ФГБОУ ВПО «Воронежский государственный технический университет»
Кафедра компьютерных интеллектуальных технологий проектирования

114-2015

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам № 10-13 по дисциплине «Среды визуального программирования» для студентов направления 09.03.02 «Информационные системы и технологии» (профиль «Информационные системы и технологии в машиностроении») очной формы обучения

Воронеж 2015

Составители: канд. техн. наук А.Н. Юров,
канд. техн. наук А.В. Бредихин

УДК 004.9

Методические указания к лабораторным работам № 10-13 по дисциплине «Среды визуального программирования» для студентов направления 09.03.02 «Информационные системы и технологии» (профиль «Информационные системы и технологии в машиностроении») очной формы обучения / ФГБОУ ВПО «Воронежский государственный технический университет»; сост. А.Н. Юров, А.В. Бредихин. Воронеж, 2015. 22с.

Методические указания содержат материал по созданию приложений на языке программирования C# в среде визуальной разработки Visual Studio, а также практические задачи и перечень заданий для выполнения лабораторных работ по дисциплине «Среды визуального программирования».

Предназначены для студентов 2 курса.

Методические указания подготовлены в электронном виде в текстовом редакторе MS Word 2013 и содержатся в файле lPart3.docx.

Табл. 3. Ил. 7. Библиогр.: 9 назв.

Рецензент канд. физ.-мат. наук, доц. Н.А. Тюкачев

Ответственный за выпуск зав. кафедрой д-р техн. наук, проф. М.И. Чижов

Издается по решению редакционно-издательского совета Воронежского государственного технического университета

ВВЕДЕНИЕ

В последнее время становится значимым разработка приложений в средах визуального программирования в связи с ростом производительности вычислительных средств ЭВМ как стационарных, так и мобильных сенсорных устройств (КПК, планшеты). Способ создания программ для ЭВМ путем манипулирования графическими объектами вместо написания кода вручную является достаточно доступным и простым.

Такие визуальные средства являются средами программирования, в которые интегрированы библиотеки готовых объектов и методов для выполнения общих для большинства приложений задач, в частности задачи наглядного и стандартизованного получения и отображения информации.

В данных методических указаниях представлен материал по созданию консольных и графических приложений на объектно-ориентированном языке C# в среде Visual Studio 2012. Все примеры могут быть использованы в иных средах разработки, поддерживающие написание программ на C# в операционных системах, включая Windows, Linux и другие.

ЛАБОРАТОРНАЯ РАБОТА № 10
РАБОТА С ФАЙЛАМИ

Цель работы: разработать приложение в среде визуального программирования (Visual Studio) согласно заданию.

Задачи и требования к выполнению:

1.Изучить работу с папками и файлами на программном уровне.

Теоретические сведения

Пространство имен System.IO содержит классы для чтения данных из файлов и записи их в файлы и проект должен ссылаться на это пространство using System.IO. Как видно из рис. 1, в System.IO содержится несколько классов предназначенных для работы с файлами, но рассматриваться будут только некоторые классы, необходимые для файлового ввода и вывода (табл.1-2).

