

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ  
ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего профессионального образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

---

**В.Б. Немировский**

## **ЛАБОРАТОРНЫЙ ПРАКТИКУМ ПО ИНФОРМАТИКЕ**

*Рекомендовано в качестве учебного пособия  
Редакционно-издательским советом  
Томского политехнического университета*

Издательство  
Томского политехнического университета  
2018

# СОДЕРЖАНИЕ

<b>ЛАБОРАТОРНАЯ РАБОТА №1 ИЗУЧЕНИЕ СРЕДЫ РАЗРАБОТКИ VISUAL STUDIO .....</b>	<b>5</b>
1.1. ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТЧИКА VISUAL STUDIO	5
1.2. НАСТРОЙКА ФОРМЫ	8
1.3. РАЗМЕЩЕНИЕ ЭЛЕМЕНТОВ УПРАВЛЕНИЯ НА ФОРМЕ	8
1.4. РАЗМЕЩЕНИЕ СТРОКИ ВВОДА	9
1.5. РАЗМЕЩЕНИЕ НАДПИСЕЙ	10
1.6. НАПИСАНИЕ ПРОГРАММЫ ОБРАБОТКИ СОБЫТИЯ	10
1.7. НАПИСАНИЕ ПРОГРАММЫ ОБРАБОТКИ СОБЫТИЯ НАЖАТИЯ КНОПКИ	10
1.8. НАПИСАНИЕ ПРОГРАММЫ ОБРАБОТКИ СОБЫТИЯ ЗАГРУЗКИ ФОРМЫ	11
1.9. ЗАПУСК И РАБОТА С ПРОГРАММОЙ	12
1.10. ДИНАМИЧЕСКОЕ ИЗМЕНЕНИЕ СВОЙСТВ	13
1.11. ВЫПОЛНЕНИЕ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ	13
Индивидуальные задания	14
<b>ЛАБОРАТОРНАЯ РАБОТА №2 ЛИНЕЙНЫЕ АЛГОРИТМЫ.....</b>	<b>15</b>
2.1. СТРУКТУРА ПРИЛОЖЕНИЯ	15
2.2. РАБОТА С ПРОЕКТОМ	17
2.3. ОПИСАНИЕ ДАННЫХ	18
<i>Целочисленные типы</i>	19
<i>Типы с плавающей точкой</i>	19
<i>Символьные типы</i>	20
<i>Логический тип данных</i>	20
2.4. Ввод/вывод данных в ПРОГРАММЕ	21
2.5. АРИФМЕТИЧЕСКИЕ ДЕЙСТВИЯ И СТАНДАРТНЫЕ ФУНКЦИИ	22
2.6. ПРИМЕР НАПИСАНИЯ ПРОГРАММЫ	23
2.7. ВЫПОЛНЕНИЕ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ	25
Индивидуальные задания	25
<b>ЛАБОРАТОРНАЯ РАБОТА №3 РАЗВЕТВЛЯЮЩИЕСЯ АЛГОРИТМЫ .....</b>	<b>27</b>
3.1. ЛОГИЧЕСКИЕ ПЕРЕМЕННЫЕ И ОПЕРАЦИИ НАД НИМИ	27
3.2. УСЛОВНЫЕ ОПЕРАТОРЫ	28
3.3. КНОПКИ-ПЕРЕКЛЮЧАТЕЛИ	32
3.4. ПРИМЕР НАПИСАНИЯ ПРОГРАММЫ	32
<i>Создание формы</i>	32
<i>Создание обработчиков событий</i>	33
Индивидуальные задания	34
<b>ЛАБОРАТОРНАЯ РАБОТА №4 ЦИКЛИЧЕСКИЕ АЛГОРИТМЫ И ОДНОМЕРНЫЕ МАССИВЫ .....</b>	<b>35</b>
4.1. ОПЕРАТОРЫ ОРГАНИЗАЦИИ ЦИКЛОВ	35
4.2. ЦИКЛ С ПРЕДУСЛОВИЕМ	35
4.3. ЦИКЛ С ПОСТУСЛОВИЕМ	36
4.4. ЦИКЛ С ПАРАМЕТРОМ	37
4.5. СРЕДСТВА ОТЛАДКИ ПРОГРАММ	37
4.6. РАБОТА С МАССИВАМИ	39
4.7. СЛУЧАЙНЫЕ ЧИСЛА	41
4.8. ПОРЯДОК ВЫПОЛНЕНИЯ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ	42
Индивидуальные задания	43
Основные алгоритмы обработки массивов данных	45
<i>Накопление</i>	45
<i>Счетчик</i>	46
<i>Поиск экстремума</i>	46

Алгоритм обмена	47
Сортировка	47
<b>ЛАБОРАТОРНАЯ РАБОТА №5 РАБОТА СО СТРОКАМИ.....</b>	<b>48</b>
5.1. КЛАССЫ И ОБЪЕКТЫ	48
5.2. ДИНАМИЧЕСКОЕ СОЗДАНИЕ ОБЪЕКТОВ	48
5.3. ОБЛАСТЬ ВИДИМОСТИ	50
5.4. СТРОКОВЫЙ ТИП ДАННЫХ	51
5.5. БОЛЕЕ ЭФФЕКТИВНАЯ РАБОТА СО СТРОКАМИ	52
5.6. ЭЛЕМЕНТ УПРАВЛЕНИЯ ListBox	53
5.7. ПОРЯДОК ВЫПОЛНЕНИЯ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ	53
Индивидуальные задания	54
<b>ЛАБОРАТОРНАЯ РАБОТА №6 ЗНАКОМСТВО С ИНТЕРФЕЙСОМ И ОСНОВНЫМИ ВОЗМОЖНОСТЯМИ МАТЕМАТИЧЕСКОГО ПАКЕТА MATHCAD .....</b>	<b>56</b>
6.1. ОСНОВНЫЕ ПОНЯТИЯ, СРЕДСТВА И ЭЛЕМЕНТЫ ИНТЕРФЕЙСА	56
Индивидуальные задания	59
<b>ЛАБОРАТОРНАЯ РАБОТА №7 ПОСТРОЕНИЕ ГРАФИКА ТАБУЛИРОВАННОЙ ФУНКЦИИ .....</b>	<b>60</b>
7.1. ПОВТОРЯЮЩИЕСЯ ВЫЧИСЛЕНИЯ, ДИСКРЕТНЫЕ ПЕРЕМЕННЫЕ, ПОСТРОЕНИЕ ГРАФИКОВ	60
7.2. ПОРЯДОК ВЫПОЛНЕНИЯ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ	62
Индивидуальные задания	63
<b>ЛАБОРАТОРНАЯ РАБОТА №8 РЕШЕНИЕ СИСТЕМЫ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ В MATHCAD .....</b>	<b>64</b>
8.1. МАССИВЫ: ВЕКТОРЫ И МАТРИЦЫ	64
8.2. РЕШЕНИЕ СИСТЕМЫ УРАВНЕНИЙ	65
Индивидуальные задания	67
<b>ЛАБОРАТОРНАЯ РАБОТА №9 СИМВОЛЬНАЯ МАТЕМАТИКА. ВЫЧИСЛЕНИЕ ПРОИЗВОДНЫХ ПЕРВОГО И ВЫСШИХ ПОРЯДКОВ В MATHCAD .....</b>	<b>67</b>
9.1. СИМВОЛЬНЫЕ ВЫЧИСЛЕНИЯ. ВЫЧИСЛЕНИЕ ПРОИЗВОДНЫХ	67
Индивидуальные задания	68
<b>ЛАБОРАТОРНАЯ РАБОТА №10 СИМВОЛЬНАЯ МАТЕМАТИКА. ВЫЧИСЛЕНИЕ НЕОПРЕДЕЛЁННЫХ И ОПРЕДЕЛЁННЫХ ИНТЕГРАЛОВ В MATHCAD .....</b>	<b>70</b>
10.1. СИМВОЛЬНЫЕ ВЫЧИСЛЕНИЯ. ВЫЧИСЛЕНИЕ ИНТЕГРАЛОВ	71
Индивидуальные задания	71
<b>ЛАБОРАТОРНАЯ РАБОТА №11 СОЗДАНИЕ И ЗАПОЛНЕНИЕ ОДНОТАБЛИЧНОЙ БАЗЫ ДАННЫХ В MS ACCESS. ....</b>	<b>72</b>
11.1. ОСНОВНЫЕ СВЕДЕНИЯ О ТЕХНОЛОГИИ БАЗ ДАННЫХ	72
11.2. МОДЕЛИ ДАННЫХ	73
11.3. ПРИМЕР РАЗРАБОТКИ БАЗЫ ДАННЫХ	75
11.4. СРЕДСТВА И ОБЪЕКТЫ СУБД	77
Выполнение индивидуального задания	78
<b>ЛАБОРАТОРНАЯ РАБОТА №12 ФОРМИРОВАНИЕ ЗАПРОСОВ НА ВЫБОРКУ .....</b>	<b>84</b>
Выполнение индивидуального задания	84
<b>ЛАБОРАТОРНАЯ РАБОТА №13 СОЗДАНИЕ ОТЧЕТА С ГРУППИРОВКОЙ ДАННЫХ ПО ДОЛЖНОСТЯМ.....</b>	<b>90</b>
Выполнение индивидуального задания	90
<b>ЛАБОРАТОРНАЯ РАБОТА №14 ОСНОВЫ РАЗРАБОТКИ И СОЗДАНИЯ WEB-СТРАНИЦ. ОФОРМЛЕНИЕ ТЕКСТА .....</b>	<b>95</b>
14.1. ОСНОВНОЙ ИНСТРУМЕНТАРИЙ	95
14.2. ЧТО ТАКОЕ ТЭГИ?	96
14.3. ОБЯЗАТЕЛЬНЫЕ ТЭГИ	97
14.4. ИЗМЕНЯЕМ ЦВЕТ ТЕКСТА	98
14.5. КАК ИЗМЕНЯТЬ ЦВЕТ ФОНА СТРАНИЦЫ.	99
14.6. ПАРАГРАФЫ И DIV. УЧИМСЯ ВЫРАВНИВАТЬ ТЕКСТ	100
14.7. ЧТО ТАКОЕ ЗАГОЛОВКИ И КАК ЗАДАВАТЬ РАЗМЕР БУКВ	103
14.8. КУРСИВ, ЖИРНЫЙ ТЕКСТ, ПОДЧЕРКНУТЫЙ И ДРУГИЕ	105

14.9. СТАНДАРТНЫЕ ШРИФТЫ	107
<b>ЛАБОРАТОРНАЯ РАБОТА №15 ОСНОВЫ РАЗРАБОТКИ И СОЗДАНИЯ WEB-СТРАНИЦ. РАБОТА С ИЗОБРАЖЕНИЯМИ. ССЫЛКИ.....</b>	<b>108</b>
15.1. ЧТО ТАКОЕ ПУТЬ? КАК ВСТАВЛЯТЬ КАРТИНКИ	108
15.2. ССЫЛКА	112
<b>ЛАБОРАТОРНАЯ РАБОТА №16 ОСНОВЫ РАЗРАБОТКИ И СОЗДАНИЯ WEB-СТРАНИЦ. РАБОТА С ТАБЛИЦАМИ.....</b>	<b>114</b>
16.1. УЧИМСЯ СОЗДАВАТЬ ТАБЛИЦЫ	114
16.2. ТАБЛИЦЫ, ВЕРТИКАЛЬНОЕ ВЫРАВНИВАНИЕ (VALIGN)	116
<b>ПРИЛОЖЕНИЕ 1. МЕТОДЫ ДЛЯ РАБОТЫ СО СТРОКАМИ .....</b>	<b>118</b>
<b>ПРИЛОЖЕНИЕ 2. МЕТОДЫ ДЛЯ РАБОТЫ С МАССИВАМИ .....</b>	<b>120</b>
<b>ЛИТЕРАТУРА.....</b>	<b>121</b>

## ЛАБОРАТОРНАЯ РАБОТА №1 ИЗУЧЕНИЕ СРЕДЫ РАЗРАБОТКИ VISUAL STUDIO

**Цель лабораторной работы:** изучить среду быстрой разработки приложений Visual Studio. Научится размещать и настраивать внешний вид элементов управления на форме.

### 1.1. Интегрированная среда разработчика Visual Studio

**Замечание.** Тем, кто работает дополнительно дома (или для удалённой работы) потребуется установить Visual Studio на свои домашние компьютеры. Как это сделать, как настроить среду для начала работы и как правильно создать проект для выполнения первой работы – вы узнаете, просмотрев видео в курсе Информатика в разделе

#### **Видео. Как установить Visual Studio и начать работать с ней Инструкция для удалённой работы.**

Там же вы увидите незначительные отличия в интерфейсе среды, обусловленные тем, что в тексте практикума иллюстрации соответствуют одной из предыдущих версий Visual Studio.

Рекомендуется вначале ознакомиться с текстом работы в практикуме, затем просмотреть видео, и после начать выполнение работы.

Среда Visual Studio визуально реализуется в виде одного окна с несколькими панелями инструментов. Количество, расположение, размер и вид панелей может меняться программистом или самой средой разработки в зависимости от текущего режима работы среды или пожеланий программиста, что значительно повышает производительность работы.

При запуске Visual Studio появляется начальная страница со списком последних проектов, а также командами *Создать проект* и *Открыть проект*. Нажмите ссылку *Создать проект* или выберите в меню *Файл* команду *Создать проект*, на экране появится диалог для создания нового проекта (рис. 1.1).

Слева в списке шаблонов приведены языки программирования, которые поддерживает данная версия Visual Studio: убедитесь, что там выделен раздел Visual C#. В средней части приведены типы проектов, которые можно создать. В наших лабораторных работах будут использоваться только один тип проектов:

*Приложение Windows Forms* – данный тип проекта позволяет создать полноценное приложение с окнами и элементами управления (кнопками, полями ввода и пр.) Такой вид приложения наиболее привычен большинству пользователей.

