Практическая работа 10
Массив. Работа с массивами в WindowsForms

Цель занятия

Получить практические навыки объявления и обработки информации в массивах с использованием WindowsForms

Перечень оборудования и программного обеспечения

Персональный компьютер
Microsoft Office (Word, Visio)
Microsoft Visual Studio 2010

Краткие теоретические сведения

Элемент управления ListBox

Windows Forms ListBox элемент управления отображает список, из которого пользователь может выбрать один или несколько элементов. Если общее количество элементов превышает номер, который может быть отображен, полосы прокрутки автоматически добавляется ListBox элемента управления.

Программное управление элементами в ListBox

Добавление элементов

Итак, все элементы списка входят в свойство Items, которое представляет собой коллекцию. Для добавления нового элемента в эту коллекцию, а значит и в список, надо использовать метод Add, например:

listBox1.Items.Add("Новый элемент");.

При использовании этого метода каждый добавляемый элемент добавляется в конец списка.

Можно добавить сразу несколько элементов, например, массив. Для этого используется метод AddRange:

```
string[] countries = { "Бразилия", "Аргентина", "Чили", "Уругвай", "Колумбия" };
listBox1.Items.AddRange(countries);
```

Вставка элементов

В отличие от простого добавления вставка производится по определенному индексу списка с помощью метода Insert:

listBox1.Items.Insert(1, "Парагвай");

В данном случае вставляем элемент на вторую позицию в списке, так как отсчет позиций начинается с нуля.

Удаление элементов

Для удаления элемента по его тексту используется метод Remove:

listBox1.Items.Remove("Чили");

Чтобы удалить элемент по его индексу в списке, используется метод RemoveAt:

listBox1.Items.RemoveAt(1);

Кроме того, можно очистить сразу весь список, применив метод Clear:

listBox1.Items.Clear();

Доступ к элементам списка

Используя индекс элемента, можно сам элемент в списке. Например, получим первый элемент списка:

string firstElement = listBox1.Items[0];

Метод Count позволяет определить количество элементов в списке:

int number = listBox1.Items.Count();

Выделение элементов списка

При выделении элементов списка мы можем ими управлять как через индекс, так и через сам выделенный элемент. Получить выделенные элементы можно с помощью следующих свойств элемента ListBox:

SelectedIndex: возвращает или устанавливает номер выделенного элемента списка. Если выделенные элементы отсутствуют, тогда свойство имеет значение -1

SelectedIndices: возвращает или устанавливает коллекцию выделенных элементов в виде набора их индексов

SelectedItem: возвращает или устанавливает текст выделенного элемента

SelectedItems: возвращает или устанавливает выделенные элементы в виде коллекции

По умолчанию список поддерживает выделение одного элемента. Чтобы добавить возможность выделения нескольких элементов, надо установить у его свойства SelectionMode значение MultiSimple.

Чтобы выделить элемент програмно, надо применить метод SetSelected(int index, bool value), где index - номер выделенного элемента. Если второй параметр - value имеет значение true, то элемент по указанному индексу выделяется, если false, то выделение наоборот скрывается:

listBox1.SetSelected(2, true); // будет выделен третий элемент

Чтобы снять выделение со всех выделенных элементов, используется метод ClearSelected.

Событие SelectedIndexChanged

Из всех событий элемента ListBox надо отметить в первую очередь событие SelectedIndexChanged, которое возникает при изменении выделенного элемента:

```
public partial class Form1 : Form
{
   public Form1()
   {
      InitializeComponent();
```

```
        string[] countries = { "Бразилия", "Аргентина", "Чили", "Уругвай",
"Колумбия" };
        listBox1.Items.AddRange(countries);
        listBox1.SelectedIndexChanged += listBox1_SelectedIndexChanged;
    }
    void listBox1_SelectedIndexChanged(object sender, EventArgs e)
    {
        string selectedCountry = listBox1.SelectedItem.ToString();
        MessageBox.Show(selectedCountry);
    }
}
```

В данном случае по выбору элемента списка будет отображаться сообщение с выделенным элементом.

Элемент управления dataGridView

В Microsoft Visual Studio элемент управления dataGridView разработан для использования в приложениях, созданных по шаблону Windows Forms Application. Данный элемент управления позволяет организовывать данные в виде таблицы. Данные могут быть получены из базы данных, коллекции, внутренних переменных — массивов или других объектов программы.

Данный элемент управления аналогичен компоненту TStringGrid в системе визуальной разработки приложений.

Данный элемент размещен на панели инструментов ToolBox во вкладках «All Windows Forms» или «Data»

После размещения на форме, система создает объект (переменную) с именем dataGridView1. С помощью этого имени можно программно оперировать методами и свойствами этого элемента управления.

В DataGridView данные могут быть получены из базы данных, коллекции, внутренних структур данных (массивов, структур и т.д.).

