1. Trình bày lớp và đối tượng. Ví dụ
2. Các thành phần static. Ví d
3. Construtor. Ví dụ
4. Properties. Ví dụ

Often in the tasks is necessary to enter a number or other data in a two-dimensional array (matrix) and be able to operate them.

https://www.bestprog.net/en/2016/04/29/011-c-an-example-of-creating-of-two-dimensional-matrix-on-the-form-the-analogue-of-tstringgrid-component-in-delphi/#contents

Contents:

Task

Create a program that carries out the product of two matrices of dimension *n*. Matrix must be entered from the keyboard in a separate form and saved in the internal data structures. The user has the ability to see the resulting matrix.

Also, it is possible to save the result matrix in the text file “Res_Matrix.txt”.

⇑

Instructions

1. Run Microsoft Visual Studio. Creating a project

A detailed example of running Microsoft Visual Studio and creating an application using the Windows Forms Application template is described in the topic:

- **Windows Forms type Application Development in Microsoft Visual Studio**

Save the project under any name.

⇑

2. Creating the main form Form1

Create the form as shown in Figure 1.

Place on the form the controls of following types:

- four controls of type Button. Automatically, four objects (variables) with names “button1”, “button2”, “button3”, “button4” will be created;
- three controls of type “Label”, which are named as “label1”, “label2”, “label3”;
- control of TextBox type, which is named “textBox1”.

You need to form the properties of controls of types “Button” and “Label”:

- in the object button1 property Text = “Input of matrix 1 …”;
- in the object button2 property Text = “Input of matrix 2 …”;
- in the object button3 property Text = “Result …”;
- in the object button4 property Text = “Save to file “Res_Matr.txt””;
- in the control label1 property Text = “n = “.

To set up the view and behavior of form you need to do following actions:

- set the title of form. To do this property Text = “The product of matrices”;
- property StartPosition = “CenterScreen” (the form is placed to the center of screen);

- property MaximizeBox = "false" (hide the button of maximize of form).

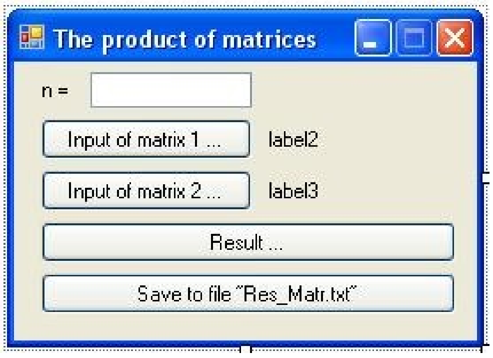

Fig. 1. The form of application

⇑

3. Developing the secondary form Form2

In the secondary form "Form2", will be inputted data into the matrices and outputted the results.

An example of creating a new form in MS Visual Studio – C# is described **here**.

Add the new form to the application using the command

```
Project -> Add Windows Form …
```

In the opened window select "Windows Form". The name of file leave as proposed "Form2.cs".

Place on the form, in any position, the control of "Button" type (Figure 2). As a result, the new object named "button1", will be given.

In the control "button1" you need to change the following properties:

- property Text = "OK";

- property DialogResult = "OK" (Figure 3). It means, when user clicks on the button1, the window will be closed with returning code "OK";
- property Modifiers = "Public". It means, that button "button1" will be visible from other modules (from form Form1).
- Set up the properties of form Form2:
- property Text = "Input of matrix";
- property StartPosition = "CenterScreen" (the form is placed on the center screen);
- property MaximizeBox = "false" (hide the maximize button).

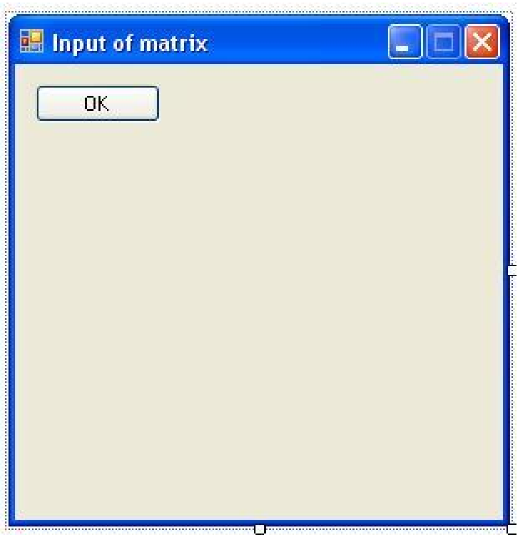

Fig. 2. The form "Form2" after setting

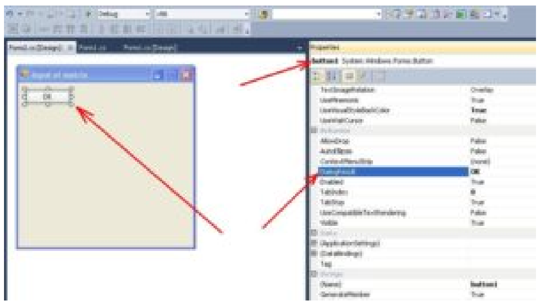

Fig. 3. The property "DialogResult" of control "button1" at the form "Form2"

⇑

4. Entering the the internal variables

Next step – entering the internal variables into the text of module "Form1.cs".

