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

113-2015

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам № 5-9 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 № 5-9 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. 24 с.

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 lPart2.docx.

Табл. 1. Ил. 17. 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

Современное развитие вычислительной техники и информационных технологий предполагает использование программных продуктов практически во всех областях хозяйственной деятельности. В последнее время все более актуальным становится разработка приложений в средах визуального программирования в связи с развитием мобильных сенсорных устройств (КПК, планшеты). Способ создания программ для ЭВМ путем манипулирования графическими объектами вместо написания кода вручную является достаточно доступным и простым.

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 № 5
ПОСТРОЕНИЕ ПРИЛОЖЕНИЙ С ГРАФИЧЕСКИМ ИНТЕРФЕЙСОМ (GUI) НА C#

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

Задачи и требования к выполнению:

1.Изучить возможности построения приложений с графическим интерфейсом.

2.Изучить работу с шаблонами графического интерфейса.

Теоретические сведения

Графический интерфейс позволяет пользователям программного обеспечения быстрее осваивать работу с разными приложениями в операционных системах, так как управляющие компоненты, как правило, идентичны и в других разработанных решениях.

Кроме того, благодаря интерактивности можно добиться возможностей, которые подправят действия пользователя во время работы, предоставят справочные материалы и т.д. (рис. 1).

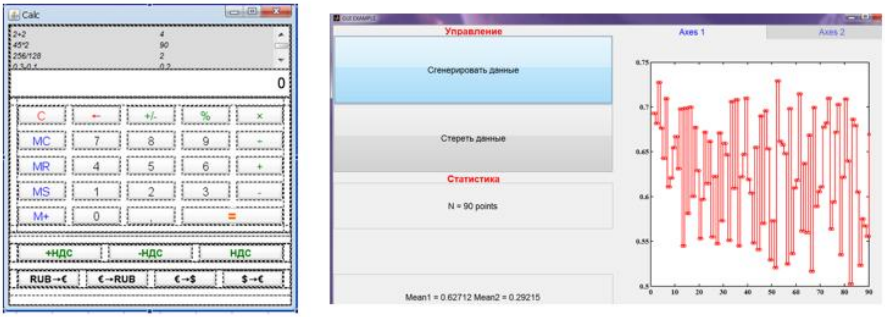


Рис. 1. Примеры разработки графических интерфейсов

В среде Visual Studio 2012 доступны следующие шаблоны для построения приложений с графическим интерфейсом: Windows Forms Application, WPF Application, ASP.NET Web Forms Application, ASP.NET 3 Web Application, ASP.NET 4 Web Application. Рассмотрим работу с шаблоном Windows Form.

В указанном шаблоне автоматически добавляются наиболее важные ссылки и файлы проекта, используемые в качестве отправной точки при создании приложения.

Пошаговая инструкция, как создать проект приложения Windows Forms.

В строке меню следует выбрать: Файл - Создать – Проект. Диалоговое окно должно выглядеть следующим образом, как показано на рис. 2.

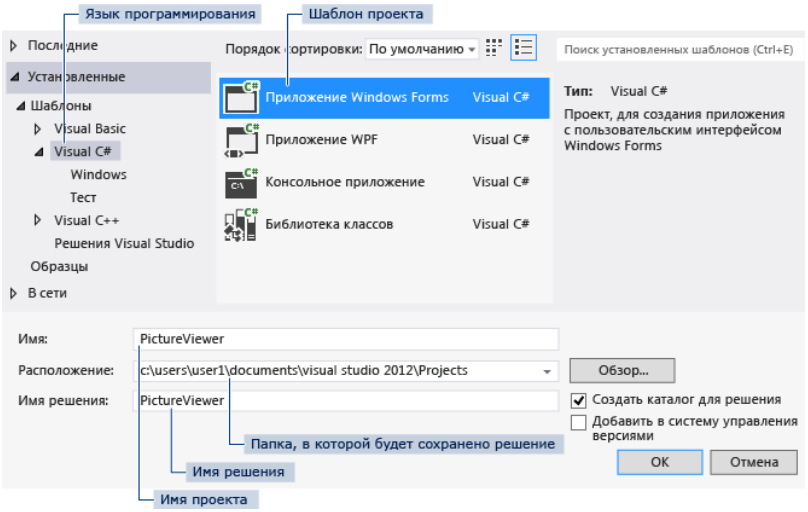


Рис. 2. Диалоговое окно "Новый проект"

В списке Установленные шаблоны указать Visual C# или Visual Basic.

