Лабораторная работа №6

ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ МЕХАТРОНИКИ ТРАНСПОРТНЫХ СРЕДСТВ

Кафедра информатики

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по проведению практических работ по дисциплине «Программирование» для студентов специальности 6.050201 “Системная инженерия”

Разработчик - доцент кафедры информатики
кандидат технических наук,
старший научный сотрудник
Тимонин Владимир Алексеевич

Харків 2012

Лабораторная работа №6
Исследование возможностей интегрированной среды разработки Visual C# для создания приложений по обработке событий клавиатуры.

Цель работы – исследовать возможности интегрированной среды разработки Visual Studio 2010 и получить практические навыки по созданию приложений по обработке событий клавиатуры.

1. Теоретические сведения

Все действия пользователя при взаимодействии с приложением сводятся к перемещению мыши, нажатию кнопок мыши и нажатию клавиш клавиатуры.

1.1. События клавиатуры

События клавиатуры (клавишные события) создаются в момент нажатия или отпускания её клавиш. В оконных компонентах Visual C# определены три события, связанные с клавиатурой:

- KeyDown - событие наступает при нажатии пользователем любой клавиши. Можно распознать нажатые клавиши, включая функциональные, и кнопки мыши, но нельзя распознать символ нажатой клавиши.

- KeyPress - событие наступает при нажатии пользователем клавиши символа. Можно распознать только нажатую клавишу символа, различить символ в верхнем и нижнем регистре, различить символы кириллицы и латинские, но нельзя распознать функциональные клавиши и кнопки.

С событием **KeyPress** тесно связаны события **KeyDown** и **KeyUp**. Событие **KeyDown** предшествует каждому событию **KeyPress** нажатия клавиши, а событие **KeyUp** происходит, когда клавишу отпускают. Если удерживать клавишу в нажатом состоянии, повторяющиеся события **KeyDown** и **KeyPress** происходят при каждом повторении знака. Одно событие **KeyUp** создается при отпускании клавиши.

- KeyUp - событие наступает при отпускании пользователем любой клавиши. Можно распознать нажатые клавиши, включая функциональные, и кнопки мыши, но нельзя распознать символ отпускаемой клавиши.

Чтобы обрабатывать события клавиатуры только на уровне формы без предоставления другим элементам управления возможности получать события клавиатуры, необходимо задать для свойства **KeyPressEventArgs.Handled** в методе обработки события **KeyPress** формы значение **true**. Некоторые клавиши, такие как <Tab>, <Enter>, <Esc> и клавиши со стрелками, автоматически обрабатываются элементами управления. Чтобы эти клавиши вызывали событие **KeyDown**, необходимо переопределить метод **IsInputKey** в каждом из элементов управления, имеющихся в форме. Код для переопределения метода **IsInputKey** позволяет выяснить, нажата ли какая-нибудь специальная клавиша, и вернуть значение **true**.

1.2. Распознавание нажатых клавиш

Заголовок обработчика события **KeyDown** может иметь, например, следующий вид:

```
private void Form1_KeyDown(object sender, KeyEventArgs e)
    {
    }
```

Объект **KeyEventArgs** предоставляет данные для событий **KeyDown** или **KeyUp**.

Объект типа **KeyEventArgs** имеет следующие свойства:

- **Alt** – содержит значение, показывающее, была ли нажата клавиша <Alt>. Имеет тип **bool**. Значение **true**, если клавиша <Alt> была нажата, в противном случае — значение **false**.

- **Control** - содержит значение, показывающее, была ли нажата клавиша <Ctrl>. Имеет тип **bool**.

Значение **true**, если клавиша <Ctrl> была нажата, в противном случае — значение **false**.

- **KeyCode** - содержит код клавиатуры. Имеет тип **Keys**, определенный в пространстве имен System.Windows.Forms как:

public Keys KeyCode

Значениями перечисления **Keys** являются, например:

Back - клавиша <Backspase>;
Tab - клавиша <Tab>;
Enter - клавиша <Enter>;
ShiftKey - клавиша <Shift>;
ControlKey - клавиша <Ctrl>;
Menu - клавиша <Alt>;
CapsLock - клавиша <CapsLock>;
Escape - клавиша <Esc>;
Space - клавиша <Пробел>;
Insert - клавиша <Ins>;
D0 ? D9 - клавиша <0> ? клавиша <9>;
A ? Z - клавиша <A> ? клавиша <Z>;
NumPad0 ? NumPad9 - клавиша <0> на цифровой клавиатуре ? клавиша <9> на цифровой клавиатуре;
Multiply - клавиша умножения;
Add - клавиша сложения;
Divide - клавиша деления;
Subtract - клавиша вычитания;
F1 ? F12 - клавиша <F1> ? клавиша <F12>.