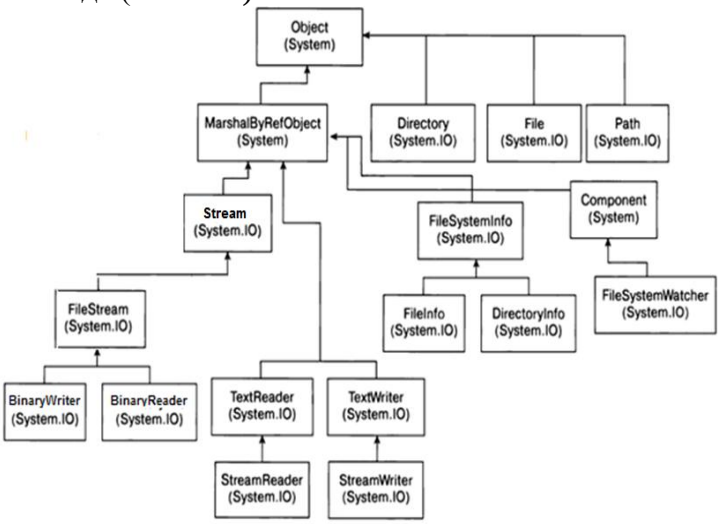

Рис. 1. Иерархия классов ввода-вывода

Набор методов класса Stream Таблица 1

Метод	Описание
void Close()	Закрывает поток
void Flush ()	Записывает содержимое потока в физическое устройство
int ReadByte ()	Возвращает целочисленное представление следующего доступного байта потока При обнаружении конца файла возвращает значение -1
int Read(byte[] buf, int offset, int numBytes	Делает попытку прочитать numBytes байт в массив buf, начиная с элемента buf[offset], возвращает количество успешно прочитанных байтов
long Seek(long offset, SeekOrigin origin)	Устанавливает текущую позицию потока равной указанному значению смещения от заданного начала отсчета
void WriteByte (byte b)	Записывает один байт в выходной поток
void Write (byte[] buf, int offset, int numBytes)	Записывает поддиапазон размером numBytes байт из массива buf, начиная с элемента buf[offset]

Классы по работе с файлами Таблица 2

Класс	Описание
File	Статический служебный класс, предоставляющий множество статических методов для перемещения, копирования и удаления файлов
Directory	Статический служебный класс, предоставляющий множество статических методов для перемещения, копирования и удаления каталогов

Продолжение табл. 2

Path	Служебный класс, используемый для манипулирования путевыми именами
FileInfo	Представляет физический файл на диске, имеет методы для манипулирования этим файлом. Для любого, кто читает или пишет в этот файл, должен быть создан объект stream
DirectoryInfo	Представляет физический каталог на диске и имеет методы для манипулирования этим каталогом
FileSystemInfo	Служит базовым классом для FileInfo и DirectoryInfo, обеспечивая возможность работы с файлами и каталогами одновременно, используя полиморфизм
FileStream	Представляет файл, который может быть записан, прочитан или то и другое. Этот файл может быть записан или прочитан как синхронно, так и асинхронно
StreamReader	Читает символьные данные из потока и может быть создан с использованием класса FileStream в качестве базового
StreamWriter	Пишет символьные данные в поток и может быть создан с использованием класса FileStream в качестве базового
FileSystemWatcher	Наиболее развитый класс. Используется для мониторинга файлов и каталогов и представляет события, которые приложение может перехватить, когда в этих объектах происходят какие-то изменения.

Выше представлены таблицы с методами и классами для работы с файлами в C#. Рассмотрим класс StreamWriter подробнее.

Получив объект FileStream, можно поместить его в оболочку StreamWriter или StreamReader и использовать их методы для работы с файлом. Эти классы значительно облегчают работу с текстовыми файлами, но не позволяют переставлять файловый указатель в произвольную позицию.

Класс StreamWriter позволяет записывать символы и строки в файл, оставляя классу внутреннее преобразование и запись объекта FileStream.

Запись в текстовый файл должна состоять из следующих шагов:

-создать поток FileStream;

-создать объект StreamWriter, связав его с потоком;

-методом Write() или WriteLine() записать данные;

-методом Close(), освободив промежуточный буфер, закрыть файл;

-методом Close(), закрыть файловый поток.

