Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Кемеровский государственный университет»
Кузбасский гуманитарно-педагогический институт

Факультет информатики, математики и экономики
Кафедра математики, физики и математического моделирования

Ю.С. Гаврилова

ЯЗЫКИ И МЕТОДЫ ПРОГРАММИРОВАНИЯ

Разработка программного приложения с пользовательским интерфейсом на языке C#. Основные элементы

Методические рекомендации по изучению дисциплины
для обучающихся по направлениям подготовки
01.03.02 Прикладная математика и информатика,
02.03.03 Математическое обеспечение и администрирование информационных систем

Новокузнецк

2022

УДК [378.147.88: 004.432.2](072)
ББК 74.484(2Рос-4Кем)я73+32.973-018.6я73
Г 12

Гаврилова Ю.С.

Языки и методы программирования. Разработка программного приложения с пользовательским интерфейсом на языке C#. Основные элементы: методические рекомендации по изучению дисциплины для студентов факультета информатики, математики и экономики, обучающихся по направлениям подготовки 01.03.02 Прикладная математика и информатика и 02.03.03 Математическое обеспечение и администрирование информационных систем / Ю.С. Гаврилова. – Новокузнецк : КГПИ ФГБОУ ВО «КемГУ», 2022. – 39 с.

Рекомендовано на заседании
кафедры математики, физики и
математического моделирования
Протокол № 5 от 15.12.2022
Заведующий каф. МФММ

/ Е.В. Решетникова

Утверждено методической комиссией
факультета информатики, математики и
экономики
Протокол № 5 от 15.12.2022
Председатель методической комиссии
ФИМЭ

/ И.А. Жибинова

УДК [378.147.88: 004.432.2](072)
ББК 74.484(2Рос-4Кем)я73+32.973-018.6я73
Г 12

Содержание

ПРЕДИСЛОВИЕ 4
1 Теоретические сведения 5
2 Лабораторная работа «Пользовательский интерфейс в C#» 28
2.1 Задания 28
2.2 Требования к отчету по лабораторной работе 29
2.3 Контрольные вопросы 30
3 Образец тестовых заданий 31
4 Индивидуальные задания 35
5 Рекомендуемая литература 38
6 Современные профессиональные базы данных и справочные системы 39

ПРЕДИСЛОВИЕ

Настоящие методические рекомендации адресованы студентам, получающим квалификацию бакалавр по направлениям подготовки: 01.03.02 Прикладная математика и информатика и 02.03.03 Математическое обеспечение и администрирование информационных систем и направлены на оказание помощи студентам в подготовке к выполнению лабораторных работ и индивидуальных заданий, прохождению тестирования по теме.

Язык C# – один из базовых языков программирования, изучение которого необходимо для студентов вуза, поскольку именно на данном языке можно разрабатывать десктопные приложения, мобильные и веб-приложения, игры.

Данные методические материалы позволят студенту закрепить теоретический материал; подготовиться к лабораторным занятиям, прохождению тестирования и выполнению индивидуальных заданий по соответствующей теме.

Методические рекомендации могут оказаться полезными при выполнении проектных работ, прохождении учебных и производственных практик, написании курсовых и выпускных квалификационных работ.

1 Теоретические сведения

Windows Forms – это платформа пользовательского интерфейса для создания классических приложений Windows. Основным элементом, на который выводится информация для пользователя, является форма (рисунок 1).

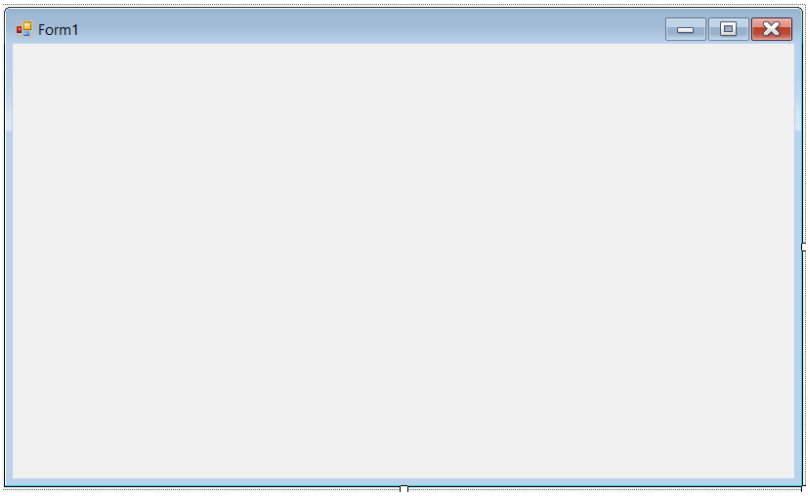


Рисунок 1 – Форма приложения

Обычно приложение Windows Forms строится путем добавления элементов управления на формы и создания кода для реагирования на действия пользователя, такие как щелчки мыши или нажатия клавиш.

Элемент управления – это отдельный элемент пользовательского интерфейса, предназначенный для отображения или ввода данных.

Платформа **Windows Forms** предоставляет разнообразные элементы управления, которые условно можно разделить на два типа: интерфейсные – те, которые видны пользователю и с которыми он может работать непосредственно (кнопки, панели, таблицы) и служебные – те, что выполняют определенные задачи и вызываются путем взаимодействия пользователя с интерфейсными элементами (диалоги, таймеры, адаптеры).

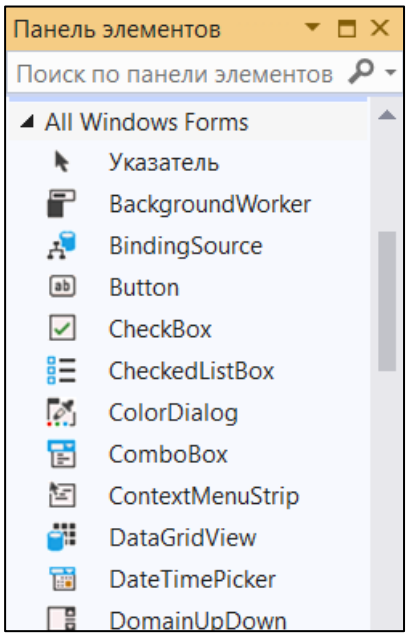
Перед использованием элемент должен быть помещен на форму (обычно при помощи drag&drop), за исключением самого элемента формы (элемент форма создается автоматически при создании проекта).

На рисунке 2 представлены элементы управления:

- на рисунке 2а – панель элементов управления, которую можно отобразить в проекте с помощью кнопок меню View(Вид)->ToolBox (Панель

элементов);

➢ на рисунке 2б – вид некоторых элементов управления после помещения их на форму.

а) 

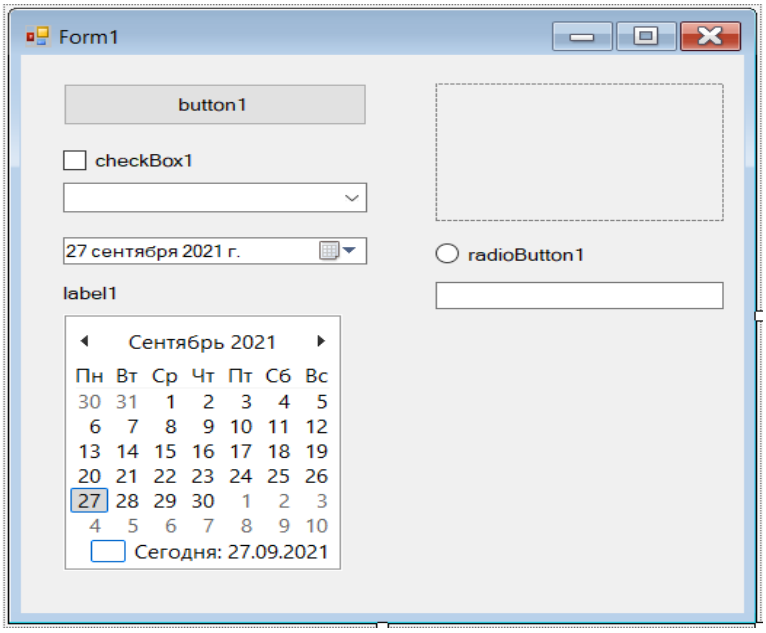
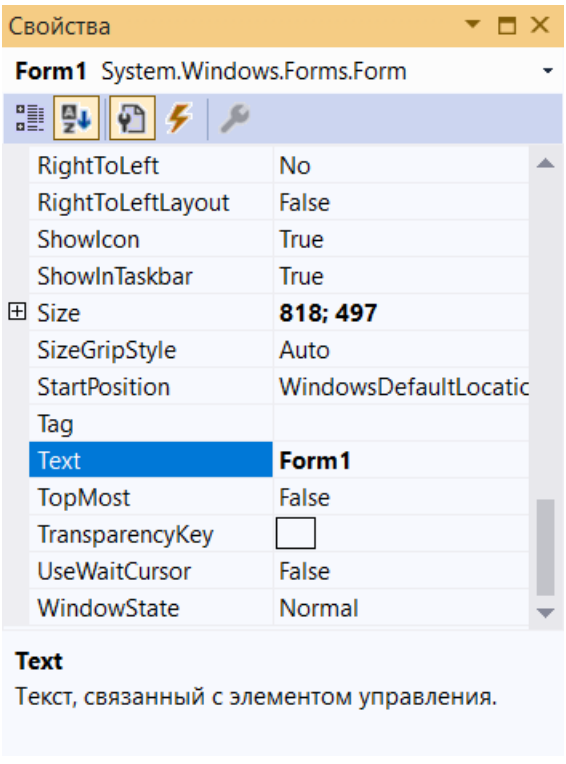
б) 

Рисунок 2 – Элементы управления

У формы и у любого элемента управления есть свойства и события. На рисунке 3 представлены свойства и события формы проекта.

