ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
СЕВЕРО-КАВКАЗСКИЙ ФИЛИАЛ
ОРДЕНА ТРУДОВОГО КРАСНОГО ЗНАМЕНИ
ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО
БЮДЖЕТНОГО ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО
ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
СВЯЗИ И ИНФОРМАТИКИ»

КАФЕДРА ИНФОРМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

П.В. Лобзенко

МУ к ЛР с 1 до 6

по дисциплине

ПРОЕКТИРОВАНИЕ КЛИЕНТ-СЕРВЕРНЫХ ПРИЛОЖЕНИЙ

Ростов-на-Дону

2019 г.

МУ к ЛР с 1 до 6

по дисциплине

ПРОЕКТИРОВАНИЕ КЛИЕНТ-СЕРВЕРНЫХ ПРИЛОЖЕНИЙ

Для студентов очной и заочной форм обучения

Направление подготовки - **09.03.01 «**Информатика и вычислительная техника»

Составитель: П.В. Лобзенко, доцент кафедры ИВТ

Рассмотрено и одобрено
на заседании кафедры ИВТ Протокол от
«26» августа 2019 г. № 1

МУ к ЛР 1

Исследование web - приложений как клиент-серверных структур

2.1.1 Цели занятия

Выработать навыки составления программ в виде WEB приложений, исследуя процесс их проектирования.

2.1.2 Теоретические основы и пример выполнения

В состав клиент-серверных приложений, как составного программного обеспечения (ПО), очевидно, входят серверная и клиентская части. Как правило, обе эти составляющие приложения разрабатываются как интерфейсные продукты, т.е. содержащие специальные модули (блоки) для обеспечения связи человека и вычислительной среды. От этого зависит, в конечном итоге, насколько простым или сложным будет использование приложения, а также его оптимальное функционирование.

Поэтому, важно научиться составлять удобные в использовании и функционально оптимальные клиент-серверные продукты.

Составление web-приложения выполняется в среде разработки Visual Studio на языке программирования C# [7, 8].

В настоящее время, C# включен в семейство продуктов этой среды разработки ПО на различных языках высокого уровня – мощной интегрированной среды разработки приложений различного уровня сложности и назначения. В этом пакете используется единая интегрированная среда разработки (IDE), состоящая из нескольких элементов: строки меню, панели инструментов, различных закрепленных или автоматически скрываемых окон инструментов в левой, нижней или правой областях, а также области редакторов. Набор доступных окон инструментов, меню и панелей инструментов зависит от типа проекта или файла, в котором выполняется разработка (рисунок 2.3).

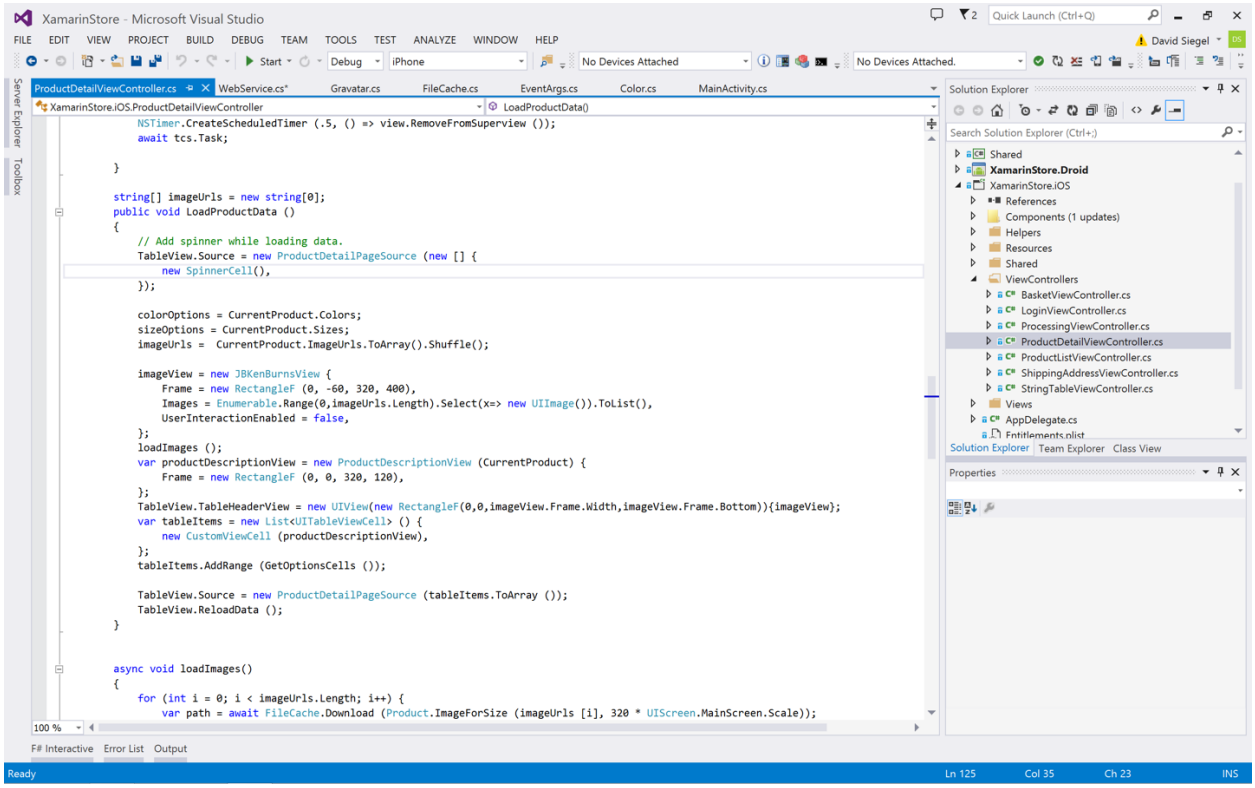

Рисунок 2.3 – Окно среды разработки Visual Studio

В этой интерактивной среде разработки программ можно создавать как приложения Windows Forms, так и web приложения в виде browser application (рис.2.4).

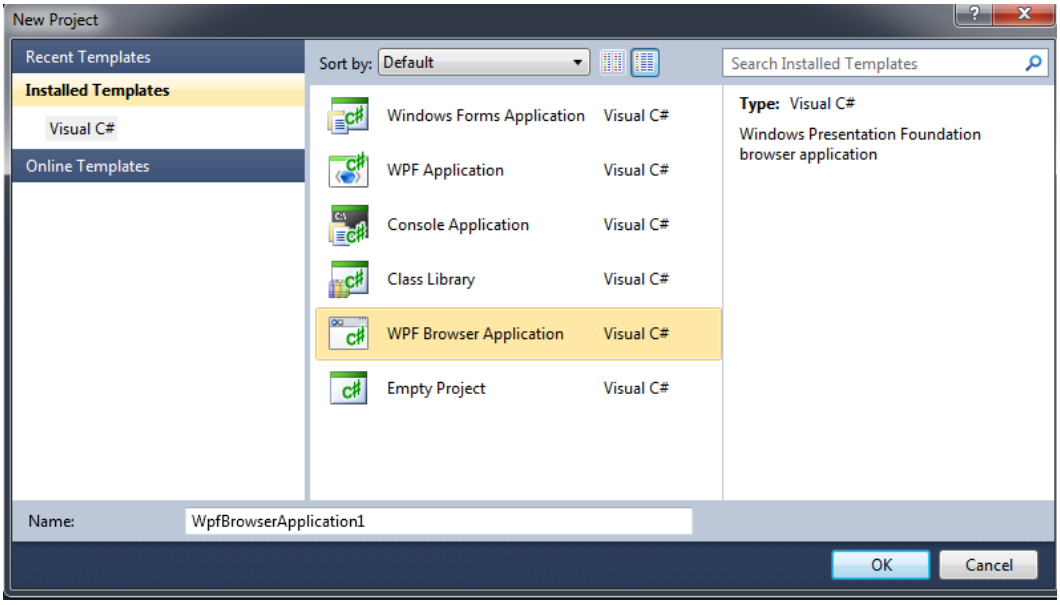

Рисунок 2.4- Создание проекта для web - приложения

Среда разработки автоматически формирует весь необходимый код во время «перетаскивания» на предоставляемую форму нужных элементов управления.

Далее, написание кода приложения очень похоже на работу в стандартном проекте Windows Form, когда интерфейс приложения составляется «перетаскиванием» элементов управления на исходную форму.

2.1.2.1 Основные элементы графического интерфейса пользователя

Для создания графических интерфейсов web - приложений используется обширная библиотека Form языка C#. Принцип составления оконного интерфейса, как указывалось ранее, прост - создается пустая форма, которая наследует (использует) библиотеку готовых решений Form. Далее, перетаскивая на эту форму необходимые элементы управления и отображения составляется клиентская часть для решения задачи, указанной на практическом занятии.

Элементов управления в указанной библиотеке множество, рассмотрим только часто употребляемые и необходимые для решения поставленных на занятии задач.

Основными элементами форм являются:

- label – метка для отображения справочной информации;
- textBox – текстовое поле для вводимых и выводимых данных;
- button – кнопка для исполнения требуемых действий.

Приведем пример проектирования web - приложения и покажем как программируются указанные элементы интерфейса.

2.1.2.2 Пример выполнения задания

Задание: составить web – приложение для отображения обучающего контента.

Так выглядит стартовая форма приложения с размещенными на ней необходимыми кнопками, текстовыми полями и тому подобное (рисунок 2.5).

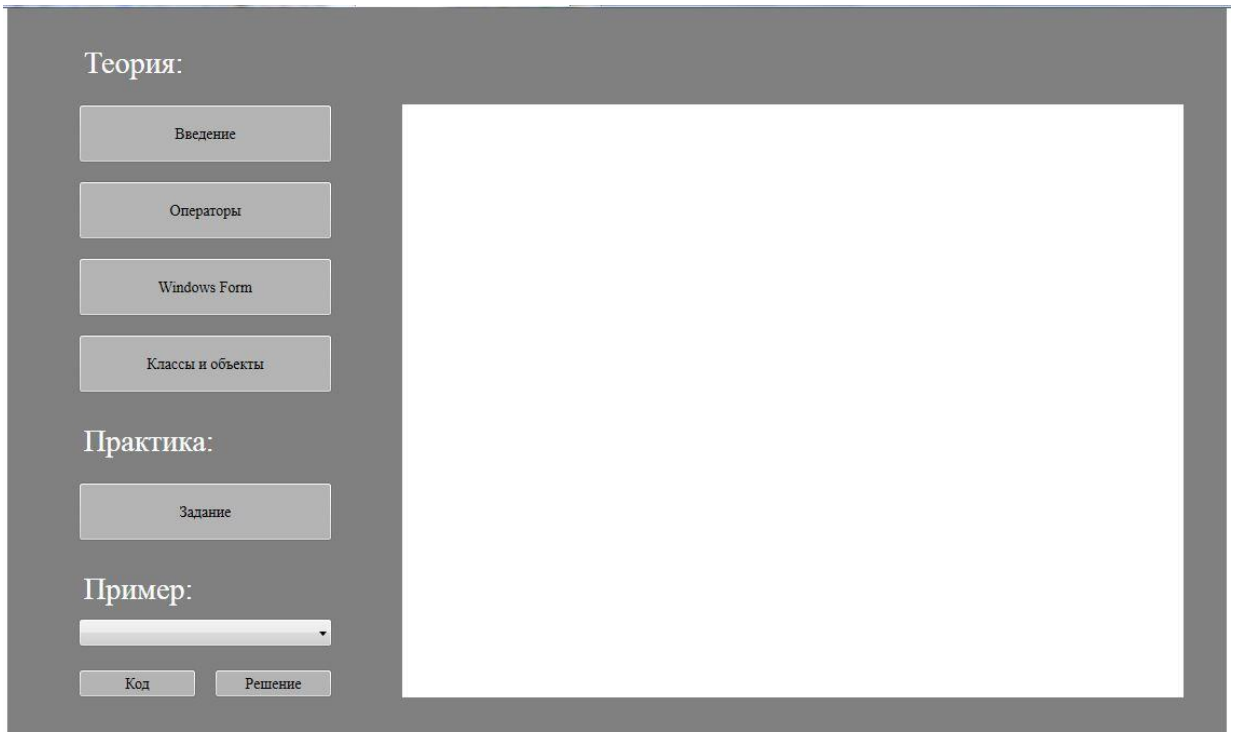

Рисунок 2.5- Стартовое окно приложения

Ее код в xaml представлении задается в виде таблицы заданных размеров (листинг 2.1).

Листинг 2.1- Описание формы

```
<Grid Height="684" Width="1130" Background="Gray">
```

Соответствующий этой странице код в C# показан в листинге 2.2.

Листинг 2.2- Программный код формы 1

```
// Необходимые автоматически подключаемые библиотеки
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
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;

// Пространство имен проекта
namespace WpfBrowserApplication1
{
    // Наследование класса Page со всеми элементами управления
    public partial class Page1 : Page
    {
        // Конструктор стартовой формы, инициализирующий ее компоненты
          public Page1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, RoutedEventArgs e)
        {
           string puti = "I:\\Pril\\Lex";
           webBrowser1.Navigate(puti + "\\" +"1.mht");
         }

        private void button2_Click(object sender, RoutedEventArgs e)
        {
            string puti = "I:\\Pril\\Lex";
            webBrowser1.Navigate(puti + "\\" + "2.mht");
        }

        private void button3_Click(object sender, RoutedEventArgs e)
        {
            string puti = "I:\\Pril\\Lex";
```

```
            webBrowser1.Navigate(puti + "\\" + "3.mht");
        }

        private void button4_Click(object sender, RoutedEven-
tArgs e)
        {
            string puti = "I:\\Pril\\Lex";
            webBrowser1.Navigate(puti + "\\" + "4.mht");
        }

        private void button5_Click(object sender, RoutedEven-
tArgs e)
        {
            string puti = "I:\\Pril\\Lex";
            webBrowser1.Navigate(puti + "\\" + "p1.mht");
        }

        private void button6_Click(object sender, RoutedEven-
tArgs e)
        {
            if (comboBox1.SelectedIndex == 0)
            {
                string puti = "I:\\Pril\\Lex";
                webBrowser1.Navigate(puti     +     "\\"     +
"Lab1.mht");
            }

            if (comboBox1.SelectedIndex == 1)
            {
                string puti = "I:\\Pril\\Lex";
                webBrowser1.Navigate(puti     +     "\\"     +
"Lab2.mht");
            }

            if (comboBox1.SelectedIndex == 2)
```

```
                {
                    string puti = "I:\\Pril\\Lex";
                    webBrowser1.Navigate(puti + "\\" + "Lab3.mht");
                }
            }

            private void button7_Click(object sender, RoutedEventArgs e)
            {
                if (comboBox1.SelectedIndex == 0)
                {
                    NavigationService.Navigate(new Uri("/Page2.xaml", UriKind.Relative));
                }

                if (comboBox1.SelectedIndex == 1)
                {
                    NavigationService.Navigate(new Uri("/Page3.xaml", UriKind.Relative));
                }

                if (comboBox1.SelectedIndex == 2)
                {
                    NavigationService.Navigate(new Uri("/Page4.xaml", UriKind.Relative));
                }
            }
        }
    }
```

Из листинга видно, что в самом начале автоматически создается стартовая страница Page1 со всеми необходимыми библиотеками, которая наследует класс Page, в котором есть все необходимые элементы управления. Конструктор этой

страницы при обращении к ней инициализирует все размещенные на ней компоненты (см. комментарии в листинге 2.2).