Виды данных, которые могут быть представлены в ячейках dataGridView:

- dataGridViewButtonColumn. Ячейки представлены в виде кнопок типа Button;
- dataGridViewCheckBoxColumn. Ячейки представлены элементами управления типа CheckBox, которые позволяют выбирать несколько вариантов (опций) из набора предложенных;
- dataGridViewComboBoxColumn. Ячейки представлены элементами управления типа ComboBox, предназначенных для выбора одного из нескольких вариантов;
- dataGridViewImageColumn. Ячейки таблицы есть изображениями типа Image;
- dataGridViewLinkColumn. Ячейки таблицы представлены ссылками;

– dataGridViewTextBoxColumn. Этот вариант предлагается по умолчанию при добавлении (создании) нового столбца. В этом случае ячейки таблицы представлены в виде полей ввода. Это позволяет вводить данные в таблицу как в матрицу.

Добавление столбца программным путем

Добавить столбец в dataGridView можно:

- с помощью специального мастера;
- программным путем.

Столбцы в dataGridView организованы в виде коллекции Columns типа DataGridViewColumnCollection.Чтобы добавить столбец программным путем используется метод (команда) Add из коллекции Columns.

Метод Add имеет 2 варианта реализации:

```
int DataGridViewColumnCollection.Add(DataGridViewColumn
dataGridViewColumn);

int DataGridViewColumnCollection.Add(string ColumnName, string HeaderText);
```

где

- DataGridViewColumn – тип System.Windows.Forms.Column который добавляется;
- ColumnName – название, по которому будет осуществляться обращение к столбцу из других методов;
- HeaderText – текст, который будет отображаться в заголовке столбца.

Фрагмент кода, который добавляет два произвольных столбца следующий:

```
// Добавить столбец с именем column-1, заголовок столбца - "Header column - 1"
dataGridView1.Columns.Add("column-1", "Header column - 1");
// Добавить столбец с именем column-2
dataGridView1.Columns.Add("column-2", "Header column - 2");
```

В реальных программах название столбца и его заголовка получаются из других элементов управления, например TextBox.

Для вставки столбца используется метод Insert, который имеет следующее объявление

```
void DataGridViewColumnCollection.Insert(int columnIndex, DataGridViewColumn
dataGridViewColumn);
```

Вызов этого метода из программного кода аналогичен методу Add.

Как программно реализовать удаление столбца?
Методы Remove() и RemoveAt()

Чтобы удалить столбец используется один из двух методов из коллекции Columns:

- метод RemoveAt() – удаляет столбец по заданному индексу в коллекции;
- метод Remove() – удаляет столбец по его имени.

Общий вид метода RemoveAt():

void DataGridViewColumnCollection.RemoveAt(int index);

где index – заданный индекс в коллекции. Индексы нумеруются с 0.

void DataGridViewColumnCollection.Remove(string ColumnName);

где ColumnName – название столбца (но не название заголовка столбца), которое задается в методе Add()первым параметром. Столбцы в коллекции могут иметь одинаковые значения ColumnName. Если при вызове метода Remove(), столбца с именем ColumnName нет, то генерируется исключительная ситуация.

Фрагмент кода удаления столбца с помощью метода RemoveAt():

```
// удаление столбца в позиции index
int index; // номер столбца, который удаляется
int n; // текущее количество столбцов в dataGridView

// задать номер столбца, который удаляется
index = 1;

// определить текущее количество столбцов в dataGridView
n = dataGridView1.Columns.Count;

// удаление
if ((n > 0) && (index >= 0) && (index < n))
{
   dataGridView1.Columns.RemoveAt(index);
   label1.Text = "Столбец удален";
}
else
{
   label1.Text = "Столбец не удален";
}
```

Программное добавление строки. Метод Add()

Добавлять строку можно одним из двух способов:

- путем непосредственного ввода с клавиатуры;
- программным путем.

Строки в DataGridView организованы в виде коллекции Rows типа dataGridViewRowCollection.

Ниже приведен фрагмент метода, добавляющего 2 произвольные строки в таблицу

```
// Добавить строки в таблицу
if (dataGridView1.Columns.Count <= 0)
{
   label1.Text = "Строки не добавлены";
   return;
}
dataGridView1.Rows.Add("Ivanov I.I.", 25, "New York");
dataGridView1.Rows.Add("Petrenko P.P.", 38, "Moscow");
label1.Text = "Строки добавлены";
```

Программное удаление строки. Методы Remove() и RemoveAt()

Для удаления строки используется один из двух методов:

- метод RemoveAt() – удаляет строку по заданному индексу;
- метод Remove() – удаляет строку, которая есть входным параметром типа DataGridViewRow.

Фрагмент кода удаления строки имеет вид:

```
// Удалить строку
int nr, nc;
nc = dataGridView1.Columns.Count; // количество столбцов
nr = dataGridView1.RowCount;
if ((nc > 0) && (nr > 1))
{
   dataGridView1.Rows.RemoveAt(0); // удалить первую строку
   label1.Text = "Строка удалена";
}
else
{
   label1.Text = "Строка не удалена";
}
```

Задание текста заголовка в заданном столбце программным путем

Чтобы задать текст заголовка в заданном столбце используется свойство HeaderText. Фрагмент кода установки текста заголовка в столбце с индексом 0 имеет вид:

```
// задать текст в заголовке
int nc = dataGridView1.ColumnCount;
if (nc > 0)
{
   // задать новый текст заголовке первого столбца
   dataGridView1.Columns[0].HeaderText = "Header - 1";
   label1.Text = "Текст задан";
}
else
{
   label1.Text = "Текст не задан";
}
```

Установка выравнивания заголовка в заданном столбце программным путем

Выравнивание заголовка в столбце задается с помощью свойства HeaderCell.Style.Alignment.

