

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГОУ ВПО Пермская государственная сельскохозяйственная
академия имени Д.Н. Прянишникова

КОЧКИНА М.А.

Язык программирования С#
в среде Microsoft Visual Studio.NET 2005

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ ДЛЯ ЛАБОРАТОРНЫХ РАБОТ



Пермь 2008

Учебно-методическое пособие составлено старшим преподавателем кафедры информационных систем Кочкиной М.А. в соответствии с программой курса «Объектно-ориентированное программирование» для студентов очного отделения специальности 230201. Может быть использовано в рамках курса «Высокоуровневые методы информатики и программирования» студентами очного отделения специальности 080801.

Содержит методические указания по выполнению лабораторных работ в среде Microsoft Visual Studio. NET, домашние и индивидуальные задания, примеры создания проектов. Основной акцент сделан на разработке приложений для Windows. Уделено внимание работе с базами данных. Приложение содержит полезную справочную информацию.

Не смотря на то, что рассмотренные примеры снабжены комментариями и легко читаемы, предполагается, что студенты уже имеют некоторый навык программирования в других средах.

Утверждено методической комиссией факультета прикладной информатики (протокол № _____ от «_____» _____ 2008 г)

ОГЛАВЛЕНИЕ

| | |
|--|-----|
| Введение | 4 |
| Лабораторная работа № 1. Знакомство со средой Visual Studio. Работа с базовыми компонентами. | 5 |
| Лабораторная работа № 2. Работа со списками, переключателями, таймером..... | 16 |
| Лабораторная работа № 3. Создание калькулятора. Разработка программы-теста. Работа с меню и несколькими формами..... | 27 |
| Лабораторная работа № 4. Работа с массивами..... | 41 |
| Лабораторная работа № 5. Обработка текстовых данных, данных типа дата/время, программа «Угадай число»..... | 54 |
| Лабораторная работа № 6. Работа с файлами | 68 |
| Лабораторная работа № 7. Работа с графикой | 84 |
| Лабораторная работа № 8. Тема: Работа с базами данных..... | 100 |
| Приложение | 120 |
| Индивидуальные задания | 130 |
| Библиографический список..... | 137 |

Введение

C# - относительно новый объектно-ориентированный язык программирования, разработанный компанией Microsoft. На нем можно писать приложения для Web, приложения под Windows, консольные программы, запускаемые из командной строки, работать с базами данных.

C# - это лишь один из языков платформы .NET Framework (помимо Visual Basic.NET, Visual C++.NET, J#, а с появлением Delphi 8/2005 еще и Delphi). Но многие скажут, что это основной язык для .NET. Язык C#, как и Java, многое позаимствовал из C++ (особенно с точки зрения синтаксиса), однако на него повлиял и Visual Basic 6.0. В целом можно сказать, что C# впитал в себя многое из того лучшего, что есть в самых разных языках программирования. Это простой, надежный, полностью объектно-ориентированный язык, используемый платформой .NET Framework.

Платформа .NET - это новая технология разработки программного обеспечения. В ее основе лежит идея универсального программного кода, который может быть выполнен любым устройством, вне зависимости от используемой этим устройством операционной системы (ОС должна поддерживать технологию .NET). Универсальность программного кода обеспечивается за счет предварительной, на этапе разработки, компиляции исходной программы в промежуточный код на языке CIL (Common Intermediate Language), который во время загрузки транслируется в выполняемую программу. Платформа .NET является независимой от используемых языков программирования. Можно использовать несколько .NET-совместимых языков даже в рамках одного проекта.

При написании пособия использовалась среда Microsoft Visual Studio 2005, в которой реализована версия Visual C# 2.0. При использовании других систем программирования (например, Borland C# Builder) и версий языка возможны небольшие отличия.

В пособии рассмотрены разнообразные примеры, демонстрирующие назначение базовых компонентов, технологию работы с графикой и базами данных. Выполнив приведенные примеры, домашние и индивидуальные задания, студент получит соответствующие навыки. Все примеры в пособии снабжены комментариями и некоторыми теоретическими выносками, что существенно облегчит их читаемость. При этом материал не перенасыщен теорией - для этого предусмотрены лекционные занятия и фундаментальные труды по платформе .NET. Научиться программировать можно только программируя, решая конкретные задачи. Не бойтесь экспериментировать - совершенствуйте программы, вносите в них изменения.

ЛАБОРАТОРНАЯ РАБОТА № 1

Тема: Знакомство со средой Visual Studio. Работа с базовыми компонентами.

Цели лабораторной работы: Настроить среду Visual Studio; использовать линейные, ветвящиеся, циклические алгоритмы решения задач; ознакомиться и использовать в приложениях компоненты label, button, textBox

Учебные вопросы:

1. Свойства компонентов label, button, textBox
2. Понятие обработчика событий
3. Функции преобразования текстовых и числовых данных
4. Метод MessageBox.Show()
5. Обработчик исключений try..catch

Задание 1. Программа «Приветствие»

Создадим приложение, окно которого показано на рис. 1

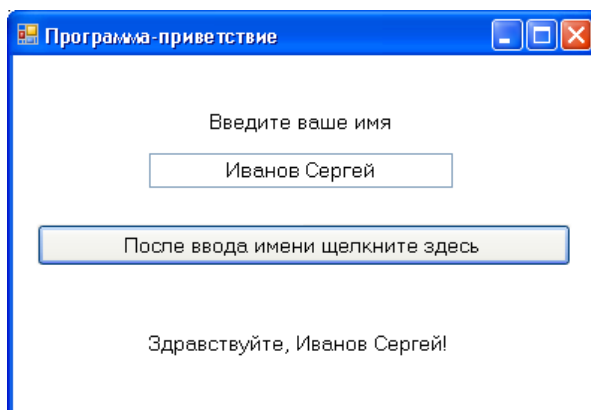


Рис. 1. Программа «Приветствие»

Для этого:

1. Запустите Microsoft Visual Studio, выберите в меню File->New->Project.

Мы хотим создать Windows- приложение, написанное на языке C#, поэтому в окне Project Types выберите Visual C# ->Windows, а в окне Templates – Windows Application.

Задайте имя первому приложению (например Проект1 или Приветствие), выберите папку, в которой будут храниться ваши работы (вашу личную папку). ОК.

2. На экране появится окно дизайнера формы с пустой формой. Начальные настройки Visual Studio 2005 таковы, что на экране нет двух важных окон: окна ToolBox с заготовками компонентов и окна Properties, в котором указываются значения свойств выбранных компонентов.

Выберите в главном меню команды View->ToolBox и View->Properties Window. В правом верхнем углу этих окон есть небольшая кнопка Auto Hide, которая позволит их расположить оптимальным лично для вас образом. Двойной щелчок по заголовкам окон позволит их зафиксировать, или напротив, перемещать по экрану (рис 2):

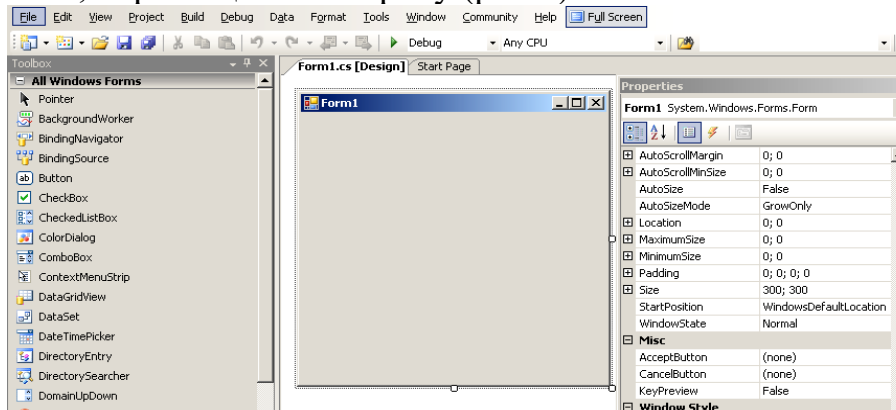


Рис. 2. Окна формы, палитры компонентов, свойств

Процесс создания приложений для Windows разбивается на две фазы: фазу проектирования формы и фазу кодирования обработчиков событий.

3. Спроектируем форму.

Проектирование формы заключается в переносе нужного компонента из окна Toolbox на форму и установке в окне Properties его некоторых свойств. Перенесите на форму компоненты label, button, textBox согласно рисунку 3 :

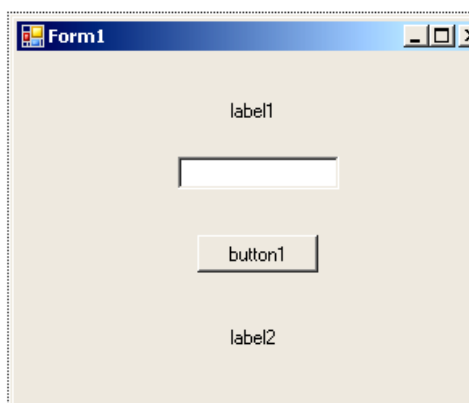


Рис. 3. Проектирование формы

В окне Properties задайте следующие свойства компонентам :

| Компонент | Значение | Пояснение |
|---------------|--------------------------|---|
| Form1 | | |
| Text | Программа-приветствие | Заголовок окна |
| FontSize | 10 | Высота шрифта для всех компонентов, помещенных на форму |
| BackColor | Любой понравившийся цвет | Цвет формы |
| StartPosition | CenterScreen | Положение формы при запуске программы |

| | | |
|-----------------|-------------------------------------|---|
| label1 | | |
| Location | 128; 46 | Положение |
| Text | Введите ваше имя: | Надпись |
| textBox1 | | |
| Location | 91; 65 | Положение |
| Size | 203; 23 | Размер |
| TextAlign | Center | Центрирование текста относительно границ компонента |
| button1 | | |
| Location | 16; 112 | Положение |
| Size | 357;28 | Размер |
| Text | После ввода имени щелкните здесь | Надпись на кнопке |
| label2 | | |
| Location | 87; 182 | Положение |
| TextAlign | MiddleCenter | Центрирование текста относительно границ компонента |

Такие свойства компонентов, как положение или размер, можно задать и непосредственно через форму, перемещая или растягивая компоненты.

4. Определим обработчики событий.

Если на этом этапе запустить программу, она создаст нужное окно, но никак не будет реагировать на щелчок по кнопке (можете попробовать). Чтобы добиться нужной реакции, следует создать обработчик события Click компонента button1. Для этого выделите кнопку на форме, в окне Properties щелкните по кнопке Events (события), дважды щелкните по событию Click. (или просто в окне формы дважды щелкните по кнопке) Возникнет окно с программной заготовкой (листинг 1.1):

Листинг 1.1

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace Приветствие
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void button1_Click(object sender, EventArgs e)
        {
        }
    }
}
```

```
}  
}
```

Отредактируйте его так, как показано в листинге 1.2 (комментарии по ходу листинга – для вас, в программу их можете не вносить). Будьте внимательны: идентификаторы, операторы, функции, ключевые слова C# чувствительны к регистру!

Листинг 1.2

```
using System;  
using System.Collections.Generic;  
using System.ComponentModel;  
using System.Data;  
using System.Drawing;  
using System.Text;  
using System.Windows.Forms;  
namespace Приветствие  
{  
    public partial class Form1 : Form  
    {  
        public Form1()  
        {  
            InitializeComponent();  
            // Удаляем из надписи Label2 ненужный текст  
            label2.Text = "";  
        }  
        //Обработчик щелчка по кнопке  
        private void button1_Click(object sender, EventArgs e)  
        {  
            // Если поле для ввода не пустое:  
            if (textBox1.Text != "")  
                label2.Text = "Здравствуйте, " + textBox1.Text + "!";  
        }  
    }  
}
```

Запустите приложение, сохраните, покажите преподавателю.

Примечание: Чтобы открыть ранее сохраненный проект, зайдите в пункт меню File->Open->Project/Solution. Откройте из вашей папки загрузочный файл проекта. Ехе файл по умолчанию создается в папке Bin\Debug.

Для открытия окна кода запустите файл Form1.cs. Для открытия формы воспользуйтесь пунктом меню View->Designer.

Задание 2. Программа «Сложение»

Создадим приложение, окно которого показано на рис. 4

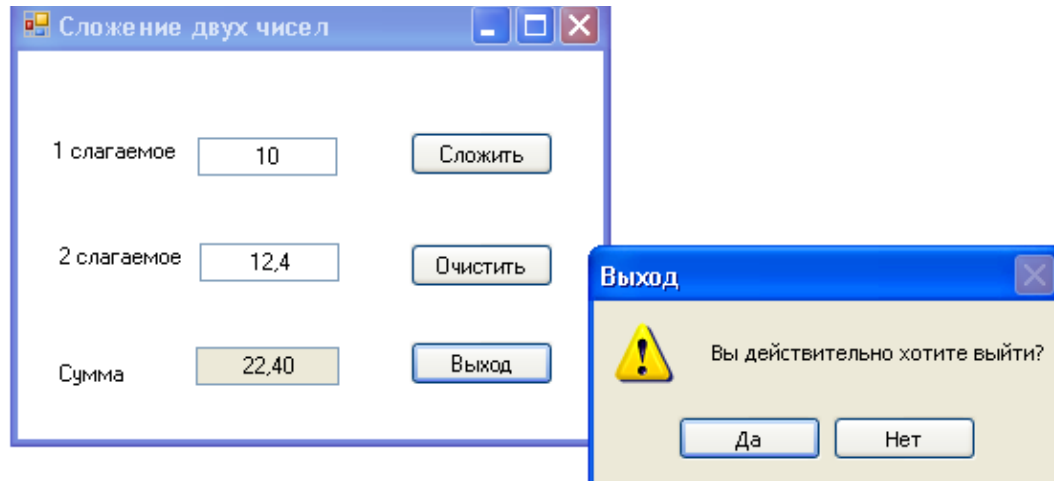


Рис. 4. Программа Сложение

1. File->New->Project. Выберите вашу папку, задайте имя новому проекту.
2. Разместите на форме необходимые компоненты, установите их начальные свойства.

У компонента textBox3 свойство ReadOnly поставьте в значение true (доступно только чтение информации, без возможности ее изменения)

3. Напишем обработчик события OnClick для кнопки «Сложить»:

```
private void button1_Click(object sender, EventArgs e)
{
    double k1, k2, k3; //Объявили локальные переменные
    k1 = Convert.ToDouble(textBox1.Text);
    k2 = Convert.ToDouble(textBox2.Text);
    k3 = k1 + k2;
    textBox3.Text = k3.ToString("N");
}
```

Функциями Convert.ToDouble() произвели преобразование текстовых значений в вещественные.

Функцией k3.ToString ("N") преобразовали вещественное значение переменной k3 в строковый тип (формат «N» используют для представления дробных чисел) или можно было бы так же воспользоваться функцией Convert.ToString(k3).

Ту же процедуру для кнопки «Сложить» можно было написать и без дополнительных локальных переменных, например так:

```
private void button1_Click(object sender, EventArgs e)
{
    textBox3.Text = (Convert.ToDouble(textBox1.Text) +
        Convert.ToDouble(textBox2.Text)).ToString("N");
}
```

```
}
```

2. Напишем обработчик события OnClick для кнопки «Очистить»:

```
private void button2_Click(object sender, EventArgs e)
{
    textBox1.Text = "";
    textBox2.Text = "";
    textBox3.Text = "";
    //установим курсор в поле для ввода первого слагаемого :
    textBox1.Focus();
}
```

3. Напишем обработчик события OnClick для кнопки «Выход»:

```
private void button3_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Вы действительно хотите выйти?", "Выход", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
    {
        this.Close();
    }
}
```

Проверьте работоспособность программы. Сохраните, но не закрывайте.

4. Добавим оператор try..catch, который позволит «отловить» возникающие программные сбои. Например, вследствие неверных исходных данных:

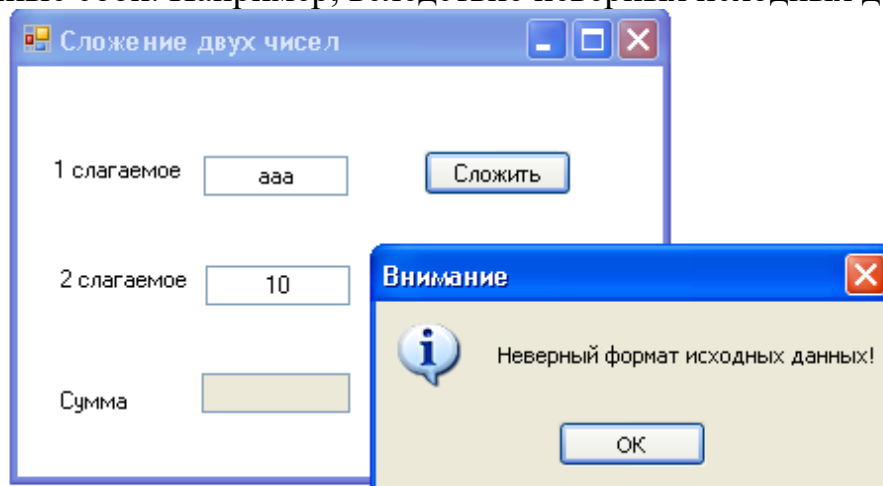


Рис. 5. Обработка исключений

```
private void button1_Click(object sender, EventArgs e)
{
    double k1, k2, k3;
    try // пробуем выполнить:
    {
        k1 = Convert.ToDouble(textBox1.Text);
        k2 = Convert.ToDouble(textBox2.Text);
        k3 = k1 + k2;
        textBox3.Text = k3.ToString("N");
    }
}
```

```

    }
    // если выполнить оказалось невозможно:
    catch
    { if ((textBox1.Text == "") || (textBox2.Text
== ""))
    MessageBox.Show("Введите данные!", "Внимание",
    MessageBoxButtons.OK, MessageBoxIcon.Information);
    else MessageBox.Show("Неверный формат
исходных данных!", "Внимание", MessageBoxButtons.OK,
    MessageBoxIcon.Information);
    }
}

```

Процедура `MessageBox.Show` формирует новое окно. Первый параметр используется в качестве текста внутри окна, второй – заголовок окна, третий (не обязательный) – формирует в окне кнопки, четвертый (не обязательный) – иконку окна. Окно с сообщением являются модальными (в терминологии Windows) – раз появившись на экране, оно блокирует работу пользователя с другими окнами вплоть до своего закрытия. Обычно с помощью модальных окон реализуется диалог, требующий от пользователя принятия некоторого решения. Использование окна `MessageBox` в условном операторе позволяет предусмотреть реакции программы на выбор в нем той или иной кнопки (см. процедуру для кнопки «Выход»)

Запустите приложение, попробуйте ввести некорректные исходные данные. Покажите преподавателю.

Задание 3. Программа «Квадратное уравнение»

Написать программу, решающую квадратное уравнение:

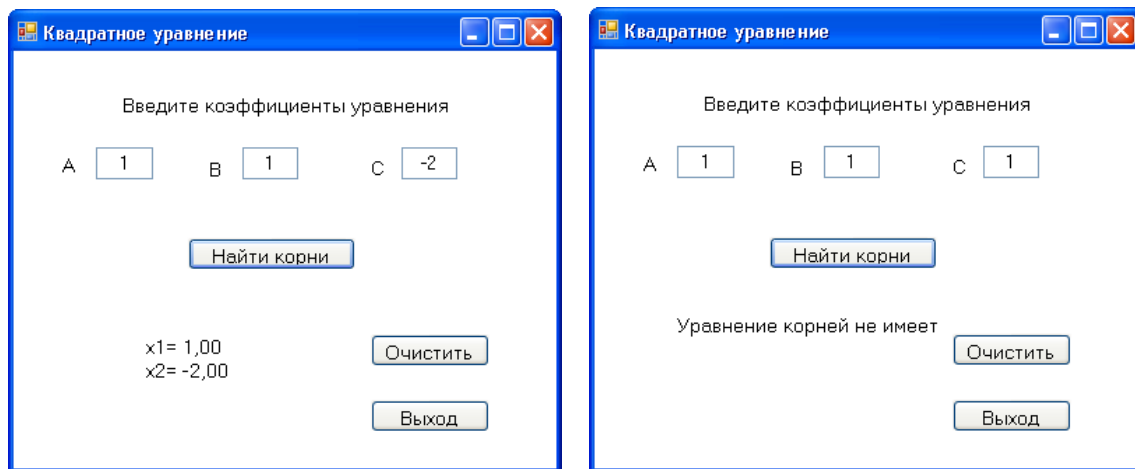


Рис. 6. Программа Квадратное уравнение

1. Разместите на форме 5 компонентов label (хотя на рисунке ответ занимает две строки, используем для его вывода один компонент label5) и 3 кнопки. Впишите обработчики событий согласно листингу 3 :

Листинг 3

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace Квадратное_уравнение
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            label5.Text = "";
        }

        private void button1_Click(object sender, EventArgs e)
        {
            double a, b, c, d, x1, x2;
            try
            {
                a = Convert.ToDouble(textBox1.Text);
                b = Convert.ToDouble(textBox2.Text);
                c = Convert.ToDouble(textBox3.Text);
                d = b * b - 4 * a * c;
                if (d >= 0)
                {
                    x1 = (-b + Math.Sqrt(d)) / (2 * a);
                    x2 = (-b - Math.Sqrt(d)) / (2 * a);
                    label5.Text = "x1= " + x1.ToString("N") + "\n"
+ "x2= " + x2.ToString("N");
                }
                else label5.Text = "Уравнение корней не имеет";
            }
            catch { MessageBox.Show("Проверьте корректность
исходных данных", "Внимание!"); }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            textBox1.Text = "";
            textBox2.Text = "";
            textBox3.Text = "";
            textBox1.Focus();
            label5.Text = "";
        }
    }
}
```

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

Обратите внимание, при использовании математических функций (методов класса Math), необходимо указать имя класса - например Math.Sqrt()

Сохраните. Покажите преподавателю.

Задание 4. Программа «Касса»

Напишем программу, которая позволит переключаться по полям с исходными данными не только по щелчку мыши, но и при нажатии клавиши Enter. Обработка результатов также будет происходить не через специально отведенную кнопку (событие OnClick), а при нажатии Enter в последнем поле входных данных (событие OnKeyPress).

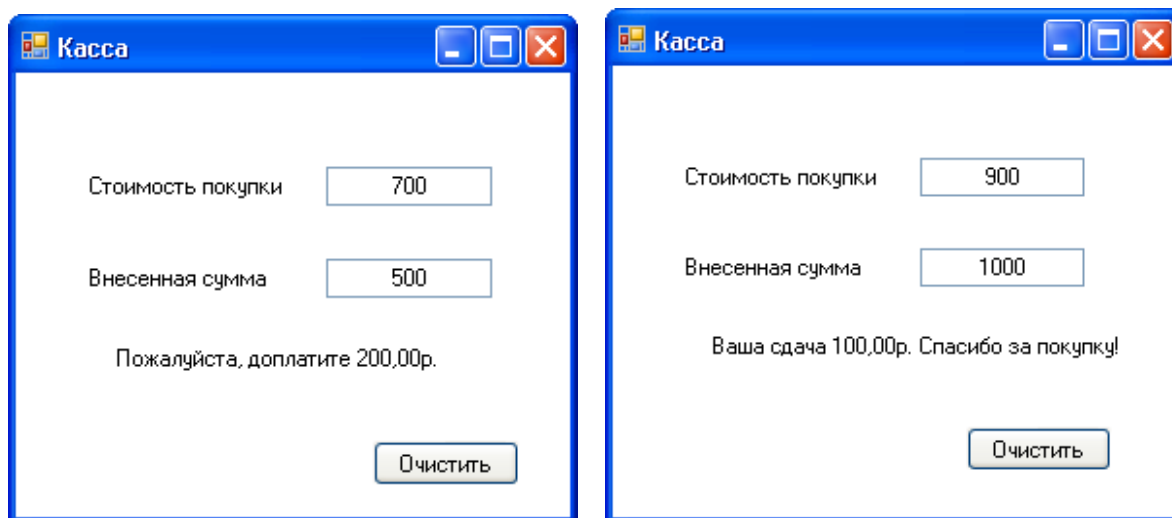


Рис. 7. Программа Касса

Расположите на форме кнопку, три компонента label и двы поля textBox. Задайте надписи компонентам согласно рисунку 7, свойство text компонента label3 оставьте пустым.

/* Вызываем через вкладку Events обработчик события OnKeyPress для поля textBox1 : */

```
private void textBox1_KeyPress (object sender,
KeyPressEventArgs e)
{
    // Клавиша Enter имеет номер 13
    if (e.KeyChar.Equals((char)13)) textBox2.Focus();
}
```

```

/* Вызываем через вкладку Events обработчик события OnKeyPress для
поля textBox2 : */
private void textBox2_KeyPress(object sender,
                               KeyPressEventArgs e)
{
    double a,b,c;
    if (e.KeyChar.Equals((char)13))
    {
        try
        {
            a = Convert.ToDouble(textBox1.Text);
            b = Convert.ToDouble(textBox2.Text);
            c = Math.Abs(a - b);
            if (a > b) label3.Text = "Пожалуйста, доплатите " + c.ToString("C");
            if (a < b) label3.Text = "Ваша сдача " + c.ToString("C")+" Спасибо за покупку!";
            if (a == b) label3.Text = "Спасибо за покупку";
        }
        catch
        {
            MessageBox.Show("Проверьте правильность входных данных", "Внимание!");
        }
    }
}

// Для кнопки «Очистить» :
private void button1_Click(object sender,EventArgs e)
{
    textBox1.Text = "";
    textBox2.Text = "";
    label3.Text = "";
    textBox1.Focus();
}

```

Обратите внимание, ответы выводятся в денежном формате (“С”). Сохраните. Покажите преподавателю.

Самостоятельные задания

1.1. Написать программу «Треугольник», рассчитывающую по координатам трех точек периметр и площадь некоторого треугольника.

1.2. Написать программу «Электроэнергия», которая бы рассчитывала сумму платежа за электричество. В качестве входных данных использовать предыдущие показания счетчика, текущие показания счетчика, стоимость 1 кВт.

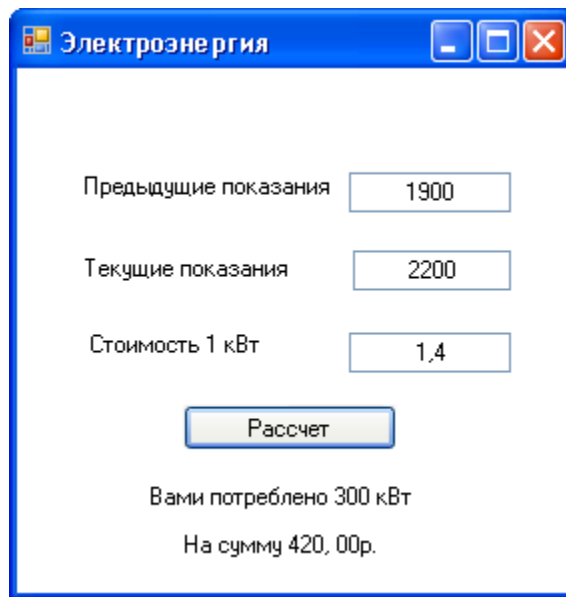


Рис. 8. Окно программы Электроэнергия

1.3. Написать программу, рассчитывающую факториал числа n:

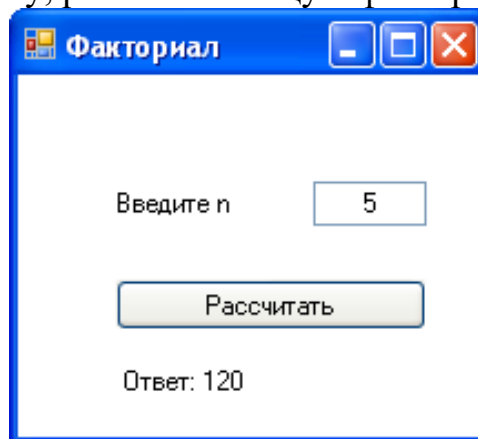


Рис. 9. Окно программы Факториал

1.4. Написать программу-тест, которая выводит на экран 2 случайных числа из интервала (-10,10), просит указать их произведение и выдает результат- верно/неверно.

Примечание: для генерации случайных чисел использовать класс Random.

Например: `Random n = new Random();`
`zadumano = n.Next(100);`

- случайное число из интервала [0,100)

1.5. Написать программу, подсчитывающую, сколько положительных и отрицательных чисел пользователь ввел с клавиатуры. Ввод прекращается, когда вносится ноль.

ЛАБОРАТОРНАЯ РАБОТА № 2

Тема: Работа со списками, переключателями, таймером

Цели лабораторной работы: ознакомиться и использовать в приложениях компоненты `comboBox`, `numericUpDown`, `checkBox`, `radioButton`, `timer`

Учебные вопросы:

1. Свойства компонентов `comboBox` и `numericUpDown`;
2. Свойства компонентов `checkBox`, `radioButton`
3. Работа с оператором выбора `switch`
4. Работа с таймером

Задание 5. Программа «Автозаправка»

Напишем программу, демонстрирующую работу компонентов `comboBox` и `numericUpDown`:

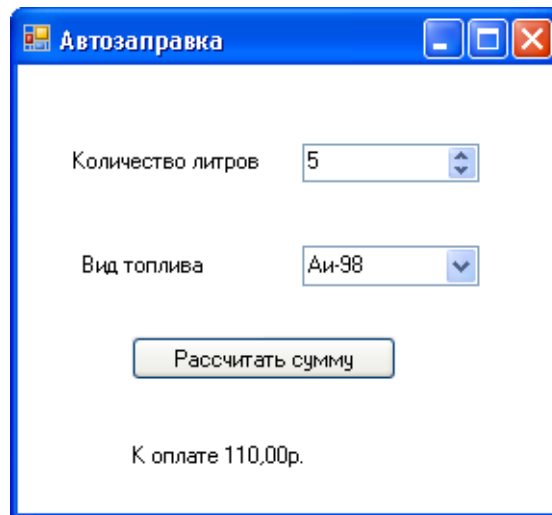


Рис. 10. Окно программы Автозаправка

1. Расположите на форме три компонента `label`, одну кнопку, один компонент `numericUpDown`, один список `comboBox` (согласно рис 10). У компонента `label3` уберите надпись в свойстве `Text`.

2. Заполнить список `comboBox` фиксированными значениями можно на этапе проектирования формы (так и поступим), а можно и непосредственно в программе.

Зайдите в свойство `Items` компонента `comboBox`, построчно заполните открывшееся окно всеми возможными значениями:

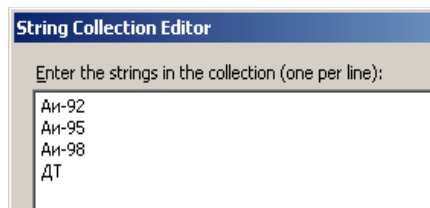


Рис.11. String Collection Editor

3. Пишем обработчик события OnClick для кнопки «Рассчитать сумму»:

```
private void button1_Click(object sender, EventArgs e)
{    //Переменную с (цена за 1 литр)необходимо
инициализировать
    double c=0, sum;
    // Смотрим, какое по счету значение в списке нами выбрано:
    switch (comboBox1.SelectedIndex)
    { case 0: c = 18.9; break;
      case 1: c = 20.3; break;
      case 2: c = 22; break;
      case 3: c = 17.6; break;
    }
    sum = Convert.ToDouble(numericUpDown1.Value) * c;
    label3.Text = "К оплате " + sum.ToString("C");
}
```

Обратите внимание, нумерация элементов списка начинается с нуля.
Запускаем, сохраняем в личной папке, показываем преподавателю.

Задание 6. Программа «Кафе»

Напишем программу, демонстрирующую работу компонентов checkbox [2].

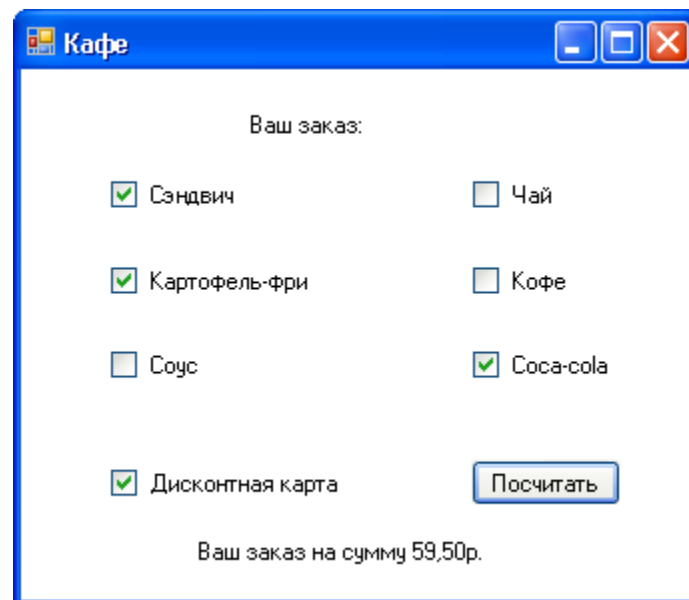


Рис. 12. Окно программы Кафе

1. Расположите на форме компоненты согласно рис 12.
2. Для формирования надписей компонентов checkBox воспользуйтесь их свойством Text. У компонента label2 уберите надпись в свойстве Text.