Далее в списке шаблонов отметить значок Приложение Windows Forms и определить имя новой формы. После этого

следует перейти к следующему этапу работы над проектом, нажав на кнопку OK.

Visual Studio создает решение для программы. Решение играет роль контейнера для всех проектов и файлов, необходимых программе.

На следующем рис. 3 показано, как теперь должен выглядеть интерфейс Visual Studio.

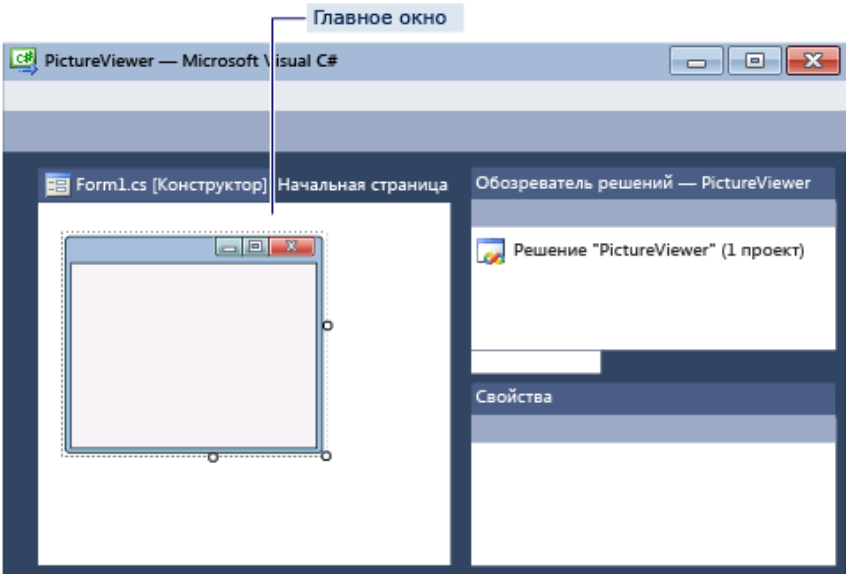


Рис. 3. Окно интегрированной среды разработки

Интерфейс содержит три окна: главное окно, Обозреватель решений и окно Свойства.

Если любое из этих окон отсутствует, можно восстановить макет окон по умолчанию, выбрав в строке меню Окно, Сброс макета окон. Возможно также отображение окон с помощью команд меню. В строке меню требуется выбрать Вид, Окно "Свойства" или Обозреватель решений. Если какие-либо другие окна открыты, работу с ними надо завершить с помощью кнопки Закрыть.

После того, как разработчиком выбран прототип приложения Windows Form, необходимо разместить панель с инструментами и свойствами управляющих элементов в среде VS. Далее производится работа с формой, необходимые элементы управления просто переносятся на форму в зависимости от идей разработчика приложения. Пример такой компоновки показан на рис.4.

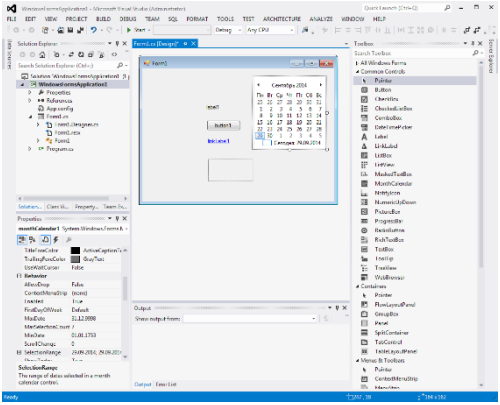
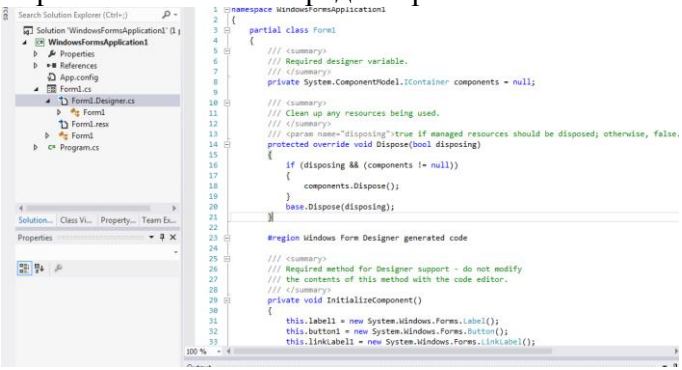


Рис. 4. Создание пользовательского интерфейса

По завершении графической компоновки, можно вернутся к текстовому редактору проекта и открыть необходимые файлы для их изменения. Поправить те или иные компоненты на форме “вручную” можно в специальном методе Windows Form- InitializeComponent(). На рис. 5 показана работа в текстовом редакторе VS 2012.



```
namespace WindowsFormsApplication1
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.label1 = new System.Windows.Forms.Label();
            this.button1 = new System.Windows.Forms.Button();
            this.linkLabel1 = new System.Windows.Forms.LinkLabel();
```

Рис. 5. Работа с текстовым редактором

Задания на самостоятельную работу

Разработать упрощенный инженерный калькулятор с поддержкой трех систем счисления.