Листинг 1. Пример консольного приложения с записью строк в файл

```
class WriteTextFile
{
    static void Main()
    {

        // Эти примеры предполагают "C: \ Users \
Public \ TestFolder" папку на вашем компьютере.
        // Вы можете изменить путь, если это
необходимо.

        // Пример # 1: Написать массив строк в файл.
        // Создание массива  состоящего из трех
строк.
        string[] lines = { "First line", "Second
line", "Third line" };
        // WriteAllLines создает файл, записывает
набор строк в файл,
        // И затем закрывает файл.

      System.IO.File.WriteAllLines(@"C:\Users\Public\T
estFolder\WriteLines.txt", lines);
```

```
        // Пример # 2: Написать одну строку в
текстовом файле.
        string text = "A class is the most powerful
data type in C#. Like a structure, " +
                      "a class defines the data and
behavior of the data type. ";
        // WriteAllText создает
файл, записывает заданную строку в файл,
         // И затем закрывает файл.
System.IO.File.WriteAllText(@"C:\Users\Public\TestFold
er\WriteText.txt", text);

        // Пример #3: Записать только некоторые строки
в массиве в файл.
        // IDisposable.Dispose на объекте потока.
        using (System.IO.StreamWriter file = new
System.IO.StreamWriter(@"C:\Users\Public\TestFolder\Wr
iteLines2.txt"))
        {
            foreach (string line in lines)
            {
                // Если строка не содержит слово
"второй", напишите строку в файл.

      if(!line.Contains("Second"))
                {
                    file.WriteLine(line);
                }
            }
        }
```

Листинг 2. Пример консольного приложения с чтением строк из текстового файла

```
static void Main(string[] args)
        {
            try
            {
                FileStream aFile =
                    new FileStream("Test.txt",
                       FileMode.Open);
                StreamReader f =
                    new StreamReader(aFile);
                string s = f.ReadLine();
                //Прочитать все строки
```

```
            while (s != null)
            {
                Console.WriteLine(s);
                s = f.ReadLine();
            }
            f.Close();
            aFile.Close();
        }
        catch (IOException e)
        {
            Console.WriteLine("Ошибка IO ");
            Console.WriteLine(e.ToString());
            return;
        }
        Console.ReadKey();
    }
```

Задания на самостоятельную работу:

1.Разработать редактор текста с элементами управления (меню), а также с возможностью чтения-записи данных в файл без графической информации.

2.Разработать графическое представление на форме работы консольной команды ‘tree’ (как показано на рис.2).

Рис. 2. Пример приложения

ЛАБОРАТОРНАЯ РАБОТА № 11
РАБОТА С КЛАВИАТУРОЙ И МЫШЬЮ В C#

Цель работы: разработать приложение в среде визуального программирования (Visual Studio) согласно заданию.

Задачи и требования к выполнению:

1.Изучить работу с клавиатурой и мышью на языке C#.

2.Изучить функционал клавиатуры и мышки в Visual Studio 2012.

Теоретические сведения

Базовыми устройствами по управлению приложениями на данный момент остаются по-прежнему клавиатура и манипулятор "мышь", хотя все большую популярность приобретают сенсорные дисплеи и специализированное ПО (ОС) по их поддержке.

Предоставление исходного входного API-интерфейс основано на базовых классах элемента UIElement, ContentElement, FrameworkElement и FrameworkContentElement. Эти классы предоставляют функциональные возможности назначения событий ввода, связанных, в том числе, с нажатием клавиш, кнопок мыши, колеса мыши, движением мыши, управлением фокуса и захватом мыши. Размещая API-интерфейс входных данных в базовых элементах вместо обработки всех событий ввода в качестве службы, архитектура ввода позволяет определенному объекту пользовательского интерфейса порождать события ввода, которые поддерживают схему маршрутизации событий. В результате становится возможным обрабатывать событие ввода нескольким элементам одновременно.

Классы клавиатуры и мыши.

В дополнение к входному API-интерфейс на базовых классах элемента, класс Keyboard и

классы Mouse предоставляют дополнительные API-интерфейс для работы с вводом с клавиатуры и мыши.

Примерами входных API-интерфейс в классе Keyboard являются свойство Modifiers, которое возвращает нажатую в данный момент ModifierKeys, и метод IsKeyDown, определяющий, нажата ли указанная клавиша.

Далее будут представлены некоторые методы и примеры их реализации по работе с мышью.