Фрагмент кода установки выравнивания в заголовке столбца с индексом 0:

```
// выравнивание заголовка
int nc;
nc = dataGridView1.ColumnCount;
if (nc > 0)
{
   // задать выравнивание по центру (по горизонтали и по вертикалу)
   dataGridView1.Columns[0].HeaderCell.Style.Alignment =
DataGridViewContentAlignment.MiddleCenter;
   label1.Text = "Выравнивание выполнено";
}
else
{
   label1.Text = "Выравнивание не выполнено";
}
```

Установка шрифта заголовка в столбцах программным путем

Для установки шрифта в заголовках столбцов используется свойство ColumnHeadersDefaultCellStyle. В этом свойстве используется свойство Font.

Во фрагменте кода создается шрифт Arial, имеющий размер 12 и курсивное начертание.

```
// задать шрифт в заголовке
int nc;
nc = dataGridView1.ColumnCount;
// создать шрифт "Arial", размер 12, начертание - "курсив"
Font F = new Font("Arial", 12, FontStyle.Italic);
if (nc > 0)
{
   // установить шрифт заголовка
   dataGridView1.ColumnHeadersDefaultCellStyle.Font = F;
   label1.Text = "Шрифт задан";
}
else
{
   label1.Text = "Шрифт не задан";
}
```

Установка цвета шрифта заголовков программным путем

Чтобы задать цвет шрифта заголовков программным путем нужно использовать свойство ColumnHeaderDefaultCellStyle. В этом свойстве есть свойства ForeColor и BackColor.

```
int nc;
nc = dataGridView1.ColumnCount;
if (nc > 0)
{
   // создать системный шрифт
   Font F = new Font("Arial", 14);
   // задать цвет в заголовках столбцов
   dataGridView1.ColumnHeadersDefaultCellStyle.ForeColor = Color.Purple;
   // задать шрифт
   dataGridView1.Columns[0].DefaultCellStyle.Font = F;
   label1.Text = "Цвет заголовка изменен";
}
else
{
   label1.Text = "Цвет не изменен";
}
```

Установка размеров dataGridView1 программным путем

```
// задать размер dataGridView1
dataGridView1.Width = 600;
dataGridView1.Height = 150;
```

Установка ширины заданного столбца dataGridView1

```
// задать ширину столбца
int nc;
nc = dataGridView1.ColumnCount;
if (nc > 0)
{
   // задать ширину столбца с индексом 0
   dataGridView1.Columns[0].Width = 70;
   label1.Text = "Ширина столбца задана";
}
else
{
   label1.Text = "Ширина столбца не задана";
}
```

Установка высоты заданной строки dataGridView1

```
// задать высоту строки
int nc, nr;
nc = dataGridView1.ColumnCount;
nr = dataGridView1.RowCount;
if ((nc > 0) && (nr > 1))
{
   dataGridView1.Rows[0].Height = 50;
   label1.Text = "Высота строки задана";
}
else
{
   label1.Text = "Высота строки не задана";
}
```

Установка выравнивания в заданном столбце и строке

```
// выравнивание в строках
int nc, nr;
nc = dataGridView1.ColumnCount;
nr = dataGridView1.RowCount;
if ((nc > 0)&&(nr>1))
{
   // выравнивание для всех строк
   dataGridView1.RowsDefaultCellStyle.Alignment =
DataGridViewContentAlignment.BottomRight;
   // выравнивание для строки с индексом 0
   dataGridView1.Rows[0].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
   // выравнивание для столбца с индексом 0
```

```
   dataGridView1.Columns[0].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.BottomLeft;
}
```

Установка шрифта, цвета символов и фона в первом столбце

Чтобы задать шрифт, цвет символов и фона в первом столбце, используется свойство DefaultCellStyleстолбца с индексом 0. В этом свойстве есть свойства Font, BackColor, ForeColor.

Ниже приведен фрагмент кода, который задает цвет шрифта, символов и фона в dataGridView1.

```
// шрифт и цвет в первом столбце
int nc, nr;
nc = dataGridView1.ColumnCount;
nr = dataGridView1.RowCount;
if ((nc > 0) && (nr > 1))
{
   // создать шрифт
   Font F = new Font("Times New Roman", 10, FontStyle.Bold);

   // цвет символов и фона в первом столбце
   dataGridView1.Columns[0].DefaultCellStyle.BackColor = Color.Red;
   dataGridView1.Columns[0].DefaultCellStyle.ForeColor = Color.Blue;
   // шрифт в первом столбце
   dataGridView1.Columns[0].DefaultCellStyle.Font = F;
   label1.Text = "Шрифт и цвет в 1-м столбце изменен";
}
else
{
   label1.Text = "Шрифт не изменен";
}
```