ЛАБОРАТОРНАЯ РАБОТА № 6
РАБОТА С ГРАФИКОЙ НА C#

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

Задачи и требования к выполнению:

1.Изучить работу с графикой с помощью GDI+.

2.Изучить объекты изображений и класс Graphics.

Теоретические сведения

В состав .NET Framework входит API GDI+ для работы с графикой. С помощью GDI+ можно создавать рисунки, рисовать текст и управлять графическими изображениями как объектами.

GDI+ отличается высоким быстродействием и удобен в использовании. Интерфейс GDI+ можно использовать для вывода графических изображений в формах Windows Forms и элементах управления. Хотя GDI+ нельзя использовать непосредственно в веб-формах, можно выводить графические изображения с использованием серверного веб-элемента управления Image.

Для рисования изображения в .NET Framework необходимо использовать объект Graphics, связанный с изображением.

В некоторых случаях можно непосредственно получить объект изображения Graphics. Например, при создании элемента управления Windows Forms, можно переопределить метод OnPaint для доступа к объекту Graphics изображения элемента управления.

Класс Graphics имеет множество методов для рисования и работы с изображениями. Ниже перечислены некоторые из часто используемых методов:

Методы для рисования линий: DrawArc, DrawBezier, DrawEllipse, DrawImage, DrawLine, DrawPolygon, DrawRectangle и DrawString.

Методы для заполнения фигур: FillClosedCurve, FillEllipse, FillPath, FillPolygon и FillRectangle.

Метод для очистки поверхности рисования: Clear.

Метод для создания нового объекта Graphics из изображения: FromImage.

На листинге 1 приведен пример с демонстрацией графических возможностей GDI на C#.

Листинг 1. Использование графики GDI

```
using System;
using System.Windows.Forms;
using System.Drawing;

class SimpleShapeMaker : Form
{
    // Метод-конструктор  класса
    public SimpleShapeMaker()
    {
        // Меняем цвет фона формы на белый

        this.BackColor = Color.White;

        // Добавляем на форму кнопку и
        //привязываем ее к обработчику событий

        Button button1 = new Button();
        button1.Text = "click me";
        button1.Location = new Point(110, 10);
        button1.BackColor = Color.SteelBlue;
        button1.Click += new
System.EventHandler(button1_Click);
        this.Controls.Add(button1);
    }
```

```
    // Обработчик события, срабатывающий при нажатии
кнопки
    void button1_Click(object o, System.EventArgs e)
    {
        // Выполнение описанного нами метода
        DrawSomeShapes();
    }
    // Метод для отрисовки на поверхности формы
     //нескольких фигур
    void DrawSomeShapes()
    {
        // Подготовка области рисования на форме
        Graphics g = this.CreateGraphics();
        // Подготавливаем перо, рисующее красную линию
       //толщиной 3 пикселя
        Pen redPen = new Pen(Color.Red, 3);
        // С помощью пера рисуем прямую линию,
       //прямоугольник и овал
        g.DrawLine(redPen, 140, 170, 140, 230);
        g.DrawRectangle(redPen, 50, 60, 50, 60);
        g.DrawEllipse(redPen, 150, 100, 100, 60);
        // Очистка
        g.Dispose();
    }
    static void Main()
    {
        // Запускаем новый экземпляр приложения
        //Windows Forms при помощи вышеописанного
класса
        Application.Run(new SimpleShapeMaker());
    }
}
```

Результаты работы показаны на рис. 6.

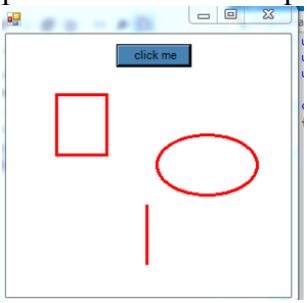


Рис. 6. Результат работы приложения

В таблице представим некоторые графические классы и структуры, а также их описание.