3. Пишем обработчик события OnClick для кнопки «Посчитать»:

```
private void button1_Click(object sender, EventArgs e)
{
    double sum = 0;
    if (checkBox1.Checked) sum += 35;
    if (checkBox2.Checked) sum += 15;
    if (checkBox3.Checked) sum += 9;
    if (checkBox4.Checked) sum += 15;
    if (checkBox5.Checked) sum += 25;
    if (checkBox6.Checked) sum += 20;
    // Проверяем наличие скидки в 15 %
    if (checkBox7.Checked) sum *= 0.85;
    abel2.Text = "Ваш заказ на сумму "+sum.ToString("C");
}
```

Проверьте работоспособность приложения. Сохраните. Покажите преподавателю.

Задание 7. Программа «Арифметические действия»

Напишем программу, демонстрирующую работу компонентов radioButton:

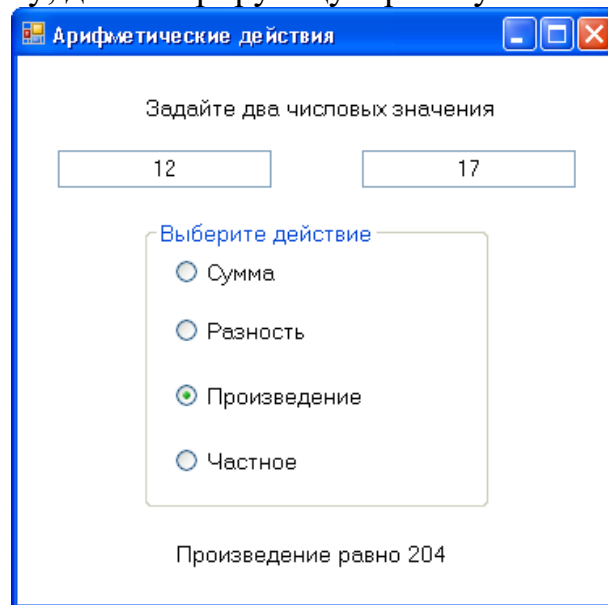


Рис. 13. Программа Арифметические действия

1. Разместите на форме компоненты согласно рис 13. (два textBox, два label, четыре radioButton, один groupBox)

Компонент radioButton предназначен для выбора одного из взаимоисключающих решений. Отдельные переключатели объединяются в группы (например с помощью компонента groupBox), в группе в каждый момент может быть «включен» только один переключатель.

2. Задайте начальные свойства компонентам (компоненту GroupBox через свойство text дайте название «Выберите действие», назовите компоненты radioButton и т.д).

Переходим в окно кода:

Листинг 7

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace Арифметические_действия
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            label2.Text="";
        }

        private void radioButton1_Click(object sender,EventArgs e)
        {
            label2.Text = "Сумма равна " +
            Convert.ToString(Convert.ToDouble(textBox1.Text) +
            Convert.ToDouble(textBox2.Text));
        }
        private void radioButton2_Click(object sender,EventArgs e)
        {
            label2.Text = "Разность равна " +
            Convert.ToString(Convert.ToDouble(textBox1.Text) -
            Convert.ToDouble(textBox2.Text));
        }
        private void radioButton3_Click(object sender,EventArgs e)
        {
            label2.Text = "Произведение равно " +
            Convert.ToString(Convert.ToDouble(textBox1.Text) *
            Convert.ToDouble(textBox2.Text));
        }
        private void radioButton4_Click(object sender,EventArgs e)
        {
            if (Convert.ToDouble(textBox2.Text) != 0)
                label2.Text = "Частное равно " +
                (Convert.ToDouble(textBox1.Text) /
                Convert.ToDouble(textBox2.Text)).ToString("N");
            else label2.Text = "Попытка деления на ноль!";
        }
    }
}
```

Запускаем, показываем преподавателю.

Задание 8. Программа «Конвертер валют»

Напишем программу, демонстрирующую работу компонентов radioButton и comboBox:

Рис. 14. Программа Конвертер валют

1. Разместите на форме необходимые компоненты согласно рис 14. (2 кнопки, 3 поля textBox, 3 radioButton, 1 comboBox, 5 label).
2. Компоненты Radiobutton сгруппируйте панелью groupBox.
3. Через свойство Items компонента comboBox задайте значения в списке подстановок.
4. Напишите обработчики событий OnClick для кнопок «Рассчитать» и «Выход», задайте начальные значения полям в соответствии с листингом:

Листинг 8

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace Конвертер_валют
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            textBox1.Text = "25,84";
            textBox2.Text = "34,85";
            label5.Text = "";
        }
    }
}
```

```

private void button1_Click(object sender, EventArgs e)
{
    double kursD, kursE, sum, result=0;
    kursD = Convert.ToDouble(textBox1.Text);
    kursE = Convert.ToDouble(textBox2.Text);
    sum = Convert.ToDouble(textBox3.Text);
    // Смотрим, из какой валюты переводим
    switch (comboBox1.SelectedIndex)
    {
        case 0: //Если переводим из рублей в другие валюты:
        {
            if (radioButton1.Checked) //RUS->RUS
            {
                result = sum;
                label5.Text=textBox3.Text+" RUS->"
                    +result.ToString ("N")+" RUS";
            }
            if (radioButton2.Checked) // RUS->USD
            {
                result = sum / kursD;
                label5.Text = textBox3.Text+" RUS->"
                    +result.ToString("N")+" USD";
            }
            if (radioButton3.Checked) // RUS->EUR
            {
                result = sum / kursE;
                label5.Text = textBox3.Text+" RUS->"
                    +result.ToString("N")+" EUR";
            }
        }
        break;
        case 1: //Если переводим из долларов в другие валюты:
        {
            if (radioButton1.Checked) // USD->RUS
            {
                result = sum * kursD;
                label5.Text=textBox3.Text+" USD->"
                    +result.ToString("N")+" RUS";
            }
            if (radioButton2.Checked) // USD->USD
            {
                result = sum;
                label5.Text=textBox3.Text+" USD->"
                    +result.ToString("N")+" USD";
            }
            if (radioButton3.Checked) //USD->EUR
            {
                result = (kursD / kursE) * sum;
                label5.Text=textBox3.Text+" USD->"
                    +result.ToString("N")+" EUR";
            }
        }
        break;
    }
}

```

```

case 2: //Если переводим из евро в другие валюты:
{
    if (radioButton1.Checked) // EUR->RUS
    {
        result = sum * kursE;
        label5.Text=textBox3.Text+" EUR->"
            +result.ToString("N")+" RUB";
    }
    if (radioButton2.Checked) // EUR->USD
    {
        result = (kursE / kursD) * sum;
        label5.Text=textBox3.Text+" EUR->"
            +result.ToString("N")+" USD";
    }
    if (radioButton3.Checked) //EUR->EUR
    {
        result = sum;
        label5.Tex=textBox3.Text+" EUR->"
            +result.ToString("N")+" EUR";
    }
} break;
} // Закрыли switch
} // Закрыли функцию button1_Click
private void button2_Click(object sender,EventArgs e)
{
    this.Close();
}
}
}

```

Задание 9. Программа « Секундомер »

Создадим приложение, демонстрирующее использование компонента timer [2]. Для него определено всего одно событие- Tick, происходящее каждый заданный интервал времени.

Пусть при нажатии кнопки «Пуск» запускается отчет времени, одновременно с этим становится недоступной кнопка «Сброс», а кнопка «Пуск» переименовывается в «Стоп». При остановке секундомера «Стоп» переименовывается обратно в «Пуск», становится доступным «Сброс». Форма программы приведена на рис 15:

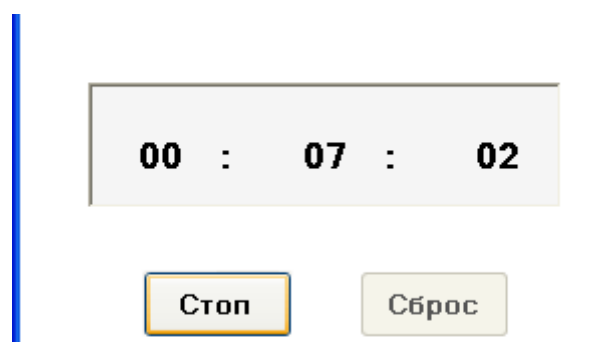


Рис. 15. Окно программы Секундомер

1. Расположите на форме два компонента button, 5 компонентов label (в примере label1, label2, label3- минуты, секунды и миллисекунды; label 4, label5 – разделители)
2. Объедините компоненты label панелью (компонент panel). Свойство панели Font->Size измените на 14 (чтобы цифры были более крупными). Можете использовать жирный шрифт.
3. Перенесите на форму компонент timer. Непосредственно на форме он не установится, его значок появится в левом нижнем углу окна Form1.
4. Объявляем переменные, пишем обработчики событий согласно листингу:

Листинг 9

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace Секундомер
{
    public partial class Form1 : Form
    {
        //минуты, секунды, миллисекунды
        int min = 0, sec = 0, msec = 0;
        public Form1()
        {
            InitializeComponent();
            label1.Text = "00";
            label2.Text = "00";
            label3.Text = "00";
            label4.Text = ":";
            label5.Text = ":";
            // Период генерации события Tick в миллисекундах:
            timer1.Interval = 10;
            // (при необходимости перенастройте его)
        }

        // Кнопка Пуск / Стоп
        private void button1_Click(object sender, EventArgs e)
        {
            if (timer1.Enabled) // Если таймер активен
            {timer1.Enabled = false; // Останавливаем таймер
```

```

        // Переименовываем кнопку
        button1.Text = "Пуск";
        //Делаем активной кнопку Сброс
        button2.Enabled = true;
    }
else // Если таймер отключен
{ // делаем компонент активным
    timer1.Enabled = true;
    // Переименовываем кнопку
    button1.Text = "Стоп";
    //делаем недоступной кнопку Сброс
    button2.Enabled = false;
}
}

// для кнопки Сброс
private void button2_Click(object sender, EventArgs e)
{
    min = 0; sec = 0; msec = 0;
    label1.Text = "00";
    label2.Text = "00";
    label3.Text = "00";
}

// Событие Tick компонента timer:
private void timer1_Tick(object sender, EventArgs e)
{
    /* Если доходим до 99-ой миллисекунды - обнуляем ее
    значение, увеличиваем количество секунд на единицу.
    Аналогично с секундами и минутами */

    if (msec == 99)
    {
        if (sec == 59)
        {
            if (min == 59) min = 0;
            else min++;
            sec = 0;
        }
        else sec++;
        msec = 0;
    }
    else msec++;
    // Форматируем надписи табло:
    if (min.ToString().Length == 1)

```



```

        label1.Text="0"+min.ToString();
else label1.Text = min.ToString();
if (sec.ToString().Length == 1)
    label2.Text="0"+sec.ToString();
else label2.Text = sec.ToString();
if (msec.ToString().Length == 1)
    label3.Text="0"+msec.ToString();
else label3.Text = msec.ToString();
// Мигание разделителей ":"
if (msec == 1)
    { label4.Text = ":"; label5.Text = ":"; }
if (msec == 50)
    { label4.Text = ""; label5.Text = ""; }
    }
}
}

```

Сохраните. Запустите. Покажите преподавателю.

Самостоятельные задания

2.1. Написать программу, рассчитывающую стоимость покрытия на пол. Задается требуемая длина и ширина, из списка выбирается тип покрытия (ковролин, линолеум, паркет), указывается, требуется ли доставка, рассчитывается сумма предполагаемых затрат. Стоимость 1 м² каждого вида материала, стоимость доставки задаются программно:

Рис. 16. Программа Стоимость покрытия

2.2. Написать программу просмотра изображений:

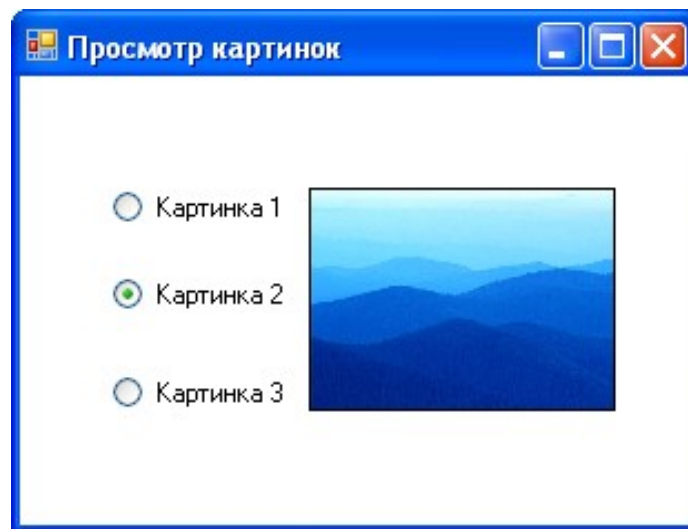


Рис. 17. Программа Просмотр картинок

Примечание: поместите на форму три компонента `pictureBox`, через свойство `Image` загрузите в них картинки. Используя свойство `Visible` (`true`, `false`) компонентов `pictureBox`, реализуйте возможность последовательного просмотра изображений.

2.3. Написать программу «Обратный отчет», периодически выводящую на форму цифры от 10 до 0. Когда отчет дойдет до нуля, программа должна выдать сообщение «Пуск!» и закончить свою работу.

ЛАБОРАТОРНАЯ РАБОТА № 3

Тема: Создание калькулятора. Разработка программы-теста. Работа с меню и несколькими формами.

Цели лабораторной работы: ознакомиться и использовать в приложениях компоненты `tabControl` `trackBar`, `menuStrip` `pictureBox`, `toolTip`; создать приложения, состоящие из нескольких форм.

Учебные вопросы:

1. Создание приложения «Калькулятор»;
2. Обзор свойств компонентов `tabControl` `trackBar`, `menuStrip` `pictureBox`, `toolTip`
3. Создание приложения «Тест».
4. Использование в программах главного меню
5. Создание приложений с несколькими формами.

Задание 10. Программа « Калькулятор »

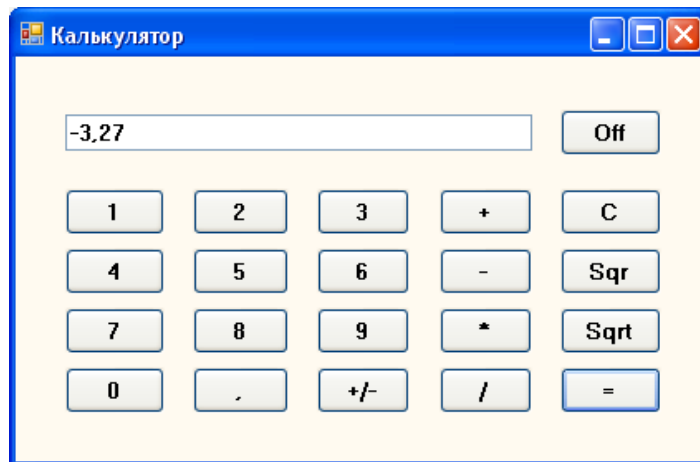


Рис. 18. Окно программы Калькулятор

Используем простую схему работы калькулятора: вводим операнд1, затем оператор, вводим операнд2 и получаем результат после нажатия клавиши «=». При вводе первого числа все вводимые цифры сохраняем в переменной `chislo1`. При вводе оператора очищаем поле табло и запоминаем вид операции в переменной `operator_` (В C# слово `operator` является служебным, поэтому в имя нашей переменной добавим символ подчеркивания). Далее вводим цифры второго числа, запоминаем их в переменной `chislo2`. Кроме того, понадобится логическая переменная `flag`, которая поможет различить операнды.

1. Расположите на форме компонент `textBox`. Поместите 21 кнопку, задайте для каждой названия (для удобства постарайтесь, чтобы номера кнопок совпадали с изображенными на них цифрами)
2. Впишем для каждой кнопки обработчики событий `OnClick`, предварительно описав все переменные:

```

. . .
public partial class Form1 : Form
{
    double chislo1=0,chislo2=0;
    bool flag;
    char operator_;
    public Form1()
    {
        InitializeComponent();
    }
. . .

// Для кнопки «1» :
private void button1_Click(object sender,EventArgs e)
{
    if (flag==true) textBox1.Text="0";
    flag=false;
    if (textBox1.Text!="0")
        textBox1.Text=textBox1.Text+"1";
    else textBox1.Text="1";
}
// Для кнопки «2» :
private void button2_Click(object sender,EventArgs e)
{
    if (flag==true) textBox1.Text="0";
    flag=false;
    if (textBox1.Text!="0")
        textBox1.Text=textBox1.Text+"2";
    else textBox1.Text="2";
}

```

Для кнопок «3»-«0» тексты процедур аналогичны, впишите их самостоятельно.

```

// Для кнопки « = » :
private void button21_Click(object sender,EventArgs e)
{
    if (flag==false)
        chislo2=Convert.ToDouble(textBox1.Text);
    else chislo2=chislo1;
    switch (operator_)
    {
        case '+' :
            textBox1.Text=Convert.ToString(chislo1+chislo2);break;

```

```

        case '-' :
            textBox1.Text=Convert.ToString(chislo1-
chislo2);break;
        case '*' :
            textBox1.Text=Convert.ToString(chislo1*chislo2);break;
        case '/' : if(chislo2!=0)
            textBox1.Text=Convert.ToString(chislo1/chislo2);
            else textBox1.Text="На ноль делить нельзя!";
break;
    }
    flag=true;
}

```

Обратите внимание, значения переменных типа char записываются в апострофах, а не в двойных кавычках, как у переменных строкового типа.

```

// Для кнопки «+» :
private void button15_Click(object sender,EventArgs e)
{
    chislo1=Convert.ToDouble(textBox1.Text);
    flag=true;
    operator_='+';
}

```

Для кнопок «-», «*», «/» тексты процедур аналогичны

```

// Для кнопки « C »
private void button14_Click(object sender,EventArgs e)
{
    textBox1.Text="";
    flag=false;
    chislo1=0;
    chislo2=0;
}

```

```

// Для кнопки « +/- »
private void button12_Click(object sender,EventArgs e)
{
    textBox1.Text=Convert.ToString(Convert.ToDouble(textBox1.Text)*(-1));
}

```

```

// Для кнопки « , »
private void button11_Click(object sender,EventArgs e)
{
    if (textBox1.Text.IndexOf(",")== -1)

```

```

        textBox1.Text = textBox1.Text + ",";
    }

    // Для кнопки « Off »
    private void button13_Click(object sender, EventArgs e)
    {
        this.Close();
    }

```

Тексты процедур для нахождения корня и квадрата чисел вписать самостоятельно.

Показываем преподавателю.

Задание 11. Программа «Тест»

Создадим приложение, использующее компонент tabControl:

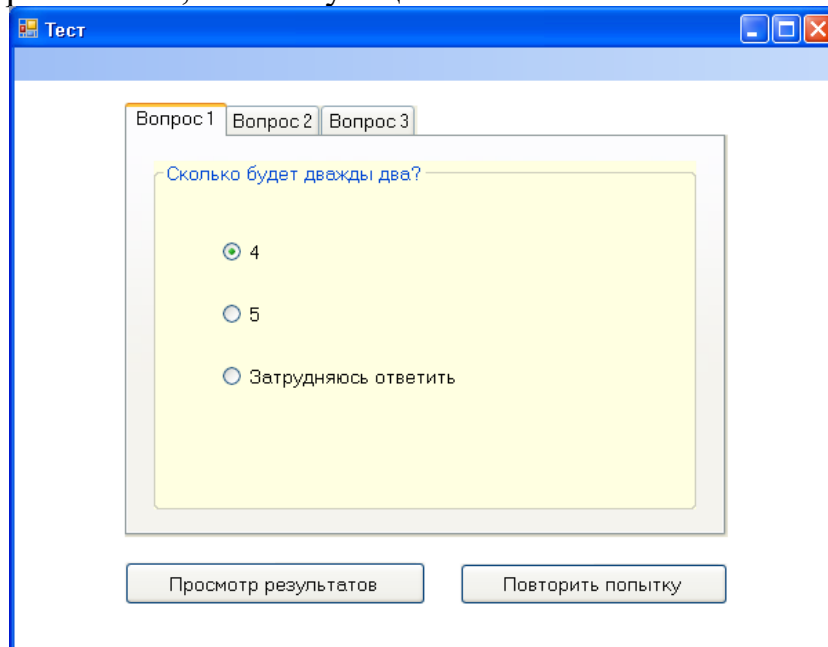


Рис. 19. Окно программы Тест

1. Расположите на форме компонент tabControl и два компонента button (рис 19).
2. Используя контекстное меню компонента tabControl, создайте 3 страницы (пункт Add Tab). Задайте названия страниц (св-во Text: Вопрос1, Вопрос2, Вопрос3), Будьте внимательны, следите за тем, какой компонент у вас выделен в данный момент.
3. На каждую из закладок поместите по 3 компонента radioButton. Задайте им надписи в виде ответов на вопросы.
4. На всех закладках группы из трех переключателей объедините компонентами GroupBox, в их свойствах Text введите формулировки вопросов (любых)

Ниже приведены обработчики событий OnClick для кнопок «Просмотр результатов» и «Повторить попытку» :

```
private void button1_Click(object sender,EventArgs e)
{
    byte k=0;
    // Делаем набор закладок невидимым
    tabControl1.Visible = false;
    /* Впишите свои номера кнопок, соответствующих
    правильным ответам. В моем примере это кнопки 1,5 и
    7*/

    if (radioButton1.Checked) k++;
    if (radioButton5.Checked) k++;
    if (radioButton7.Checked) k++;
    if (k==3) MessageBox.Show("Вы верно ответили
на все вопросы!", "Результат:");
    else MessageBox.Show("Вы ответили верно на "+
Convert.ToString(k)+" из трех вопросов", "Результат:");
}
private void button2_Click(object sender,EventArgs e)
{
    tabControl1.Visible = true;
}
```

Запускаем, показываем преподавателю.

Задание 12. Работа с несколькими формами

Напишем небольшое приложение, демонстрирующее работу с несколькими формами в рамках одного проекта.

На главной форме (Form1) разместим две кнопки:

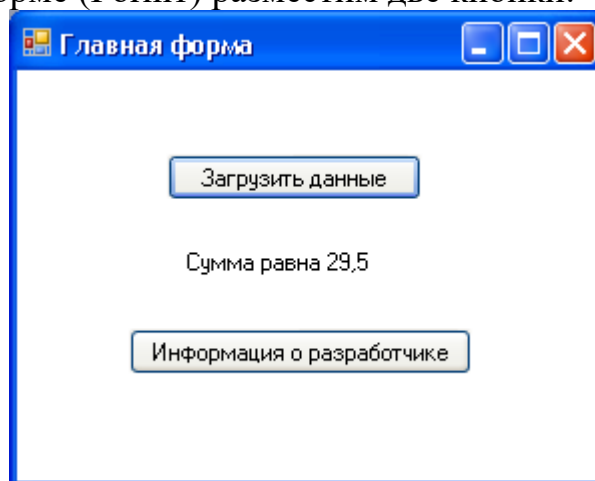


Рис. 20. Главная форма

При щелчке по кнопке «Загрузить данные» должна открыться форма 2 с приглашением к вводу слагаемых:

Рис. 21. Форма для ввода данных

После ввода двух чисел и щелчка по кнопке «Вернуться на главную форму» на форме 1 должен появиться результат сложения этих чисел. При щелчке по кнопке «Информация о разработчике» должна открыться форма3:

Рис. 22. Форма для просмотра информации

1. Назовите форму 1 «Главная форма» (через свойство Text). Разместите на ней две кнопки и один компонент label. Назовите кнопки в соответствии с рис. 20, свойство Text у компонента label1 пока оставьте пустым.
2. Зайдите в пункт меню Project / Add Windows Form. Откроется окно «Add New Item». Выберите Windows Form, оставьте предлагаемое имя Form2.cs. Щелкните Add.
3. Сделайте заголовок для новой формы- «Ввод данных» (свойство Text); поместите на неё 2 поля для ввода, 2 надписи и одну кнопку (рис. 21)
4. По аналогии создайте форму 3. Сделайте заголовок для новой формы- «Информация о разработчике». Поместите на нее компонент label и кнопку (рис. 22)

5. Пишем обработчики событий.

5.1 Для главной формы (Form1) :

```
// Кнопка «Загрузить данные»
private void button1_Click(object sender, EventArgs e)
{
```



```

        // Создаем объект f2, имеющий тип Form2:
        Form2 f2 = new Form2();
        f2.ShowDialog(); //Показываем форму 2
        label1.Text="Сумма равна "
                    +Convert.ToString(f2.a + f2.b);
    }
    // Кнопка «Информация о разработчике»
    private void button2_Click(object sender,EventArgs e)
    {
        Form3 f3 = new Form3();
        f3.ShowDialog(); //Показываем форму 3
    }

```

5.2 Для формы «Ввод данных» (Form2):

```

    // Кнопка «Вернуться к главной форме»
    private void button1_Click(object sender,EventArgs e)
    {
        this.a = Convert.ToDouble(textBox1.Text);
        this.b = Convert.ToDouble(textBox2.Text);
        this.Close();
    }

```

Переменные a и b предварительно описываем с пометкой public:

```

...
public partial class Form2 : Form
{
    public double a, b;
    public Form2()
    { InitializeComponent();
    ...
}

```

5.3 Для формы «Информация о разработчике» (Form3):

Как только форма 3 активизируется, через ее компонент label должно быть передано сообщение.

```

    // Обработчик события Activated для формы:
    private void Form3_Activated(object sender,EventArgs e)
    {
        this.label1.Text="Приложение написано студентом
        потока ИС \Ивановым Павлом Олеговичем, 2007";
    }

```

(для вызова обработчика выделяем форму, заходим на вкладку Events окна Properties, щелкаем по событию Activated)

```

    // Кнопка «Вернуться к главной форме»
    private void button1_Click(object sender,EventArgs e)
    {
        this.Close();
    }

```

Запускаем, показываем преподавателю.

Задание 13. Работа с меню

Разработаем приложение, демонстрирующее работу с компонентом menuStrip.

1. Создайте три формы по аналогии с предыдущим примером. Форму 1 назовите «Главная форма», форму 2- «Ввод данных», форму 3- «Информация о разработчике»
2. На форму 1 поместите компонент menuStrip и компонент label (рис 23)

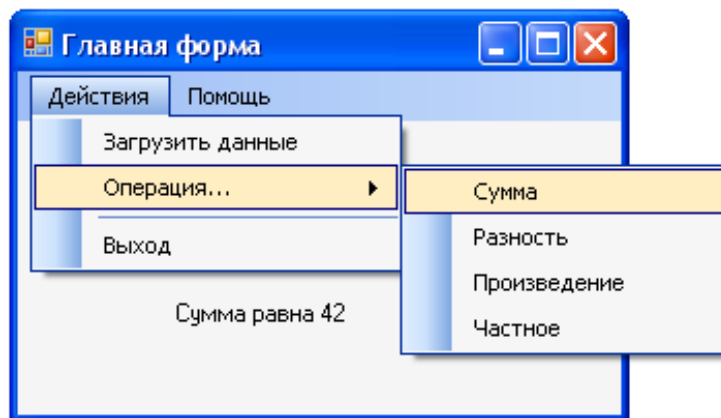


Рис. 23. Главная форма с меню

3. На форму 2 поместите два компонента label, два поля для ввода, одну кнопку (рис 24)

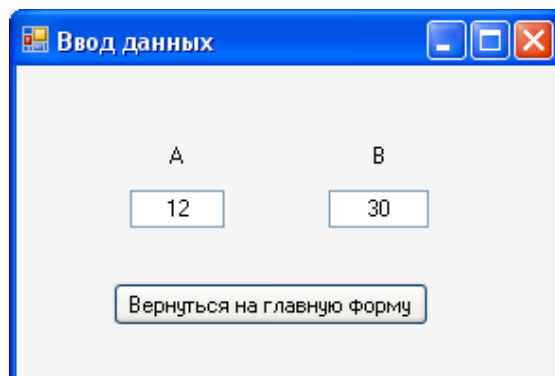


Рис. 24. Форма ввода данных

4. На форму 3 поместите один компонент label и одну кнопку (рис 25)

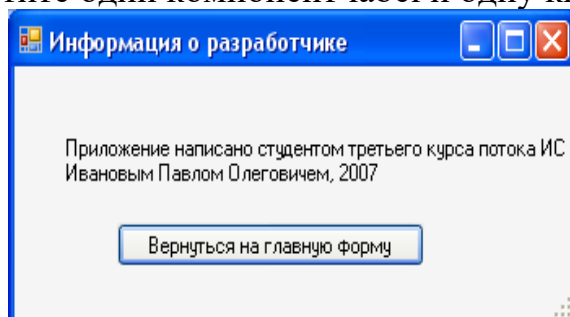


Рис. 25. Форма с информацией о разработчике

Зададим пункты меню на главной форме:

5. Встаньте мышкой на компонент `menuStrip`, на месте надписи `TypeHere` введите «Действия». Сместите курсор вниз, следующий пункт меню назовите «Загрузить данные», еще ниже создайте пункт «Операция...». Пусть над данными будут допустимы операции сложения, вычитания, умножения и деления. Щелкаем справа от пункта «Операция...», создаем пункт «Сумма», под суммой создаем пункты «Разность», «Произведение», «Частное».

В левой колонке, под пунктом «Операция...», подведите курсор к надписи `TypeHere` и щелкните по изображению перевернутого треугольника. Раскроется список, выберите в нем пункт `Separator` (это позволит отделить пункты меню горизонтальной чертой). Создайте последний пункт в левой колонке «Выход». Сверьтесь с рисунком 23.

Щелкните справа от пункта «Действия», введите название второй колонки – «Помощь». Создайте в ней один пункт- «Информация о разработчике».

6. Пишем обработчики событий для формы 1:

Пусть при запуске приложения, пока не введены никакие данные, пункт меню «Операция» будет недоступен (свойство `Enabled` в значении `false`). Для этого вызовем обработчик события `Activated` для формы 1 (встаньте мышкой на любое свободное место формы, на вкладке `Events` окна `Properties` щелкните по событию `Activated`), впишите:

```
private void Form1_Activated(object sender, EventArgs e)
{
    операцияToolStripMenuItem.Enabled = false;
}
```

Постарайтесь не запутаться в длинных именах пунктов меню (в крайности любой пункт можно переименовать через свойство `Name` окна `Properties`)

Дважды щелкаем по пункту «Загрузить данные», вызывая обработчик события `OnClick` для этого пункта:

```
private void загрузитьДанныеToolStripMenuItem_Click
(object sender, EventArgs e)
{
    Form2 f2 = new Form2();
    f2.ShowDialog(); //Показываем форму 2
    k1 = f2.a;
    k2 = f2.b;
    //После внесения данных делаем доступным пункт Операция:
    операцияToolStripMenuItem.Enabled = true;
}
```

Здесь `f2` – объект, имеющий тип `Form2`, в переменные `a` и `b` будут занесены значения входных данных с формы 2, переменные `k1` и `k2` предварительно опишите:

```
...
public partial class Form1 : Form
```

```

        {   public double k1, k2;
            public Form1 ()
        . . .
Обработчики для пунктов Сумма, Разность, Произведение, Частное,
Выход:
private void суммаToolStripMenuItem_Click
                (object sender, EventArgs e)
{label1.Text= "Сумма равна "+Convert.ToString(k1+k2);
}
private void разностьToolStripMenuItem_Click
                (object sender, EventArgs e)
{label1.Text="Разность равна"+Convert.ToString(k1-
k2);}
private void произведениеToolStripMenuItem_Click
                (object sender, EventArgs e)
{ label1.Text = "Произведение равно"+
                Convert.ToString(k1 * k2);
}
private void частноеToolStripMenuItem_Click
                (object sender, EventArgs e)
{ if (k2 != 0)
label1.Text ="Частное равно"+Convert.ToString(k1/k2);
else label1.Text = "На ноль делить нельзя!";
}
private void выходToolStripMenuItem_Click
                (object sender, EventArgs e)
{
    this.Close();
}

```

Для пункта Информация о разработчике:

```

private void информацияОРазработчикеToolStripMenuItem_
Click(object sender, EventArgs e)
{
    Form3 f3 = new Form3 ();
    f3.ShowDialog();           //Показываем форму 3
}

```

7. Пишем обработчики событий для формы 2:

Кнопка «Вернуться на главную форму» :

```

private void button1_Click(object sender,EventArgs e)
{ try
{
    this.a = Convert.ToDouble(textBox1.Text);
    this.b = Convert.ToDouble(textBox2.Text);
    this.Close();
}
}

```

```

    }
    catch { MessageBox.Show("Проверьте
корректность входных данных!", "Внимание"); }
}

```

Переменные a и b описываем с пометкой public:

```

. . . public partial class Form2 : Form
    { public double a, b;
      public Form2 ()
. . .

```

8. Пишем обработчики событий для формы 3:

Обработчик события Activated для формы 3:

```

private void Form3_Activated(object sender, EventArgs e)
{
this.label1.Text = "Приложение написано студентом
третьего курса потока ИС \nИвановым Павлом
Олеговичем, 2007";
}

```

Для кнопки «Вернуться на главную форму» :

```

private void button1_Click(object sender, EventArgs e)
{
    this.Close();
}

```

Запустите, потестируйте. Попробуйте внести некорректные данные.

Задание № 14. Программа «Масштабирование изображения»

Программа демонстрирует работу компонентов trackBar, pictureBox, toolTip и label:

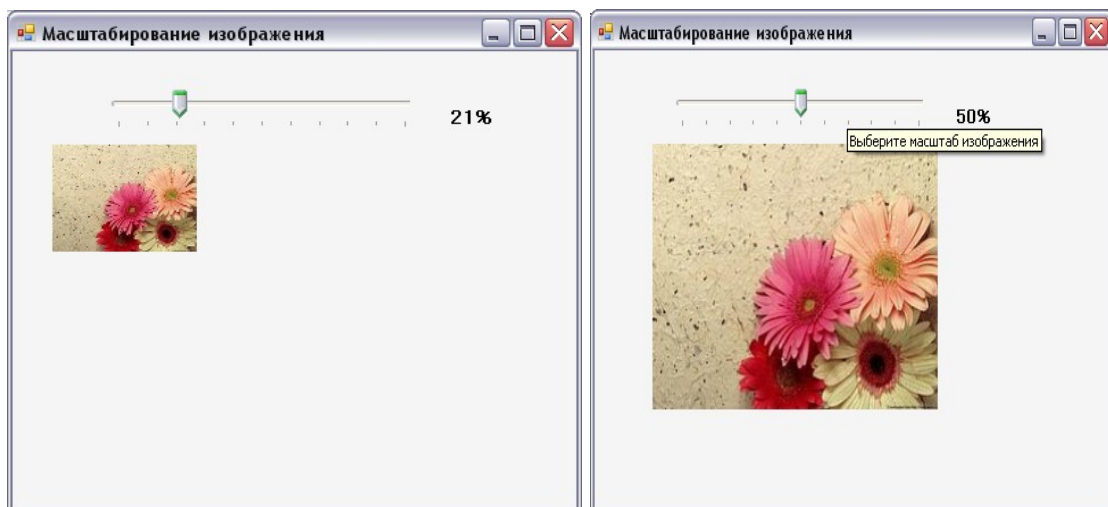


Рис. 26. Масштабирование изображения

1. Поместите на форму trackBar, pictureBox, toolTip и label.

2. Настроим pictureBox: через свойство Image загрузите картинку; в свойстве SizeMode выберите значение StretchImage (картинка занимает всю область рамки и пропорционально меняет размеры. Можете просмотреть другие значения этого свойства)
3. Настроим trackBar: через свойства Maximum и Minimum задаем диапазон (100 и 0); через Value – начальное положение (50); через TickFrequency – шаг делений (10).
4. Пусть при подведении указателя мыши к компоненту trackBar всплывает подсказка «Выберите масштаб изображения». Для этого в свойстве ToolTipOnToolTip1 у регулятора пропишите текст подсказки. Через свойства AutoPopDelay, InitialDelay компонента ToolTip можно задать время отображения подсказки и время, в течении которого указатель должен быть неподвижен на компоненте для ее вызова.

При изменении значения на линейке TrackBar наступает событие ValueChange. Пишем для него обработчик из листинга 14:

Листинг 14

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace Трек_бар
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            //Если свойства устанавливать программно, впишите:
            pictureBox1.SizeMode =
                PictureBoxSizeMode.StretchImage;
            trackBar1.Maximum = 100;
            trackBar1.Minimum = 0;
            trackBar1.Value = 50;
            trackBar1.TickFrequency = 10;
            label1.Text = Convert.ToString(trackBar1.Value) + "%";
        }

        private void trackBar1_ValueChanged (object sender,
            EventArgs e)
        {
            float a, b;
```

```

a = pictureBox1.Image.PhysicalDimension.Width;
b = pictureBox1.Image.PhysicalDimension.Height;
label1.Text=Convert.ToString(trackBar1.Value)+"%";
pictureBox1.Width = (Int16) (a/50*trackBar1.Value);
pictureBox1.Height =(Int16) (b/50*trackBar1.Value);
}
}
}

```

Здесь свойство `PhysicalDimension` компонента `pictureBox1` открывает доступ к исходным размерам загруженной картинки;

При определении новой ширины и высоты компонента `pictureBox1` использовано явное преобразование типов (из `float` правой части выражения к `int16` левой части)

Запускаем, показываем преподавателю.