Определение количества столбцов

```
// определить количество столбцов
int n;
n = dataGridView1.Columns.Count;
label1.Text = n.ToString();
```

Определение количества строк

```
// определить количество строк без строки заголовка
int n;
n = dataGridView1.Rows.Count;
label1.Text = (n - 1).ToString();
```

Определение ширины заданного столбца в пикселях

```
// ширина столбца в пикселях
int w;
int nc;
nc = dataGridView1.Columns.Count;
if (nc > 0)
{
   w = dataGridView1.Columns[0].Width;
   label1.Text = w.ToString();
}
```

Определение высоты заданной строки в пикселях

```
// определить высоту строки в пикселях
int h;
int nr, nc;
nc = dataGridView1.Columns.Count;
nr = dataGridView1.RowCount;
if ((nr>1)&&(nc>0))
{
   h = dataGridView1.Rows[0].Height;
   label1.Text = h.ToString();
}
```

Задания

1 Изучить теоретические сведения и задание к работе

2 Разработать и отладить программу ввода одномерного массива с помощью TextBox, вычисления суммы и произведения максимального и минимального элементов этого массива и вывода массива, состоящего из квадратов элементов исходного массива, с помощью ListBox.

3 Создать двумерный массив в WindowsForms с помощью таблиц, после чего выполнить действия по варианту.

4** Создать двумерный массив в WindowsForms с помощью массива TextBox, после чего выполнить действия по варианту.

Порядок выполнения работы (Пример выполнения)

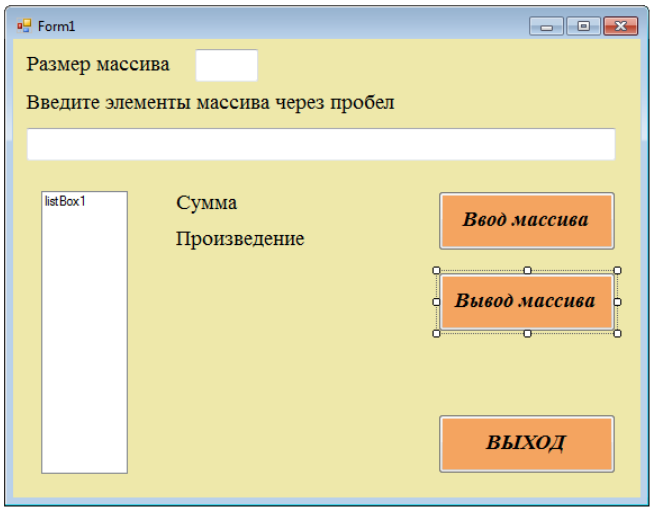
Задание 2

Исходные данные:

Ввести одномерный массив с помощью TextBox, вычислить сумму и произведение элементов этого массива и вывести массив с помощью ListBox.

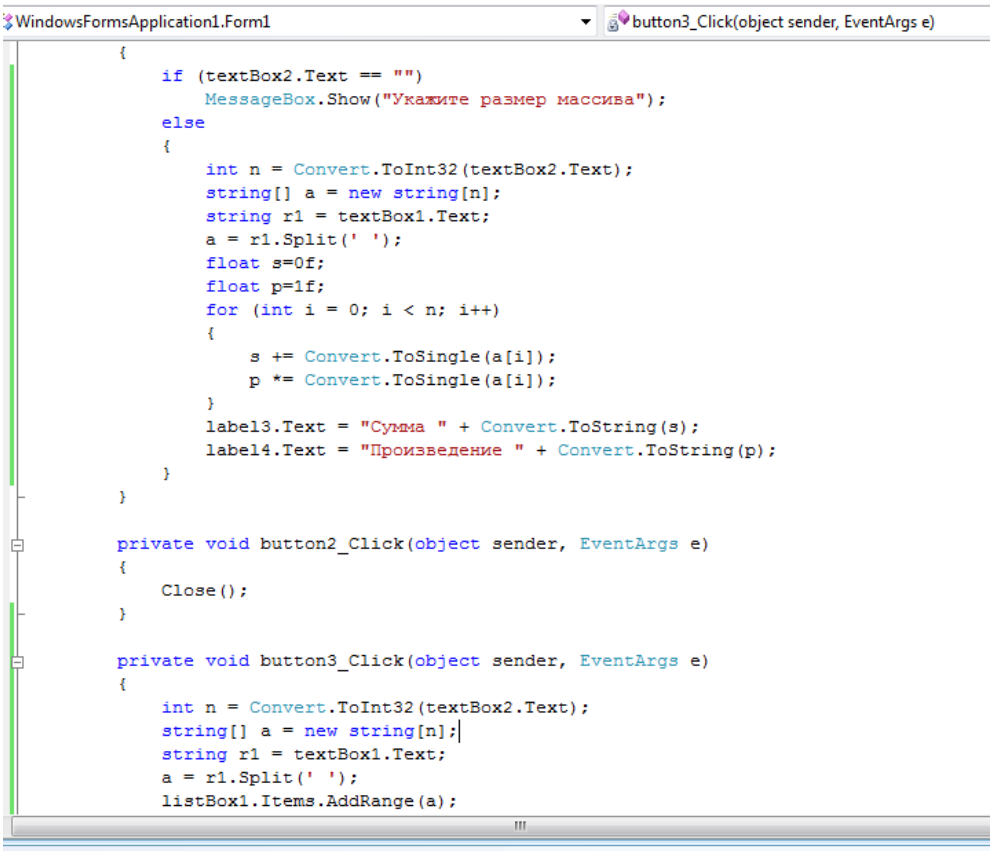
Решение:

1 Интерфейс формы в VisualStudio

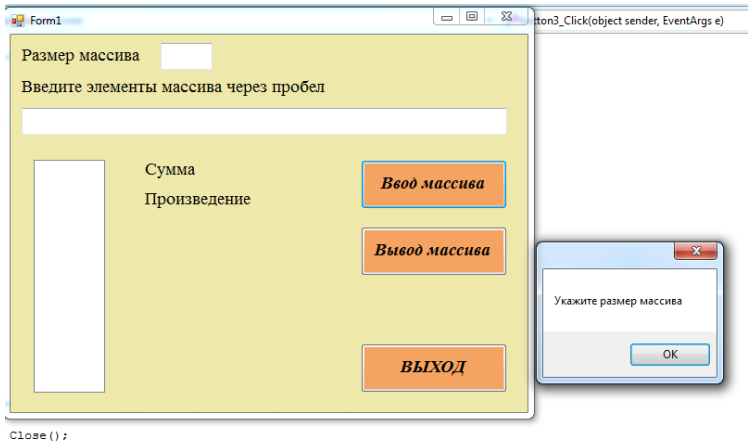


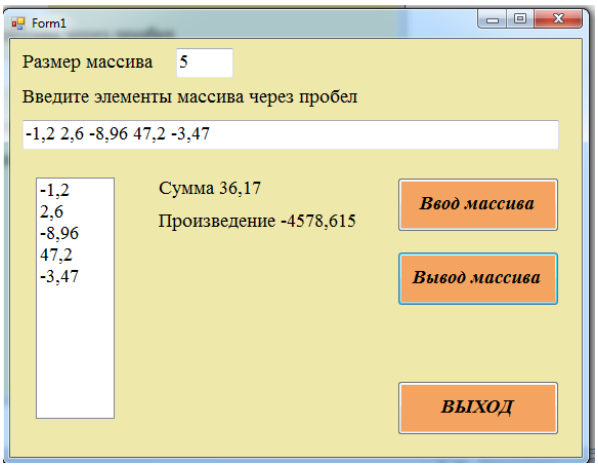
2 Текст программы.

3 VisualStudio.



4 Результаты работы программы:





Задание 3

Исходные данные

Дан целочисленный двумерный массив, размерности n x m. Найти количество и номера нечетных элементов, стоящих на четных местах.

Решение

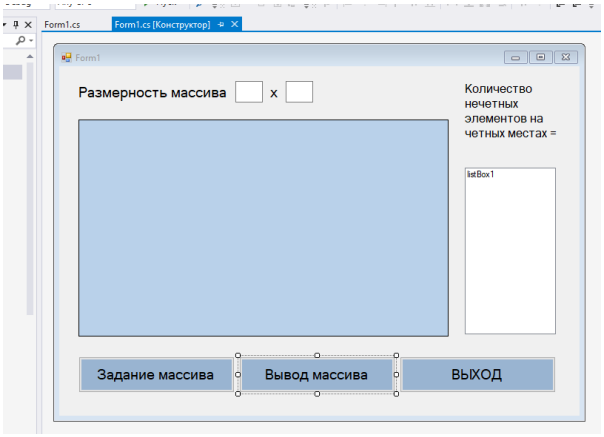
1 Текст программы.

```
private void button1_Click(object sender, EventArgs e)
      {
         int n = Convert.ToInt32(textBox1.Text);
         int m = Convert.ToInt32(textBox2.Text);
         for (int i = 0; i <= m; i++)
…
         label4.Text=Convert.ToString(k);
      }
```