Графические классы и структуры

Класс или структура	Описание
System.Drawing.Bitmap	Инкапсулирует растровый рисунок GDI+, содержащий данные точек и атрибуты графического изображения. Объект Bitmap используется для работы с изображениями, определяемыми данными точками.
System.Drawing.Brushes	Определяет кисти для всех стандартных цветов.
System.Drawing.Color	Представляет цвет ARGB
System.Drawing.Font	Определяет формат текста, включая начертание шрифта, его размер и атрибуты стиля.
System.Drawing.Pen	Определяет объект, используемый для рисования прямых линий и кривых.
System.Drawing.Pens	Определяет перья для всех стандартных цветов.
System.Drawing.Point	Представляет упорядоченную пару целых чисел, определяющую точку на двумерной плоскости.
System.Drawing.Rectangle	Содержит набор из четырех целых чисел, определяющих расположение и размер прямоугольника. Расширенные функции для работы с областью предусмотрены в объекте Region.

Продолжение таблицы

System.Drawing.SolidBrush	Определяет кисть одного цвета. Кисти используются для заливки графических фигур, таких как прямоугольник, эллипсы, круги, многоугольники и траектории
System.Drawing.TextureBrush	Каждое свойство класса TextureBrush является объектом Brush, использующим изображение для заливки внутренней части фигуры

Задание на самостоятельную работу:

Разработать GUI проект, где будут построены графики функций y=sin(x) и y=cos(x). Диапазон периодов определить с помощью элементов управления (Button, TextBox и т.д.). Отобразить координатные оси x и y.

ЛАБОРАТОРНАЯ РАБОТА № 7
ИНТЕРФЕЙСНЫЙ ЭЛЕМЕНТ DATAGRIDVIEW

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

Задачи и требования к выполнению:

1.Изучить интерфейсный элемент DataGridView, его возможности, и рассмотреть их на примере.

Теоретические сведения

Элемент управления DataGridView предоставляет настраиваемую таблицу для отображения данных (рис. 7).

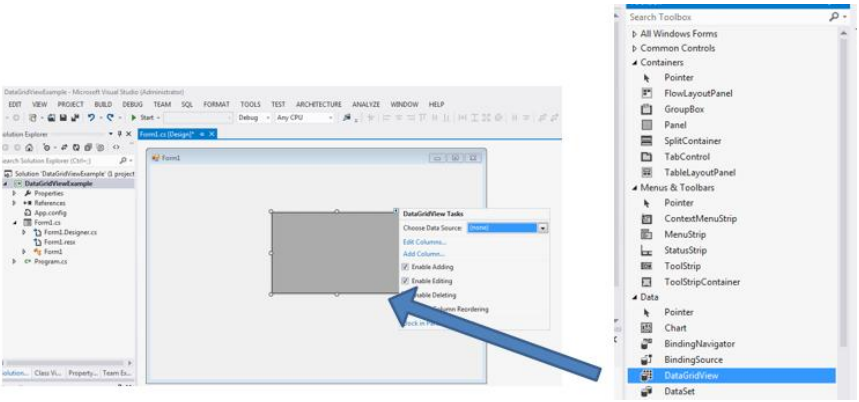


Рис. 7. Элемент управления DataGridView

Класс DataGridView допускает настройку ячеек, строк, столбцов и границ с помощью использования свойств, таких как DefaultCellStyle, ColumnHeadersDefaultCellStyle, CellBorderStyle и GridColor.

Элемент управления DataGridView предоставляет мощный и гибкий способ отображения данных в табличном формате. Элемент управления DataGridView можно использовать для представления в режиме только чтения небольших объектов данных; можно расширить этот элемент для представления крупных объемов данных в режиме редактирования.

Функциональные возможности элемента управления DataGridView можно расширить несколькими способами, чтобы реализовать пользовательское поведение в приложениях. Например, можно программно задать собственные алгоритмы сортировки, а также создать собственные типы ячеек. Внешний вид элемента управления DataGridView можно настроить, задав несколько свойств. В качестве источника данных могут использоваться различные типы хранилищ данных. Кроме того, элемент управления DataGridView может работать без связанных источников данных.

На листинге 2 показана работа с элементом управления DataGridView.

Листинг 2. Вывод значений массива в DataGridView (Form.cs)

```
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace DataGridViewExample
{
    public partial class Form1 : Form
    {
        const int vElements = 2;
        const int hElements = 3;
        static object [,] MyArray;
        public Form1()
        {
            InitializeComponent();
            MyArray = new object[vElements, hElements]
                            { { 1, 2, 3 }, { 4, 5, 6 }
};
          //object [,] MyArray ={ { 1, 1, 1 }, { 2, 2,
2 } };
        }
        private void button1_Click(object sender,
EventArgs e)
        {
            dataGridView1.RowCount = vElements;
            dataGridView1.ColumnCount = hElements;
            for (int i = 0; i < vElements; ++i)
               for (int j = 0; j < hElements; ++j)
                  dataGridView1.Rows[i].Cells[j].Value
= MyArray[i, j];
        }
    }
}
```

После сборки и запуска приложения на экран будет выведена таблица с значениями, как показано на рис. 8.