На этой странице размещены кнопки управления (компонент button), «выпадающие» меню для выбора одного вида занятий из списка (элемент comboBox), метка для помещения заглавий разделов страницы (элемент label), а также окно для просмотра выбранного контента (webBrowser).

2.1.2.3 Разработка страниц раздела «Теория»

Срабатывают управляющие элементы, описанные выше, по клику на них левой клавишей мышки. При этом включаются их обработчики событий. Например, для отображения материала раздела «Введение» после нажатия на эту кнопку на стартовой странице исполняется следующий код обработчика событий этой кнопки (листинг 2.3).

Листинг 2.3- Код обработчика события «Нажатие» кнопки «Введение»

```
        private void button1_Click(object sender, RoutedEventArgs
e)
            {
               string puti = "I:\\Pril\\Lex";
               webBrowser1.Navigate(puti + "\\" +"1.mht");
             }
```

Видно, что строковой переменной puti присваивается путь к соответствующему файлу. Далее компоненту этой страницы webBrowser1, который нужен для отображения выбираемого контента на ней, присваивается содержимое файла, который находит и возвращает метод Navigate() согласно пути, который храниться в переменной puti.

Аналогично обрабатываются события использования других компонентов страницы.

Так, чтобы отобразить нужную теорию необходимо будет нажать соответствующую кнопку на стартовой форме.

Т.е. при нажатии кнопки идет обращение к файлу, берется из него информация и выводится на экран (рисунки 2.6, 2.7).

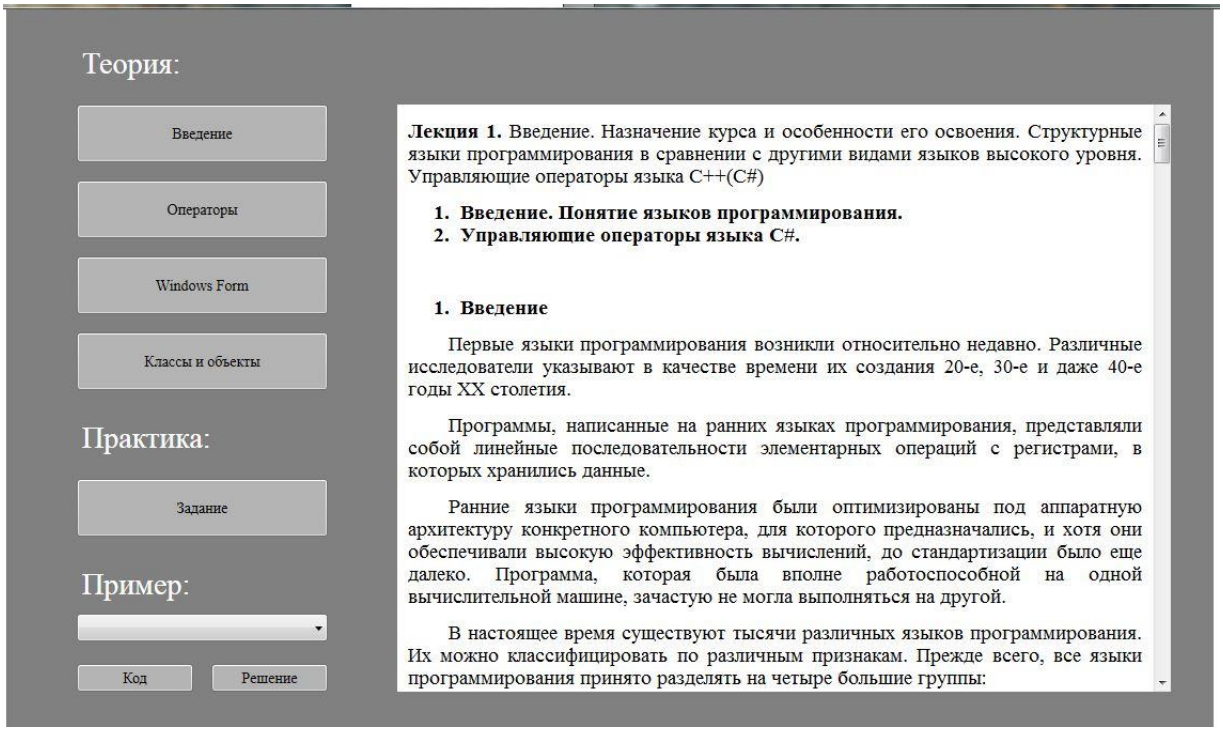

Рисунок 2.6 – Отображение контента в области «Теория»

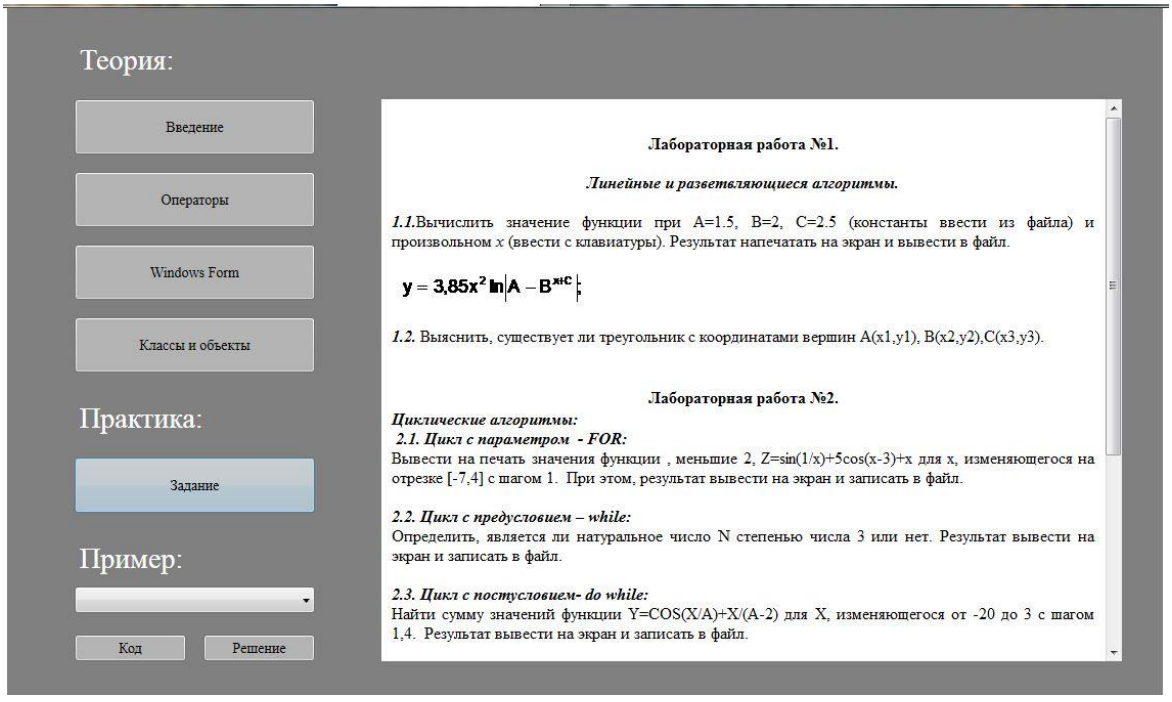

Рисунок 2.7 – Отображение контента в области «Практика»

Как было сказано выше, файлы контента должны быть представлены в формате html и размещены в папке Lex по указанному пути.

Теперь нужно разработать следующие страницы приложения.

2.1.2.4 Разработка страниц раздела «Практика»

Этот раздел стартовой страницы нужен для реализации практической части курса. Он содержит кнопку «Задание», по нажатию на которую выводится для прочтения текст задания для практических занятий и лабораторных работ (листинг 2.4).

Листинг 2.4- Код обработчика события кнопки «Задание»

```
        private void button5_Click(object sender,
RoutedEventArgs e)
        {
            string pup = "I:\\ Pril\\Lex";
            webBrowser1.Navigate(pup + "\\" + "p1.mht");
        }
```

В разделе «Пример» находится comboBox1 для того, чтобы можно было выбрать нужную лабораторную работу, которые задаются списком в его свойстве Select item (рисунок 2.8).

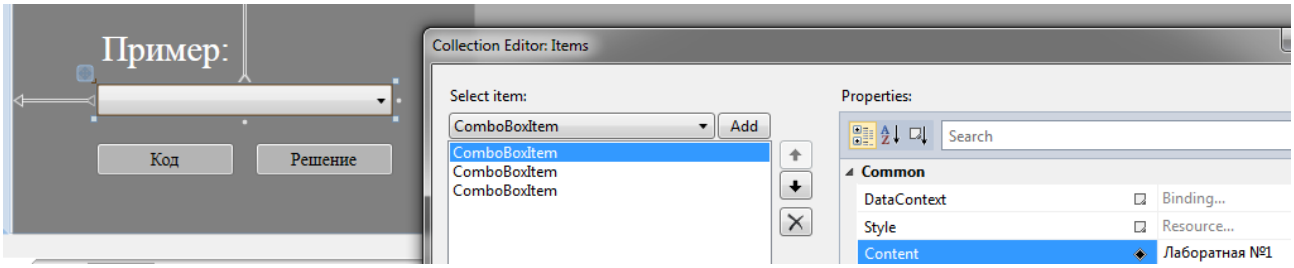

Рисунок 2.8 – Определение списка лабораторных работ

После выбора номера лабораторной работы по нажатию кнопки «Код» должен быть переход на страницу этой лабораторной работы (листинг 2.5).

Листинг 2.5- Код обработчика события кнопки «Код»

```
        private void button6_Click(object sender,
RoutedEventArgs e)
        {
```

```
        if (comboBox1.SelectedIndex == 0)
        {
            string pup = "I:\\ Pril\\Lex";
            webBrowser1.Navigate(pup + "\\" + "Lab1.mht");
        }

        if (comboBox1.SelectedIndex == 1)
        {
            string pup = "I:\\ Pril\\Lex";
            webBrowser1.Navigate(pup + "\\" + "Lab2.mht");
        }

        if (comboBox1.SelectedIndex == 2)
        {
            string pup = "I:\\ Pril\\Lex";
            webBrowser1.Navigate(pup + "\\" + "Lab3.mht");
        }
    }
```

Видно, что выбор нужной лабораторной работы выполняется через удовлетворение соответствующему условию в приведенном обработчике событий. При этом, в webBrowser1 загружается текст программы соответствующей задачи.

Кнопка «Решение» должна запускать еще одну страницу для выбранной лабораторной работы, где можно ввести исходные данные и получить решение в интерактивном режиме (листинг 2.6).

Листинг 2.6- Код обработчика события кнопки «Решение»

```
        private void button7_Click(object sender,
RoutedEventArgs e)
        {
            if (comboBox1.SelectedIndex == 0)
            {
                NavigationService.Navigate(new
Uri("/Page2.xaml", UriKind.Relative));
```

```
                }

                if (comboBox1.SelectedIndex == 1)
                {
                    NavigationService.Navigate(new
Uri("/Page3.xaml", UriKind.Relative));
                }
                if (comboBox1.SelectedIndex == 2)
                {
                    NavigationService.Navigate(new
Uri("/Page4.xaml", UriKind.Relative));
                }
            }
```

Из этого обработчика событий видно, что по выполнению соответствующего условия метод Navigate(new Uri(,)) запускает соответствующую страницу с новым адресом из списка возможных страниц (Page2.xaml-Page4.xaml).

После запуска одной из лабораторных работ, управление должно передаваться этой странице (рисунок 2.9).

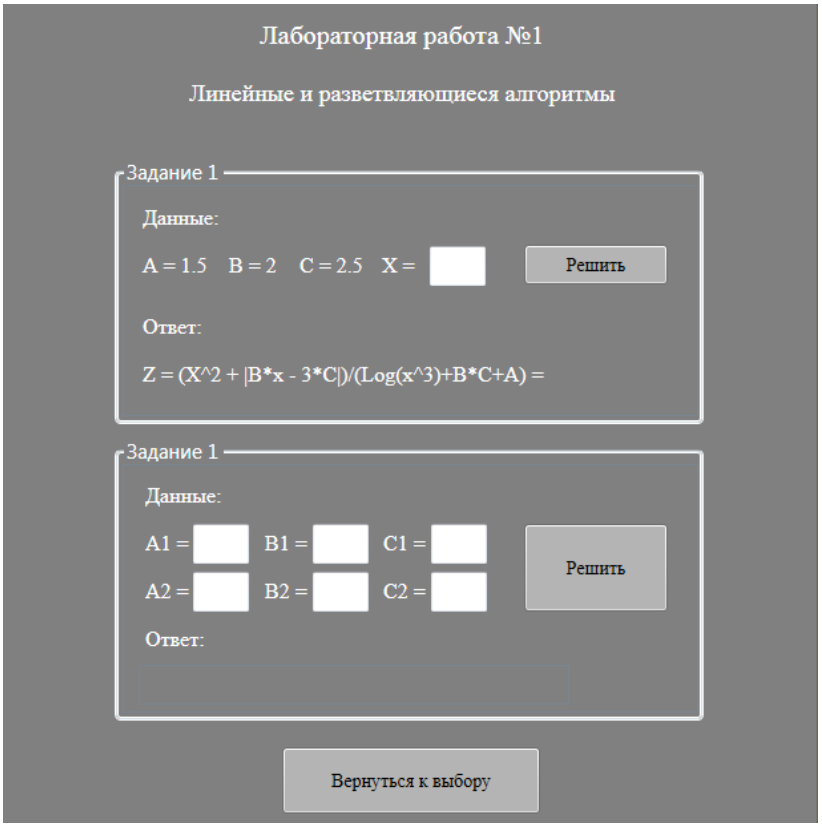

Рисунок 2.9 – Страница выбранной лабораторной работы

На этих страницах есть нужные элементы управления, чтобы можно было подставив исходные данные, получить решение задачи из лабораторной работы. Вернуться на стартовую страницу приложения можно по соответствующей кнопке (листинг 2.7).

Листинг 2.7- Код обработчика события кнопки «Вернуться к выбору»

```
            private void button1_Click(object sender,
RoutedEventArgs e)
            {
                NavigationService.Navigate(new Uri("/Page1.xaml",
UriKind.Relative));
            }
```

2.1.2.6 Подготовка контента приложения

Для того, чтобы можно было отображать обучающие материалы в оболочке web - приложения они должны быть представлены в html формате, с которым наиболее простым образом могут работать web приложения.

Подготовка контента идет по следующему алгоритму:

- проставляются необходимые гиперссылки в файлах – исходниках, которые открываются в редакторе MS Word;
- файл сохраняется в указанной папке (Lex) в формате html.

Текст всей программы представлен в приложении А.

2.1.3 Порядок проведения исследований

Алгоритм выполнения заданий ЛР следующий:

- выбрать 5 задач по следующему правилу: номер по журналу- первая задача; номер каждой последующей задачи определяется прибавлением цифры 3 к номеру первой задачи, который только что вычислили (если достигнуто окончание списка вариантов задач, то перейти в его начало);

- составить методы (функции) решения всех задач, поместить их в класс Form1 и снабдить пользовательским интерфейсом в виде web – приложения;
- исследовать запуск приложения через Internet Explorer, разместив приложение на Local host. Сделать выводы о достоинствах и недостатках web – приложений как определенного вида клиент – серверных структур;
- оформить отчет для всего приложения вцелом, включив в него задание, блок-схему алгоритма (в электронном виде), текст программы и skrinshert результата выполнения каждой задачи и представить его на проверку.