Пример 1. Нижеприведенный фрагмент кода, используя перечисление Keys, демонстрирует возможность определения типа символа, введенного в элемент управления TextBox.

```
private void textBox1_KeyDown(object sender, KeyEventArgs e)
                {
                  if (e.KeyCode >= Keys.A && e.KeyCode <= Keys.Z)
                      label1.Text = "Введена буква";
                   else label1.Text = "Введен символ, не являющийся буквой";
                 }
```

Пример 2. Нижеприведенный фрагмент кода, используя перечисление Keys, демонстрирует возможность определения нажатия сочетания клавиш <Alt>+<E>.

```
private void textBox1_KeyDown(object sender, KeyEventArgs e)
                {
                 if (e.Alt && e.KeyCode == Keys.E)
                   label1.Text = "Нажата комбинация Alt + E";
                }
```

- **KeyData** - содержит данные, касающиеся клавиши. Константы из значения **Keys** служат для извлечения сведений из свойства **KeyData**. Побитовая операция **И** служит для сравнения данных, возвращенных классом **KeyData**, с константами в значении **Keys** для получения сведений о каждой нажатой клавише. Свойства **Control**, **Shift** и **Alt** служат для определения, была ли нажата клавиша <Alt>, <Ctrl> или <Shiht>.

- **KeyValue** - содержит значение клавиатуры (целое число типа Int32).

- **Modifiers** - содержит флаги модификаторов, которые указывают, какая комбинация клавиш <Ctrl>, <Shift> и <Alt> была нажата.

- **Shift** - содержит значение, показывающее, была ли нажата клавиша <Shift>. Имеет тип **bool**. Значение **true**, если клавиша <Shift> была нажата, в противном случае — значение **false**.

Событие **KeyPress** вызывается только нажатием клавиш с символами. Заголовок обработчика события **KeyPress** может иметь, например, следующий вид:

```
private void Form1_KeyPress(object sender, KeyPressEventArgs e)
    {
    }
```

Объект **KeyPressEventArgs** предоставляет данные для событий **KeyPress** и имеет следующие свойства:

- **Handled** – содержит значение, показывающее, было ли обработано событие KeyPress. Имеет тип **bool**. Значение **true**, если событие обработано, в противном случае — значение **false**. Если событие не обработано, оно будет отправлено для обработки по умолчанию операционной системой.

- **KeyChar** используется для выбора символа клавиши во время выполнения и для использования или изменения подмножества стандартных нажатий клавиш. Получаемый знак ASCII, например, при нажатии клавиш <Shift> +<K> данное свойство возвращает прописной знак K.

Можно получить или задать следующие клавиши: a ? z, A ? Z, Ctrl, знаки пунктуации, числовые клавиши, находящиеся на клавиатуре или на цифровой панели, Enter.

Нельзя получить или задать следующие клавиши: Tab, Insert, Delete, Home, End, PageUp, PageDown, F1 ? F12, Alt, клавиши со стрелками.

Пример 3. Нижеприведенный фрагмент кода демонстрирует возможность определения символа, введенного в элемент управления TextBox.

```
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
        {
         switch (e.KeyChar)
             {
              case 'G': label1.Text = "Введена буква G"; break;
              case 'g': label1.Text = "Введена буква g"; break;
              case 'П': label1.Text = "Введена буква П"; break;
              case 'п': label1.Text = "Введена буква п"; break;
              default: label1.Text = "Нажата другая клавиша"; break;
             }
        }
```

Пример 4. Нижеприведенный фрагмент кода демонстрирует возможность ввода в текстовое поле TextBox только цифры и символа Backspace:

```
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
        {
         if (char.IsDigit(e.KeyChar) == true) return;
         if (e.KeyChar == (char)Keys.Back) return;
         e.Handled = true; // запрет на ввод других вводимых символов
        }
```

2. Рабочее задание

Задание 1. Разработать приложение «Исследуем клавиатуру», с помощью которого фиксируется название клавиши и символа.

При разработке интерфейса формы «Исследуем клавиатуру» использовать компоненты, с помощью которых выводятся сообщения, например, **Label** (один из вариантов интерфейса

представлен на рис. 1).