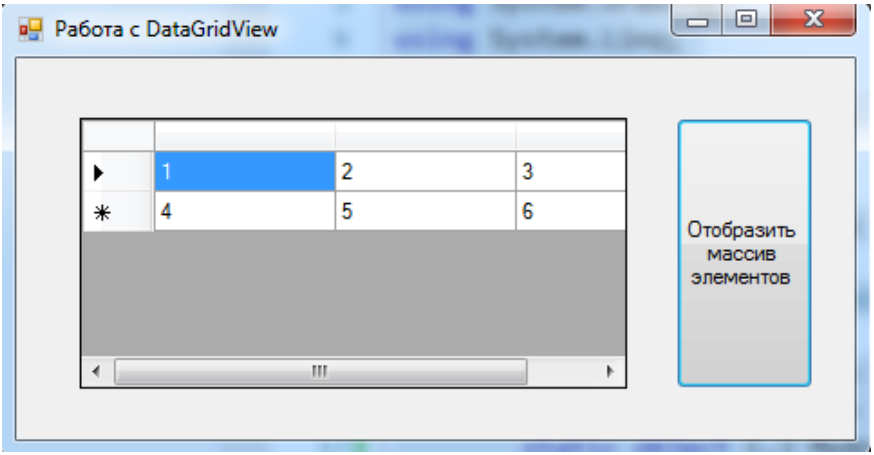


Рис. 8. Результат выполнения программы

Некоторые возможности по работе с dataGridView в Visual Studio:

1) Можно добавлять столбцы с помощью: dataGridView columns

Пример: this.dataGridView1.Columns.Add(Name, Text);

Name - имя по которому будет упоминаться столбец.

Text - Текст для заголовка столбца

2) Добавлять строки с помощью: dataGridView rows

Пример: this.dataGridView1.Rows.Add();

3) Очищать строки: this.dataGridView1.Rows.Clear();

4) Очищать столбцы this.dataGridView1.Columns.Clear();

5)Удалятьстолбцы: this.dataGridView1.Columns.Remove (Name);

Name - имя столбца, который нужно удалить

6) Удалять строки: this.dataGridView1.Rows.Remove (Rows);

Задания на самостоятельную работу:

1.Разработать электронную таблицу умножения размером 16x16, используя элемент управления DataGridView (рис. 9).

×	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32
3	3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48
4	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64
5	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80
6	6	12	18	24	30	36	42	48	54	60	66	72	78	84	90	96
7	7	14	21	28	35	42	49	56	63	70	77	84	91	98	105	112
8	8	16	24	32	40	48	56	64	72	80	88	96	104	112	120	128
9	9	18	27	36	45	54	63	72	81	90	99	108	117	126	135	144
10	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160
11	11	22	33	44	55	66	77	88	99	110	121	132	143	154	165	176
12	12	24	36	48	60	72	84	96	108	120	132	144	156	168	180	192
13	13	26	39	52	65	78	91	104	117	130	143	156	169	182	195	208
14	14	28	42	56	70	84	108	112	126	140	154	168	182	196	210	224
15	15	30	45	60	75	90	105	120	135	150	165	180	195	210	225	240
16	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	256

Рис. 9. Пример результата выполнения задания

ЛАБОРАТОРНАЯ РАБОТА № 8
РАБОТА С ДИНАМИЧЕСКИМИ БИБЛИОТЕКАМИ WINDOWS (.DLL)

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

Задачи и требования к выполнению:

1.Познакомиться с динамическими библиотеками Windows.

2.Изучить работу с библиотеками.

Теоретические сведения

Динамические библиотеки представляют собой готовые решения в виде классов и функций, которые используют приложения. Основным их преимуществом является загрузка в память ЭВМ лишь в тот момент, когда требуется вызов необходимой функции или метода класса, что позволяет эффективно использовать память компьютера.

Для создания библиотеки в Visual Studio есть специальный тип проекта, «библиотека классов».

Библиотеки DLL используются для переноса какой-либо программы (например, интерпретатора языка программирования) из приложения в приложение. Не обязательно писать библиотеки на языке C#, подойдет любой другой язык, интегрированный с платформой .NET.

По умолчанию библиотеки вызываются, если они находятся в одной из папок Windows (рис. 10).

