

МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное учреждение высшего
образования «Национальный исследовательский университет
Московский институт электронной техники»

Институт Микроприборов и систем управления

НАПРАВЛЕНИЕ: 27.03.04 «Управление в технических системах»

ДИСЦИПЛИНА: Объектно-ориентированное программирование

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

«Разработка приложения на языке программирования C#»

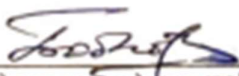
Донченко 05.12.22 Бобков
Вомышев 05.12.22 Бобков
Защипин 05.12.22 Бобков

Работу выполнил
студент гр. УТС-21


(подпись студента)

Р. В. Закшевский
(Ф.И.О. студента)

Преподаватель


(подпись преподавателя)

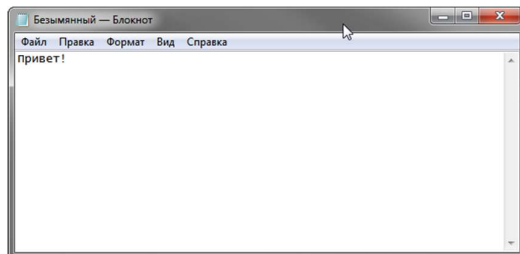
В. Д. Бобков
(Ф.И.О. преподавателя)



Москва, 2022 г.

Цель работы: Научиться создавать приложения в среде разработки Visual Studio на языке программирования C#.

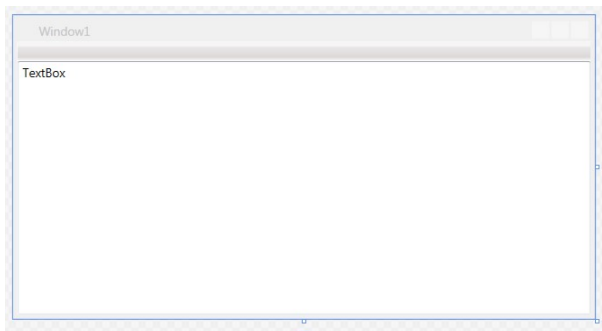
Теоретическая часть



Программа блокнот представляет собой простейший редактор, позволяющий создавать и изменять текстовые файлы. Попробуем создать аналогичную программу самостоятельно.

Создание основного окна

В VisualStudio создаем новый проект WPF Application и на основную форму приложения добавляем элементы TextBox и Menu. Располагаем их в соответствии с рисунком:



Для того, чтобы элементы правильно изменяли свои размеры при масштабировании окна приложения нужно правильно установить привязки к краям окна:

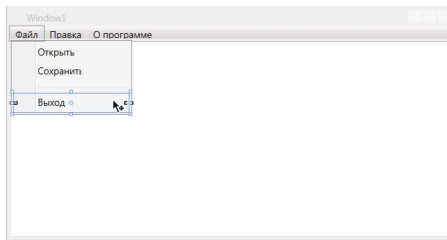


Для TextBox привязки должны быть закрыты со всех четырех сторон, а для Menu – для всех кроме нижнего края.

После этого добавляем к меню новые пункты (правый клик → Add MenuItem):

- Файл
 - Открыть
 - Сохранить
 - Выход
- Правка
 - Отменить
 - Повторить
- О программе

После этого окно программы должно выглядеть следующим образом:



Затем всем элементам, включая пункты меню, задаем осмысленные названия, чтобы затем использовать их в коде программы, а для текстового окна включаем свойство `AcceptsReturn`, для возможности переноса строк по клавише Enter.

Открытие и сохранение файлов

Для того чтобы пользователь мог указать файл для открытия и имя файла для сохранения используются диалоги открытия и сохранения. Windows предоставляет стандартные диалоги, которые мы можем использовать в своих программах. Для этого в начале программы необходимо указать директиву `using Microsoft.Win32`; разрешающую использование стандартных диалогов.

Там же указываем директиву `using System.IO`; необходимую для использования класса `File`, который используется для работы с файлами.