а) 

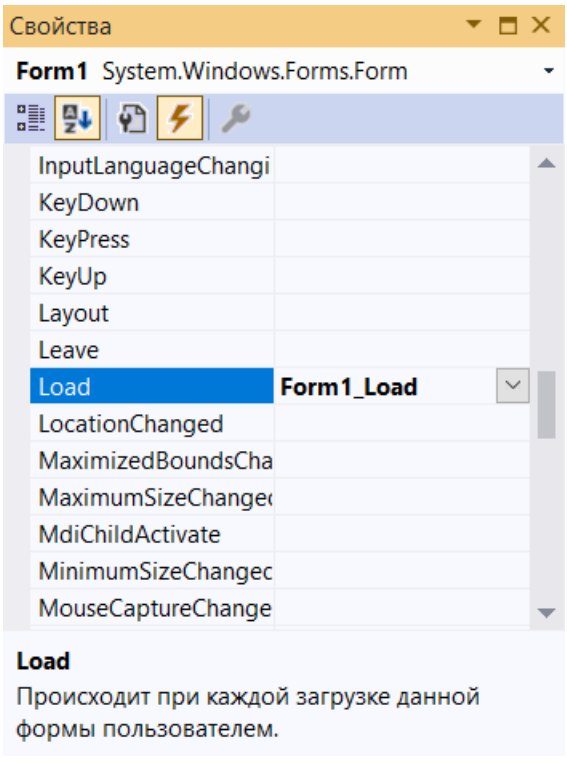
б) 

Рисунок 3 – Элементы управления

Свойства задают, например, внешний вид приложения – размер окна или

кнопки, цвет и шрифт текста в текстовом поле и т.д. Например, на рисунке 3а выделено свойство Text, значение которого изменит надпись в заголовке окна приложения.

При выполнении пользователем какого-либо действия с формой или одним из ее элементов управления создается **событие**. К событиям относятся, например, нажатие на кнопку, изменение текста в текстовом поле, выбор одного из пунктов выпадающего меню и т.д. На рисунке 3б представлено событие Form1_Load, которое произойдет при загрузке этой формы пользователем.

Для того чтобы в коде приложения можно было задать команды, которые выполнятся при возникновении соответствующего события, нужно сначала создать обработчик этого события.

Если два раза щелкнуть левой кнопки мыши по кнопке button1, расположенной на форме, то Visual Studio автоматически сгенерирует обработчик события «Нажатие левой кнопкой мыши на кнопку button1»:

```
private void button1_Click(object sender, EventArgs e)
  {

  }
```

Для того чтобы добавить обработчик других событий, нужно на вкладке событий панели форм выбрать необходимое событие и два раза щелкнуть левой кнопки мыши по пустому полю рядом с названием события. Visual Studio автоматически сгенерирует обработчик нужного события.

Если просто скопировать приведенный выше текст в код проекта, то ничего не произойдет, поскольку код не «привязан» к событию, не является обработчиком, а является просто отдельной процедурой, которую дальше следует отдельно вызывать.

Рассмотрим некоторые элементы управления, некоторые их свойства и события.

Кнопка (Button)

Основные свойства кнопки:

- Text – получает или задает текст, сопоставленный с этим элементом управления;

- TextAlign – получает или задает выравнивание текста в элементе управления «Кнопка»;
- Image – возвращает или задает изображение, отображаемое на элементе управления «Кнопка»;
- Visible – получает или задает значение, указывающее, отображается ли элемент управления;
- Enabled – возвращает или задает значение, показывающее, сможет ли элемент управления отвечать на действия пользователя;
- Name – возвращает или задает имя элемента управления. Стандарты те же что и для переменных C++.

Основные события кнопки:

- Click – происходит при щелчке по элементу управления;
- DoubleClick – происходит при двойном щелчке мышью по элементу управления Button.

Пример 1. При нажатии на кнопку проверить, окрашена ли кнопка, если кнопка не окрашена – покрасить ее в зеленый цвет, если окрашена – убрать цвет.

Добавляем на форму один элемент управления – button1, после чего двойным щелчком мыши вызываем обработчик события «Нажатие на кнопку».

Чтобы покрасить кнопку в какой-либо цвет, нужно присвоить значение с этим цветом в свойство BackColor кнопки. Например, чтобы присвоить зеленый цвет, нужно выполнить команду:

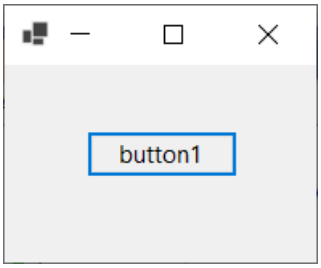
```
button1.BackColor = Color.Green;
```

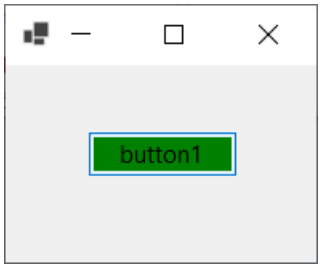
Цвет, который установлен по умолчанию для кнопки – Control.DefaultBackColor. Тогда можно проверить следующее: если кнопка не окрашена в цвет по умолчанию, то окрашиваем ее в этот цвет, иначе красим в зеленый. Код в обработчике события будет выглядеть следующим образом:

```
private void button1_Click(object sender, EventArgs e)
  {
     if (button1.BackColor != Control.DefaultBackColor) {
         button1.BackColor = Control.DefaultBackColor;
         }
     else{
         button1.BackColor = Color.Green;
```

```
        }
    }
```

Если запустить программу, то получим представленный на рисунке 4 результат: 4а – форма до нажатия на кнопку, 4б – форма после нажатия на кнопку и 4в – форма после повторного нажатия на кнопку.

а) 

б) 

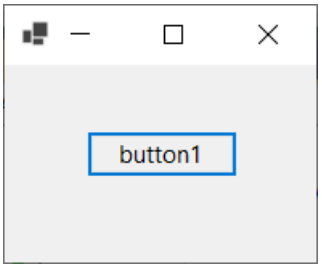
в) 

Рисунок 4 – Результат запуска программы

Текстовое поле (TextBox)

Текстовое поле – это элемент управления текстовым окном для ввода данных пользователем.

Основные свойства текстового поля:

- Text – получает или задает текст, сопоставленный с этим элементом управления;
- Multiline – определяет поддерживает ли элемент многострочный текст;
- Name – возвращает или задает имя элемента управления. Стандарты те же что и для переменных C++;
- ReadOnly – возвращает или задает значение, определяющее возможность изменения содержимого элемента управления TextBox;
- Rows – возвращает или задает число строк, отображаемых в многострочном текстовом окне.

Основные события текстового поля:

- Click – происходит при щелчке мышью по элементу управления;
- DoubleClick – происходит при двойном щелчке мышью по элементу управления TextBox;
- KeyDown – происходит при нажатии клавиши на клавиатуре, когда фокус находится на элементе TextBox;

- TextChanged – происходит при изменении содержимого текстового окна TextBox.

Пример 2. При нажатии на кнопку взять из textBox1 введенную в него строку и заменить каждый второй символ знаком подчеркивания. Полученный результат поместить в textBox2.

Разместим на форме 2 элемента textbox и один элемент button. Для того чтобы можно было работать с содержимым, записанным в textbox1, как со строкой, необходимо объявить переменную строкового типа и присвоить ей значение textbox1.Text:

```
string str = textBox1.Text;
```

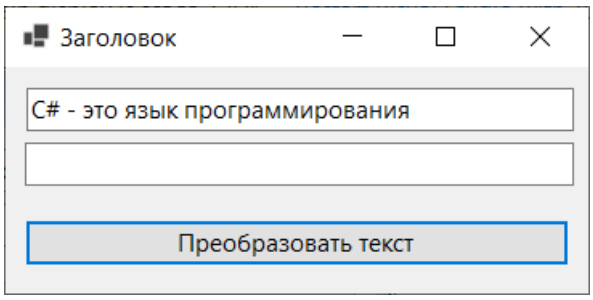
Длину строки str можно получить с помощью str.Length, а обратиться к ее i-му символу можно с помощью str[i]. Далее в цикле for можно пройти по всем символам строки и сохранить из нее каждый второй в строку str2.

Необходимо учесть, что нумерация символов в строке начинается с 0, поэтому каждый второй символ – это все символы с нечетным индексом. Тогда если остаток от деления индекса символа на 2 равен нулю, то этот символ переносим из первой строки во вторую без изменений, иначе – во вторую строку вставляем символ подчеркивания.

Код в обработчике события будет выглядеть следующим образом:

```
private void button1_Click(object sender, EventArgs e)
  {
     string str = textBox1.Text;
     string str2 = "";
     for (int i = 0; i < str.Length; i++){
          if (i % 2 == 0) {
              str2 = str2 + str[i];
              }
          else{
              str2 = str2 + "_";
              }
          }
     textBox2.Text = str2;
  }
```

Если запустить программу, то получим представленный на рисунке 5 результат: 5а – форма до нажатия на кнопку, 5б – форма после нажатия на кнопку.

а) 

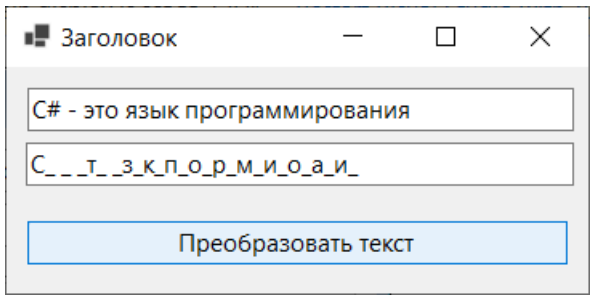
б) 

Рисунок 5 – Результат запуска программы

Метка (Label)

Метка представляет собой стандартную надпись Windows и находится в группе элементов Common Controls.

Основные свойства метки:

- Text – получает или задает текст, сопоставленный с этим элементом управления;
- TextAlign – получает или задает выравнивание текста в элементе управления Label;
- Image – возвращает или задает изображение, отображаемое на элементе управления Label;
- Visible – получает или задает значение, указывающее, отображается ли элемент управления;
- Enabled – возвращает или задает значение, показывающее, сможет ли элемент управления отвечать на действия пользователя;
- Name – возвращает или задает имя элемента управления. Стандарты те же что и для переменных C++.

Основные события метки:

- Click – происходит при щелчке по элементу управления;
- DoubleClick – происходит при двойном щелчке мышью по элементу управления Label.

Пример 3. Для двух целых чисел из textBox1 и textBox2 организовать вывод большего из них в textBox3. Если числа в текстовых полях одинаковые – значение также должно быть выведено в textBox3.

Если на форме расположить только 3 элемента textBox и 1 элемент button,

то интерфейс программы будет непонятен пользователю. Однако если мы добавим 3 метки, написав в них «Первое число», «Второе число», «Большее из них» (рисунок 6), то сразу становится понятно, что делает программа. В данном случае метки нужны нам для того чтобы сделать интерфейс более простым и понятным.

Для того чтобы присвоить переменной целого типа значение из textbox, необходимо выполнить преобразование типов с помощью метода Parse или Convert. Использование этих методов дает одинаковый результат.