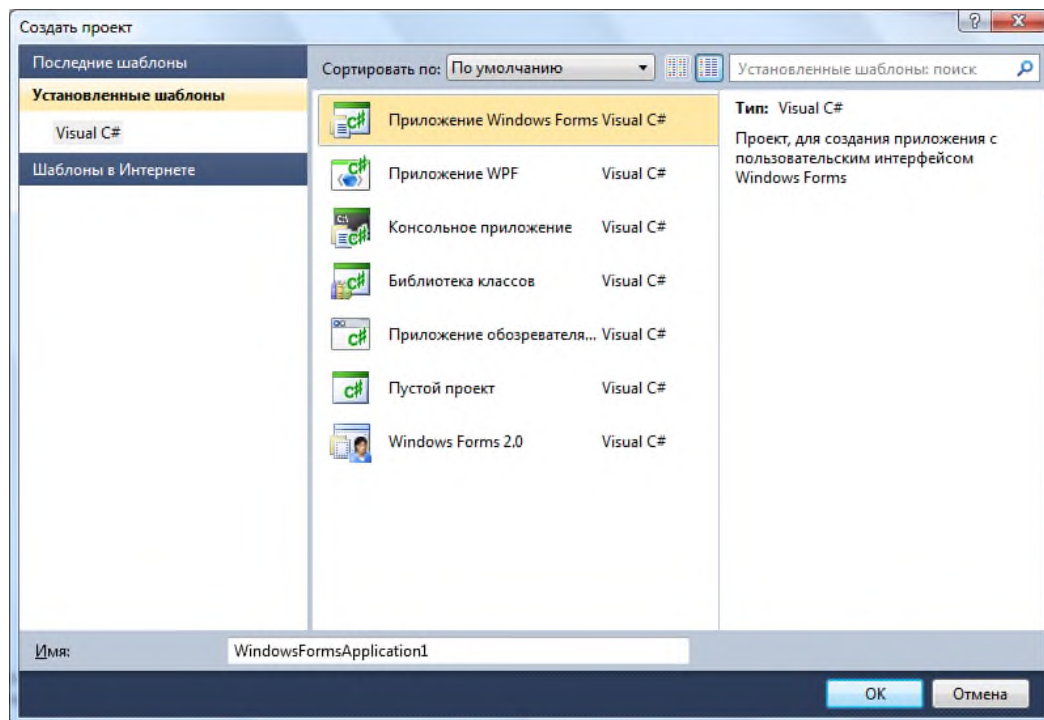


Рис 1.1. Диалог создания нового проекта.

Выберите в списке тип проекта «Приложение Windows Forms», в поле «Имя» внизу окна введите желаемое имя проекта (например, MyFirstApp, а лучше имя по номеру работы LabWork1), справа от поля «Расположение» нажмите кнопку Обзор..., в появившемся диалоге выберите свою личную папку, войдите в неё и нажмите кнопку ОК. Через несколько секунд Visual Studio создаст проект и Вы сможете увидеть на экране картинку, подобную представленной на рис. 1.2.

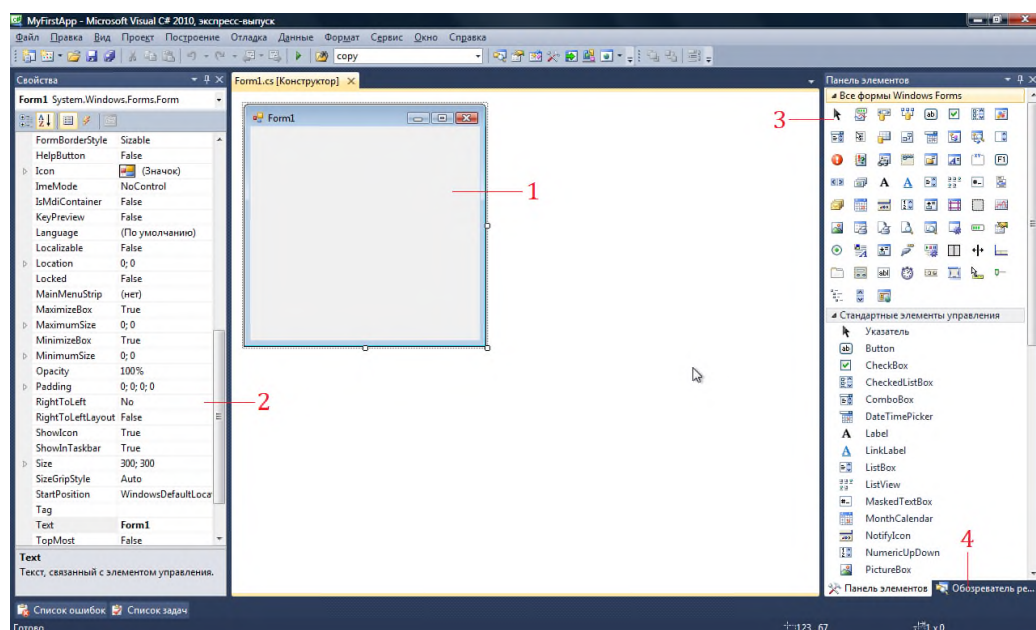





Рис 1.2. Главное окно Visual Studio

В главном окне Visual Studio присутствует несколько основных элементов, которые будут помогать нам в работе. Прежде всего, это **форма** (1) – будущее окно нашего приложения, на котором будут размещаться элементы управления. При выполнении программы помещенные элементы управления будут иметь тот же вид, что и на этапе проектирования.

Второй по важности объект – это **окно свойств** (2), в котором приведены все основные свойства выделенного элемента управления или окна. С помощью кнопки  можно просматривать свойства элемента управления, а кнопка  переключает окно в режим просмотра событий. Чтобы было удобнее искать нужные свойства, можно отсортировать их по алфавиту с помощью кнопки . Если этого окна на экране нет, его можно активировать в меню Вид → Окно свойств (иногда этот пункт вложен в подпункт Другие окна).

Сами элементы управления можно брать на панели элементов (3). Все элементы управления разбиты на логические группы, что облегчает поиск нужных элементов. Если панели нет на экране, её нужно активировать командой Вид → Панель элементов.

Наконец, **обозреватель решений** (4) содержит список всех файлов, входящих в проект, включая добавленные изображения и служебные файлы. Активируется командой Вид → Обзорщик решений.

Указанные панели могут уже присутствовать на экране, но быть скрытыми за другими панелями или свёрнуты к боковой стороне окна. В этом случае достаточно щёлкнуть на соответствующем ярлычке, чтобы вывести панель на передний план.

**Окно текста** программы предназначено для просмотра, написания и редактирования текста программы. Переключаться между формой и текстом программы можно с помощью команд Вид → Код и Вид → Конструктор. При первоначальной загрузке в окне текста программы находится текст, содержащий минимальный набор операторов для нормального функционирования пустой формы в качестве Windows-окна. Если взять нужный элемент с панели элементов и поместить его в окно формы, текст программы в окне кода автоматически дополняется описанием необходимых для его работы библиотек стандартных программ (раздел **using**) и переменных для доступа к элементу управления (в скрытой части класса формы).

Программа на языке C# составляется как описание алгоритмов, которые необходимо выполнить, если возникает определенное событие, связанное с формой (например, щелчок «мыши» на кнопке – событие **Click**, загрузка формы – событие **Load**). Для каждого обрабатываемого в форме события, с помощью окна свойств, в тексте программы организуется метод (подпрограмма, «обработчик событий»), в котором программист записывает на языке C# требуемый алгоритм.

## 1.2. Настройка формы

Настройка формы начинается с настройки размера формы. С помощью мыши, «захватывая» одну из кромок формы или выделенную строку заголовка отрегулируйте нужные размеры формы.

Для настройки будущего окна приложения задаются свойства формы. Для задания любых свойств формы и элементов управления на форме используется **окно свойств**.

Новая форма имеет одинаковые значения свойств Name (имя) и Text (заголовок) – Form1.

Для изменения заголовка щелкните кнопкой мыши на форме, в окне свойств найдите и щелкните мышкой на строчке с названием Text. В выделенном окне наберите текст, включающий номер работы, вашу фамилию и номер группы, например «*Работа1. Иванов, 5A92*». (В дальнейшем рекомендуется в каждой работе использовать именно такой заголовок окна.)


**Важно.** В отличие от свойства Text, свойство Name никогда не изменяйте. Исправить это потом будет сложно.

Для задания цвета окна используйте свойство BackColor.

## 1.3. Размещение элементов управления на форме

Для размещения различных элементов управления на форме используется панель элементов. Панель элементов содержит элементы управления, сгруппированные по типу. Каждую группу элементов управления можно свернуть, если она в настоящий момент не нужна. Для выполнения лабораторных работ потребуются элементы управления из группы *Стандартные элементы управления*.

Щёлкните на элементе управления, который хотите добавить, а затем щёлкните в нужном месте формы – элемент появится на форме.

Элемент можно перемещать по форме, схватившись за него левой кнопкой мышки (иногда это можно сделать лишь за появляющийся при нажатии на элемент квадрат со стрелками  ). Если элемент управления позволяет изменять размеры, то на соответствующих его сторонах появятся белые кружки, ухватившись за которые и можно изменить размер. После размещения элемента управления на форме, его можно выделить щелчком мыши и при этом получить доступ к его свойствам в окне свойств.



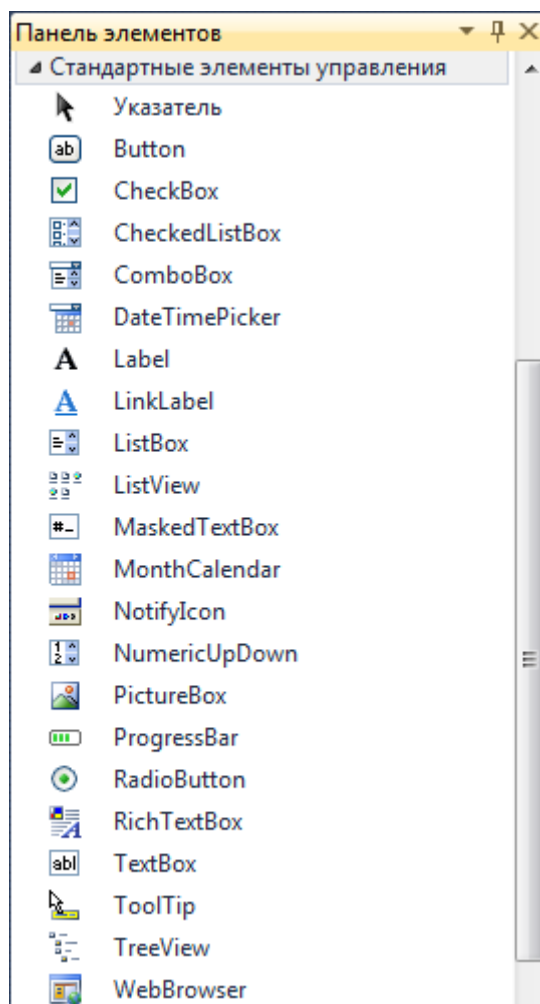


Рис.1.3. Панель элементов

#### 1.4. Размещение строки ввода

Если необходимо ввести из формы в программу или вывести на форму информацию, которая вмещается в одну строку, используют окно однострочного редактора текста, представляемого элементом управления `TextBox`.

В данной программе с помощью однострочного редактора будет вводиться имя пользователя.

Выберите на панели элементов пиктограмму с названием `TextBox`, щелкните мышью в том месте формы, где вы хотите ее поставить. Захватив его мышкой, отрегулируйте размеры элемента управления и его положение. Обратите внимание на то, что теперь в тексте программы можно использовать переменную `textBox1`, которая соответствует добавленному элементу управления. В этой переменной в свойстве `Text` будет содержаться строка символов (тип `string`) и отображаться в соответствующем окне `TextBox`.

С помощью окна свойств установите шрифт и размер символов, отражаемых в строке `TextBox` (свойство `Font`).

## 1.5. Размещение надписей

На форме могут размещаться пояснительные надписи. Для нанесения таких надписей на форму используется элемент управления `Label`. Выберите на панели элементов пиктограмму с названием `Label`, щелкните на ней мышью. После этого в нужном месте формы щелкните мышью, появится надпись `label1`. Щелкнув на ней мышью, отрегулируйте её размер и измените её свойство `Text` в окне свойств (напишите там строку *Введите своё имя:*), а также выберите размер символов (свойство `Font`).

Обратите внимание, что в тексте программы теперь можно обращаться к новой переменной `label1` типа `Label`. В ней хранится пояснительная строка, которую можно изменять в процессе работы программы.

## 1.6. Написание программы обработки события

С каждым элементом управления на форме и с самой формой могут происходить события во время работы программы. Например, с кнопкой может произойти событие – нажатие кнопки, а с окном, которое проектируется с помощью формы, может произойти ряд событий: создание окна, изменение размера окна, щелчок мыши на окне и т. п. Эти события могут быть обработаны в программе, если необходимо обеспечить реакцию программы на такие события. Если не обеспечить обработку события, то событие будет происходить, но программа не будет на него реагировать. Для обработки таких событий необходимо создать обработчики события – специальные *методы*. Для создания обработчика события существует два способа.

Первый способ – создать обработчик для *события по умолчанию* (обычно это самое часто используемое событие данного элемента управления). Для этого нужно дважды щёлкнуть на элементе, расположенном на форме, для которого требуется создать обработчик события. Например, для кнопки таким образом создаётся обработчик события нажатия на кнопку `Click`.

**Важное примечание!** Так как начинающий программист не знает, какое событие для выделенного элемента является *событием по умолчанию*, то начинающим **не рекомендуется (!!!)** использовать этот способ для создания обработчика события. Никогда не делайте без необходимости двойной щелчок на элементе формы, иначе для него будет создан пустой обработчик события по умолчанию, который потом удалить будет непросто. Лучше использовать второй способ создания обработчиков событий, о котором сказано дальше в разделе 1.8.