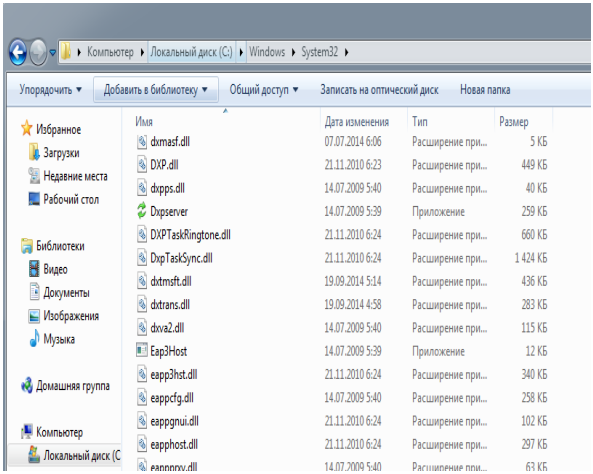


Рис. 10. Динамически-подключаемые библиотеки (.dll)

На листинге 3 представлен фрагмент кода динамической библиотеки.

Листинг 3. Пример проекта .dll

```
using System;
namespace create_console_dll
{
    public class Address
    {
        private string City;
        private string Street;
        private string House;
     private string Flat;
        public String CreateString()
```

```
        {
            String temp = "394000 " + City + '\u0020'
+ Street + ' ' + House + '-' + Flat;
            return temp;
        }
        public Address()
        {
            City = "Voronezh";
        }
        public string MyStreet
        {
            get { return Street; }
            set { Street = value; }
        }
        public string MyHouse
        {
            get { return House; }
            set { House = value; }
        }
        public string MyFlat
        {
            get { return Flat; }
            set { Flat = value; }
        }
    }
}
```

Одним из условий сборки проекта dll является указание в свойствах проекта, что решение собирается как библиотека (рис. 11).

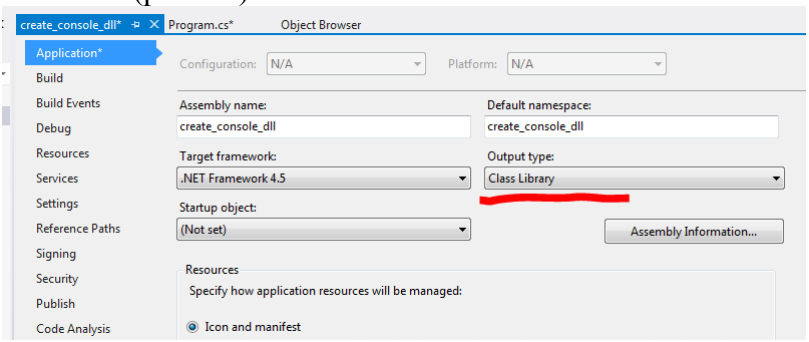


Рис. 11 Сборка проекта

Ниже приведен алгоритм подключения библиотек .dll к приложению:

1.Компиляция:

-необходимо выбрать конфигурацию «Release» и нажать клавишу F6.

В директории Visual Studio %VERSION%Projects%PROJECTNAME%binRelease будет лежать библиотека.

2.Подключение:

-создать проект типа Windows Form. Выполнить команды “References -> Add reference”;

-в окне «Add reference» выбрать «Обзор» и указать путь к вашей библиотеке;

-выделить библиотеку и нажать «Добавить».

После этого в списке «References» должна появиться библиотека.

3.Обращение к объектам в библиотеке

Если не указано пространство имен в библиотеке, можно обращаться к классам следующим образом:

var $VarName$ = new $ClassName$;

Или пользоваться конструкцией вида:

var $VarName$ = new $namespace$.$ClassName$;

На рис. 12 показано, как в проекте должно выглядеть подключение dll.

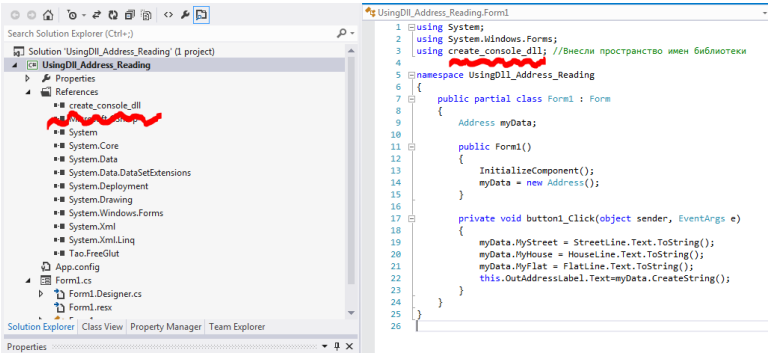


Рис. 12 Реализация проекта с формой и подключенной динамической библиотекой

Реализация проекта с формой и подключенной динамической библиотекой.