После этого дважды кликаем по пункту меню открыть и в созданном методе пишем следующее:

```
private void menuOpen_Click(object sender, RoutedEventArgs e)
{
    // создаем объект диалога открытия
    var dialog = new OpenFileDialog();
    // устанавливаем фильтр файлов
    dialog.Filter = "Текстовые файлы | *.txt";
    // показываем диалог
    var result = dialog.ShowDialog();

    //если диалог закрыли кнопкой "Открыть",
    // то result будет равен true,
    //если закрыли крестиком или кнопкой "Отмена"
    // то result будет равен false
    if (result == true)
    {
        //Метод ReadAllText - открывает текстовый файл
        //и возвращает его содержимое
        textBox.Text = File.ReadAllText(dialog.FileName);
    }
}
```

И аналогично для команды сохранения файла:

```
private void menuSave_Click(object sender, RoutedEventArgs e)
{
    // создаем объект диалога сохранения
    var dialog = new SaveFileDialog();
    // устанавливаем фильтр файлов
    dialog.Filter = "Текстовые файлы | *.txt";
    // показываем диалог
    var result = dialog.ShowDialog();

    //если диалог закрыли кнопкой "Сохранить",
    // то result будет равен true,
    //если закрыли крестиком или кнопкой "Отмена"
    // то result будет равен false
    if (result == true)
    {
        // Метод WriteAllText - записывает указанный текст
        // в файл с выбранным в диалоге именем
        File.WriteAllText(dialog.FileName, textBox.Text);
    }
}
```

```

    }
}

```

Так же добавляем обработчики пунктов меню Отменить и Повторить, код для них выглядит следующим образом:

```

private void MenuUndo_Click(object sender, RoutedEventArgs e)
{
    textBox.Undo();
}

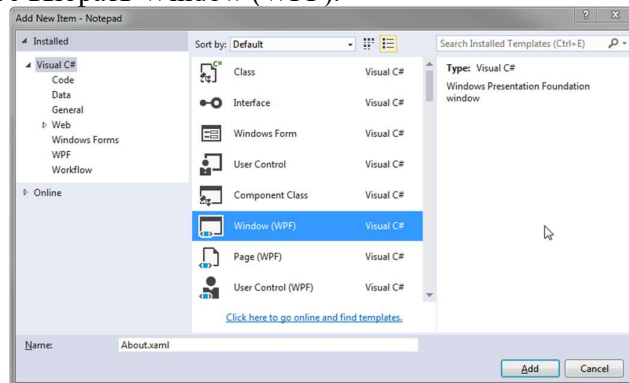
private void MenuRedo_Click(object sender, RoutedEventArgs e)
{
    textBox.Redo();
}

```

В них мы вызываем встроенные в класс TextBox методы Undo и Redo, которые позволяют отменить или повторить изменения текста.

Добавление окна «О программе»

Для добавления к программе нового окна нужно выбрать Project->Add Window и в открывшемся диалоге выбрать Window (WPF).



После этого откроется редактор форм в котором будет открыто свежесозданное окно. Оформите его по своему вкусу, обязательно укажите имена авторов, группу и год создания.

Затем, для того, чтобы это окно появлялось по кнопке меню, возвращаемся к главной форме и дважды кликаем по пункту меню «О программе». В открывшемся редакторе кода пишем следующее:

```

private void MenuAbout_Click(object sender, RoutedEventArgs e)
{
    // создаем новое окно
    var about = new About();

    // и показываем его на экране
    about.Show();
}

```

Задание

После выполнения шагов выше все базовые функции текстового редактора у вас реализованы. Необходимо добавить еще несколько штрихов:

1. Текущий пункт меню «Сохранить» переименовать в «Сохранить как». Добавить новый пункт меню «Сохранить», который будет сохранять файл без диалога, если файл уже выбран пользователем.
2. Добавить пункт меню «Создать», который будет очищать текущий текст.
3. Добавить предупреждение если файл был изменен после последней операции сохранения перед открытием, созданием нового или закрытием программы. Используйте событие TextChanged элемента TextBox для того, чтобы узнать об изменении текста.
4. Добавьте элемент StatusBar в который выводите количество слов в текущем файле.

Код программы:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Win32;
using System.IO;

namespace Lab4_Zakshevskij_UTS_22
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }
        private void menuOpen_Click(object sender, RoutedEventArgs e)
        {
            var dialog = new OpenFileDialog();
            dialog.Filter = "Текстовые файлы | *.txt";
            var result = dialog.ShowDialog();
            if (result == true)
            {
                Tb.Text = File.ReadAllText(dialog.FileName);
            }
        }
        public bool checkSave = false;
        public string justDialogSave;
        public void menuSave_Click(object sender, RoutedEventArgs e)
        {
            var dialog = new SaveFileDialog();
            dialog.Filter = "Текстовые файлы | *.txt";
            var result = dialog.ShowDialog();
            if (result == true)
            {
                checkSave = true;
                justDialogSave = dialog.FileName;
                File.WriteAllText(dialog.FileName, Tb.Text);
            }
        }
        private void menuJustSave_Click(object sender, RoutedEventArgs e)
        {

```

