Пример создания двумерной матрицы на форме.

Часто в задачах нужно ввести числа или другие данные в двумерный массив (матрицу) и иметь возможность их обрабатывать.

В работе для представления данных в виде двумерной таблицы строк используется двумерный массив элементов управления типа TextBox.

Содержание

Условие задачи

Составить программу, которая осуществляет произведение двух матриц размерностью *n*. Матрицы вводятся из клавиатуры в отдельной форме и заносятся во внутренние структуры данных. Пользователь имеет возможность просмотреть результирующую матрицу.

Также есть возможность сохранения результирующей матрицы в текстовом файле “Res_Matrix.txt”.

Выполнение

1. Запуск Microsoft Visual Studio. Создание проекта

Подробный пример запуска Microsoft Visual Studio и создания приложения по шаблону Windows Forms Application описывается в теме:

- **Создание приложения Windows Forms Application в MS Visual Studio**

Сохранить проект под любым именем.

2. Создание главной формы Form1

Создать форму, как показано на рисунке 1.

Разместить на форме элементы управления следующих типов:

- четыре элемента управления типа Button. Автоматически будут созданы четыре объекта (переменные) с именами button1, button2, button3, button4;
- три элемента управления типа Label с именами label1, label2, label3;
- один элемент управления типа TextBox, доступ к которому можно получить по имени textBox1.

Сформировать свойства элементов управления типа Button и Label:

- в объекте button1 свойство Text = “Ввод матрицы 1 …«;
- в объекте button2 свойство Text = “Ввод матрицы 2 …«;
- в объекте button3 свойство Text = “Результат …»;
- в объекте button4 свойство Text = “Сохранить в файле “Res_Matr.txt” ”;
- в элементе управления label1 свойство Text = “n = ”.

Для настройки вида и поведения формы нужно выполнить следующие действия:

- установить название формы. Для этого свойство Text = “Произведение матриц”;
- свойство StartPosition = “CenterScreen” (форма размещается по центру экрана);
- свойство MaximizeBox = “false” (убрать кнопку развертывания на весь экран).

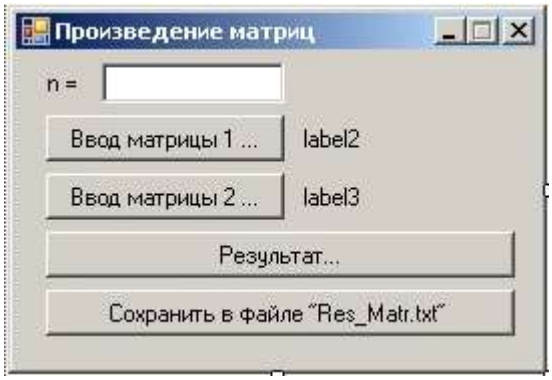

Рис. 1. Форма приложения

3. Создание второстепенной формы Form2

Во второстепенной форме Form2 будут вводиться данные в матрицы и выводиться исходный результат.

Пример создания новой формы в MS Visual Studio – C# подробно описан **здесь**.

Добавить новую форму к приложению, вызвав команду

```
Project -> Add Windows Form ...
```

В открывшемся окне выбрать «Windows Form». Имя файла оставить без изменений «Form2.cs».

Разместить на форме в любом месте элемент управления типа Button (рис. 2). В результате будет получен объект с именем button1.

В элементе управления button1 нужно установить следующие свойства:

- свойство Text = “OK”;
- свойство DialogResult = “OK” (рис. 3). Это означает, что при нажатии (клике «мышкой») на button1, окно закроется с кодом возвращения равным “OK”;
- свойство Modifiers = “Public”. Это означает, что кнопка button1 будет видимой из других модулей (из формы Form1).

Настроить свойства формы Form2:

- свойство Text = “Ввод матрицы”;
- свойство StartPosition = “CenterScreen” (форма размещается по центру экрана);
- свойство MaximizeBox = “false” (убрать кнопку развертывания на весь экран).