Позиционирование указателя мыши:

```
 private void button1_Click(object sender, EventArgs e)
            {
                Point pt = Cursor.Position;
                pt.X += 100;
                Cursor.Position = pt;
            }
```

Скрыть указатель с экрана достаточно просто, достаточно использовать следующий метод:

```
Cursor.Hide();
```

Чтобы снова показать курсор, используйте метод:

```
Cursor.Show();
```

Получить координаты и название кнопок мыши:

```
private void Form1_MouseClick(object sender, MouseEventArgs e)
            {
                String str;
                str = e.Button.ToString() + " " + e.Location.ToString();
                statusLabel.Text = str;
            }
```

Поменять местами кнопки мыши:

```
    [DllImport("user32.dll", EntryPoint =
"SwapMouseButton")]
            internal extern static int
SwapMouseButton(int bSwap);
            private                    void
button1_Click(object sender, EventArgs e)
            {
                SwapMouseButton(1);
            }
            private                    void
button2_Click(object sender, EventArgs e)
            {
                 SwapMouseButton(0);
            }
```

Теперь рассмотрим работу с клавиатурой. Она является основным инструментом по вводу данных в ЭВМ. Далее приведены некоторые коды по работе с клавиатурой.

Изменение языка ввода данных:

```
    using System.Globalization;
    private    void    button1_Click(object
sender, EventArgs e)
            {

InputLanguage.CurrentInputLanguage =

InputLanguage.FromCulture(new
CultureInfo("ru-RU"));
            }
            private                    void
button2_Click(object sender, EventArgs e)
            {

InputLanguage.CurrentInputLanguage =
```

```
InputLanguage.FromCulture(new
CultureInfo("en-US"));
            }
```

Результат выполнения указан на рис.3.

Определение текущего состояния клавиш-индикаторов:

```
String str;
                bool                CapsKey
=Control.IsKeyLocked(Keys.CapsLock);
                bool        NumKey        =
Control.IsKeyLocked(Keys.NumLock);
                if (CapsKey)
                {
                    str  =  "Caps  Key:  "  +
"ON";
                }
                else
                {
                    str  =  "Caps  Key:  "  +
"OFF";
                }
                if (NumKey)
                {
                    str = str+"  "+"Num Key:
" + "ON";
                }
                else
                {
                    str = str + "   " + "Num
Key: " + "OFF";
                }
```

StatusLabel1.Text = str;

Результат выполнения указан на рис.4.

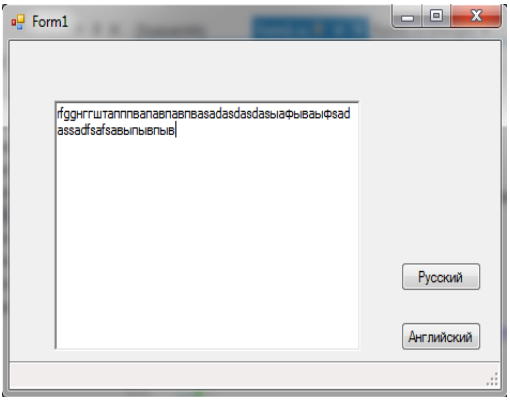

Рис. 3. Результат выполнения

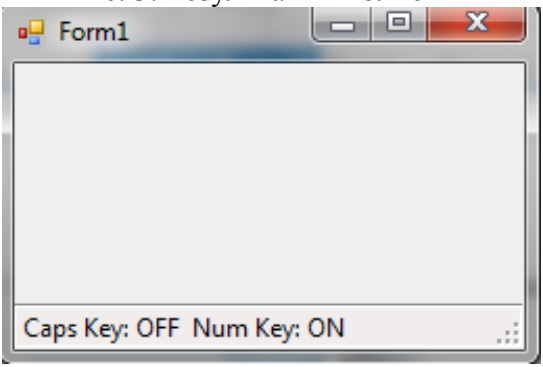

Рис. 4.Результат выполнения

Задания на самостоятельную работу:

1.На рис.5 представлены варианты объектов. Известна только сторона квадрата, а окружность является вписанной в квадрат. Если пользователь указывает место курсором мыши на контур, образованный фигурами, то он должен быть закрашен. При этом должно появиться сообщение со значением площади закрашенной фигуры.