2.1.4 Варианты заданий

1. Найти среднее арифметическое положительных чисел, введенных с клавиатуры. Всего ввести N различных чисел.
2. Ввести с клавиатуры N чисел. Найти сумму тех из них, которые принадлежат интервалу (2;9).
3. Для N введенных с клавиатуры чисел найти сумму положительных кратных трем.
4. Для арифметической прогрессии 4, 9, 14, 19... найти первые n членов этой прогрессии.
5. Найти сумму отрицательных значений функции Z=sin(5-x)/cos(x-2) для x, изменяющегося на отрезке [-5,12] с шагом 1.
6. Найти среднее арифметическое отрицательных чисел, введенных с клавиатуры. Всего ввести N различных чисел.
7. Найти среднее арифметическое чисел, принадлежащих отрезку [2,184], кратных 2 и введенных с клавиатуры. Всего ввести N различных чисел.
8. Найти сумму значений функции, больших 2 Z=sin(1/x)+5cos(1/(x-3))+x для x, изменяющегося на отрезке [-3,8] с шагом 1.
9. Найти n членов последовательности $x_1 = x_2 = x_3 = 1$; $x_k = x_{k-1} + x_{k-3}$.

10. Вычислить последовательность N чисел $A_0 = x$, $A_1 = 2$, $A_k = A_{k-1} - A_{k-2}$.
11. Для x1 =0,3 и x2 =-0,3 найти xk =k+sin(xk-2) для k, изменяющегося следующим образом: k = 3,4,...,14.
12. Составить таблицу перевода дюймов в сантиметры для расстояний от 1 до 13 дюймов с шагом 1.
13. Вывести на печать значения функции, меньшие 2, Z=sin(1/x)+5cos(x-3)+x для x, изменяющегося на отрезке [-7,4] с шагом 1.
14. Напечатать таблицу значений функции Y = tg(x/b)+x/(b-2) для x, изменяющегося от 0 до 10 с шагом 1 (b - произвольное число).
15. Вычислить N -ый член последовательности $x_k = x_{k-2} - x_{k-1}$, $x_0 = 2,4$ $x_1 = 3,8$.
16. Составить таблицу перевода суток (от 1 до 7) в часы, минуты, секунды.
17. Вычислить N-ый член последовательности $x_k = x_{k-1} + (2/3)x_{k-2} + 1$, $x_1 = -1$, $x_2 = 1,38$.
18. Напечатать значения функции z = 1/(x-2)+1/(x-5)+ln(12,8-X) для x, изменяющегося на отрезке [-4,14] с шагом 1,23.
19. Вывести на печать отрицательные значения функции z=sin(5-x)/cos(x-2) для x, изменяющегося на отрезке [-6,13] с шагом 1 (учесть область допустимых значений функции).
20. Из N введенных с клавиатуры чисел напечатать кратные 3 и меньшие 58.
21. Ввести с клавиатуры N чисел. Напечатать те из них, которые принадлежат интервалу (1,11) и являются четными.
22. Из N введенных с клавиатуры чисел напечатать положительные, кратные трем.
23. Вывести на печать значения функции z = sin(x/(x-2)), находящиеся в интервале (-0,4;0,8) для x, изменяющегося от 8 до -6 с шагом 1.
24. Ввести с клавиатуры N чисел. Напечатать те из них, которые принадлежат интервалу (2;9).
25. Для геометрической прогрессии 2, 6, 18, 54, 162 ... определить первые n членов этой прогрессии.

26. Ввести с клавиатуры N чисел. Напечатать те из них, которые не принадлежат интервалу (1;5).
27. Найти n членов последовательности $x_1 = x_2 = x_3 = 1$; $x_k = x_{k-1} - 2x_{k-3}$.
28. Вычислить последовательность N чисел $A_0 = x$, $A_1 = 2$, $A_k = A_{k-1} + A_{k-2}$
29. Составить таблицу перевода килограммов (от 1 до 13) в граммы с шагом
30. Найти сумму значений функции Y=COS(x/A)+x/(A-2) для x, изменяющегося от 2 до 13 с шагом 1 (A - произвольное число).

МУ к ЛР 2

Исследование простых двухуровневых клиент-серверных приложений в Java

2.2.1 Цели занятия

Выработать умения и навыки по составлению программ простых одноуровневых клиент-серверных приложений на основе технологии сокетов.

2.2.2 Теоретические основы и пример выполнения

2.2.2.1 Создание сетевых приложений с использованием TCP

Java поддерживает классы и методы, которые позволяют устанавливать соединение с удаленным компьютером, используя протокол TCP. В отличие от UDP, TCP является протоколом, ориентированным на установление соединения, которое гарантирует надежную связь между приложениями клиента и сервера. Взаимодействие с использованием протокола TCP, начинается после установления соединения между сокетами клиента и сервера. Сокет сервера "слушает" запросы на установление соединения, отправленные сокетами клиентов, и устанавливает соединение. После установления соединения между приложениями клиента и сервера, они могут взаимодействовать друг с другом.

Java упрощает сетевое программирование, путем инкапсуляции функциональности соединения сокета TCP в классы сокета, в которых класс Socket предназначен для создания сокета клиента, а класс ServerSocket для создания сокета сервера.

2.2.2.2 Идентификация методов классов Socket и ServerSocket

Socket является базовым классом, поддерживающим протокол TCP. Класс Socket обеспечивает методы для потокового ввода/вывода, облегчает выполнение операций чтения и записи в сокет и является обязательным для программ, выполняющих сетевое взаимодействие.

Для создания объектов класса Socket используются следующие конструкторы, определенные в классе Socket:

- public Socket (InetAddress IP_address, int port) - создает объект Socket, который соединяется хостом, заданным в параметрами IP_address и port;
- public Socket (String hostname, int port) – создает объект Socket, который соединяется с хостом, заданным параметрами имя хоста или IP адрес и port, который сервер "слушает".

ServerSocket представляет собой класс, используемый программами сервера для прослушивания запросов клиентов. ServerSocket реально не выполняет сервис, но создает объект Socket от имени клиента, через который выполняется взаимодействие с сокетом клиента.

Для создания и инициализации объектов ServerSocket используются следующие конструкторы, определенные в классе ServerSocket:

- public ServerSocket(int port_number)- создает сокет сервера на заданный порт на локальной машине. Клиентам следуетиспользовать этот порт, чтобы общаться с сервером. Если номер порта 0, то сокет сервера создается на любой свободный порт локальной машины;
- public ServerSocket(int port, int backlog) - создает сокет сервера на заданный порт на локальной машине. Второй параметр задает максимальное количество соединений клиентов, которые сокет сервера поддерживает на заданном порту;
- public ServerSocket(int port, int backlog, InetAddress bindAddr) - создает сокет сервера на заданный порт. Третий параметр используется для создания сокета сервера хоста, подключенного к нескольким физическим линиям (multi-homed host). Сокет сервера принимает запросы клиента только с заданных IP адресов.

2.2.2.3 Создание северной части по протоколу TCP/IP в Java

Для создания серверного приложения сокета TCP, необходимо выполнить следующие шаги:

- создать объект сокета сервера ServerSocket (класс Server);
- прослушивать запросы клиента на соединение;
- запустить сервер;
- создать поток соединения для запросов клиентов.

2.2.2.4 Разработка класса Server

Класс Server наследует класс Thread и, таким образом, поддерживает многопоточное выполнение. Объект ServerSocket "слушает" запросы клиентов и конструктор класса Server создает объект ServerSocket. Сообщение об ошибке отображается, если возникает исключительная ситуация при запуске сервера.

Фрагмент кода для конструктора сервера показан в листинге 2.8.

Листинг 2.8- Фрагмент конструктора класса Server

```
public Server()
{
 try
 {
 serverSocket = new ServerSocket(1001);
 }
 catch(IOException e)
 {
 fail(e, "Невозможно запустить сервер.");
 }
 System.out.println("Сервер запущен. . .");
 this.start(); // Запускается поток
}
```

В приведенном фрагменте кода используется общий метод обработки ошибок fail(), который обеспечивает обработку всех исключительных ситуаций. Метод принимает два аргумента (объект Exception и объект String) и выводит сообщение об ошибке.

Фрагмент кода для метода fail() выглядит следующим образом (листинг 2.9):

Листинг 2.9- Формирование сообщение об ошибке

```
public static void fail(Exception e, String str)
{
 System.out.println(str + "." + e);
}
```

2.2.2.5 «Прослушивание» запросов клиентов

Метод run() класса Server, как любой поток, который реализует интерфейс Runnable, содержит инструкции для потока. В этом случае сервер переходит в бесконечный цикл и прослушивает запросы клиентов. Когда сервер обнаруживает подключение клиента, метод accept() класса ServerSocket выполняет соединение. При этом сервер создает объект класса Connection для клиента. Объект класса Socket передается конструктору класса Connection и взаимодействие между клиентом и сервером выполняется через этот сокет.

Фрагмент кода для метода run() выглядит показан на листинге 2.10.

Листинг 2.10- Метод для прослушивания запросов клиентов

```
public void run()
{
 try
 {
 while(true)
 {
 Socket client = serverSocket.accept();
 Connection con = new Connection(client);
 }
 }
 catch(IOException e)
 {
 fail(e, “Не прослушивается”);
```

```
 }
}
```

2.2.2.6 Запуск сервера

Фрагмент кода для метода main() приведен в листинге 2.11.

Листинг 2.11- Запуск сервера

```
public static void main(String args[])
{
 new Server();
}
```

В этом фрагменте кода создается объект класса Server, который запускает поток.

2.2.2.7 Создание потокового соединения

Класс Connection нужен для потокового соединения с клиентом (листинг 2.12).

Листинг 2.12- Класс для создания потокового соединения

```
class Connection extends Thread
{
protected Socket netClient;
 protected BufferedReader fromClient;
 protected PrintStream toClient;
 public Connection(Socket client)
 {
 netClient = client;
 try
 {
fromClient = new BufferedReader(new
InputStreamReader(netClient.getInputStream()));
toClient = new PrintStream(netClient.getOutputStream());
 }
 catch(IOException e)
 {
```

```
try
{
netClient.close();
}
catch(IOException e1)
{
System.err.println("Unable to set up streams"
+ e1);
return;
}
}
this.start();
}
public void run()
{
String clientMessage;
try
{
for(;;)
{
clientMessage = fromClient.readLine();
if(clientMessage == null)
break;
// Посылает подтверждение клиенту
toClient.println("Received");
}
}
catch(IOException e)
{}
finally
{
try
{
netClient.close();
}
```

```
 catch(IOException e)
 {}
 }
}
}
```

В представленном фрагменте кода класс Connection создает объект fromClient класса BufferedReader, который получает ввод от клиента, используя метод getInputStream(). Объект класса PrintStream (toClient) дает возможность серверу отправлять клиенту данные, используя метод getOutputStream(). Таким образом, возникают необходимые (прием и передача) возможности взаимодействия.

Когда клиент соединяется с сервером, сервер использует метод readLine() объекта fromClient, чтобы запомнить сообщение, посланное клиентом в переменной clientMessage типа String. Метод println() используется для вывода сообщения “Received ” сокету.

Для выхода из системы сервер завершает цикл. При этом выполняется блок finally для закрытия сокета клиента. Закрытие сокета является важным действием, поскольку сохранение активного соединения неизбежно приводит к потери памяти сервера. Блок finally обеспечивает закрытие ранее установленного соединения. Следует отметить, что сервер является многопоточным, и каждый клиент получает свой собственный поток на сервере.

2.2.2.8 Пример создания класса Server

В листинге 2.13 показан создания класса Server, который принимает запросы соединения клиента и посылает строку Login, как ответное сообщение клиенту.

Листинг 2.13- Пример кода класса Server

```
import java.io.*;
import java.net.*;
public class Server extends Thread {
```

```
    ServerSocket serverSocket; // Определяется переменная server-
Socket
    public Server() {
    try {
    /*
    * Создание объекта ServerSocket, который принимает запросы
    * соединения от клиентов через порт 1001
    */
    serverSocket = new ServerSocket(1001);
    System.out.println(serverSocket.toString());
    } catch (IOException e) {
    fail(e, "Could not start server.");
    }
    System.out.println("Сервер запущен . . .");
    /* Стартует поток */
    this.start();
    }
    public static void fail(Exception e, String str) {
    System.out.println(str + "." + e);
    }
    public void run() {
    try {
    while (true) {
    /* Принимаются запросы от клиентов */
    Socket client = serverSocket.accept();
    /*
 * Создается объект соединение, чтобы управлять
    взаимодействием
    * кдиента с сервером
    */
    Connection con = new Connection(client);
    }
    } catch (IOException e) {
    fail(e, "Not listening");
    }
```

```
}
public static void main(String args[]) {
/* Запускается сервер */
new Server();
}
}
class Connection extends Thread {
/* Declare the variables */
protected Socket netClient;
protected BufferedReader fromClient;
protected PrintStream toClient;
public Connection(Socket client) {
netClient = client;
try {
/* Создается входной поток, чтобы принимать данные от
клиента */
fromClient = new BufferedReader(new
InputStreamReader(
netClient.getInputStream()));
/* Создается выходной поток, чтобы посылать данные
клиенту */
toClient = new
PrintStream(netClient.getOutputStream());
} catch (IOException e) {
try {
/* Закрывается сокет клиента */
netClient.close();
} catch (IOException e1) {
System.err.println("Unable to set up streams" +
e1);
return;
}
}
/* Start the thread */
this.start();
```

```
}
public void run() {
String login, password;
try {
while (true) {
toClient.println("Login: ");
/* Принимается login как ввод от клиента */
login = fromClient.readLine();
/* Завершить соединение, когда Bye вводится как login */
if (login == null || login.equals("Bye")) {
System.out.println("Exit");
return;
} else
System.out.println(login + " logged
in");
// Посылается подтверждение клиенту
toClient.println("Password: ");
/* Принимается пароль то клиента */
password = fromClient.readLine();
}
} catch (IOException e) {
} finally {
try {
netClient.close();
} catch (IOException e) {
}
}
}
}
```

2.2.2.9 Создание клиента TCP/IP

Первый шаг выполнения клиента состоит в установлении соединения между клиентом и сервером. Для установления соединения между клиентом и

сервером необходимо создать объект Socket. Для создания приложения клиента сокета TCP необходимо выполнить следующие задачи:

- создать сокет клиента, используя объект Socket;
- читать и писать в сокет;
- закрыть соединение.

2.2.2.10 Создание сокета клиента

Объект сокета клиента создается с помощью конструктора класса Socket, принимающего два параметра, IP адрес и номер порта, которые прослушиваются сервером. Следующий фрагмент кода используется для создания сокета клиента (листинг 2.14).

Листинг 2.14- Создание сокета клиента

```
try
{
 Socket clientSocket = new Socket("127.0.0.1", 1001);
}
catch(UnknownHostException e)
{
System.err.println("Неопределенное имя хоста ");
System.exit(1);
}
```

В предыдущем фрагменте кода IP адрес равный ‘127.0.0.1’ и порт равный ‘1001’ определяют сокет, на котором сервер прослушивает запросы клиента.