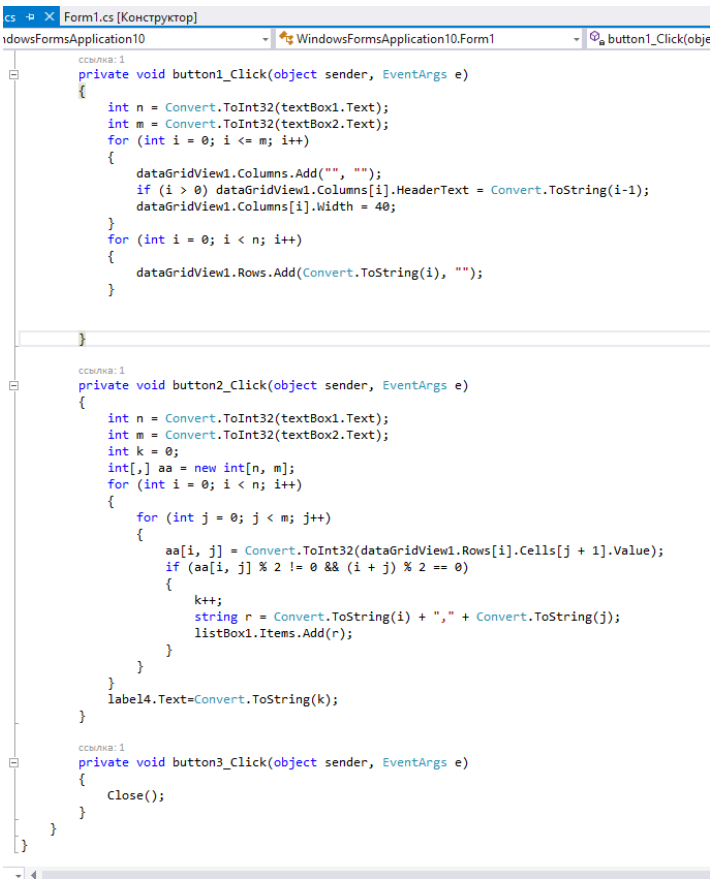
```
private void button3_Click(object sender, EventArgs e)
{
   Close();
}
```

Задание 4**

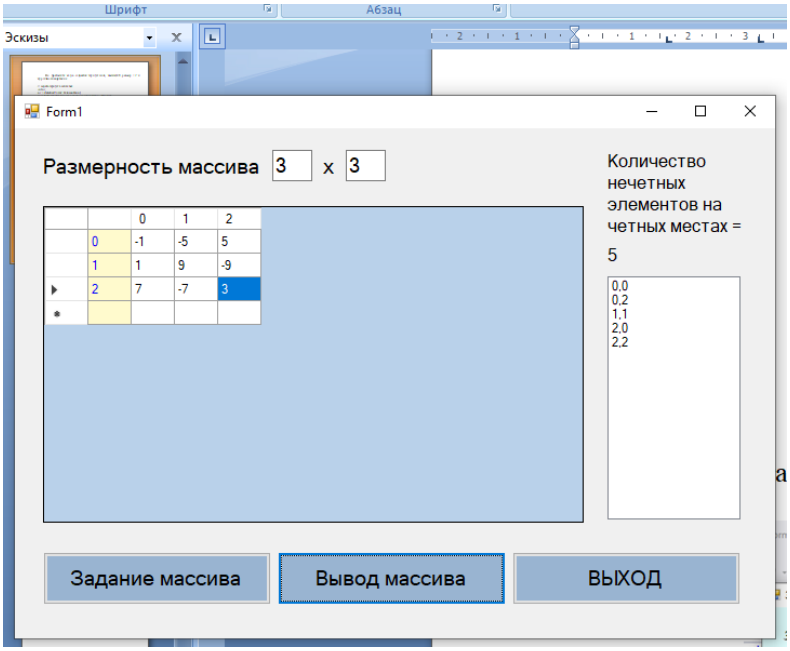
2 Интерфейс формы в VisualStudio



3 Код программы в VisualStudio

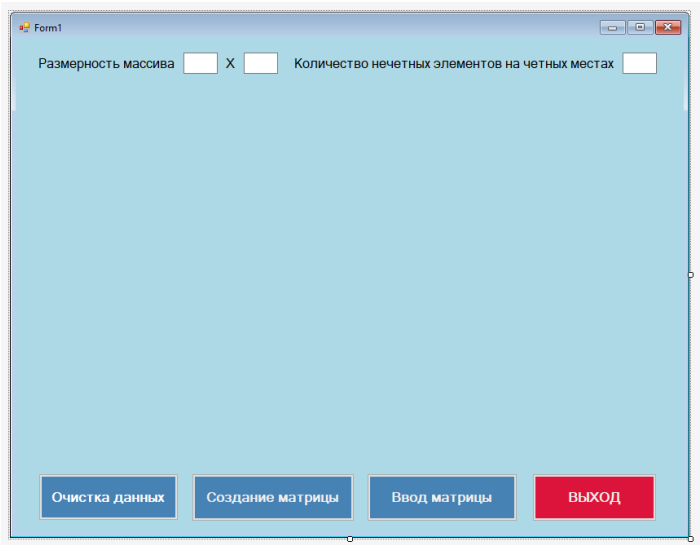


4 Результат работы программы:



4** Рассмотрим другой способ ввода двумерного массива с WindowsForms (задание 3). Для ввода каждого элемента этого массива разместим TextBox. Только проблема в том, что размерность массива заранее не известна, а значит, количество TextBox на форме тоже.