Рис. 2. Форма Form2 после настройки

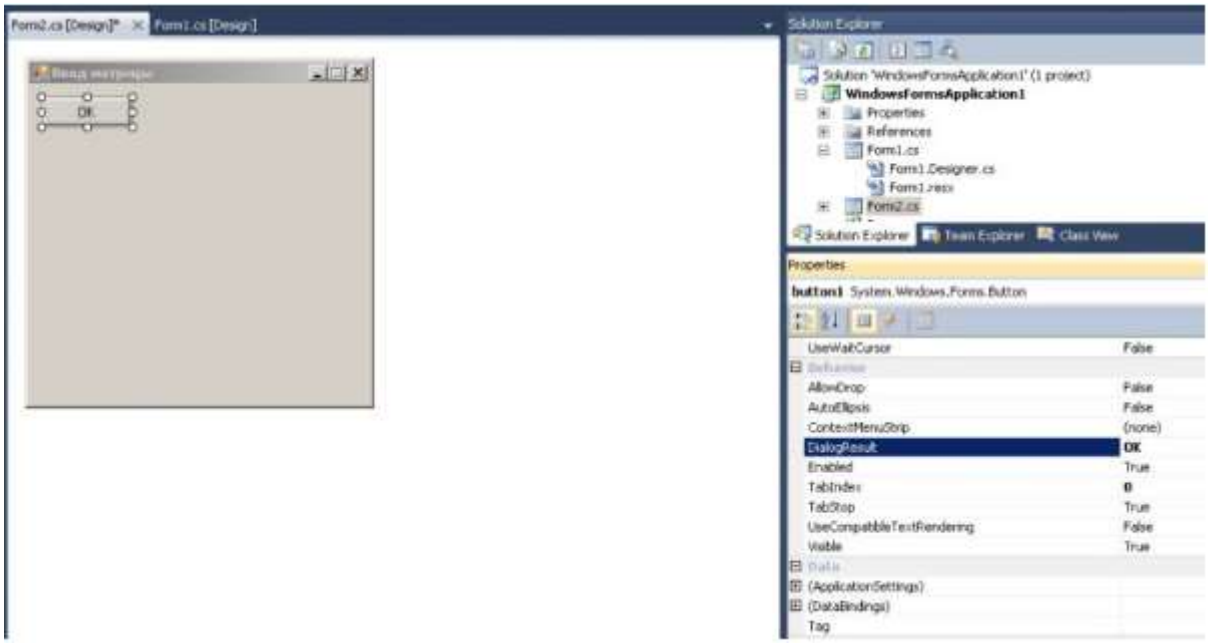

Рис. 3. Свойство DialogResult элемента управления button1 формы Form2

4. Ввод внутренних переменных

Следующий шаг – введение внутренних переменных в текст модуля “Form1.cs”.

Для этого сначала нужно активировать модуль “Form1.cs”.

В тексте модуля “Form1.cs” добавляем следующий код:

```
...
namespace WindowsFormsApplication1
```

```
{
public partial class Form1 : Form
{
const int MaxN = 10; // максимально допустимая размерность матрицы
int n = 3; // текущая размерность матрицы
         TextBox[,] MatrText = null; // матрица элементов типа
TextBox
double[,] Matr1 = new double[MaxN, MaxN]; // матрица 1 чисел с
плавающей точкой
double[,] Matr2 = new double[MaxN, MaxN]; // матрица 2 чисел с
плавающей точкой
double[,] Matr3 = new double[MaxN, MaxN]; // матрица результатов
bool f1; // флажок, который указывает о вводе данных в матрицу
Matr1
bool f2; // флажок, который указывает о вводе данных в матрицу
Matr2
int dx = 40, dy = 20; // ширина и высота ячейки в MatrText[,]
Form2
form2 = null;   // экземпляр (объект) класса формы Form2
public Form1()
{
InitializeComponent();
          }
    }
 }
...
```

Объясним некоторые значения переменных:

- Max – максимально-допустимая размерность матрицы;
- n – размерность матрицы, введенная пользователем из клавиатуры в элементе управления TextBox1;
- MatrText – двумерная матрица элементов управления типа TextBox. В эту матрицу будут вводиться элементы матрицы в виде строк. Ввод данных будет формироваться в форме Form2;
- Matr1, Matr2 – матрицы элементов типа double, в которые будут копироваться данные из матрицы MatrText;
- Matr3 – результирующая матрица, которая равная произведению матриц Matr1 и Matr2;
- f1, f2 – переменные, определяющие были ли введенные данные соответственно в матрицы Matr1 и Matr2;
- dx, dy – габариты одной ячейки типа TextBox в матрице MatrText;

- form2 – объект класса формы Form2, по которому будет получен доступ к этой форме.

5. Программирование события Load формы Form1

Процесс программирования любого события в Microsoft Visual C# подробно описан **здесь**.