Самостоятельные задания

3.1 Написать программу, в которой через пункты меню «Размер шрифта» и «Цвет» меняются соответствующие параметры текста. Например:

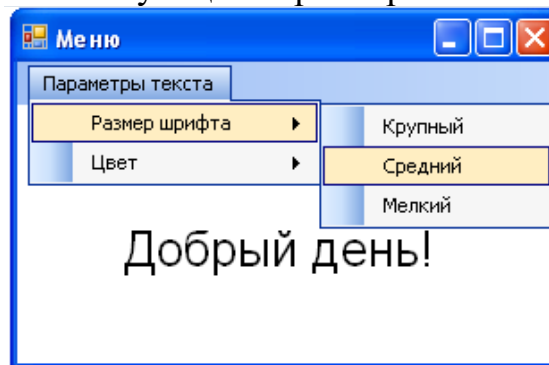


Рис. 27. Окно программы Меню

Примечание:

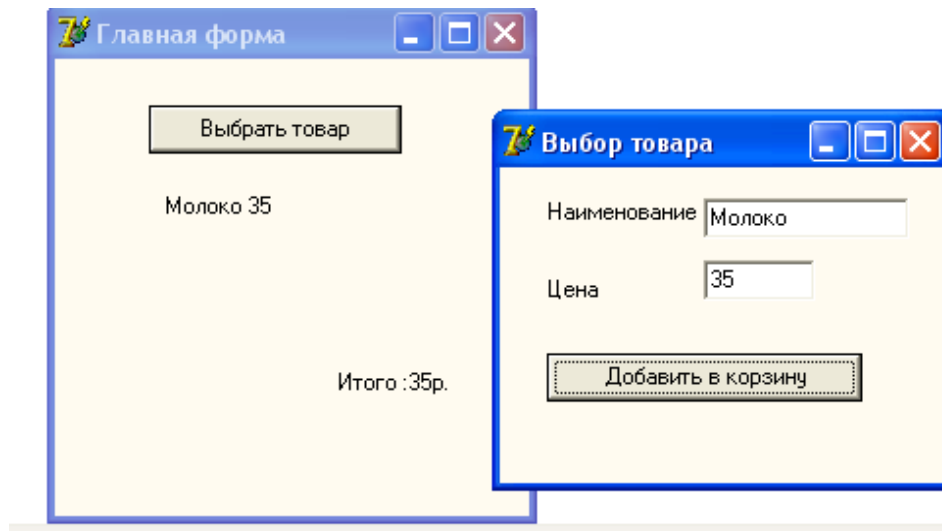
Задать цвет шрифта можно, например, так:

```
label1.ForeColor = Color.Red;
```

Задать тип и размер шрифта можно, например, так:

```
label1.Font = new Font("Arial", 11);
```

3.2 Написать приложение, состоящее из двух форм. При щелчке по кнопке «Выбрать товар» в окне «Главная форма», должно открыться вспомогательное окно «Выбор товара». После ввода наименования и цены, при щелчке по кнопке «Добавить в корзину», на главной форме должны отразиться соответствующие данные и текущая общая сумма покупки:



При повторном добавлении товаров в корзину данные должны принять вид:

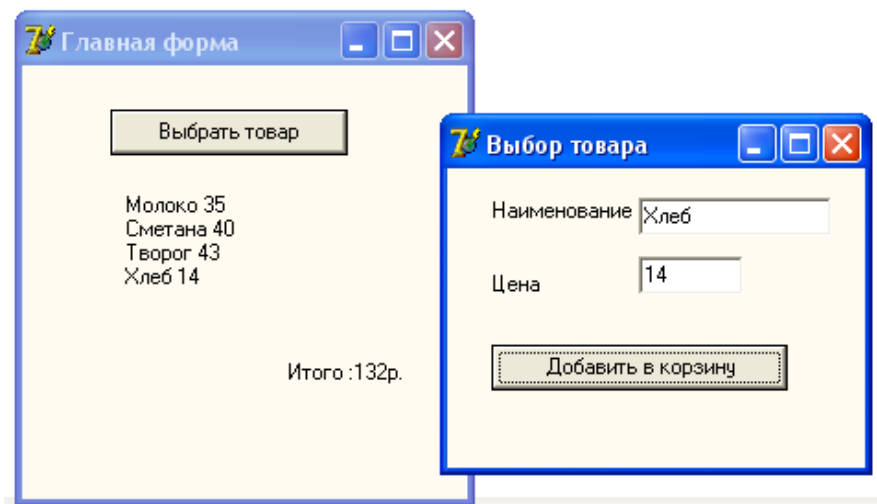


Рис.28. Окна программы расчета стоимости покупки

3.3 Измените программу «Масштабирование изображения» таким образом, чтобы изменение размеров по горизонтали и вертикали были независимы (добавьте вертикальный trackBar).

ЛАБОРАТОРНАЯ РАБОТА № 4

Тема: Работа с массивами

Цели лабораторной работы: ознакомиться и использовать в приложениях компонент `dataGridView`.

Учебные вопросы:

1. Обзор свойств `dataGridView`.
2. Работа с коллекцией строк
3. Цикл `foreach`

Задание 15. Программа «Массив, как коллекция строк»

Напишем программу, демонстрирующую возможность использования компонента `textBox` для хранения коллекции строк. Свойство `Text` используется для ввода единственной строки, а свойство `Lines` - для ввода нескольких. Строки в этом случае хранятся в виде массива, что позволяет организовать к ним индексный доступ. Для перехода в многострочный режим свойство `Multiline` ставят в значение `true`.

Пусть поэлементно вводится массив числовых данных, при нажатии на кнопку «Показать список» можно просмотреть все значения, при нажатии «Обработать данные» - найти количество элементов списка, их сумму, среднее значение, максимум и минимум.

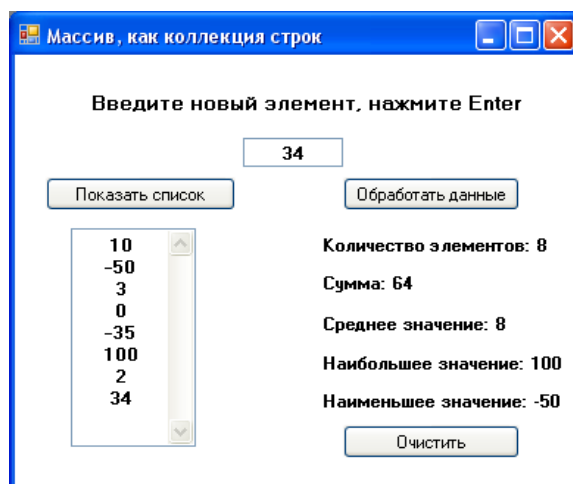


Рис. 29 а

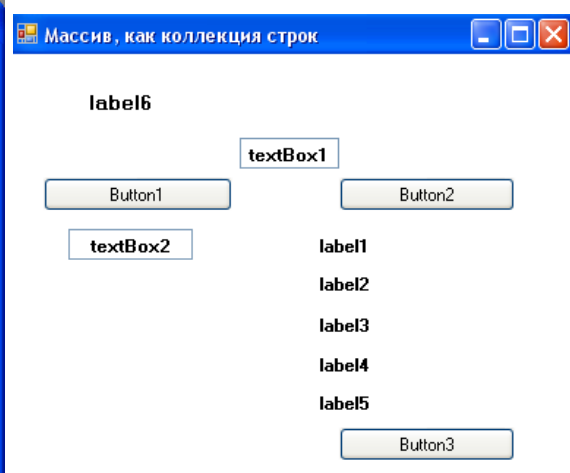


Рис. 29 б

1. Расположите на форме компоненты согласно рис 29 б. Задайте трем кнопкам, форме и компоненту `label6` названия, согласно рис 29 а.

Внимательно изучите и впишите необходимые обработчики событий согласно листингу 15:

Листинг 15

```
using System;
```

```

using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace Коллекция_строк
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            // Переходим в многострочный режим
            textBox1.Multiline = true;
            textBox2.Multiline = true;
            textBox2.Visible = false;
            label1.Text = "";
            label2.Text = "";
            label3.Text = "";
            label4.Text = "";
            label5.Text = "";
        }
        // Кнопка «Построить список»:
        private void button1_Click(object sender, EventArgs e)
        {
            //Создаем полосу прокрутки
            textBox2.ScrollBars = ScrollBars.Vertical;
            // Задаем высоту поля вывода
            textBox2.Height = 150;
            textBox2.Text = textBox1.Text;
            textBox2.Visible = true;
        }

        // Кнопка «Обработать данные»
        private void button2_Click(object sender, EventArgs e)
        {
            double sum=0;
            double max,min;
            int i,n;
            try
            {
                // количество элементов:
                n = textBox1.Lines.Length - 1;
                //минус один, т.к. последний Enter создает лишнюю пустую строку
                label1.Text = "Количество элементов:" +

```

```

Convert.ToString(n);

// Находим сумму:
for (i = 0; i < n; i++)
    sum =sum+ Convert.ToDouble(textBox1.Lines[i]);
label2.Text = "Сумма: " + Convert.ToString(sum);
// Находим среднее значение:
label3.Text = "Среднее значение: " +
Convert.ToString(sum/n);
//Находим максимум и минимум по стандартному алгоритму:
max=min= Convert.ToDouble(textBox1.Lines[0]);
for (i = 0; i < n; i++)
{ if (Convert.ToDouble(textBox1.Lines[i])> max)
    max = Convert.ToDouble(textBox1.Lines[i]);
  if (Convert.ToDouble(textBox1.Lines[i])< min)
    min = Convert.ToDouble(textBox1.Lines[i]);
}
label4.Text="Наибольшее значение: " +
Convert.ToString(max);
label5.Text="Наименьшее значение: "+
Convert.ToString(min);
}
catch
{ label1.Text = "Проверьте корректность входных
данных"; }
}

// Кнопка «Очистить»:
private void button3_Click(object sender,EventArgs e)
{
    textBox1.Text = "";
    textBox1.Focus();
    textBox2.Visible= false;
    label1.Text = label2.Text= label3.Text =
    label4.Text = label5.Text = "" ;
}
}
}

```

Сохраните программу, запустите. Покажите преподавателю.

Задание 16. Программа « Сортировка массива »

Напишем программу, форма которой изображена на рис 30 а. Массив заполняется случайными числами.

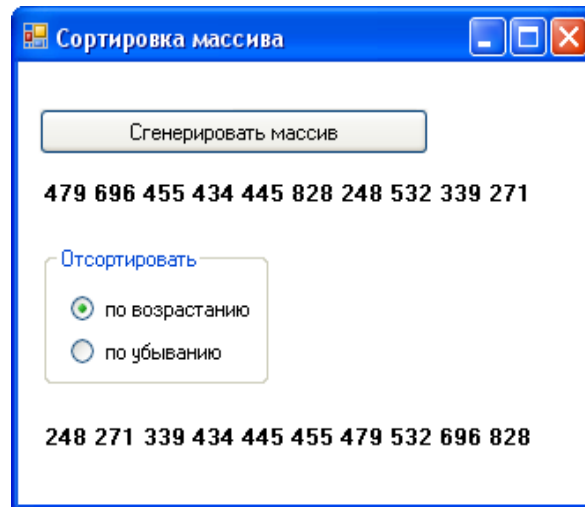


Рис. 30а

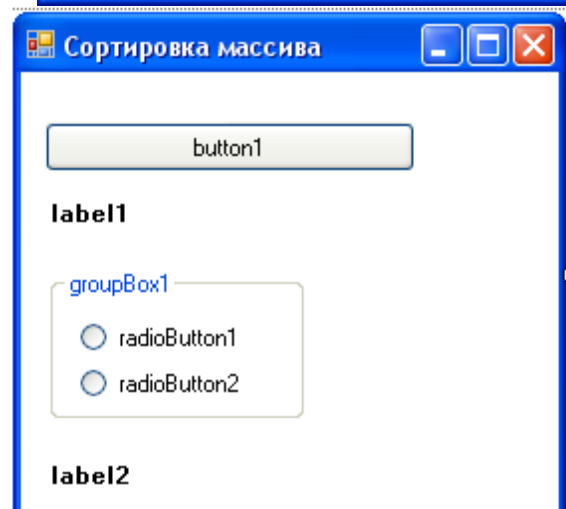


Рис. 30б

1. Расположите на форме компоненты согласно рис 30 б. Задайте надпись форме, кнопке, компоненту `groupBox` и двум компонентам `radioButton`.
2. Внимательно изучите и впишите обработчики событий согласно листингу 16:

Листинг 16

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace Сортировка_массива
{
```

```

public partial class Form1 : Form
{
    int[] Mas;
    int i;
    public Form1()
    {
        InitializeComponent();
        label1.Text = "";
        label2.Text = "";
    }

    // Для кнопки « Сгенерировать массив »
    private void button1_Click(object sender, EventArgs e)
    {
        Mas=new int[10];
        Random n = new Random();
        for (i=0; i<10; i++)
            Mas[i]= n.Next(1000);
        // Выводим на экран:
        foreach ( int elem in Mas)
            label1.Text = label1.Text + elem + " ";
    }

    /*      Для вывода массива на экран можно было
использовать и традиционный цикл for, тогда последние
две строки процедуры заменились бы на:
        for (i = 0; i < 10; i++)
            label1.Text = label1.Text + Mas[i] + "\n";
    */

    // Через вкладку Events заходим в обработчик события
Click для radioButton1:

    private void radioButton1_Click(object sender, EventArgs e)
    {
        try
        {
            Array.Sort(Mas);
            label2.Text = "";
            foreach (int elem in Mas)
                label2.Text = label2.Text + elem + " ";
        }
        catch
        {
            label2.Text = "Вы еще не заполнили массив
данными!";
        }
    }
}

```

```

    }
}

// Через вкладку Events заходим в обработчик события
Click для radioButton2:
private void radioButton2_Click(object sender, EventArgs e)
{
    try
    {
        Array.Sort(Mas);
        Array.Reverse(Mas);
        label2.Text = "";
        foreach (int elem in Mas)
            label2.Text = label2.Text + elem + " ";
    }
    catch
    {
        label2.Text = "Вы еще не заполнили массив
данными!";
    }
}
}
}

```

Использованный метод Sort для сортировки массивов позволил заменить громоздкий код типа:

```

for (int i = 0; i < 9; i++)
    for (int j = 0; j < 9; j++)
        if (Mas[j] > Mas[j + 1])
        {
            temp = Mas[j];
            Mas[j] = Mas[j + 1];
            Mas[j + 1] = temp;
        }

```

на строчку: `Array.Sort(Mas);`

Метод Reverse изменил порядок следования элементов одномерного массива на обратный.

Задание 17. Программа «Поиск максимума и минимума в таблице»

Напишем приложение, демонстрирующее работу компонента `dataGridView`. Сформируем таблицу 5x5, заполним ее случайными числами из интервала от -100 до 100, выделим отдельными цветами ячейки с максимальным и минимальным значениями:

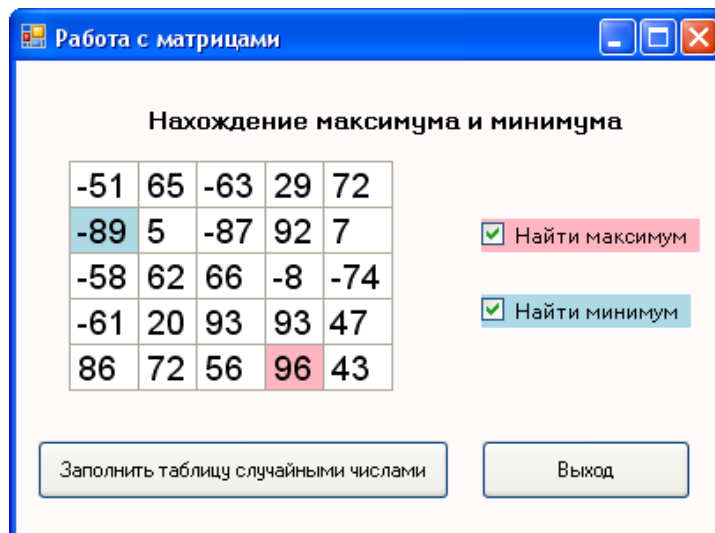


Рис 31 Окно программы поиска max и min значения

1. Поместите на форму надпись, два компонента checkbox, две кнопки. Подпишите их.
2. Поместите компонент dataGridView. Появится окно DataGridView Tasks. В пункте ChooseDataSource (источник данных) оставьте значение none, повторным щелчком по компоненту уберите окно.

Чаще всего данные попадают в DataGridView из подключенного источника данных. При этом встроенный механизм Windows Forms Data Binding автоматически заполняет каждую ячейку значением из соответствующей ячейки источника. Однако компонент поддерживает также специальный режим отображения «свободных» данных. Кроме того, поддерживается комбинированный режим одновременного отображения связанных и свободных данных.

Свободными данными называются данные, заносимые в ячейки вручную (в примере будем работать с ними). Связанными данными называются данные, заносимые в ячейки автоматически из привязанного источника данных.

Главное в ячейках – их значения. В свободных колонках значения хранятся в экземплярах ячеек. В случае ячеек, связанных с источником данных, ячейки вообще ничего не знают о значениях и не хранят их. Если требуется значение ячейки, grid обращается к источнику данных за значением.

За тип данных, хранящихся в ячейке, отвечает свойство ValueType. По умолчанию считается, что в свойстве Value ячейки хранится объект неопределенного типа (ValueType равно object). Если изменить это значение на значение конкретного типа, ячейка начнет контролировать значение, конвертируя его в/из типа, указанного в свойстве ValueType. Предлагается 6 типов ячеек:

-  DataGridCheckBoxColumn
-  DataGridButtonColumn
-  DataGridComboBoxColumn
-  DataGridImageColumn
-  DataGridLinkColumn
-  DataGridTextBoxColumn

В примере будем использовать тип TextBoxColumn, он уже выбран по умолчанию.

3. Задайте компоненту dataGridView следующие свойства:

AutoSizeColumnsMode -> AllCells, AutoSizeRowsMode -> AllCells

(размеры строк и столбцов таблицы будут автоматически подгоняться под размеры данных)

Остальное сделаем программно:

Листинг 17

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace Матрица
{
    public partial class Form1 : Form
    {
        const int n = 5; // n- число столбцов
        const int m = 5; // m- число строк
        int i, j;
        /* в k и l будем заносить номера ячеек с max/min
        значениями */
        int k, l;
        public Form1()
        {
            InitializeComponent();
            // Цвет фона таблицы задаем такой же, как у формы:
            dataGridView1.BackgroundColor =
this.BackColor;
            // Убираем внешнюю границу компонента:
            dataGridView1.BorderStyle = BorderStyle.None;
            //задаем число столбцов
            dataGridView1.ColumnCount = n;
            //задаем число строк
            dataGridView1.RowCount = m;
            // убираем шапку таблицы:
```



```

dataGridView1.ColumnHeadersVisible = false;
dataGridView1.RowHeadersVisible = false;
    // Убираем полосу прокрутки:
dataGridView1.ScrollBars = ScrollBars.None;
    // Задаем размеры таблицы (при необходимости измените):
dataGridView1.Width = 320;
dataGridView1.Height = 160;
    // Задаем параметры шрифта для ячеек таблицы:
dataGridView1.Font = new Font("Arial", 14);
}

// Кнопка «Заполнить таблицу случайными числами»
private void button1_Click(object sender, EventArgs e)
{
    // Задаем белый цвет всем ячейкам:

dataGridView1.DefaultCellStyle.BackColor=Color.White;
    // «Выключаем» checkbox-ы :
checkBox1.Checked = false;
checkBox2.Checked = false;
    // Заполняем ячейки:
Random r = new Random();
for (i=0;i<n;i++)
    for (j = 0; j < m; j++)
        dataGridView1.Rows[i].Cells[j].Value =
            (r.Next(200)-100).ToString();
}

// Обработчик события CheckedChange для checkBox1:
private void checkBox1_CheckedChanged
    (object sender, EventArgs e)
{
    // За max принимаем значение ячейки (0,0):
    int max =
Convert.ToInt16(dataGridView1.Rows[0].Cells[0].Value);
    //далее- по стандартному алгоритму поиска
максимума:
    for (i = 0; i < n; i++ )
        for (j = 0; j < m; j++)
            if (Convert.ToInt16(dataGridView1.Rows[i].
Cells[j].Value)>max)
                { max =
Convert.ToInt16(dataGridView1.Rows[i].Cells[j].Value);
                    k = i;
                    l = j;

```

```

    }

// Если кнопка «Включена» -орашиваем найденную ячейку в розовый цвет:
    if (checkBox1.Checked)
dataGridView1.Rows[k].Cells[1].Style.BackColor =
                                                Color.LightPink;
    // Если нет- меняем цвет ячейки на фоновый белый:
    else
dataGridView1.Rows[k].Cells[1].Style.BackColor
                                                = Color.White;
}

// Аналогично для checkBox2:
private void checkBox2_CheckedChanged
                (object sender, EventArgs e)
{
    int min =
Convert.ToInt16(dataGridView1.Rows[0].Cells[0].Value);
    for (i = 0; i < n; i++)
        for (j = 0; j < m; j++)
            if (Convert.ToInt16(dataGridView1.Rows[i].
Cells[j].Value)<min)
                {
min =
Convert.ToInt16(dataGridView1.Rows[i].Cells[j].Value);
                    k = i;
                    l = j;
                }
            if (checkBox2.Checked)
                dataGridView1.Rows[k].Cells[1].Style.BackColor
= Color.LightBlue;
            else
dataGridView1.Rows[k].Cells[1].Style.BackColor
                                                = Color.White;
}

// Выход:
private void button2_Click(object sender,EventArgs e)
{
    if (MessageBox.Show("Вы действительно хотите
выйти?", "Выход", MessageBoxButtons.YesNo,
MessageBoxIcon.Question) == DialogResult.Yes)
    {
        this.Close();
    }
}

```

```
}
}
```

Задание 18. Программа « Успеваемость студентов »

Продолжим изучение возможностей компонента `dataGridView`.

Задаются фамилии студентов, номера групп и результаты сдачи ими трех лабораторных работ. Требуется определить, кто получил допуск к экзамену. При щелчке по кнопке «Обработать данные» программа формирует дополнительный шестой столбец «Допуск к экзамену» с результатами. Кроме того, при наличии допуска, соответствующая строка окрашивается другим цветом:

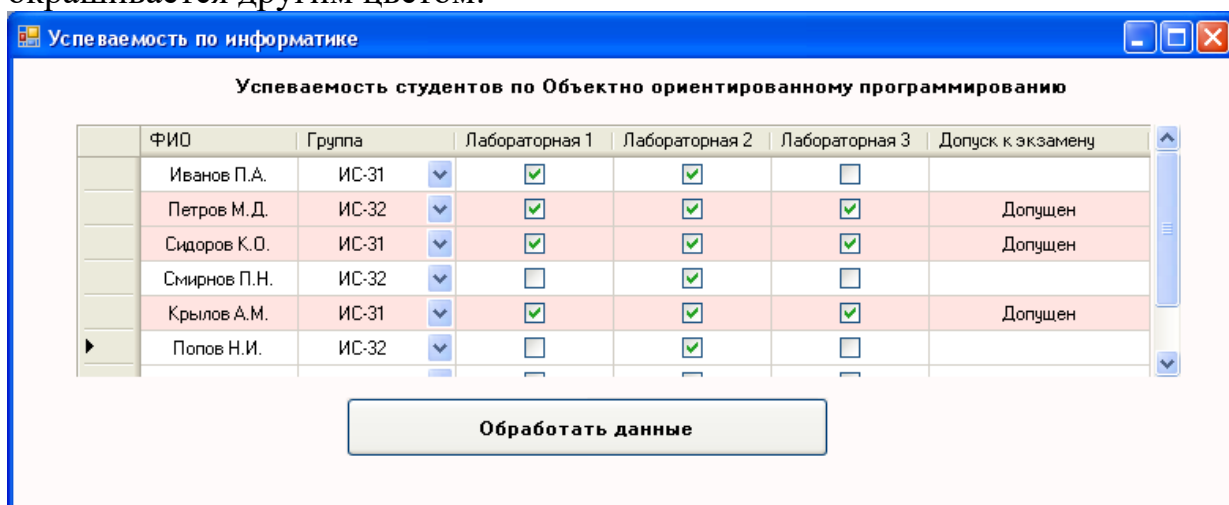


Рис 32 Программа Успеваемость студентов

1. Поместите на форму `label` и `button`. Задайте им надписи.
2. Поместите `dataGridView`. В пункте `ChooseDataSource` оставьте значение `none`. Зайдите в пункт `Edit Columns` контекстного меню компонента. Добавьте пять колонок со следующими свойствами:

| Name | Type | Header Text |
|---------|----------------------------|----------------|
| Column1 | DataGridViewTextBoxColumn | ФИО |
| Column2 | DataGridViewComboBoxColumn | Группа |
| Column3 | DataGridViewCheckBoxColumn | Лабораторная 1 |
| Column4 | DataGridViewCheckBoxColumn | Лабораторная 2 |
| Column5 | DataGridViewCheckBoxColumn | Лабораторная 3 |

Инициализируем объекты и вписываем функцию для кнопки «Обработать» :

Листинг 18

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
```

```

using System.Text;
using System.Windows.Forms;
namespace Оценки
{ public partial class Form1 : Form
    { public Form1()
        { InitializeComponent();
/* Создаем элементы списка comboBox для второй
колонки:(не забудьте, индексация строк и столбцов
начинается с нуля) */

((DataGridViewComboBoxColumn)dataGridView1.Columns[1]
).Items.AddRange (new string[] { "ИС-31", "ИС-32" });
// Задаем цвет фона компонента:
dataGridView1.BackgroundColor = this.BackColor;
// Убираем рамку:
dataGridView1.BorderStyle = BorderStyle.None;
// Задаем размеры таблицы:
dataGridView1.Width = 700;
dataGridView1.Height = 160;
// Создаем вертикальную полосу прокрутки:
dataGridView1.ScrollBars = ScrollBars.Vertical;
// Задаем выравнивание текста в ячейках по центру:
dataGridView1.DefaultCellStyle.Alignment =
    DataGridViewContentAlignment.MiddleCenter;
// Все элементы оформления можете и перенастроить
    }

// Кнопка «Обработать данные»
private void button1_Click(object sender,EventArgs e)
{ int i;
// Если у нас 5 начальных колонок, формируем еще одну
    if (dataGridView1.ColumnCount==5)
        dataGridView1.Columns.Add("Column5",
                                "Допуск к экзамену");
    dataGridView1.Columns[5].Width = 140;
    /* Смотрим во всех строках на результаты сдачи
лабораторных: */
    for (i=0;i<dataGridView1.RowCount;i++)
        if (dataGridView1.Rows[i].Cells[2].Value != null
&&      dataGridView1.Rows[i].Cells[3].Value != null
&&      dataGridView1.Rows[i].Cells[4].Value != null)
            { // Если все сдано- пишем «Допущен» и
              // закрашиваем строку:

```

