32(VectorSize.Text);

// n representa el tamaño del vector, por lo que

// Se valida que el vector tenga un tamaño diferente de cero. No se validan números negativos ni decimales porque el TextBox tiene restricciones para estos

if (n == 0)

{

MessageBox.Show("No se puede crear un vector de este tamaño");

}

// Cuando se valido que el vector tiene un tamaño válido se ejecuta el siguiente código

else

{

// Permite que los botones se activen, ya que estos están inactivos para evitar errores con el evento Enable = false

button2.Enabled = true;

button3.Enabled = true;

button4.Enabled = true;

// Se declara la variable Num que captura los valores de los InputBox

// Se iniciliza el vector con un tamaño de n que fue introducido por el usuario en el TextBox

// Se declara la variable contador que servirá de interruptor de ciclos

string Num = "";

Vector = new int[n];

int cont = 1;

// Se genera una ventana que tendrá los botoes "Si" y "No"

MessageBoxButtons botones = MessageBoxButtons.YesNo;

DialogResult dr = MessageBox.Show("¿Desea ingresar los valores del vector?", "Valores del vector", botones, MessageBoxIcon.Question);

// Verifica si el usuario dió click en "Si" e ingresa para introducir los valores

if (dr == DialogResult.Yes)

{

//Al ingresar se muestra un mensaje de confirmación que solo tiene el botón "Ok" para entrar a ingresar los valores

MessageBoxButtons botones1 = MessageBoxButtons.OK;

DialogResult dr1 = MessageBox.Show("Ingrese los valores", "Valores del vector", botones1, MessageBoxIcon.Question);

// Se genera un ciclo que ejecuta otro ciclo mientras se cumpla que el usuario no di click en "Cancelar" ni el contador se hace n, que es dónde se ha terminado de introducir los valores del vector

do

{

//ciclo para rellenar el vector en todas sus posiciones

for (int i = 0; i < n; i++)

{

//Se toma el valor del InputBox y se guarda en la variable Num, a la cuál se verifica que si sea un valor válido (númerico) con el método isNumeric()

Num = Interaction.InputBox("Ingrese el número que va en la posición " + (i + 1), "Entrada de valores", "0");

test = IsNumeric(Num);

// Se verifica de nuevo que no se de click en "Cancelar", en caso de que se de en cancela el valor del ciclo se iguala a n para salir de este.

if ((String)Num == "")

{

i = n;

}

//Si se cumplen las condiciones y el valor introducido es válido, se introduce este valor enb el vector una vez se convirtió en un valor de tipo entero

else if (test == true)

{

Vector[i] = Convert.ToInt32(Num);

}

// Esta parte es para cuando no se cumple ninguna de las condiciones anteriores

else

{

//Se ejecuta este cpodigo mientras que el usuario o introduzca un valor válido, lo que lo obliga a hacer para poder continuar

do

{

//Se repite el proceso anterior en dónde se pide el valor y se comprueba si este valor es válido para introducirlo en el vector

MessageBox.Show("Ingrese un valor númerico");

Num = Interaction.InputBox("Ingrese el número que va en la posición " + (i + 1), "Entrada de valores", "0");

test = IsNumeric(Num);

if (test == true)

{

Vector[i] = Convert.ToInt32(Num);

}

} while (test == false);

}

// Aumento a la variable contador que permite que si el usuario da click en cancelar, pero quiere volver a seguir introduciendo valores se comience en el valor que iba

cont = cont + 1;

}

} while ((String)Num != "" && cont<=n);

// Verifica si se da click en el botón cancelar y se envía mensaje para que el usuario decida si seguir o no llenado el vector

if ((String)Num == "")

{

//Se genera un mensaje con los botones "Si" y "No", para verificar si se desea o no seguir llenando el vector

MessageBoxButtons botones2 = MessageBoxButtons.YesNo;

DialogResult dt = MessageBox.Show("Le faltan posiciones por llenar, ¿Desea seguir llenándolas?", "Valores del vector", botones2, MessageBoxIcon.Question);

//En caso de que se desee seguir llenando el vector se entre en la condición que se muestra (Da click en "Si")

if (dt == DialogResult.Yes)

{

// Ciclo para seguir llenando el vector desde la posición en la que iba, para lo que se usa la variable contador

for (int k=cont-1; k<=n; k++)

{

//Se vuelve asignar un valor de un InputBox a la variable Num y se comprueba que no se de clic en "Cancelar"

Num = Interaction.InputBox("Ingrese el número que va en la posición " + k , "Entrada de valores", "0");

if((String)Num == "")

{

// Si se da click en "Cancelar" Se vuelve a emitir el mensaje que dice que le faltan posicione a llenar y se resta una unidad a k para que quede en la posición que iba