Листинг обработчика события Load формы Form1 следующий:

```
private void Form1_Load(object sender, EventArgs e)
{
// I. Инициализация элементов управления и внутренних переменных
textBox1.Text = "";
f1 = f2 = false; // матрицы еще не заполнены
label2.Text = "false";
label3.Text = "false";
// II. Выделение памяти и настройка MatrText
int i, j;
// 1. Выделение памяти для формы Form2
form2 = new Form2();
// 2. Выделение памяти под самую матрицу
MatrText = new TextBox[MaxN, MaxN];
// 3. Выделение памяти для каждой ячейки матрицы и ее настройка
   for (i = 0; i < MaxN; i++)
     for (j = 0; j < MaxN; j++)
     {
     // 3.1. Выделить память
     MatrText[i, j] = new TextBox();
     // 3.2. Обнулить эту ячейку
     MatrText[i, j].Text = "0";
     // 3.3. Установить позицию ячейки в форме Form2
     MatrText[i, j].Location = new System.Drawing.Point(10 + i *
dx, 10 + j * dy);
     // 3.4. Установить размер ячейки
     MatrText[i, j].Size = new System.Drawing.Size(dx, dy);
     // 3.5. Пока что спрятать ячейку
     MatrText[i, j].Visible = false;
     // 3.6. Добавить MatrText[i,j] в форму form2
     form2.Controls.Add(MatrText[i, j]);
     }
```

```
    }
```

Объясним некоторые фрагменты кода в методе Form1_Load().

Событие Load генерируется (вызывается) в момент загрузки любой формы. Поскольку форма Form1 есть главной формой приложения, то событие Load формы Form1 будет вызываться сразу после запуска приложения на выполнение. Поэтому, здесь целесообразно ввести начальную инициализацию глобальных элементов управления и внутренних переменных программы. Эти элементы управления могут быть вызваны из других методов класса.

В обработчике события Form1_Load() выделяется память для двумерной матрицы строк MatrText один лишь раз. При завершении приложения эта память будет автоматически освобождена.

Память выделяется в два этапа:

- для самой матрицы MatrText – как двумерного массива;
- для каждого элемента матрицы, который есть сложным объектом типа TextBox.

После выделения памяти для любого объекта осуществляется настройка основных внутренних свойств (позиция, размер, текст, видимость в некоторой форме).

Также каждая созданная ячейка добавляется (размещается) на форму Form2 с помощью метода Add() из класса Controls. Каждая новая ячейка может быть добавлена в любую другую форму приложения.

6. Разработка дополнительного метода обнуления данных в матрице MatrText

В будущем, чтобы многократно не использовать код обнуления строк матрицы MatrText, нужно создать собственный метод (например, Clear_MatrText()) реализующий этот код.

Листинг метода Clear_MatText() следующий:

```
private void Clear_MatrText()
{
// Обнуление ячеек MatrText
for (int i = 0; i < n; i++)
     for (int j = 0; j < n; j++)
     MatrText[i, j].Text = "0";
     }
```

7. Программирование события клика на кнопке button1 («Ввод матрицы 1 …»)

При нажатии (клике) на button1 должно вызываться окно ввода новой матрицы. Размер матрицы зависит от значения n.

Листинг обработчика события клика на кнопке button1 следующий:

```
private void button1_Click(object sender, EventArgs e)
{
// 1. Чтение размерности матрицы
if (textBox1.Text == "") return;
n = int.Parse(textBox1.Text);
// 2. Обнуление ячейки MatrText
Clear_MatrText();
// 3. Настройка свойств ячеек матрицы MatrText
  //    с привязкой к значению n и форме Form2
for (int i = 0; i < n; i++)
     for (int j = 0; j < n; j++)
     {
     // 3.1. Порядок табуляции
     MatrText[i, j].TabIndex = i * n + j + 1;
     // 3.2. Сделать ячейку видимой
     MatrText[i, j].Visible = true;
     }
     // 4. Корректировка размеров формы
     form2.Width = 10 + n * dx + 20;
     form2.Height = 10 + n * dy + form2.button1.Height + 50 ;
     // 5. Корректировка позиции и размеров кнопки на форме Form2
     form2.button1.Left = 10;
     form2.button1.Top = 10 + n * dy + 10;
     form2.button1.Width = form2.Width - 30;
     // 6. Вызов формы Form2
     if (form2.ShowDialog() == DialogResult.OK)    {
     // 7. Перенос строк из формы Form2 в матрицу 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. Данные в матрицу Matr1 внесены
```

```
      f1 = true;
      label2.Text = "true";
      }
      }
```

В вышеприведенном листинге читается значение n. После этого осуществляется настройка ячеек матрицы строк MatrText.