```

        dataGridView1.Rows[i].Cells[5].Value= "Допущен";
dataGridView1.Rows[i].DefaultCellStyle.BackColor =
                                Color.MistyRose;
    }
}
}
}

```

Запускаем, если работает- показываем преподавателю.

Самостоятельные задания

4.1. Написать программу, проверяющую, есть ли в массиве элемент, равный заданному. Если есть- указать позицию этого элемента в массиве. Массив заполняется целыми случайными числами:

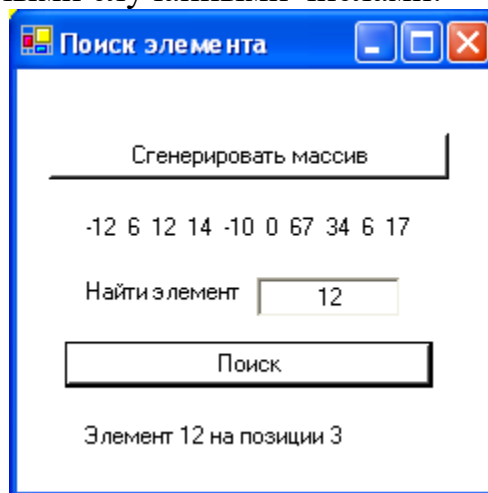


Рис. 33. Окно программы Поиск элемента

4.2. Написать программу, вычисляющую определитель матрицы 3x3:

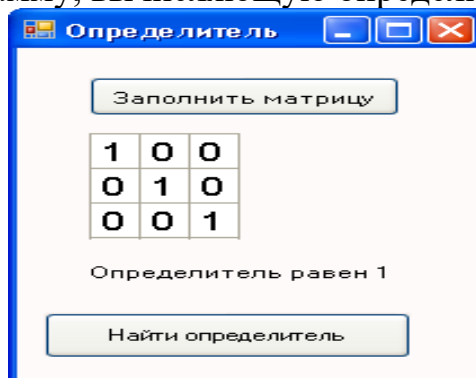


Рис.34. Нахождение определителя

4.3. В программе «Успеваемость студентов» из задания 18 определить, сколько допущенных студентов в каждой из групп:

Успеваемость по информатике

Успеваемость студентов по Объектно ориентированному программированию

| ФИО | Группа | Лабораторная 1 | Лабораторная 2 | Лабораторная 3 | Допуск к экзамену |
|--------------|--------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------|
| Иванов Т.Л. | ИС-31 | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | Допущен |
| Петров Г.Д. | ИС-31 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| Сидоров О.Н. | ИС-32 | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | Допущен |
| Воронов П.А. | ИС-31 | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | Допущен |
| Орлов А.А. | ИС-32 | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | |
| Крылов А.В. | ИС-31 | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | |

Обработать данные

В группе ИС-31 допущено: 2
В группе ИС-32 допущено: 1

Рис.35. Определение количества допущенных студентов

ЛАБОРАТОРНАЯ РАБОТА № 5

Тема: Обработка текстовых данных, данных типа дата/время, программа «Угадай число»

Цели лабораторной работы: ознакомиться и использовать в приложениях компоненты progressBar, statusStrip, fontDialog, colorDialog, treeView, DateTimePicker

Учебные вопросы:

1. Составление программы «Угадай число», демонстрирующую работу компонентов progressBar, statusStrip и timer
2. Обзор методов, использующихся при обработке тестовой информации
3. Диалоговые окна fontDialog, colorDialog
4. Функции для работы с типом дата/время. Компонент DateTimePicker
5. Программа «Браузер файлов». Возможности компонента treeView

Задание 19. Программа « Угадай число »

Напишем программу, демонстрирующую работу компонентов progressBar, statusStrip и timer. Компьютер загадывает произвольное число из интервала от 1 до 999, пользователю предлагается его угадать за 60 сек. При этом выдаются подсказки- задуманное число больше или меньше. Параллельно подсчитывается количество попыток и наглядно отображается оставшийся отрезок времени:

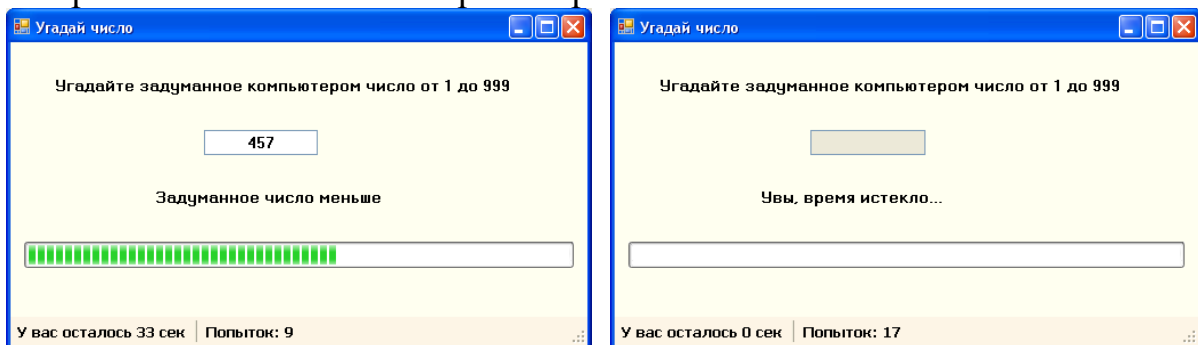


Рис. 36. Программа Угадай число

1. Поместите на форму два компонента label, по одному textBox, progressBar, statusStrip, timer.
2. Форме и компоненту label1 задайте надписи, согласно рис 36.
3. Задайте выравнивание текста по центру в поле textBox.
4. В свойстве style компонента progressBar выберите Blocks. Начальное значение компонента progressBar, его максимальное и минимальные значения, шаг можно задать и сейчас, но лучше сделаем это программно.
5. В строке состояния должен отображаться обратный отчет времени и количество сделанных попыток. Выделите компонент statusStrip. Зайдите в его свойство Items. Добавим в его коллекцию два элемента. Для этого в

списке *Select item and add to list below* выберите StatusLabel, нажмите Add. Добавьте еще одну StatusLabel. OK.

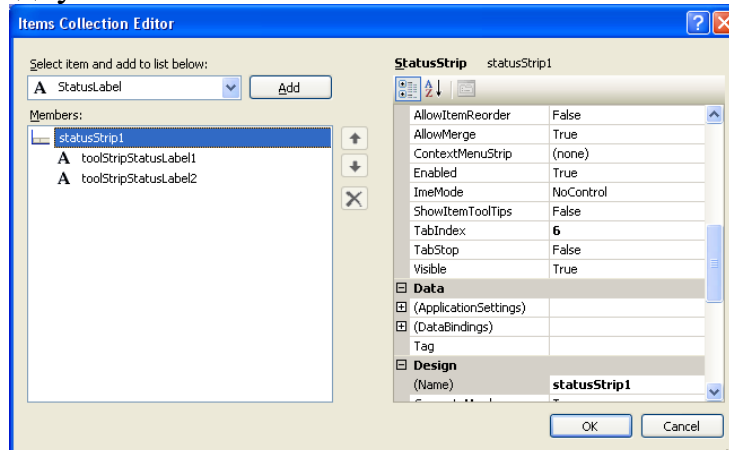


Рис. 37. Редактор компонента statusStrip

Добавим разделительную черту между пунктами StatusStrip. Для этого выделите компонент ToolStripStatusLabel1, в его свойстве BorderSides выберите пункт Right.

6. Для перебора значений с заданным временным шагом идеально подойдет компонент timer. Его начальные свойства зададим программно.

Внимательно изучите, впишите необходимые объявления и обработчики событий согласно листингу 19:

Листинг 19

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace Угадай_число
{
    public partial class Form1 : Form
    {
        const int t = 60;
        int zadumano = 0;
        int ostalos = 60;
        int nomer_popitki = 0;
        public Form1()
        {
            InitializeComponent();
            textBox1.Focus();
            label2.Text = "";
            toolStripStatusLabel1.Text = "У вас осталось " +
            Convert.ToString(t) + " сек";
        }
    }
}
```



```

        toolStripStatusLabel2.Text = " Попыток: 0 ";
        timer1.Enabled = true;
        timer1.Interval = 1000;
        // (при необходимости перенастройте)
        progressBar1.Maximum = t;
        progressBar1.Value = t;
        progressBar1.Step = 1;
        Random n = new Random();
        zadumano = n.Next(1000); // от 1 до 999
    }

    private void timer1_Tick(object sender, EventArgs e)
    {
        ostalos--;
        progressBar1.Value--;
        toolStripStatusLabel1.Text = "У вас осталось" +
Convert.ToString(ostalos) + " сек ";
        if (ostalos == 0)
        {
            timer1.Enabled = false;
            textBox1.Enabled = false;
            progressBar1.Enabled = false;
            label2.Text = "Увы, время истекло...";
        }
    }

    private void textBox1_KeyPress(object sender,
                                   KeyPressEventArgs e)
    {
        // Если в поле нажат Enter
        if (e.KeyChar.Equals((char)13))
        {
            try
            {
                if (Convert.ToInt16(textBox1.Text)==zadumano)
                {
                    timer1.Enabled = false;
                    textBox1.Enabled = false;
                    label2.Text = "Вы угадали!,
задумывалось число " + Convert.ToString(zadumano);
                };
                if (Convert.ToInt16(textBox1.Text) > zadumano)
                    label2.Text = "Задуманное число меньше";
                if (Convert.ToInt16(textBox1.Text) < zadumano)
                    label2.Text = "Задуманное число больше";
            } // для try
        }
    }

```

```

        catch { label2.Text = "Некорректные входные
данные!"; }
    nomer_popitki++;
    toolStripStatusLabel2.Text = " Попыток:" +
        Convert.ToString(nomer_popitki);
    textBox1.Text = "";
    textBox1.Focus();
} // для if
} // закрыли функцию textBox1_KeyPress
}
}

```

Запустите. Потестируйте. Попробуйте выбрать стиль Marquee для progressBar (св-во Style)

Задание № 20. Программа « Обработка текста»

Программа демонстрирует ряд методов, использующихся при обработке тестовой информации. Пусть в поле TextBox водится некоторый текст (или загружается из файла). Выделяется некоторый его фрагмент (или текст целиком), при щелчке по кнопке «Обработать выделенный фрагмент текста» подсчитывается количество цифр, букв, знаков препинания и пробелов в нем. Через кнопки «Задать шрифт» и «Задать фон» устанавливаются параметры шрифта и фона (рис 38):

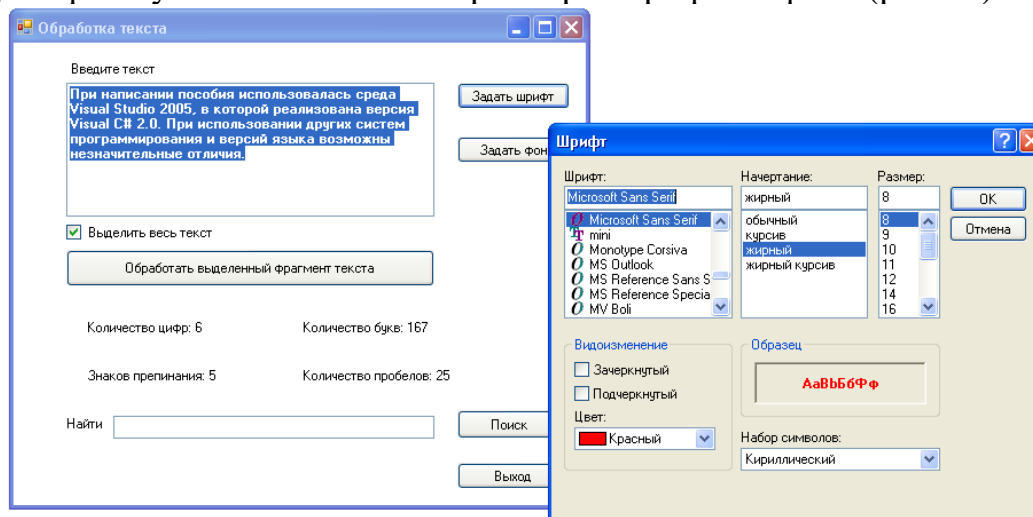


Рис. 38. Обработка текста

Помимо этого реализуется поиск слов или фраз с последующим их выделением в исходном тексте (рис 39):

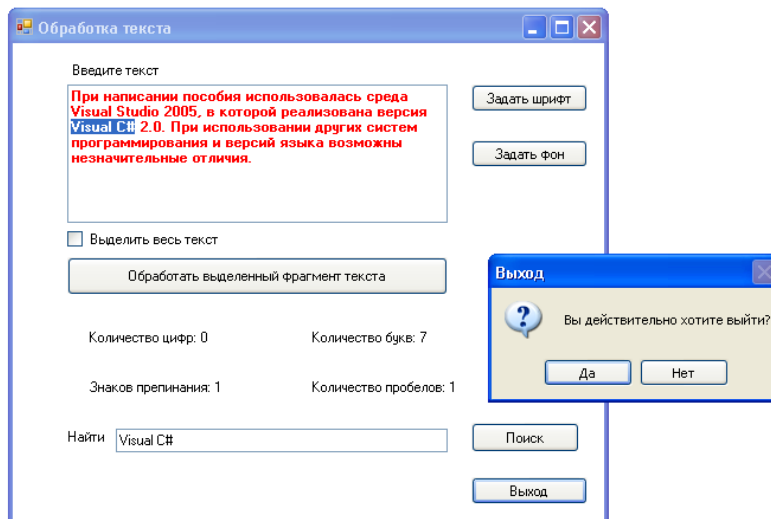


Рис. 39. Поиск по тексту

Разместите на форме шесть компонентов label, два textBox (textBox1 сразу переведите в многострочный режим), checkBox, пять кнопок, fontDialog и colorDialog. Задайте надписи меткам 1 и 6, кнопкам и компоненту checkBox. (в примере label 2,3,4,5 оставлены под ответ, их пока не трогайте).

Листинг 20

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace Текстовый_редактор
{
    public partial class Form1 : Form
    {
        public int p = 0;
        //p - глобальная переменная, используемая при поиске
        public Form1()
        {
            InitializeComponent();
            /* Пусть при установке параметров шрифта в том
            же окне будет доступно задание его цвета:*/
            fontDialog1.ShowColor = true;
            label2.Text = "";
            label3.Text = "";
            label4.Text = "";
            label5.Text = "";
            //(что бы текст выделялся при поиске):
            textBox1.HideSelection = false;
        }
    }
}
```

```

// Кнопка «Обработать выделенный фрагмент текста»:
private void button1_Click(object sender, EventArgs e)
{
    int k=0,n=0,t=0,m=0;
    for (int i = 0;i< textBox1.SelectedText.Length; i++)
        {
            if(Char.IsDigit(textBox1.SelectedText[i])) k++;
            if(Char.IsLetter(textBox1.SelectedText[i])) n++;
            if(Char.IsPunctuation(textBox1.SelectedText[i])) t++;
            if (Char.IsWhiteSpace(textBox1.SelectedText[i])) m++;
        } // for
    label2.Text = "Количество цифр: " + Convert.ToString(k);
    label3.Text = "Количество букв: " + Convert.ToString(n);
    label4.Text = "Знаков препинания: " + Convert.ToString(t);
    label5.Text = "Пробелов: " + Convert.ToString(m);
}
// Флажок «Выделить все»:
private void checkBox1_CheckedChanged(object sender,
                                       EventArgs e)
{
    if (checkBox1.Checked) textBox1.SelectAll();
    else textBox1.SelectionLength=0;
}
// Кнопка «Поиск»:
private void button2_Click(object sender, EventArgs e)
{
    if (textBox1.Text.IndexOf(textBox2.Text, p) >= 0)
        {
            /* в переменной p запомним позицию первого
            вхождения фрагмента, в следующий раз поиск будем
            вести не сначала текста, а с позиции p */
            if (p <= textBox1.Text.Length)
                {
                    textBox1.SelectionStart =
                        textBox1.Text.IndexOf(textBox2.Text, p);
                    textBox1.SelectionLength=textBox2.Text.Length;
                    p = textBox1.SelectionStart + 1;
                }
        }
    else
        {
            MessageBox.Show("Поиск не дал результатов");
        }
}
//Поиск следующих вхождений будем вести сначала текста:
p = 0;
} }

// Кнопка «Выход»:
private void button3_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Вы действительно хотите
    выйти?", "Выход", MessageBoxButtons.YesNo,
    MessageBoxIcon.Question)==DialogResult.Yes) this.Close();
}

```

```

// Кнопка «Задать шрифт»:
private void button4_Click(object sender, EventArgs e)
{
    if (fontDialog1.ShowDialog() == DialogResult.OK)
    {
        textBox1.Font = fontDialog1.Font;
        textBox1.ForeColor = fontDialog1.Color;
    }
}
// Кнопка «Задать фон»:
private void button5_Click(object sender, EventArgs e)
{
    if (colorDialog1.ShowDialog() == DialogResult.OK)
        textBox1.BackColor = colorDialog1.Color;
}
}
}

```

Здесь методы IsDigit(), IsLetter(), IsPunctuation(), IsWhiteSpace() проверяют, не является ли текущий символ цифрой, буквой, знаком препинания или пробелом.

SelectAll()- выделение всего текста;

textBox1.SelectedText.Length-длина выделенного участка;

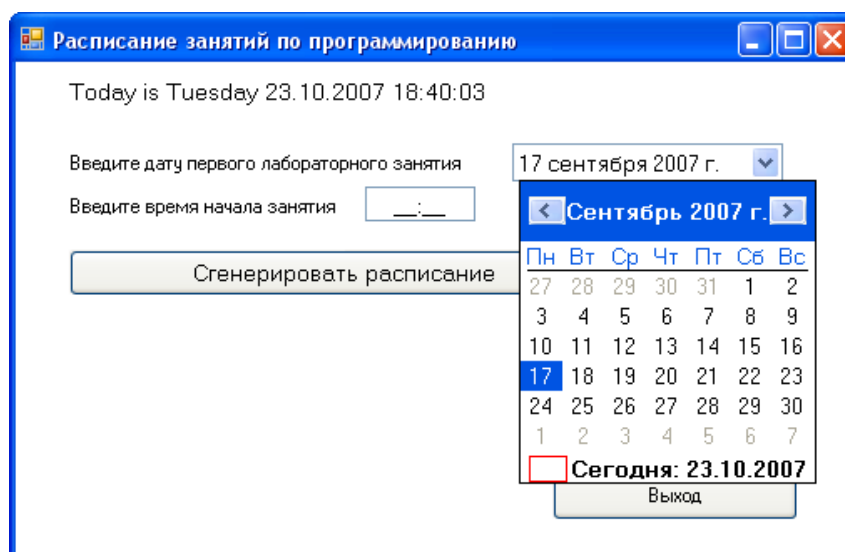
textBox1.SelectedText[i]- обращение к i-ому символу выделенного участка;

textBox1.SelectionStart- позиция, с которой начинается выделенный участок;

Задание № 21. Программа «Расписание занятий»

Напишем программу, демонстрирующую работу с типом Дата/ Время.

Пользователь выбирает дату первого занятия по программированию, при щелчке по кнопке «Сгенерировать расписание» выводятся даты остальных занятий (с периодом в 2 недели). Кроме этого отображаются текущая дата, день недели и время. При щелчке по кнопке «Выход» пусть окно закрывается не сразу, а сначала постепенно становится прозрачным.



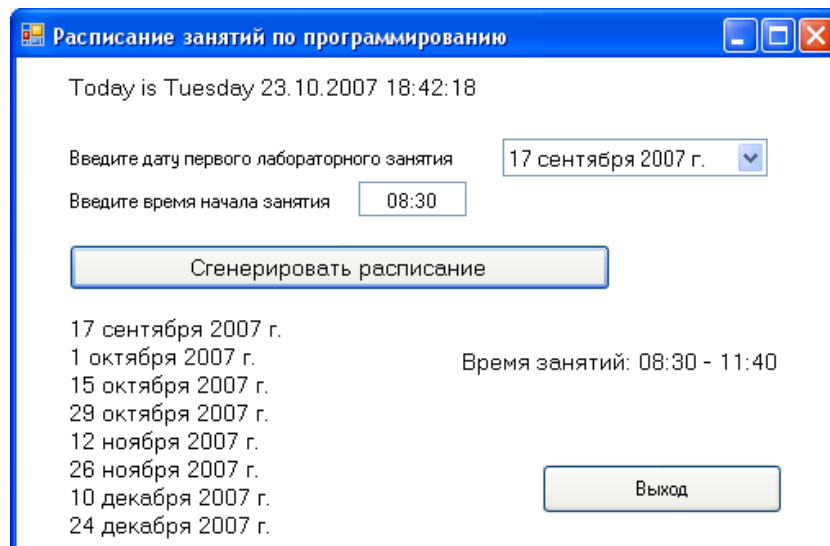


Рис. 40 Окна программы Расписание занятий

1. Поместите на форму пять компонентов label, две кнопки, DateTimePicker, maskedTextBox, Timer1 (нужен для динамического обновления текущего времени), Timer2 (понадобится при обработке кнопки Выход).
2. Задайте надписи для label2: Введите дату первого лабораторного занятия, label3 : Введите время начала занятия. Надписи остальным меткам зададим программно. (в label1 выведется текущие дата и время, в label4-расписание занятий, в label5 – время занятий)
3. Зададим маску для ввода времени: свойству PromptChar компонента maskedTextBox задайте значение _ (этот символ будет отображаться в шаблоне), свойство Mask установите в значении Time(European/Military)

Листинг 21

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace Расписание
{
    public partial class Form1 : Form
    {
        DateTime d0,d1,d2;
        int t = 100; // Степень прозрачности формы
        public Form1()
        {
            InitializeComponent();
            timer1.Enabled = true;
            timer2.Enabled = false;
        }
    }
}
```

```

        timer1.Interval = 500;
        timer1.Interval = 50;
        label1.Text = "";
        label4.Text = "";
        label5.Text = "";
    }

    /*С заданным периодом обновляем текущую дату, время, день недели
    (особо актуально для секунд)*/
    private void timer1_Tick(object sender, EventArgs e)
    {
        // В d0 записываем текущую дату и время:
        d0 = DateTime.Now;
        label1.Text="Today is"+Convert.ToString(d0.DayOfWeek)
        + " " + Convert.ToString(d0);
    }
    // Кнопка «Сгенерировать расписание»
    private void button1_Click(object sender, EventArgs e)
    {
        try
        {
            label4.Text = "";
            //В d1 записываем выбранную дату из DateTimePicker:
            d1 = dateTimePicker1.Value;
            // Допустим, предполагается 8 лабораторных занятий:
            for (int i = 1; i < 9; i++)
            {
                label4.Text = label4.Text +
                    d1.ToLongDateString() + "\n";
                // Сдвигаем d1 на 2 недели вперед:
                d1 = d1.AddDays(14);
            }
            //В d2 записываем время из maskedTextBox :
            d2 = Convert.ToDateTime(maskedTextBox1.Text);
            // Допустим, занимаемся 2 пары с перерывом в 10 минут:
            label5.Text="Время занятий: "+maskedTextBox1.Text+
            "_"
            +d2.AddHours(3).AddMinutes(10).ToShortTimeString();
        }
        catch { MessageBox.Show("Проверьте корректность
        входных данных!"); }
    }
    // Для кнопки «Выход»:
    private void button2_Click(object sender, EventArgs e)
    {
        // Запускаем генерацию событий Tick для второго таймера
        timer2.Enabled = true;
    }

```

```

    }
    // С каждым тиком делаем форму все более прозрачной:
    private void timer2_Tick(object sender, EventArgs e)
    {
        if (t != 0)
        {
            this.Opacity=(float)t/100;
            this.Refresh(); //отрисовываем форму заново
            t--;
        } else this.Close();
    }
}
}

```

Здесь функция **DayOfWeek** возвращает день недели. **Now** – возвращает текущие дату и время; **ToLongDateString()** – дата с названием месяца; **AddDays(n)** – сдвигает дату на n дней, **AddMinutes(n)**- прибавляет (если n положительное) n минут, **AddHours(n)**- прибавляет n часов.

Можно использовать форматный вывод дат и времени. Например, вместо строки вывода текущих дат и времени:

```

label1.Text="Today is
"+Convert.ToString(d0.DayOfWeek)
+ " " + Convert.ToString(d0);

```

попробуйте написать:

```

label1.Text = "Сегодня " + DateTime.Now.ToString("d
MMMM yyyy, dddd HH:mm:ss"); (приятный плюс- день недели
выведется в русском варианте).

```

Подробнее о спецификаторах формата см. в приложении.

Свойство **Opacity** (степень прозрачности) принимает значения от 0 до 100% (в программе 1% - 0,01; 99% - 0,99)

Задание № 22. Программа « Браузер файлов»

Рассмотрим возможности компонента **TreeView** на следующем примере [5]. Через диалоговое окно выбирается некоторый каталог:

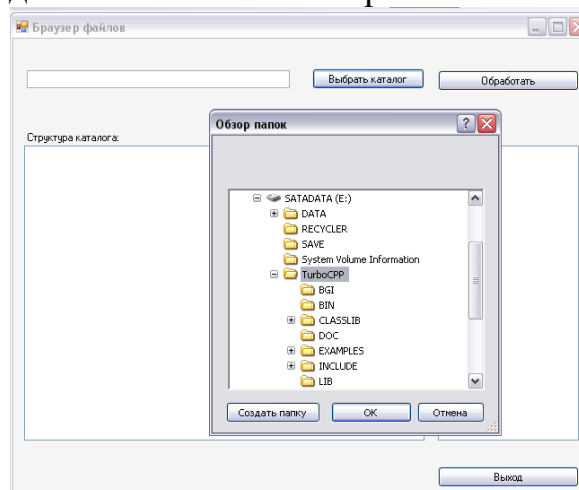


Рис 41 Диалоговое окно открытия каталога

при щелчке по кнопке «Обработать» отрисовывается структура выбранного каталога, при навигации по вложенным каталогам структуры в списке listBox (справа) отображается список содержащихся в них файлов (рис 42):

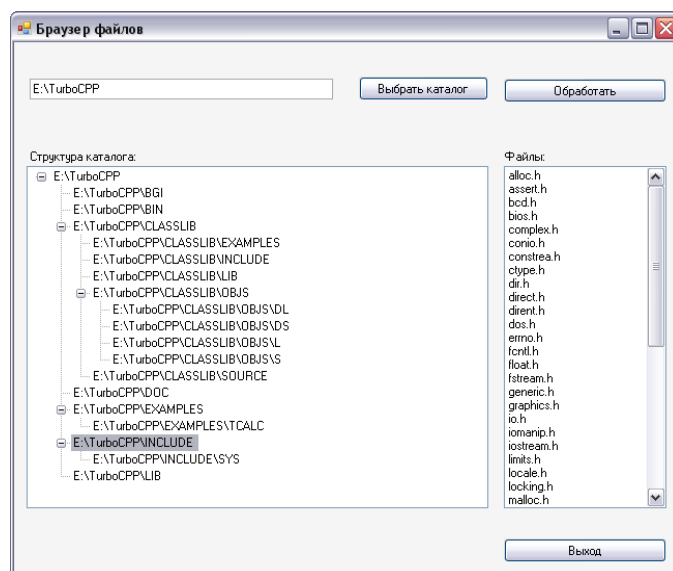


Рис 42 Отрисовка структуры каталога

В основе программы лежит рекурсивный обход дерева каталогов с помощью процедуры `ProcessDirectory()`. Перед началом изменения дерева вызывается метод `BeginUpdate()`, который блокирует отрисовку изменений. После формирования дерева вызывается метод `EndUpdate()`, который отрисовывает новое дерево. Так как процесс может быть довольно долгим, в метке `label1` (на рисунках она не видна) отображаются имена обрабатываемых в данный момент каталогов. Метод `Application.DoEvents()` приостанавливает работу программы до завершения обработки всех событий.

Для работы с каталогами и файлами используется соответствующие классы пространства имен System.IO (необходимо будет вписать вручную в раздел `using`).

Поместите на форму три кнопки, поле `textBox`, `treeView`, `listBox`, три надписи (`label1` поместите между `textBox` и `treeView`, `label2` и `label3` видны на рисунке- «Структура каталога» и «Файлы»). Также понадобится невидимый компонент `FolderBrowserDialog`. Установите его.

Листинг 22

```
using System;  
using System.Collections.Generic;  
using System.ComponentModel;  
using System.Data;
```

```

using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;
namespace Дерево
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            label1.Text = "";
        }

        // Кнопка «Выбрать каталог» :
        private void button1_Click(object sender, EventArgs e)
        {
            // Каталог выбран? Помещаем его имя в textBox1:
            if (folderBrowserDialog1.ShowDialog() ==
                DialogResult.OK)
            {
                textBox1.Text=
                folderBrowserDialog1.SelectedPath;
            }

            // Кнопка «Обработать» :
            private void button2_Click(object sender, EventArgs e)
            {
                treeView1.Nodes.Clear(); // Очищаем компонент
                // Имя исходного каталога берем из textBox1:
                string DirRoot = textBox1.Text;
                if (!Directory.Exists(DirRoot))
                    MessageBox.Show("Каталог " + DirRoot + " не
найден!", "Внимание");
                else
                {
                    // Начинаем наполнение:

                    treeView1.BeginUpdate();
                    // Создаем новый узел:
                    TreeNode TN = new TreeNode(DirRoot);
                    //Вызываем процедуру рекурсивного обхода:
                    ProcessDirectory(DirRoot, TN);
                    treeView1.Nodes.Add(TN); // Добавляем узел
                    treeView1.Nodes[0].Expand(); //Раскрываем его
                    treeView1.Select(); // Выбираем его
                    treeView1.EndUpdate(); //Завершаем наполнение
                    label1.Text = "";
                }
            }
        }
    }
}

```

```

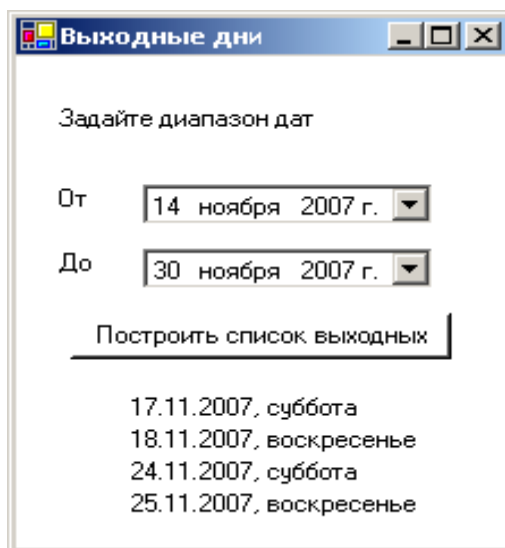
/* Эта процедура- не обработчик события. Пишем
полностью вручную: */
private void ProcessDirectory(string Dir,TreeNode Node)
// Dir - текущий каталог, Node-новый пустой узел
{ string[] SubDir; // Массив для имен подкаталогов
// Заполняем этот массив
SubDir = Directory.GetDirectories(Dir);
// Просматриваем все подкаталоги:
foreach (string SB in SubDir)
{ // Показываем имя текущего каталога
label1.Text = SB;
// Приостанавливаем до прорисовки
Application.DoEvents();
// Создаем узел
TreeNode tempNode = new TreeNode(SB);
// Рекурсивно все повторяем
ProcessDirectory(SB,tempNode);
// Добавляем узел
Node.Nodes.Add(tempNode);
}
}
/* При выборе очередного узла наполняем ListBox
файлами из каталога: */
private void treeView1_AfterSelect(object sender,
                                   TreeViewEventArgs e)
{ listBox1.Items.Clear();
string[] FileList; // Массив имен файлов
// Наполняем массив:
FileList =
    Directory.GetFiles(treeView1.SelectedNode.Text);
// Помещаем имена в список ListBox:
foreach (string fileName in FileList)
listBox1.Items.Add((Path.GetFileName(fileName)).
                                                             ToLower());
} /* Методом ToLower() изменили регистр имен файлов
на нижний */
// Выход:
private void button3_Click(object sender,EventArgs e)
{ if (MessageBox.Show("Вы действительно хотите
выйти?" , "Выход", MessageBoxButtons.YesNo,
MessageBoxIcon.Question)==DialogResult.Yes)this.Close
(); }
}}
Показываем преподавателю.