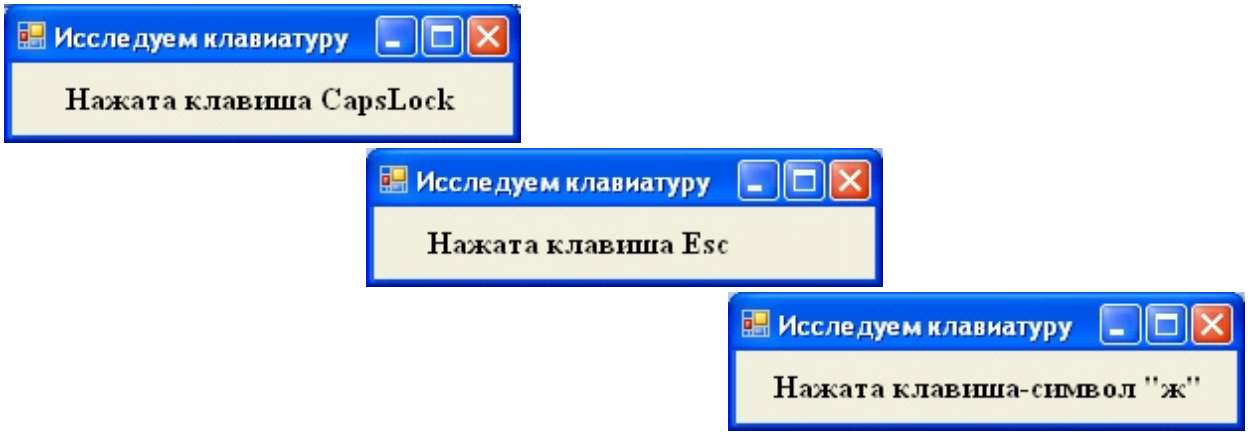

Рис. 1. Внешний вид приложения «Исследуем клавиатуру»

Задание 2. Разработать приложение «Протокол событий», с помощью которого отслеживаются действия пользователя по работе с клавиатурой и мышью. Действия отображаются в окне наблюдения (например, компонент **ListBox**).

При разработке интерфейса формы «Протокол событий» использовать компоненты **ListBox, Button** (один из вариантов формы «Работа с текстом» представлен на рис. 2).

Для работы с датой и временем использовать статическое свойство **Now** класса **DateTime**, например, оператор ***DateTime dt = DateTime.Now;*** инициализирует переменную **dt** текущим значением даты и времени. Для преобразования даты и времени в строку использовать метод **ToString()**.

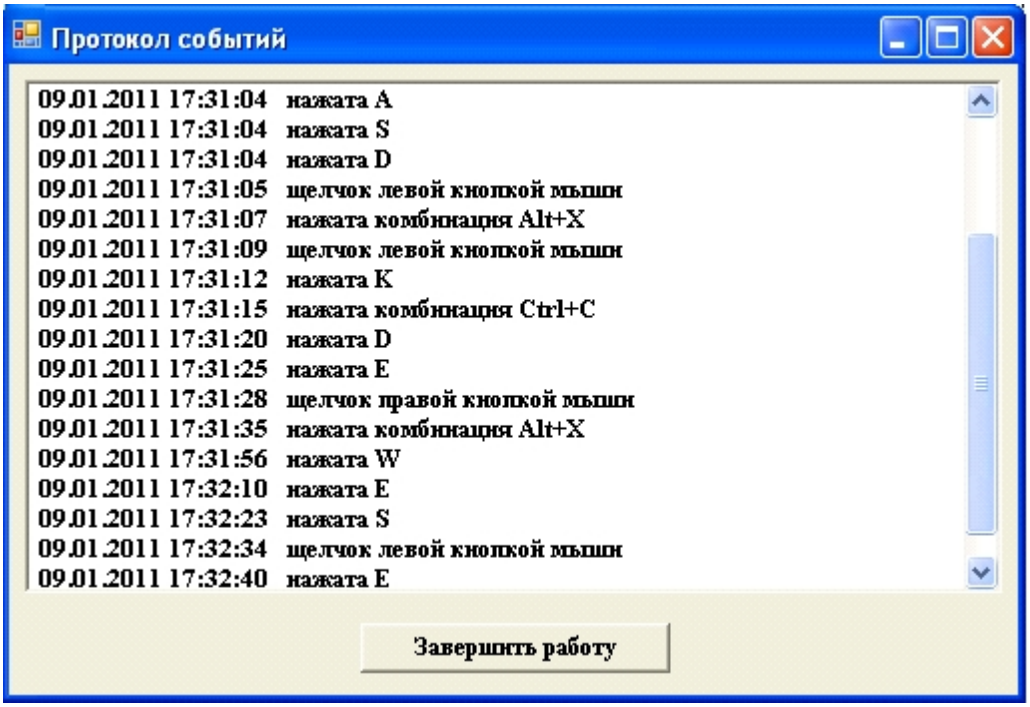

Рис. 2. Внешний вид приложения «Протокол событий»

Задание 3. Разработать приложение «Скорость набора текста», с помощью которого исследовать скорость набора текста пользователем.

Варианты текста записаны в компонентах **TextBox**, выбор варианта текста осуществляется с помощью компонента **RadioButton**. Началом отсчета считается момент выбора варианта текста, окончанием – нажатие кнопки «Определить».