## 1.7. Написание программы обработки события нажатия кнопки

Поместите на форму кнопку, которая описывается элементом

управления **Button**. С помощью окна свойств измените заголовок (**Text**) на слово «Привет» или другое по вашему желанию. Отрегулируйте положение и размер кнопки.

После этого два раза щелкните мышью на кнопке, появится текст программы:


```
private void button1_Click(object sender, EventArgs e)
{

}
```

Это и есть обработчик события нажатия кнопки. Вы можете добавлять свой код между скобками { }. Например, наберите:

```
MessageBox.Show("Привет, " + textBox1.Text + "!");
```

## 1.8. Написание программы обработки события загрузки формы

Второй способ создания обработчика события заключается в выборе соответствующего события для выделенного элемента на форме. При этом используется окно свойств и внутри него окно событий (закладка  ). Рассмотрим этот способ. Выделите форму щелчком по ней, чтобы вокруг неё появилась рамка из точек. В окне событий найдите событие **Load**. Щелкните по данной строчке дважды мышкой. Появится метод:

```
private void Form1_Load(object sender, EventArgs e)
{

}
```

Между скобками { } вставим текст программы:

```
BackColor = Color.AntiqueWhite;
```

Каждый элемент управления имеет свой набор обработчиков событий, однако некоторые из них присущи большинству элементов управления. Наиболее часто применяемые события представлены в таблице:

Событие	Описание события
Activated	Форма получает это событие при активации
Load	Возникает при загрузке формы. В обработчике данного события следует задавать действия,

	которые должны происходить в момент создания формы, например установка начальных значений
KeyPress	Возникает при нажатии кнопки на клавиатуре. Параметр <code>e.KeyChar</code> имеет тип <code>char</code> и содержит код нажатой клавиши (клавиша <i>Enter</i> клавиатуры имеет код <code>#13</code> , клавиша <i>Esc</i> – <code>#27</code> и т. д.). Обычно это событие используется в том случае, когда необходима реакция на нажатие одной из клавиш
KeyDown	Возникает при нажатии клавиши на клавиатуре. Обработчик этого события получает информацию о нажатой клавише и состоянии клавиш <i>Shift</i> , <i>Alt</i> и <i>Ctrl</i> , а также о нажатой кнопке мыши. Информация о клавише передается параметром <code>e.KeyCode</code> , который представляет собой перечисление <code>Keys</code> с кодами всех клавиш, а информацию о клавишах-модификаторах <i>Shift</i> и др. можно узнать из параметра <code>e.Modifiers</code>
KeyUp	Является парным событием для <code>KeyDown</code> и возникает при отпускании ранее нажатой клавиши
Click	Возникает при нажатии кнопки мыши в области элемента управления
DoubleClick	Возникает при двойном нажатии кнопки мыши в области элемента управления

**Важное примечание!** Если какой-то обработчик был добавлен по ошибке или больше не нужен, то для его удаления нельзя просто удалить программный код обработчика! Сначала нужно удалить строку с именем обработчика в окне свойств на закладке ⚡. В противном случае программа может перестать компилироваться и даже отображать форму в конструкторе Visual Studio.

## 1.9. Запуск и работа с программой

Запустить программу можно, выбрав в меню *Отладка* команду *Начать отладку* или нажав кнопку **Пуск** на панели инструментов. При этом происходит трансляция и, если нет ошибок, компоновка программы и создание единого загружаемого файла с расширением `.exe`. На экране появляется активное окно программы.

Если в программе есть ошибки, то в окне *Список ошибок* появятся найденные ошибки, а программа обычно не запускается. Иногда может появиться окно запроса на запуск предыдущей версии программы, в

которой ещё нет последних изменений: чтобы этого не происходило, **нужно отказаться от запуска предыдущей версии.**

Если ошибок нет, программа запустится. Для завершения работы программы и возвращения в режим проектирования формы не забудьте **закрыть окно программы!**

### 1.10. Динамическое изменение свойств

Свойства элементов окна могут быть изменены не только в процессе создания окна программы с помощью окна свойств, но и динамически во время выполнения программы. Например, можно изменить текст надписи или цвет формы, т. е. не в конструкторе при разработке интерфейса окна, а программно в коде программы. Изменение свойств происходит внутри обработчика события (например, обработчика события нажатия на кнопку). Для этого используют оператор присвоения вида:

```
<имя элемента>.<свойство> = <значение>;
```

Например:

```
label1.Text = "Привет";
```

<Имя элемента> определяется на этапе проектирования формы, при размещении элемента управления на форме. Например, при размещении на форме ряда элементов TextBox, эти элементы получают имена textBox1, textBox2, textBox3 и т. д. Список свойств для конкретного элемента можно посмотреть в окне свойств, а также в приложении к данным методическим указаниям.

Если требуется изменить свойства формы, то никакое имя элемента перед точкой вставлять не нужно, как и саму точку. Например, чтобы задать цвет формы, нужно просто написать:

```
BackColor = Color.Green;
```

### 1.11. Выполнение индивидуального задания

Ниже приведено 15 вариантов задач. По указанию преподавателя выберите свое индивидуальное задание. Уточните условие задания, количество, наименование, типы исходных данных. Прочтите в Приложении 1 описание свойств и описание элементов управления Form, Label, TextBox, Button. С помощью окна свойств установите первоначальный цвет формы, шрифт выводимых символов.

**Замечание.** Если выбранный Вами вариант индивидуального задания покажется слишком сложным, можно выбрать любой из первых четырёх вариантов, которые очень просты.

## Индивидуальные задания

1. Разместите на форме четыре кнопки (Button). Сделайте на кнопках следующие надписи: «красный», «зеленый», «синий», «желтый». Создайте четыре обработчика события нажатия на данные кнопки, которые будут менять цвет формы в соответствии с текстом на кнопках.
2. Разместите на форме две кнопки (Button) и одну метку (Label). Сделайте на кнопках следующие надписи: «привет», «до свидания». Создайте обработчики события нажатия на данные кнопки, которые будут менять текст метки на слова, написанные на кнопках. Создайте обработчик события создания формы (Load), который будет устанавливать цвет формы и менять текст метки на строку «Начало работы».
3. Разместите на форме две кнопки (Button) и одну метку (Label). Сделайте на кнопках следующие надписи: «скрыть», «показать». Создайте обработчики события нажатия на данные кнопки, которые будут скрывать или показывать метку. Создайте обработчик события создания формы (Load), который будет устанавливать цвет формы и менять текст метки на строку «Начало работы».
4. Разместите на форме три кнопки (Button) и одно поле ввода (TextBox). Сделайте на кнопках следующие надписи: «скрыть», «показать», «очистить». Создайте обработчики события нажатия на данные кнопки, которые будут скрывать или показывать поле ввода. При нажатии на кнопку «очистить» текст из поля ввода должен быть удален.
5. Разместите на форме две кнопки (Button) и одно поле ввода (TextBox). Сделайте на кнопках следующие надписи: «заполнить», «очистить». Создайте обработчики события нажатия на данные кнопки, которые будут очищать или заполнять поле ввода знаками «\*\*\*\*\*». Создайте обработчик события создания формы (Load), который будет устанавливать цвет формы и менять текст в поле ввода на строку «+++++».
6. Разработайте игру, которая заключается в следующем. На форме размещены пять кнопок (Button). При нажатии на кнопку некоторые кнопки становятся видимыми, а другие – невидимыми. Цель игры – скрыть все кнопки.
7. Разработайте игру, которая заключается в следующем. На форме размещены четыре кнопки (Button) и четыре метки (Label). При нажатии на кнопку часть надписей становится невидимыми, а часть наоборот становятся видимыми. Цель игры – скрыть все надписи.
8. Разместите на форме ряд кнопок (Button). Создайте обработчики события нажатия на данные кнопки, которые будут делать неактивными текущую кнопку. Создайте обработчик события изменения размера формы (Resize), который будет устанавливать все кнопки в активный режим.
9. Разместите на форме ряд кнопок (Button). Создайте обработчики события нажатия на данные кнопки, которые будут делать неактивными следующую кнопку. Создайте обработчик события нажатия кнопки мыши

на форме (Click), который будет устанавливать все кнопки в активный режим.

10. Разместите на форме три кнопки (Button) и одно поле ввода (TextBox). Сделайте на кнопках следующие надписи: «\*\*\*\*\*», «+++++», «00000». Создайте обработчики события нажатия на данные кнопки, которые будут выводить текст, написанный на кнопках, в поле ввода. Создайте обработчик события создания формы (Load), который будет устанавливать цвет формы и менять текст в поле ввода на строку «Готов к работе».

11. Разместите на форме ряд полей ввода (TextBox). Создайте обработчики события нажатия кнопкой мыши на данные поля ввода, которые будут выводить в текущее поле ввода его номер. Создайте обработчик события изменение размера формы (Resize), который будет очищать все поля ввода.

12. Разместите на форме поле ввода (TextBox), метку (Label) и кнопку (Button). Создайте обработчик события нажатия на кнопку, который будет копировать текст из поля ввода в метку. Создайте обработчик события нажатия кнопки мышки на форме (Click), который будет устанавливать цвет формы и менять текст метки на строку «Начало работы» и очищать поле ввода.

13. Разместите на форме поле ввода (TextBox), и две кнопки (Button) с надписями: «блокировать», «разблокировать». Создайте обработчики события нажатия на кнопки, которые будут делать активным или неактивным поле ввода. Создайте обработчик события нажатия кнопки мышки на форме (Click), который будет устанавливать цвет формы и делать невидимыми все элементы.

14. Реализуйте игру минер на поле 3x3 из кнопок (Button). Первоначально все кнопки не содержат надписей. При попытке нажатия на кнопку на ней либо показывается количество мин, либо надпись «мина!» и меняется цвет окна.

15. Разместите на форме четыре кнопки (Button). Напишите для каждой обработчик события, который будет менять размеры и местоположение на окне других кнопок.

## **ЛАБОРАТОРНАЯ РАБОТА №2**

### **ЛИНЕЙНЫЕ АЛГОРИТМЫ**

**Цель лабораторной работы:** научиться составлять каркас простейшей программы в среде Visual Studio. Написать и отладить программу линейного алгоритма.

#### **2.1. Структура приложения**

Перед началом программирования необходимо создать проект. Проект содержит все исходные материалы для приложения, такие как

файлы исходного кода, ресурсов, значки, ссылки на внешние файлы, на которые опирается программа, и данные конфигурации, такие как параметры компилятора.

Кроме понятия проект часто используется более глобальное понятие – *решение (solution)*. Решение содержит один или несколько проектов, один из которых может быть указан как стартовый проект. Выполнение решения начинается со стартового проекта.

Таким образом, при создании простейшей C# программы в Visual Studio создается папка решения, в которой для каждого проекта создается подпапка проекта, а уже в ней будут создаваться другие подпапки с результатами компиляции приложения.

Проект – это основная единица, с которой работает программист. При создании проекта можно выбрать его тип, а Visual Studio создаст каркас проекта в соответствии с выбранным типом.

В предыдущей лабораторной работе мы попробовали создавать оконные приложения, или иначе *Приложения Windows Forms*. Примером другого типа проекта является проект *консольного* приложения.

Проект в Visual Studio состоит из файла проекта (файл с расширением *.csproj*), одного или нескольких файлов исходного текста (с расширением *.cs*), файлов с описанием окон формы (с расширением *.designer.cs*), файлов ресурсов (с расширением *.resx*), а также ряда служебных файлов.

В *файле проекта* находится информация о модулях, составляющих данный проект, входящих в него ресурсах, а также параметрах построения программы. Файл проекта автоматически создается и изменяется средой Visual Studio и не предназначен для ручного редактирования.

*Файл исходного текста* – программный модуль предназначен для размещения текстов программ. В этом файле программист размещает текст программы, написанный на языке C#. Модуль имеет следующую структуру:

```
// Раздел подключенных пространств имен
using System;

// Пространство имен нашего проекта
namespace MyFirstApp
{
    // Класс окна
    public partial class Form1 : Form
    {
        // Методы окна
        public Form1()
        {
```



```

        InitializeComponent();
    }
}

```

В разделе подключения пространств имен (каждая строка которого располагается в начале файла и начинается ключевым словом `using`) описываются используемые пространства имён. Каждое пространство имён включает в себя классы, выполняющие определённую работу, например классы для работы с сетью располагаются в пространстве `System.Net`, а для работы с файлами – в `System.IO`. Большая часть пространств, которые используются в обычных проектах, уже подключена при создании нового проекта, но при необходимости можно дописать дополнительные пространства имён.

Пространство имён – это своеобразный контейнер, в котором имена программных объектов (классов и переменных) имеют значения, определённые в тех частях программы (модулях), которые вложены в этот контейнер. Это даёт возможность использовать такие же имена в других частях программы, помещая эти части в отдельные пространства имён, без конфликта между одинаковыми именами в разных частях программы.

Для того, чтобы не происходило конфликтов имён классов и переменных, классы нашего проекта также помещаются в отдельное пространство имен. Определяется оно ключевым словом `namespace`, после которого следует имя пространства (**обычно оно совпадает с именем проекта**).

Внутри пространства имён помещаются наши классы – в новом проекте это класс окна, который содержит все методы для управления поведением окна. Обратите внимание, что в определении класса присутствует ключевое слово `partial`, это говорит о том, что в исходном тексте представлена только часть класса, с которой мы работаем непосредственно, а служебные методы для обслуживания окна скрыты в другом модуле (при желании их тоже можно посмотреть, но редактировать вручную не рекомендуется).

Наконец, внутри класса располагаются переменные, методы и другие элементы программы. Фактически, основная часть программы размещается внутри класса при создании обработчиков событий.

При компиляции программы Visual Studio создает исполняемые `.exe`-файлы в каталоге `bin`.

## 2.2. Работа с проектом

Проект в Visual Studio состоит из многих файлов, и создание сложной программы требует хранения каждого проекта в отдельной папке. При создании нового проекта Visual Studio по умолчанию сохраняет его в

отдельной папке. Рекомендуется создать для себя свою папку со своей фамилией внутри папки своей группы, чтобы все проекты хранились в одном месте. После этого можно запускать Visual Studio и создавать новый проект (как это сделать показано в предыдущей лабораторной работе).