```

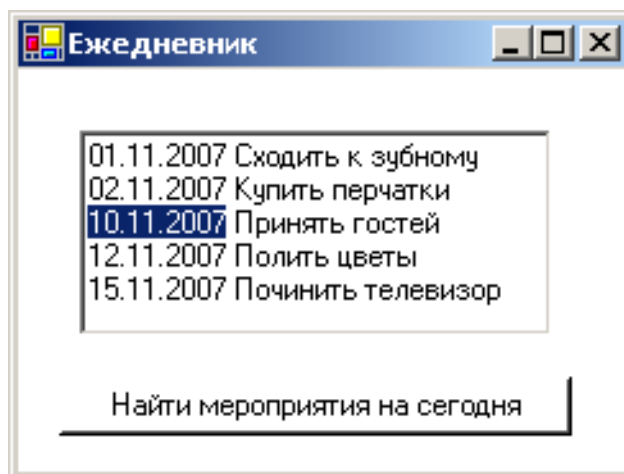
Самостоятельные задания

5.1 Написать программу, в окне которой отображалась бы иерархия типов данных C# (использовать компонент TreeView и его методы AddRoot, AddChild)

5.2. Написать программу, определяющую выходные дни (субботы и воскресенья) в заданном диапазоне дат. Например:



5.3. Написать программу, выделяющую в некотором тексте сегодняшнюю дату. Например:



ЛАБОРАТОРНАЯ РАБОТА № 6

Тема: Работа с файлами

Цели лабораторной работы: ознакомиться и использовать в приложениях компоненты monthCalendar, openFileDialog, saveFileDialog

Учебные вопросы:

1. Использование компонента monthCalendar
2. Запись данных в файл
3. Чтение данных из файла
4. Диалоговые окна openFileDialog и saveFileDialog

Задание 23. Программа «Погода»

Напишем приложение, демонстрирующее выполнение операций с файлами [2]. Программа «Погода» добавляет в базу данных, представляющую собой текстовый файл, сведения о температуре воздуха. Каждая строка данных содержит дату и значение температуры. Если файл данных нет, программа его создает. Кроме того, осуществляется проверка вносимых данных в поле «Температура» - можно заносить только числовые значения. Окно приложения приведено на рис 43:

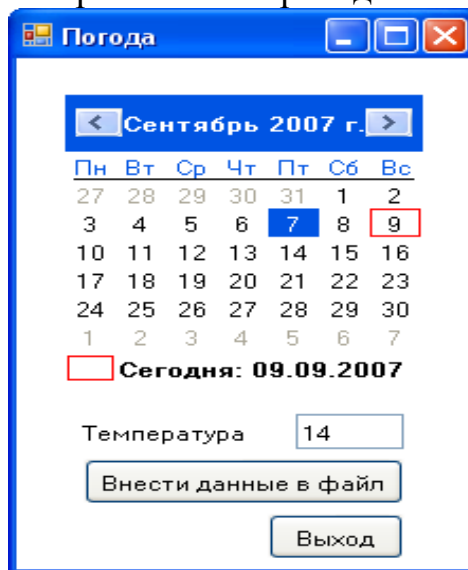


Рис. 43. Окно программы Погода

1. Разместите на форме компоненты monthCalendar, label, textBox и две кнопки.
2. Впишите необходимые объявления и обработчики событий согласно листингу 23:

Листинг 23

```
using System;  
using System.Collections.Generic;  
using System.ComponentModel;
```

```

using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace Погода
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            // Устанавливаем сегодняшнюю дату на
            monthCalendar:
            monthCalendar1.TodayDate =
            System.DateTime.Now;
        }
        // Проверка вносимых символов в поле textBox1:
        private void textBox1_KeyPress(object sender,
                                     KeyPressEventArgs e)
        {
            if (char.IsDigit(e.KeyChar) ||
                (e.KeyChar.ToString()=="-" && textBox1.Text=="") ||
                (e.KeyChar.ToString()=="," &&
                 textBox1.Text.IndexOf(",")==-1))
            {
                e.Handled = false;
            }
            else e.Handled = true;
        }
        // То есть: вносимый символ может быть цифрой, или
        // знаком «-» (если символ первый), или запятой (если
        // запятой до этого не было)

        // Для кнопки « Внести данные в файл » :
        private void button1_Click(object sender, EventArgs e)
        {
            DateTime d;
            // Получим информацию о файле meteo.txt :
            // Создаем объект fi типа FileInfo:
            System.IO.FileInfo fi = new
            System.IO.FileInfo(Application.StartupPath +
                               "\\meteo.txt");

            // Создаем поток для записи:
            System.IO.StreamWriter potok;

            /* Если файл meteo.txt найден, будем добавлять в
            него записи, иначе создаем файл: */

```

```

        if (fi.Exists) potok = fi.AppendText();
        else potok = fi.CreateText();
        /* Переменной d задаем значение, равное выбранной
        в календаре дате: */
        d = monthCalendar1.SelectionStart;

        // переводим значение переменной d в строку с
        кратким форматом записи даты: d.ToShortDateString(),
        заносим эту строку, знак пробела и значение
        температуры в поток:
        potok.WriteLine(d.ToShortDateString() + " " +
        textBox1.Text);
        potok.Close();
        textBox1.Text = "";
    }
    // Для кнопки « Выход » :
    private void button2_Click(object sender, EventArgs e)
    {
        this.Close();
    }
}

```

Запускаем приложение, показываем преподавателю. Найдите созданный вами файл meteo.txt и просмотрите его содержимое.

Замечания:

1. **Application.StartupPath** - путь до текущего каталога приложения. Файл meteo.txt можно было создать и по другому адресу, например:

```

System.IO.FileInfo fi = new System.IO.FileInfo("E:\\
DATA\\meteo.txt");

```

Обратите внимание, в качестве разделителя использован двойной обратный слэш, т.к. чтобы представить специальный символ (коим является “\”) внутри строки, необходимо использовать символ экранирования \.

В C# можно создать **литеральные строки**, в них спец символы (\n, \\, \r, \t и пр.) можно использовать без символа экранирования. Литеральная строка начинается с @. Например строку "E:\\DATA\\meteo.txt" можно заменить на @ "E:\\DATA\\meteo.txt".

2. Можно было задать значение одинаковой температуры для выделенного диапазона дат:

```

...
d = monthCalendar1.SelectionStart;
do

```

```

    {
        potok.WriteLine(d.ToShortDateString() + " "
+ textBox1.Text);
        // Сдвигаем дату на 1 день вперед
        d = d.AddDays(1);
    }
    while (d <= monthCalendar1.SelectionEnd);
    potok.Close();
    textBox1.Text = "";
}

```

Задание 24. Программа «Средняя температура»

Программа «Средняя температура» демонстрирует чтение данных из текстового файла [2]. В качестве источника данных возьмем файл `meteo.txt`, сформированный программой «Погода». После того, как данные будут считаны, для каждого месяца можно рассчитать среднюю температуру. Окно приложения приведено на рис 44:

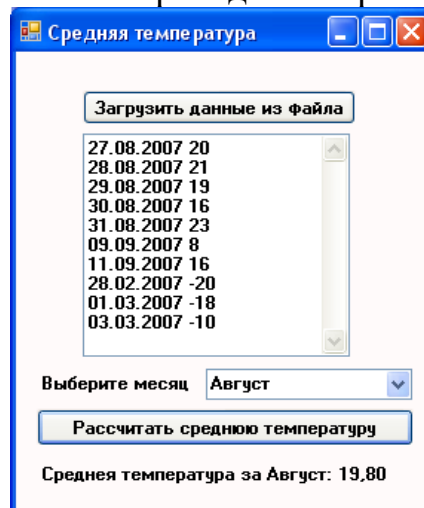


Рис. 44. Окно программы Средняя температура

1. Разместите на форме компоненты `textBox`, `comboBox` (для выбора месяца), два компонента `label` и две кнопки. Подпишите кнопки и компонент `label1`, остальное выполним программно. Впишите необходимые объявления и обработчики событий (листинг 24):

Листинг 24

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace Средняя_температура

```



```

{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            //Задаем многострочный режим
            textBox1.Multiline = true;
            //Задаем ширину и высоту
            textBox1.Width = 180;
            textBox1.Height = 150;
            // Создаем вертикальную полосу прокрутки:
            textBox1.ScrollBars = ScrollBars.Vertical;
            textBox1.Visible = false;
            label2.Text = "";
            // Формируем элементы списка:
            comboBox1.Items.Add("Январь");
            comboBox1.Items.Add("Февраль");
            comboBox1.Items.Add("Март");
            comboBox1.Items.Add("Апрель");
            comboBox1.Items.Add("Май");
            comboBox1.Items.Add("Июнь");
            comboBox1.Items.Add("Июль");
            comboBox1.Items.Add("Август");
            comboBox1.Items.Add("Сентябрь");
            comboBox1.Items.Add("Октябрь");
            comboBox1.Items.Add("Ноябрь");
            comboBox1.Items.Add("Декабрь");
        }

        // Кнопка «Загрузить данные из файла» :
        private void button1_Click(object sender, EventArgs e)
        {
            // Создаем поток для чтения:
            System.IO.StreamReader potok;

            // Пытаемся считать данные из файла meteo.txt
            (путь до него, разумеется, пропишите каждый свой)
            try
            {
                potok = new System.IO.StreamReader(@"E:\DATA\
                C#\Погода\
                Погода\bin\Debug\
                meteo.txt", System.Text.Encoding.GetEncoding(1251));
                //Считываем весь текст до конца
            }
        }
    }
}

```

```

        textBox1.Text = potok.ReadToEnd();
        potok.Close();
        // Делаем поле видимым
        textBox1.Visible = true;
    }
    catch {MessageBox.Show("Файл исходных данных не
найден", "Внимание!");}
}

// Кнопка « Рассчитать среднюю температуру » :
private void button2_Click(object sender,EventArgs e)
{
    //Запоминаем номер выбранного месяца в переменную m:
    int m = comboBox1.SelectedIndex + 1;
    // (+1, т.к. нумерация элементов списка с нуля)
    int n = 0; //количество записей по выбранному месяцу
    int i;
    double sum = 0;
    // В цикле просматриваем все строки:
    for (i = 0; i < textBox1.Lines.Length; i++)
        /* если строка не пустая- сравниваем номер месяца
в ней с m */
        {if (textBox1.Lines[i].Length>0)
            if Convert.ToInt16(textBox1.Lines[i].
Substring(3, 2)) == m)
                { n++;
                    sum += Convert.ToDouble
(textBox1.Lines[i].Substring(textBox1.Lines[i].IndexOf
(" ")));
                }
            }
        /* определили позицию пробела в строке, все символы
после этого пробела преобразовали в вещественное
число и просуммировали с общей суммой */
        }
    if (n == 0)
        label2.Text = "В файле нет данных о температуре
за " + comboBox1.Text;
    else label2.Text = "Средняя температура за " +
comboBox1.Text+ ": " + (sum/n).ToString("N");
}
}
}

```

Замечания:

1. Метод **StreamReader** (имя файла) создает и открывает для чтения поток, соответствующий указанному файлу. Поток открывается

для чтения в формате UTF-8 (Протокол UTF-8 обеспечивает поддержку расширенного набора знаков ASCII и трансляцию UCS-2, 16-разрядного набора знаков Юникод.)

Метод **StreamReader** (имя файла, **encd**) открывает поток для чтения в кодировке, заданной параметром **encd**. Для чтения текста в кодировке Windows 1251 (стандартная кириллическая 8-битная кодировка для русских версий Microsoft Windows) параметр **encd** необходимо инициализировать значением `System.Text.Encoding.GetEncoding(1251)`.

2. Метод **S.Substring(n)** выделяет подстроку из строки **S** начиная с позиции **n**. Метод **S.Substring(n, k)** выделяет из строки **S** подстроку длиной **k** символов начиная с позиции **n**.

3. Метод **S.IndexOf(S1)** определяет положение подстроки **S1** в строке **S**. Если указанной подстроки нет, возвращается -1.

Задание 25. Программа «Диалоговые окна»

Программа «Диалоговые окна» демонстрирует использование компонентов `OpenFileDialog` и `SaveFileDialog`. На форме будет располагаться меню с пунктами «Создать файл», «Открыть файл», «Сохранить файл», «Выход» (рис. 45):

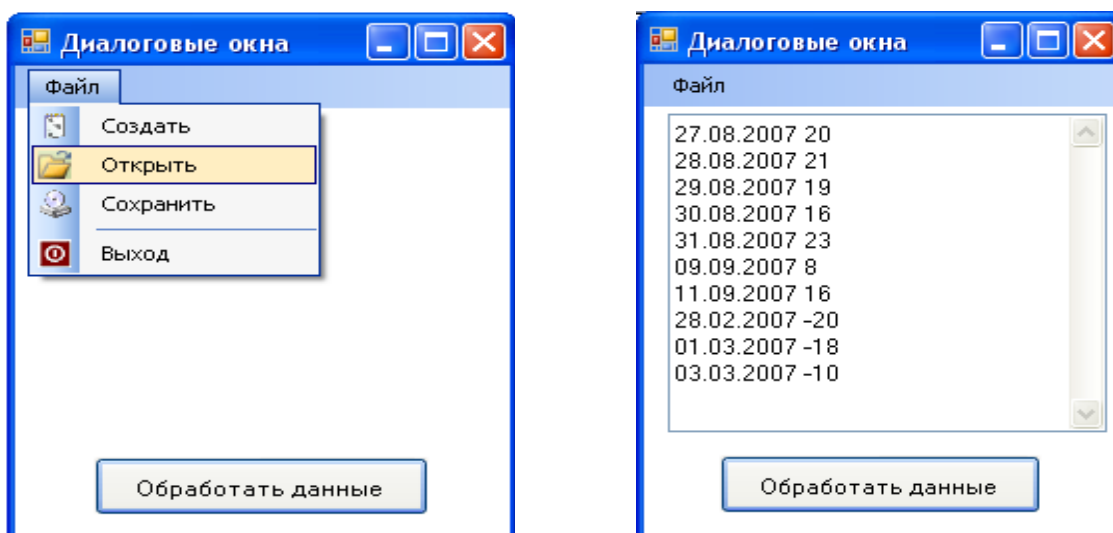


Рис. 45. Открытие файла `meteo.txt`

Через пункты «Создать»/ «Сохранить» во внешней памяти можно создать свой текстовый файл. При открытии созданного ранее файла `meteo.txt` предусмотрена процедура обработки его данных, а именно поиск наибольшей и наименьшей температуры (рис. 46) :

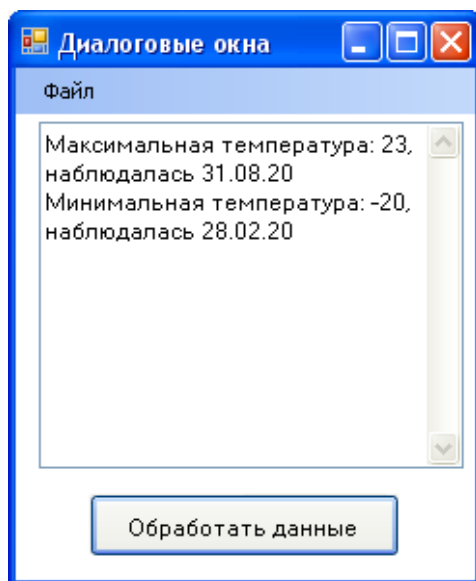


Рис. 46. Обработка данных

1. Расположите на форме компоненты MenuStrip, OpenFileDialog, SaveFileDialog и кнопку «Обработать данные».
2. Создайте пункты меню согласно рис 45 (картинки рядом с названиями пунктов установлены через их свойства Image).

Изучите листинг 25, впишите необходимые объявления и обработчики событий:

Листинг 25

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace Файлы
{
    public partial class Form1 : Form
    {
        string fname = "";
        public Form1()
        {
            InitializeComponent();
            textBox1.Multiline = true;
            textBox1.Height = 180;
            textBox1.Width = 220;
            textBox1.ScrollBars = ScrollBars.Vertical;
            textBox1.Visible = false;
            openFileDialog1.DefaultExt = ".txt";
            openFileDialog1.Filter = "Текстовый документ|*.txt";
        }
    }
}
```

```

        saveFileDialog1.DefaultExt = "txt";
        saveFileDialog1.Filter = "Текстовый
документ|*.txt";
    }

    private void создатьToolStripMenuItem_Click(object
sender, EventArgs e)
    {
        fname = "";
        textBox1.Text = "";
        textBox1.Visible = true;
    }

    private void открытьToolStripMenuItem_Click(object
sender, EventArgs e)
    {
        openFileDialog1.FileName = "";
        if (openFileDialog1.ShowDialog() == DialogResult.OK)
            fname = openFileDialog1.FileName;
        try
        {
            System.IO.StreamReader potok = new
                System.IO.StreamReader(fname);
            textBox1.Text = potok.ReadToEnd();
            textBox1.Visible = true;
            potok.Close();
        }
        catch { MessageBox.Show("Ошибка чтения файла");
        }
    }

    private void выходToolStripMenuItem_Click(object
sender, EventArgs e)
    {
        this.Close();
    }

    private void сохранитьToolStripMenuItem_Click(object
sender, EventArgs e)
    {
        if (saveFileDialog1.ShowDialog() == DialogResult.OK)
            fname = saveFileDialog1.FileName;
        try { System.IO.FileInfo fi = new
            System.IO.FileInfo(fname);
            System.IO.StreamWriter potok = fi.CreateText();
            potok.Write(textBox1.Text);
            potok.Close();
        }
    }

```

```

        catch { MessageBox.Show("Ошибка сохранения файла");
    }
}
// Кнопка «Обработать данные»
private void button1_Click(object sender, EventArgs e)
{
    int i;
    double max, min;
    string m1=""; //для промежуточного сохранения дат
    string m2="";
    try
    {
        // за min и max принимаем первое значение
        температуры:
        max = min =
        Convert.ToDouble(textBox1.Lines[0].Substring(11));
        for (i = 0; i < textBox1.Lines.Length; i++)
            if (textBox1.Lines[i].Length > 0)
                { if (Convert.ToDouble(textBox1.Lines[i].
Substring(11)) > max)
                    { max =
Convert.ToDouble(textBox1.Lines[i]. Substring(11));
                    m1 = textBox1.Lines[i].Substring(0,8);
                }
                if (Convert.ToDouble(textBox1.Lines[i].
Substring(11)) < min)
                    { min =
Convert.ToDouble(textBox1.Lines[i]. Substring(11));
                    m2 = textBox1.Lines[i].Substring(0, 8)
                }
            }
        textBox1.Text = "Максимальная температура: " +
Convert.ToString(max) + ", наблюдалась " + m1+"
Минимальная температура: " + Convert.ToString(min) +
", наблюдалась " + m2;
    } // try
    catch {MessageBox.Show("Некорректные данные",
"Внимание!"); }
}
}
}

```

Запустите приложение, откройте файл meteo.txt, найдите наименьшую и наибольшую температуру, сохраните результаты во внешнем файле result.txt. Покажите преподавателю.

Замечания:

1. Свойство **DefaultExt** задает расширение файла по умолчанию, если пользователь не укажет его явно при открытии или сохранении файла.

2. Свойство **Filter** задает фильтр (маску) для имени файла. Например значение Текст| *.txt указывает, что в списке файлов надо отобразить только файлы с расширением txt.

Задание 26. Программа «Запрос информации по счетам»

Напишем программу, которая по введенной фамилии клиента и его личного пароля будет выдавать информацию о состоянии его счета (данные о клиентах, их кодах доступа и суммах берутся из внешнего текстового файла).

Информация по счетам

Просмотреть состояние счета

Login: Иванов А.Н.

Password: xxxxxx

OK Очистить

Уважаемый Иванов А.Н., на вашем счете 300 рублей

Рис. 47. Окно под профилем клиента

При входе в систему под логином administrator, становится доступной для просмотра и изменения информация по всем клиентам (рис 48):

Информация по счетам

Просмотреть состояние счета

Login: administrator

Password: xxxxxx

OK Очистить

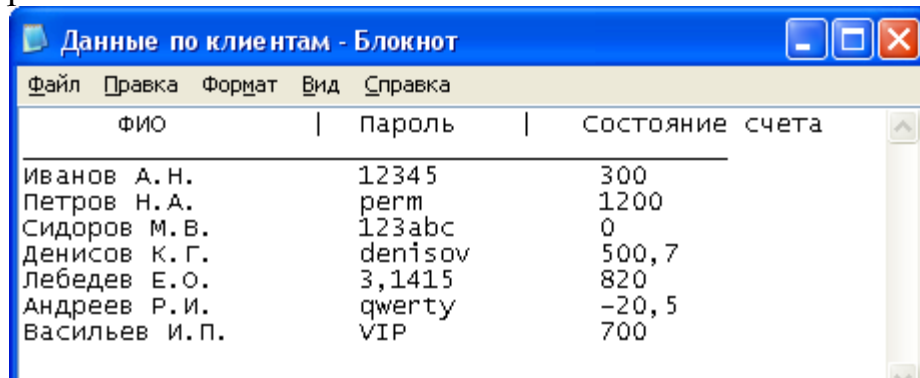
Информация по всем счетам

| ФИО | Пароль | Состояние счета |
|---------------|---------|-----------------|
| Иванов А.Н. | 12345 | 300 |
| Петров Н.А. | perm | 1200 |
| Сидоров М.В. | 123abc | 0 |
| Денисов К.Г. | denisov | 500,7 |
| Лебедев Е.О. | 3,1415 | 820 |
| Андреев Р.И. | qwerty | -20,5 |
| Васильев И.П. | VIP | 700 |

Сохранить изменения в исходном файле

Рис. 48. Окно под профилем администратора

1. Создайте внешний текстовый файл «Данные по клиентам.txt». Заполните его вручную начальными данными, например:



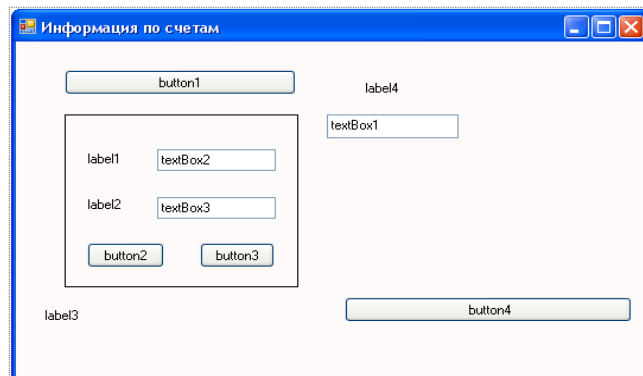
| ФИО | Пароль | Состояние счета |
|---------------|---------|-----------------|
| Иванов А.Н. | 12345 | 300 |
| Петров Н.А. | perm | 1200 |
| Сидоров М.В. | 123abc | 0 |
| Денисов К.Г. | denisov | 500,7 |
| Лебедев Е.О. | 3,1415 | 820 |
| Андреев Р.И. | qwerty | -20,5 |
| Васильев И.П. | VIP | 700 |

Рис.49. Исходный текстовый файл

Для удобства последующей обработки данных, пароли вводите, например, с 21-ой позиции в строке (нумерацию позиций в строке будем вести с нуля), суммы- с 36-ой позиции.

Сохраните, закройте. Запомните путь до этого файла.

2. Создайте новый проект в Visual Studio. Разместите на форме четыре кнопки, три поля для ввода/вывода, четыре надписи label (рис 50). Используя компонент Panel, объедините компоненты label1, label2, textBox2, textBox3, button2, button3 в одну группу:



The form contains the following controls:

- button1 (top left)
- label4 (top right)
- textBox1 (top right, below label4)
- A container panel containing:
 - label1 (left)
 - textBox2 (right of label1)
 - label2 (left)
 - textBox3 (right of label2)
 - button2 (bottom left of panel)
 - button3 (bottom right of panel)
- label3 (bottom left)
- button4 (bottom right)

Рис. 50. Конструктор формы

3. Подпишите все кнопки и компоненты label1 и label2 согласно рисунку 48. Остальное выполним программно (см листинг 26).

Листинг 26

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace Пароль
```



```

{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            panel1.Visible = false;
            textBox1.Multiline = true;
            textBox1.Width = 270;
            textBox1.Height = 160;
            // (Размеры при необходимости измените)
            textBox1.Visible = false;
            // Задаем вид вводимых символов
            textBox3.PasswordChar = '*';
            label3.Text = "";
            label4.Text = "";
            button4.Visible = false;
        }
        // Кнопка «Просмотреть состояние счета»
        private void button1_Click(object sender, EventArgs e)
        {
            panel1.Visible = true;
            // Создаем поток для чтения:
            System.IO.StreamReader potokR;

            /* Пытаемся считать данные из файла «Данные по
            клиентам.txt» Путь до него пользователю программы
            знать не нужно, поэтому диалоговое окно открытия
            файла использовать не будем (путь пропишем
            программно). Всю информацию помещаем в поле textBox1
            */
            try
            {
                // Путь до файла укажите каждый свой
                potokR = new System.IO.StreamReader(@"E:\
                DATA\C#\Пароль\Данные по
                клиентам.txt", System.Text.Encoding.GetEncoding(1251));
                textBox1.Text = potokR.ReadToEnd();
                potokR.Close();
            }
            catch { MessageBox.Show("Ошибка чтения данных!",
            "Внимание!"); }
        }
    }
}

```

```

// Кнопка «OK»
private void button2_Click(object sender, EventArgs e)
{
    /* Нулевая и первая строка файла-заголовки
    таблицы, их не смотрим (поэтому цикл с двух).
    Если в строке есть текст, сравниваем логин и пароль в
    полях textBox2 textBox3 с соответствующими данными в
    файле. Метод Trim() отбрасывает все пробелы с начала
    и конца подстрок, в которых находятся ФИО, пароли и
    суммы. Начальный файл мы сформировали так, что ФИО
    клиента находится где-то в промежутке с нулевой до 14
    позиции, пароль - с двадцать первой до тридцать первую
    позицию, сумма - примерно с тридцать шестой позиции
    (начать смотреть можно и раньше)*/

    int flag=0;
    //(flag =1, если логин подошел к паролю, иначе
    flag=0)
    try
    {for (int i = 2; i < textBox1.Lines.Length; i++)
        if (textBox1.Lines[i].Length > 0)
            if ((textBox1.Lines[i].Substring(0,14).Trim()
== textBox2.Text)
                && (textBox1.Lines[i].Substring(21,10).Trim()
== textBox3.Text))
            {
                label3.Text = "Уважаемый "+textBox2.Text+ ",
на вашем счете" + textBox1.Lines[i].Substring(35).
Trim()+ " рублей";
                flag = 1;
            }

        /* Пусть логин администратора системы-
        administrator, пароль- master */
        if (textBox2.Text == "administrator" &&
textBox3.Text == "master")
            { /* Становятся доступны для просмотра все
            данные, кнопка «Сохранить изменения в исходном
            файле», надпись «Информация по всем счетам» */
                label4.Text = "Информация по всем счетам";
                textBox1.Visible = true;
                button4.Visible = true;
                flag = 1;
            }
        // Если логин не подошел к паролю:

```

```

        if (flag == 0) label3.Text = "Проверьте имя
пользователя и пароль!";
    }
    catch { MessageBox.Show("Ошибка обработки
данных!", "Внимание!"); };
}

// Кнопка «Очистить»
private void button3_Click(object sender, EventArgs e)
{
    textBox2.Text = "";
    textBox3.Text = "";
    label3.Text = "";
    label4.Text = "";
    textBox1.Visible = false;
    button4.Visible = false;
    textBox2.Focus();
}
// Кнопка « Сохранить изменения в исходном файле »
private void button4_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Перезаписать файл?",
"Сохранение", MessageBoxButtons.YesNo,
MessageBoxIcon.Question) == DialogResult.Yes)
    {
        System.IO.StreamWriter potokW;
        try
        {
            potokW = new
System.IO.StreamWriter(@"E:\DATA\C#\Пароль\Данные по
клиентам.txt", false, System.Text.Encoding.GetEncoding
(1251));
            potokW.Write(textBox1.Text);
            potokW.Close();
            MessageBox.Show("Данные были обновлены", "Внимание!");
        }
        catch { MessageBox.Show("Ошибка сохранения
файла", "Внимание!"); }
    }
}
}
}

```

Здесь метод **StreamWriter (f, a, cod)** создает и открывает для записи поток, соответствующий файлу *f*. Поток открывается для записи в

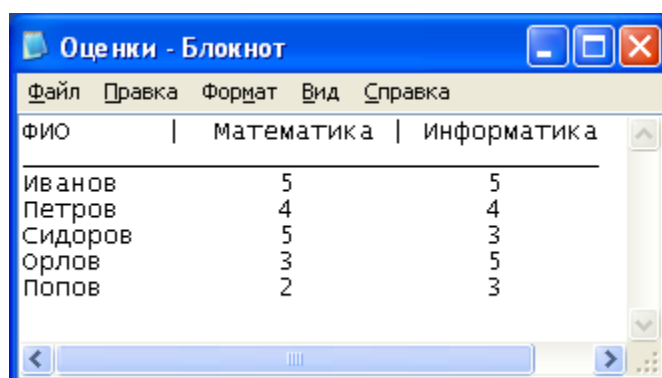
кодировке, заданной параметром *cod*. Параметр *a* задает режим записи: true-добавление в файл, false – перезапись.

Метод **S.Trim()** возвращает строку, полученную путем отбрасывания из строки *S* пробелов, находящихся в начале и конце.

Запустите. Потестируйте и с позиции администратора, и с позиции клиента. Сохраните, покажите преподавателю.

Самостоятельные задания

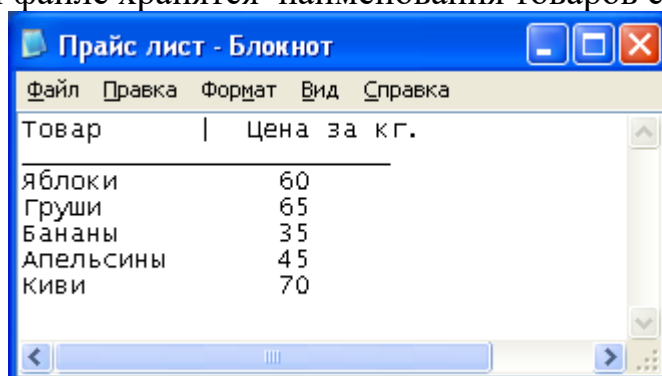
6.1. Во внешнем файле хранятся оценки студентов вида



| ФИО | Математика | Информатика |
|---------|------------|-------------|
| Иванов | 5 | 5 |
| Петров | 4 | 4 |
| Сидоров | 5 | 3 |
| Орлов | 3 | 5 |
| Попов | 2 | 3 |

Рассчитать для каждого студента средний балл.

6.2. Во внешнем файле хранятся наименования товаров с ценами вида:



| Товар | Цена за кг. |
|-----------|-------------|
| яблоки | 60 |
| груши | 65 |
| бананы | 35 |
| апельсины | 45 |
| киви | 70 |

Определить, какие товары попадают по цене в диапазон от *X* до *Y* рублей.

ЛАБОРАТОРНАЯ РАБОТА № 7

Тема: Работа с графикой

Цели лабораторной работы: ознакомиться и использовать в приложениях различные методы формирования изображений

Учебные вопросы:

1. Методы DrawLines() DrawCurve() для отрисовки линий и кривых
2. Методы DrawString (), DrawLine () DrawRectangle () для формирования текста, изображения линии и прямоугольника
3. Методы FillPie (), DrawPie() для формирования круговой диаграммы
4. Формирование изображения из нескольких битовых образов

Задание № 27. Программа « Отрисовка линий и кривых »

Рассмотрим применение методов DrawLines() DrawCurve() для отрисовки линий и кривых [5].

Метод DrawLines() соединяет заданные точки прямыми линиями (рис. 51):

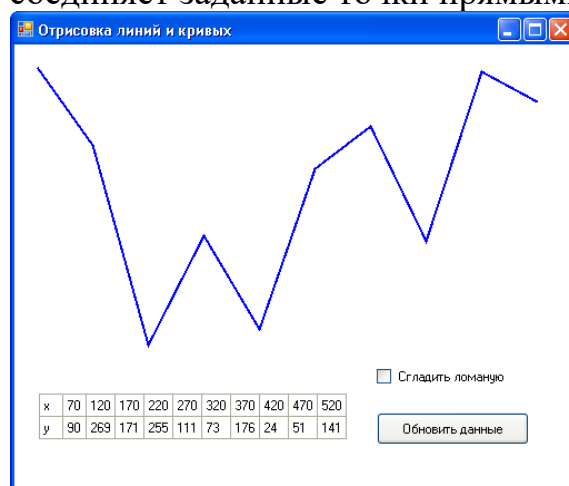


Рис. 51. Отрисовка ломанной линии

метод DrawCurve() соединяет точки кривыми линиями, используя сплайн-интерполяцию (рис. 52):

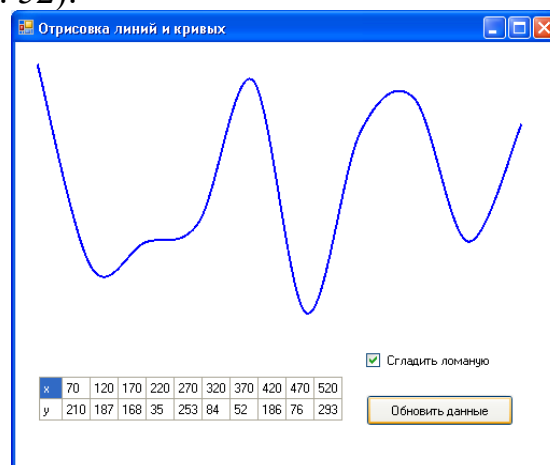


Рис. 52. Отрисовка кривой

Координаты точек зададим через массив Points: координаты x – фиксированные, с шагом 50; координаты y – случайные числа.

Расположите на форме dataGridView, checkbox, button.

Задайте компоненту dataGridView следующие свойства:

AutoSizeColumnsMode -> AllCells, AutoSizeRowsMode -> AllCells

Пишем код:

Листинг 27

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace Линии
{
    public partial class Form1 : Form
    {
        public Point[] Points = new Point[10];
        public Form1()
        {
            InitializeComponent();
            this.BackColor = System.Drawing.Color.White;
            this.Text = "Отрисовка линий и кривых";
            this.ClientSize = new
System.Drawing.Size(450, 400);
            // Цвет фона таблицы задаем такой же, как у формы:
            dataGridView1.BackgroundColor=this.BackColor;
            // Убираем внешнюю границу компонента:
            dataGridView1.BorderStyle =BorderStyle.None;
            dataGridView1.ColumnCount = 11;
            dataGridView1.RowCount = 2;
            // убираем шапку таблицы:
            dataGridView1.ColumnHeadersVisible = false;
            dataGridView1.RowHeadersVisible = false;
            // Убираем полосу прокрутки:
            dataGridView1.ScrollBars = ScrollBars.None;
            // Задаем ширину таблицы
            dataGridView1.Width = 300;
        }

        private void button1_Click(object sender,EventArgs e)
        {
            int x = 20, y = 20;
            Random r = new Random();
            dataGridView1.Rows[0].Cells[0].Value = "x";
            dataGridView1.Rows[1].Cells[0].Value = "y";
        }
    }
}
```

```

    for (int i = 0; i < 10; i++)
    {
        Points[i] = new Point(x, y);
        x += 50;
        y = r.Next(300);
        dataGridView1.Rows[0].Cells[i + 1].Value =
x.ToString();
        dataGridView1.Rows[1].Cells[i + 1].Value =
y.ToString();
    }
    this.Refresh();
}
// Отрисовка формы:
private void Form1_Paint(object sender, PaintEventArgs e)
{ if (checkBox1.Checked)
    { // Рисуем кривую линию
      Pen P = new Pen(Color.Blue, 2);
      e.Graphics.DrawCurve(P, Points);
    }
  else
    { // Рисуем ломанную линию
      Pen p = new Pen(Color.Blue, 2);
      e.Graphics.DrawLines(p, Points);
    }
}
}
}

```

Задание № 28. Программа «График изменения курса доллара»

Напишем программу, отображающую график изменения курса доллара[2]:

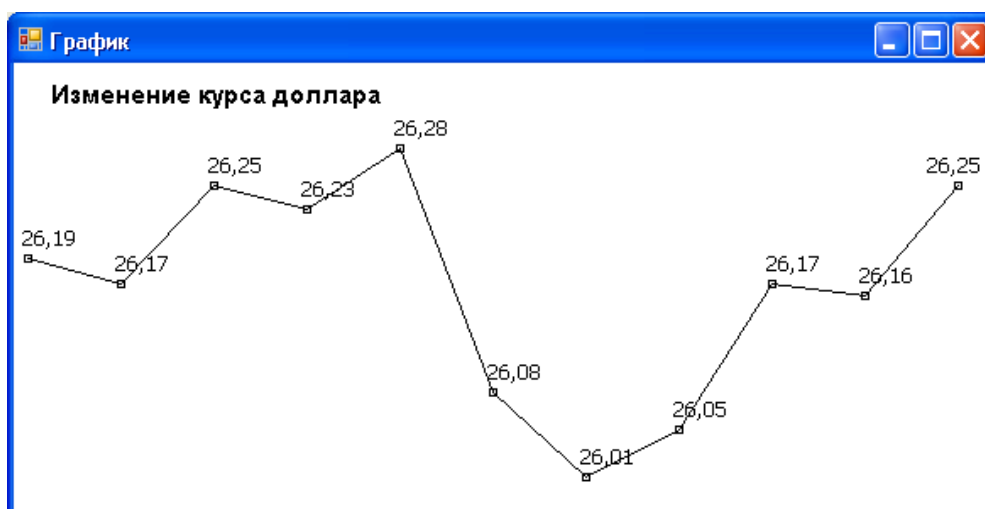


Рис. 53. График изменения курса доллара

Входные данные загружаются из предварительно сформированного файла Данные.txt :

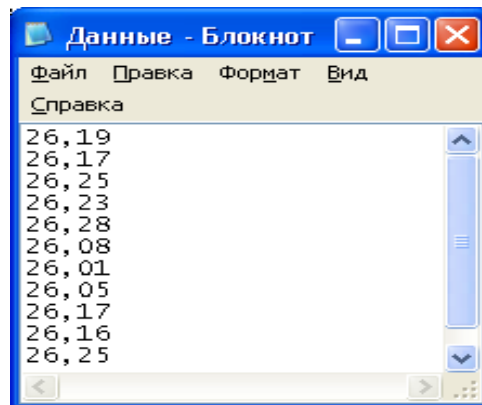


Рис.54. Файл с входными данными

Кроме того, сделаем так, чтобы при изменении размеров формы график автоматически перерисовывался, занимая всю предоставленную область формы, не нарушая пропорций.