2.2.2.11 Чтение и запись в сокет

После установления соединения между клиентом и сервером, клиент посылает запрос на сервер через сокет. Чтение и запись в сокет аналогичны чтению из файла и записи в файл. Чтобы обеспечить возможность клиенту общаться с сервером, необходимо выполнить следующие действия:

- объявляются два объекта по одному для классов PrintStream и BufferedReader. Эти объекты будут использоваться для чтения и записи в сокет socket. PrintStream out = null; // Объект для записи в сокет BufferedReader in = null; // Объект для чтения из сокета;
- объекты PrintStream и BufferedReader связываются с сокетом. out = new PrintStream(clientSocket.getOutputStream()); in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
- методы getInputStream() и getOutputStream() класса Socket позволяют клиенту взаимодействовать с сервером. Метод getInputStream() позволяет объекту BufferedReader читать из сокета, а метод getOutputStream() позволяет объекту PrintStream писать в сокет.

Объявляется еще один объект класса BufferdReader для связи со стандартным входом, чтобы данные, введенные в приложении клиенте, могли передаваться на сервер. Следующий фрагмент кода используется для чтения данных из окна консоли (листинг 2.15).

Листинг 2.15- Чтение данных из окна консоли

```
BufferdReader stdin = new BufferedReader(new
InputStreamReader(System.in));
String str;
while((str = stdin.readLine()).length() != 0)
{
out.println(str);
}
```

Представленный фрагмент кода позволяет пользователю вводить данные с клавиатуры. Цикл while продолжается до тех пор, пока пользователь не введет символ завершения ввода (Ctrl-Z).

2.2.2.12 Закрытие соединения

Операторы, представленные ниже, закрывают потоки и соединения с сервером: out.close(); in.close(); stdin.close().

Сокет клиента принимает имя пользователя и пароль, и обеспечивает связь. Для завершения соединения, пользователь должен ввести ‘Bye’. Можно использовать следующий код для создания класса Client (листинг 2.16).

Листинг 2.16- Пример создания класса Client

```
import java.net.*;
import java.io.*;
public class Client {
public static void main(String[] args) throws IOException {
Socket clientSocket;
PrintStream out = null;
BufferedReader in = null;
try {
/* Создается объект сокет, чтобы соединиться с сервером
*/
clientSocket = new Socket("127.0.0.1", 1001);
/* Создается выходной поток, чтобы посылать данные на
сервер */
out = new
PrintStream(clientSocket.getOutputStream());
/* Создается входной поток, чтобы принимать данные с
сервера */
in = new BufferedReader(new InputStreamReader(
clientSocket.getInputStream()));
} catch (UnknownHostException e) {
System.err.println("Unidentified hostname ");
System.exit(1);
} catch (IOException e) {
System.err.println("Couldn't get I/O ");
System.exit(1);
}
/* Создается входной поток, чтобы читать данные из окна консо-
ли
```

```
*/
BufferedReader stdin = new BufferedReader(new
InputStreamReader(
(System.in)));
/* Чтение из сокета */
String login = in.readLine();
System.out.println(login);
/* Прием login */
String logName = stdin.readLine();
out.println(logName);
/* Чтение из сокета */
String password = in.readLine();
System.out.println(password);
/* Прием password */
String pass = stdin.readLine();
out.println(pass);
String str = in.readLine();
System.out.println(str);
while ((str = stdin.readLine()) != null) {
out.println(str);
if (str.equals("Bye"))
break;
}
out.close();
in.close();
stdin.close();
}
}
```

Представленный выше код сохраняется как Client.java. При выполнении класса Client, отображается сообщение для ввода Login. После приема Login на консоли класса Client появляется сообщение для ввода password. Рис. 7 показывает вывод класса Client, который посылает login XXXX и password для входа на Server. Введенный login в окне консоли класса Client передается классу

Server. Когда на сервер передается строка «Bye» в качестве login от клиента, соединение между клиентом и сервером прекращается.

2.2.3 Порядок проведения исследований

Алгоритм выполнения задач практического занятия следующий:

- выбрать 5 задач по следующему правилу: номер по журналу- первая задача; номер каждой последующей задачи определяется прибавлением цифры 3 к номеру первой задачи, который только что вычислили (если достигнуто окончание списка вариантов здач, то перейти в его начало);
- составить классы Swrver и Client и установить соединение между ними;
- составить методы (функции) решения всех задач, поместить их в класс Server. Все исходные данные вводить в консоли клиента, передавать их на сервер, а полученные результаты решений задач выводить на экран в консоли клиента;
- оформить отчет для всего приложения вцелом, включив в него задание, блок-схему алгоритма (в электронном виде), текст программы и skrinshert результата выполнения каждой задачи и представить его на проверку.

2.2.4 Варианты заданий

1. Составить программу для перевода длины в метрах в длину в сантиметрах, определив функцию, выполняющую это преобразование и передав длину в метрах в качестве параметра.

2. Составить программу для нахождения суммы элементов каждого из трех массивов, введенных с клавиатуры, определив функцию, выполняющую это действие, и передавая массивы в качестве параметра.

3. Даны числа S, T. Получить с использованием функции пользователя F(T,-2S,1.17)+F(2.2,T,S-T) где F(A, B, C) = (2A-B-sin(C))/(5+C)

4. Составить программу перевода двоичной записи натурального числа в десятичную, описав соответствующую функцию с параметром. Перевод осуществлять для чисел, вводимых с клавиатуры. Признак конца ввода - число 0.

5. Даны числа S, T. Получить с использованием функции пользователя с параметрами:

G(1,sin(S))+2G(T*S,24)-G(5,-S), где G(A,B)=(2A+B*B)/(A*B*2+B*5).

6. Составить программу для расчета значений гипотенузы треугольника, определив функцию, выполняющую этот расчет. Катеты передаются в качестве параметров.

7. Найти периметр десятиугольника, координаты вершин которого заданы. Определить процедуру вычисления расстояния между двумя точками, заданными своими координатами, которые передаются функции в качестве параметров из основной программы.

8. Найти периметр шестиугольника, координаты вершин которого заданы. Определить процедуру вычисления расстояния между двумя точками, заданными своими координатами. Координаты передаются функции в качестве параметров из основной программы.

9. Найти площадь пятиугольника, координаты вершин которого заданы. Определить процедуру вычисления расстояния между двумя точками, заданными своими координатами, и процедуру вычисления площади треугольника по трем сторонам. Описать функции с соответствующими формальными параметрами.

10. Составить программу вывода на экран всех натуральных чисел, не превосходящих N и делящихся на каждую из своих цифр. Описать соответствующую функцию, получающую из основной программы в качестве параметра натуральное число и возвращающую TRUE, если оно удовлетворяет указанному условию.

11. Используя подпрограмму - функцию, составить программу для нахождения максимального из трех чисел. Числа передаются функции в качестве параметров.

12. Используя подпрограмму - функцию, составить программу для печати знаков трех чисел, введенных с клавиатуры и передаваемых функции в качестве параметра.

13. Используя подпрограмму - функцию, составить программу для возведения чисел в целую положительную степень. Число передаются функции в качестве параметра из основной программы. Расчет вести для чисел, пока не будет введено число, равное 0.

14. Используя подпрограмму - функцию, составить программу для вычисления функции Z=(X1+Y1)/(X1*Y1), где X1 - первый корень уравнения X^2-4*X-1=0; Y1 - первый корень уравнения $2*Y^2 + A*Y - A^2 = 0$ (A - произвольное).

15. Задав функцию, вывести на печать средние арифметические двух массивов, введенных с клавиатуры. Массив передается функции в качестве параметра.

16. Задав функцию, рассчитать и вывести на печать максимальные значения в трех парах чисел, вводимых с клавиатуры. Пара чисел передается функции в качестве параметра.

17. Найти периметр восьмиугольника, координаты вершин которого заданы. Определить функцию вычисления расстояния между двумя точками, заданными своими координатами. Координаты передать функции в качестве параметров.

18. Даны четыре пары чисел. Получить с использованием функции пользователя наибольший общий делитель для каждой пары.

19. Даны числа A, B, C. Получить с использованием функции пользователя наименьшее значение. Числа передаются функции из основной программы в качестве параметров.

20. Даны числа x = 1,2,...,N. Получить с использованием функции пользователя значения 3*P(X+3)*P(X) для заданных x, где $P(X) = 10*X^3 - 14*X^2 + 12*X - 2$.

21. Составить программу для расчета значений катета треугольника, определив функцию, выполняющую этот расчет. Гипотенуза и второй катет передаются в качестве параметров.

22. Даны целые числа a,b,c,d. Проверить с использованием функции пользователя их четность. Число для проверки передается в функцию в качестве параметра из основной программы.

23. Для каждого из 10 введенных с клавиатуры чисел напечатать сообщение: является ли оно простым или нет, описав функцию логического типа, возвращающую значение “ИСТИНА”, если число, переданное ей в качестве параметра, является простым.

24. Даны числа S, T. Получить с использованием функции пользователя Y(T,S)=G(12,S)+G(T,S)-G(2S-1,S*T), где G(A,B)=(2*A+B*B)/(A*B*2+B*5).

25. Определите функцию, определяющую, какой целой степенью числа 2 является ее аргумент (если число не является степенью двойки - выдать соответствующее сообщение).

26. Определите функцию, подсчитывающую сумму N первых элементов целочисленного массива A. N и массив A передать в качестве параметров.

27. Вычислить количество простых чисел, не превосходящих заданного N. Описать функцию логического типа, возвращающую значение true, если число простое и false в противном случае.

28. Используя подпрограмму - функцию с параметрами, составить программу для вычисления функции $F(X,Y) = (2X^3-4*X^2+X+1)/(9*Y^3+Y+4) + 3*Y^2+5*Y$.

29. Составить программу для перевода веса в граммах в вес в килограммах, определив функцию, выполняющую это преобразование. Вес в граммах передается функции в качестве параметра.

30. Даны числа S, T. Получить с использованием функции пользователя G(12, S)+G(T, S)-G(2S-1, S*T) где G(A, B) = (2*A+B*B)/(A*B*2+B*5).

МУ к ЛР 3

Исследование многоуровневых клиент-серверных приложений в Java

2.3.1 Цели занятия

Развить навыки составления клиент-серверных приложений с основной обработкой информации на стороне клиента путем исследования достоинств и недостатков такого подхода.

2.3.2 Теоретические основы и пример выполнения

Основным отличием разработки такой разновидности приложения является то, что вся обработка информации, еще говорят: бизнес – логика приложения выполняется на стороне клиента.

Есть преимущества и недостатки такого подхода к созданию клиент-серверных приложений.

Достоинства в том, что сервер является очень простым и служит для организации и хранения данных, а на стороне клиента происходит вся основная обработка информации.

Недостатки – это сложная структура клиентской части, реализация которой предъявляет жесткие требования к ресурсам устройств, на которых она располагается.

С точки зрения технологии разработки ничего не меняется по сравнению с созданием классических клиент – серверных разработок, которые были рассмотрены в предыдущей лабораторной работе. Поэтому, необходимо взять за основу алгоритм решения задачи, показанный выше и разработать клиент – серверное приложение по выбранному варианту с той только разницей, что решение логических заданий теперь необходимо представить на стороне клиентской части приложения. Другими словами, все методы, которые нужно будет составить для решения указанных заданий, должны быть размещены в одном из классов клиентской части приложения, а исходные данные – в классе сервера.

2.3.3 Порядок проведения исследований

Алгоритм выполнения заданий лабораторной работы следующий:

- выбрать 5 задач по следующему правилу: номер по журналу- первая задача; номер каждой последующей задачи определяется прибавлением цифры 3 к номеру первой задачи, который только что вычислили (если достигнуто окончание списка вариантов здач, то перейти в его начало);
- составить классы Swrver и Client и установить соединение между ними;
- составить методы (функции) решения всех задач, поместить их в класс Server. Все исходные данные вводить в консоли клиента, передавать их на сервер, а полученные результаты решений задач выводить на экран в консоли клиента;
- оформить отчет для всего приложения вцелом, включив в него задание, блок-схему алгоритма (в электронном виде), текст программы и skrinshert результата выполнения каждой задачи и представить его на проверку.

2.3.4 Варианты заданий

1. НАИБОЛЬШИЙ ПРЯМОУГОЛЬНИК. Дана матрица, состоящая из нулей и единиц. Найти наибольший по площади прямоугольник, состоящий из одних единиц.
2. ПЕРИОД ДРОБИ. Найти период десятичной дроби, равной M/N, где M и N - натуральные числа.
3. НЕСОСТАВЛЯЕМОЕ ЧИСЛО. Задан массив натуральных чисел. Найти минимальное натуральное число, не представимое суммой чисел из массива. Каждый элемент массива может входить в сумму не более одного раза.

4. СООТВЕТСТВИЕ ШАБЛОНУ. Установить соответствие имени файла заданному шаблону. Шаблоном называется строка, в которой “,” означает любой символ, а “*” означает любую последовательность символов, в том числе пустую.
5. КОД ГРЕЯ. Построить N -разрядный код Грея. Кодом Грея называется такая последовательность N -разрядных двоичных чисел, в которой два соседних, а также первое и последнее число отличаются одним разрядом. Например, для N=2 один из кодов Грея таков: 00, 01, 11, 10.
6. ОБМЕН ЧАСТЕЙ МАССИВА. Задан одномерный массив и граница между двумя его частями. Поменять порядок расположения частей массива, не делая лишних присваивания. Например: 1 2 3 4 | 5 6 7 → 5 6 7 | 1 2 3 4.
7. ЛЕВЫЕ ПОВОРОТЫ. Ломаная задана координатами своих узлов. Сосчитать количество левых поворотов при движении вдоль ломаной.
8. ПЯТНА НА ШКУРЕ. Подсчитать количество черных пятен на белой шкуре. Шкуру представить в виде 0,1 - матрицы, где 0 - белый, а 1 - черный цвет.
9. ТОЖДЕСТВО КОМПОСТЕРОВ. Имеется два автобусных талона. Установить, пробиты ли они одним компостером или разными. Талон можно вкладывать в компостер любой стороной так, чтобы края талона были параллельны краям компостера и все отверстия компостера помещались на талоне. Отверстия компостера размещаются на матрице размером M×N.
10. ПОДСЧЕТ СЛОВ. Написать программу подсчета количества слов в файле с русским текстом, которые начинаются с заданного буквосочетания. Учесть возможность переноса слов.
11. ВЫРАВНИВАНИЕ ТЕКСТА. В ASCII файле находится русский текст. Выровнять его по левой и правой границам, добиваясь наиболее плотного и равномерного распределения слов в строках.
12. ТОЛКОВЫЙ СЛОВАРЬ. В толковом словаре одни слова определяются через другие так, что нигде нет порочного круга. Описать процедуру, ко-

торая пронумеровывает слова таким образом, чтобы слова с большим номером определялись через слова с меньшими номерами.

13. ЛЕВЫЕ ПОВОРОТЫ. Ломаная задана координатами своих узлов. Описать функцию подсчета количества левых поворотов при движении вдоль ломаной, а также процедуру вывода изображения ломаной на экран.
14. МАТРИЦА ОТНОШЕНИЙ. Задана матрица вида

X_1	X_2	X_3	...	X_N
Y_1	R_{11}	R_{12}	...	R_{1N}
Y_2	R_{21}	R_{22}	...	R_{2N}
...				
Y_M	R_{M1}	R_{M2}	...	R_{MN},

где X_i и Y_j целые числа, а R_{ij} - арифметические отношения: <, =, >. Найти значения X_i и Y_j, которые удовлетворяли бы отношениям R_{ij}.

15. НЕУБЫВАЮЩАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ. В программе описать процедуру, печатающую самую длинную неубывающую последовательность в последовательности целых чисел, избежав при этом полного перебора.
16. БОЙ ШАШЕК. На доске стоят белые шашки и одна черная. Описать процедуру, определяющую, какие шашки может побить черная за один ход.
17. КРАТЧАЙШИЙ ПУТЬ КОНЯ. Имеется бесконечная шахматная доска. В программе описать функцию, подсчитывающую, за какое наименьшее число ходов конь сможет перейти с поля (x1,y1) на поле (x1,y2)?
18. ВЗАИМНЫЙ ЗАЧЕТ ДОЛГОВ. Есть множество предприятий, которые должны друг другу. Произвести взаимный зачет долгов на максимальную сумму.

19. САМОРОДКИ. Есть множество золотых самородков известного веса. Описать процедуру, которая позволяет разделить самородки на две кучи, наиболее близкие по весу.

МУ к ЛР 4

Исследование серверной части WEB приложений

2.4.1 Цели занятия

Исследовать серверную часть web – приложения, размещенного в сети Интернет. Изучить особенности работы подобных приложений.

2.4.2 Теоретические основы и пример выполнения

Опишем алгоритм составления типового web – приложения.

1. Создание проекта типа web – приложение: WpfBrowserApplication1 в Visual Studio C#.
2. Составление необходимого интерфейса, используя библиотеку управляющих элементов.
3. Отладка приложения на Local host.
4. Размещение приложения на компьютере, подключенного к Internet и доступ к нему по указанному URL адресу.

2.4.2.1 Пример разработки web – приложения

Составим приложение для сортировки двумерного массива случайных простых чисел методом вставки.

Приложение должно задавать двумерный массив размерностью m x n элементов, заполнять его случайными простыми числами, затем переписывать его построчно в одномерный массив, сортировать его указанным методом по возрастанию элементов и снова переписывать построчно в двумерный массив.

Реализуем алгоритм, указанный в пункте 2.2.2.

Пункты 1 и 2 дают следующий проект в среде разработки (рисунок 2.10).

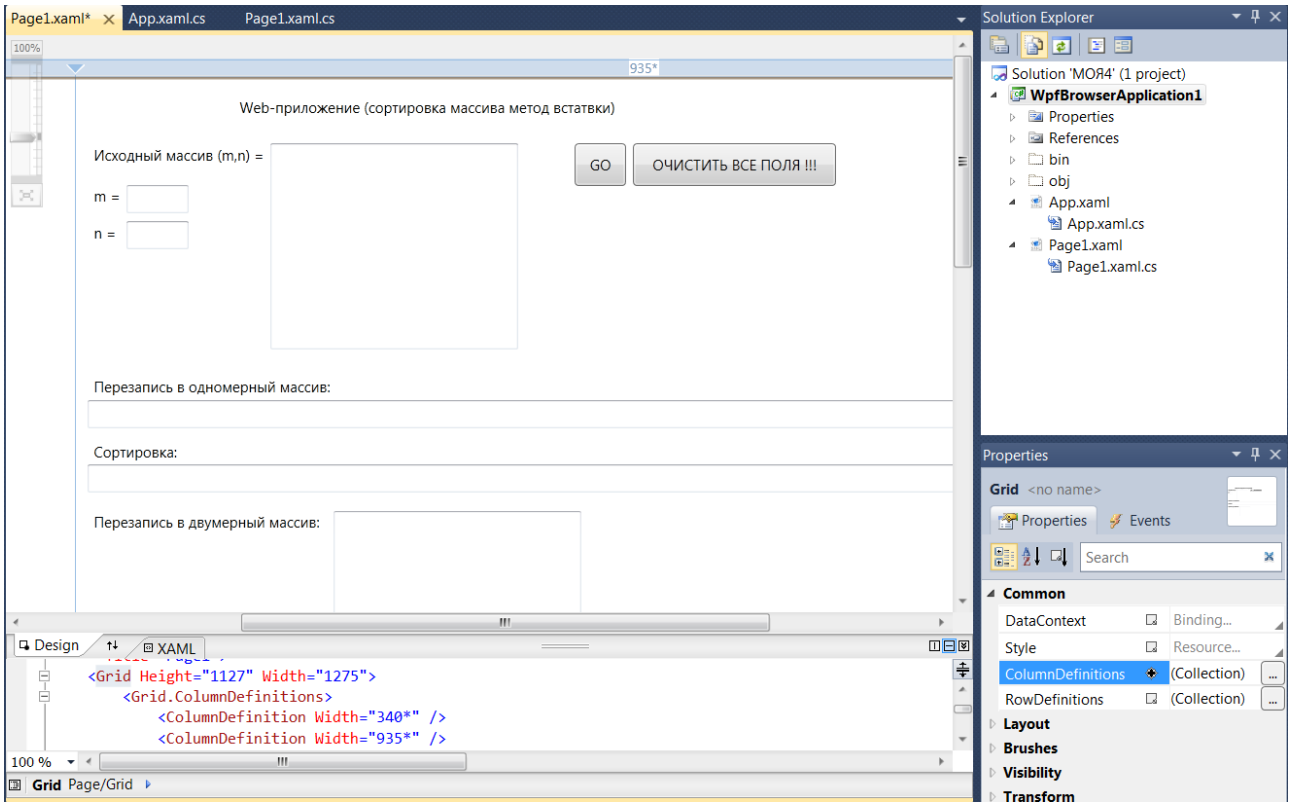

Рисунок 2.10- Вид проекта и интерфейс приложения

Отладка приложения (пункт 3 алгоритма, приведенного в 2.2.2) заключается в его корректном запуске и устранении ошибок.

Так, перед тем, как запускать проект нужно во вкладке Свойства проект Visual Studio и конкретно в свойстве Безопасность дать полное «доверие» проекту (рисунок 2.11).

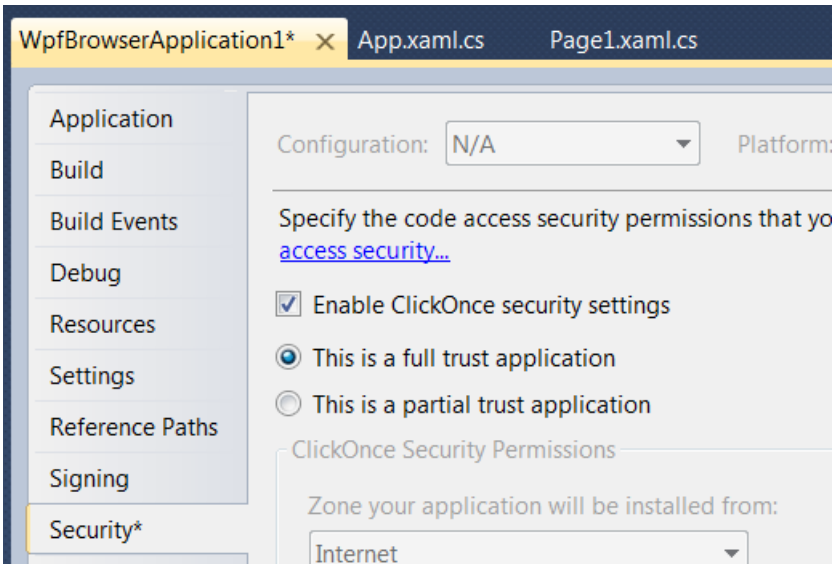

Рисунок 2.11- Открытие полных прав для проекта

Далее, необходимо установить Entenet Explorer браузером по умолчанию, для этого в Пуске и в Программах по умолчанию нужно выбрать эту программу (рисунок 2.12).

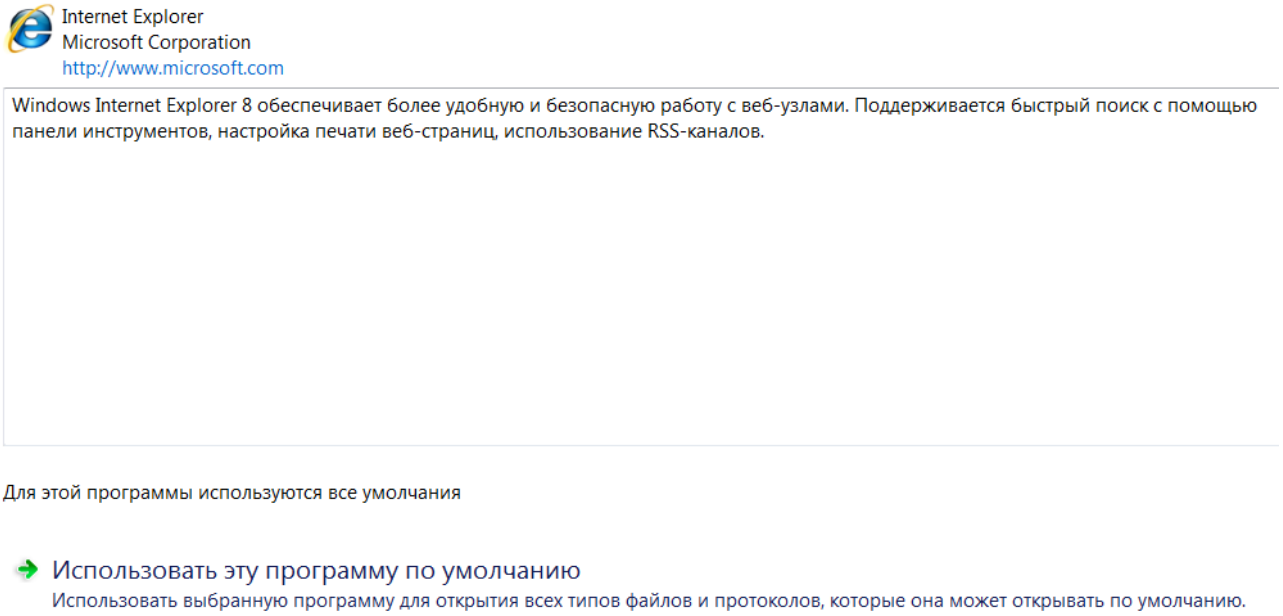

Рисунок 2.12- Настройка браузера по умолчанию

Далее, запускаем приложение и используем его (рисунок 2.13).

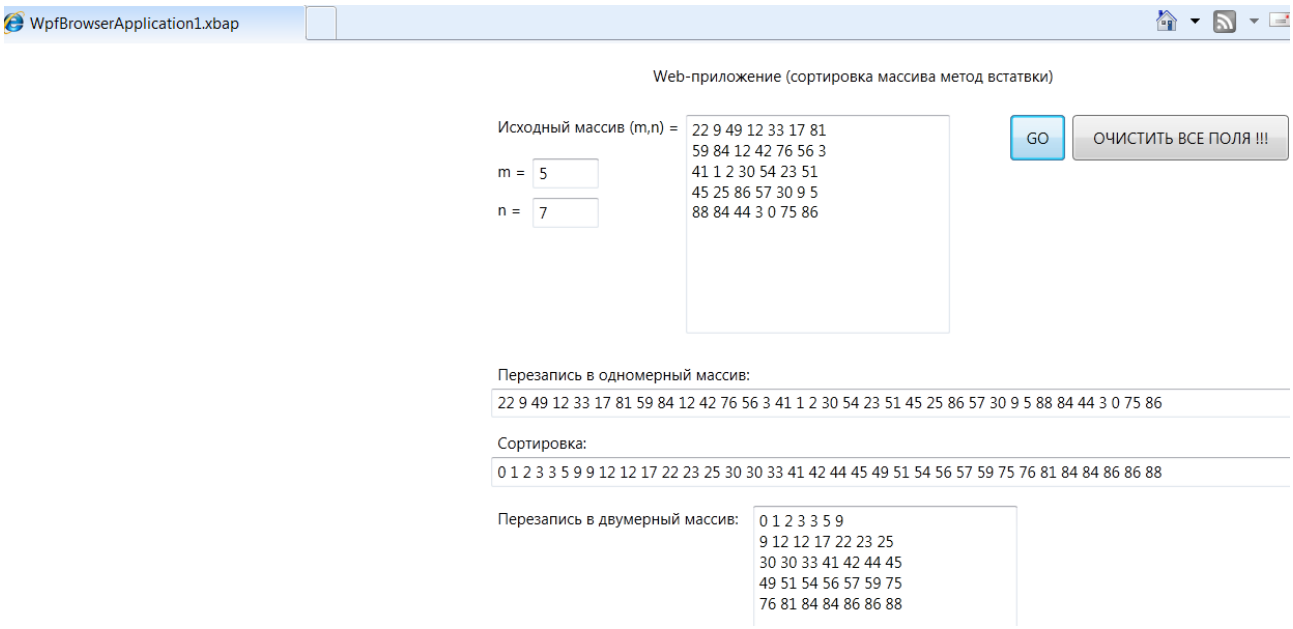

Рисунок 2.13- Проверка работы web – приложения

Для реализации 4-го пункта алгоритма из 2.2.2 запустим удаленно через сеть Интернет разработанное приложение. Для этого необходимо его разместить на зарегистрированном ресурсе и указать его URL адрес.

Протокол TCP запрашивает два элемента данных: IP адрес и номер порта. URL - это стандартизированный способ записи адреса ресурса в сети Интернет. Интернет протокол (Internet Protocol - IP) обеспечивает логический адрес, называемый IP-адресом сетевого устройства. Каждому доменному имени компьютера в системе доменных имен (DNS - Domain Name System) соответствует IP-адрес. IP-адреса, используемые в Интернет, имеют определенный формат. Каждый адрес представляется 32-разрядным числом (т.е. 4-мя байтами), состоящим из четырех 8-разрядных чисел (байт), каждое в диапазоне значений от 0 до 255. Поэтому, для зарегистрированного в сети Интернет ресурса, например, http://www.skf-mtusi.ru/ имеется свой IP адрес. Следовательно, чтобы удаленно запустить разработанное приложение нужно указать, например, http://www.dm.ru/ WpfBrowserApplication1.exe. Очевидно, что указанный домен dm.ru должен быть зарегистрирован в сети Интернет.

Текст программы web – приложения показан в листинге 2.17.