Листинг 4. Пример приложения

```
using System;
using System.Windows.Forms;
using create_console_dll; \\ Внесли пространство имен
библиотеки

namespace UsingDll_Address_Reading
{
   public partial class Form1 : Form
  {
       Address myData;
       public Form1()
      {
           InitializeComponent();
           myData = new Address();
     }
    private void button1_Click(object sender,
EventArgs e)
   {
         myData.MyStreet = StreetLine.Text.ToString();
         myData.MyHouse = HouseLine.Text.ToString();
         myData.MyFlat = FlatLine.Text.ToString();
         this.OutAddressLabel1.Text=
myData.CreateString();
    }
  }
}
```

Задания на самостоятельную работу:

1. Построить библиотеку (.dll), в которой производится поиск простых чисел в диапазоне от 1 до 1000, а затем в главной форме приложения подготовить GUI интерфейс для их вывода в список по запросу пользователя.

ЛАБОРАТОРНАЯ РАБОТА № 9
РАБОТА С БАЗОЙ ДАННЫХ НА C#

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

Задачи и требования к выполнению:

1.Изучить работу с базой данных.

Теоретические сведения

База данных — организованная в соответствии с определёнными правилами и поддерживаемая в памяти компьютера совокупность данных, характеризующая актуальное состояние некоторой предметной области и используемая для удовлетворения информационных потребностей пользователей (рис. 13).

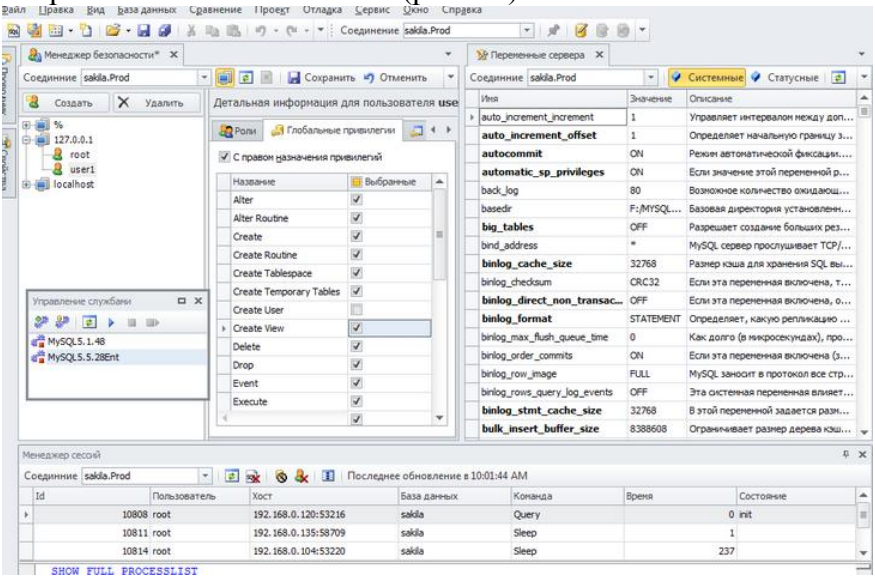


Рис. 13. Список баз данных

Чтобы начать работать с базой данных, нужно сначала внести ее в проект. На рис.14 показано первое действие по подключению локальной базы данных.

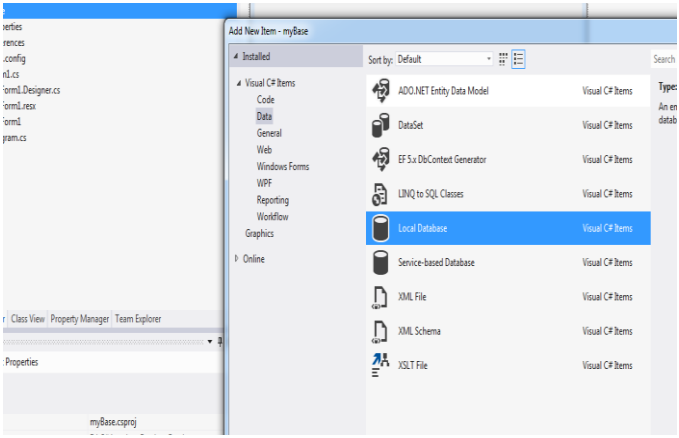


Рис. 14. Внесение БД в проект

Далее создается таблица, где определяются колонки и их свойства, с которыми надо работать. На рис. 15 показан пример создания таблицы, где Age (возраст) - колонка, Data Type - тип вносимых данных, то есть целое число (int), допустимая длина и так далее.