1 Интерфейс формы в VisualStudio



Для решения этой задачи создадим массив, каждый элемент которого – textBox:

```
private TextBox[,] tt = new TextBox[0, 0];
```

Сначала пользователь задает размерность будущего массива, а затем по нажатию на кнопку «Создание матрицы» выполняется следующий программный код, в котором в цикле каждому элементу массива текстовых полей задается имя (Name), размер и позиция на форме, после чего командой Controls.Add(tt[i, j]) этот textbox размещается в указанном месте.

```
public Form1()
{
    InitializeComponent();
}
private TextBox[,] tt = new TextBox[0, 0];
private int n;
private int m;

ссылка: 1
private void button1_Click(object sender, EventArgs e)
{

    n = Convert.ToInt32(textBox1.Text);
    m = Convert.ToInt32(textBox2.Text);
    tt = new TextBox[n, m];

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            tt[i, j] = new TextBox()
            {
                Name = "tt" + (i).ToString() + (j).ToString(),
                Size = new Size(40, 30)
            };
            tt[i, j].Location = new System.Drawing.Point(50 + j * 50, 70 + i * 30);
            tt[i, j].Size = new System.Drawing.Size(40, 30);
            Controls.Add(tt[i, j]);
        }
    }

}
```

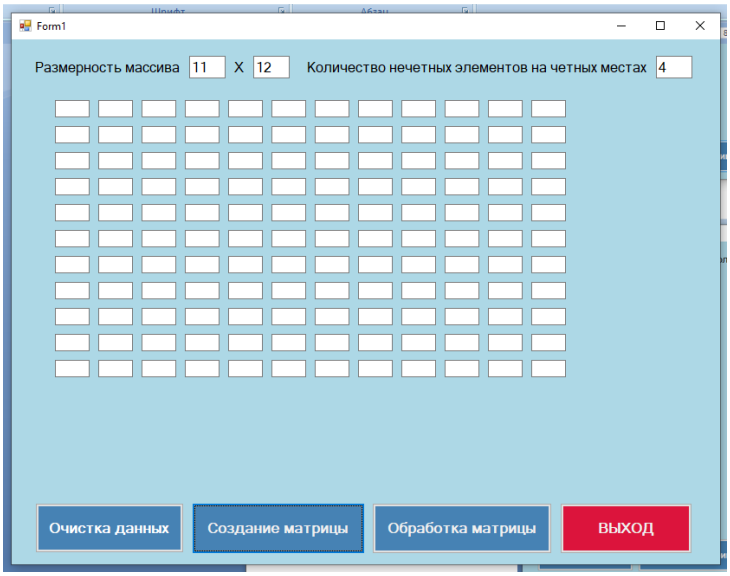
2 Код программы в VisualStudio для обработки матрицы. Здесь содержимое текстовых полей преобразуется в числовой формат и выполняется расчет по заданию.

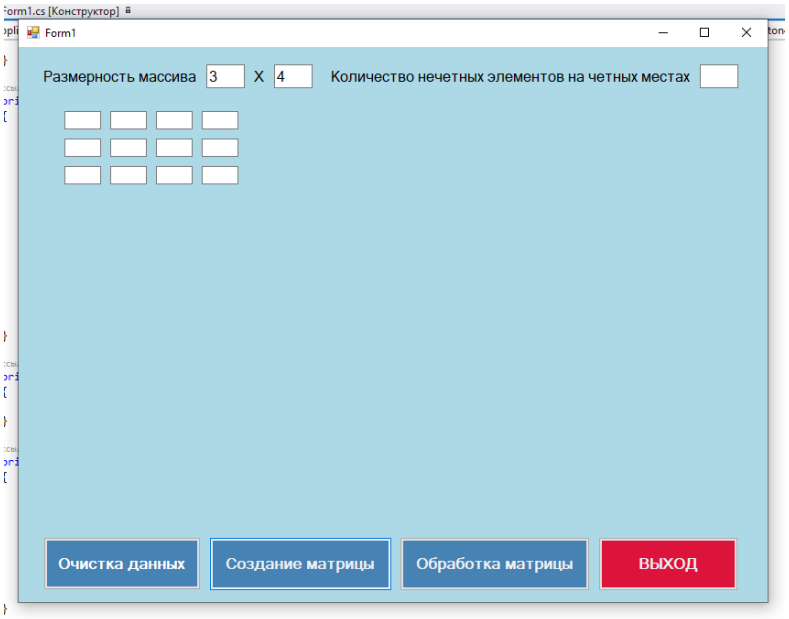
```
ссылка: 1
private void button2_Click(object sender, EventArgs e)
{
    n = Convert.ToInt32(textBox1.Text);
    m = Convert.ToInt32(textBox2.Text);
    int k = 0;
    int[,] aa = new int[n, m];
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            aa[i, j] = Convert.ToInt32(tt[i, j].Text);
            if (aa[i, j] % 2 != 0 && (i + j) % 2 == 0) k++;
        }
    }
    textBox3.Text = Convert.ToString(k);

}
```