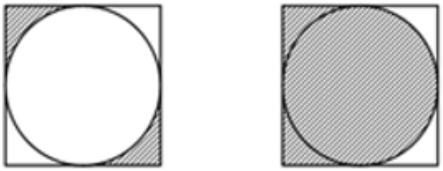

Рис. 5.Варианты к выполнению задания

ЛАБОРАТОРНАЯ РАБОТА № 12
РАЗРАБОТКА ПАРАМЕТРИЧЕСКОГО ИНТЕРФЕЙСА ПРИЛОЖЕНИЯ

Цель работы: разработать приложение в среде визуального программирования (Visual Studio) согласно заданию.

Задачи и требования к выполнению:

1.Изучить работу с параметрическим интерфейсом в C#.

2.Рассмотреть некоторые возможные примеры методов для работы с параметрами интерфейса.

Теоретические сведения

Вследствие того, что экраны персональных ЭВМ или мобильных устройств имеют разную разрешающую способность, актуальность приобретает применение параметрического интерфейса для всего приложения.

Определить разрешение экрана (рис. 6) возможно с помощью следующего кода:

```
      private     void     button1_Click(object
sender, EventArgs e)
              {
     textBox1.Text=Screen.PrimaryScreen.Bound
s.Width.ToString()+”x”

Screen.PrimaryScreen.Bounds.Height.ToString()
+”x”
```

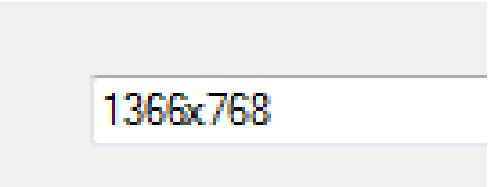

Рис. 6. Пример определения расширения экрана

Далее, определить позицию формы на экране:

```
      public Form1()
              {
                  InitializeComponent();

this.SetBounds(Screen.GetWorkingArea(this).Wi
dth -

this.Width,Screen.GetWorkingArea(this).Height
-
             this.Height,          this.Width,
this.Height);
```

Или же программно свернуть (развернуть) форму. У формы есть свойство WindowState. Для того, чтобы свернуть форму, необходимо добавить следующий код:

WindowState = FormWindowState.Minimized

Вывод запроса (рис.7) при закрытии формы:

```
     private  void  Form1_FormClosing(object
sender, FormClosingEventArgs e)
              {
                  if  (MessageBox.Show("Работа
с программой будет завершена?",

"Warning",MessageBoxButtons.YesNo)==DialogRes
ult.No)
                  {
                      e.Cancel=true;
                  }
              }
```

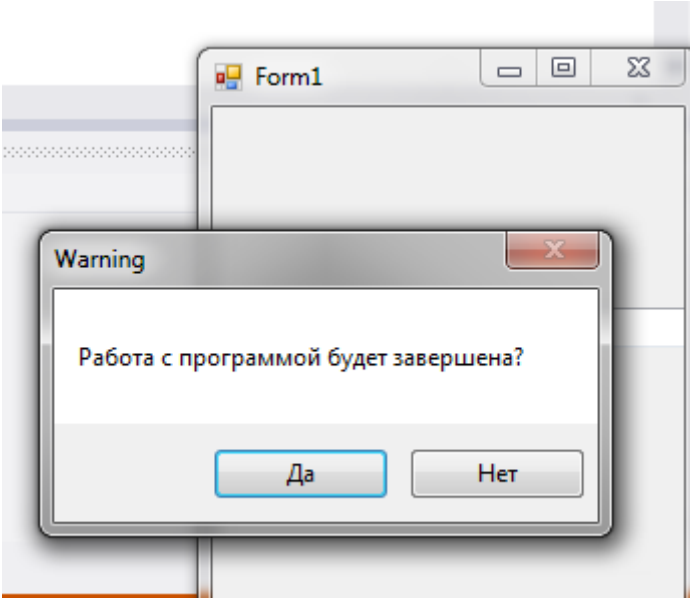

Рис. 7. Результат вывода запроса

Код для отображения формы на весь экран:

```
FormBorderStyle = FormBorderStyle.None;
                WindowState                    =
FormWindowState.Maximized;

Установка ограничения на размер формы
MaximumSize=new Size(400,400);
MinimumSize=new Size(250,250);

Эффект полупрозрачности формы
this.Opacity=(double)numericUpDown1.Valu
e/100;
```