С помощью метода Parse запишем в переменную x значение из textbox1:

```
int x = Int32.Parse(textBox1.Text);
```

Теперь выполним то же самое с помощью метода Convert:

```
int x = Convert.ToInt32(textBox1.Text);
```

Чтобы вывести значение целого типа в текстовое поле, также необходимо выполнить преобразование типов. Выполним это с помощью метода Convert:

```
textBox2.Text = Convert.ToString(x);
```

Код в обработчике события будет выглядеть следующим образом:

```
private void button1_Click(object sender, EventArgs e)
  {
     int x = Convert.ToInt32(textBox1.Text);
     int y = Convert.ToInt32(textBox3.Text);
     if (x>y){
         textBox2.Text = Convert.ToString(x);
         }
     else{
         textBox2.Text = Convert.ToString(y);
     }
  }
```

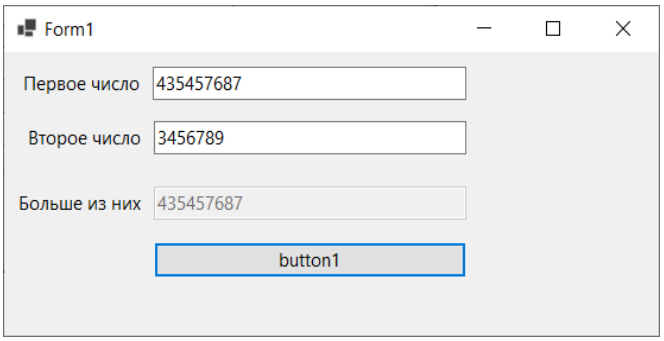


Рисунок 6 – Пример использования на форме элементов Label

Пример 4. Для предыдущего задания добавить окрашивание текстового поля с наибольшим числом. Если числа в текстовых полях одинаковые – окрасить

оба текстовых поля.

В примере 3 не был рассмотрен отдельно вариант, когда введенные в текстовые поля числа одинаковые. Это было не обязательно, поскольку этот вариант подходил под часть, описанную в else. Однако в данном примере необходимо реализовать одновременное окрашивание двух текстовых полей, если введенные в них числа одинаковые.

Окрашивание текстового поля будем реализовывать следующим образом:

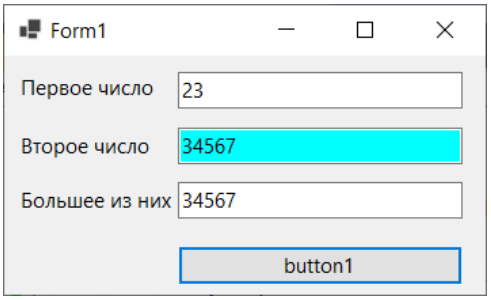
```
textBox2.BackColor = Color.Aqua;
```

Кроме того, окрашивая одно текстовое поле, нам нужно реализовать сброс цвета для другого поля. Если этого не сделать, то повторное нажатие на кнопку может привести к окрашиванию второго текстового поля (если изменить введенные в текстовые поля числа).

Код в обработчике события будет выглядеть следующим образом:

```
private void button1_Click(object sender, EventArgs e)
  {
     int a = Convert.ToInt32(textBox1.Text);
     int b = Convert.ToInt32(textBox2.Text);
     if (b > a) {
         textBox1.BackColor = Color.White;
         textBox2.BackColor = Color.Aqua;
         }
     else if (b < a){
         textBox2.BackColor = Color.White;
         textBox1.BackColor = Color.Aqua;
         }
     else{
         textBox1.BackColor = Color.Aqua;
         textBox2.BackColor = Color.Aqua;
         }
  }
```

Результат работы программы представлен на рисунке 7.

а) 

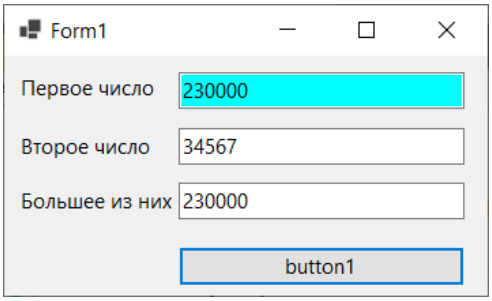
б) 

Рисунок 7 – Результат работы программы

Пример 5. Для предыдущего задания добавить окрашивание текстового

поля с наибольшим числом и вывод наибольшего из чисел в textBox3 не только при нажатии кнопки, но и при изменении текста в textBox1 и textBox2.

Для того чтобы выполнить это задание, нужно добавить обработчик события «Изменение текста» (TextChanged) для textBox1 и textBox2. В целом, код, который размещен в обработчике события button1_Click, можно просто скопировать в обработчики textBox1_TextChanged и textBox2_TextChanged. Однако наилучшим решением будет вынести данный код в отдельную процедуру, а затем просто вызывать ее в обработчике:

```
public void Sravn()
  {
     int a = Convert.ToInt32(textBox1.Text);
     int b = Convert.ToInt32(textBox2.Text);
     if (a > b){
         textBox2.BackColor = Color.White;
         textBox1.BackColor = Color.Aqua;
         }
     else if (b > a) {
         textBox1.BackColor = Color.White;
         textBox2.BackColor = Color.Aqua;
         }
     else {
         textBox1.BackColor = Color.Aqua;
         textBox2.BackColor = Color.Aqua;
         }
   }
```

Код в обработчиках событий будет выглядеть следующим образом:

```
private void button1_Click(object sender, EventArgs e){
     Sravn();
  }
private void textBox1_TextChanged(object sender, EventArgs e){
     Sravn();
  }
private void textBox2_TextChanged(object sender, EventArgs e){
     Sravn();
  }
```

«Флажок» RadioButton

RadioButton представляет собой флажок, который позволяет пользователю выбрать одну опцию (пункт) из предопределенного набора (группы). Находится в группе элементов Common Controls..

Основные свойства RadioButton:

- Text – получает или задает текст, сопоставленный с этим элементом

управления;

- Visible – получает или задает значение, указывающее, отображается ли элемент управления;
- Enabled – возвращает или задает значение, показывающее, сможет ли элемент управления отвечать на действия пользователя;
- Name – возвращает или задает имя элемента управления. Стандарты те же что и для переменных C++;
- Checked – возвращает или задает значение, указывающее, находится ли RadioButton во включенном состоянии.

Основные события RadioButton:

- Click – происходит при щелчке по элементу управления;
- DoubleClick – происходит при двойном щелчке мышью по элементу управления RadioButton;
- CheckedChanged – возникает при переходе RadioButton во включённое состояние.

Пример 6. Организовать обработку выбора цвета кнопки через элемент RadioButton.

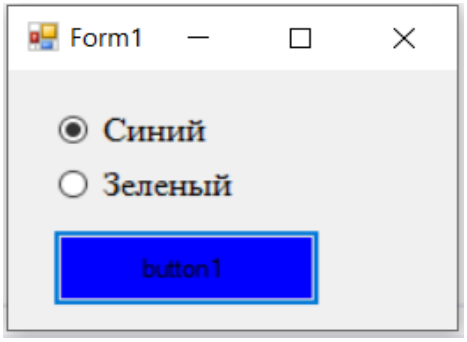
Свойство Checked у элемента RadioButton принимает значение true в том случае, если данный элемент включен. Создадим отдельную процедуру, которая красит кнопку в текст, соответствующий включенному RadioButton:

```
public void Painting()
  {
     if (radioButton1.Checked == true)
           {button1.BackColor = Color.Blue;}
     if (radioButton2.Checked == true)
           {button1.BackColor = Color.Green;}
  }
```

Вызов процедуры в обработчике события «Нажатие на кнопку» выглядит следующим образом:

```
private void button1_Click(object sender, EventArgs e)
  {
     Painting();
  }
```

Результат работы программы представлен на рисунке 8.

а) 

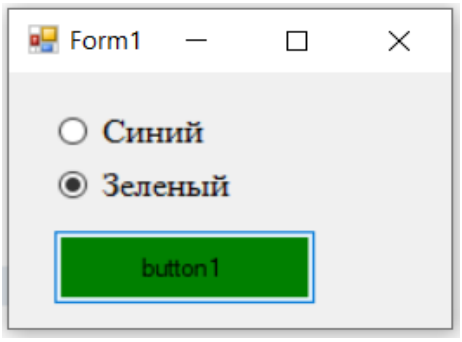
б) 

Рисунок 8 – Результат работы программы

«Флажок» CheckBox

CheckBox представляет элемент управления – флажок, который пользователь может устанавливать и снимать. Находится в группе элементов Common Controls.

Основные свойства CheckBox:

- Text – получает или задает текст, сопоставленный с этим элементом управления;
- Visible – получает или задает значение, указывающее, отображается ли элемент управления;
- Enabled – возвращает или задает значение, показывающее, сможет ли элемент управления отвечать на действия пользователя;
- Name – возвращает или задает имя элемента управления. Стандарты те же что и для переменных C++;
- Checked – возвращает или задает значение, указывающее, находится ли CheckBox во включенном состоянии;
- CheckState – Возвращает или задает состояние CheckBox. CheckState позволяет задать для флажка одно из трех состояний – Checked (отмечен), Indeterminate (флажок не определен – отмечен, но находится в неактивном состоянии) и Unchecked (не отмечен).

Основные события CheckBox:

- Click – происходит при щелчке по элементу управления;
- DoubleClick – происходит при двойном щелчке мышью по элементу управления RadioButton;

- CheckedChanged – возникает при переходе CheckBox из одного состояния в другое.

Пример 7. Организовать обработку выбора цвета кнопки через элемент CheckBox.

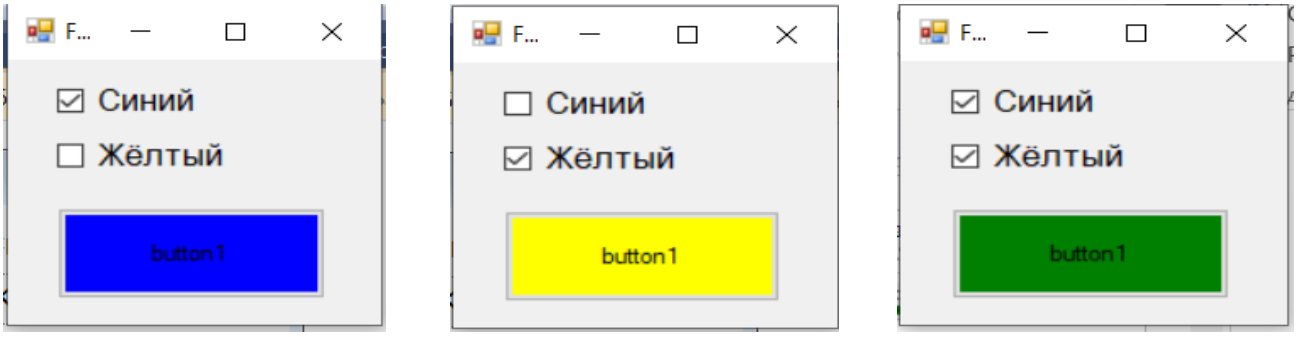
Добавим на форму 2 элемента CheckBox и 1 элемент Button. Организуем раскраску кнопки в отдельной процедуре по следующему алгоритму: если отмечен checkBox1, то красим кнопку в синий цвет, если отмечен checkBox2 – в желтый, а если отмечены они оба – в зеленый. Код процедуры выглядит следующим образом:

```
public void Painting()
  {
     button1.BackColor = Color.White;
     if (checkBox1.Checked == true){
         button1.BackColor = Color.Blue;}
     if (checkBox2.Checked == true) {
         button1.BackColor = Color.Yellow;}
     if ((checkBox1.Checked == true) &&
         (checkBox2.Checked == true)){
         button1.BackColor = Color.Green;}
  }
```

В обработчике события будем вызывать процедуру:

```
private void button1_Click(object sender, EventArgs e)
  {
     Painting();
  }
```

Результат представлен на рисунке 9.