На основе введенного значения n формируются размеры формы form2 и позиция кнопки button1.

Если в форме Form2 пользователь нажал на кнопке OK (button2), то строки с MatrText переносятся в двумерную матрицу вещественных чисел Matr1. Преобразование из строки в соответствующее вещественное число выполняется методом Parse() из класса Double.

Также формируется переменная f1, которая указывает что данные в матрицу Matr1 внесены.

8. Программирование события клика на кнопке button2 (“Ввод матрицы 2…«)

Листинг обработчика события клика на кнопке button2 подобен листингу обработчика события клика на кнопке button1. Только он отличается шагами 7-8. На этом участке формируются матрица Matr2 и переменная f2.

```
private void button2_Click(object sender, EventArgs e)
{
// 1. Чтение размерности матрицы
if (textBox1.Text == "") return;
n = int.Parse(textBox1.Text);
// 2. Обнулить ячейки MatrText   Clear_MatrText();
// 3. Настройка свойств ячеек матрицы MatrText
  //    с привязкой к значению n и форме Form2
for (int i = 0; i < n; i++)
      for (int j = 0; j < n; j++)
      {
      // 3.1. Порядок табуляции
      MatrText[i, j].TabIndex = i * n + j + 1;
      // 3.2. Сделать ячейку видимой
```

```
MatrText[i, j].Visible = true;
}
// 4. Корректировка размеров формы
form2.Width = 10 + n * dx + 20;
form2.Height = 10 + n * dy + form2.button1.Height + 50;
// 5. Корректировка позиции и размеров кнопки на форме Form2
form2.button1.Left = 10;
form2.button1.Top = 10 + n * dy + 10;
form2.button1.Width = form2.Width - 30;
// 6. Вызов формы Form2
if (form2.ShowDialog() == DialogResult.OK)
{
// 7. Перенос строк из формы Form2 в матрицу Matr2
for (int i = 0; i < n; i++)
for (int j = 0; j < n; j++)
Matr2[i, j] = Double.Parse(MatrText[i, j].Text);
// 8. Матрица Matr2 сформирована
f2 = true;
label3.Text = "true";
}
}
```

9. Программирование события Leave потери фокуса ввода элементом управления textBox1

В приложении может возникнуть ситуация, когда пользователь изменяет значение n на новое. В этом случае должны заново формироваться флажки f1 и f2. Также изменяется размер матрицы MatrText, которая выводится в форме Form2.

Изменение значения n можно проконтролировать с помощью события Leave элемента управление textBox1. Событие Leave генерируется в момент потери фокуса ввода элементом управления textBox1 (рис. 4).