Задания на самостоятельную работу:

1.Подготовить параметрический интерфейс приложения и выполнить решение следующей задачи:

Стороны прямоугольника заданы натуральными числами A и B. Определить насколько квадратов, стороны которых выражены натуральными числами, можно разрезать данный прямоугольник, если от него каждый раз отрезается квадрат максимально большой площади.

ЛАБОРАТОРНАЯ РАБОТА № 13
РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ

Цель работы: разработать приложение в среде визуального программирования (Visual Studio) согласно заданию.

Задачи и требования к выполнению:

1.Изучить работу с регулярными выражениями в C#.

2.Изучить основные выражения и их действие.

Теоретические сведения

Регулярное выражение (regular expression, regexp) - это стандарт для поиска и замены текста в строках.

Поддержка регулярных выражений в .Net выполняется классами пространства имен:

using System.Text.RegularExpressions

Основные классы:

Regex - постоянное регулярное выражение.

Match - предоставляет результаты очередного применения всего регулярного выражения к исходной строке.

MatchCollection - предоставляет набор успешных сопоставлений, при итеративном применении образца регулярного выражения к строке.

Capture - предоставляет результаты отдельного захвата подвыражения.

Group - предоставляет результаты для одной регулярной группы.

GroupCollection - предоставляет коллекцию найденных групп и возвращает набор групп как одно соответствие.

CaptureCollection - предоставляет последовательность найденных подстрок и возвращает наборы соответствий отдельно для каждой группы.

Конструктор класса Regex имеет два перегружаемых метода, для второго из которых options - поразрядная комбинация OR значений перечисления RegexOption.

public Regex
(string pattern,
RegexOptions options);

RegexOptions может содержать следующие записи:

Compiled - регулярное выражение компилируется в сборку, что значительно увеличивает скорость их выполнения, но снижает скорость загрузки. Кроме того, скомпилированные регулярные выражения выгружаются только при завершении работы приложения, даже когда сам объект Regex освобождён и уничтожен сборщиком мусора.

CultureInvariant - игнорировать культурные различия в языках.

ECMAScript - включить поведение для выражения, соответствующее ECMAScript. Используется только в соединении с флагами IgnoreCase, Multiline и Compiled, с другими дает ошибку.

ExplicitCapture - Указывает, что только верные явно названные или пронумерованные (с помощью конструкции (?)) группы сохраняются. Это позволяет избежать излишнего использования конструкции (?::).

IgnoreCase - игнорировать регистр.

IgnorePatternWhitespace - устраняет пробелы из шаблонов и позволяет использовать комментарии, отмеченные знаком "#".

Multiline - значения ^ и $ обозначают начало и конец каждой строчки текста.

RightToLeft - поиск справа налево.

SingleLine - весь текст как одна строка. Символ точки в данном режиме включает и \n.

Основные методы класса - это:

Match - поиск соответствия. Метод возвращает объект класса Match, содержащий результаты найденного соответствия.

Matches - поиск всех непересекающихся соответствий и возвращение объекта MatchCollection.

NextMatch - продолжение поиска соответствия для Match с позиции последнего найденного.

Приведенный на листинге 3 пример разделяет строки по заданному шаблону:

Листинг 3. Пример программы

```
using System.Text.RegularExpressions;
namespace RegularExp
{
    class Program
    {
        static void Main(string[] args)
        {
            string s = "cdabcdeabrtfabqwe";
            Regex regex = new Regex("ab");
            string[] s1 = regex.Split(s, 4);
            string text = "";
            for (int i = 0; i < s1.Length; i++)
            {
                text += s1[i] + " ";
            }
            Console.WriteLine(text);
        }
    }
}
//Результат:
//cd cde rtf qwe
```

В табл. 3 приведены наиболее часто употребляемые записи инструкций, а также определено их действие в составе регулярного выражения.