Создаем новое приложение, не помещая на форму никаких компонентов, вписываем код:

Листинг 28

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO; // Этот модуль подключаем вручную
namespace График
{
    public partial class Form1 : Form
    {
        // массив Kurs будет содержать курсы доллара:
        private double[] Kurs;
        // Объявляем поток для чтения:
        public System.IO.StreamReader potok;
        public Form1()
        {
            InitializeComponent();
            // задаем цвет формы:
            this.BackColor= System.Drawing.Color.White;
            try
            {
                //Создаем объект stream типа FileStream:
```



```

    FileStream stream = new FileStream("E:\\DATA\\C#\\
График\\ Данные.txt", FileMode.Open, FileAccess.Read);
/* (естественно, файл уже должен быть создан по этому
адресу) */
    // создаем поток для чтения и считываем данные;
    potok = new System.IO.StreamReader(stream);

/* подсчитываем в переменной n количество записей в
файле: */
    int n = 0;
    string t = potok.ReadLine();
    while (t != null)
    {
        t = potok.ReadLine();
        n++;
    }

    / Устанавливаем указатель на начало потока (т.к.
методы ReadLine() сместили его в конец файла): */
    stream.Seek(0, SeekOrigin.Begin);
    // инициализируем массив
    Kurs = new double[n];
/* считываем записи о курсах доллара в переменную t,
затем конвертируем в числовой формат и переносим в
массив Kurs: */
    int i = 0;
    t = potok.ReadLine();
    while (t != null)
    {
        // записываем считанное число в массив
        Kurs[i] = Convert.ToDouble(t);
        t = potok.ReadLine();
        i++;
    }

    // закрываем поток
    potok.Close();
    // если записей нет
    if (Kurs.Length == 0)
        MessageBox.Show("Файл исходных данных пуст", "График",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    else this.Paint += new
        PaintEventHandler(drawDiagram);
/* процедура прорисовки диаграммы drawDiagram будет
вызываться каждый раз, когда будет происходить
перерисовка формы */
} // (закрыли блок try)

```

```

        // Обработка исключений:
        // файл исходных данных не найден:
        catch (System.IO.FileNotFoundException exc)
        {
            MessageBox.Show("Файл исходных данных не найден"
+ "\n" + exc.Message);
        }
        // ошибка формата исходных данных
        catch (FormatException exc)
        {
            MessageBox.Show("Ошибка формата исходных
данных" + "\n" + exc.Message);
        }
        // прочие исключения (выводим текст ошибки):
        catch (Exception exc)
        {
            MessageBox.Show(exc.Message);
        }
    }

    // Вписываем процедуру вручную:
private void drawDiagram(object sender, PaintEventArgs e)
    { // Графическая поверхность:
        Graphics g=e.Graphics;
        // Задаем шрифт заголовка:
        Font font1=new Font ("Arial",11,FontStyle.Bold);
        // Задаем шрифт для подписи значений:
        Font font2=new Font ("Tahoma",10,FontStyle.Regular);

        /*Выводим заголовок черной кистью, начиная с
координат (20,10): */
        g.DrawString("Изменение курса доллара", font1,
Brushes.Black,20,10);

        /* Определяем расстояние между узлами графика по оси
х: */
        int d=(int)((this.ClientSize.Width-20)/(Kurs.Length-1));
        /* Ищем максимальное и минимальное значения (от этих
значений будем отталкиваться при определении
вертикальных координат узлов) */
        double maxKurs=Kurs[0];
        double minKurs=Kurs[0];
        for (int i=0;i<Kurs.Length;i++)
        {
            if (Kurs[i]>maxKurs) maxKurs=Kurs[i];

```

```

        if (Kurs[i]<minKurs) minKurs=Kurs[i];
    }
    // Рисуем график:
    int x1,y1,x2,y2;
    // Задаем координаты первой точки:
    x1=8;
    y1=this.ClientSize.Height-20-(int)
((this.ClientSize.Height -70)*(Kurs[0]-minKurs) /
(maxKurs-minKurs));

    // Рисуем прямоугольник- метку первого узла:
    g.DrawRectangle(Pens.Black,x1-2,y1-2,4,4);

    // Определяем координаты следующих точек:
    for (int i = 1; i <Kurs.Length; i++)
        {
            x2 = 8 + i * d;
            y2 = this.ClientSize.Height - 20 - (int)
((this.ClientSize.Height - 70) *(Kurs[i] - minKurs) /
(maxKurs - minKurs));
            // Рисуем прямоугольник- метку следующего узла:
            g.DrawRectangle(Pens.Black,x2 - 2,y2 - 2, 4, 4);
            // Соединяем узлы линией:
            g.DrawLine(Pens.Black,x1,y1,x2,y2);
            // Подписываем значения:
            g.DrawString(Convert.ToString(Kurs[i-1]),
font2, Brushes.Black,x1-5,y1-20);
            x1 = x2;
            y1 = y2;
        } // для for
    // Подписываем последний узел графика:
    g.DrawString(Convert.ToString(Kurs[Kurs.Length-1]),
font2, Brushes.Black, x1 - 20, y1 - 20);
} // для drawDiagram

// При изменении размеров окна перерисовываем форму:
private void Form1_SizeChanged(object sender, EventArgs e)
{
    this.Refresh();
}
}
}

```

Здесь метод **DrawString (S,Font, Brush,x,y)** выводит на графическую поверхность строку S. Параметр Font определяет шрифт; Brush – цвет символов; x и y – точку, от которой будет выведен текст.

Метод **DrawRectangle (Pen, x,y,w,h)** рисует контур прямоугольника. Параметр *Pen* определяет цвет, толщину и стиль границы; параметры *x* и *y* – координаты левого верхнего угла; *w* и *h* задают ширину и высоту прямоугольника.

Метод **DrawLine (Pen, x1,y1,x2,y2)** рисует линию. Параметр *Pen* определяет цвет, толщину и стиль линии; параметры *x1*, *y1*, *x2*, *y2* – координаты точек начала и конца линии.

Задание № 29. Программа «Круговая диаграмма»

Напишем программу, отображающую диаграмму популярности интернет-браузеров [2]. Входные данные загружаются из предварительно сформированного файла Данные.txt (рис. 56), определяется их процентное соотношение, результат наглядно отображается через диаграмму.

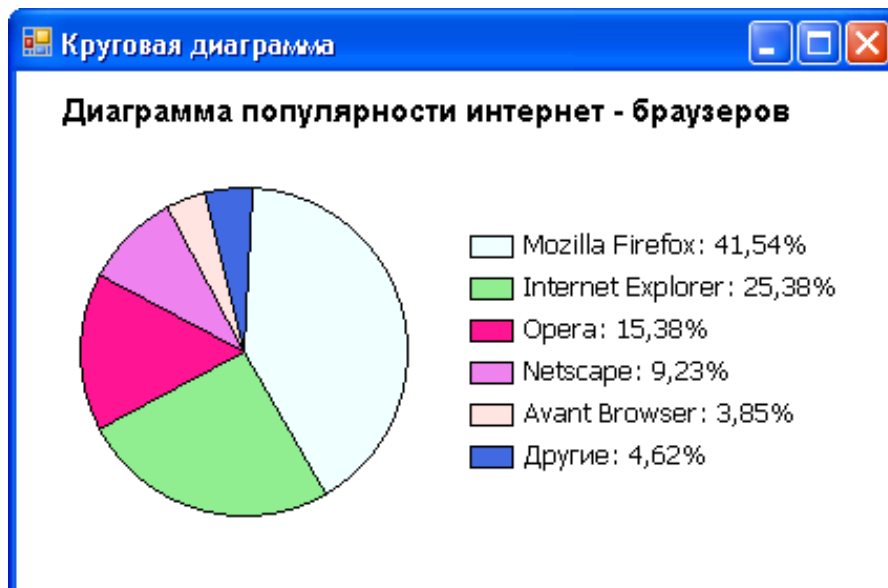


Рис. 55. Круговая диаграмма

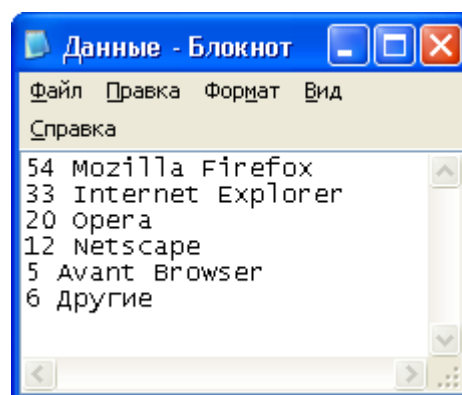


Рис. 56. Файл с исходными данными

Листинг 29

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;
namespace Круговая_диаграмма
{
    public partial class Form1 : Form
    {
        // Объявляем массив численных значений:
        private double[] K;
        // Объявляем массив подписей:
        private string[] L;
        // Сумма элементов массива чисел:
        private double sum;
        // Количество элементов в файле:
        private int n = 0;
        // Объявляем поток для чтения:
        public System.IO.StreamReader potok;
        public Form1()
        {
            InitializeComponent();
            // задаем цвет формы:
            this.BackColor = System.Drawing.Color.White;
            try
            {
                // Создаем объект stream типа FileStream:
                FileStream stream = new FileStream("E:\\DATA \\C# \\
Круговая диаграмма\\Данные.txt", FileMode.Open,
                FileAccess.Read);
                //(естественно, файл уже должен быть создан)
                // создаем поток для чтения и считываем данные:
                potok = new
                System.IO.StreamReader(stream,
                System.Text.Encoding.GetEncoding(1251));
                //подсчитываем в переменной n количество записей в файле:
                string t = potok.ReadLine();
                while (t != null)
                {
                    t = potok.ReadLine();
                    n++;
                }
            }
        }
    }
}
```

```

    /* Устанавливаем указатель на начало потока (т.к.
    методы ReadLine() сместили его в конец файла) */
    stream.Seek(0, SeekOrigin.Begin);
    // инициализируем массивы
    K = new double[n];
    L = new string[n];
    int i = 0;
    t = potok.ReadLine();
    while (t != null)
    {
        // записываем числа до пробела в массив:
        K[i] = Convert.ToDouble(t.Substring(0,
t.IndexOf(" ")));
        // Прибавляем к общей сумме:
        sum += K[i];
        // считываем текст после пробела, заносим в массив L:
        L[i] = t.Substring(t.IndexOf(" "));
        // Считываем следующую строку
        t = potok.ReadLine();
        i++;
    } // для while
    // закрываем поток
    potok.Close();
    // если записей нет
    if (K.Length == 0)
        MessageBox.Show("Файл исходных данных пуст");
    else this.Paint += new PaintEventHandler(drawDiagram);
    } // для try
    // Обработка исключений:
    // файл исходных данных не найден:
    catch (System.IO.FileNotFoundException exc)
    {
        MessageBox.Show("Файл исходных данных не найден"
+ "\n" + exc.Message);
    }
    // ошибка формата исходных данных
    catch (FormatException exc)
    {
        MessageBox.Show("Ошибка формата исходных
данных" + "\n" + exc.Message);
    }
    // прочие исключения (выводим текст ошибки):
    catch (Exception exc)
    {

```

```

        MessageBox.Show(exc.Message);
    }
} // закрыли описание public Form1()

// Вписываем процедуру:
private void drawDiagram(object sender, PaintEventArgs e)
{
    // Графическая поверхность:
    Graphics g = e.Graphics;
    // Задаем шрифт заголовка:
    Font font1 = new Font("Arial", 11,
    FontStyle.Bold);
    // Задаем шрифт легенды:
    Font font2 = new
    Font("Tahoma", 10, FontStyle.Regular);
    /* Выводим заголовок черной кистью, начиная с
    координат (20,10): */
    g.DrawString("Диаграмма популярности интернет -
    браузеров", font1, Brushes.Black, 20, 10);
    // диаметр сектора круговой диаграммы
    int d = (int)(this.ClientSize.Height * 4 / 5 - 40);
    // Начальная точка:
    int x = 30;
    int y = (int)((this.ClientSize.Height - d) / 2 + 10);
    /* координаты верхнего левого угла местоположения
    легенды */
    int lx = 60 + d;
    int ly = y + (int)((d - n * 20 + 10) / 2);
    // начальная точка дуги сектора
    int M = -90;
    // угол дуги сектора
    int alpha;
    // кисть для заливки сектора диаграммы
    Brush fBrush = Brushes.White;
    // рисуем диаграмму
    for (int i = 0; i < K.Length; i++)
    {
        // Определяем угол:
        alpha = (int)(360 * (K[i] / sum)) + 1;
        /* определяем цвет секторов (допустим,
        предполагается максимум 10 секторов) */
        switch (i)
        {
            case 0: fBrush = Brushes.Azure; break;
            case 1: fBrush = Brushes.LightGreen; break;

```

```

        case 2: fBrush = Brushes.DeepPink; break;
        case 3: fBrush = Brushes.Violet; break;
        case 4: fBrush = Brushes.MistyRose; break;
        case 5: fBrush = Brushes.RoyalBlue; break;
        case 6: fBrush = Brushes.SteelBlue; break;
        case 7: fBrush = Brushes.Chocolate; break;
        case 8: fBrush = Brushes.LightGray; break;
        case 9: fBrush = Brushes.Gold; break;
    }
    // сектор диаграммы
    g.FillPie(fBrush, x, y, d, d, M, alpha);
    // граница сектора
    g.DrawPie(System.Drawing.Pens.Black, x, y, d, d, M, alpha);
    // начальная точка дуги для следующего сектора
    M = M + (int) (360 * (K[i] / sum)) + 1;
    // Легенда:
    // рисуем закрашенный прямоугольник:
    g.FillRectangle(fBrush, lx, ly + i * 20, 20, 10);
    // рисуем черный контур прямоугольника:
    g.DrawRectangle(System.Drawing.Pens.Black, lx, ly
+ i * 20, 20, 10);
    /* подпись легенды (текст из массива L + значение
в процентах): */
    g.DrawString(L[i]
+ ":" + Convert.ToDouble(K[i] / sum * 100).ToString("N") +
"%", font2, System.Drawing.Brushes.Black, lx + 20, ly +
i * 20 - 4);
    } // закрыли цикл for
} // закрыли процедуру drawDiagram

/* При изменении размеров окна перерисовываем форму
полностью: */
private void Form1_SizeChanged(object sender,
                                EventArgs e)
{
    this.Refresh();
}
}
}

```

Здесь метод **FillPie (Brush, x, y, d, d, M, alpha)** – закрашивает внутреннюю область сектора эллипса, вписанного в прямоугольник (x, y – координаты левого верхнего угла; d, d – ширина и высота прямоугольника), M и alpha – радиальные лучи. Параметр Brush определяет цвет и стиль закрашки.

Метод **DrawPie (Pen, x, y, d, d, M, alpha)** - рисует дугу-границу сектора эллипса. Параметр Pen определяет цвет, толщину и стиль границы;

Задание № 30. Программа « Полет в облаках»

Программа демонстрирует формирование изображения из нескольких битовых образов (самолет движется на фон неба) [2]. Самолет перемещается по горизонтальной координате с разной скоростью. Вертикальная координата изменяется произвольным образом. Изображения фона и объекта загружаются из файлов (см. рис 57)



Рис. 57. Полет в облаках

Понадобится компонент Timer, поместите его на форму.

Листинг 30

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace Полет
{
    public partial class Form1 : Form
    {
        /* переменные sky и plane будут содержать
        изображения неба и самолета */
        private System.Drawing.Bitmap sky, plane;

        // рабочая поверхность
        private Graphics g;
```

```

        /* приращение координаты по X определяет
        скорость полета - чем больше будет приращение dx за
        некоторый интервал времени, тем выше скорость */
        private int dx;
        // прямоугольная область, в которой находится самолет
        private Rectangle rct;

        // генератор случайных чисел
        private System.Random r;
        public Form1()
        {
            InitializeComponent();
            this.Text = "Полет в облаках";
        }

        // Загрузка формы:
        private void Form1_Load(object sender, EventArgs e)
        { try
            { // загружаем битовые образы
              // небо:
              sky = new Bitmap(@"E:\DATA\C#\sky.bmp");
              // самолет:
              plane = new Bitmap(@"E:\DATA\C#\plane.bmp");
              plane.MakeTransparent();
              /*Системная палитра определяет один цвет в качестве
              стандартно прозрачного, или альфа цвета. Метод
              MakeTransparent() делает стандартно прозрачные цвета
              просвечиваемыми для данного объекта Bitmap. Иными
              словами, пусть будет видно только самолет, без его
              белого прямоугольного фона */
            }
            catch (Exception exc)
            {
                MessageBox.Show("Ошибка загрузки битовых
образов \n"+exc.ToString(), "Полет в облаках",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                this.Close();
                return;
            }
            // фоновый рисунок формы:
            BackgroundImage=new Bitmap(@"E:\DATA\C#\sky.bmp");
            /* задаем размер формы, соответствующий фоновому
            рисунку: */
            this.ClientSize = new System.Drawing.Size
            (BackgroundImage.Width, BackgroundImage.Height);

```

```

// делаем границу окна фиксированной
this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedSingle;
// блокируем кнопку "развернуть"
this.MaximizeBox = false;
/* g - графическая поверхность, на которой будем
формировать рисунок */
g = Graphics.FromImage(BackgroundImage);

// инициализируем генератор случайных чисел
r = new System.Random();

// начальное положение области с самолетом
rct.X = -40;
rct.Y = 20 + r.Next(20); //Координата по y случайна
rct.Width = plane.Width;
rct.Height = plane.Height;

/* скорость полета определяется периодом сигналов
таймера и величиной приращения dx */

dx = 2; // скорость полета - 2 пикселя/тик таймера
timer1.Interval = 20;
timer1.Enabled = true;
}

private void timer1_Tick(object sender, EventArgs e)
{
    // стираем изображение самолета путем заполнения
    // всей области фоном:
    g.DrawImage(sky, 0, 0);
    // изменяем положение самолета
    // Если не достигли границы, выполняем
    // приращение по координате X:
    if (rct.X < this.ClientRectangle.Width) rct.X += dx;
    else
    {
        /* если достигли границы, задаем положение
        самолета заново */
        rct.X = -40;
        rct.Y = 20 + r.Next(this.ClientSize.Height -
        40 - plane.Height);
        // Задаем скорость - от 2 до 6 пикселей/тик
        // таймера
        dx = 2 + r.Next(5);
    }
}

```

```

    }
    // рисуем самолет на рабочей поверхности
    g.DrawImage(plane, rct.X, rct.Y);

    // Задаем скрытие самолета в облаках:
    // Рисуем прямоугольную рамку
    Rectangle rct2 = new Rectangle(20, 20, sky.Width -
40, sky.Height - 40);
    g.DrawRectangle(Pens.Black, rct2.X, rct2.Y, rct2.Width
- 1, rct2.Height - 1);
    // обновляем область
    this.Invalidate(rct2);
}
}
}

```

Метод Refresh перерисовывает всю форму. Метод Invalidate без параметра - тоже. Метод **Invalidate** с параметром обеспечивает перерисовку только той области формы, которая указана в параметрах вызова метода. Если в качестве параметра методу Invalidate передать объект rct, то самолет долетит до правой границы окна, если rct2 - то исчезнет, как только выйдет за обозначенную рамку.

Самостоятельные задания

7.1. Измените приложение «Отрисовка линий и кривых» (задание 27) таким образом, чтобы и ломаная линия, и сглаженная кривая показывались одновременно в одном окне (разными цветами), соединяя одни и те же точки.

ЛАБОРАТОРНАЯ РАБОТА № 8

Тема: Работа с базами данных

Цели лабораторной работы: ознакомиться и использовать в приложениях различные приемы и методы для работы с базами данных

Учебные вопросы:

1. Использование компонентов `odbcConnection`, `odbcDataAdapter`, `DataSet` и `DataGrid` для работы с базой данных MS Access
2. Создание запросов через Query Builder
3. Создание приложения для работы с базой данных «Телефонная книга»
4. Создание приложения для работы с базой данных «Ежедневник»

Задание № 31. Программа « Телефонная книга »

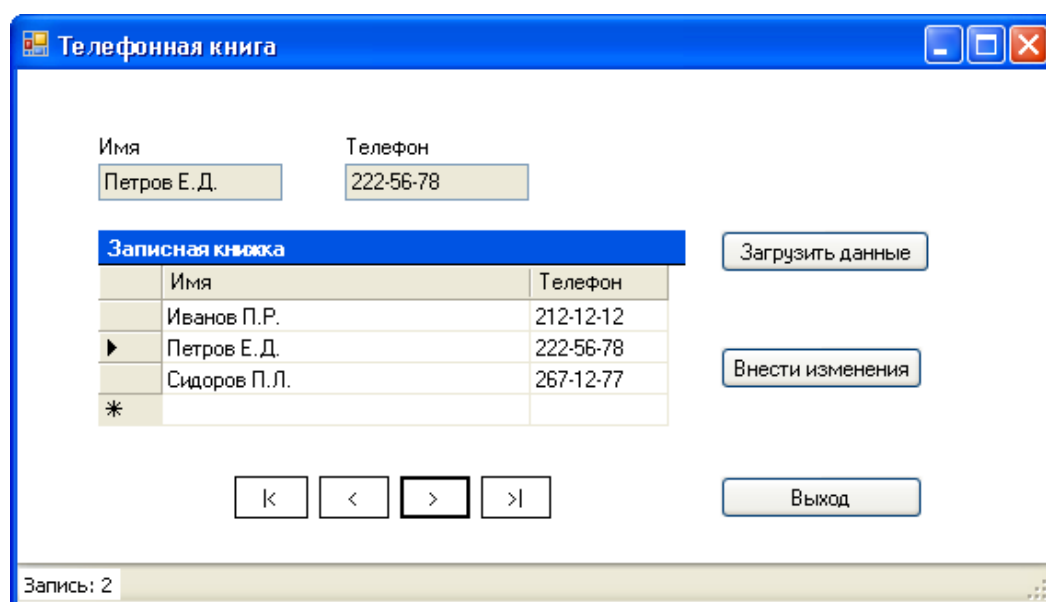


Рис. 58. Окно программы « Телефонная книга »

Программа «Телефонная книга» является простым примером программы для работы с базой данных Microsoft Access [2]. Демонстрирует использование компонентов `odbcConnection`, `odbcDataAdapter`, `DataSet` и `DataGrid`.

Соединение с базой данных MS Access осуществим через компонент `odbcConnection`. Взаимодействие с базой данных, после того, как соединение установлено, осуществим через `odbcDataAdapter`. Для хранения таблиц базы данных используем `DataSet`. Для вывода содержимого `DataSet` используем компонент `DataGrid`. (Если каких-то из этих компонентов в Toolbox не наблюдается, добавьте их через пункт Choose Items контекстного меню)

1. Создайте в MS Access базу Телефонная книга.mdb с одной таблицей Phones. В таблице имеются два поля: Title (Тип: текстовый; Размер: 60) и

Phone (Тип: текстовый; Размер: 20). Заполните таблицу тестовыми данными. Например:

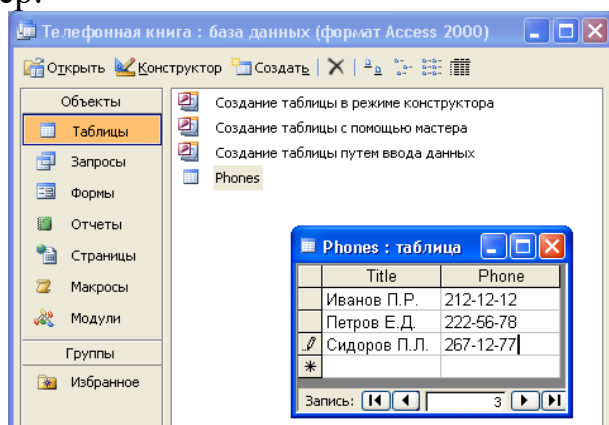


Рис. 59. БД Телефонная книга. mdb

2. Создайте новое приложение в Visual Studio.

Подключаемся к источнику данных:

Поместите на форму компонент `odbcConnection`. Зайдите в свойство `ConnectionString`, выберите пункт `New Connection`. В строке *Use user or system data source name* выберите `База данных MS Access`. В строке *Use connection string* выберите источник данных: на закладке «Файловый источник данных» выберите папку, в которой хранится ваша база Телефонная книга.mdb, в пункте `Имя DSN` выбираем драйвер, для которого задается источник (driver do Microsoft Access (*.mdb)). Далее. Выбираем файл Телефонная книга.mdb:

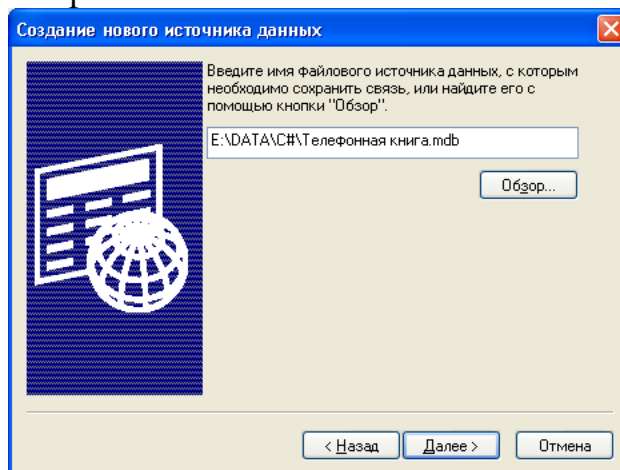


Рис. 60. Подключение к источнику данных

Далее. Готово. Откроется окно Установка драйвера ODBC для Microsoft Access, Выберите ваш файл с базой. ОК.

3. Перенесите на форму компонент `DataSet`. В появившемся окне выберите `Untyped Dataset`.

Зайдите в свойство `Tables`, откроется `Tables Collection Editor`. Добавьте одну таблицу (рис. 61) :

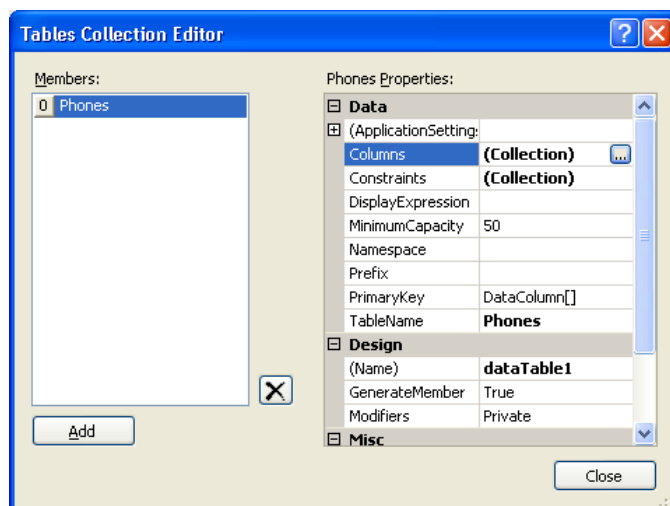


Рис. 61. Окно Tables Collection Editor

Откорректируем ее свойства: в свойстве TableName впишите Phones; зайдите в свойство Columns - откроется Columns Collection Editor. Создайте два столбца – Column1 и Column2.

Задайте им свойства:

| | | |
|---------|------------|-------|
| Column1 | Caption | Title |
| Column1 | ColumnName | Title |
| Column2 | Caption | Phone |
| Column2 | ColumnName | Phone |

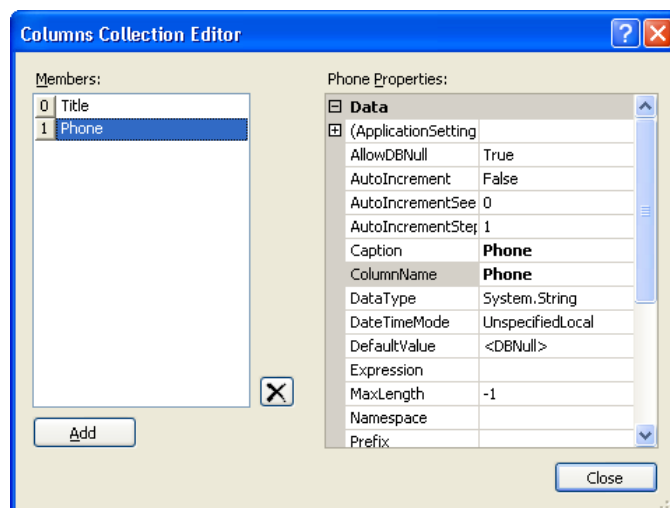


Рис. 62. Окно Columns Collection Editor

4. Перенесите на форму dataGrid. В его свойстве DataSource выберите DataSet1. В свойстве DataMember выбираем таблицу Phones. В свойстве CaptionText пишем «Записная книжка».

Зайдите в свойство TableStyles – откроется окно DataGridTableStyle Collection Editor. Добавьте один элемент (dataGridTableStyle1). В его свойстве MappingName выберите Phones. Выбирая свойство

GridColumnStyles, открываем DataGridViewColumnStyle Collection Editor. Добавляем два элемента: dataGridViewTextBoxColumn1 и dataGridViewTextBoxColumn2.

Задаем им свойства:

| | | |
|----------------------------|-------------|---------------|
| dataGridViewTextBoxColumn1 | HeaderText | Имя |
| dataGridViewTextBoxColumn1 | NullText | Пустая строка |
| dataGridViewTextBoxColumn1 | Width | 200 |
| dataGridViewTextBoxColumn1 | MappingName | Title |
| dataGridViewTextBoxColumn2 | HeaderText | Телефон |
| dataGridViewTextBoxColumn2 | NullText | Пустая строка |
| dataGridViewTextBoxColumn2 | Width | 100 |
| dataGridViewTextBoxColumn2 | MappingName | Phone |

5. Для заполнения полученными от источника данными объекта DataSet и выполнения некоторых операций используют OleDbDataAdapter.

Adapter позволяет не только заполнить DataSet данными, полученными от источника, но и помещать измененные данные обратно в источник при помощи стандартных команд SQL. Переносим его на форму.

В окне DataAdapterConfiguration Wizard оставляем предложенное соединение, Next, Next, заходим в построитель запросов (Query Builder). Добавляем таблицу Phones. Помечаем галочками столбцы Title и Phone. Выполняем запрос (Execute Query):

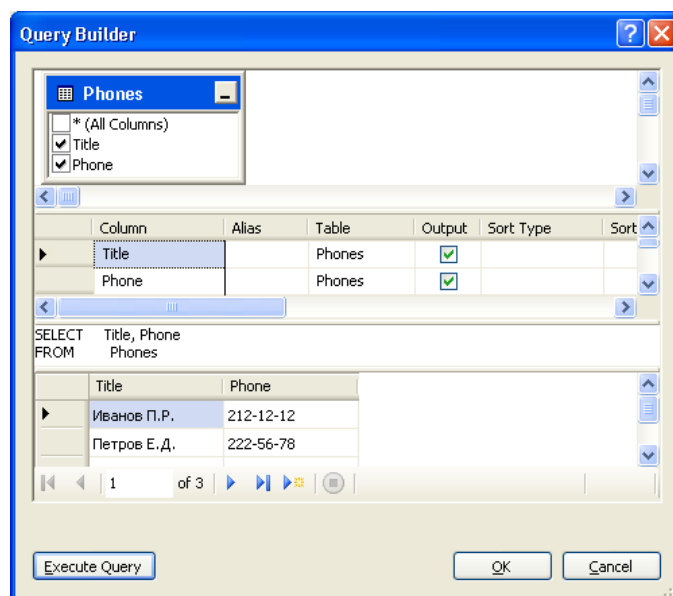


Рис. 63. Создание запроса через Query Builder.

Можете откорректировать SQL- команды и посмотреть на результаты. Например: сделаем выборку по фамилии:

| SELECT Title, Phone FROM Phones WHERE (Title = 'Иванов П.Р.') | | |
|--|-------------|-----------|
| | Title | Phone |
| ► | Иванов П.Р. | 212-12-12 |
| * | NULL | NULL |

Рис. 64. Выборка данных по фамилии

Посмотрим отсортированный по убыванию список фамилий:

| SELECT Title FROM Phones ORDER BY Title DESC | |
|--|--------------|
| | Title |
| ► | Сидоров П.Л. |
| | Петров Е.Д. |
| | Иванов П.Р. |
| * | NULL |

Рис. 65. Сортировка данных по полю «Фамилия»

Выберем только те записи, телефоны в которых начинаются с 212:

| SELECT Title, Phone FROM Phones WHERE (Phone LIKE '212%') | | |
|--|-------------|-----------|
| | Title | Phone |
| ► | Иванов П.Р. | 212-12-12 |
| * | NULL | NULL |

Рис. 66. Выборка через оператор LIKE

Оставьте в Query Builder команду:

SELECT Title, Phone

FROM Phones

(в DataSet будут загружаться все данные из источника)

Finish.

6. Зайдите в свойство TableMappings. Проверьте соответствие столбцов источника и DataSet:

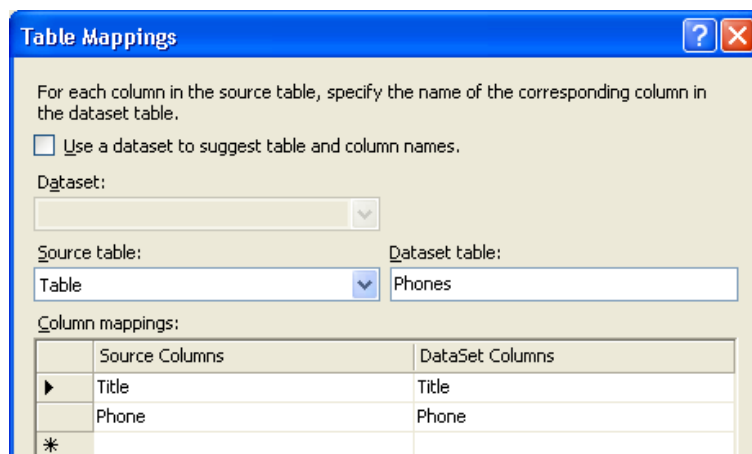


Рис. 67. Окно Table Mappings

7. Через разделы DeleteCommand, InsertCommand, SelectCommand, UpdateCommand определяем, какая именно SQL-команда будет передана на источник данных при вызове метода Update():

| | |
|--|--|
| SelectCommand.CommandText | SELECT Title, Phone FROM Phones |
| SelectCommand.Connection | odbcConnection1 |
| DeleteCommand.Connection | odbcConnection1 |
| DeleteCommand. CommandText | DELETE FROM Phones WHERE (Title = ?) AND (Phone = ?) |
| DeleteCommand.Parameters.Title.odbcType | Text |
| DeleteCommand.Parameters.Title.Precision | 60 |
| DeleteCommand.Parameters.Title.SourceColumn | Title |
| DeleteCommand.Parameters.Title.SourceVersion | Original |
| DeleteCommand.Parameters.Phone.odbcType | Text |
| DeleteCommand.Parameters.Phone.Precision | 20 |
| DeleteCommand.Parameters.Phone.SourceColumn | Phone |
| DeleteCommand.Parameters.Phone.SourceVersion | Original |
| InsertCommand.Connection | odbcConnection1 |
| InsertCommand. CommandText | INSERT INTO Phones (Title, Phone) VALUES (?, ?) |
| InsertCommand.Parameters.Title.odbcType | Text |
| InsertCommand.Parameters.Title.Precision | 60 |
| InsertCommand.Parameters.Title.SourceColumn | Title |
| InsertCommand.Parameters.Title.SourceVersion | Current |
| InsertCommand.Parameters.Phone.odbcType | Text |
| InsertCommand.Parameters.Phone.Precision | 20 |

| | |
|--|--|
| InsertCommand.Parameters.Phone.SourceColumn | Phone |
| InsertCommand.Parameters.Phone.SourceVersion | Current |
| UpdateCommand.Connection | odbcConnection1 |
| UpdateCommand.CommandText | UPDATE Phones SET Title = ?, Phone = ? WHERE (Title = ?) AND (Phone = ?) |

| | |
|--|----------|
| UpdateCommand.Parameters.Title.odbcType | Text |
| UpdateCommand.Parameters.Title.Precision | 60 |
| UpdateCommand.Parameters.Title.SourceColumn | Title |
| UpdateCommand.Parameters.Title.SourceVersion | Current |
| UpdateCommand.Parameters.Phone.odbcType | Text |
| UpdateCommand.Parameters.Phone.Precision | 20 |
| UpdateCommand.Parameters.Phone.SourceColumn | Phone |
| UpdateCommand.Parameters.Phone.SourceVersion | Current |
| UpdateCommand.Parameters.OriginalTitle.odbcType | Text |
| UpdateCommand.Parameters.OriginalTitle.Precision | 60 |
| UpdateCommand.Parameters.OriginalTitle.SourceColumn | Title |
| UpdateCommand.Parameters.OriginalTitle.SourceVersion | Original |
| UpdateCommand.Parameters.OriginalPhone.odbcType | Text |
| UpdateCommand.Parameters.OriginalPhone.Precision | 20 |
| UpdateCommand.Parameters.OriginalPhone.SourceColumn | Phone |
| UpdateCommand.Parameters.OriginalPhone.SourceVersion | Original |