Сразу после создания проекта рекомендуется сохранить его в подготовленной папке: *Файл* → *Сохранить всё*. При внесении значительных изменений в проект следует еще раз сохранить проект той же командой, а перед запуском программы на выполнение среда обычно сама сохраняет проект на случай какого-либо сбоя. Для открытия существующего проекта используется команда *Файл* → *Открыть проект*, либо можно найти в папке файл проекта с расширением *.sln* и сделать на нём двойной щелчок.

### **2.3. Описание данных**

Типы данных имеют особенное значение в C#, поскольку это строго типизированный язык. Это означает, что все операции подвергаются строгому контролю со стороны компилятора на соответствие типов, причем недопустимые операции не компилируются. Такая строгая проверка типов позволяет предотвратить ошибки и повысить надежность программ. Для обеспечения контроля типов все переменные, выражения и значения должны принадлежать к определенному типу. Такого понятия, как бестиповая переменная, допустимая в ряде скриптовых языков, в C# вообще не существует. Более того, тип значения определяет те операции, которые разрешается выполнять над ним. Операция, разрешенная для одного типа данных, может оказаться недопустимой для другого.

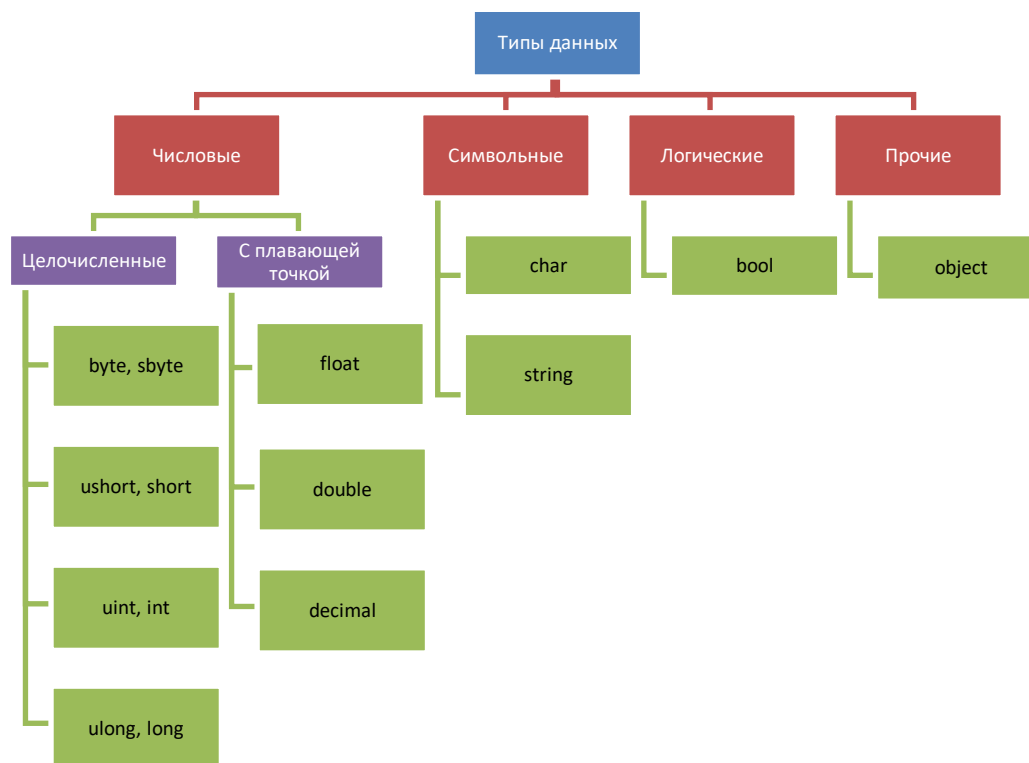


Рис.2.1 Структура типов данных

### ***Целочисленные типы***

В C# определены целый тип другие целочисленные типы: `char`, `byte`, `sbyte`, `short`, `ushort`, `uint`, `long` и `ulong`. Тип `char` может хранить числа, но чаще используется для представления символов. Остальные восемь целочисленных типов предназначены для числовых расчетов. Для начинающих изучать программирование достаточно использовать только тип `int`.

Некоторые целочисленные типы могут хранить как положительные, так и отрицательные значения (`sbyte`, `short`, `int` и `long`), другие же – только положительные (`char`, `byte`, `ushort`, `uint` и `ulong`).

### ***Типы с плавающей точкой***

Такие типы позволяют представлять вещественные числа с дробной частью. В C# имеются две разновидности типов данных с плавающей точкой: `float` и `double`. Они представляют числовые значения с одинарной и двойной точностью, вычисления над ними выполняются аппаратно и поэтому быстро.

Целые числа, входящие в выражения, C# по умолчанию считает целочисленными. Для целого типа определена операция целочисленного деления, обозначаемая символом `/`, таким же, как и для вещественного деления переменных с плавающей точкой. Но выполняются они по-разному. При целочисленном делении от результата деления

отбрасывается дробная часть, и остаётся только целая часть. Поэтому следующее выражение будет иметь значение 0, ведь если 1 нацело разделить на 2, то получится как раз 0:

```
double x = 1 / 2;
```

Чтобы этого не происходило, в подобных случаях нужно явно указывать тип чисел с помощью символов-модификаторов: `f` для `float` и `d` для `double`. Приведённый выше пример правильно будет выглядеть так:

```
double x = 1d / 2d;
```

Иногда в программе возникает необходимость записать числа в экспоненциальной форме. Для этого после мантиссы числа записывается символ «e» и сразу после него – порядок. Например, число  $2,5 \cdot 10^{-2}$  будет записано в программе следующим образом:

```
2.5e-2
```

### **Символьные типы**

В C# символы представлены не 8-разрядным кодом, как во многих других языках программирования, а 16-разрядным кодом, который называется юникодом (Unicode). В юникоде набор символов представлен настолько широко, что он охватывает символы практически из всех естественных языков на свете. Тип символа обозначается ключевым словом `char`.

Основным типом при работе со строками является тип `string`, задающий строки переменной длины. Тип `string` представляет последовательность из нуля или более символов в кодировке Юникод. По сути, текст хранится в виде последовательной доступной только для чтения коллекции объектов `char`.

### **Логический тип данных**

Тип `bool` представляет два логических значения: «истина» и «ложь». Эти логические значения обозначаются в C# зарезервированными словами `true` и `false` соответственно. Следовательно, переменная или выражение типа `bool` будет принимать одно из этих логических значений.

Рассмотрим самые популярные данные – *переменные* и *константы*. Переменная – это ячейка памяти, которой присвоено некоторое имя и это имя используется для доступа к данным, расположенным в данной ячейке. Для каждой переменной задаётся *тип данных* – диапазон всех возможных значений для данной переменной. Объявляются переменные

непосредственно в тексте программы. Лучше всего сразу присвоить им начальное значение с помощью знака присвоения «=» (переменная = значение):

```
int a;      // Только объявление
int b = 7;  // Объявление и инициализация
```

Для того чтобы присвоить значение символьной переменной, достаточно заключить это значение (т. е. символ) в одинарные кавычки:

```
char ch;          // Только объявление
char symbol = 'Z'; // Объявление и инициализация
```

Частным случаем переменных являются *константы*. Константы – это переменные, значения которых не меняются в процессе выполнения программы. Константы описываются как обычная переменная, только с ключевым словом `const` впереди:

```
const int c = 5;
```

## 2.4. Ввод/вывод данных в программе

Рассмотрим один из способов ввода данных через элементы, размещенные на форме. Для ввода данных чаще всего используют элемент управления `TextBox`, через обращение к его свойству `Text`. Свойство `Text` хранит в себе строку введенных символов. Поэтому данные можно считать таким образом:

```
private void button1_Click(object sender, EventArgs e)
{
    string s = textBox1.Text;
}
```

Однако со строкой символов нельзя производить арифметические операции, поэтому лучше всего при вводе числовых данных перевести строку в целое или вещественное число. Для этого у типов `int` и `double` существуют методы `Parse` для преобразования строк в числа. С этими числами можно производить различные арифметические действия. Таким образом, предыдущий пример можно переделать следующим образом:

```
private void button1_Click(object sender, EventArgs e)
{
    string s = textBox1.Text;
    int a = int.Parse(s);
}
```

```
int b = a * a;  
}
```

**Важное примечание!** В языках программирования при записи дробных чисел в выражениях чаще всего используется точка, например, «15.7». Но если число вводится с помощью строки символов, то необходимо преобразовать строку, изображающую число, в само число. Однако в C# методы преобразования строк в числа (вроде `double.Parse()` или `Convert.ToFloat()`) учитывают региональные настройки Windows, в которых в качестве десятичной точки используется символ *запятой* (например, «15,7»). Поэтому в полях `TextBox` в формах следует вводить дробные числа с *запятой*, а не с точкой. В противном случае преобразование не выполнится, а программа остановится с ошибкой.

Перед выводом числовые данные следует преобразовать назад в строку. Для этого у каждой переменной существует метод `ToString()`, который возвращает в результате строку с символьным представлением значения. Вывод данных можно осуществлять в элементы `TextBox` или `Label`, используя свойство `Text`.

Например:

```
private void button1_Click(object sender, EventArgs e)  
{  
    string s = textBox1.Text;  
    int a = int.Parse(s);  
    int b = a * a;  
    label1.Text = b.ToString();  
}
```

## 2.5. Арифметические действия и стандартные функции

При вычислении выражения стоящего в правой части оператора присвоения могут использоваться арифметические операции:

- \* – умножение;
- + – сложение;
- – вычитание;
- / – деление;
- % – остаток от деления.

Для задания приоритетов операций могут использоваться круглые скобки ( ). Также могут использоваться стандартные математические функции, представленные методами класса `Math`:

- `Math.Sin(a)` – синус;

- `Math.Sinh(a)` – гиперболический синус;
- `Math.Cos(a)` – косинус (аргумент задается в радианах);
- `Math.Atan(a)` – арктангенс (аргумент задается в радианах);

- `Math.Log(a)` – натуральный логарифм;
- `Math.Exp(a)` – экспонента;
- `Math.Pow(x, y)` – возводит переменную `x` в степень `y`;
- `Math.Sqrt(a)` – квадратный корень;
- `Math.Abs(a)` – модуль числа;
- `Math.Truncate(a)` – целая часть числа;
- `Math.Round(a)` – округление числа;

В тригонометрических функциях все аргументы задаются в радианах.

## 2.6. Пример написания программы

Пусть нужно составить программу вычисления для заданных значений  $x, y, z$  арифметического выражения

$$u = tg^2(x + y) - e^{y-z} \sqrt{\cos x^2 + \sin z^2}.$$

Панель диалога программы организовать в виде, представленном на рис:

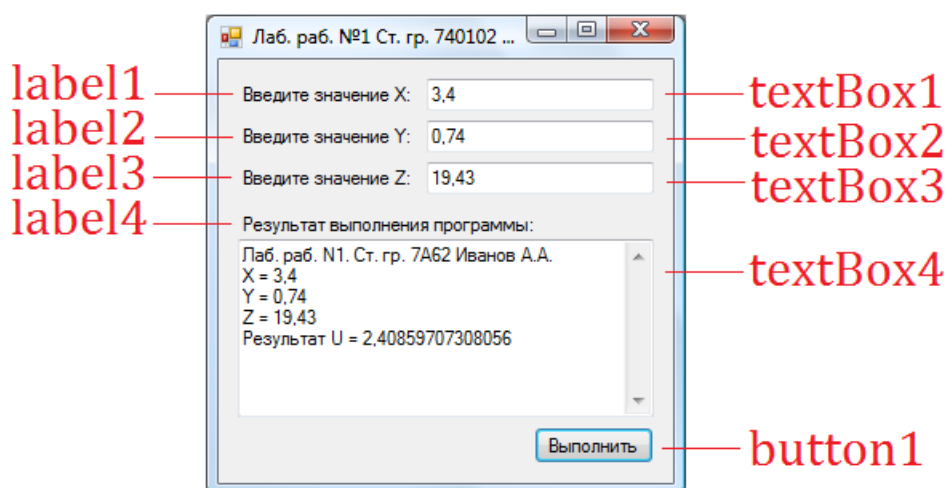


Рис 2.1. Внешний вид программы.

Для вывода результатов работы программы в программе используется текстовое окно, которое представлено обычным элементом управления. После установки свойства `Multiline` в `True` появляется возможность растягивать элемент управления не только по горизонтали, но и по вертикали. А после установки свойства `ScrollBars` в значение `Both` в окне появится вертикальная, а при необходимости и

горизонтальная полосы прокрутки.

Информация, которая отображается построчно в окне, находится в массиве строк `Lines`, каждая строка которого имеет тип `string`. Однако нельзя напрямую обратиться к этому свойству для добавления новых строк, поскольку размер массивов в C# определяется в момент их инициализации. Для добавления нового элемента используется свойство `Text`, к текущему содержимому которого можно добавить новую строку:

```
textBox4.Text = textBox4.Text + Environment.NewLine + "Привет";
```

В этом случае в C# вместо символа присваивания `=` можно использовать сокращенную запись с символом `+=`.

```
textBox4.Text += Environment.NewLine + "Привет";
```

В этом примере к текущему содержимому окна (свойство `Text`) добавляется символ перевода курсора на новую строку (который может отличаться в разных операционных системах, и потому представлен свойством класса `Environment`) и сама новая строка. Если добавляется числовое значение, то его предварительно нужно привести в символьный вид методом `ToString()`.

Работа с программой происходит следующим образом. Нажмите (щелкните мышью) кнопку «*Выполнить*». В окне `textBox4` появляется результат. Измените исходные значения `x`, `y`, `z` в окнах `textBox1`–`textBox3` и снова нажмите кнопку «*Выполнить*» – появятся новые результаты.