DialogResult dx = MessageBox.Show("Le faltan posiciones por llenar, ¿Desea seguir llenándolas?", "Valores del vector", botones2, MessageBoxIcon.Question);

k--;

if(dx == DialogResult.No)

{

k = n+1;

}

}

// En caso de que se de click en aceptar, el vector será llenado con el valor de la variable Num

else

{

Vector[k-1] = Convert.ToInt32(Num);

}

}

}

//En caso de que no se desee seguir llenando el vector se entra en la condición que se muestra (Da click en "No")

else

{

//Ciclo para llenar el vector con ceros en todas las posiciones restantes

for (int j = cont; j < n; j++)

{

Vector[j] = 0;

}

}

}

}

//Condición que se ejecuta cuando el usuario da click en "No" en la ventana que pregunta si desea llenar el vector

else

{

//Ciclo para llenar el vector con valores aleatorios entre -100 y 100

for (int i = 0; i < n; i++)

{

Vector[i] = aleatorio.Next(-101,101);

}

MessageBox.Show("El vector se llenará de forma aleatoria con valores enteros entre -100 y 100");

}

//Método que permite dibujar el vector en una tabla

mostrar\_DataGridView(Vector, n);

}

}

El segundo botón permite cambiar un valor del vector creado, haciendo las verificaciones pertinentes para que el valor introducido sea válido

private void button2\_Click(object sender, EventArgs e)

{

// El botón 2 permite cambiar un valor del vector, se declaran variables que capturan la posición, el valor a agregar y booleanos que verifican las entradas

int PosA;

int Valor;

bool test;

bool test1;

// Captura el valor del InputBox y pregunta con el método IsNumeric() si el valor representa un valor númerico

string m = Interaction.InputBox("Ingrese la posición en dónde desea cambiar un valor", "Cambiar valor", "");

test = IsNumeric(m);

//Si el valor es númerico se entra de acuerdo a la siguiente condición

if (test==true)

{

//Una vez se sabe que m es númerica, entonces se convierte esta variable a una de tipo entero

PosA = Convert.ToInt32(m);

//Se verifica que esa variable entera se encuentre entre los valores del vector creado

if (PosA<=n && PosA>0)

{

// Se genera una venta que pregunta si el valor que desea ingresar sea aleatorio, tiene las posibilidades "Sí" y "No"

MessageBoxButtons botones = MessageBoxButtons.YesNo;

DialogResult dr = MessageBox.Show("¿Desea ingresar un valor aleatorio?", "Valor a cambiar", botones, MessageBoxIcon.Question);

// Si el usuario da click en "Si" entrará a la condición siguiente y se llenará la posición que el usuario ha indicado con un valor aleatorio

if (dr == DialogResult.Yes)

{

// Se llena el vector en la posición PosA-1. El valor aleatorio se hace con método Next de la clase Random con valores entre -100 y 100

Vector[PosA - 1] = aleatorio.Next(-101, 101);

mostrar\_DataGridView(Vector, n);// Se pinta el vector en el DataGridView

}

//En caso que el usuario de click en no se debe generar un InputBox que le permita ingresar el valor que este desee

else

{

// Se crea el InputBox que captura el dato ingresado por el usuario y se usa la variable booleano con el método IsNumeric() para saber si el valor es numérico

string l = Interaction.InputBox("Ingrese el valor a agregar", "Cambiar valor", "");

test1 = IsNumeric(l);

// Si el valor es numérico, test1 se hará true e ingresará en la siguiente condición

if (test1 == true)

{

//Dado a que el valor ingresado si es numérico, se convierte este a una variable de tipo entero, se almacena en el vector en la posición PosA - 1 y se pinta en el DataGridView

Valor = Convert.ToInt32(l);

Vector[PosA - 1] = Valor;

mostrar\_DataGridView(Vector, n);

}

//Si no se tiene un valor numérico, se informa al usuario que debe ingresar valores válidos

else

{

MessageBox.Show("Ingrese valores válidos (numéricos enteros)");

}

}

}

// Si la posición que introduce el usuario no está dentro del rango del vector se muestra el mensaje que se muestra

else

{

MessageBox.Show("EL vector solo tiene " + n + " posiciones, verifique que ingreso una posición válida");

}

}

//Si es valor que introduce el usuario no es válido se muestra el siguiente mensaje

else

{

MessageBox.Show("Ingrese valores válidos");

}

}

// El botón de invertir usa el método InvertirORden() y lo pinta en el DataGridView

El botón 3 permite invertir el orden del vector, de la siguiente manera: La posición 0 pasará a la posición n, la posición 1 pasará a la posición n-1 y así sucesivamente hasta n-i sea igual a posición i

private void button3\_Click(object sender, EventArgs e)

{

InvertirOrden(Vector,n);

mostrar\_DataGridView(Vector, n);