а) б) в)

Рисунок 9 – Результат работы программы

ListBox

Пример 8. Организовать добавление текста в listBox через элемент textBox. Организуем по нажатию кнопки button1 прямое добавление элементов в

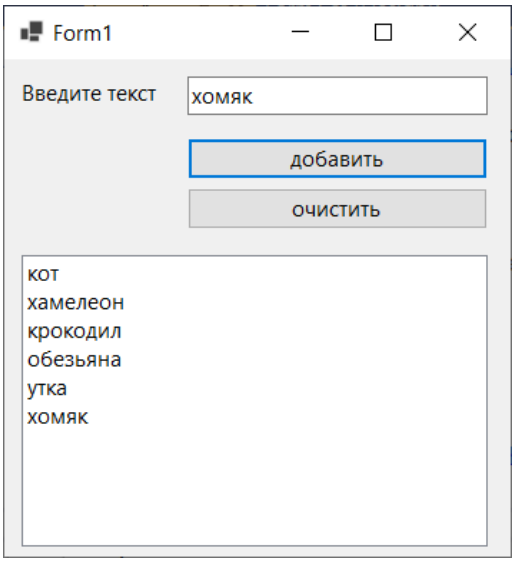
коллекцию Items компонента ListBox с помощью listBox1.Items.Add(). В скобках должны быть указаны значения, которые будут переданы в коллекцию Items:

```
private void button1_Click(object sender, EventArgs e)
  {
     listBox1.Items.Add(textBox1.Text);
  }
```

Создадим обработчик нажатия на кнопку «Очистить». Эта кнопка будет очищать listBox1 с помощью listBox1.Items.Clear():

```
private void button2_Click(object sender, EventArgs e)
  {
     listBox1.Items.Clear();
  }
```

Результат работы программы представлен на рисунке 10. На рисунке 10а представлен результат добавления текста в listBox, а на рисунке 10б – вид окна после нажатия на кнопку «Очистить».

а) 

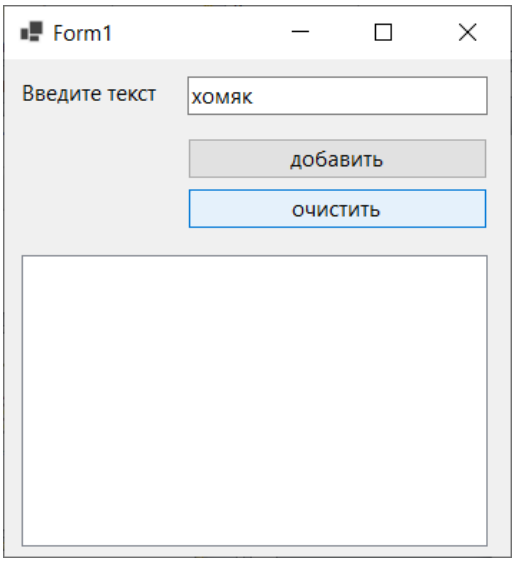
б) 

Рисунок 10 – Результат работы программы

Пример 9. Реализовать программу: на форме (рисунок 11) в метке Label находится какой-либо вопрос, есть 2 кнопки с разными ответами, одна кнопка статична, другая «убегает» от указателя мыши.

Для решения данной задачи потребуется событие, возникающее, когда указатель мыши попадает на кнопку. Таким событием является **MouseEnter**. После наведения указателя мыши на кнопку положение данной кнопки на форме должно измениться. Менять положение будем с помощью двух параметров **button2.Left** и **button2.Top**, задающих отступ от границы формы слева и сверху соответственно.

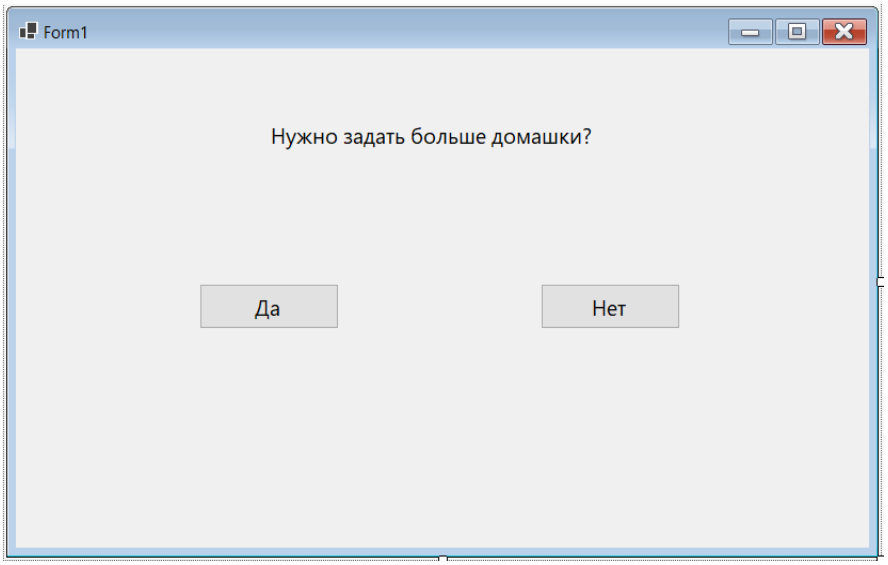


Рисунок 11 – Возможный вид формы приложения для примера 9

Отступ нужно организовать так, чтобы кнопка не «вылетала» за видимую часть формы, поэтому можно взять генератор случайных чисел и задать диапазон от 0 до ширины формы. При этом если случайное число приближается к правой и нижней границам формы, то кнопка будет частично невидима, поэтому следует из характеристик формы (Width, Height) вычесть соответствующие характеристики кнопки.

Код в обработчике события будет выглядеть следующим образом:

```
private void button2_MouseEnter(object sender, EventArgs e)
  {
     Random rnd = new Random();
     button2.Left = rnd.Next(this.Width - 2 * button2.Width);
     button2.Top = rnd.Next(this.Height - 2 * button2.Height);
  }
```

Пример 10. Для примера 7 реализовать вывод пользователю сообщения, информирующего его о зеленом цвете кнопки.

Вывод пользователю сообщения реализуется с помощью элемента MessageBox. Сообщение, которое мы хотим показать пользователю, нужно передать в метод Show() этого элемента. Если мы хотим просто показать сообщение с текстом «Оно зелёное!», то добавляем в обработчик к условию ((checkBox1.Checked == true) && (checkBox2.Checked == true)) следующую строку:

```
MessageBox.Show("Оно зелёное!");
```

Если же хотим получить более сложное окно, с заголовком (рисунок 12), то

строка будет выглядеть следующим образом:

```
MessageBox.Show("Оно зелёное!", "Информация");
```

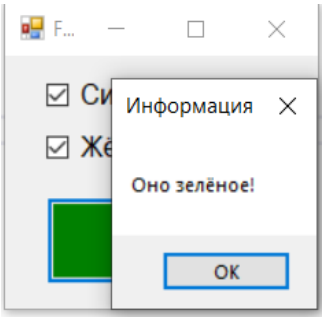


Рисунок 12 – Вывод сообщения пользователю

PictureBox

Элемент PictureBox нужен для отображения на форме файлов в формате bmp, jpg, gif, а также метафайлов изображений и иконок.

Пример 11. Реализовать программу: Проводники с сопротивлением 20 Ом, 15 Ом и 30 Ом соединены последовательно / параллельно. Нарисуйте схему соединения проводников. Определите при заданном значении силы тока (напряжения) оставшиеся характеристики цепи.

На форме необходимо разместить следующие элементы: PictureBox для отображения схемы соединения проводников, 2 элемента RadioButton для выбора типа соединения (последовательное / параллельное), одну кнопку Button для запуска расчетов, метки и текстовые поля в соответствии с рисунком 13.

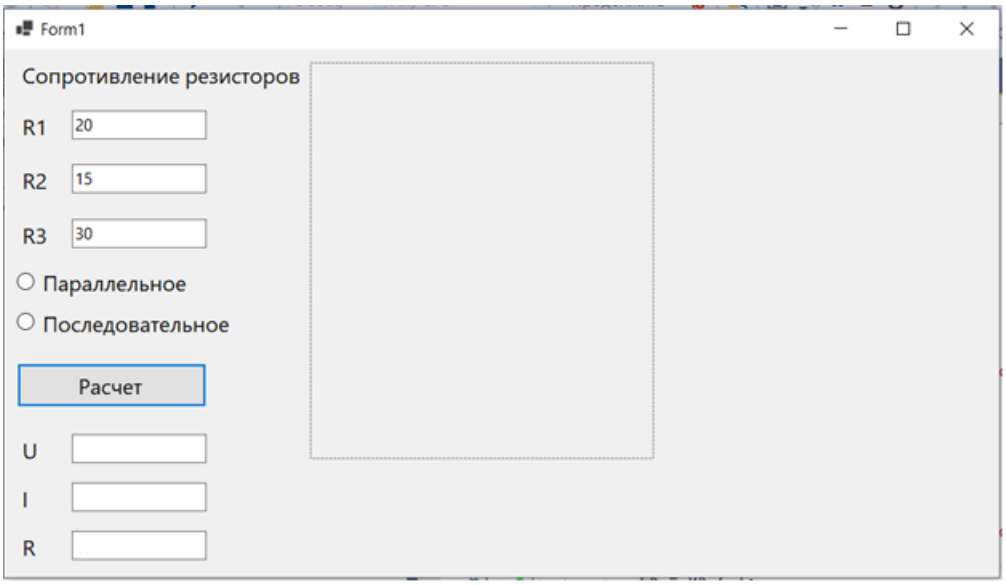


Рисунок 13 – Интерфейс окна программы

Для того чтобы в PictureBox отображалась конкретная картинка, нужно

прописать путь к ней. Если расположить картинку в директории проекта (WinFormsApp1\bin\Debug\netcoreapp3.1\), то вместо полного пути можно указать только название файла:

```
pictureBox1.Image = Image.FromFile("1.jpg");
```

Если картинка отображается в PictureBox не полностью, то необходимо настроить автоматический размер данного элемента, тогда размер будет подстраиваться под содержимое, т.е. под размер картинки (рисунок 14). Для этого нужно открыть свойства элемента pictureBox1 и поменять значение SizeMode на AutoSize.