При разработке интерфейса формы «Скорость набора текста» использовать компоненты **Label, Button, TextBox, RadioButton** (один из вариантов формы «Скорость набора текста» представлен на рис. 3).

Для работы с датой и временем использовать статическое свойство **Now** класса **DateTime**, например, оператор ***DateTime dt = DateTime.Now;*** инициализирует переменную **dt** текущим значением даты и времени. Для получения значение минут для даты, представленной экземпляром класса **DateTime**, использовать свойство **Minutes**, секунд – **Seconds**.

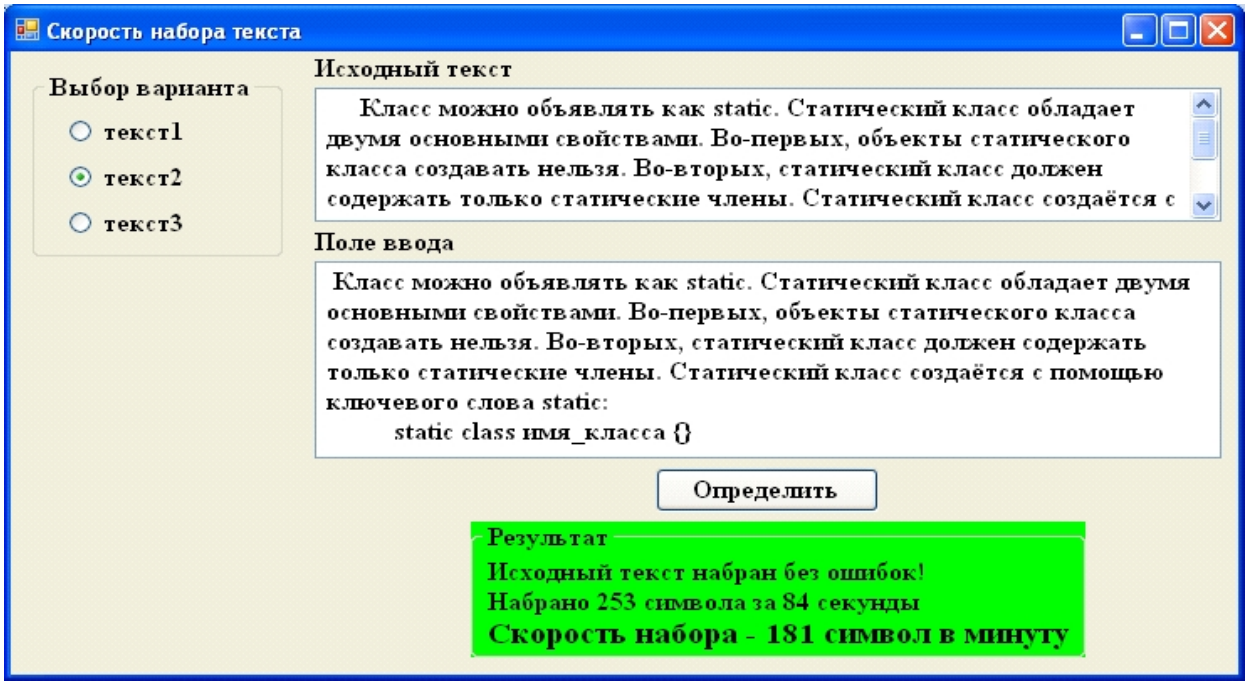

Рис. 3. Внешний вид приложения «Скорость набора текста»

3. Контрольные вопросы

Литература

1. Голощапов А.Л. Microsoft Visual Studio 2010. – СПб.:БХВ-Петербург, 2011. – 544 с.: ил.
2. Культин Н.Б. Microsoft Visual C# в задачах и примерах. – СПб.: БХВ-Петербург, 2009. – 320 с.: ил.
3. Лабор В.В. Си Шарп: Создание приложений для Windows. – Мн.: Харвест, 2003. – 384 с.
4. Петцольд Ч. Программирование для Microsoft Windows на C#. В 2-х томах. Том 1. Пер. с англ. - М.: «Русская Редакция», 2002.- 576 с.: ил.
5. Петцольд Ч. Программирование для Microsoft Windows на C#. В 2-х томах. Том 2. Пер. с англ. - М.: «Русская Редакция», 2002.- 624 с.: ил.
6. Троелсен Э. Язык программирования C# 2010 и платформа .NET 4.0. Пер. с англ. - М.: Издательский дом "Вильямс", 2011. — 1392 с.: ил.
7. Фаронов В.В. Программирование на языке C#. – СПб.: Питер, 2007. – 240 с.: ил.
8. Фленов М.Е. Библия C#. - СПб.: БХВ-Петербург, 2011.– 560с.: ил.