Полный текст программы имеет следующий вид:

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

namespace MyFirstApp
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            // Установка начальных значений при загрузке формы
            private void Form1_Load(object sender, EventArgs e)
            {
                // Начальное значение X
                textBox1.Text = "3,4";
                // Начальное значение Y
                textBox2.Text = "0,74";
                // Начальное значение Z
                textBox3.Text = "19,43";
            }
        }
    }
}
```



```

    }

    private void button1_Click(object sender, EventArgs e)
    {
        // Считывание значения X
        double x = double.Parse(textBox1.Text);
        // Вывод значения X в окно
        textBox4.Text += Environment.NewLine + "X = " + x.ToString();
        // Считывание значения Y
        double y = double.Parse(textBox2.Text);
        // Вывод значения Y в окно
        textBox4.Text += Environment.NewLine + "Y = " + y.ToString();
        // Считывание значения Z
        double z = double.Parse(textBox3.Text);
        // Вывод значения Z в окно
        textBox4.Text += Environment.NewLine + "Z = " + z.ToString();
        // Вычисляем арифметическое выражение
        double a = Math.Tan(x + y) * Math.Tan(x + y);
        double b = Math.Exp(y - z);
        double c = Math.Sqrt(Math.Cos(x*x) + Math.Sin(z*z));
        double u = a - b * c;
        // Выводим результат в окно
        textBox4.Text += Environment.NewLine + "Результат U = " + u.ToString();
    }
}

```

Если просто скопировать этот текст и заменить им то, что было в редакторе кода Visual Studio, то программа не заработает. Правильнее будет создать обработчики событий Load у формы и Click у кнопки и уже в них вставить соответствующий код. Это замечание относится и ко всем последующим лабораторным работам.

## 2.7. Выполнение индивидуального задания

Ниже приведено 15 вариантов задач. По указанию преподавателя выберите свое индивидуальное задание. Уточните условие задания, количество, наименование, типы исходных данных. В соответствии с этим установите необходимое количество окон TextBox, тексты заголовков на форме, размеры шрифтов, а также типы переменных и функции преобразования при вводе и выводе результатов. Для проверки правильности программы после задания приведен контрольный пример: тестовые значения переменных, используемых в выражении, и результат, который при этом получается.

### Индивидуальные задания

$$1. \ t = \frac{2 \cos\left(x - \frac{\pi}{6}\right)}{0.5 + \sin^2 y} \left(1 + \frac{z^2}{3 - z^2/5}\right).$$

При  $x=14.26$ ,  $y=-1.22$ ,  $z=3.5 \times 10^{-2}$   $t=0.564846$ .

$$2. u = \frac{\sqrt[3]{8+|x-y|^2+1}}{x^2+y^2+2} - e^{|x-y|} (tg^2 z + 1)^x.$$

При  $x=-4.5$ ,  $y=0.75 \times 10^{-4}$ ,  $z=0.845 \times 10^2$   $u=-55.6848$ .

$$3. v = \frac{1 + \sin^2(x+y)}{\left|x - \frac{2y}{1+x^2y^2}\right|} x^{|y|} + \cos^2\left(\arctg \frac{1}{z}\right).$$

При  $x=3.74 \times 10^{-2}$ ,  $y=-0.825$ ,  $z=0.16 \times 10^2$ ,  $v=1.0553$ .

$$4. w = |\cos x - \cos y|^{(1+2\sin^2 y)} \left(1 + z + \frac{z^2}{2} + \frac{z^3}{3} + \frac{z^4}{4}\right).$$

При  $x=0.4 \times 10^4$ ,  $y=-0.875$ ,  $z=-0.475 \times 10^{-3}$   $w=1.9873$ .

$$5. \alpha = \ln\left(y^{-\sqrt{|x|}}\right) \left(x - \frac{y}{2}\right) + \sin^2 \arctg(z).$$

При  $x=-15.246$ ,  $y=4.642 \times 10^{-2}$ ,  $z=20.001 \times 10^2$   $\alpha=-182.036$ .

$$6. \beta = \sqrt{10(\sqrt[3]{x} + x^{y+2})} (\arcsin^2 z - |x-y|).$$

При  $x=16.55 \times 10^{-3}$ ,  $y=-2.75$ ,  $z=0.15$   $\beta=-40.631$ .

$$7. \gamma = 5 \arctg(x) - \frac{1}{4} \arccos(x) \frac{x+3|x-y|+x^2}{|x-y|z+x^2}.$$

При  $x=0.1722$ ,  $y=6.33$ ,  $z=3.25 \times 10^{-4}$   $\gamma=-205.306$ .

$$8. \phi = \frac{e^{|x-y|} |x-y|^{x+y}}{\arctg(x) + \arctg(z)} + \sqrt[3]{x^6 + \ln^2 y}.$$

При  $x=-2.235 \times 10^{-2}$ ,  $y=2.23$ ,  $z=15.221$   $\phi=39.374$ .

$$9. \psi = \left|x^{\frac{y}{x}} - \sqrt[3]{\frac{y}{x}}\right| + (y-x) \frac{\cos y - \frac{z}{(y-x)}}{1 + (y-x)^2}.$$

При  $x=1.825 \times 10^2$ ,  $y=18.225$ ,  $z=-3.298 \times 10^{-2}$   $\psi=1.2131$ .

$$10. a = 2^{-x} \sqrt{x + \sqrt[4]{|y|}} \sqrt[3]{e^{x-1/\sin z}}.$$

При  $x=3.981 \times 10^{-2}$ ,  $y=-1.625 \times 10^3$ ,  $z=0.512$   $a=1.26185$ .

$$11. b = y^{\sqrt[3]{|x|}} + \cos^3(y) \frac{|x-y| \left(1 + \frac{\sin^2 z}{\sqrt{x+y}}\right)}{e^{|x-y|} + \frac{x}{2}}.$$

При  $x=6.251$ ,  $y=0.827$ ,  $z=25.001$   $b=0.7121$ .

$$12. c = 2^{(y^x)} + (3^x)^y - \frac{y \left(\arctgz - \frac{\pi}{6}\right)}{|x| + \frac{1}{y^2+1}}.$$

При  $x=3.251$ ,  $y=0.325$ ,  $z=0.466 \times 10^{-4}$   $c=4.251$ .

$$13. f = \frac{\sqrt[4]{y + \sqrt[3]{x-1}}}{|x-y|(\sin^2 z + tgz)}.$$

При  $x=17.421$ ,  $y=10.365 \times 10^{-3}$ ,  $z=0.828 \times 10^5$   $f=0.33056$ .

$$14. g = \frac{y^{x+1}}{\sqrt[3]{|y-2|} + 3} + \frac{x + \frac{y}{2}}{2|x+y|} (x+1)^{-1/\sin z}.$$

При  $x=12.3 \times 10^{-1}$ ,  $y=15.4$ ,  $z=0.252 \times 10^3$   $g=82.8257$ .

$$15. h = \frac{x^{y+1} + e^{y-1}}{1+x|y-tgz|} (1+|y-x|) + \frac{|y-x|^2}{2} - \frac{|y-x|^3}{3}.$$

При  $x=2.444$ ,  $y=0.869 \times 10^{-2}$ ,  $z=-0.13 \times 10^3$   $h=-0.49871$ .

### ЛАБОРАТОРНАЯ РАБОТА №3 РАЗВЕТВЛЯЮЩИЕСЯ АЛГОРИТМЫ

**Цель лабораторной работы:** научиться пользоваться элементами управления для организации переключений (RadioButton). Написать и отладить программу разветвляющегося алгоритма.

#### 3.1. Логические переменные и операции над ними

Переменные логического типа описываются посредством служебного слова `bool`. Они могут принимать только два значения – `False` (ложь) и `True` (истина). Результат `False` (ложь) и `True` (истина) возникает при использовании операций сравнения `>` (больше), `<` (меньше), `!=` (не равно), `>=` (больше или равно), `<=` (меньше или равно), `==` (равно). Описываются логические переменные следующим образом:

`bool b;`

В языке C# имеются логические операции, применяемые к переменным логического типа. Это операции *логического отрицания* (`!`), *логическое И* (`&&`) и *логическое ИЛИ* (`||`). Операция логического отрицания является *унарной операцией*. Результат операции `!` есть `False`, если операнд истинен, и `True`, если операнд имеет значение ложь. Так,

`!True → False` (неправда есть ложь)

`!False → True` (не ложь есть правда)

Результат операции *логическое И* (`&&`) есть истина, только если оба ее операнда истинны, и ложь во всех других случаях. Результат операции *логическое ИЛИ* (`||`) есть истина, если какой-либо из ее операндов истинен, и ложь только тогда, когда оба операнда ложны.

### 3.2. Условные операторы

Операторы ветвления позволяют изменить порядок выполнения операторов в программе. К операторам ветвления относятся условный оператор `if` и оператор выбора `switch`.

*Условный оператор `if`* используется для разветвления процесса обработки данных на два направления. Он может иметь одну из форм: сокращенную или полную.

Форма сокращенного оператора `if`:

```
if (В) S;
```

где В – логическое или арифметическое выражение, истинность которого проверяется; S – оператор.

При выполнении сокращенной формы оператора `if` сначала вычисляется выражение В, затем проводится анализ его результата: если В истинно, то выполняется оператор S; если В ложно, то оператор S пропускается. Таким образом, с помощью сокращенной формы оператора `if` можно либо выполнить оператор S, либо пропустить его.

Форма полного оператора `if`:

```
if (В) S1;  
else S2;
```

где В – логическое или арифметическое выражение, истинность которого проверяется; S1, S2 – операторы.

При выполнении полной формы оператора `if` сначала вычисляется выражение В, затем анализируется его результат: если В истинно, то выполняется оператор S1, а оператор S2 пропускается; если В ложно, то выполняется оператор S2, а S1 – пропускается. Таким образом, с помощью полной формы оператора `if` можно выбрать одно из двух альтернативных действий процесса обработки данных.

Для примера вычислим значение функции:

$$y(x) = \begin{cases} \sin(x), & \text{если } x \leq a \\ \cos(x), & \text{если } a < x < b \\ tg(x), & \text{если } x \geq b \end{cases}$$

Указанное выражение может быть запрограммировано в виде

```
if (x <= a)  
    y = Math.Sin(x);  
if ((x > a) && (x < b))
```

```

        y = Math.Cos(x);
    if (x >= b)
        y = Math.Sin(x) / Math.Cos(x);

```

или с помощью оператора else:

```

    if (x <= a)
        y = Math.Sin(x);
    else
        if (x < b)
            y = Math.Cos(x);
        else
            y = Math.Sin(x) / Math.Cos(x);

```

**Важное примечание!** В С-подобных языках программирования, к которым относится и C#, операция сравнения представляется двумя знаками равенства, например:

```
if (a == b)
```

**Замечание.** Условный оператор можно использовать для проверки числа на чётность. Для проверки целого числа на чётность можно использовать проверку остатка от деления числа на 2.

Пример:

```

int i;
if ((i % 2) == 0)
    MessageBox.Show("Число чётное");
else
    MessageBox.Show("Число нечётное");

```

Оператор выбора switch предназначен для разветвления процесса вычислений по нескольким направлениям. Формат оператора:

```

switch (<выражение>)
{
    case <константное_выражение_1>:
        [<оператор 1>];
        <оператор перехода>;
    case <константное_выражение_2>:
        [<оператор 2>];
        <оператор перехода>;
    ...
    case <константное_выражение_n>:
        [<оператор n>];
        <оператор перехода>;
}

```

```

    [default:
        <оператор>;]
}

```

**Замечание.** Выражение, записанное в квадратных скобках, является необязательным элементом в операторе `switch`. Если оно отсутствует, то может отсутствовать и оператор перехода.

Выражение, стоящее за ключевым словом `switch`, должно иметь арифметический, символьный или строковый тип. Все константные выражения должны иметь разные значения, но их тип должен совпадать с типом выражения, стоящим после `switch` или приводиться к нему. Ключевое слово `case` и расположенное после него константное выражение называют также *меткой case*.

Выполнение оператора начинается с вычисления выражения, расположенного за ключевым словом `switch`. Полученный результат сравнивается с меткой `case`. Если результат выражения соответствует метке `case`, то выполняется оператор, стоящий после этой метки, за которым обязательно должен следовать оператор перехода:

`break`, `goto`, `return` и т. д. При использовании оператора `break` происходит выход из `switch` и управление передается оператору, следующему за `switch`. Если же используется оператор `goto`, то управление передается оператору, помеченному меткой, стоящей после `goto`. Наконец, оператор `return` завершает выполнение текущего метода.

Если ни одно выражение `case` не совпадает со значением оператора `switch`, управление передается операторам, следующим за необязательной подписью `default`. Если подписи `default` нет, то управление передается за пределы оператора `switch`.

Пример использования оператора `switch`:

```

int dayOfWeek = 5;
switch (dayOfWeek)
{
    case 1:
        MessageBox.Show("Понедельник");
        break;
    case 2:
        MessageBox.Show("Вторник");
        break;
    case 3:
        MessageBox.Show("Среда");
        break;
    case 4:
        MessageBox.Show("Четверг");
        break;
    case 5:

```

```

        MessageBox.Show("Пятница");
        break;
    case 6:
        MessageBox.Show("Суббота");
        break;
    case 7:
        MessageBox.Show("Воскресенье");
        break;
    default:
        MessageBox.Show("Неверное значение!");
        break;
}

```

Поскольку на момент выполнения оператора `switch` в этом примере переменная `dayOfWeek` равнялась 5, то будут выполнены операторы из блока `case 5`.

В ряде случаев оператор `switch` можно заменить несколькими операторами `if`, однако не всегда такая замена будет легче для чтения. Например, приведённый выше пример можно переписать так:

```

int dayOfWeek = 5;
if (dayOfWeek == 1)
    MessageBox.Show("Понедельник");
else
    if (dayOfWeek == 2)
        MessageBox.Show("Вторник");
    else
        if (dayOfWeek == 3)
            MessageBox.Show("Среда");
        else
            if (dayOfWeek == 4)
                MessageBox.Show("Четверг");
            else
                if (dayOfWeek == 5)
                    MessageBox.Show("Пятница");
                else
                    if (dayOfWeek == 6)
                        MessageBox.Show("Суббота");
                    else
                        if (dayOfWeek == 7)
                            MessageBox.Show("Воскресенье");
                        else
                            MessageBox.Show("Неверное
значение!");

```

### 3.3. Кнопки-переключатели

При создании программ в Visual Studio для организации разветвлений часто используются элементы управления в виде кнопок-переключателей. Состояние такой кнопки (включено – выключено) визуальное отражается на форме. Если пользователь выбирает один из вариантов переключателя в группе, все остальные автоматически отключаются.

Группу составляют все элементы управления `RadioButton` в заданном контейнере, таком как `Form`. Чтобы создать на одной форме несколько групп, поместите каждую группу в собственный контейнер, такой как элемент управления `GroupBox` или `Panel`. На форме (рис. 3.1) представлены кнопки-переключатели `RadioButton` в контейнере `GroupBox`.

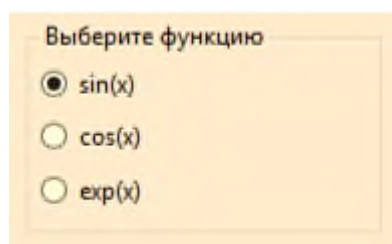


Рис 3.1. Группа радиокнопок

В программу передается номер включенной кнопки (0, 1, 2, ...). Например, чтобы вывести сообщение о том, какая радиокнопка была выбрана, в обработчик события можно вставить такой код:

```
if (radioButton1.Checked)
    MessageBox.Show("Выбрана функция: синус");
else if (radioButton2.Checked)
    MessageBox.Show("Выбрана функция: косинус");
else if (radioButton3.Checked)
    MessageBox.Show("Выбрана функция: экспонента");
```

### 3.4. Пример написания программы

Задание: ввести три числа –  $x$ ,  $y$ ,  $z$ . Вычислить

$$U = \begin{cases} y \cdot f(x) + z, & \text{при } z - x = 0 \\ y \cdot e^{f(x)} - z, & \text{при } z - x < 0 \\ y \cdot \sin(f(x)) + z, & \text{при } z - x > 0 \end{cases},$$

где  $f(x)$  по выбору пользователя может быть  $\sin(x)$ ,  $x^2$  или  $e^x$ .

#### Создание формы

Создайте форму, в соответствии с рис. 3.2.



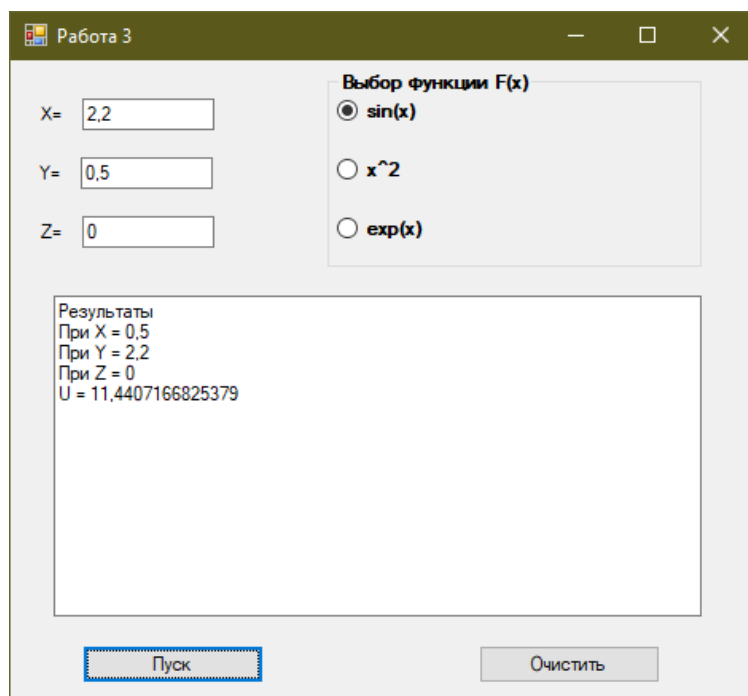


Рис 3.2. Окно лабораторной работы

Выберите в панели элементов из контейнеров `GroupBox` и поместите его в нужное место формы. На форме появится окаймленный линией чистый прямоугольник с заголовком `GroupBox1`. Замените заголовок (`Text`) на Выбор функции  $F(x)$ . Далее, как показано на рисунке, разместите в данном контейнере три радиокнопки (`RadioButton`). Для первой из них установите свойство `Checked` в значение `True`.

Далее разместите на форме элементы `Label`, `TextBox` и `Button`. Поле для вывода результатов также является элементом `TextBox` с установленным в `True` свойством `Multiline` и свойством `ScrollBars` установленным в `Both`.

### Создание обработчиков событий

Обработчики событий создаются аналогично тому, как и в предыдущих лабораторных работах. Текст обработчика события нажатия на кнопку ПУСК приведен ниже. Обратите внимание, что в тексте обработчика описана локальная переменная с именем `fx`, играющая роль функции, которую пользователь программы должен предварительно выбрать с помощью одной из радиокнопок. При описании локальной переменной необходимо присвоить ей начальное значение. Это требование компилятора языка `C#`. Поэтому достаточно присвоить ей начальное значение `0`, которое при выборе функции будет переопределено на значение выбранной функции.

```
private void button1_Click(object sender, EventArgs e)
{
    // Получение исходных данных из TextBox
    double x = Convert.ToDouble(textBox1.Text);
```

```

double y = Convert.ToDouble(textBox2.Text);
double z = Convert.ToDouble(textBox3.Text);
double fx = 0;
// Ввод исходных данных в окно результатов
textBox4.Text = "Результаты " + Environment.NewLine;
textBox4.Text += "При X = " + textBox1.Text + Environment.NewLine;
textBox4.Text += "При Y = " + textBox2.Text + Environment.NewLine;
textBox4.Text += "При Z = " + textBox3.Text + Environment.NewLine;
// Выбор функции
if (radioButton1.Checked)
    fx = Math.Sin(x);
else if (radioButton2.Checked)
    fx = Math.Pow(x, 2);
else if (radioButton3.Checked)
    fx = Math.Exp(x);
// Вычисление выражения
double u;
if ((z - x) == 0)
    u = y * fx + z;
else if ((z - x) < 0)
    u = y * Math.Exp(fx) - z;
else
    u = y * Math.Sin(fx) + z;
// Вывод результата
textBox4.Text += "U = " + u.ToString() + Environment.NewLine;
}

```

Запустите программу и убедитесь в том, что все ветви алгоритма выполняются правильно.

### Индивидуальные задания

По указанию преподавателя выберите индивидуальное задание из нижеприведенного списка. В качестве  $f(x)$  использовать по выбору:  $sh(x)$ ,  $x^2$ ,  $e^x$ . Отредактируйте вид формы и текст программы, в соответствии с полученным заданием.

$$1. \quad a = \begin{cases} (f(x) + y)^2 - \sqrt{f(x)y}, & xy > 0 \\ (f(x) + y)^2 + \sqrt{|f(x)y|}, & xy < 0 \\ (f(x) + y)^2 + 1, & xy = 0. \end{cases}$$

$$2. \quad b = \begin{cases} \ln(f(x)) + (f(x)^2 + y)^3, & x/y > 0 \\ \ln|f(x)/y| + (f(x) + y)^3, & x/y < 0 \\ (f(x)^2 + y)^3, & x = 0 \\ 0, & y = 0. \end{cases}$$

$$3. \quad c = \begin{cases} f(x)^2 + y^2 + \sin(y), & x - y = 0 \\ (f(x) - y)^2 + \cos(y), & x - y > 0 \\ (y - f(x))^2 + \operatorname{tg}(y), & x - y < 0. \end{cases}$$

$$4. \quad d = \begin{cases} (f(x) - y)^3 + \operatorname{arctg}(f(x)), & x > y \\ (y - f(x))^3 + \operatorname{arctg}(f(x)), & y > x \\ (y + f(x))^3 + 0.5, & y = x. \end{cases}$$

$$5. \quad e = \begin{cases} i\sqrt{f(x)}, & i - \text{нечетное}, x > 0 \\ i/2\sqrt{|f(x)|}, & i - \text{четное}, x < 0 \\ \sqrt{|if(x)|}, & \text{иначе.} \end{cases}$$

$$6. \quad g = \begin{cases} e^{f(x)-|b|}, & 0.5 < x < 10 \\ \sqrt{|f(x) + b|}, & 0.1 < x < 0.5 \\ 2f(x)^2, & \text{иначе.} \end{cases}$$

- $$7. \quad s = \begin{cases} e^{f(x)}, & 1 \leq x \leq 10 \\ \sqrt{|f(x) + 4 * b|}, & 12 \leq x \leq 40 \\ bf(x)^2, & \text{иначе.} \end{cases}$$
- $$8. \quad j = \begin{cases} \sin(5f(x) + 3m|f(x)|), & -1 \leq m \leq x \\ \cos(3f(x) + 5m|f(x)|), & x \leq m \\ (f(x) + m)^2, & x = m. \end{cases}$$
- $$9. \quad l = \begin{cases} 2f(x)^3 + 3p^2, & x \leq |p| \\ |f(x) - p|, & 3 \leq x \leq |p| \\ (f(x) - p)^2, & x = |p|. \end{cases}$$
- $$10. \quad k = \begin{cases} \ln(|f(x)| + |q|), & |xq| \leq 10 \\ e^{f(x)+q}, & |xq| \leq 10 \\ f(x) + q, & |xq| = 10 \end{cases}$$
- $$11. \quad m = \frac{\max(f(x), y, z)}{\min(f(x), y)} + 5.$$
- $$12. \quad n = \frac{\min(f(x) + y, y - z)}{\max(f(x), y)}.$$
- $$13. \quad p = \frac{|\min(f(x), y) - \max(y, z)|}{2}.$$
- $$14. \quad q = \frac{\max(f(x) + y + z, xyz)}{\min(f(x) + y + z, xyz)}.$$
- $$15. \quad r = \max(\min(f(x), y), z).$$

## ЛАБОРАТОРНАЯ РАБОТА №4 ЦИКЛИЧЕСКИЕ АЛГОРИТМЫ И ОДНОМЕРНЫЕ МАССИВЫ

**Цель лабораторной работы:** изучить простейшие средства отладки программ в среде Visual Studio. Изучить способы получения случайных чисел. Написать программу для работы с одномерными массивами с использованием циклического алгоритма.

### 4.1. Операторы организации циклов

Под *циклом* понимается многократное выполнение одних и тех же операторов при различных значениях промежуточных данных. Число повторений может быть задано в явной или неявной форме.

К операторам цикла относятся: *цикл с предусловием while*, *цикл с постусловием do while*, *цикл с параметром for* и *цикл перебора foreach*. Рассмотрим некоторые из них.

### 4.2. Цикл с предусловием

Оператор цикла **while** организует выполнение одного оператора неизвестное заранее число раз. Оператор может быть простым и составным. Составной оператор — это любое число любых операторов, заключённых в фигурные скобки. Составной оператор иногда называют блоком. Формат цикла **while**:

**while** (В) S;

где В — выражение, истинность которого проверяется (условие завершения цикла); S — тело цикла — оператор (простой или составной).

Перед каждым выполнением тела цикла анализируется значение выражения В: если оно истинно, то выполняется тело цикла, и управление

передается на повторную проверку условия В; если значение В ложно – цикл завершается и управление передается на оператор, следующий за оператором цикла.

Если результат выражения В окажется ложным при первой проверке, то тело цикла не выполнится ни разу. Отметим, что если условие В во время работы цикла не будет изменяться, то возможна ситуация заикливания, то есть невозможность выхода из цикла. Поэтому внутри тела должны находиться операторы, приводящие к изменению значения выражения В так, чтобы цикл мог корректно завершиться.

В качестве иллюстрации выполнения цикла `while` рассмотрим программу вывода целых чисел от 1 до n внутри обработчика события нажатия кнопки на форме:

```
private void button1_Click(object sender, EventArgs e)
{
    int n = 10;    // Количество повторений цикла
    int i = 1;     // Начальное значение
    while (i <= n) // Пока i меньше или равно n
    {
        MessageBox.Show(i.ToString()); // Показываем i
        i++;                          // Увеличиваем i на 1
    }
}
```

### 4.3. Цикл с постусловием

Оператор цикла `do while` также организует выполнение одного оператора (простого или составного) неизвестное заранее число раз. Однако в отличие от цикла `while` условие завершения цикла проверяется после выполнения тела цикла. Формат цикла `do while`:

```
do S while (В);
```

где В – выражение, истинность которого проверяется (условие завершения цикла); S – тело цикла – оператор (простой или блок).

Сначала выполняется оператор S, а затем анализируется значение выражения В: если оно истинно, то управление передается оператору S, если ложно – цикл завершается, и управление передается на оператор, следующий за условием В. Так как условие В проверяется после выполнения тела цикла, то в любом случае тело цикла выполнится хотя бы один раз.

В операторе `do while`, так же, как и в операторе `while`, возможна ситуация заикливания в случае, если условие В всегда будет оставаться истинным.

## 4.4. Цикл с параметром

Цикл с параметром имеет следующую структуру:

```
for (<инициализация>; <выражение>; <модификация>)  
    <оператор>;
```

*Инициализация* используется для объявления и/или присвоения начальных значений величинам, используемым в цикле в качестве параметров (счетчиков). Областью действия переменных, объявленных в части инициализации цикла, является цикл и вложенные блоки. Инициализация выполняется один раз в начале исполнения цикла.

*Выражение* определяет условие выполнения цикла: если его результат истинен, цикл выполняется. Истинность выражения проверяется перед каждым выполнением тела цикла, таким образом, цикл с параметром реализован как *цикл с предусловием*.

*Модификация* выполняется после каждой итерации цикла и служит обычно для изменения параметров цикла.

*Оператор* (простой или составной) представляет собой тело цикла.

Пример формирования строки, состоящей из чисел от 0 до 9, разделенных пробелами:

```
string s = ""; // Инициализация строки  
for (int i = 0; i <= 9; i++) // Перечисление всех чисел  
    s += i.ToString() + " "; // Добавляем число и пробел  
MessageBox.Show(s.ToString()); // Показываем результат
```

Данный пример работает следующим образом. Сначала вычисляется начальное значение переменной *i*. Затем, пока значение *i* меньше или равно 9, выполняется тело цикла и затем повторно вычисляется значение *i*. Когда значение *i* становится больше 9, условие становится ложным и управление передается за пределы цикла.

## 4.5. Средства отладки программ

Практически в каждой вновь написанной программе после запуска обнаруживаются ошибки.

Ошибки первого уровня (ошибки компиляции) связаны с неправильной записью операторов (орфографические, синтаксические). При обнаружении ошибок компилятор формирует список, который отображается по завершению компиляции (рис. 4.1). При этом возможен только запуск программы, которая была успешно скомпилирована для предыдущей версии программы.

Список ошибок		
❌ Ошибок: 2    ⚠ Предупреждений: 0    ⓘ Сообщений: 0		
	Описание	Файл      Строка
❌ 1	Использование локальной переменной "u", которой не присвоено значение	Form1.cs      45
❌ 2	Аргумент отсутствует	Form1.cs      39

Рис. 4.1. Окно со списком ошибок компиляции.

При выявлении ошибок компиляции в нижней части экрана появляется текстовое окно (рис 4.1), содержащее сведения обо всех ошибках, найденных в проекте. Каждая строка этого окна содержит имя файла, в котором найдена ошибка, номер строки с ошибкой и характер ошибки. Для быстрого перехода к интересующей ошибке необходимо дважды щелкнуть на строке с ее описанием. Следует обратить внимание на то, что одна ошибка может повлечь за собой другие, которые исчезнут при ее исправлении. Поэтому следует исправлять ошибки последовательно, сверху вниз и, после исправления каждой ошибки компилировать программу снова.

Ошибки второго уровня (ошибки выполнения) связаны с ошибками выбранного алгоритма решения или с неправильной программной реализацией алгоритма. Эти ошибки проявляются в том, что результат расчета оказывается неверным, либо происходит переполнение, деление на ноль и др. Поэтому перед использованием отлаженной программы ее надо протестировать, т. е. сделать просчеты при таких комбинациях исходных данных, для которых заранее известен результат. Если тестовые расчеты указывают на ошибку, то для ее поиска следует использовать встроенные средства отладки среды программирования.

В простейшем случае для локализации места ошибки рекомендуется поступать следующим образом. В окне редактирования текста установить точку останова перед подозрительным участком, которая позволит остановить выполнение программы и далее более детально следить за ходом работы операторов и изменением значений переменных. Для этого достаточно в окне редактирования кода щелкнуть слева от нужной строки. В результате чего данная строка будет выделена красным (рис. 4.2).



```
double y = Convert.ToDouble(textBox2.Text);
double z = Convert.ToDouble(textBox3.Text);
// Ввод исходных данных и их вывод в окно результатов
textBox4.Text = "Результаты работы программы ст. Петрова И.И. " + Environment.NewLine;
textBox4.Text += "При X = " + textBox1.Text + Environment.NewLine;
textBox4.Text += "При Y = " + textBox2.Text + Environment.NewLine;
textBox4.Text += "При Z = " + textBox3.Text + Environment.NewLine;
int n = 0;
if (radioButton2.Checked) n = 1; else n = 2;
double u;
switch (n)
{
    case 0:
        if ((z - x) == 0) u = y * Math.Sin(x) * Math.Sin(x) + z;
        else if ((z - x) < 0) u = y * Math.Exp(Math.Sin(x)) - z;
```

Рис. 4.2. Фрагмент кода с точкой останова

При выполнении программы и достижения установленной точки, программа будет остановлена, и далее можно будет выполнять код по шагам с помощью команд *Отладка* → *Шаг с обходом* (без захода в методы) или *Отладка* → *Шаг с заходом* (с заходом в методы) (рис 4.3).



```
textBox4.Text += "При Z = " + textBox3.Text + Environment.NewLine;
int n = 0;
if (radioButton2.Checked) n = 1;
else if (radioButton3.Checked) n = 2;
double u;
switch (n)
{
    case 0:
        if ((z - x) == 0) u = y * Math.Sin(x) * Math.Sin(x) + z;
        else if ((z - x) < 0) u = y * Math.Exp(Math.Sin(x)) - z;
        else u = y * Math.Sin(Math.Sin(x)) + z;
        textBox4.Text += "U = " + Convert.ToString(u) + Environment.NewLine;
```

Рис. 4.3. Отладка программы

Желтым цветом выделяется оператор, который будет выполнен. Значение переменных во время выполнения можно увидеть, наведя на них курсор. Для прекращения отладки и остановки программы нужно выполнить команду меню *Отладка* → *Остановить отладку*.

Для поиска алгоритмических ошибок можно контролировать значения промежуточных переменных на каждом шаге выполнения подозрительного кода и сравнивать их с результатами, полученными вручную.

## 4.6. Работа с массивами

*Массив* – набор элементов одного и того же типа, объединенных общим именем. Массивы в С# можно использовать по аналогии с тем, как они используются в других языках программирования. Однако С#-массивы имеют существенные отличия: они относятся к ссылочным типам данных, более того – реализованы как объекты. Фактически имя массива является ссылкой на область кучи (динамической памяти), в которой последовательно размещается набор элементов определенного типа. Выделение памяти под элементы происходит на этапе инициализации



массива. А за освобождением памяти следит система сборки мусора – неиспользуемые массивы автоматически утилизируются данной системой.

Рассмотрим в данной лабораторной работе одномерные массивы. *Одномерный массив* – это фиксированное количество элементов одного и того же типа, объединенных общим именем, где каждый элемент имеет свой номер. Нумерация элементов массива в С# начинается с нуля, то есть, если массив состоит из 10 элементов, то его элементы будут иметь следующие номера: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Одномерный массив в С# реализуется как объект, поэтому его создание представляет собой двухступенчатый процесс. Сначала объявляется ссылочная переменная на массив, затем выделяется память под требуемое количество элементов базового типа, и ссылочной переменной присваивается адрес нулевого элемента в массиве. Базовый тип определяет тип данных каждого элемента массива. Количество элементов, которые будут храниться в массиве, определяет размер массива.

В общем случае процесс объявления переменной типа массив, и выделение необходимого объема памяти может быть разделено. Кроме того, на этапе объявления массива можно произвести его инициализацию. Поэтому для объявления одномерного массива может использоваться одна из следующих форм записи:

```
тип[] имя_массива;
```

В этом случае описывается ссылка на одномерный массив, которая в дальнейшем может быть использована для адресации на существующий массив. Размер массива при таком объявлении не задаётся.

Пример, в котором объявляется массив целых чисел с именем *a*:

```
int[] a;
```

Другая форма объявления массива включает и его инициализацию указанным количеством элементов:

```
тип[] имя_массива = new тип[размер];
```

В этом случае объявляется одномерный массив указанного типа и выделяется память под указанное количество элементов. Адрес данной области памяти записывается в ссылочную переменную. Элементы массива инициализируются значениями, которые по умолчанию приняты для данного типа: массивы числовых типов инициализируются нулями, строковые переменные – пустыми строками, символы – пробелами, объекты ссылочных типов – значением *null*. Пример такого объявления:



```
int[] a = new int[10];
```

Здесь выделяется память под 10 элементов типа `int`.

Наконец, третья форма записи даёт возможность сразу инициализировать массив конкретными значениями:

```
тип[] имя_массива = {список инициализации};
```

При такой записи выделяется память под одномерный массив, размерность которого соответствует количеству элементов в списке инициализации. Адрес этой области памяти записан в ссылочную переменную. Значение элементов массива соответствует списку инициализации. Пример:

```
int[] a = { 0, 1, 2, 3 };
```

В данном случае будет создан массив `a`, состоящий из четырёх элементов, и каждый элемент будет инициализирован очередным значением из списка.

Обращение к элементам массива происходит с помощью индекса: для этого нужно указать имя массива и в квадратных скобках его номер. Например, `a[0]`, `b[10]`, `c[i]`. **Следует помнить, что нумерация элементов начинается с нуля!**

Так как массив представляет собой набор элементов, объединенных общим именем, то обработка массива обычно производится в цикле. Например:

```
int[] myArray = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };  
for (int i = 0; i < 10; i++)  
    MessageBox.Show(myArray[i]);
```

#### 4.7. Случайные числа

Одним из способов инициализации массива является определение элементов через случайные числа. Для работы со случайными числами используются класс `Random`. Метод `Random.Next` создает случайное число в диапазоне значений от нуля до максимального значения типа `int` (его можно узнать с помощью свойства `Int32.MaxValue`). Для создания случайного числа в диапазоне от нуля до какого-либо другого положительного числа используется перегрузка метода `Random.Next(Int32)` – единственный параметр метода указывает верхнюю границу диапазона, сама граница в диапазон не включается. Для

создания случайного числа в другом диапазоне используется перегрузка метода `Random.Next(Int32, Int32)` – первый аргумент задаёт нижнюю границу диапазона, а второй – верхнюю.

#### 4.8. Порядок выполнения индивидуального задания

Создайте форму с элементами управления как приведено на рис. 7.1. Опишите одномерный массив. Создайте обработчики события для кнопок (код приведен ниже). Данная программа заменяет все отрицательные числа нулями. Протестируйте правильность выполнения программы. Модифицируйте программу в соответствии с индивидуальным заданием.

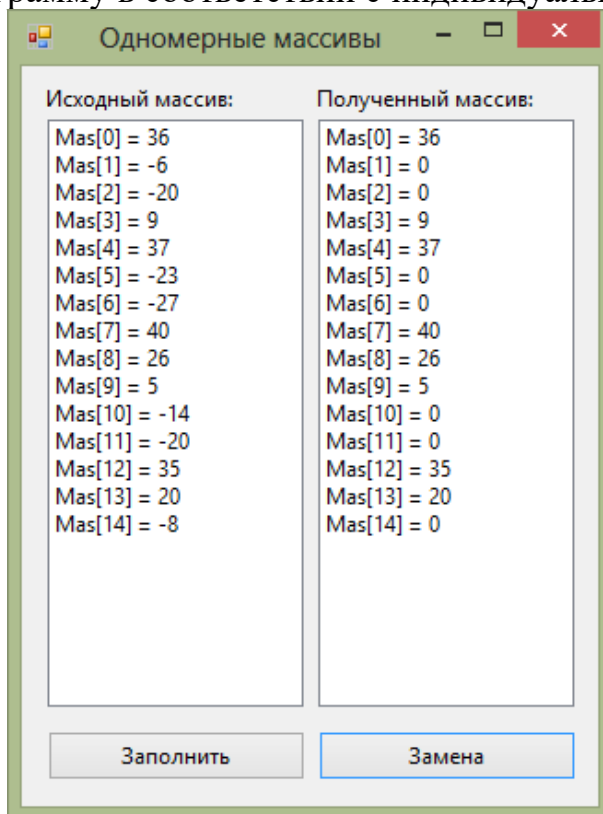


Рис. 7.1. Окно программы для работы с одномерными массивами

```
// Глобальная переменная видна всем методам
int[] Mas = new int[15];

// Заполнение исходного массива
private void button1_Click(object sender, EventArgs e)
{
    // Очищаем элемент управления
    listBox1.Items.Clear();
    // Инициализируем класс случайных чисел
    Random rand = new Random();
    // Генерируем и выводим 15 элементов
    for (int i = 0; i < 15; i++)
    {
        Mas[i] = rand.Next(-50, 50);
    }
}
```

```

        listBox1.Items.Add("Mas[" + i.ToString() +
            "]" = " + Mas[i].ToString());
    }
}