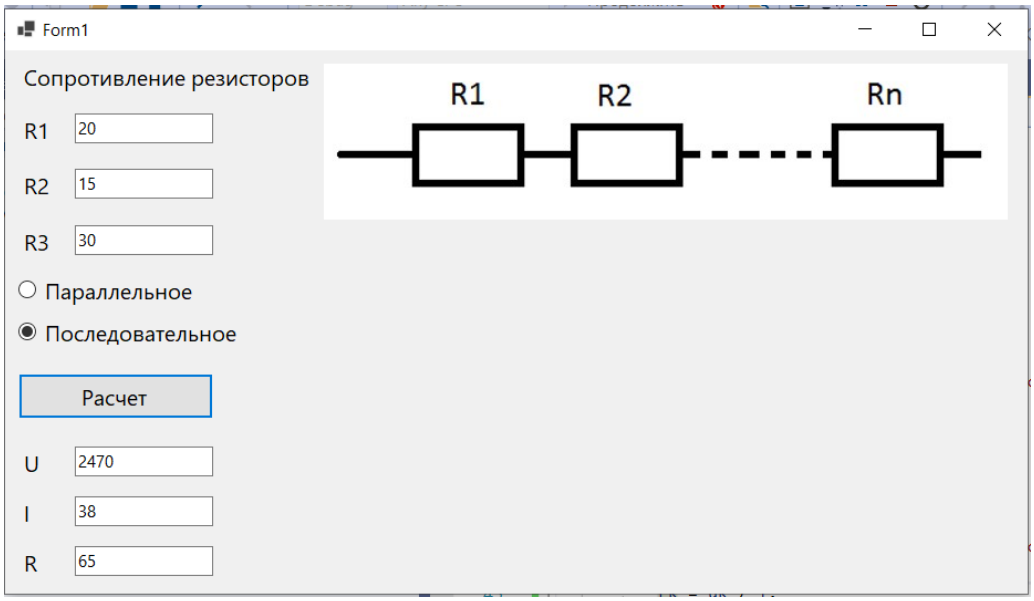


Рисунок 14 – Интерфейс окна программы

Выбор конкретной картинки в соответствии со значением RadioButton:

```
private void button1_Click(object sender, EventArgs e)
  {
     if (radioButton1.Checked == true){
         pictureBox1.Image = Image.FromFile("1.jpg");
         }
     if (radioButton2.Checked == true){
         pictureBox1.Image = Image.FromFile("2.jpg");
         }
  }
```

После того как был настроен вывод картинки на форму, нужно запрограммировать расчеты для каждого типа соединений:

1. последовательное соединение: сила тока I=const, пользователь вводит значение в соответствующее текстовое поле, напряжение вычисляется по

формуле:

$$U = U_1 + U_2 + U_3 = I \cdot (R_1 + R_2 + R_3). \quad (1)$$

2. параллельное соединение: напряжение U=const, пользователь вводит значение в соответствующее текстовое поле, сила тока вычисляется по формуле:

$$I = I_1 + I_2 + I_3 = \frac{U}{R_1} + \frac{U}{R_2} + \frac{U}{R_3}. \quad (2)$$

Сопротивление цепи R для обоих видов соединения вычисляется по формуле:

$$R = \frac{U}{I}. \quad (3)$$

Разработать алгоритм вычисления и вывода на форму характеристик цепи (задание 1 из лабораторной работы 1).

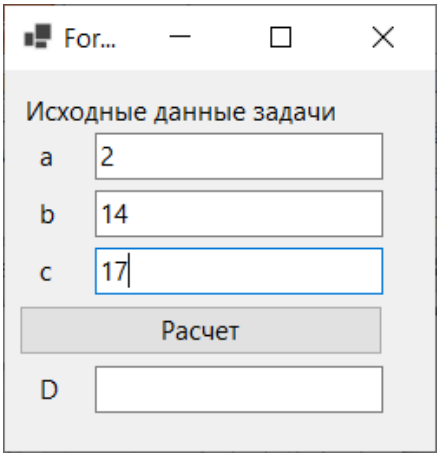
Пример 12. Реализовать программу для вычисления арифметического выражения d = a / b / c при заданных переменных a=2, b=14, c=17.

По условию задачи переменные a, b и c – это целые числа, поэтому объявим переменные типа int и воспользуемся методом Convert.ToInt32:

```
private void button1_Click(object sender, EventArgs e)
  {
     int a = Convert.ToInt32(textBox1.Text);
     int b = Convert.ToInt32(textBox2.Text);
     int c = Convert.ToInt32(textBox3.Text);
     double d = a / b / c;
     textBox4.Text = Convert.ToString(d);
  }
```

Результат запуска программы с заданными переменными a, b и c представлен на рисунке 15.

а)



б)

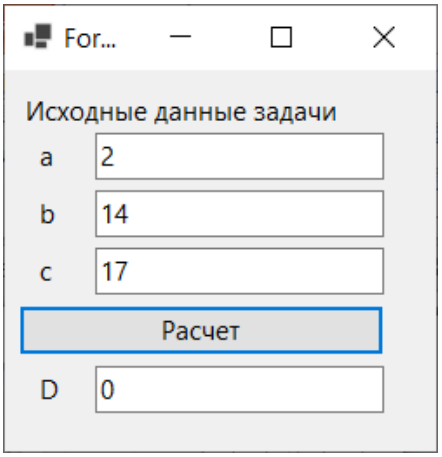


Рисунок 15 – Результат запуска программы с тестовыми данными

Если мы проверим расчеты с помощью калькулятора, то получим 0,008403361, однако в программе получается в ответе 0. Это связано с тем, что оператор деления /, если использовать его с целыми числами, отбрасывает дробную часть. Поэтому в таких случаях следует задавать переменные вещественного типа:

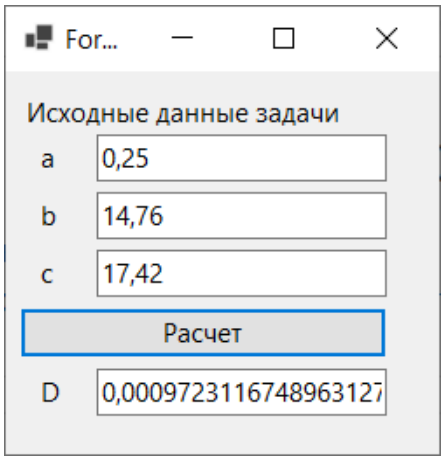
```
double a = Convert.ToDouble(textBox1.Text);
```

После чего можно вводить в текстовые поля значения вещественного типа (рисунок 16а). Для того чтобы округлить число в ответе, нужно воспользоваться функцией Round() библиотеки Math:

```
textBox1.Text = Convert.ToString(Math.Round(x,2));
```

Число 2, в аргументе функции Round(), означает количество знаков после запятой, которые должны остаться в числе (рисунок 16б).

а)



б)

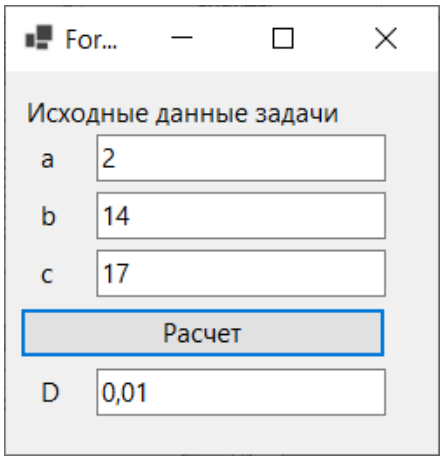


Рисунок 16 – Результат запуска программы с тестовыми данными

FontDialog

Пример 13. С помощью элемента FontDialog реализовать изменение шрифта в элементе TextBox.

Добавим на форму 1 элемент FontDialog, 1 элемент TextBox и 2 элемента Button. Обратите внимание, что элемент FontDialog не отобразился на форме, а переместился в отдельную область в нашем проекте. Это связано с тем, что данный элемент является примером служебных элементов.

Изменим у textBox1 свойство Multiline с false на true, после чего растянем текстовое поле по высоте. Это позволит нам написать объемный текст, который

будет отображаться не в одну строку, а в несколько.

Теперь с помощью метода ShowDialog() обеспечим видимость окна настройки шрифтов для текста. Допустим, пользователь изменил шрифт, нужно сделать так, чтобы текст в элементе textBox1 получил обновленные значения. Для этого в свойство Font элемента textBox1 запишем свойство Font элемента fontDialog1:

```
private void button1_Click(object sender, EventArgs e)
  {
     fontDialog1.ShowDialog();
     textBox1.Font = fontDialog1.Font;
  }
```

На рисунке 17а представлен результат запуска программы до нажатия кнопки «шрифт», на рисунке 17б – после выбора шрифта в диалоговом окне выбора шрифтов (рисунок 18).