}

El botón 4 permite intercambiar dos valores de dos posiciones diferentes en el vector

//Botón de intercambiar

private void button4\_Click(object sender, EventArgs e)

{

//Se declaran variables de tipo string, boolano y enteras.

string r;

string s;

bool test;

bool test1;

int PosA;

int PosB;

// Las variables de tipo string almacenan los datos que se introducen en los InputBox y representarán las posiciones a intercambiar

r = Interaction.InputBox("Ingrese la posición inicial", "Cambio posición", "");

s = Interaction.InputBox("Ingrese la posición a cambiar", "Cambio posición", "");

//Se comprueba que las variables introducidas sean numéricas con el método IsNumeric()

test = IsNumeric(r);

test1 = IsNumeric(s);

//Si ambas variables son numéricas se entra en la siguiente condición

if(test == true && test1 == true)

{

// Se convierten las variables numéricas que son tipo string a tipo int

PosA = Convert.ToInt32(r);

PosB = Convert.ToInt32(s);

// con las variables tipo int se comprueba que estás estén dentro del rango del vector

if (PosA<=n && PosA>0 && PosB<=n && PosB>0)

{

//Se usa el método intercambiarPosicion() cambiar los valores que se encuentran en las posiciones introducidas y se pinta el DataGridView

intercambiarPosicion(Vector, PosA, PosB);

mostrar\_DataGridView(Vector, n);

}

// Si los valores no son positivos o no están dentro del rango del vector se muestra el siguiente mensaje

else

{

MessageBox.Show("Solo se admiten valores positivos y recuerde que la matriz tiene " + n + " posiciones");

}

}

// Si los valores no son numéricos se muestra el siguiente mensaje

else

{

MessageBox.Show("Introduzca valores numéricos");

}

}

Los métodos utilizados son:

* IntercambiarPosición: Sirve para cambiar dos posiciones dentro del vector
* InvertirOrden: Invierte el orden de los valores del vector
* InNumeric: Permite indagar si un string representa un valor numérico
* InsertaValor: Permite agregar un valor dentro del vector
* Mostrar\_DataGridView: Llena una matriz y la pinta dentro de un DataGridView

//Métodos

public int[] intercambiarPosicion(int[] Vector, int Pos1, int Pos2)

{

// Este método toma un vector y dos posiciones y los intercambiar usando una variable auxiliar llamada dato

int dato;

dato = Vector[Pos1-1];

Vector[Pos1-1] = Vector[Pos2-1];

Vector[Pos2-1] = dato;

return(Vector);

}

public int[] InvertirOrden(int[] v, int n)

{

// Este método invierte el orden del vector, pasando la primera posición a la última, la segunda a la penúltima y así sucesivamente; usando la variable auxiliar dato

int j = 0;

for (int i = n-1; i > j; i--)

{

int dato;

dato = v[i];

v[i] = v[j];

v[j] = dato;

j++;

}

return(v); // retorna el vector invertido

}

public bool IsNumeric(string a)

{

// Verifica si un valor de tipo string es numérico y lo puede convertir, devuelve un booleano

int number1 = 0;

bool canConvert = int.TryParse(a, out number1);

if (canConvert == true)

{

}

else

{

}

return (canConvert);

}

public int[] InsertarValor(int[] Vector, int PosA, int Value)

{

// método que con una posición y un valor lo reemplaza en el vector y retorna el vector

Vector[PosA] = Value;

return (Vector);

}

private void button5\_Click(object sender, EventArgs e)

{

// Sirve para salir de la aplicación

Application.Exit();

}

public void mostrar\_DataGridView(int[] Vector, int n)

{

// Usa las variables i y j para llenar una matriz a partir de un vector

int i, j;

// Se inicializa la matriz, que tendrá 2 filas y n columnas

int[,] matriz = new int[2, n];

// Ciclo for para recorrer la matriz por filas y adentro por columnas

for (i = 0; i < 2; i++)

{

// condición para llenar la primera fila

if (i == 0)

{

//Ciclo for para llenar la primera fila columna por columna

for (j = 0; j < n; j++)

{

matriz[i, j] = j + 1;

}

}

//Condición para llenar la segunda fila

else

{

//Ciclo for para llenar la segunda fila columna por columna con los valores del vector

for (int k = 0; k < n; k++)

{

matriz[i, k] = Vector[k];

}

}

}

// Sirve para pintar la DataGridView

dataGridView1.ColumnCount = n;

dataGridView1.RowCount = 2;

//Ciclo para llenar los campos del DataGridView

for(int l=0; l < 2; l++)

{

//Ciclo que llenar filas y columnas del DataGridView

for(int m=0; m < n; m++)

{

dataGridView1.Rows[l].Cells[m].Value = matriz[l,m];

}

}

}

}

}