```

8. // Вписываем код для кнопки «Загрузить данные»:
private void button1_Click(object sender, EventArgs e)
{
    try
    { // заполняем таблицу в DataSet данными:
        odbcDataAdapter1.Fill(dataSet1.Tables["Phones"]);
    }
    catch (Exception exc)
    {
        MessageBox.Show(exc.Message);
    }
}

// Вписываем код для кнопки «Внести изменения»:
private void button2_Click(object sender, EventArgs e)

```

```

{
    try
    {
        odbcDataAdapter1.Update(dataSet1);
        button2.Text = "Выполнено!";
    }
    catch
    {
        button2.Text = "Ошибка!";
    }
}

```

Запустите, потестируйте. Попробуйте добавить, удалить данные.

9. У поля textBox1 свойству DataBindings.Text задайте значение Title, свойство ReadOnly переведите в true. У поля textBox2 свойству DataBindings.Text задайте значение Phone, свойство ReadOnly переведите в true. В этих полях будут отображаться текущие данные таблицы.

Поместите на форму statusStrip, через свойство Items добавьте элемент toolStripStatusLabel1 (будет отображать номер текущей записи).

Настроим кнопки, обеспечивающие навигацию по записям таблицы (| <, <, >, > |). Допишите в код программы некоторые объявления и функции:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace Телефонная_книга
{
    public partial class Form1 : Form
    {
        /* создаем объект myManagerBase, который
        используется для связи компонента DataSet с таблицей
        базы данных Phones (для индексирования выбранной
        записи) */
        private System.Windows.Forms.BindingManagerBase
            myManagerBase;

        // Если изменился индекс выбранной записи БД:
        private void PositionChanged(object sender,
            System.EventArgs e)
        {
            if (myManagerBase.Position != -1)

```

```

        { /* записи в БД нумеруются с нуля, поэтому к
позиции прибавляем 1, отображаем номер в statusStrip
*/
            toolStripStatusLabel1.Text= "Запись: "+
(myManagerBase.Position + 1).ToString();
            dataGrid1.CurrentRowIndex=myManagerBase.Position;
        }
    }
    public Form1()
    {
        InitializeComponent();
        dataGrid1.BackgroundColor = this.BackColor;
        dataGrid1.BorderStyle = BorderStyle.None;

        /* Создаем объект типа BindingContext и определяем
процедуру обработки события PositionChanged */
        myManagerBase =
            this.BindingContext[dataSet1.Tables["phones"]];
        myManagerBase.PositionChanged += new
System.EventHandler(this.PositionChanged);
        if (myManagerBase.Count == 0)
            toolStripStatusLabel1.Text = "База данных не содержит
записей.";
        else toolStripStatusLabel1.Text ="Запись: " +
(myManagerBase.Position+1).ToString();
    }
    //Для кнопки «К первой записи» (|< )
    private void button3_Click(object sender,EventArgs e)
    {
        if (myManagerBase.Count > 0)
            myManagerBase.Position = 0;
    }
    //Для кнопки «К предыдущей записи» ( < )
    private void button4_Click(object sender,EventArgs e)
    {
        if (myManagerBase.Position > 0)
            myManagerBase.Position -= 1;
    }
    //Для кнопки «К следующей записи» ( > )
    private void button5_Click(object sender,EventArgs e)
    {
        if ((myManagerBase.Position<myManagerBase.Count-1)
&&

```

```

(myManagerBase.Count >0)) myManagerBase.Position+= 1;
    }

//Для кнопки «К последней записи» ( >| )
private void button6_Click(object sender,EventArgs e)
{
    if (myManagerBase.Count > 0)
        myManagerBase.Position=myManagerBase.Count - 1;
    }
    // Выход
private void button7_Click(object sender,EventArgs e)
{ this.Close();
}

// Если изменился индекс выбранной ячейки в DataGridView:
private void dataGridView1_CurrentCellChanged
                                (object sender,EventArgs e)
    {
if (myManagerBase.Position!
=dataGrid1.CurrentRowIndex)
    myManagerBase.Position=dataGrid1.CurrentRowIndex;
    }

```

Плюс описанные выше процедуры для button1 и button2 (кнопки «Загрузить данные» и «Внести изменения»).
Запустите, проверьте работоспособность.

Задание № 32. Программа « Ежедневник »

Демонстрирует работу с базой данных MS Access (journal.mdb) [2]. База содержит таблицу journal, в которой содержатся записи запланированных дел. Программа позволяет просмотреть список дел на сегодня, завтра, эту неделю и следующую неделю. Кроме того, по указанному мероприятию можно найти дату его проведения. Предусмотрена возможность сортировки данных:

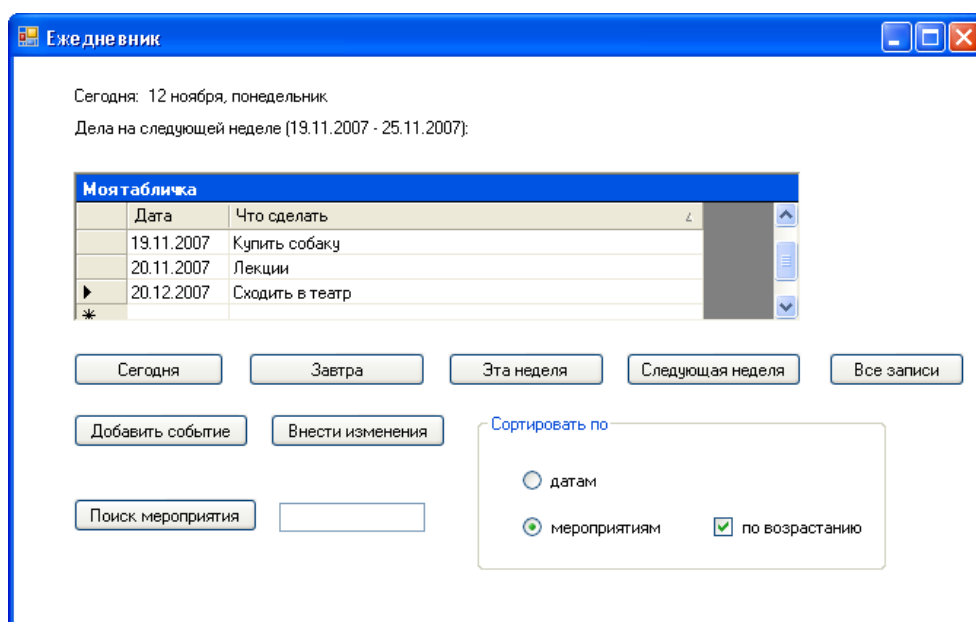


Рис. 68а. Окно программы « Ежедневник »

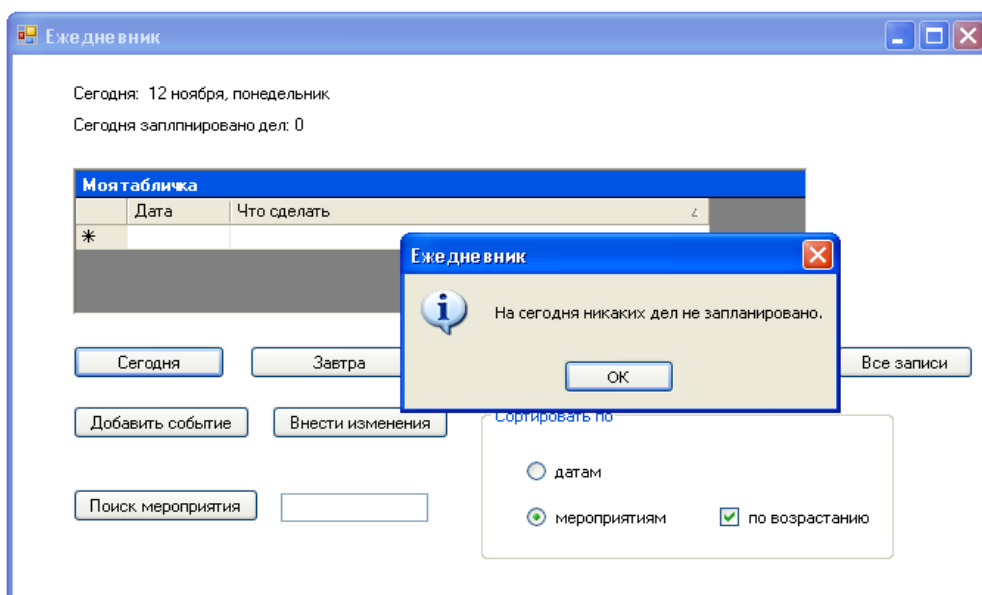


Рис. 68б. Демонстрация некоторых реакций программы

Добавление нового события осуществляется при помощи вспомогательной формы «Новое событие» (см. рис. 69):

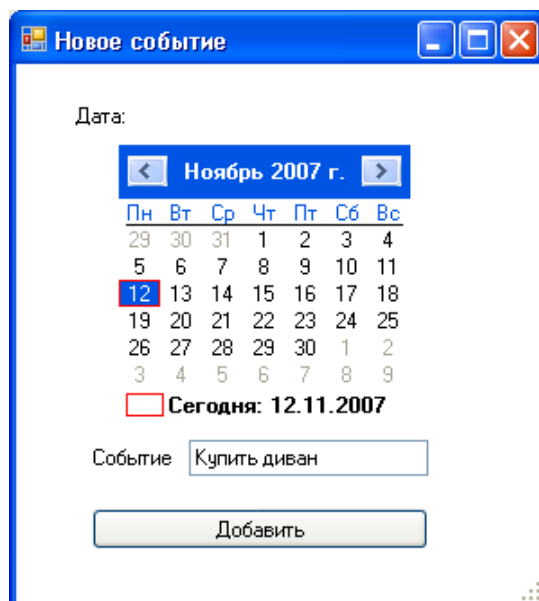


Рис. 69. Окно для добавления данных

Выборка данных при просмотре ежедневника осуществляется из таблицы компонента DataSet1, а не непосредственно из базы данных.

1. Создайте в Access базу journal.mbd, заполните ее произвольным набором данных. Например:

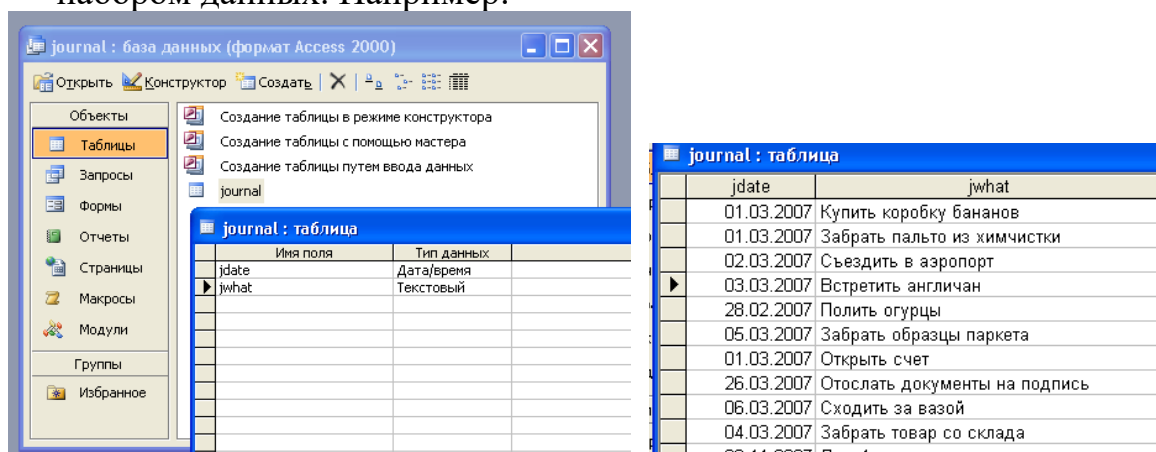


Рис. 70. БД journal.mbd

2. Подключите odbcConnection1 к базе journal.mdb.
3. Настраиваем компонент dataSet1:

Зайдите в свойство Table, откроется Tables Collection Editor. Добавьте две таблицы. Откорректируем их свойства: в свойстве TableName у Table1 впишите journal; у Table2- journalSelect;

Зайдите в свойство Columns таблицы journal- откроется Columns Collection Editor. Создайте два столбца – Column1 и Column2.

Задайте им свойства:

| | | |
|---------|------------|-------|
| Column1 | Caption | jdate |
| Column1 | ColumnName | jdate |
| Column2 | Caption | jwhat |

| | | |
|---------|------------|-------|
| Column2 | ColumnName | jwhat |
|---------|------------|-------|

Зайдите в свойство Columns таблицы journalSelect- откроется Columns Collection Editor. Создайте два столбца – Column1 и Column2. Задайте им точно такие же свойства.

4. Настраиваем dataGrid:

В его свойстве DataSource выберите DataSet1. В свойстве DataMember выбираем таблицу journalSelect. В свойстве CaptionText пишем «Ежедневник». ReadOnly ->true.

Зайдите в свойство TableStyles – откроется окно DataGridTableStyle Collection Editor. Добавьте один элемент (dataGridTableStyle1). В его свойстве MappingName выберите JournalSelect. Выбирая свойство GridColumnStyles, открываем DataGridColumnStyle Collection Editor. Добавляем два элемента: dataGridTextBoxColumn1 и dataGridTextBoxColumn2.

Задаем их свойства:

| | | |
|------------------------|-------------|---------------|
| dataGridTextBoxColumn1 | HeaderText | Дата |
| dataGridTextBoxColumn1 | NullText | Пустая строка |
| dataGridTextBoxColumn1 | Width | 70 |
| dataGridTextBoxColumn1 | Format | dd/MM/yyyy |
| dataGridTextBoxColumn1 | MappingName | jdate |
| dataGridTextBoxColumn2 | HeaderText | Что сделать |
| dataGridTextBoxColumn2 | NullText | Пустая строка |
| dataGridTextBoxColumn2 | Width | 325 |
| dataGridTextBoxColumn2 | MappingName | jwhat |

5. Настраиваем odbcDataAdapter1.

Оставьте в Query Builder команду:

```
SELECT  jdate, jwhat
FROM    journal
```

(в DataSet будут загружаться все данные из источника)

Зайдите в свойство TableMappings. Проверьте соответствие столбцов источника и DataSet (согласно рис. 71):

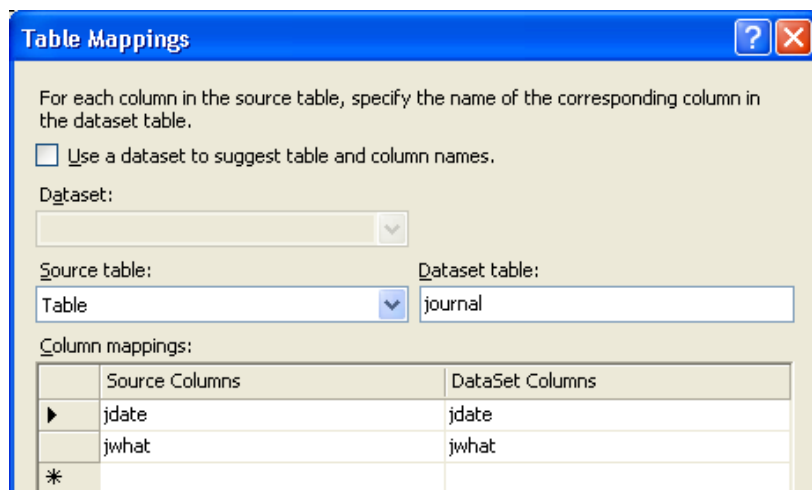


Рис. 71. Окно TableMappings

Через разделы InsertCommand, SelectCommand, UpdateCommand определяем, какая именно SQL-команда будет передана на источник данных при вызове метода Update().

| | |
|--|---|
| SelectComand.CommandText | SELECT jdate, jwhat FROM journal |
| InsertComand.CommandText | INSERT INTO journal (jdate, jwhat) VALUES (?, ?) |
| InsertCommand.Parameters.jdate.odbcType | Text |
| InsertCommand.Parameters.jdate.Precision | 20 |
| InsertCommand.Parameters.jdate.SourceColumn | jdate |
| InsertCommand.Parameters.jdate.SourceVersion | Current |
| InsertCommand.Parameters.jwhat.odbcType | Text |
| InsertCommand.Parameters.jwhat.Precision | 100 |
| InsertCommand.Parameters.jwhat.SourceColumn | jwhat |
| InsertCommand.Parameters.jwhat.SourceVersion | Current |
| UpdateComand.CommandText | UPDATE journal SET jdate = ?, jwhat = ? WHERE (jdate = ?) AND (jwhat = ?) |

Вписываем код:

Листинг 32

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
```

```

using System.Text;
using System.Windows.Forms;
namespace Ежедневник
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            label1.Text = "Сегодня: " +
                DateTime.Today.ToString("d MMMM, dddd");
            label2.Text = "";
            try
            {
                //загружаем данные из journal.mdb в dataSet1
                odbcDataAdapter1.Fill(dataSet1.Tables["journal"]);
            }
            catch (Exception exc)
            {
                MessageBox.Show( "Ошибка доступа к базе данных.\n" + exc.Message, "Ежедневник", MessageBoxButtons.OK,
                MessageBoxIcon.Error);
            }
        }

        // Кнопка «Сегодня»
        private void button1_Click(object sender, EventArgs e)
        {
            // формируем выборку дел на сегодня (даты
            // выбираются через формат #month/day/year#)
            System.Data.DataRow[] dr =
                dataSet1.Tables["journal"].Select("jdate = #" +
                    DateTime.Today.Month.ToString() + "/" +
                    DateTime.Today.Day.ToString() + "/" +
                    DateTime.Today.Year.ToString() + "#");

            // добавляем полученную выборку в таблицу
            dataSet1.Tables["journalSelect"].Clear();
            if (dr.Length > 0)
                for (int i = 0; i < dr.Length; i++)
                    dataSet1.Tables["journalSelect"].Rows.Add(dr[i].
                        ItemArray);
            else

```

```

        MessageBox.Show("На сегодня никаких дел не
запланировано.", "Ежедневник", MessageBoxButtons.OK,
MessageBoxIcon.Information);
        label2.Text = "Сегодня запланировано дел: " +
Convert.ToString(dr.Length);
    }

```

// Кнопка «Завтра»:

```

private void button2_Click(object sender, EventArgs e)
{
    DateTime tm = DateTime.Today.AddDays(1);
    // формируем выборку дел на завтра
    System.Data.DataRow[] dr =
        dataSet1.Tables["journal"].Select("jdate = #" +
            tm.Month.ToString() + "/" +
            tm.Day.ToString() + "/" +
            tm.Year.ToString() + "#");

    // добавляем полученную выборку в таблицу
    dataSet1.Tables["journalSelect"].Clear();
    if (dr.Length > 0)
        for (int i = 0; i < dr.Length; i++)
            dataSet1.Tables["journalSelect"].Rows.Add(dr[i].
                ItemArray);
    else
        MessageBox.Show("На завтра никаких дел не
запланировано.", "Ежедневник", MessageBoxButtons.OK,
MessageBoxIcon.Information);
        label2.Text = "На завтра запланировано дел: " +
Convert.ToString(dr.Length);
}

```

// Кнопка «Эта неделя»:

```

private void button3_Click(object sender, EventArgs e)
{
    // определяем дату начала и конца этой недели
    DateTime wf = DateTime.Today;
    while (wf.DayOfWeek != System.DayOfWeek.Monday)
        wf = wf.AddDays(-1);
    DateTime wl = wf.AddDays(6);
    label2.Text = "Дела на этой неделе " +
        wf.ToString("dd/MM/yyyy -") +
        wl.ToString("dd/MM/yyyy):";
}

```

```

        // формируем выборку
        System.Data.DataRow[] dr =
        dataSet1.Tables["journal"].Select("jdate >= #" +
        wf.Month.ToString() + "/" + wf.Day.ToString()+ "/" +
        wf.Year.ToString() + "# and " + "jdate <= #" +
        wl.Month.ToString() + "/" + wl.Day.ToString()+ "/" +
        wl.Year.ToString() + "#");

        // добавляем полученную выборку в таблицу
        dataSet1.Tables["journalSelect"].Clear();
        if (dr.Length > 0)
            for (int i = 0; i < dr.Length; i++)
                dataSet1.Tables["journalSelect"].Rows.Add(dr[i].
                ItemArray);
        else
            MessageBox.Show("На этой неделе никаких дел
            не запланировано.", "Ежедневник", MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    }
    // Кнопка «Следующая неделя»:
    private void button4_Click(object sender, EventArgs e)
    {
        // определяем дату начала и конца этой недели
        DateTime wf = DateTime.Today;
        // если сегодня понедельник
        if (wf.DayOfWeek == System.DayOfWeek.Monday)
            wf = wf.AddDays(7);
        else
            while (wf.DayOfWeek != System.DayOfWeek.Monday)
                wf = wf.AddDays(1);
        DateTime wl = wf.AddDays(6);
        label2.Text = "Дела на следующей неделе " +
        wf.ToString("dd/ММ/уууу-")+wl.ToString("dd/ММ/уууу:");
        // формируем выборку
        System.Data.DataRow[] dr =
        dataSet1.Tables["journal"].Select("jdate >= #" +
        wf.Month.ToString()+ "/" + wf.Day.ToString()+ "/" +
        wf.Year.ToString() + "# and " + "jdate <= #" +
        wl.Month.ToString()+ "/" +wl.Day.ToString() + "/" +
        wl.Year.ToString() + "#");

        // добавляем полученную выборку в таблицу
        dataSet1.Tables["journalSelect"].Clear();
        if (dr.Length > 0)
    }

```

```

        for (int i = 0; i < dr.Length; i++)
dataSet1.Tables["journalSelect"].Rows.Add(dr[i].ItemA
rray);
    else
        MessageBox.Show("На следующей неделе никаких
дел не запланировано.", "Ежедневник",
MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
// Кнопка «Все записи»
private void button5_Click(object sender, EventArgs e)
{
    label2.Text = "Все записи ежедневника:";
    dataSet1.Tables["journalSelect"].Clear();
    for (int i=0; i<dataSet1.Tables["journal"].Rows.
Count; i++)
        dataSet1.Tables["journalSelect"].Rows.
Add(dataSet1.Tables["journal"].Rows[i].ItemArray);
}
// Кнопка «Добавить событие»:
private void button6_Click(object sender, EventArgs e)
{
    Form2 af = new Form2();
    af.ShowDialog();
    // введено новое событие
    if (af.IsDtEventSet())
    { // добавляем событие в dataSet1
        dataSet1.Tables["journal"].Rows.
            Add(new string[2] { af.dtime,
af.dwhat });
    }
}
// Кнопка «Поиск мероприятия»
private void button7_Click(object sender, EventArgs e)
{
    // формируем выборку дат для заданного дела:
    System.Data.DataRow[] dr =
dataSet1.Tables["journal"].Select("jwhat='"+textBo
x1.Text+"'"); /* (обратите внимание, текст в запросе
должен быть заключен в апострофы) */

    // добавляем полученную выборку в таблицу
    dataSet1.Tables["journalSelect"].Clear();
    if (dr.Length > 0)
        for (int i = 0; i < dr.Length; i++)

```

```

dataSet1.Tables["journalSelect"].Rows.Add(dr[i].ItemA
rray);
    else
        MessageBox.Show("Таких мероприятий нет");
    label2.Text = "Дата запрошенного мероприятия";
}

// сортировка по датам выводимой информации
private void radioButton1_Click (object sender,
EventArgs e)
{
    if (checkBox1.Checked) // по возрастанию:
        dataSet1.Tables["journalSelect"].DefaultView.Sort =
        "jdate ASC";
    else // по убыванию:
        dataSet1.Tables["journalSelect"].DefaultView.Sort =
        "jdate DESC";
    dataGrid1.DataSource =
        dataSet1.Tables["journalSelect"].DefaultView;
}
/* сортировка по мероприятиям выводимой в dataGrid1
информации */
private void radioButton2_Click(object sender,
EventArgs e)
{
    if (checkBox1.Checked) // по возрастанию:
        dataSet1.Tables["journalSelect"].DefaultView.Sort =
        "jwhat ASC";
    else // по убыванию:
        dataSet1.Tables["journalSelect"].DefaultView.Sort =
        "jwhat DESC";
    dataGrid1.DataSource =
        dataSet1.Tables["journalSelect"].DefaultView;
}

// Кнопка «Внести изменения»
private void button8_Click(object sender,EventArgs e)
{
    try
    {
        odbcDataAdapter1.Update(dataSet1);
    }
    catch (Exception exc)
    {
        MessageBox.Show(exc.Message);
    }
}

```

```

    }
}
}
}
// Для формы 2:
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace Ежедневник
{
    public partial class Form2 : Form
    {
        public string dtime = string.Empty;
        public string dwhat= string.Empty;

        public Boolean IsDtEventSet()
        { // возвращает True, если поля заполнены
            if (dtime != string.Empty && dwhat !=
string.Empty)
                return true;
            else return false;
        }
        public Form2()
        {
            InitializeComponent();
            monthCalendar1.MaxSelectionCount = 1;
        }
        // Кнопка «Добавить»
        private void button1_Click(object sender,EventArgs e)
        {
            this.dtime =
monthCalendar1.SelectionStart.ToShortDateString();
            this.dwhat = textBox1.Text;
            this.Close();
        }
    }
}

```

Проверяем работоспособность.

Самостоятельные задания

По курсу «Объектно-ориентированное программирование» и «Высокоуровневые методы информатики и программирования» студенты пишут курсовые работы. Информацию по темам и требованиям к курсовой узнайте у преподавателя.

Приложение

Основные операторы языка

Составной оператор – это произвольное количество операторов, заключенных в фигурные скобки { };

Операторы перехода: break, goto, continue, return. Используют для прерывания последовательного выполнения программы.

Оператор присваивания: <переменная> = <выражение>.

Обязательное условие- совместимость типов переменной и выражения. Допустимы сокращенные формы +=, -=, *= и пр.

Условный оператор: if (<Выражение>) <Оператор1>; else <Оператор2>

Оператор выбора:

```
switch (<Выражение>)
{
    case <Константное выражение 1>: <Операторы 1>; <Оператор перехода 1>;
    case <Константное выражение 2>: <Операторы 2>; <Оператор перехода 2>;
    ...
    case <Константное выражение N>: <Операторы N>; <Оператор перехода N>;
    default: <Операторы N+1>; <Оператор перехода N+1>;
}
```

Оператор цикла с параметрами:

```
for (<Инициализация счетчиков>; <Условие>; <Изменение счетчиков>)
<Оператор>
```

Оператор цикла while

Используют, если число итераций заранее не известно.

Префиксная форма:

```
while (<Выражение>)
    <Оператор>
```

Постфиксная форма:

```
do
    <Оператор>
while (<Выражение>)
```

Цикл foreach

Предназначен для обработки массивов, коллекций и других контейнеров, рассчитанных на хранение множества данных.

foreach (<Элемент> in <Контейнер>) <Оператор>

Обработка исключений:

```
try
{ // Защищенный блок кода }
catch (T1 e1)
{ // Обработчик исключения типа T1 }
...
catch (Tn en)
{ // Обработчик исключения типа Tn }
finally
{ // Блок финализации. Срабатывает в любом случае }
```

Форматы отображения чисел, используемые в функции ToString()

| Формат | Формат | Пример |
|--------|---|---------------|
| c, C | Currency – финансовый. Используют для представления денежных величин. Обозначение денежной единицы, разделитель групп разрядов определяют соответствующие настройки операционной системы. | 55 055,28 p. |
| e, E | Scientific (exponential)- научный. Используется для представления очень маленьких или очень больших чисел. | 5,505528+E004 |
| f, F | Fixed- число с фиксированной точкой. Используют для представления дробных чисел. Количество цифр дробной части, способ отображения отрицательных чисел определяют соответствующие настройки операционной системы. | 55055,28 |
| g, G | General – универсальный формат. Похож на Number, но разряды не разделены на группы. | 55055,275 |
| n, N | Number – числовой. Используют для представления дробных чисел. Обозначение денежной единицы, разделитель групп разрядов определяют соответствующие настройки операционной системы. | 55 055,28 |
| r, R | Roundtrip- без округления. В отличие от формата N, не выполняет округления. Количество цифр дробной части зависит от значения числа | 55055,2775 |

Целое число, указанное сразу после спецификатора формата, означает количество цифр в дробной части. Например: `double t = 1.545666;`
`t.ToString("N3") -> "1.546"`

Некоторые свойства разных компонентов

| Свойство | Описание |
|-----------------|---|
| Appearance | Внешний вид компонента |
| BackColor | Цвет фона |
| BackGroundImage | Фоновое изображение |
| BorderStyle | Вид границы |
| Dock | Привязка компонента к границе контейнера |
| Enabled | Признак доступности компонента |
| Font | Параметры шрифта |
| ForeColor | Цвет текста, отображаемого в компоненте. Для формы-цвет, используемый по умолчанию компонентами, находящимися на поверхности формы. |
| Height | Высота объекта |
| Icon | Значок в заголовке формы |
| Image | Изображение на поверхности компонента |
| Interval | Период генерации события Tick у таймера |
| Items | Создание коллекций элементов |
| Location | Положение компонента на поверхности формы |
| Maximum | Максимально допустимое значение свойства Value |
| Minimum | Минимально допустимое значение свойства Value |
| Multiline | Перевод поля редактирования в многострочный режим |
| Name | Имя компонента, по которому к нему обращаются в программе |
| Opacity | Степень прозрачности компонента |
| ReadOnly | Разрешение (false) или запрет (true) на редактирование |
| ScrollBars | Создание полос прокрутки |
| SelectedIndex | Номер выбранного в списке элемента |
| Size | Размер компонента |
| Sorted | Признак необходимости сортировки элементов списка |
| Step | Шаг приращения значений компонента |
| Text | Текст в компоненте |
| TextAlign | Способ выравнивания текста в компоненте |
| TickFrequency | Частота нанесения отметок на регуляторе |

| | |
|---------|--|
| ToolTip | Подсказка, появляющаяся рядом с указателем мыши при позиционировании его на компоненте (требуется, чтобы на форме присутствовал компонент Tooltip) |
| Value | Значение в компоненте |
| Visible | Степень видимости компонента |
| Width | Ширина объекта |
| Checked | «Включенное» состояние переключателя, признак того, что выбран некоторый элемент |

Функции манипулирования строками

| Функция | Значение |
|--------------------|--|
| s.Length | Длина (количество символов) строки |
| s.IndexOf(st) | Позиция первого вхождения подстроки st в строку s. Если указанной подстроки в строке нет, то значение функции равно минус один. |
| s.IndexOf(st,k) | Позиция вхождения подстроки st в строку s. Поиск ведется начиная с позиции k. |
| s.LastIndexOf(st) | Положение (от конца) подстроки st в строке s. |
| s.Trim() | Строка, полученная из строки s путем отбрасывания пробелов, находящихся в начале и в конце строки. |
| s.TrimStart() | Строка, полученная из строки s путем отбрасывания пробелов, находящихся в начале. |
| s.TrimEnd() | Строка, полученная из строки s путем отбрасывания пробелов, находящихся в конце. |
| s.Substring(n) | Подстрока, выделенная из строки s, начиная с символа под номером n и до конца строки. |
| s.Substring(n, k) | Подстрока длиной k символов, выделенная из строки s, начиная с символа под номером n. |
| s.Insert(pos,st) | Строка, полученная путем вставки в строку s строки st. Параметр pos задает номер символа строки s, после которого вставляется строка st. |
| s.Remove(pos,n) | Строка, полученная путем удаления из строки s цепочки символов. Параметр pos задает положение подстроки, n – количество удаляемых символов |
| s.Replace(old,new) | Строка, полученная из строки s путем замены всех символов old на символ new. |
| s.ToUpper() | Строка, полученная из строки s путем замены строчных символов на прописные. |
| s.ToLower() | Строка, полученная из строки s путем замены прописных символов на строчные. |
| s.Split(sep) | Массив строк, полученный разбиением строки s на подстроки. Параметр sep (массив типа char) задает |

| | |
|--------------------------------------|--|
| | символы, которые используются методом Split для обнаружения границ подстрок. |
| Compare(s1,s2,a) | Сравнивает строки s1 и s2. Возвращает -1, если s1<s2, ноль, если s1=s2, единицу, если s1>s2. Если параметр a - true, сравнение без учета регистра. |
| s.StartsWith(s2) | Возвращает true, если текущая строка начинается подстрокой s2 |
| s.EndsWith(s2) | Возвращает true, если текущая строка заканчивается подстрокой s2 |
| s.PaddLeft(n, c) s.PaddRight(n,c) | Дополняют текущую строку слева (справа) символами c до тех пор, пока длина строки не станет равной n. |

Функции манипулирования файлами и каталогами

Принадлежат пространству имен System.IO

| Функция | Значение |
|---------------------|--|
| DirectoryInfo(Path) | Создает объект типа DirectoryInfo, соответствующий каталогу, заданному параметром Path. |
| di.GetFiles(f) | Формирует список файлов каталога, как массив объектов типа FileInfo. Каждый элемент массива соответствует файлу каталога, заданного объектом di (тип DirectoryInfo). Параметр f задает критерий отбора файлов. |
| di.Exists | Проверяет, существует ли в системе каталог. Если существует, то значение функции равно true. |
| di.Create(d) | Создает каталог. |
| fi.AppendText() | Создает объект StreamWriter для добавления текста к файлу. (тип fi- FileInfo), |
| fi.CreateText() | Создает объект StreamWriter для записи текстовых данных в новый файл. |
| fi.OpenText() | Создает объект StreamReader, который позволяет считывать информацию из существующего текстового файла. |
| fi.CopyTo(Path) | Копирует файл, заданный объектом fi в каталог и под именем, заданным параметром Path. Значение метода- объект типа FileInfo, соответствующий файлу, созданному в результате копирования. |
| fi.MoveTo(Path) | Перемещает файл, заданный объектом fi, в каталог и под именем, заданным параметром Path. |
| fi.Delete() | Удаляет файл, соответствующий объекту fi. |

| | |
|------------------------------------|---|
| StreamReader(f) | Создает и открывает для чтения поток, соответствующий файлу, заданному параметром f. Значение метода- объект типа StreamReader. Поток открывается для чтения в кодировке UTF-8 |
| StreamReader(f,encd) | Создает и открывает для чтения поток, соответствующий файлу, заданному параметром f. Значение метода- объект типа StreamReader. Поток открывается для чтения в кодировке encd. |
| sr.Read() | Читает символ из потока sr (объект типа StreamReader). |
| sr.Read(buf,p,n) | Читает из потока sr символы и записывает их в массив символов buf. Параметр p задает номер элемента массива, в который будет помещен первый прочитанный символ, параметр n- количество символов, которое надо прочитать. |
| sr.ReadToEnd() | Читает весь текст из потока sr. |
| sr.ReadLine() | Читает строку из потока sr. |
| StreamWriter(f) | Создает и открывает для записи поток, соответствующий файлу, заданному параметром f. Значение метода- объект типа StreamWriter. Поток открывается для записи в кодировке UTF-8 |
| StreamWriter(f,a,encd) | Создает и открывает для записи поток, соответствующий файлу, заданному параметром f. Значение метода- объект типа StreamWriter. Поток открывается для записи в кодировке encd. Параметр a задает режим записи: true- добавление в файл, false- перезапись |
| sw.Write(v) | Записывает в поток sw строку символов, соответствующую значению v. В качестве v можно использовать выражение строкового или числового типа. |
| sw.WriteLine(v) | Записывает в поток sw строку символов, соответствующую значению v, и символ «новая строка». В качестве v можно использовать выражение строкового или числового типа. |
| FileStream(f,FileMode, FileAccess) | Создает объект FileStream, соответствующий файлу f. Параметр FileMode определяет способ создания потока, FileAccess регулирует доступ потока к данным. |
| fs.Seek(k,Origin) | Положение текущей записи. fs-объект типа FileStream, k- смещение относительно позиции, указанной параметром Origin (SeekOrigin.Begin- начало потока, SeekOrigin.Current - текущая |

| | |
|-----------|---------------------------------------|
| | запись, SeekOrigin.End- конец потока) |
| s.Close() | Закрывает поток s. |

Значения параметра FileMode

| | |
|--------------|--|
| Append | Добавляет записи в существующий файл или создает новый. Требуется, чтобы параметр FileAccess имел значение Write. |
| Create | Создает новый файл или переписывает существующий. Требуется, чтобы параметр FileAccess имел значение Write. |
| CreateNew | Создает новый файл, если же он уже существует, возникает исключение. Требуется, чтобы параметр FileAccess имел значение Write. |
| Open | Открывает существующий файл. Если файла нет, возникает исключение. |
| OpenOrCreate | Открывает существующий файл или создает новый, если он еще не создан. |
| Truncate | Открывает существующий файл и делает его размер равным нулю. |

Значения параметра FileAccess

| | |
|-----------|--|
| Read | Поток может читать данные |
| ReadWrite | Поток может читать и записывать данные |
| Write | Поток может записывать данные |

Математические функции

Принадлежат пространству имен Math

| | |
|-------------------|---|
| Abs(n) | Абсолютное значение n |
| Sign(n) | Возвращает 1- если n положительное, -1 – если n отрицательное, 0- если n=0 |
| Round(n,m) | Дробное число, полученное путем округления n по правилам математики. Параметр m задает количество цифр дробной части |
| Round(n) | Округляет n до ближайшего целого. Результат имеет тип double. |
| Ceiling(n) | Дробное, полученное путем округления n «с избытком» (Наименьшее целое число, которое больше или равно аргументу) |
| Floor(n) | Дробное, полученное путем округления n «с недостатком» (отбрасыванием дробной части) |
| DivRem(a,b,out r) | Возвращает результат деления целых чисел a и b, в переменную r записывается остаток от деления. Например: <code>Math.DivRem(5, 2, out r) = 2, r=1.</code> |
| Log(n) | Натуральный логарифм n |

| | |
|----------------|--|
| Log(n,m) | Логарифм n по основанию m |
| Log10(n) | Десятичный логарифм n |
| Exp(n) | e^x |
| Cos(x) | Косинус угла x, заданного в радианах |
| Sin(x) | Синус угла x, заданного в радианах |
| Tan(x) | Тангенс угла x, заданного в радианах |
| ASin(n) | Угол (в радианах), синус которого равен n |
| ACos(n) | Угол (в радианах), косинус которого равен n |
| ATan(n) | Угол (в радианах), тангенс которого равен n |
| Sqrt(n) | Квадратный корень из n |
| Max(a,b) | Возвращает наибольшее из двух чисел |
| Min(a,b) | Возвращает наименьшее из двух чисел |
| Pow(a,b) | Возвращает результат возведения числа a в степень b. |
| r.Next(n) | Случайное число из диапазона [0..n). r – объект типа System.Random |
| r.NextDouble() | Случайное вещественное число из диапазона [0.0, 1.0) |

Помним, что оператор / , применимый к целым операндам, выдает целое число – целую часть от частного; Оператор % возвращает остаток от деления двух целых чисел ($3\%2=1$ $3/2=1$). Если нужно - используем явное преобразование типов. Например: (float) $3/2 = 1.5$

Функции манипулирования датами и временем

| | |
|---------------------|---|
| DateTime.Now | Структура типа DateTime. Текущие дата и время. |
| DateTime.Today | Структура типа DateTime. Текущая дата. |
| d.ToString() | Строка вида dd.mm.yyyy hh:mm:ss. d- объект типа DateTime. Например, 05.06.2007 10:00:00 |
| d.ToString(f) | Строка – дата и / или время. Вид строки определяет формат f. О спецификаторах формата см. ниже. |
| d.Day | Номер дня месяца |
| d.Month | Номер месяца |
| d.Hour | Час |
| d.Year | Год |
| d.Minute | Минуты |
| d.DayOfWeek | День недели |
| d.DayOfYear | Номер дня года (отчет от 1 января) |
| d.ToLongDateString | «Длинная» дата. Например: 5 января 2007 |
| d.ToShortDateString | «Короткая дата». Например: 05.01.2007 |
| d.ToLongTimeString | Время в формате hh:mm:ss |
| d.ToShortTimeString | Время в формате hh:mm |
| d.AddDays(n) | Дата, отстоящая от d на n дней. Положительное n |

| | |
|-----------------|---|
| | сдвигает дату вперед, отрицательное- назад. |
| d.AddYears(n) | Дата, отстоящая от d на n лет. |
| d.AddMonth(n) | Дата, отстоящая от d на n месяцев. |
| d.AddHours(n) | Дата, отстоящая от d на n часов. |
| d.AddMinutes(n) | Дата, отстоящая от d на n минут. |

Спецификаторы формата даты/ времени

| | |
|------|---|
| d | Показывает день месяца одной или двумя цифрами (1..9, 10..31) |
| dd | Показывает день месяца в виде двух цифр (01..31) |
| ddd | Показывает сокращенное название дня недели (пн..вс) |
| dddd | Показывает полное название дня недели (понедельник...) |
| h | Показывает одну (1..9) или две (10..12) цифры часа |
| hh | Показывает две цифры часа (01..12) |
| H | Показывает одну (1..9) или две (10..23) цифры часа |
| HH | Показывает две цифры часа (01..23) |
| m | Показывает одну (1..9) или две (10..59) цифры минут |
| mm | Показывает две цифры минут (01..59) |
| s | Показывает одну (1..9) или две (10..59) цифры секунд |
| ss | Показывает две цифры секунд (01..59) |
| M | Показывает одну (1..9) или две (10..12) цифры месяца |
| MM | Показывает две цифры месяца (01..12) |
| MMM | Показывает сокращенное название месяца (янв .. дек) |
| MMMM | Показывает полное название месяца (январь .. декабрь) |
| yy | Показывает две последние цифры года (00..99) |
| yyyy | Показывает все цифры года (0001..9999) |

Некоторые события

| | |
|----------|--|
| Click | При щелчке кнопкой мыши |
| Closed | Возникает сразу за событием Closing |
| Closing | Возникает в результате щелчка по системной кнопке Заккрыть окно |
| DblClick | При двойном щелчке кнопкой мыши |
| Enter | При получении элементом управления фокуса |
| KeyDown | При нажатии клавиши клавиатуры. Сразу за событием KeyDown возникает событие KeyPress. Если нажатая клавиша удерживается, то событие KeyDown возникает еще раз. Т.о, пара событий KeyDown-KeyPress генерируется до тех пор, пока не будет отпущена удерживаемая клавиша |
| KeyPress | При нажатии клавиши клавиатуры. Событие KeyPress возникает сразу после события KeyDown |
| KeyUp | При отпуске нажатой клавиши клавиатуры |
| Leave | При потере элементом управления фокуса |
| Load | В момент загрузки формы. Используется для |

| | |
|-----------|--|
| | инициализации программы. |
| MouseDown | При нажатии кнопки мыши |
| MouseMove | При перемещении мыши |
| MouseUp | При отпуске кнопки мыши |
| Paint | При появлении окна на экране в начале работы программы, после появления части окна, которая, например, была закрыта другим окном, и в других случаях. Только процедура обработки события Paint имеет доступ к классу Graphics, методы которого обеспечивают отображение графики. |
| Resize | При изменении размеров окна. Если размер формы увеличивается, то сначала происходит событие Paint, затем-Resize. Если размер формы уменьшается, то происходит только событие Resize |

Некоторые исключения

| | |
|--------------------------|--|
| FormatException | Ошибка преобразования. Возникает, если величина не приводится к требуемому типу. |
| IndexOutOfRangeException | Выход значений индекса за допустимые границы. Возникает, например, при обращении к несуществующему элементу массива. |
| OverflowException | Переполнение. Возникает, если результат выполнения операции выходит за границы допустимого диапазона. |
| FileNotFoundException | Ошибка ввода-вывода при выполнении файловых операций. Например, при обращении к несуществующему файлу. |

Индивидуальные задания

1. Написать программу «Продуктовый супермаркет»: выбирается ряд продуктов, рассчитывается их сумма с учетом скидки в 5% (скидка предоставляется, если покупка сделана с 8.00 до 12.00 по текущему времени)

2. В файле содержатся слова русского алфавита. Приложение случайным образом выбирает одно из слов и предлагает его отгадать. Каждая буква загаданного слова отображается звездочкой. Пользователь выбирает букву из алфавита, и, если она (или они) есть, то соответствующая звездочка заменяется на данную букву. Параллельно ведется учет количества сделанных попыток.

3. Написать программу «Магазин бытовой техники»: выбирается несколько товаров, рассчитывается их сумма с учетом скидки в 10%. Скидка предоставляется, если покупка сделана в выходной день (проверяется по текущей дате)

4. Написать программу- опросник, предлагающую нескольким пользователям ответить на вопрос: с какого языка программирования они начинали? Предлагаемые варианты ответа: C, Pascal и т.п. Вывести результаты опроса в наглядной форме через полосы разной длины, соответствующей процентному отношению количества ответов к общему числу опрошенных. Например:



5. Имеется файл с расшифровкой счета за телефонные разговоры (дата, номер телефона, количество минут). Узнать, на какой номер было совершено больше всего звонков.

6. Написать программу, при помощи которой можно рассчитать стоимость изготовления окна. Задается ширина и высота окна (для определения общей площади), тип стеклопакета (однокамерный, двухкамерный, трехкамерный), указывается, будет ли меняться подоконник.

7. Имеется файл с расшифровкой счета за телефонные разговоры (дата, номер телефона, количество минут). Узнать суммарное число минут разговоров по конкретному номеру.

8. Написать программу, тестирующую память. С интервалом в 1 секунду компьютер показывает 10 произвольных двузначных чисел, пользователь пытается их запомнить и затем ввести с клавиатуры. По окончании теста программа сообщает, сколько чисел пользователь запомнил верно.

9. Имеется файл с расшифровкой счета за телефонные разговоры (дата, номер телефона, количество минут). Узнать, когда были совершены звонки на заданный номер.

10. Написать программу-приветствие. В зависимости от текущего времени суток при запуске должно появляться сообщение «Доброе утро!», «Добрый день!», «Добрый вечер!» или «Доброй ночи!» на фоне соответствующего изображения (звезд, восхода, заката, солнечного дня).

11. Написать приложение, в котором настраивается цвет некоторого объекта через комбинацию красного, синего и зеленого цветов. Для выбора интенсивности вспомогательных цветов использовать регуляторы.

12. Написать программу, отвечающую за выдачу наличности в банкоматах. Пусть запрашиваемая клиентом сумма выдается максимально крупными купюрами (по принципу наименьшего количества купюр). Например: $2000 = 1000 + 1000$; $1700 = 1000 + 500 + 100 + 100$; В общем случае доступны купюры достоинством 5000, 1000, 500, 100, 50, 10 р. Если купюр некоторого достоинства нет (отметить, например, флажками), они заменяются на более мелкие. Если выдать требуемую сумму невозможно (например, запросили 1010р, а десятки закончились), сообщить об этом.

13. Написать программу, рассчитывающую сумму коммунальных платежей: есть базовые тарифы на отопление (на 1 м^2 площади), на воду (на 1 чел), на газ (на 1 чел), на текущий ремонт (на 1 м^2 площади). Задается метраж помещения, количество проживающих людей, сезон (осенью и зимой отопление дороже), наличие льгот (ветеран труда – 30 % от его части; ветеран войны – 50% от его части). Вывести таблицу со столбцами: Вид платежа, Начислено, Льготная скидка, Итого. Посчитать итоговую сумму.

14. Написать программу вычисления дохода по вкладу в банке. Сумма вклада, срок вклада и процентная ставка (годовых) задаются пользователем. Отследить, когда было использовано это приложение – вывести все даты и времена запуска во внешний файл.

15. Написать программу, решающую систему линейных уравнений вида:

$$a_1x + b_1y + c_1z = d_1$$

$$a_2x + b_2y + c_2z = d_2$$

$$a_3x + b_3y + c_3z = d_3$$

Коэффициенты уравнений задаются, на выходе должны получить одно из четырех сообщений: «Найдено единственное решение: $x = \dots$, $y = \dots$, $z = \dots$ »

=...», «Система уравнений не совместна», «Система имеет бесконечно много решений», «Заданные коэффициенты не корректны».

16. В файле хранятся фамилии сотрудников фирмы и их оклады. Написать программу, которая формирует новый файл, выводя помимо начальных данных столбец премий (50%) от оклада и итоговые суммы для каждого сотрудника.

17. В файле имеются результаты сессии одной студенческой группы (ФИО, оценки по математике, информатике, английскому языку). Вывести фамилии отличников.

18. Написать программу, выводящую некоторую строку с текстом посимвольно (перед выводом каждого следующего символа должен пройти некоторый промежуток времени)

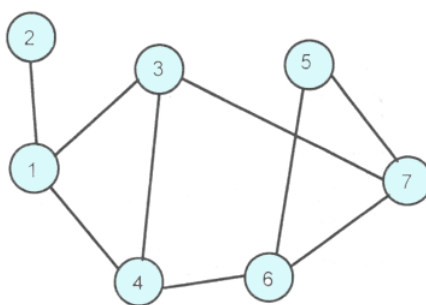
19. Написать программу, наглядно изображающую долю расходов человека на те или иные потребности. Пользователь вводит суммы, ежемесячно затрачиваемые им на еду, коммун. платежи, одежду, проезд, развлечения и пр. Программа рассчитывает процентное отношение этих сумм и выводит результат в виде круговой диаграммы.

20. Написать программу «Обслуживание по пластиковой карте». Пользователь вводит пароль от своей карты, если пароль подходит и требуемая сумма меньше остатка его средств на счете, снимает некоторую сумму. Данные о текущем состоянии счета перезаписываются (во внешнем файле). При трех попытках ввода неверного пароля счет блокируется (программа прекращает свою работу), если средств на счете меньше, чем требуется, выдается сообщение об этом.

21. Написать программу – тест. Перед началом тестирования вводится фамилия студента, затем он отвечает на ряд вопросов. Программа подсчитывает число правильных ответов и выдает оценку: если верных ответов более 90% - отлично, 70-89% - хорошо, 50-69% - удовлетворительно. Вывести в файл фамилии и оценки протестированных студентов.

22. Написать программу, определяющую дни недели, на которые выпадут ваши дни рождения в ближайшие 10 лет.

23. Написать приложение, решающую задачу поиска пути между двумя городами. Карта дорог между городами задана в виде графа — набора вершин, означающих города, и ребер, обозначающих дороги:



Если несколько городов соединены дорогами, то очевидно, что попасть из одного города в другой можно различными маршрутами. Задача состоит в нахождении всех возможных маршрутов

24. Написать приложение, реализующую игру «Крестики-нолики». На поле размером 3x3 игроки по-очереди ставят крестики и нолики, выигрывает тот, кто первым выстроит три своих знака в ряд или по диагонали. Пусть игроками выступают компьютер и человек.

25. В текстовом файле даны значения объемов экспорта нефти по странам (Название страны, Объем). Найти общий объем экспортируемой нефти и долю каждой страны в общем объеме (в процентах). Полученные результаты вывести в файл.

26. В файле хранится информация об объеме продаж товара за пять последних месяцев. С помощью метода наименьших квадратов спрогнозировать объемы продаж на следующие три месяца. В качестве линии тренда выбрать линейную функцию.

27. В файле имеются результаты сессии одной студенческой группы (ФИО, оценки по математике, информатике, английскому языку). Рассчитать и вывести среднюю для группы оценку по каждой из дисциплин.

28. В файле имеются результаты сессии одной студенческой группы (ФИО, оценки по математике, информатике, английскому языку). Организовать доступ к этому файлу через некоторый пароль с возможностью перезаписи; вывести для каждого студента его средний балл.

29. В поле вводится некоторый текст. Найти и выделить в нем самое длинное слово.

30. Написать программу для нахождения оптимальных значений переменных x_1 и x_2 , соответствующих системе ограничений:

$$a_{11}x_1 + a_{12}x_2 \leq b_1$$

$$a_{21}x_1 + a_{22}x_2 \leq b_2$$

таких, чтобы функция $Z = c_1x_1 + c_2x_2$ принимала максимальное значение. Коэффициенты a_{ij} , b_i , c_j задаются. Программа должна реализовать симплекс-метод.

31. В файле имеются результаты сессии одной студенческой группы (ФИО, оценки по математике, информатике, английскому языку). Сформировать два файла – в один занести данные по студентам, сдавшим сессию на 3, 4, 5; в другой- фамилии и оценки студентов, имеющих двойки.
32. В файле хранятся имена и даты рождения ваших друзей. Написать приложение, которое по введенному месяцу напоминает, у кого скоро (в этом месяце) день рождения.
33. Написать программу, в которой некоторый объект (например, прямоугольник) через небольшие интервалы времени меняет свой цвет.
34. В файле хранится прайс-лист продукции некоторой фирмы (название товара, цена). Вывести на экран этот лист, предварительно отсортировав его либо по ценам, либо по наименованиям (способ сортировки указывается дополнительно).
35. В файле хранится прайс-лист продукции некоторой фирмы (название товара, цена). Выбирая из списка некоторое название, получить цену этого товара. Элементы списка формируются в момент запуска приложения и соответствуют перечню товаров из файла.
36. Написать программу «Ежедневник». Пусть при запуске приложения выдается список дел на сегодняшнюю дату (данные берутся из файла). Реализовать возможность добавления в ежедневник новых событий через отдельное окно.
37. Написать программу, отображающую график функции $y = \sin(x)$.
38. Написать приложение, выдающее расписание занятий вашего потока. Номер группы, день недели, тип недели (первая или вторая) задаются пользователем.
39. Пользователь задает координаты центра и радиус круга. Сформировать таблицу с произвольными координатами десяти точек. Определить, какие точки будут принадлежать данному кругу (выделить соответствующие ячейки цветом или сформировать дополнительный столбец с результатами – на ваш выбор)
40. Пользователь определяет область координатной плоскости (максимумы и минимумы по осям x и y). Компьютер произвольным образом загадывает левый верхний угол и длину стороны некоторого квадрата, целиком лежащего в этой области. Рассчитать вероятность, с которой предложенная пользователем точка (с целочисленными координатами), попадет в задуманный квадрат. Проверить экспериментально.
41. В старояпонском календаре был принят 12-ти летний цикл. Годы внутри цикла носили названия животных: крысы, коровы, тигра, зайца,

дракона, змеи, лошади, овцы, обезьяны, петуха, собаки и свиньи. Составить программу, которая по введенному номеру года печатает его название по старояпонскому календарю.

42. Написать программу, которая вычисляет стоимость междугородного телефонного разговора. Исходными данными являются код города и длительность разговора в минутах. Результатными данными- название города и итоговая стоимость разговора.

43. Написать программу, реализующую умножение матриц размерности 3×3 .

44. В файле хранятся оценки студентов по информатике (ФИО, оценка). Рассчитать процентное отношение этих оценок (например «Отлично»- 45%, «Хорошо»- 20%, «Удовлетворительно»- 30%, «Неудовлетворительно»- 5%)

45. В файле хранятся сведения о продажах автомобилей (дата, марка, цена). Определить, какие автомобили были проданы в указанную дату, рассчитать суммарную выручку за этот день.

46. Написать приложение, реализующее работу с массивами: 1. Произвести сортировку числовых данных одномерного массива по возрастанию значений. 2. Посчитать, сколько в матрице различных элементов. Задачи оформить на трех формах: главная с меню, форма для работы с одномерным массивом; форма для работы с двумерным массивом. Предусмотреть возможность ввода данных из файла, с клавиатуры, через генератор случайных чисел.

47. Написать приложение, реализующее работу с массивами: 1. Посчитать количество простых чисел в одномерном массиве. 2. Указать, на каких позициях в матрице находятся элементы, большие введенного с клавиатуры числа. Задачи оформить на трех формах: главная с меню, форма для работы с одномерным массивом; форма для работы с двумерным массивом. Предусмотреть возможность ввода данных из файла, с клавиатуры, через генератор случайных чисел.

48. Написать приложение, реализующее работу с массивами: 1. Определить, на каких позициях в одномерном массиве находятся простые числа. 2. Указать, на каких позициях в матрице находятся элементы, большие ее среднеарифметического значения. Задачи оформить на трех формах: главная с меню, форма для работы с одномерным массивом; форма для работы с двумерным массивом. Предусмотреть возможность ввода данных из файла, с клавиатуры, через генератор случайных чисел.

49. Написать приложение, реализующее работу с массивами:

1. Вывести на экран в нескольких строках количество «звездочек», равное текущему значению элемента одномерного массива:

| | | | | |
|---|---|---|---|----|
| 1 | 3 | 5 | 2 | 10 |
|---|---|---|---|----|

```
*
***
*****
**
*****
```

2. Поменять местами в матрице наибольший и наименьший элементы. Задачи оформить на трех формах: главная с меню, форма для работы с одномерным массивом; форма для работы с двумерным массивом. Предусмотреть возможность ввода данных из файла, с клавиатуры, через генератор случайных чисел.

50. Написать приложение, реализующее работу с массивами:

1. Вывести в разные строки положительные и отрицательные элементы одномерного массива :

| | | | | |
|---|----|----|----|---|
| 4 | -3 | 13 | -5 | 6 |
|---|----|----|----|---|

Положительные элементы: 4 13 6

Отрицательные элементы: -3 -5

2. Реализовать сложение, вычитание, умножение двух матриц 3x3. Задачи оформить на трех формах: главная с меню, форма для работы с одномерным массивом; форма для работы с двумерным массивом. Предусмотреть возможность ввода данных из файла, с клавиатуры, через генератор случайных чисел.

Библиографический список

1. Бабушкина И.А., Окулов С.М. Практикум по объектно-ориентированному программированию. : - М:БИНОМ, Лаборатория знаний, 2004. – 366с.
2. Культин Н.Б. С# в задачах и примерах. -СПб.:БХВ-Петербург, 2007.-240с
3. Мохика Х. Язык С# . Разработка Web-приложений на ASP.NET. – М.: НТ Пресс, 2006.- 464с.
4. Троелсен Э. С# и платформа .NET. – СПб.: Питер, 2007.-796с.
5. Фаронов В.В. Программирование на языке С#. – СПб.: Питер, 2007. -240с.
6. <http://forums.realcoding.net>
7. <http://www.sql.ru>
8. <http://www.gotdotnet.ru/LearnDotNet/CSharp/default.aspx>
9. <http://support.microsoft.com>