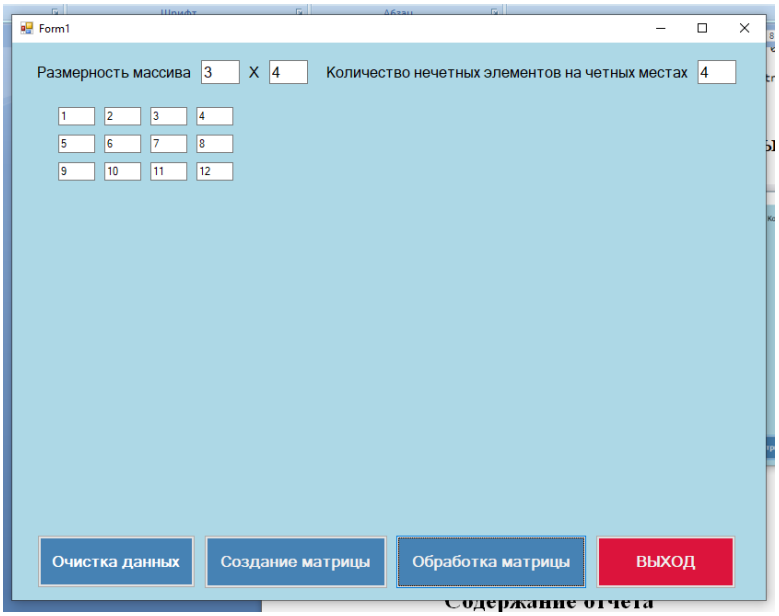
3 Код программы в VisualStudio для очистки формы для работы с новой матрицей. В цикле с помощью команды this.Controls.Remove(tt[i, j]) удаляются текстовые поля для ввода матрицы.

```
ссылка: 1
private void button3_Click(object sender, EventArgs e)
{
    Close();
}

ссылка: 1
private void button4_Click(object sender, EventArgs e)
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            this.Controls.Remove(tt[i, j]);
```

4 Результат работы программы:







Содержание отчета

1 Название работы
2 Цель работы
3 Технические средства обучения
4 Задания (условия задач)
5 Порядок выполнения работы
6 Ответы на контрольные вопросы
7 Вывод

Варианты к практической работе:

Задание 3, 4**

1 Дан целочисленный двумерный массив, размерности n x m, найти сумму всех элементов массива.

2 Дан целочисленный двумерный массив, размерности n x m, найти сумму всех положительных элементов массива.

3 Дан целочисленный двумерный массив, размерности n x m, найти суммы всех элементов каждой строки массива и создать из них массив.

4 Дан целочисленный двумерный массив, размерности n x m, найти сумму всех отрицательных элементов массива.

5 Дан целочисленный двумерный массив, размерности n x m, найти наименьший элемент массива и номер строки, в которой он находится.

6 Дан целочисленный двумерный массив, размерности n x m, найти сумму всех четных элементов массива.

7 Дан целочисленный двумерный массив, размерности n x m, найти наибольший элемент массива и номер строки, в которой он находится.

8 Дан целочисленный двумерный массив, размерности n x m, найти суммы всех элементов каждого столбца массива и создать из них массив.

9 Дан целочисленный двумерный массив, размерности n x m, найти суммы всех нечетных элементов каждого столбца массива и создать из них массив.

10 Дан целочисленный двумерный массив, размерности n x m, найти сумму всех нечетных элементов массива.

11 Дан целочисленный двумерный массив, размерности n x m, найти наибольший элемент массива.

12 Дан целочисленный двумерный массив, размерности n x m, найти наименьший элемент массива.

13 Дан целочисленный двумерный массив, размерности n x m, найти наибольшие элементы каждого столбца массива и создать из них массив.

14 Дан целочисленный двумерный массив, размерности n x m, найти наибольшие элементы каждой строки массива и создать из них массив.

15 Дан целочисленный двумерный массив, размерности n x m. Найти сумму и произведение элементов, кратных 3 и 5.

16 Дан целочисленный двумерный массив, размерности n x m. Найти количество отрицательных элементов, больше -9.

17 Дан целочисленный двумерный массив, размерности n x m. Найти количество отрицательных элементов, больших -20.

18 Дан целочисленный двумерный массив, размерности n x m. Найти среднее арифметическое всех элементов массива.

19 Дан целочисленный двумерный массив, размерности n x m. Найти количество положительных элементов, больших 10.

20 Дан целочисленный двумерный массив, размерности n x m. Найти количество положительных элементов, меньших 15.

21 Дан целочисленный двумерный массив, размерности n x m, найти наибольший элемент каждого столбца массива и создать из них массив.

22 Дан целочисленный двумерный массив, размерности n x m, найти наименьший элемент каждого столбца массива и создать из них массив.

23 Дан целочисленный двумерный массив, размерности n x m, найти наименьший элемент каждой строки массива и создать из них массив.

Контрольные вопросы

1 Какие элементы управления используются для отображения массивов?

2 Как осуществляется доступ программы к элементам массива в Windows Forms?

3 Виды данных, которые могут быть представлены в ячейках таблиц формы.

4 Чем отличается ввод массива в консольном приложении и Windows Forms?

Используемая литература

1. Гниденко, И. Г. Технология разработки программного обеспечения : учеб. пособие для СПО / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — М.: Издательство Юрайт, 2017.
2. Шарп Джон Ш26 Microsoft Visual C#. Подробное руководство. 8-е изд. — СПб.: Питер, 2017.
3. Васильев А.Н. Программирование на C# для начинающих. Основные сведения. – Москва: Эксмо, 2018.
4. Васильев А.Н. Программирование на C# для начинающих. Особенности языка. – Москва: Эксмо, 2019.
5. http://msdn.microsoft.com/ru-ru/library/67ef8sbd.aspx.