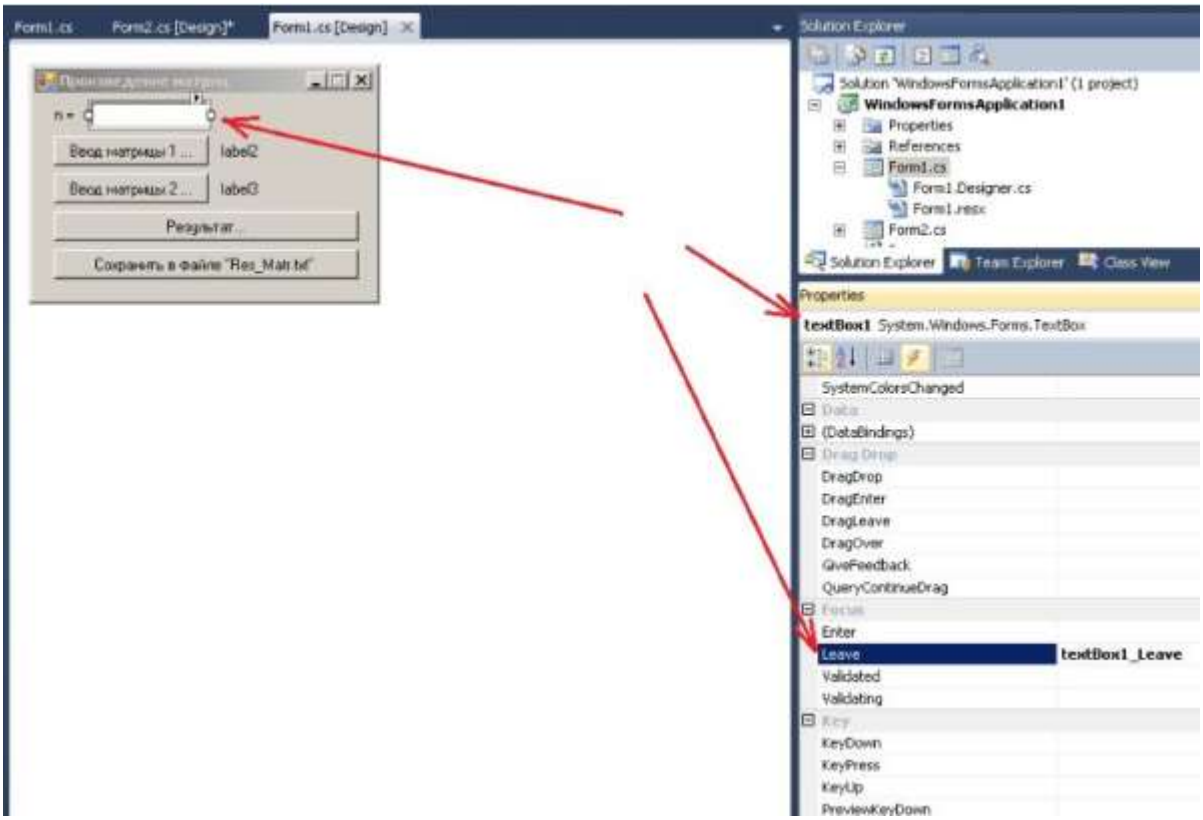

Рис. 4. Событие Leave элемента управления textBox1

Листинг обработчика события Leave следующий:

```
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. Программирование события клика на кнопке button3 («Результат»)

Вывод результата будет осуществляться в ту же форму, в которой вводились матрицы Matr1 и Matr2. Сначала произведение этих матриц будет сформировано в матрице Matr3. Потом значение с Matr3 переносится в MatrText и отображается на форме Form2.

Листинг обработчика события клика на кнопке button3.

```
private void button3_Click(object sender, EventArgs e)
{
// 1. Проверка, введены ли данные в обеих матрицах
if (!((f1 == true) && (f2 == true))) return;
// 2. Вычисление произведения матриц. Результат в 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. Внесение данных в MatrText
for (int i = 0; i < n; i++)
     for (int j = 0; j < n; j++)
     {
     // 3.1. Порядок табуляции
     MatrText[i, j].TabIndex = i * n + j + 1;
     // 3.2. Перевести число в строку
     MatrText[i, j].Text = Matr3[i, j].ToString();
     }
     // 4. Вывод формы
     form2.ShowDialog();
     }
```

11. Программирование события клика на кнопке button4 («Сохранить в файле «Res_Matr.txt”»)

Для сохранения результирующей матрицы Matr3 можно использовать возможности класса FileStream.

Класс FileStream описан в модуле System.IO. Поэтому в начале приложения нужно добавить следующий код:

```
using System.IO;
```

Листинг обработчика события клика на кнопке button4 следующий:

```
private void button4_Click(object sender, EventArgs e)
```

```
{
FileStream fw = null;
string msg;
byte[] msgByte = null; // байтовый массив
// 1. Открыть файл для записи   fw = new FileStream("Res_Matr.txt",
FileMode.Create);
// 2. Запись матрицы результата в файл

// 2.1. Сначала записать число элементов матрицы Matr3
msg = n.ToString() + "\r\n";
// перевод строки msg в байтовый массив msgByte
msgByte = Encoding.Default.GetBytes(msg);
// запись массива msgByte в файл
fw.Write(msgByte, 0, msgByte.Length);
// 2.2. Теперь записать саму матрицу
msg = "";
for (int i = 0; i < n; i++)
{
// формируем строку msg из элементов матрицы
for (int j = 0; j < n; j++)
msg = msg + Matr3[i, j].ToString() + "  ";
msg = msg + "\r\n";
// добавить перевод строки
}
// 3. Перевод строки msg в байтовый массив msgByte
msgByte = Encoding.Default.GetBytes(msg);
// 4. запись строк матрицы в файл
fw.Write(msgByte, 0, msgByte.Length);
// 5. Закрыть файл
if (fw != null) fw.Close();
}
```

12. Запуск приложения на выполнение

После этого можно запускать приложение на выполнение и тестировать его работу.