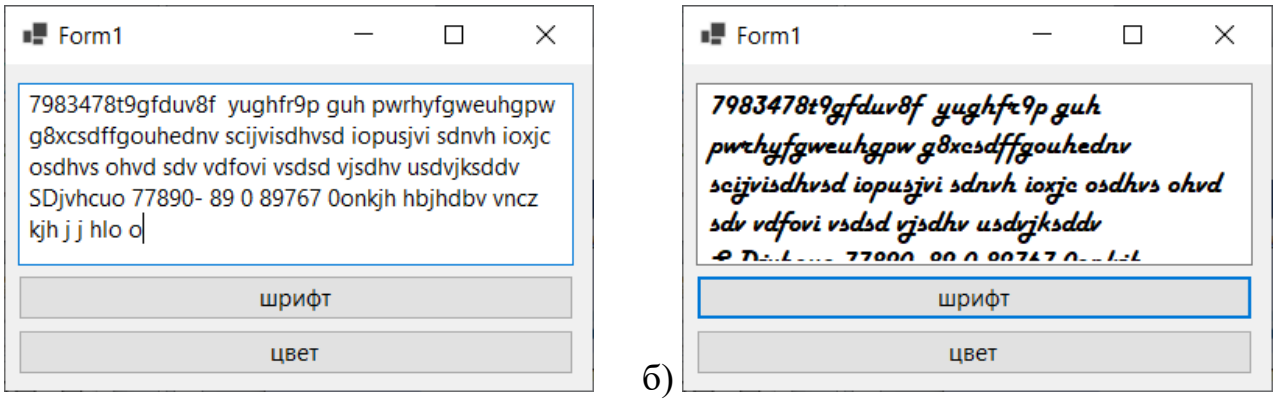


Рисунок 17 – Результат запуска программы

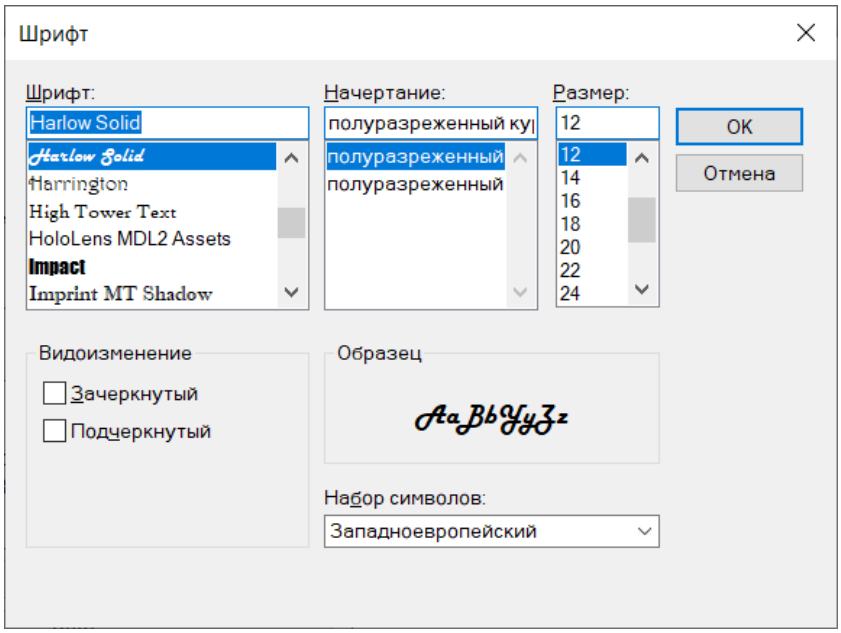
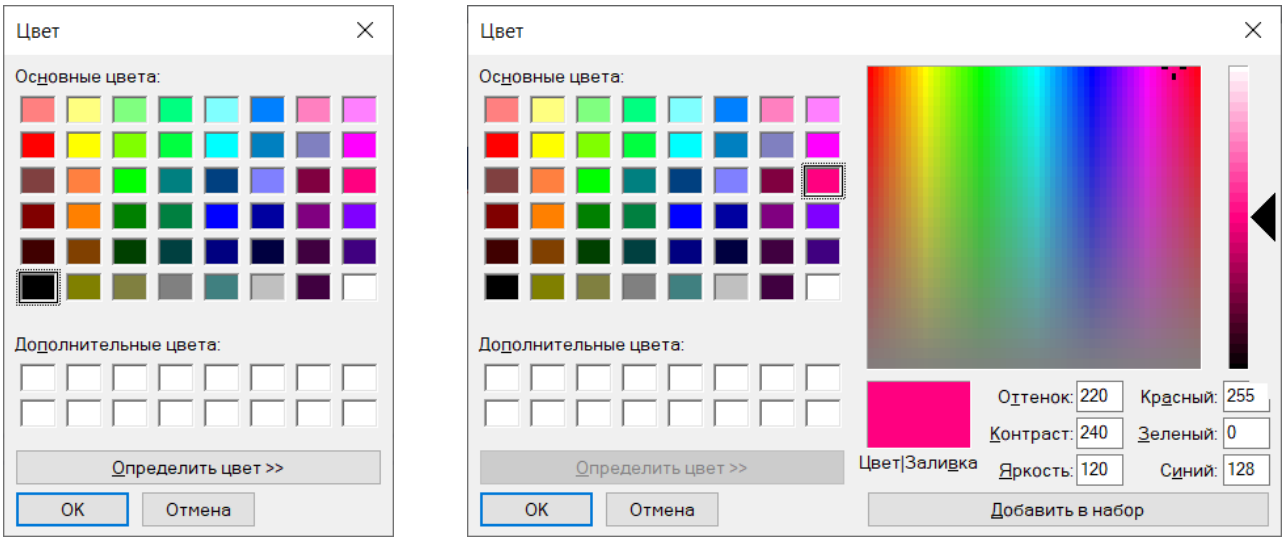


Рисунок 18 – Окно выбора шрифта

ColorDialog

Пример 14. С помощью элемента ColorDialog реализовать изменение шрифта в элементе TextBox из примера 13.

Добавим на форму 1 элемент ColorDialog, заметим, что он, как и FontDialog, является служебным. Диалоговое окно выбора цвета может быть с обычными (рисунок 18а) и расширенными настройками (рисунок 18б).



а) б)

Рисунок 18 – Окно выбора цвета

Чтобы диалоговое окно выбора цвета открылось с дополнительными настройками, нужно присвоить true свойству FullOpen элемента ColorDialog. Теперь с помощью метода ShowDialog() обеспечим видимость окна настройки цвета. Допустим, пользователь изменил цвет, нужно сделать так, чтобы текст в элементе textBox1, а также кнопки button1 и button2 получили обновленные значения. Для этого в свойство BackColor элементов button1 и button2, и в свойство ForeColor элемента textBox1 запишем свойство Color элемента colorDialog1:

```
private void button2_Click(object sender, EventArgs e)
  {
     colorDialog1.FullOpen = true;
     colorDialog1.ShowDialog();
     button1.BackColor = colorDialog1.Color;
     button2.BackColor = colorDialog1.Color;
     textBox1.ForeColor = colorDialog1.Color;
  }
```

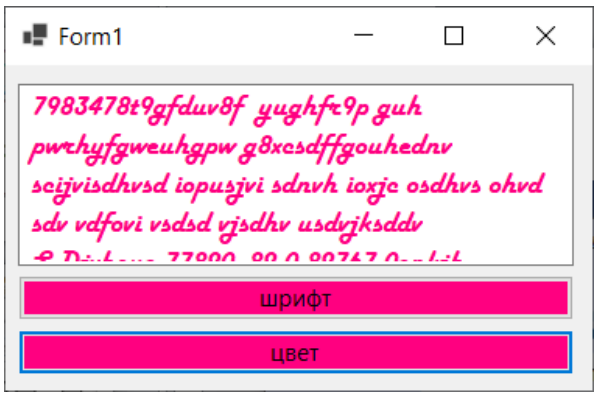


Рисунок 19 – Окно выбора цвета

Timer

Пример 15. С помощью элемента timer реализовать ежесекундный вывод на экран текущего времени (рисунок 20).

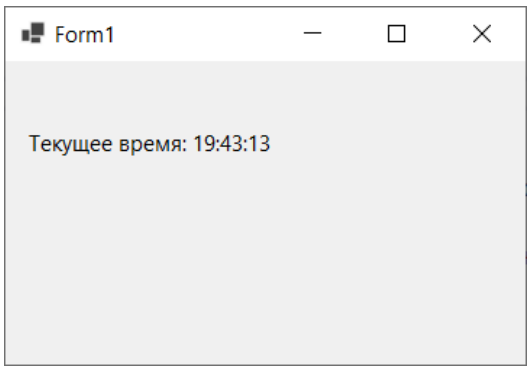


Рисунок 20 – Результат работы программы

У таймера есть только одно событие Tick, которое возникает, когда число в миллисекундах, указанное в свойстве Interval, прошло либо с момента запуска программы, либо с момента предыдущего Tick события. В данном случае, т.к. вывод должен быть ежесекундным, в свойство Interval нужно присвоить значение 1000.

Форматирование – это встраивание в строку различных элементов в заданном формате. Для того чтобы текущее время представить в виде HH:mm:ss, следует использовать выражение DateTime.Now.ToString("HH:mm:ss"). Код в обработчике события будет выглядеть следующим образом:

```
private void timer1_Tick(object sender, EventArgs e)
  {
     label1.Text = string.Format("Текущее время: {0}",
                   DateTime.Now.ToString("HH:mm:ss"));
  }
```

Результат вызова функции DateTime.Now.ToString("HH:mm:ss") будет

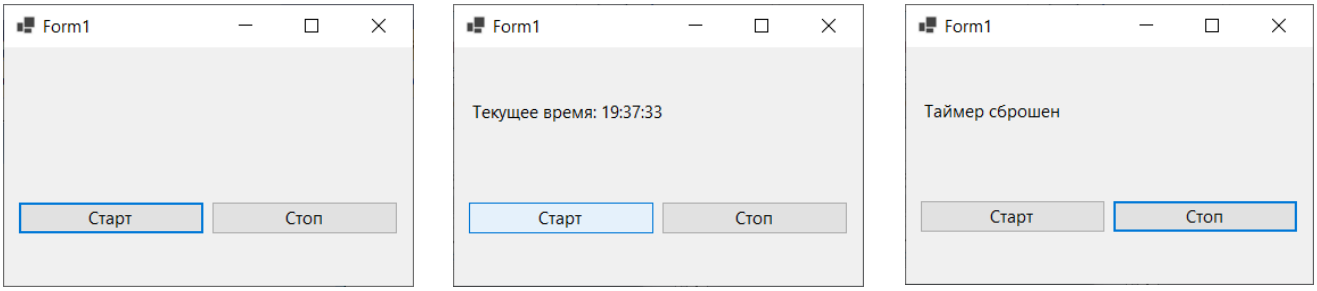
подставлен в строку вместо {0}.

Пример 16. Для предыдущего задания реализовать запуск и остановку часов.

Метод Start(): запускает таймер. Свойство Enabled при значении true указывает, что таймер будет запускаться вместе с запуском формы. Метод Stop(): останавливает таймер. После остановки таймера свойству Enabled нужно присвоить значение false:

```
private void button1_Click(object sender, EventArgs e)
  {
     timer1.Enabled = true;
     timer1.Start();
  }
private void button2_Click(object sender, EventArgs e)
  {
     timer1.Stop();
     timer1.Enabled = false;
     label1.Text = "Таймер сброшен";
  }
```

Результат выполнения программы представлен на рисунке 21. Рисунок 21а – окно до запуска таймера. Рисунок 21б – таймер запущен, время отображается ежесекундно. Рисунок 21в – таймер сброшен.



а) б) в)

Рисунок 21 – Результат работы программы

2 Лабораторная работа «Пользовательский интерфейс в C#»

2.1 Задания

Лабораторная работа состоит из двух заданий.