Листинг 2.17- Текст программы web – приложения

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
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;

namespace WpfBrowserApplication1
{
    /// <summary>
    /// Interaction logic for Page1.xaml
    /// </summary>
    public partial class Page1 : Page
    {
 public Page1()
        {
            InitializeComponent();
        }
        private void button1_Click(object sender, RoutedEven-
tArgs e)
        {
            met1();
        }
        Random rd1 = new Random();
        void met1()
        {
            int i = 0, j = 0;
```

```
int n = int.Parse(textBox4.Text);
int m = int.Parse(textBox5.Text);

int[,] X = new int[n, m];
for (i = 0; i < n; i++)
{
    for (j = 0; j < m; j++)
    {
        X[i, j] = rd1.Next(90);
    }
}
for (i = 0; i < n; i++)
{
    for (j = 0; j < m; j++)
    {
        textBox1.Text += X[i, j] + " ";
    }
    textBox1.Text += "\n";
}
int[] Y = new int[n * m];
int index = 0;
for (i = 0; i < n; i++)
{
    for (j = 0; j < m; j++)
    {
        Y[index] = X[i, j];
        index++;
    }
}
for (index = 0; index < n * m; index++)
{
    textBox6.Text += Y[index] + " ";
}
int outt, inn, temp;
```

```
                for (outt = 1; outt < n * m; outt++) // out - раз-
делительный маркер
                {
                    temp = Y[outt]; //Скопировать помеченный эле-
мент
                    inn = outt; //Начать перемещения с out
                    while(inn>0 && Y[inn-1] >= temp) // Пока не
найден меньший элемент
                    {
                        Y[inn] = Y[inn-1]; // Сдвинуть элемент
вправо
                        --inn; // Перейти на одну позицию влево
                    }
                    Y[inn] = temp; // Вставить помеченный элемент
                }
                int r = 0;
                for (r = 0; r < n * m; r++)
                {
                    textBox2.Text += Y[r] + " ";
                }
                index = 0;
                r = 0;
                for (i = 0; i < n; i++)
                {
                    for (j = 0; j < m; j++)
                    {
                        X[i, j] = Y[r];
                        r++;
                    }
                }
                for (i = 0; i < n; i++)
                {
                    for (j = 0; j < m; j++)
                    {
                        textBox3.Text += X[i, j] + " ";
```

```
                }
                textBox3.Text += "\n";
            }
        }
        private void button1_Click_1(object sender,
RoutedEventArgs e)
        {
            met1();
        }
        private void button2_Click_1(object sender,
RoutedEventArgs e)
        {
            textBox1.Text = "";
            textBox2.Text = "";
            textBox3.Text = "";
            textBox4.Text = "";
            textBox5.Text = "";
            textBox6.Text = "";}}}
```

2.4.3 Порядок проведения исследований

Алгоритм выполнения заданий ЛР следующий:

- выбрать 5 задач по следующему правилу: номер по журналу- первая задача; номер каждой последующей задачи определяется прибавлением цифры 3 к номеру первой задачи, который только что вычислили (если достигнуто окончание списка вариантов задач, то перейти в его начало);
- составить методы (функции) решения всех задач, поместить их в класс Form1 и снабдить пользовательским интерфейсом в виде web – приложения;
- исследовать запуск приложения через Internet Explorer, разместив приложение на Local host. Сделать выводы о достоинствах и недостатках web – приложений как определенного вида клиент – серверных структур;

– оформить отчет для всего приложения вцелом, включив в него задание, блок-схему алгоритма (в электронном виде), текст программы и skrin-shert результата выполнения каждой задачи и представить его на проверку.

2.4.4 Варианты заданий

1. Найти среднее арифметическое положительных чисел, введенных с клавиатуры. Всего ввести N различных чисел.

2. Ввести с клавиатуры N чисел. Найти сумму тех из них, которые принадлежат интервалу (2;9).

3. Для N введенных с клавиатуры чисел найти сумму положительных кратных 3.

4. Для арифметической прогрессии 4, 9, 14, 19... найти первые n членов этой прогрессии.

5. Найти сумму отрицательных значений функции Z=sin(5-x)/cos(x-2) для x, изменяющегося на отрезке [-5,12] с шагом 1,23.

6. Найти среднее арифметическое отрицательных чисел, введенных с клавиатуры. Всего ввести N различных чисел.

7. Найти среднее арифметическое чисел, принадлежащих отрезку [2,184], кратных 2 и введенных с клавиатуры. Всего ввести N различных чисел.

8. Найти сумму значений функции, больших 2 Z=sin(1/x)+5cos(1/(x-3))+x для x, изменяющегося на отрезке [-3,8] с шагом 1,123.

9. Найти n членов последовательности $x_1 = x_2 = x_3 = 1$; $x_k = x_{k-1} + x_{k-3}$.

10. Вычислить последовательность N чисел $A_0 = x$, $A_1 = 2$, $A_k = A_{k-1} - A_{k-2}$.

11. Для $x_1 = 0,3$ и $x_2 = -0,3$ найти $x_k = k + \sin(x_{k-2})$ для k, изменяющегося следующим образом: k = 3,4,...,14.

12. Составить таблицу перевода дюймов в сантиметры для расстояний от 1 до 13 дюймов с шагом 1.

13. Вывести на печать значения функции, меньшие 2, Z=sin(1/x)+5cos(x-3)+x для x, изменяющегося на отрезке [-7,4] с шагом 1.

14. Напечатать таблицу значений функции Y = tg(x/b)+x/(b-2) для x, изменяющегося от 0 до 10 с шагом 1 (b - произвольное число).

15. Вычислить N -й член последовательности $x_k = x_{k-2} - x_{k-1}$, $x_0 = 2,4$ $x_1=3,8$.

16. Составить таблицу перевода суток (от 1 до 7) в часы, минуты, секунды.

17. Вычислить N-й член последовательности $x_k = x_{k-1} + (2/3)x_{k-2} + 1$, $x_1=-1$, x2 =1,38.

18. Напечатать значения функции z = 1/(x-2)+1/(x-5)+ln(12,8-X) для x, изменяющегося на отрезке [-4,14] с шагом 1,152.

19. Вывести на печать отрицательные значения функции z=sin(5-x)/cos(x-2) для x, изменяющегося на отрезке [-6,13] с шагом 1,541 (учесть область допустимых значений функции).

20. Из N введенных с клавиатуры чисел напечатать кратные 3 и меньшие 58.

21. Ввести с клавиатуры N чисел. Напечатать те из них, которые принадлежат интервалу (1,11) и являются четными.

22. Из N введенных с клавиатуры чисел напечатать положительные, кратные 3.

23. Вывести на печать значения функции z = sin(x/(x-2)), находящиеся в интервале (-0,4;0,8) для x, изменяющегося от 8 до -6 с шагом 1,235.

24. Ввести с клавиатуры N чисел. Напечатать те из них, которые принадлежат интервалу (2;9).

25. Для геометрической прогрессии 2, 6, 18, 54, 162 ... определить первые n членов этой прогрессии.

26. Ввести с клавиатуры N чисел. Напечатать те из них, которые не принадлежат интервалу (1;5).

27. Найти n членов последовательности $x_1 = x_2 = x_3 = 1$; $x_k = x_{k-1} - 2x_{k-3}$.

28. Вычислить последовательность N чисел $A_0 = x, A_1 = 2$, $A_k = A_{k-1} + A_{k-2}$

29. Составить таблицу перевода килограммов (от 1 до 13) в граммы с шагом

30. Найти сумму значений функции Y=COS(x/A)+x/(A-2) для x, изменяющегося от 2 до 13 с шагом 1 (A - произвольное число).

МУ к ЛР 5

Исследование простых двухуровневых клиент-серверных приложений в C#

2.5.1 Цели занятия

Выработать умения и навыки составлять типовые программы решения задач на выбранном языке программирования, снабженные элементами графического интерфейса пользователя в виде клиент-серверных приложений.

2.5.2 Теоретические основы и пример выполнения

Для создания клиент-серверного приложения в C#, аналогично предыдущей ЛР, необходимо создать два класса Server и Client, используя технологию сокетов.

2.5.2.1 Программирование класса Server

Вначале в этом классе создадим серверное соединение и используем для этого систему сокетов.

Сокеты- это концепция сетевого программирования, когда существуют два вида «программных разъемов»- сокетов: клиентские и серверные.

Серверный сокет как розетка, которая «висит на стене» готовая к работе, в ожидании, когда к ней подключат штекер. Точно так же серверный сокет переходит в режим ожидания подключения на определенном адресе и определенном порту.

Клиентский сокет как вилка, которую втыкают в розетку. Как только клиент подключается к серверному сокету, информация начинает передаваться между ними.

Для того чтобы создать виртуальное подключение между клиентом и сервером, надо знать место, где находится нужный нам серверный сокет.

В нашем случае, создается слушатель событий подключений в сети по протоколу TCP IP – TcpListener, в качестве параметра которому передается так

называемая точка входа, которую определяет метод IPEndPoint из переданных ему IP адреса компьютера, на котором будет запущен сервер (пока это вариант localhost, т.е. IP 127.0.0.1) и номер порта Port, на котором создаваемой серверное подключение будет прослушивать подключения клиентов. Т.о., в конструкторе класса Server появится следующий слушатель (листинг 2.18).

Листинг 2.18- Серверный слушатель подключения клиентов

```
TcpListener listner;
        public Program(int Port)
        {
            listner = new TcpListener(new IPEnd-
Point(IPAddress.Parse("127.0.0.1"), Port));
            listner.Start();
            Console.WriteLine("Сервер запущен, ожидает подклю-
чений...");
. . . }
```

Видно, что после запуска прослушивания сетевых подключений выдается сообщение о том, что сервер готов и ожидает подключений.

Далее, нужно в это подключение с помощью метода AcceptTcpClient() и создать новый поток для него с помощью метода Thread из одноименной библиотеки, которую нужно подключить заранее: using System.Threading.

В качестве параметра этому методу передается метод clientThread, в котором для каждого нового клиента открывается TCP соединение в новом потоке (листинг 2.19).

Листинг 2.19- Метод создания нового TCP соединения для нового клиента

```
static void clientThread(Object StateInfo)
        {
           new Client((TcpClient)StateInfo);
        }
```

В итоге получим следующий текст класса Server (листинг 2.20).

Листинг 2.20- Класс Server

```
class Server
    {
       TcpListener listner;
        public Server (int Port)
        {
            listner = new TcpListener(new IPEnd-
Point(IPAddress.Parse("127.0.0.1"), Port));
            listner.Start();
            Console.WriteLine("Сервер запущен, ожидает подклю-
чений...");
            while (true)
            {

                TcpClient client = listner.AcceptTcpClient();
                // Создаем поток


     Thread Thread = new Thread(new ParameterizedThread-
Start(clientThread));
                // И запускаем этот поток, передавая ему при-
нятого клиента
                Thread.Start(client);
            }
        }

        static void clientThread(Object StateInfo)
        {
           new Client((TcpClient)StateInfo);
        }

        static void Main(string[] args)
        {
            new Program(12000);
```

```
        }
    }
```

Видно, что сюда же включен и главный метод модуля – метод Main, из которого происходит запуск всего приложения и сервер начинает «прослушивание» на порту №12000, который в качестве параметра передается конструктору класса.

2.5.2.2 Класс Client

Этот класс необходим для считывания запроса от клиентских приложений, поиска нужного контента согласно запросу и для его передачи в сеть клиентам.

Начнем с задания объектов для методов чтения из потока и записи в поток. Для этого создаются новые экземпляры соответствующих классов (листинг 2.21).

Листинг 2.21- Организация потокового чтения и записи

```
StreamReader sr = new StreamReader(client.GetStream()); // чтение из потока клиента

                StreamWriter sw = new StreamWriter(client.GetStream()); // запись в поток клиента

                sw.AutoFlush = true;
```

Этот пример клиент-серверного приложения обеспечивает запрос от клиента на сервер необходимого файла информации, передачу его клиенту, сохранение в темповском файле на клиенте и отображение в клиентской части приложения. Поэтому, определим какой файл необходим: прочитаем из потока в за-просе клиента значение строковой переменной str, конвертируем его в целочисленную переменную kluch, и по ее значению в структуре switch определим необходимый файл (листинг 2.22).

Листинг 2.22- Определение требуемого контента

```
// Принимаем передачу от клиента
```

```
                string cl = sr.ReadLine(); // Отображение типа
обучающего модуля
                Console.WriteLine(cl);
                string str = sr.ReadLine();
                int kluch = int.Parse(str); // по kluch определяем
нужный файл
                string FilePath = "C:/www/"; // составляем путь к
файлу
                switch (kluch)
                {
                    case 1:
                        {
                            FilePath += "L1.htm";
                            Console.WriteLine("Путь к файлу:   " +
FilePath);
                        }
                        break;

                    case 2:
                        {
                            FilePath += "L2.htm";
                            Console.WriteLine("Путь к файлу:   " +
FilePath);
                        }
                        break;
```

```
}
```

Видно, что весь контент преобразован в html формат и находится в соответствующих папках на сервере. Аналогично программируется определение оставшихся файлов обучающих материалов.

Далее нужно открыть найденный файл. Сделаем это, проверяя правильность его открытия с помощью блока try/catch и если возникнет ошибка открытия, выдадим соответствующее сообщение (листинг 2.23).

Листинг 2.23- Безопасное открытие файла контента

```
// Открываем файл, страхуясь на случай ошибки
            FileStream FS;
            try
            {
                FS = new FileStream(FilePath, FileMode.Open, FileAccess.Read, FileShare.Read);
                sw.WriteLine("Запрашиваемый файл открыт и готов к передаче");
            }
            catch (Exception)
            {
                // Если случилась ошибка, посылаем клиенту ошибку открытия файла
                sw.WriteLine("Ошибка открытия файла");
                return;
            }
```

Учитывая, что данные передаются в сети в формате байт, нужно зарезервировать буфер для чтения материалов из открытого файла. Затем, перенести

эти данные в буфер и через него передать их клиенту в ответе на запрос. Весь этот процесс будет выполняться до конца данных в открытом файле (листинг 2.24).

Листинг 2.24- Передача контента в ответном сообщении клиенту

```
// Задаем буфер для размещения данных из выбранного файла
            byte[] Buffer = new byte[1024];
            // Переменная для хранения количества байт, пере-
даваемых клиенту
            int Count;