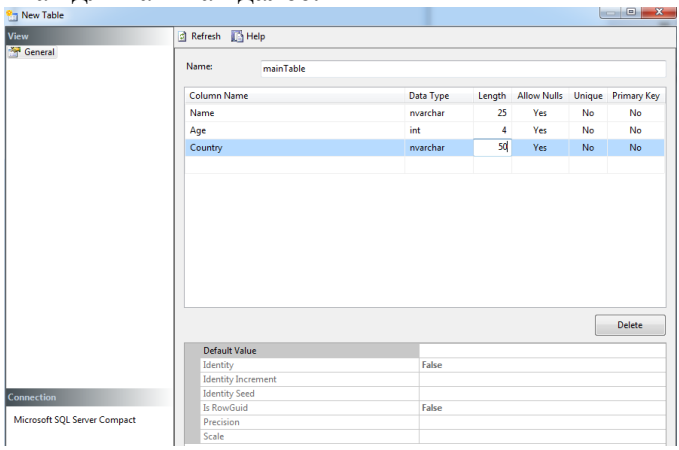


Рис. 15. Создание таблицы

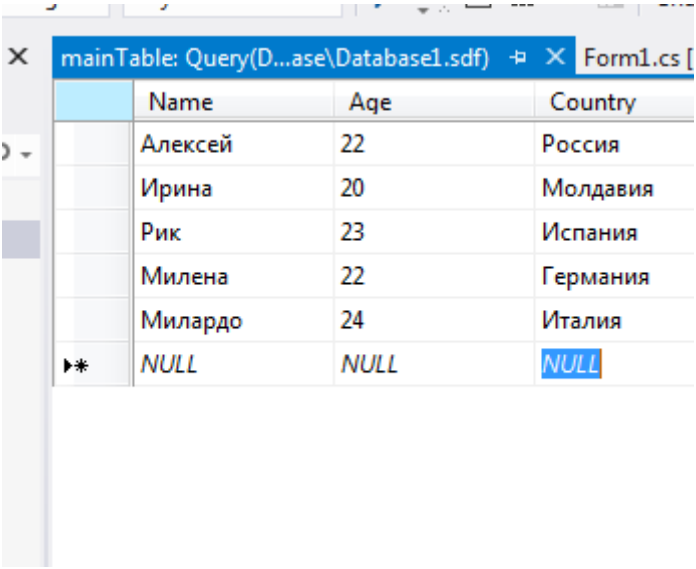


Рис. 16. Внесение данных в таблицу

После создания таблицы, необходимо занести в нее данные, что и показано на рисунке 16.

Чтобы соединить таблицу с формой, нужно разместить компоненты на ней, далее следовать пошагово согласно рекомендациям мастера подключения БД (рис. 17).

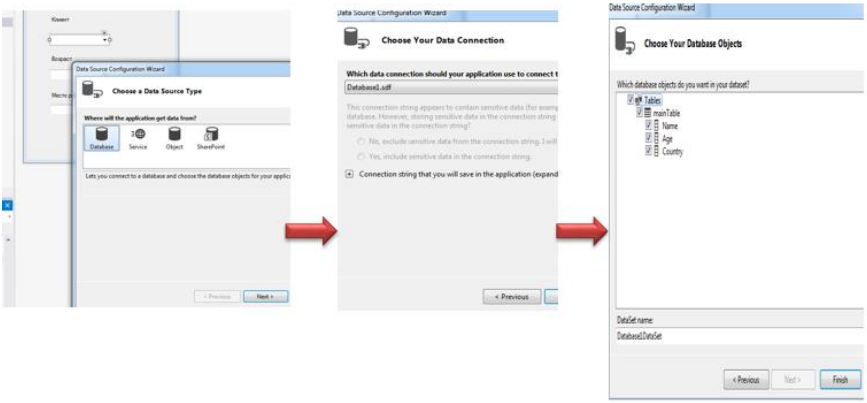


Рис. 17. Связь заданного элемента управления на форме с табличными данными в базе

Задания на самостоятельную работу:

1.Используя элемент DataGridView, доработать приложение:

-производить поиск по возрасту в заданном диапазоне;

-отмечать совпадения цветом в таблице;

-предусмотреть наполнение и редактирование таблицы в БД.

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 № 5-9 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. 1,5. «С»

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