Задание 1. Для примера 11 запрограммировать расчеты по формулам для параллельного и последовательного соединений. Результаты расчетов вывести в соответствующие текстовые поля.

Задание 2. Реализовать программу для расчета значения выражения. В качестве тестовых данных взять расположенные под формулой значения.

Таблица 1 – Задания

Вариант	Задание
1	$t=\dfrac{2\cos\left(x-\dfrac{\pi}{6}\right)}{0.5+\sin^2 y}\left(1+\dfrac{z^2}{3-z^2/5}\right).$ При $x=14.26,\ y=-1.22,\ z=3.5\times10^{-2}$ $t=0.564849.$
2	$u=\dfrac{\sqrt[3]{8+\lvert x-y\rvert^2+1}}{x^2+y^2+2}-e^{\lvert x-y\rvert}\left(\operatorname{tg}^2 z+1\right)^x.$ При $x=-4.5,\ y=0.75\times10^{-4},\ z=0.845\times10^2$ $u=-55.6848.$
3	$v=\dfrac{1+\sin^2(x+y)}{\left\lvert x-\dfrac{2y}{1+x^2y^2}\right\rvert}x^{\lvert y\rvert}+\cos^2\left(\operatorname{arctg}\dfrac{1}{z}\right).$
4	$w=\lvert\cos x-\cos y\rvert^{\left(1+2\sin^2 y\right)}\left(1+z+\dfrac{z^2}{2}+\dfrac{z^3}{3}+\dfrac{z^4}{4}\right).$ При $x=0.4\times10^4,\ y=-0.875,\ z=-0.475\times10^{-3}$ $w=1.9873.$
5	$\alpha=\ln\left(y^{-\sqrt{\lvert x\rvert}}\right)\left(x-\dfrac{y}{2}\right)+\sin^2\operatorname{arctg}(z).$ При $x=-15.246,\ y=4.642\times10^{-2},\ z=20.001\times10^2$ $\alpha=-182.036.$
6	$\beta=\sqrt{10\left(\sqrt[3]{x}+x^{y+2}\right)}\left(\arcsin^2 z-\lvert x-y\rvert\right).$ При $x=16.55\times10^{-3},\ y=-2.75,\ z=0.15$ $\beta=-38.902.$
7	$\gamma=5\operatorname{arctg}(x)-\dfrac{1}{4}\arccos(x)\dfrac{x+3\lvert x-y\rvert+x^2}{\lvert x-y\rvert z+x^2}.$ При $x=0.1722,\ y=6.33,\ z=3.25\times10^{-4}$ $\gamma=-172.025.$
8	$\varphi=\dfrac{e^{\lvert x-y\rvert}\lvert x-y\rvert^{x+y}}{\operatorname{arctg}(x)+\operatorname{arctg}(z)}+\sqrt[3]{x^6+\ln^2 y}.$ При $x=-2.235\times10^{-2},\ y=2.23,\ z=15.221$ $\varphi=39.374.$

9	$\psi = \left\lvert x^{\frac{y}{x}} - \sqrt[3]{\left\lvert \frac{y}{x} \right\rvert} \right\rvert + (y-x)\frac{\cos y - \frac{z}{(y-x)}}{1+(y-x)^2}.$ При $x = 1.825\times10^{2}$, $y = 18.225$, $z = -3.298\times10^{-2}$ $\psi = 1.2131$.
10	$a = 2^{-x}\sqrt{x + \sqrt[4]{\lvert y \rvert}}\sqrt[3]{e^{x-1/\sin z}}.$ При $x = 3.981\times10^{-2}$, $y = -1.625\times10^{3}$, $z = 0.512$ $a = 1.26185$.
11	$b = y^{\sqrt[3]{\lvert x \rvert}} + \cos^3(y)\frac{\lvert x-y \rvert\left(1+\frac{\sin^2 z}{\sqrt{x+y}}\right)}{e^{\lvert x-y \rvert} + \frac{x}{2}}.$ При $x = 6.251$, $y = 0.827$, $z = 25.001$ $b = 0.7121$.
12	$c = 2^{(y^x)} + (3^x)^y - \frac{y\left(\operatorname{arctg} z - \frac{\pi}{6}\right)}{\lvert x \rvert + \frac{1}{y^2+1}}.$ При $x = 3.251$, $y = 0.325$, $z = 0.466\times10^{-4}$ $c = 4.025$.
13	$f = \frac{\sqrt[4]{y + \sqrt[3]{x-1}}}{\lvert x-y \rvert(\sin^2 z + \operatorname{tg} z)}.$ При $x = 17.421$, $y = 10.365\times10^{-3}$, $z = 0.828\times10^{5}$ $f = 0.33056$.
14	$g = \frac{y^{x+1}}{\sqrt[3]{\lvert y-2 \rvert} + 3} + \frac{x + \frac{y}{2}}{2\lvert x+y \rvert}(x+1)^{-1/\sin z}.$ При $x = 12.3\times10^{-1}$, $y = 15.4$, $z = 0.252\times10^{3}$ $g = 82.8257$.
15	$h = \frac{x^{y+1} + e^{y-1}}{1 + x\lvert y - \operatorname{tg} z \rvert}(1 + \lvert y-x \rvert) + \frac{\lvert y-x \rvert^2}{2} - \frac{\lvert y-x \rvert^3}{3}.$ При $x = 2.444$, $y = 0.869\times10^{-2}$, $z = -0.13\times10^{3}$ $h = -0.49871$.

2.2 Требования к отчету по лабораторной работе

В отчете по лабораторной работе «Пользовательский интерфейс в C#» необходимо для каждого задания представить:

1. отформатированный код, сопровождаемый разумным количеством комментариев (комментировать каждую строку не требуется);

2. результат запуска программы (снимок окна приложения с результатом, выведенным на экран после нажатия кнопки).

Отчет по лабораторной работе оформляется в соответствии с требованиями, принятыми в КГПИ ФГБОУ ВО «КемГУ», предоставляется преподавателю в электронном виде.

2.3 Контрольные вопросы

1. С помощью какого элемента можно обозначить для пользователя раздел ввода / вывода данных на форме, обозначить переменные?
2. Какое событие вызывается при загрузке формы пользователем?
3. Как можно создать обработчик события «Нажатие на кнопку»?
4. Какие типы элементов управления вы знаете?
5. Приведите пример интерфейсных элементов управления.
6. Приведите пример служебных элементов управления.
7. Что произойдет, если щелкнуть левой кнопкой мыши по кнопке?
8. Как называется свойство, с помощью которого можно заблокировать текстовое поле от ввода в него данных пользователем?
9. С помощью какого элемента можно настроить цвет кнопок или текстовых полей?
10. Как можно получить длину введенной пользователем строки?
11. Как можно обратиться к i-му элементу строки?
12. Как в переменную x вещественного типа считать данные из текстового поля?
13. С помощью какого элемента можно предоставить изменить шрифт текста в текстовом поле?
14. В какой элемент данные добавляются с помощью метода Add()?
15. Какое событие вызывается при наведении указателя мыши на кнопку?

3 Образец тестовых заданий

1. Для того чтобы вывести значение переменной str (строкового типа) в listBox, нужно использовать

а) listBox1.Items.Add=str;

б) listBox1.Items=str;

в) listBox1.Items[i]=str;

г) listBox1.Items.Add(str).

2. Для того чтобы какие-либо действия выполнились после нажатия на кнопку, их нужно поместить в процедуру

```
private void button1_Click(object sender, EventArgs e)
    {
    }
```

а) верно;

б) неверно.

3. Для того чтобы изменить текст, расположенный на кнопке, необходимо написать новый текст в свойстве text

а) верно;

б) неверно.

4. Строковый тип данных объявляется служебным словом:

а) string;

б) char;

в) word;

г) text.

5. Тип данных определяет …

а) множество значений, которые могут принимать объекты программы, а также совокупность операций, допустимых над этими значениями;

б) множество значений, которые могут принимать объекты программы;

в) совокупность операций, допустимых над этими значениями;

г) ничего из этого.

6. Целочисленные типы, символьный, логический и пользовательские

типы данных образуют группу:

а) порядковых типов;

б) непорядочных типов;

в) произвольных типов;

г) упорядоченных типов.

7. Операторы, которые не содержат внутри себя других операторов, называются …

а) составными;

б) пустыми;

в) простыми;

г) сложными.

8. В выражении summa = Math.Sqrt(x)+3*a переменными являются

а) Sqrt, x, a;

б) summa, Sqrt, x, a;

в) a, x, summa;

г) только summa.

9. Проверить, нажат ли флажок checkBox, можно с помощью условия

а) if (checkBox1.Checked == true);

б) if (checkBox1.CheckedBox == true);

в) if (checkBox1.CheckedText == true);

г) if (checkBox1.Checked = true).

10. Какому свойству текстового поля необходимо присвоить значение true, чтобы оно поддерживало многострочный текст?

а) Multiline;

б) Multirows;

в) Multistring;

г) Multicheck.

11. В переменную строкового типа передали следующее: string str=" Индексация массива начинается с нуля. ". Что будет выведено в текстовое поле, если команда выглядит следующим образом:

textBox1.Text=Convert.ToString(str[8])?

а) а;

б) ц;

в) с;

г) и.

12. В переменную строкового типа передали следующее: string str=" Индексация массива начинается с нуля. ". Команда выглядит следующим образом: textBox1.Text=Convert.ToString(str[8]). Почему в данном случае применяется преобразование типов?

а) потому что у str[8] тип char, а у свойства textBox1.Text тип string;

б) потому что всегда для вывода в текстовое поле необходимо выполнить преобразование типов;

в) потому что у str[8] тип int, а у свойства textBox1.Text тип string;

г) нет верного варианта.

13. В программе реализована процедура public void Painting(){…}, окрашивающая кнопки в заданные цвета. Как можно вызвать выполнение данной процедуры при нажатии на кнопку button1?

а)
```
private void button1_Click(object sender, EventArgs e)
{
   Painting();
};
```
б)
```
private void button1_Click(object sender, EventArgs e)
{
   x=Painting();
};
```
в)
```
private void button1_Click(object sender, EventArgs e)
{
   int x=Painting();
};
```
г)
```
private void button1_Click(object sender, EventArgs e)
{
   x.Painting();
}.
```

14. Элемент таймер относится к служебным элементам

а) верно;

б) неверно.

15. Чтобы поместить элемент на форму, его можно просто перетащить туда мышкой

а) верно;

б) неверно.