            // Пока не достигнут конец файла
            while (FS.Position < FS.Length)
            {
                // Читаем данные из файла и пишем их в буфер
                Count = FS.Read(Buffer, 0, Buffer.Length);
                // И передаем их клиенту
                client.GetStream().Write(Buffer, 0, Count);
            }
            sw.WriteLine("Файл передан");
            // Закроем файл и соединение
            FS.Close();
            sw.WriteLine("Соединение прервано");
            client.Close();
```

Из листинга следует, что по завершению передачи данных клиенту мы закрываем файл и соединение.

2.5.3 Порядок проведения исследований

Алгоритм выполнения заданий лабораторной работы следующий:

- выбрать 5 задач по следующему правилу: номер по журналу- первая задача; номер каждой последующей задачи определяется прибавлением цифры 3 к номеру первой задачи, который только что вычислили (если достигнуто окончание списка вариантов здач, то перейти в его начало);
- составить классы Swrver и Client и установить соединение между ними;
- составить методы (функции) решения всех задач, поместить их в класс Server. Все исходные данные вводить в консоли клиента, передавать их на сервер, а полученные результаты решений задач выводить на экран в консоли клиента;
- оформить отчет для всего приложения вцелом, включив в него задание, блок-схему алгоритма (в электронном виде), текст программы и skrinshert результата выполнения каждой задачи и представить его на проверку.

2.5.4 Варианты заданий

1. Вывести на печать положительные значения функции y=sin(x)+5cos(x-2) для x изменяющегося на отрезке [-5, 12] с шагом 1,2.

2. Вывести на печать значения функции z=tg(2x)-sin(x) для x изменяющегося на отрезке [-3, 3] с шагом 0,3.

3. Ввести с клавиатуры и напечатать модули N чисел; если введено отрицательное число, ввод и печать прекратить.

4. Вывести на печать значения функции z=ln(x)+tg(2x), большие 1, для x изменяющегося на отрезке [3, 8] с шагом 0,9.

5. Определить, является ли натуральное число N степенью числа 5 или нет.

6. Напечатать значения функции y=ln(x+12/x),где значения x вводятся с клавиатуры. При вводе числа, не входящего в область определения функции, вычисления прекратить.

7. Напечатать значения функции y=ln(x-1/x),где значения x вводятся с клавиатуры. При вводе числа,не входящего в область определения функции, вычисления прекратить.

8. Для x из интервала (-2;8) с шагом 0,75 вычислить y=(4x-3x+tg(x))/A, где A вводится с клавиатуры.

9. Вывести на печать значения функции z=sin(x)+cos(x),находящиеся в интервале (-0,2; 0,8) для x изменяющегося на отрезке [4,-6] с шагом 0,91.

10. Дано натуральное число N. Получить наименьшее число вида 4^k, большее N.

11. Для x из интервала (2;8) с шагом 0,75 вычислить y=(4x-3x+cos(x))/A, где A вводится с клавиатуры.

12. Найти первый член последовательности ln(9n)/(n*n), меньший 1, для n изменяющегося следующим образом: n=1,2,3... .

13. Определить, является ли натуральное число N степенью числа 3 или нет.

14. Вывести на печать отрицательные значения функции z=cos(x)-5sin(x-2) для x изменяющегося на отрезке [-3, 11] с шагом 0,9.

15. Ввести с клавиатуры и напечатать квадраты N чисел, если введено кратное 3 положительное число, ввод и печать прекратить.

16. Вывести на печать отрицательные значения функции z=tg(x)+5cos(x-2) для x изменяющегося на отрезке [12, 1] с шагом 1,2.

17. Ввести с клавиатуры и напечатать N чисел, если введено равное нулю или кратное 2 число, ввод и печать прекратить.

18. Вывести на печать значения функции z=ln(|x|)+tg(2x), большие 2 для x изменяющегося на отрезке [3, -8] с шагом 0,9.

19. Найти первый отрицательный член последовательности sin(tg(n/2)) для n изменяющегося на следующим образом: n=1,2,3... .

20. Напечатать значения функции y=ln(x+12/x),где значения x вводятся с клавиатуры. При вводе числа, не входящего в область определения функции, вычисления прекратить.

21. Найти первую цифру в целом положительном числе.

22. Дано натуральное число N. Получить наибольшее число вида 3^k, меньшее N.

23. Вывести на печать значения функции z=sin(x)+cos(x),находящиеся в интервале (-0,3;0,7) для x изменяющегося на отрезке [-4,6] с шагом 0,91.

24. Дано натуральное число N. Получить наименьшее число вида 5^k, большее N.

25. Для x из интервала (-2;8) с шагом 0,75 вычислить y=(4x-3x+tg(x))/A, где A вводится с клавиатуры.

26. Найти первый член последовательности ln(9n/(n*n+1), меньший 0, для n изменяющегося на следующим образом: n=1,2,3... .

27. Определить, является ли натуральное N степенью числа 4 или нет.

28. Вывести на печать положительные значения функции z=sin(x)-5cos(x-2) для x изменяющегося на отрезке [5,-12] с шагом 1,2.

29. Напечатать значения функции $Y = \sqrt{2x^2 - x^3}$ для произвольных *x,* вводимых с клавиатуры. При вводе числа, не входящего в область определения функции, ввод и печать прекратить.

30. Найти первый отрицательный член последовательности cos(ctg(n)) для n изменяющегося на следующим образом: n=1,2,3... .

МУ к ЛР 6

Показатели качества в клиент-серверных приложениях в C#. Проектирование архитектуры и дизайна клиент-серверных приложений

2.6.1 Цели занятия

Практически освоить основные правила проектирования архитектуры и дизайна приложений. Практически потренироваться в составлении интерфейсов клиентской части.

2.6.2 Теоретические основы и пример выполнения

В состав клиент-серверных приложений, как составного программного обеспечения (ПО), очевидно, входят серверная и клиентская части. Как правило, обе эти составляющие приложения разрабатываются как интерфейсные продукты, т.е. содержащие специальные модули (блоки) для обеспечения связи человека и вычислительной среды. От этого зависит, в конечном итоге, насколько простым или сложным будет использование приложения, а также его оптимальное функционирование.

Поэтому, важно научиться составлять удобные в использовании и функционально оптимальные клиент-серверные продукты.

Тема данного практического занятия больше относится к реализации клиентских частей приложения. Однако, правила разработки приложений относятся в равной степени и к созданию серверной части приложения.

Сейчас при разработке ПО большое значение придается его архитектуре. И это не случайно. У любого программного продукта (ПП) есть свое конкретное назначение, которое определяется тем, как с помощью ПП пользователь может удовлетворить потребности в трех областях: пользовательской (интерфейс), предметной (бизнес, исследования, потребительская и т.д.) и в системной с точки зрения структурности продукта. Поэтому, удачная архитектура приложения

делает его гибким, простым в использовании и сопровождении и высоко функциональным продуктом и наоборот.

Под **архитектурой** клиент-серверной системы будем понимать ее структуру, включающую программные элементы, видимые извне свойства этих элементов и взаимоотношения между ними [5].

Типовая архитектура для каждой из частей клиент-серверной системы представлена на рисунке 2.14.

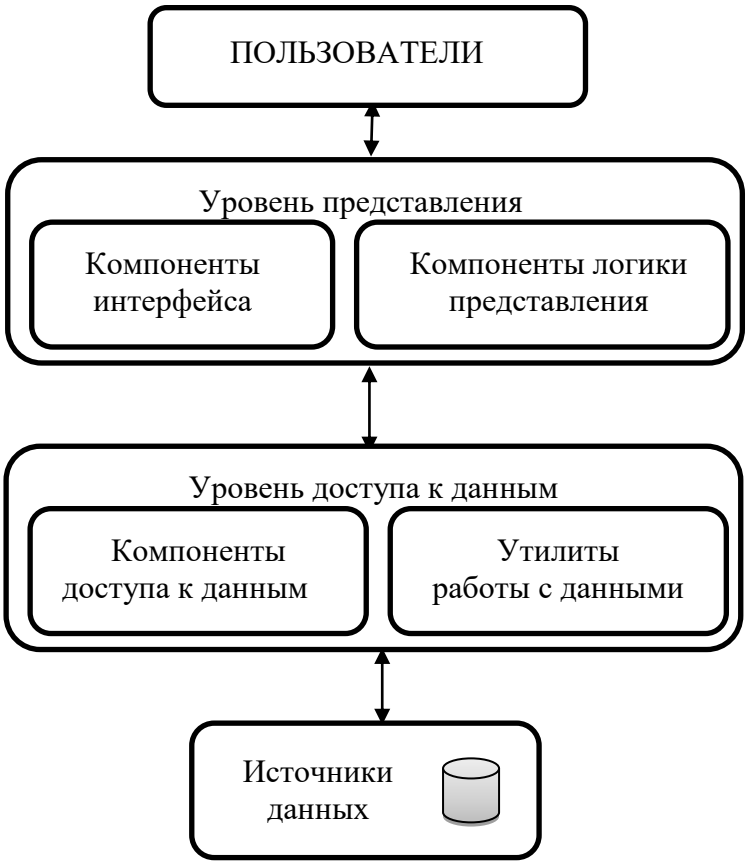

Рисунок 2.14-Типовая структура клиент-серверной системы

Для серверной части характерно еще и то, что ее пользователями могут быть не только люди, а и клиентские части системы.

Дадим краткие пояснения приведенной структуры.

Источники данных – это исходный уровень структуры, т.е. непосредственно данные, которые необходимо передавать или преобразовывать, если это

структура серверной части системы, или можно просто использовать, как говорят: «на стороне клиента». Традиционно, все данные в клиент-серверных системах располагаются на серверной части в виде базы данных, клиентские части системы имеют к ним доступ при обращении на сервер. Однако, структура системы позволяет размещать и хранить данные и на клиентских частях, если это необходимо [6].

Уровень доступа к данным обеспечивает работу с данными, поэтому он включает компоненты доступа к данным и утилиты работы с данными. Под «доступом к данным» подразумеваются алгоритмы их записи, извлечения и обработки. Соответственно этому и выбираются утилиты, их реализующие. Так, если используется база данных, то этот уровень реализуется в виде приложения, управляющего базой данных, если же данные просто размещаются по какому-то принципу на дисках сервера, то и приложение реализует поисковые алгоритмы доступа к ним.

Уровень представления служит для необходимой обработки данных и представления их в требуемом виде согласно клиентским запросам, а также для обеспечения возможности пользователю вообще работать с приложением. Поэтому этот уровень реализует алгоритмы обработки данных и компоненты пользовательского интерфейса. Обычно, это больше касается пользовательских частей системы, т.к. именно через них пользователи работают с клиент-серверными системами, однако и на серверной части возможны минимальные реализации этого уровня для настройки и эксплуатации серверной ее части. Сюда, как показано на рисунке 2.1, входят компоненты интерфейса пользователя и логики представления. Компоненты логики представления необходимы для реализации сценария подачи необходимой информации пользователю. Интерфейсные компоненты, в свою очередь, обеспечивают представление нужной информации пользователю.

Настоящее практическое занятие включает отработку всех описанных уровней реализации клиентского приложения. При этом, особое внимание уделяется уровню представления, т.к. уровень доступа к данным и сами данные

могут на клиентской стороне клиент-серверного приложения отсутствовать вообще.

Клиентскую часть приложения необходимо будет разработать, используя Visual Studio- интегрированную среду разработки программных продуктов, снабженную всеми необходимыми средствами написания, отладки и компиляции программ [7]. Воспользуемся языком программирования C# потому, что именно он представляет массу инструментов для создания интерфейсных приложений [8].

2.6.2.1 Основные элементы графического интерфейса пользователя

Для создания клиентских частей клиент-серверных приложений используется обширная библиотека Form языка C# для приложений типа Windows Forms. Принцип составления оконного интерфейса прост- создается пустая форма, которая наследует (использует) библиотеку готовых решений Form. Далее, перетаскивая на эту форму необходимые элементы управления и отображения составляется клиентская часть для решения задачи, указанной на практическом занятии.

Элементов управления в указанной библиотеке множество, рассмотрим только часто употребляемые и необходимые для решения поставленных на занятии задач.

Основными элементами форм являются:

- label – метка для отображения справочной информации;
- textBox – текстовое поле для вводимых и выводимых данных;
- button – кнопка для исполнения требуемых действий.

Приведем пример решения одной из задач и покажем как программируются указанные элементы интерфейса.

2.6.2.2 Пример выполнения задания

Задание: даны действительные числа x,y. Вычислить значение функции z=ln(x)-x/y.

Из задания следует, что необходимо будет на форму ввести две переменные и вычислить значение функции для их определенных величин. Это задается видом искомой функции. В данном случае – это логарифм, а областью определения этой функции являются только положительные числа, большие нуля. Т.е., функция имеет смысл только для x,y>0. На форме также, очевидно, должна быть кнопка для выполнения расчетов и метка, либо текстовое поле для отображения результата.

Изначально создается проект Windows Forms с названием Form1 (это имя будет сформировано по умолчанию и может быть изменено). Далее, используя панель инструментов Visual Studio в режиме конструктора размещаем элементы интерфейса на форму (рисунок 2.15).

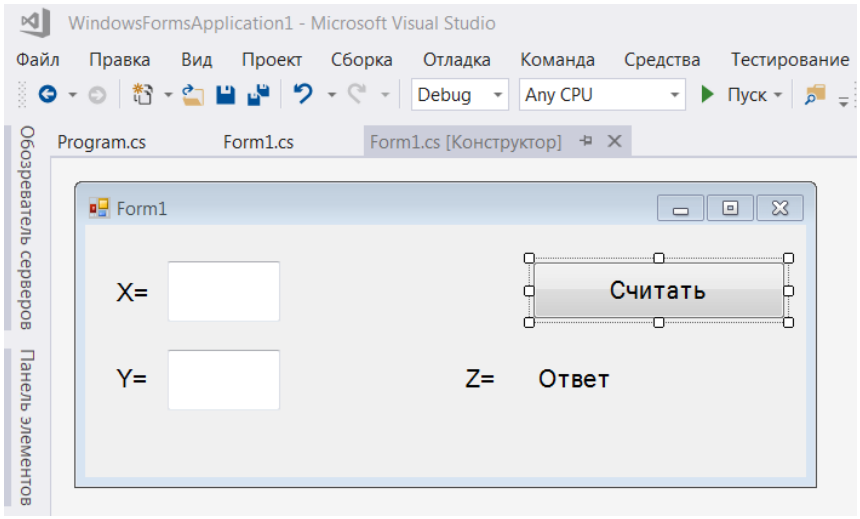

Рисунок 2.15- Конструирование формы интерфейса

При этом необходимый код программы создается компоновщиком форм автоматически.

Далее необходимо создать метод для вычисления функции Z. Дело в том, что весь проект содержит два класса: Program и Form1 (рисунок 2.16).

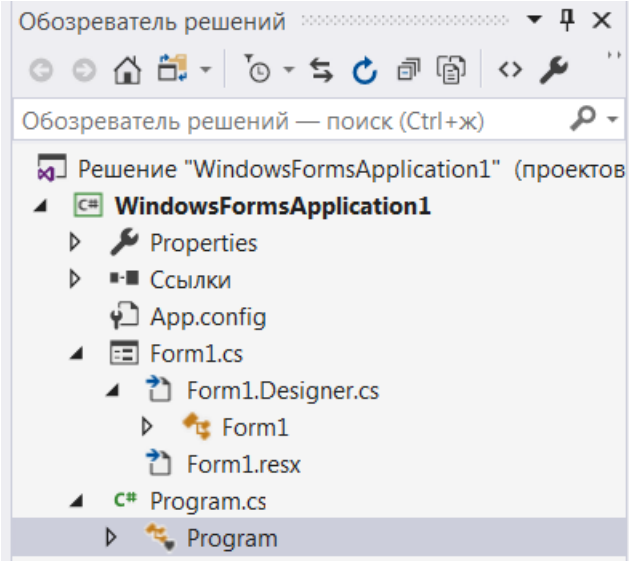

Рисунок 2.16- Состав клиентской части

Program нужен для запуска всего проекта и формирования разработанной формы, он содержит основной метод всего проекта – метод Main(). Класс Form1 содержит все необходимое для задания и отображения элементов формы. Коды этих классов, которые содержатся в файлах Program.cs и Form1.cs, представлены в листинге 2.25.

Листинг 2.25- Коды классов проекта

```
namespace WindowsFormsApplication1
{
    static class Program  // код класса Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
        public static double Metod(int x, int y) // Метод для вы-
числения Z
        {
            double z=0;
            if (y != 0)
```

```
            {
                z = Math.Log(x) - x / y;
            }
            else {
                z = 0;
            }
            return z;
        }
    }
    }
```

```
using System;
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 WindowsFormsApplication1
{
    public partial class Form1 : Form // Код класса Form1
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e) //
Обработчик кнопки
        {
label4.Text = Con-
vert.ToString(Program.Metod(Convert.ToInt16(textBox1.Text),
                            Convert.ToInt16(textBox2.Text)));
        }