Запись выражений и их действие Таблица 3

Выражение	Действие
[абв]	Любой символ: а, б или в
[^абв]	Любой символ кроме: а, б или в
.	Любой знак кроме \n
\w	Буква либо цифра
\W	Любой символ кроме букв и цифр
\s	Пробел
\S	Любой символ кроме пробела
\d	Десятичная цифра
\D	Любой символ кроме десятичной цифры
^	Начало строки
$	Конец строки
\A	В начале строки
\Z	В конце строки
\G	В месте, где заканчивается пред. соответствие
\b	Граница между \w и \W
\B	Не граница между \w и \W
\r	Возврат каретки
\n	Новая строка
*	Ноль или более раз
+	Один или более раз
?	Ноль или один раз
{n}	n раз
{n,}	Минимум n раз
{n,m}	Минимум n раз, но меньше чем m
*?	Не повторяется, либо повторяется как можно меньше
+?	Повторяется один или больше раз, но как можно меньше
??	Повторяется один или ноль раз, но как можно меньше
( )	Выделяет часть выражения в

Продолжение табл. 3

	группу
(?<имя>часть_выражения)	Именованная группа
(?=)	Положительный просмотр вперёд
(?!)	Отрицательный просмотр вперёд
(?<=)	Положительный просмотр назад
(?<!)	Отрицательный просмотр

Задания на самостоятельную работу:

1. Разработать диалог авторизации пользователя в приложении с использованием регулярных выражений.

Условие:

-в имени пользователя использовать только латинские буквы (без учета регистра) от 8 символов до 15;

-в пароле использовать только цифры. Количество цифр определено диапазоном от 6 до 10.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Прата С. Язык программирования С++. Лекции и упражнения / С. Прата. 5-е изд. – М.: ООО ”И.Д. Вильямс”, 2007. – 1184 с.
2. Страуструп Б Язык программирования С++ / Б. Страуструп. - М.: Бином, 2011. – 1136 с.
3. Шилдт Г. С++ Базовый курс / Г. Шилдт. 3-е изд. – М.: ООО ”И.Д. Вильямс”, 2010. – 624 с.
4. Шилдт Г. Полный справочник по C# / Г. Шилдт. – 4-е изд. – М.: Вильямс, 2009. – 800 с.
5. Шилдт Г. Самоучитель C#/ Г. Шилдт. – 3-е изд. – 3-е изд. – СПб.: БХВ-Петербург, 2002. – 688 с.
6. Дейтел Х. C# / Х. Дейтел, П. Дейтел, Дж. Листфилд, Т.Нието, Ш. Йегер и др. – СПб.: БХВ-Петербург, 2006. – 1056 с.
7. Агупов П.В. C#. Разработка компонентов в MS Visual Studio 2005/2008 / П.В. Агупов - СПб.: БХВ-Петербург, 2008. – 480 с.
8. Бишоп Дж. C# в кратком изложении / Дж. Бишоп, Н. Хорспул. - М.: Бином, 2005. – 472 с.
9. London J. Modeling Derivatives in C++ / London J. Wiley, 2005. - 841p.

СОДЕРЖАНИЕ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам № 10-13 по дисциплине
«Среды визуального программирования» для студентов
направления 09.03.02 «Информационные системы
и технологии» (профиль «Информационные системы
и технологии в машиностроении») очной формы обучения

Составители:
Юров Алексей Николаевич
Бредихин Алексей Вячеславович

В авторской редакции

Компьютерный набор А.Н. Юрова

Подписано к изданию 11.02.2015.
Уч.-изд. л. 1,3. «С»

ФГБОУ ВПО «Воронежский государственный технический
университет»
394026 Воронеж, Московский просп., 14