16. Если нужно, чтобы checkBox1 появился на форме сразу во включенном состоянии, нужно в коде программы записать инструкцию

а) `checkBox1.Checked = true;`

б) `checkBox1.Checked == true;`

в) `checkBox1.Checked = false;`

г) `checkBox1.Checked == false.`

17. Содержимое listBox можно очистить с помощью метода

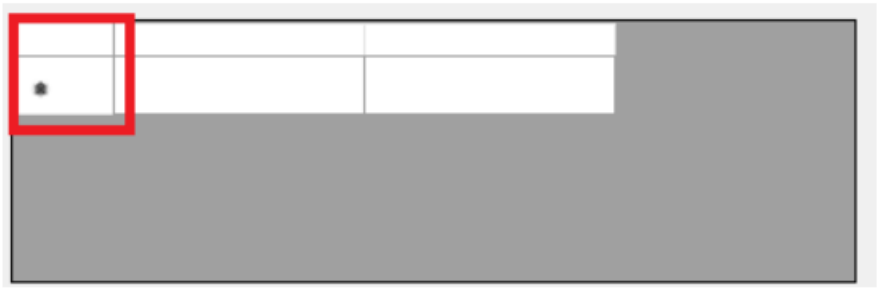
а) `Clear();`

б) `Delete();`

в) `Drop();`

г) нет верного ответа.

18. Для того чтобы убрать отображение заголовков строк (область, выделенную красным цветом),



нужно чтобы было выполнено:

а) `RowHeadersVisible=False;`

б) `ColumnHeadersVisible=False;`

в) `RowHeadersVisible=True;`

г) `ColumnHeadersVisible=True.`

4 Индивидуальные задания

1. Разместить на форме кнопки в соответствии с рисунком 22.

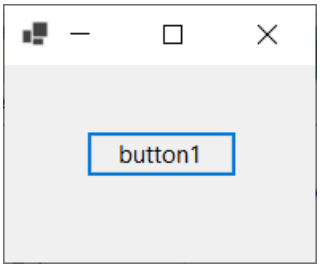


Рисунок 22 – Размещение элемента button на форме программы

Задать программно цвет кнопки в соответствии с вариантом (Цвет 1). Реализовать смену цвета на Цвет 2 после нажатия кнопки. Реализовать обратный переход к Цвету 1 при следующем нажатии на кнопку.

№ варианта	Цвет 1	Цвет 2
1	Красный (Red)	Желтый (Yellow)
2	Зеленый (Green)	Синий (Blue)
3	Черный (Black)	Белый (White)
4	Оранжевый (Orange)	Оливковый (Olive)
5	Золотой (Gold)	Серый (Gray)
6	Розовый (Pink)	Фиолетовый (BlueViolet)
7	Коричневый (Chocolate)	Темно-коричневый (Brown)
8	Салатовый (GreenYellow)	Коричневый (Chocolate)
9	Желтый (Yellow)	Розовый (Pink)
10	Синий (Blue)	Золотой (Gold)
11	Белый (White)	Оранжевый (Orange)
12	Оливковый (Olive)	Черный (Black)
13	Серый (Gray)	Зеленый (Green)
14	Фиолетовый (BlueViolet)	Красный (Red)
15	Темно-коричневый (Brown)	Салатовый (GreenYellow)

2. Разместить на форме элементы в соответствии с рисунком 23: 6 элементов radioButton, один элемент pictureBox и один элемент button.

Изменить программно заголовок формы (свойство text у формы), текст на кнопке и текст у каждого элемента radioButton (как показано на рисунке).

Реализовать после выбора элемента radioButton и нажатия на кнопку загрузку соответствующего изображения в элемент pictureBox.

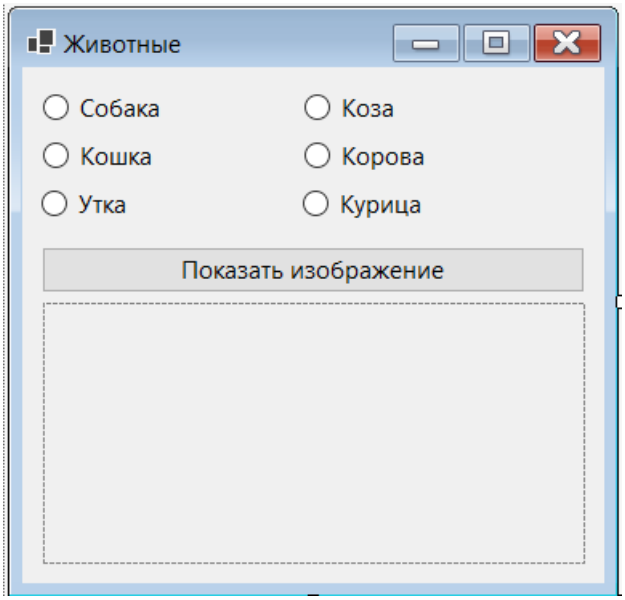


Рисунок 23 – Размещение элементов на форме программы

3. Разместить на форме элементы в соответствии с рисунком 24. В элемент textBox1 поместить текст:

Платформа Windows Forms предоставляет разнообразные элементы управления, которые условно можно разделить на два типа: интерфейсные – те, которые видны пользователю и с которыми он может работать непосредственно (кнопки, панели, таблицы) и служебные – те, что выполняют определенные задачи и вызываются путем взаимодействия пользователя с интерфейсными элементами (диалоги, таймеры, адаптеры).

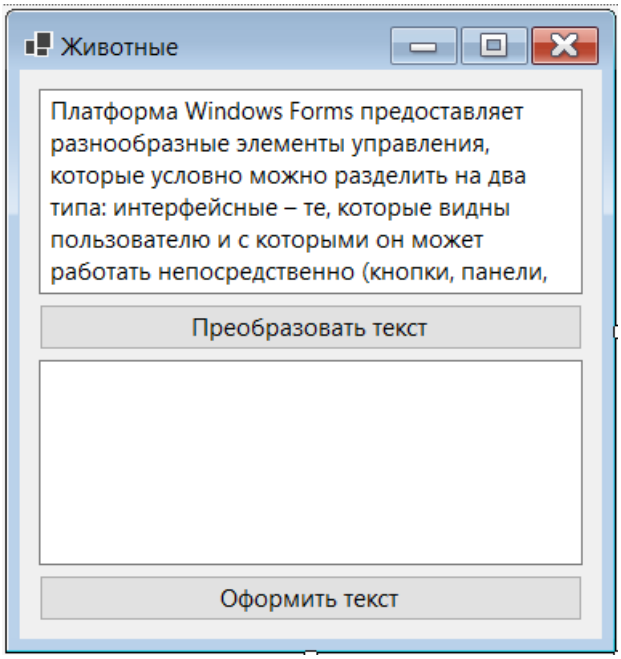


Рисунок 24 – Размещение элементов на форме программы

Реализовать по нажатию кнопки button1 считывание строки из textBox1, ее преобразование в соответствии с вариантом и вывод в textBox2.

Реализовать по нажатию кнопки button2 оформление цвета и шрифта текста (с помощью элементов ColorDialog и FontDialog).

№ варианта	Задание
1	Каждый третий символ строки заменить восклицательным знаком, а каждый четвертый – точкой с запятой
2	Каждый второй символ строки заменить вопросительным знаком, а каждый третий – процентом
3	Каждый третий символ строки заменить пробелом, а каждый пятый – буквой А
4	Каждый второй символ строки заменить плюсом, а каждый третий – решеткой
5	Каждый третий символ строки заменить собакой, а каждый четвертый – нижним подчеркиванием
6	Каждый второй символ строки заменить звездочкой, а каждый третий – значком доллара
7	Каждый третий символ строки заменить открывающейся скобкой, а каждый пятый – буквой В
8	Каждый второй символ строки закрывающейся скобкой, а каждый третий – амперсандом
9	Каждый третий символ строки заменить знаком равно, а каждый четвертый – минусом
10	Каждый второй символ строки заменить знаком <, а каждый третий – двойной кавычкой
11	Каждый третий символ строки заменить знаком >, а каждый пятый – буквой Ж
12	Каждый второй символ строки заменить знаком \, а каждый третий – буквой С
13	Каждый третий символ строки заменить знаком /, а каждый четвертый – запятой
14	Каждый второй символ строки заменить точкой, а каждый третий – многоточием

5 Рекомендуемая литература

Основная учебная литература

1. Подбельский В.В. Программирование. Базовый курс C# : учебник для вузов / В.В. Подбельский – Москва : Изд-во Юрайт, 2023. – 369 с. – ISBN 978-5-534-10616-9. - URL: https://urait.ru/viewer/programmirovanie-bazovyy-kurs-s-511747#page/2. - (дата обращения: 12.12.2022). – Текст : электронный.

Дополнительная литература

1. Казанский, А.А. Программирование на Visual C#: учебное пособие для вузов/ А.А. Казанский. – Москва: Изд-во Юрайт, 2023. – 192 с. - ISBN 978-5-534-12338-8. – URL: https://urait.ru/viewer/programmirovanie-na-visual-c-512404#page/1. - (дата обращения: 01.12.2022). – Текст : электронный.

2. Предварительная спецификация C# 6.0 : сайт. – 2020 - . - URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/language-specification/introduction (дата обращения: 01.12.2022). – Текст : электронный.

3. Visual Studio : сайт. – 2020 - . - URL: https://visualstudio.microsoft.com/ru/ (дата обращения: 01.12.2022). – Текст : электронный.

4. IntelliJ IDEA : сайт. – 2000 - . - URL: https://www.jetbrains.com/idea/ (дата обращения: 01.12.2022). – Текст : электронный.

Литература для оформления отчета

1. Правила оформления учебных работ студентов : учебно-методическое пособие / И.А. Жибинова, А.Е. Аракелян, О.В. Соколова, Ю.Н. Соина-Кутищева. – Новокузнецк : НФИ КемГУ, 2018. – 124 с. – Текст : непосредственный.

2. ГОСТ 19.701-90 (ИСО 5807-85) Единая система программной документации (ЕСПД). Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения : межгосударственный стандарт : издание официальное : введен впервые : дата введения 1992-01-01 / Москва Стандартинформ, 2010 – 158 с. – Текст: непосредственный.

6 Современные профессиональные базы данных и справочные системы

CITForum.ru : on-line библиотека свободно доступных материалов по информационным технологиям на русском языке : сайт. – 2001 – URL: http://citforum.ru (дата обращения: 01.12.2022). – Текст: электронный.

eLIBRARY.RU : научная электронная библиотека : сайт. – Москва, 2000 - . – URL: http://www.elibrary.ru (дата обращения: 01.12.2022). – Режим доступа: для зарегистрир. пользлвателей. – Текст: электронный.

Единое окно доступа к образовательным ресурсам : сайт. – Москва, 2005 - . – URL: http://window.edu.ru/ (дата обращения: 01.12.2022). –Текст: электронный.