To do this, you need to activate module "Form1.cs".

In the text of module "Form1.cs" you need to add the following code:

```
...

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        const int MaxN = 10; // the maximum allowable dimension of the matrix
        int n = 3; // The current dimension of the matrix
        TextBox[,] MatrText = null; // The matrix of TextBox type elements
        double[,] Matr1 = new double[MaxN, MaxN]; // The matrix 1 of floating point numbers
        double[,] Matr2 = new double[MaxN, MaxN]; // The matrix 1 of floating point numbers
        double[,] Matr3 = new double[MaxN, MaxN]; // The matrix of results
        bool f1; // flag, which indicates about that the data were entered into the matrix
Matr1

        bool f2; // flag, which indicates about that the data were entered into the matrix
Matr1

        int dx = 40, dy = 20; // width and height of cells in MatrText [,]

        Form2 form2 = null;   // an instance (object) of the class form "Form2"

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

Let’s explain some values of variables:

- MaxN – the maximum allowable dimension of the matrix;
- n – dimension of the matrix, which user types on keyboard into control textBox1;
- MatrText – two-dimensional matrix of controls TextBox type. In this matrix is entered the cells of matrix as strings. The data entering will be formed in the form “Form2”.
- Matr1, Matr2 – matrices of elements of “double” type. Data will be copied from MatrText into Matr1 and Matr2;
- Matr3 – the resulting matrix, which is equal to product of matrices Matr1 and Matr2;
- f1, f2 – variables that determine whether the data has been entered, respectively, in Matr1 and Matr2 matrix;
- dx, dy – the dimensions of of one cell of type TextBox in the MatrText matrix;
- form2 – object of class of form “Form2”, using which we will have access to this form.

⇑

5. Programming the event Load of form “Form1”

Process of programming of any event in Microsoft Visual Studio – C# is described **here** in details.

The code listing of event handler Load of form “Form1” is following:

```
private void Form1_Load(object sender, EventArgs e)
{
  // I. Initializing of controls and internal variables
  textBox1.Text = "";
  f1 = f2 = false; // matrices are not yet filled
  label2.Text = "false";
  label3.Text = "false";

  // II. Memory allocation and configure MatrText
  int i, j;
  // 1. Memory allocation for Form2
  form2 = new Form2();

  // 2. Memory allocation for the whole matrix (not for cells)
  MatrText = new TextBox[MaxN, MaxN];

  // 3. Memory allocation for each cell of the matrix and its setting
  for (i = 0; i < MaxN; i++)
    for (j = 0; j < MaxN; j++)
    {
      // 3.1. Allocate memory
      MatrText[i, j] = new TextBox();

      // 3.2. Set the value to zero
      MatrText[i, j].Text = "0";

      // 3.3. Set the position of cell in the Form2
      MatrText[i, j].Location = new System.Drawing.Point(10 + i * dx, 10 + j * dy);

      // 3.4. Set the size of cell
      MatrText[i, j].Size = new System.Drawing.Size(dx, dy);

      // 3.5. Hide the cell
      MatrText[i, j].Visible = false;
      // 3.6. Add MatrText[i,j] into the form2
      form2.Controls.Add(MatrText[i, j]);
    }
}
```

Let's explain some of the code snippet in method Form1_Load().

The event "Load" is generated (called) when form is loading. Since there Form1 is the main form of the application, the "Load" event of "Form1" will be called immediately after the application starts to run. So, here it is expedient to introduce the initial initialization of global controls and internal variables of the program. These controls can be called from other methods of the class.

In the event handler Form1_Load() the memory is allocated for two-dimensional matrix MatrText of strings only one time. This memory will be automatically freed upon completion of the application.

The memory is allocated in two stages:

- for the whole matrix MatrText as two-dimensional array;
- for every element of matrix, which is the object of type "TextBox".

After allocating memory, for any object is carried out the setting of main internal properties (position, size, text and visibility).

Also, every cell, which is created, is added (placed) on the form "Form2" using method Add() from class "Controls". Every new cell can be added on the any other form of application.

⥣

6. The development of an additional method of resetting the data of matrix "MatrText"

In order to many times not use the code of resetting the matrix, you need to create own method (for example Clear_MatrText()), that realizes this code.

Listing of method Clear_MatrText() is following:

```
private void Clear_MatrText()
{
  // Setting the cells of MatrText to zero
  for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
      MatrText[i, j].Text = "0";
}
```

⥣

7. Programming the event of clicking on the button1 ("Input of matrix 1 …")

When button1 is clicked must be called the window of inputting a new matrix. The matrix size depends on the value of n.

Listing of the event handler of clicking on the button1 is following:

```
private void button1_Click(object sender, EventArgs e)
{
  // 1. Reading of the matrix dimension
  if (textBox1.Text == "") return;
  n = int.Parse(textBox1.Text);

  // 2. Zeroing of cell MatrText
  Clear_MatrText();
  // 3. Setting the properties of the matrix cells
  //    with binding to the value of n and the form Form2
  for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
    {
      // 3.1. Tab order
      MatrText[i, j].TabIndex = i * n + j + 1;

      // 3.2. Set the cell as visible
      MatrText[i, j].Visible = true;
    }

  // 4. Correcting of form size
  form2.Width = 10 + n * dx + 20;
  form2.Height = 10 + n * dy + form2.button1.Height + 50 ;

  // 5. Correcting of the position and size of the button on the Form2
  form2.button1.Left = 10;
  form2.button1.Top = 10 + n * dy + 10;
  form2.button1.Width = form2.Width - 30;

  // 6. Calling the form Form2
  if (form2.ShowDialog() == DialogResult.OK)
  {
    // 7. Moving lines from the Form2 form into the matrix Matr1
    for (int i = 0; i < n; i++)
      for (int j = 0; j < n; j++)
        if (MatrText[i, j].Text != "")
          Matr1[i, j] = Double.Parse(MatrText[i, j].Text);
        else
          Matr1[i, j] = 0;

        // 8. Data were entered into matrix
        f1 = true;
        label2.Text = "true";
  }
}
```

In the listing above, the value of n is read. After that, is carry out the setting of cells of matrix MatrText.

Based on the inputted value of n are formed the sizes of form "form2" and position of button "button1".

If, into the form "Form2", user is pressed on the button "OK" (button2) then the rows from MatrText are moved into the two-dimensional matrix "Matr1" of floating point numbers. Converting from string to the corresponding real number is performed by the method Paste() from the class Double.

Also, is formed the variable f1, which points that data were inputted into matrix "Matr1".

⇑

8. Programming of event of clicking on the button “button2” (“Input of matrix 2…”)

Code listing of event handler of clicking on the button2 is similar to the listing of event handler of clicking on the button1. It differs only in steps 7-8. In this section formed Matr2 matrix and variable f2.

```
private void button2_Click(object sender, EventArgs e)
{
  // 1. Reading of the matrix dimension
  if (textBox1.Text == "") return;
  n = int.Parse(textBox1.Text);

  // 2. Zeroing of cell MatrText
  Clear_MatrText();

  // 3. Setting the properties of the matrix cells
  //    with binding to the value of n and the form Form2
  for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
    {
      // 3.1. Tab order
      MatrText[i, j].TabIndex = i * n + j + 1;

      // 3.2. Set the cell as visible
      MatrText[i, j].Visible = true;
    }

  // 4. Correcting of form size
  form2.Width = 10 + n * dx + 20;
  form2.Height = 10 + n * dy + form2.button1.Height + 50;

  // 5. Correcting of the position and size of the button on the Form2
  form2.button1.Left = 10;
  form2.button1.Top = 10 + n * dy + 10;
  form2.button1.Width = form2.Width - 30;

  // 6. Calling the form Form2
  if (form2.ShowDialog() == DialogResult.OK)
  {
    // 7. Moving lines from the Form2 form into the matrix Matr1
    for (int i = 0; i < n; i++)
      for (int j = 0; j < n; j++)
        Matr2[i, j] = Double.Parse(MatrText[i, j].Text);

    // 8. Matrix Matr2 is formed
    f2 = true;
    label3.Text = "true";
  }
}
```

⇑

9. Programming of the leaving of input focus in the control textBox1

In the application may be a situation when the user changes n to a new value. In this case, the flags f1 and f2 must be set to the new values. Also, the size of matrix MatrText must be changed.

You can control the changing of value n using the event "Leave" of control textBox1. The event "Leave" is generated in time when control "textBox1" leaves the input focus (Figure 4).

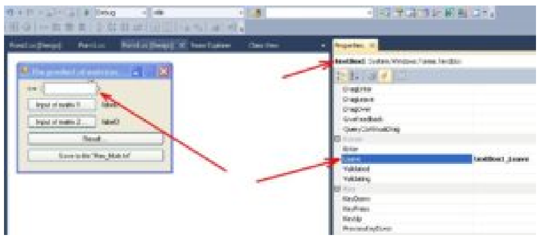

Fig. 4. The event Leave of the control textBox1

The code listing of event handler is following:

```
private void textBox1_Leave(object sender, EventArgs e)
{
  int nn;
  nn = Int16.Parse(textBox1.Text);
  if (nn != n)
  {
    f1 = f2 = false;
    label2.Text = "false";
    label3.Text = "false";
  }
}
```

⇑

10. Programming of the event of clicking on the button3 ("Result ...")

The output of result will be realized in the same form, in which were entered the matrices Matr1 and Matr2.

First of all, the product of these matrices will be formed in the matrix Matr3. After that, the value from Matr3 is moved in "MatrText" and is displayed on the form Form2.

The listing of event handler is following:

```
private void button3_Click(object sender, EventArgs e)
{
  // 1. Checking, were inputted data in the both matrices?
  if (!((f1 == true) && (f2 == true))) return;

  // 2. Calculating of the product of matrices. Result is in Matr3
  for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
    {
      Matr3[j,i] = 0;
      for (int k = 0; k < n; k++)
        Matr3[j, i] = Matr3[j, i] + Matr1[k, i] * Matr2[j, k];
    }

  // 3. Inputting data into MatrText
  for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
    {
      // 3.1. Tab order
      MatrText[i, j].TabIndex = i * n + j + 1;

      // 3.2. Converting the number to a string
      MatrText[i, j].Text = Matr3[i, j].ToString();
    }

  // 4. Show the form
  form2.ShowDialog();
}
```

⇑

11. Programming an event of clicking on the button4 ("Save to file "Res_Matr.txt"")

To save the result matrix Matr3, you need to use the capabilities of class "FileStream".

The class FileStream is described in the namespace System.IO. Therefore, in the beginning of module "Form1.cs" you need to add the following code:

```
using System.IO;
```

Listing of event handler of clicking on the button4 is the following:

```
private void button4_Click(object sender, EventArgs e)
{
  FileStream fw = null;
  string msg;
  byte[] msgByte = null; // array of bytes

  // 1. Open file for writing
  fw = new FileStream("Res_Matr.txt", FileMode.Create);

  // 2. Saving the matrix of result in file

  // 2.1. Save the number of elements of the matrix Matr3
  msg = n.ToString() + "\r\n";

  // Converting the string msg into a byte array msgByte
  msgByte = Encoding.Default.GetBytes(msg);

  // save of array msgByte into the file
  fw.Write(msgByte, 0, msgByte.Length);

  // 2.2. Now saving of the matrix
  msg = "";
  for (int i = 0; i < n; i++)
  {
    // forming of a string based on the matrix
    for (int j = 0; j < n; j++)
      msg = msg + Matr3[i, j].ToString() + "  ";
    msg = msg + "\r\n"; // new line
  }

  // 3. Converting the strings into a byte array
  msgByte = Encoding.Default.GetBytes(msg);

  // 4. Saving the strings into the file
  fw.Write(msgByte, 0, msgByte.Length);

  // 5. Close the file
  if (fw != null) fw.Close();
}
```

⥣

12. Run the application

Now you can run the application.

⥣

In above program, use the following code to use operator +. -. * by adding 3 more buttons Add, Sub and Mul

```
public static matrix operator +(matrix a, matrix b)
        {
            int row = a.Row;
            int col = a.Col;
            matrix c = new matrix(row, col);
            for (int i = 0; i < row; i++)
            {
                for (int j = 0; j < col; j++)
                {
                    c.mt[i, j] = a.mt[i, j] + b.mt[i, j];
                }
            }
            return c;
        }
        public static matrix operator -(matrix a, matrix b)
        {
            int row = a.Row;
            int col = a.Col;
            matrix c = new matrix(row, col);
            for (int i = 0; i < row; i++)
            {
                for (int j = 0; j < col; j++)
                {
                    c.mt[i, j] = a.mt[i, j] - b.mt[i, j];
                }
            }
            return c;
        }
        public static matrix operator *(matrix a, matrix b)
        {
            int row = a.Row;
            int col = b.Col;
            matrix c = new matrix(row, col);
            for (int i = 0; i < row; i++)
                for (int j = 0; j < col; j++)
                {
                    c.mt[i, j] = 0;
                    for (int k = 0; k < b.Row; k++)
                    {
                        c.mt[i, j] += a.mt[i, k] * b.mt[k, j];
                    }
                }
            return c;
        }
```