    }
    }
```

В классе Program составлен метод Metod(int x, int y), в котором вычисляется искомая функция. При этом, вычисления осуществляются по нажатию на кнопку «Считать», обработчик которой содержится в классе Form1. Обработчик события – это специальный метод (или еще говорят: «функция»), который создается автоматически, если дважды кликнуть по кнопке (или по другому эле-

менту интерфейса) на форме в режиме конструктора. Этот метод как бы «слушает» изменение состояния выбранного элемента интерфейса и если, например, кнопка будет нажата, то выполнится программный код, содержащийся в ее обработчике событий (листинг 2.1).

В обработчике кнопки находится строка присваивания значения метке label4, которая служит для отображения результатов вычислений. Оператор label4.Text указывает на то, что объект label4 обращается к методу Text, чтобы поместить текстовую переменную в метку. Вообще, в C# ввод/вывод возможен только для текста. Поэтому, правая часть оператора присваивания содержит вызов метода Metod(int x, int y), который расположен в классе Program и содержит два параметра- переменные X и У. Эти значения конвертируются сначала из текста текстовых полей textBox1 и textBox2, куда они вводятся на форму, а потом результат выполнения всего метода конвертируется в текст и выводится в метку на форме как результат всей задачи.

2.6.3 Порядок выполнения задач ПЗ

Алгоритм выполнения задач практического занятия следующий:

- выбрать 5 задач по следующему правилу: номер по журналу- первая задача; номер каждой последующей задачи определяется прибавлением цифры 3 к номеру первой задачи, который только что вычислили (если достигнуто окончание списка вариантов здач, то перейти в его начало);
- составить методы (функции) решения всех задач, поместить их в класс Form1 и снабдить пользовательским интерфейсом в Windows Forms. Это клиентская часть будущего клиент-серверного приложения, которое разрабатывается на следующих занятиях;
- аформить отчет для всего приложения вцелом, включив в него задание, блок-схему алгоритма (в электронном виде), текст программы и skrinshert результата выполнения каждой задачи и представить его на проверку.

2.6.4 Варианты заданий

Решить задачи, используя оператор условного перехода if:

1. Даны действительные числа A,B,C,D. Выяснить, можно ли уместить прямоугольник со сторонами A,B внутри прямоугольника со сторонами C,D.

2. Даны действительные числа x, y, z. Найти минимальное из них.

3. Даны действительные положительные числа A,B,C. Выяснить, пройдет ли кирпич с ребрами A,B,C в прямоугольное отверстие со сторонами x,y.

4.Определить, лежит ли точка D(c,b), где $c=\sqrt{a1+a2}$, $b = a1 + 0{,}7a3$, внутри прямоугольника со сторонами 5 и 10 (a1,a2,a3 - произвольные числа)?

5. Выяснить, существует ли треугольник с координатами вершин A(x1,y1), B(x2,y2), C(x3,y3), если да, то найти его площадь.

6. Даны действительные числа A,B,C. Проверить выполняются ли неравенства A<B<C, если да, то присвоить A=B+C иначе A=C-B.

7. Даны действительные числа x,y. Вычислить значение функции z=log(x-y)-x/y

8. На плоскости расположена окружность радиуса R с центром в начале координат. Определить положение точки x с координатами (A,B) относительно окружности.

9. Даны круг радиуса R и квадрат со стороной A. Определить их взаимное положение.

10. Вывести на печать переменные A,B,C в порядке их возрастания

11. Проверить, какие из чисел A,B,C,D принадлежат интервалу (1,25).

12. Даны действительные числа A,B. Если они оба отрицательные, то заменить каждое из них его квадратом, иначе положительные из них увеличить в два раза.

13. Выяснить, существует ли треугольник с координатами вершин A(x1,y1), B(x2,y2),C(x3,y3).

14. Даны действительные числа x,y. Вычислить значение функции z=log(x/y)-1/x

15. Даны действительные числа A,B. Если они оба неотрицательные, то заменить каждое из них его кубом, иначе отрицательные из них заменить их модулями.

16. Даны площадь квадрата S1 и круга S2. Определить поместится ли круг в квадрат и наоборот.

17. На плоскости расположена окружность радиуса R с центром в начале координат. Определить, лежат ли точки A(x1,y1) и B(x2,y2) на окружности.

18. Составить программу вычисления корней системы уравнений с двумя неизвестными методом Крамера.

$$\begin{cases} a1x + b1y = c1 \\ a2x + b2y = c2 \end{cases}$$ Проверить, что главный определитель системы не должен быть равен 0.

19. Даны действительные числа A,B,C,D. Выяснить, можно ли уместить прямоугольник со сторонами A,B внутри прямоугольника со сторонами C,D.

20. Вывести на печать переменные A,B,C в порядке их убывания

21. Даны действительные числа x, y, z. Найти максимальное из них.

22. Проверить, какие из чисел A,B,C,D не принадлежат интервалу (3,15).

23. Даны действительные числа x,y. Вычислить значение функции z=ln(x)-x/y.

24. Даны действительные числа A,B. Если они имеют разные знаки, то напечатать их произведение, иначе напечатать их квадраты.

25. Выяснить, существует ли треугольник с длинами сторон A,B,C. Если да, то найти его площадь.

26. Даны действительные числа x,y. Вычислить значение функции z=arcsin(x)-y.

27. Даны действительные числа x,y,z. Получить максимальное из них по модулю.

28. Даны действительные числа x,y. Вычислить значение функции z=arcsin(x+y).

29. На каком из интервалов $(-\infty;k1),(k1;k2),(k2;+\infty)$ лежит точка x? Где k1, k2, x – вводимые произвольные числа, причем k1<k2.

30. Лежат ли обе точки D(a1;b1) и C(a2;b2) внутри круга радиуса R с центром в начале координат? Если такой точки нет, выдать соответствующее сообщение.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Дубаков А.А. Сетевое программирование: учебное пособие / А.А. Дубков – СП: НИУ ИТМО, 2013. – 248 с.

2 Грамотная клиент-серверная архитектура: как правильно проектировать и разрабатывать web API. [Электронный ресурс] // [сайт] [2018], URL: https://tproger.ru/articles/web-api/. (дата обращения: 03.05.2018).

3 Технологии разработки клиент-серверных приложений. [Электронный ресурс] // [сайт] [2018], URL: https://bourabai.ru/dbt/client2.htm. (дата обращения: 05.05.2018).

4 Сенина А.А., Тузовский А.Ф Обзор основных современных технологий разработки web-приложений. [Электронный ресурс] // [сайт] [2018], URL: https://core.ac.uk/download/pdf/53095773.pdf. (дата обращения: 03.05.2018).

5 Басс Лен, Клементс Пол, Кацман Рик. Архитектура программного обеспечения на практике. 2-е издание. — СПб.: Питер, 2006. — 575 с.

6 П. Дейтел, Х. Дейтел, Э. Дейтел, М. Моргано. Android для программистов: создаём приложения. — СПб.: Питер, 2013. — 560 с.

7 Краткое описание Visual Studio [Электронный ресурс] // [сайт] [2018], URL: http://habrahabr.ru. (дата обращения: 10.05.2018).

8 Фленов М. Е. Библия C#. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2011. — 560 с.:

Приложение А

Код программы web – приложения ЛР № 1

```
Код стартовой страницы Page1
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
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;

namespace WpfBrowserApplication1
{
    /// <summary>
    /// Interaction logic for Page1.xaml
    /// </summary>
    public partial class Page1 : Page
    {
        public Page1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, RoutedEventArgs e)
        {
           //string pup = System.IO.Directory.GetCurrentDirectory();
          //  string task = zad.Text
           string pup = "I:\\Pril\\Lex";
           webBrowser1.Navigate(pup + "\\" +"1.mht");
          // webBrowser1.Navigate("E:\\Lobz\\Laba3\\Lex\\1.html");
```

```
}

private void button2_Click(object sender, RoutedEventArgs e)
{
    string pup = "I:\\Pril\\Lex";
    webBrowser1.Navigate(pup + "\\" + "2.mht");
}

private void button3_Click(object sender, RoutedEventArgs e)
{
    string pup = "I:\\Pril\\Lex";
    webBrowser1.Navigate(pup + "\\" + "3.mht");
}

private void button4_Click(object sender, RoutedEventArgs e)
{
    string pup = "I:\\Pril\\Lex";
    webBrowser1.Navigate(pup + "\\" + "4.mht");
}

private void button5_Click(object sender, RoutedEventArgs e)
{
    string pup = "I:\\Pril\\Lex";
    webBrowser1.Navigate(pup + "\\" + "p1.mht");
}

private void button6_Click(object sender, RoutedEventArgs e)
{
    if (comboBox1.SelectedIndex == 0)
    {
        string pup = "I:\\Pril\\Lex";
        webBrowser1.Navigate(pup + "\\" + "Lab1.mht");
    }

    if (comboBox1.SelectedIndex == 1)
    {
        string pup = "I:\\Pril\\Lex";
        webBrowser1.Navigate(pup + "\\" + "Lab2.mht");
```

```
            }

            if (comboBox1.SelectedIndex == 2)
            {
                string pup = "I:\\Pril\\Lex";
                webBrowser1.Navigate(pup + "\\" + "Lab3.mht");
            }
        }

        private void button7_Click(object sender, RoutedEventArgs e)
        {
            if (comboBox1.SelectedIndex == 0)
            {
                NavigationService.Navigate(new    Uri("/Page2.xaml",
UriKind.Relative));
            }

            if (comboBox1.SelectedIndex == 1)
            {
                NavigationService.Navigate(new    Uri("/Page3.xaml",
UriKind.Relative));
            }

            if (comboBox1.SelectedIndex == 2)
            {
                NavigationService.Navigate(new    Uri("/Page4.xaml",
UriKind.Relative));
            }
        }

        private void comboBox1_SelectionChanged(object sender, Se-
lectionChangedEventArgs e)
        {

        }
    }
}
```

Код страницы Page2

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
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;

namespace WpfBrowserApplication1
{
    /// <summary>
    /// Логика взаимодействия для Page2.xaml
    /// </summary>
    public partial class Page2 : Page
    {
        public Page2()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, RoutedEventArgs e)
        {
            NavigationService.Navigate(new        Uri("/Page1.xaml",
UriKind.Relative));
        }

        private void button5_Click(object sender, RoutedEventArgs e)
        {
            double A = 1.5, B = 2, C = 2.5;
            double x = double.Parse(textBox1.Text);
```

```
                double z = (Math.Pow(x, 2) + Math.Abs(B * x - 3 * C)) /
(Math.Log(Math.Pow(x, 3) + B * C + A));
                label9.Content    =    "Z    =    (X^2    +    |B*x    -
3*C|)/(Log(x^3)+B*C+A) = " + Math.Round(z, 3);
            }

            private void button2_Click(object sender, RoutedEventArgs e)
            {
                double opr, a1, a2, b1, b2, c1, c2;

                a1 = Convert.ToInt16(textBox2.Text);
                a2 = Convert.ToInt16(textBox3.Text);
                b1 = Convert.ToInt16(textBox5.Text);
                b2 = Convert.ToInt16(textBox4.Text);
                c1 = Convert.ToInt16(textBox7.Text);
                c2 = Convert.ToInt16(textBox6.Text);
                opr = (a1 * b2) - (b1 * a2);

                if (opr == 0)
                {
                    label16.Content = "Определитель равен '0'";
                }
                else
                {
                    label16.Content = "Определитель не равен '0'";
                }
            }
        }
    }
```

Код страницы Page3

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
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;

namespace WpfBrowserApplication1
{
    /// <summary>
    /// Логика взаимодействия для Page3.xaml
    /// </summary>
    public partial class Page3 : Page
    {
        public Page3()
        {
            InitializeComponent();
        }

        private void button5_Click(object sender, RoutedEventArgs e)
        {
            double x1, x2, x;
            double z;

            x1 = Convert.ToDouble(textBox1.Text);
            x2 = Convert.ToDouble(textBox2.Text);
            for (x = x1; x < x2; x++)
            {
                z = (1 / (x + 2)) + (1 / (x + 5)) + Math.Log(12, 8 -
x);
                textBox3.Text += "Z [" + x + "] = " + Math.Round(z,
2) + "\r\n";
            }
        }

        private void button1_Click(object sender, RoutedEventArgs e)
        {
            double y1, y2, y;
```

```
            double Z = 3;

            //y1 = Convert.ToDouble(textBox6.Text);
            y1 = double.Parse(textBox10.Text);
            y2 = Convert.ToDouble(textBox5.Text);

            y = y1;
            while ((Z > 2) && (y != y2))
            {
                Z = Math.Log(Math.Abs(y)) + Math.Tan(2 * y);
                y = y - 0.9;
                textBox6.Text += "Z [" + y + "] = " + Math.Round(Z,
2) + "\r\n";
            }
        }

        private void button2_Click(object sender, RoutedEventArgs e)
        {
            double k1, k2, k;
            double Z = 3;

            k1 = Convert.ToDouble(textBox7.Text);
            k2 = Convert.ToDouble(textBox8.Text);

            k = k1;

            do
            {
                Z = Math.Cos(k) - 5 * Math.Sin(k - 2);
                k = k - 0.9;
                textBox9.Text += "Z [" + k + "] = " + Math.Round(Z,
2) + "\r\n";
            }
            while ((Z < 0) && (k != k2));
        }

        private void button3_Click(object sender, RoutedEventArgs e)
        {
```

```
                NavigationService.Navigate(new        Uri("/Page1.xaml",
UriKind.Relative));
            }
        }
    }

    Код страницы Page4
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Text;
    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;

    namespace WpfBrowserApplication1
    {
        /// <summary>
        /// Логика взаимодействия для Page4.xaml
        /// </summary>
        public partial class Page4 : Page
        {
            public Page4()
            {
                InitializeComponent();
            }

            private void Page_Loaded(object sender, RoutedEventArgs e)
            {

            }
```

```
        private void button5_Click(object sender, RoutedEventArgs e)
        {
            double i, k, sum = 0;

            textBox1.Text += "B = ";
            for (i = 1; i < 8; i++)
            {
                if ((i % 2) == 0)
                {
                    k = Math.Sin(i) + 3;
                    textBox1.Text += Math.Round(k, 0) + " ";
                    sum = sum + k;
                }
                else
                {
                    k = i + Math.Cos(i - 1);
                    textBox1.Text += Math.Round(k, 0) + " ";
                    sum = sum + k;
                };
            }
            textBox3.Text = Convert.ToString(Math.Round(sum, 0));
        }

        private void button3_Click(object sender, RoutedEventArgs e)
        {
            NavigationService.Navigate(new        Uri("/Page1.xaml",
UriKind.Relative));
        }

        private void button1_Click(object sender, RoutedEventArgs e)
        {
            textBox2.Text = "";
            if (Convert.ToBoolean(radioButton1.IsChecked == true))
            {
                double[,] brr = new double[2, 2];
                double k = 2;

                for (int i = 0; i < 2; i++)
```

```
    {
        for (int j = 0; j < 2; j++)
        {
            k = 1 / (k * k);
            k++;
            textBox2.Text += Math.Round(k, 0) + " ";
        }
        textBox2.Text += "\r" + "\n";
    }

}

if (radioButton2.IsChecked == true)
{
    double[,] brr = new double[3, 3];
    double k = 2;

    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            k = 1 / (k * k);
            k++;
            textBox2.Text += Math.Round(k, 0) + " ";
        }
        textBox2.Text += "\r" + "\n";
    }
}

if (radioButton3.IsChecked == true)
{
    double[,] brr = new double[4, 4];
    double k = 2;

    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 4; j++)
        {
```

```
                    k = 1 / (k * k);
                    k++;
                    textBox2.Text += Math.Round(k, 0) + " ";
                }
                textBox2.Text += "\r" + "\n";
            }
        }
    }
}
```