```

        if (checkSave)
        {
            File.WriteAllText(justDialogSave, Tb.Text);
        }
    }
    private void MenuUndo_Click(object sender, RoutedEventArgs e)
    {
        Tb.Undo();
    }

    private void MenuRedo_Click(object sender, RoutedEventArgs e)
    {
        Tb.Redo();
    }
    private void MenuAbout_Click(object sender, RoutedEventArgs e)
    {
        var about = new About();
        about.Show();
    }

    private void menuCreate_Click(object sender, RoutedEventArgs e)
    {
        Tb.Text = "";
    }
    private void TbEditor_SelectionChanged(object sender, TextChangedEventArgs
e)
    {
        string text = Tb.Text;
        if (text != "")
        {
            text = text.Trim(new char[] { ',', '.' });
            string[] textArray = text.Split(new char[] { ' ' });
            WordCheck.Text = "Количество слов: " + (textArray.Length-
1).ToString();
        }
        else
        {
            WordCheck.Text = "Количество слов: 0";
        }
    }
}

```

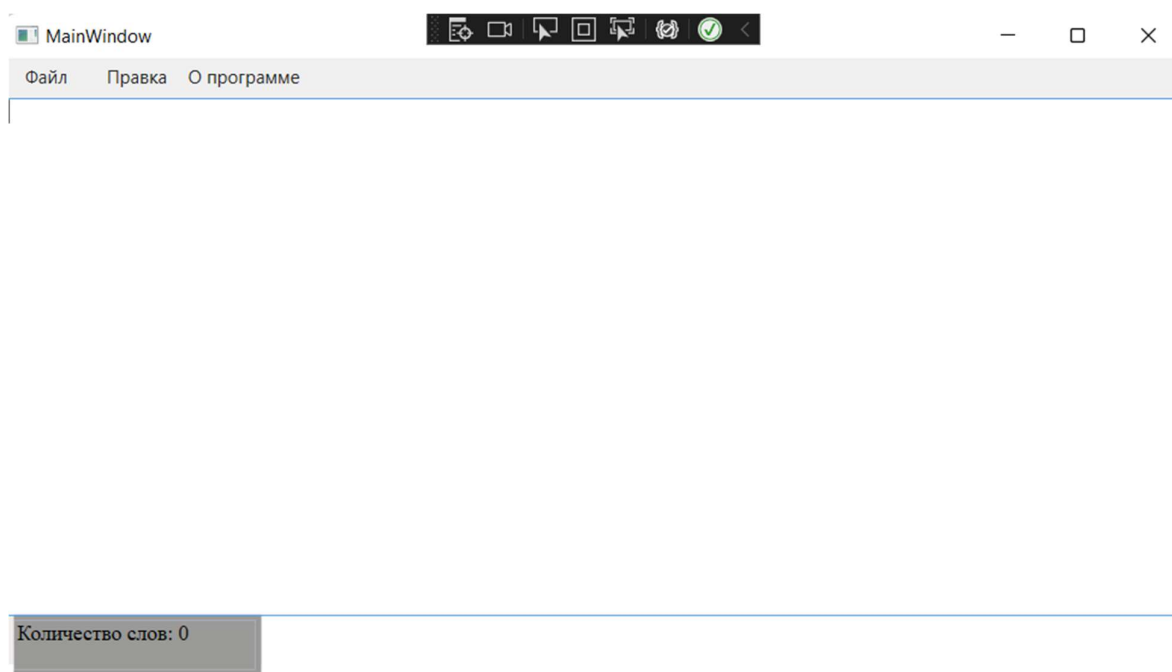


Рисунок 1. Внешний вид программы

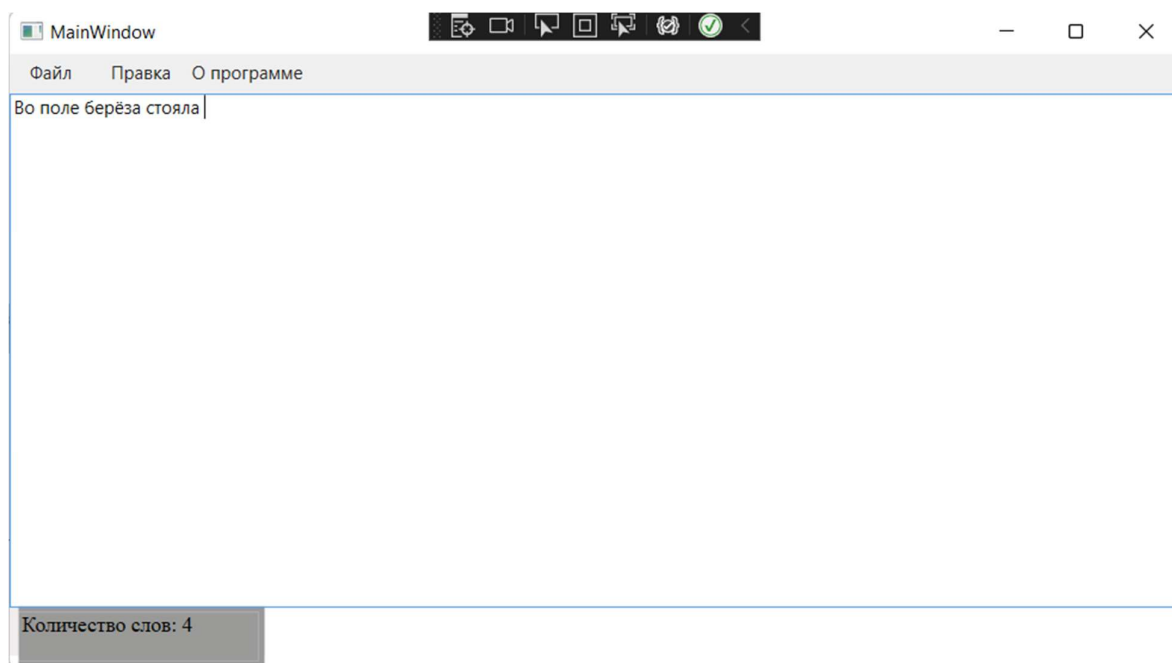


Рисунок 2. Пример работы программы

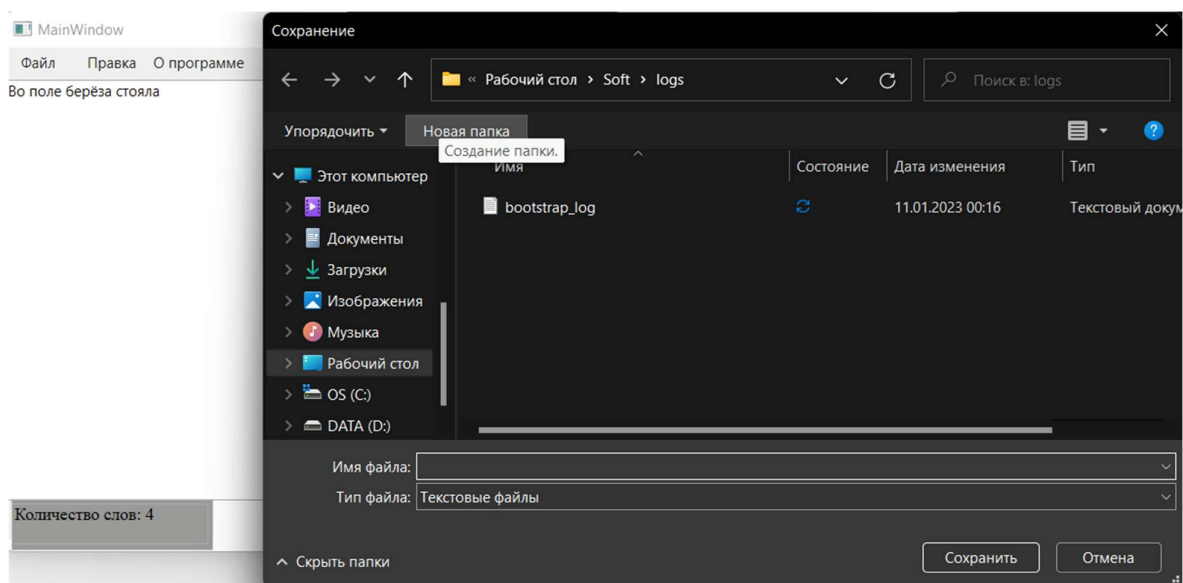


Рисунок 3. Пример взаимодействия с программой

Вывод: в данной работе мы научились создавать приложения в среде разработки Visual Studio на языке программирования C#. Создали игру “Саймон говорит”.