// Замена отрицательных элементов нулями
private void button2_Click(object sender, EventArgs e)
{
    // Очищаем элемент управления
    listBox2.Items.Clear();
    // Обрабатываем все элементы
    for (int i = 0; i < 15; i++)
    {
        if (Mas[i] < 0)
            Mas[i] = 0;
        listBox2.Items.Add("Mas[" + Convert.ToString(i)
            + "]" = " + Mas[i].ToString());
    }
}

```

### Индивидуальные задания

**Замечание.** Часть вариантов заданий требует знания простейших алгоритмов обработки массивов (таких как накопление суммы или произведения элементов массивов, поиска максимального или минимального элемента в массиве и алгоритма обмена значений двух элементов через третий элемент такого же типа (метод трёх стаканов). Простейшие алгоритмы обработки массивов данных приведены в конце этого раздела после списка индивидуальных заданий. Материал, связанный с алгоритмами обработки массивов данных, является обязательным для студентов, занимающихся изучением углубленной информатики.

Если выбранный Вами вариант индивидуального задания покажется слишком сложным, можно выбрать любой из вариантов 11, 23, 24, 25, которые очень просты.

- 1) В массиве из 20 целых чисел найти наибольший элемент и поменять его местами с первым элементом.
- 2) В массиве из 10 целых чисел найти наименьший элемент и поменять его местами с последним элементом.
- 3) В массиве из 15 вещественных чисел найти наибольший элемент и поменять его местами с последним элементом.
- 4) В массиве из 25 вещественных чисел найти наименьший элемент и поменять его местами с первым элементом.

5) Дан массив X, содержащий 27 элементов. Вычислить и вывести элементы нового массива Y, где  $y_i = 6.85x_i^2 - 1.52$ . Если  $y_i < 0$ , то вычислить и вывести  $a = x_i^3 - 0.62$  и продолжить вычисления; если  $y_i \geq 0$ , то вычислить и вывести  $b = 1/x_i^2$  и продолжить вычисления.

6) Дан массив X, содержащий 16 элементов. Вычислить и вывести значения  $d_i$ , где

$$d_i = \frac{e^{x_i} + 2e^{-x_i}}{\sqrt{5 + \sin x_i}},$$

и значения  $d_i > 0.1$ .

7) Дан массив Y, содержащий 25 элементов. Записать в массив R и вывести значения элементов, вычисляемые по формуле

$$r_i = \frac{5y_i + \cos^2 y_i}{2.35},$$

$i = 1, 2, \dots, 25$ .

8) Дан массив F, содержащий 18 элементов. Вычислить и вывести элементы нового массива  $p_i = 0.13f_i^3 - 2.5f_i + 8$ . Вывести отрицательные элементы массива P.

9) Вычислить и вывести элементы массива Z, где  $z_i = i^2 + 1$ , если  $i$  – нечетное, и  $z_i = 2i - 1$ , если  $i$  – четное. Сформировать и вывести массив D:  $d_i = 2.5z_i$ , если  $z_i < 2.5$  и  $d_i = z_i / 2.5$ , если  $z_i \geq 2.5$ .

10) Заданы массивы D и E. Вычислить и вывести значения  $f_i = (2d_i + \sin e_i) / d_i$ , где  $i = 1, 2, \dots, 16$ ; вывести  $1 < f_i < 3$ .

11) В массиве R, содержащем 25 элементов, заменить значения отрицательных элементов квадратами значений, значения положительных увеличить на 7, а нулевые значения оставить без изменения. Вывести массив R.

12) Дан массив A целых чисел, содержащий 30 элементов. Вычислить и вывести сумму тех элементов, которые кратны 5.

13) Дан массив A целых чисел, содержащий 30 элементов. Вычислить и вывести сумму тех элементов, которые нечетны и отрицательны.

14) Дан массив A целых чисел, содержащий 30 элементов. Вычислить и вывести сумму тех элементов, которые удовлетворяют условию  $|a_i| < i^2$ .

15) Дан массив A целых чисел, содержащий 30 элементов. Вычислить и вывести количество и сумму тех элементов, которые делятся на 5 и не делятся на 7.

16) Дан массив A вещественных чисел, содержащий 25 элементов. Вычислить и вывести число отрицательных элементов и число членов, принадлежащих отрезку  $[1, 2]$ .

17) Дан массив C, содержащий 23 элемента. Вычислить и вывести среднее арифметическое всех значений  $c_i > 3.5$ .

18) Дан массив  $Z$  целых чисел, содержащий 35 элементов. Вычислить и вывести  $R = S + P$ , где  $S$  – сумма четных элементов, меньших 3,  $P$  – произведение нечетных элементов, больших 1.

19) Дан массив  $Q$  натуральных чисел, содержащий 20 элементов. Найти и вывести те элементы, которые при делении на 7 дают остаток 1, 2 или 5.

20) Дан массив  $Q$  натуральных чисел, содержащий 20 элементов. Найти и вывести те элементы, которые обладают тем свойством, что корни уравнения  $q_i^2 + 3q_i - 5 = 0$  действительны и положительны.

21) Дан массив, содержащий 10 элементов. Вычислить произведение элементов, стоящих после первого отрицательного элемента. Вывести исходный массив и результат вычислений.

22) Дан массив, содержащий 14 элементов. Вычислить сумму элементов, стоящих до первого отрицательного элемента. Вывести исходный массив и результат вычислений.

23) Дан массив, содержащий 12 элементов. Все четные элементы сложить, вывести массив и результат.

24) Дан массив, содержащий 15 элементов. Все положительные элементы возвести в квадрат, а отрицательные умножить на 2. Вывести исходный и полученный массив.

25) Дан массив, содержащий 14 элементов. Все отрицательные элементы заменить на 3. Вывести исходный и полученный массив.

### **Основные алгоритмы обработки массивов данных**

Наиболее часто в информационных и вычислительных задачах используются алгоритмы, связанные с обработкой массивов различных типов данных. К ним относятся:

- алгоритмы накопления (счетчик, вычисление суммы и произведения элементов массива);
- поиск экстремума (поиск минимального или максимального элементов массива);
- алгоритм обмена значениями;
- сортировка.

Далее эти алгоритмы приводятся в виде фрагментов кода, которые можно использовать в обработчиках событий.

#### **Накопление**

Алгоритм накопления вычисляет сумму или произведение элементов массива, соответствующих определённому условию. Для вычисления суммы и произведения используется любой подходящий оператор цикла. Начальное значение переменной для суммы равно нулю, а для произведения равно единице. приводятся в виде фрагментов кода, которые можно использовать в обработчиках событий.

```

{
.....
// массив, нужный в нескольких обработчиках, можно описать вне
// обработчика событий
int[] MassivChisel = new int[100];
int Summa, Proizvedenie;
.....
Summa = 0;
Proizvedenie = 1;
for (int i = 0; i < 100; i++)
{
    Summa = Summa + MassivChisel[i];
    Proizvedenie = Proizvedenie* MassivChisel[i];
}

```

### **Счетчик**

Алгоритм счётчика — это алгоритм вычисления суммы числа ситуаций, соответствующих определённому условию, поэтому слагаемым на каждом шаге цикла является 1.

```

// В этом фрагменте вычисляется количество отрицательных
// элементов массива
int Kol0tr = 0;
for (int i = 0; i < 100; i++)
{
    if (MassivChisel[i] < 0)
        Kol0tr = Kol0tr + 1; // варианты записи Kol0tr += 1; Kol0tr++;
}

```

### **Поиск экстремума**

Алгоритм поиска экстремума приведен здесь на примере поиска максимального элемента массива и его номера (индекса).

```

// массив, нужный в нескольких обработчиках, можно описать вне
// обработчика событий
int[] MassivChisel = new int[100];
int Max, MaxNumber;
.....
Max = MassivChisel[1];
MaxNumber = 1;

for (int i = 0; i < 100; i++)
{
    if (MassivChisel[i] > Max)
    {
        Max = MassivChisel[i];
        MaxNumber = i;
    }
}

```

## Алгоритм обмена

Чтобы поменять значение двух переменных одного типа, необходимо использовать третью переменную такого же типа для временного хранения значения одной из переменных. Обычно этот алгоритм называют «методом трёх стаканов», подразумевая аналогию с ситуацией, когда используют третий стакан для обмена содержимого двух стаканов. В третий стакан переливают жидкость из одного стакана, в освободившийся стакан переливают жидкость из другого стакана, а затем из стакана временного хранения заливают жидкость во второй освободившийся стакан. В следующем фрагменте программного кода приводится алгоритм обмена значениями двух переменных целого типа

```
int x, y; // обмениваемые переменные
int temp; // от англ. слова temporary - временный

temp = x;
x = y;
y = temp;
```

Если необходимо часто использовать этот алгоритм (например для сортировки массива), то его можно оформить как процедуру – метод, который описывается в нужном классе. При этом передаваемые в процедуру параметры, содержащие обмениваемые значения, нужно описать служебным словом **ref**, чтобы они передавались по ссылке, а не значениями.

```
static void Obmen(ref int a, ref int b)
{
    int temp = a;
    a = b;
    b = temp;
}
```

Там, где понадобится обмен, можно использовать этот метод

```
int x, y;
.....
Obmen(x, y);
```

## Сортировка

Сортировкой называется процесс упорядочивания данных по какому-либо признаку (например, расположение числовых данных в порядке возрастания). Существует несколько алгоритмов сортировки, различающихся эффективностью, большей или меньшей сложностью для программирования. Здесь рассмотрен простейший метод сортировки, получивший название "метод пузырька". Суть его в том, что наименьший элемент массива, подобно поднимающемуся в жидкости пузырьку воздуха, перемещается в начало массива, меняясь местом с первым элементом. Затем этот процесс многократно повторяется с оставшейся частью массива до тех пор, пока все элементы не будут упорядочены.

Алгоритм сортировки сводится к сочетанию двух алгоритмов: алгоритма

поиска экстремума (минимума), который последовательно применяется к исходному массиву, затем к подмассиву, начинающемуся со второго элемента, с третьего и т. д. После нахождения минимума выполняется процедура обмена минимального элемента с первым в подмассиве. Процесс повторяется, пока все элементы не будут расположены в нужном порядке.

```
const int NumOfElements = 100;
int[] MassivChisel = new int[NumOfElements];
int MinNumber;
.....
for (int i = 0; i < NumOfElements-1; i++)
{
    MinNumber = i;
    for (int j = i+1; j < NumOfElements; j++)
        if (MassivChisel[j] < MassivChisel[MinNumber])
            MinNumber = j;
    if (MinNumber <> i)
        Obmen(MassivChisel[MinNumber], MassivChisel[i]);
}
```

## ЛАБОРАТОРНАЯ РАБОТА №5 РАБОТА СО СТРОКАМИ

**Цель лабораторной работы:** изучить основные понятия, относящиеся к классам и объектам, изучить действия со строковыми объектами, правила работы с элементом управления `ListBox`. Написать программу для работы со строками.

### 5.1. Классы и объекты

В объектно-ориентированном подходе существуют понятия *класс* и *объект*.

*Класс* – это программная единица, которая задаёт общий шаблон для конкретных объектов. Класс содержит все необходимые описания переменных, свойств и методов, которые относятся к объекту. Примером класса в реальной жизни является *понятие «автомобиль»*: как правило, автомобиль содержит некоторое количество колёс, дверей, имеет какой-то цвет, но эти конкретные детали в классе не описываются.

*Объект* – это экземпляр класса. Свойства объекта содержат конкретные данные, характерные для данного экземпляра. В реальной жизни примером объекта будет конкретный экземпляр автомобиля с 4 колёсами, 5 дверками и синего цвета.

### 5.2. Динамическое создание объектов

Чаще всего для размещения на форме кнопки, поля ввода или других управляющих элементов используется дизайнер среды Visual Studio:



нужный элемент выделяется в панели элементов и размещается на форме. Однако иногда создавать элементы нужно уже в процессе выполнения программы. Поскольку каждый элемент управления представляет собой отдельный класс, его помещение на форму программным способом включает несколько шагов:

- Создание экземпляра класса
- Привязка его к форме
- Настройка местоположения, размеров, текста и т. п.

Например, чтобы создать кнопку, нужно выполнить следующий код (его следует разместить в обработчике события Load или в каком-либо другом методе):

```
Button b = new Button();
```

Здесь объявляется переменная `b`, относящаяся к классу `Button`, как и в предыдущих лабораторных работах. Однако дальше идёт нечто новое: с помощью оператора `new` создаётся экземпляр класса `Button` и ссылка на него присваивается переменной `b`. При этом выполняется целый ряд дополнительных действий: выделяется память под объект, инициализируются все свойства и переменные.

Далее нужно добавить объект на форму. Для этого служит свойство `Parent`, которое определяет родительский элемент, на котором будет размещена кнопка:

```
b.Parent = this;
```

Ключевое слово `this` относится к тому объекту, в котором размещён выполняемый в данный момент метод. Поскольку все методы в лабораторных работах размещаются в классе формы, то и `this` относится к этому конкретному экземпляру формы.

Вместо формы кнопку можно поместить на другой контейнер. Например, если на форме есть элемент управления `Panel`, то можно поместить кнопку на него следующим образом:

```
b.Parent = panel1;
```

Чтобы задать положение и размеры кнопки нужно использовать свойства `Location` и `Size`:

```
b.Location = new Point(10, 20);  
b.Size = new Size(200, 100);
```

Обратите внимание, что `Location` и `Size` – это тоже объекты. Хотя

внутри у Location содержатся координаты x и y, задающие левый верхний угол объекта, не получится поменять одну из координат, нужно менять целиком весь объект Location. То же самое относится и к свойству Size.

На самом деле, каждый раз, когда на форму помещается новый элемент управления или вносятся какие-то изменения в свойства элементов управления, Visual Studio генерирует специальный служебный код, который проделывает приведённые выше операции по созданию и настройке элементов управления. Попробуйте поместить на форму кнопку, изменить у неё какие-нибудь свойства, а затем найдите в обозревателе решений ветку формы Form1, разверните её и сделайте двойной щелчок по ветке Form1.Designer.cs. Откроется файл с текстом программы на языке C#, которую среда создала автоматически. **Менять этот код вручную крайне не рекомендуется!** Однако можно его изучить, чтобы понять принципы создания элементов управления в ходе выполнения программы.

### 5.3. Область видимости

Переменные, объявленные в программе, имеют область видимости. Это значит, что переменная, описанная в одной части программы, не обязательно будет видна в другой. Вот наиболее часто встречающиеся ситуации:

- Переменные, описанные внутри метода, не будут видны за пределами этого метода. Например:

```
void MethodA()
{
    // Описываем переменную delta
    int delta = 7;
}

void MethodB()
{
    // Ошибка: переменная delta в этом методе неизвестна!
    int gamma = delta + 1;
}
```

- Переменные, описанные внутри блока или составного оператора, видны только внутри этого блока. Например:

```
void Method()
{
    if (a == 7)
    {
        int b = a + 5;
    }
}
```

```

    }
    // Ошибка: переменная b здесь уже неизвестна!
    MessageBox.Show(b.ToString());
}

```

- Переменные, описанные внутри класса, являются глобальными и доступны для всех методов этого класса, например:

```

class Form1 : Form
{
    int a = 5;

    void Method()
    {
        // Переменная a здесь действительна
        MessageBox.Show(a.ToString());
    }
}

```

## 5.4. Строковый тип данных

Для хранения строк в языке C# используется тип `string`. Чтобы объявить (и, как правило, сразу инициализировать) строковую переменную, можно написать следующий код:

```

string a = "Текст";
string b = "строки";

```

Над строками можно выполнять операцию сложения – в этом случае текст одной строки будет добавлен к тексту другой:

```

string c = a + " " + b; // Результат: Текст строки

```

Тип `string` на самом деле является псевдонимом для класса `String`, с помощью которого над строками можно выполнять ряд более сложных операций. Например, метод `IndexOf` может осуществлять поиск подстроки в строке, а метод `Substring` возвращает часть строки указанной длины, начиная с указанной позиции:

```

string a = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
int index = a.IndexOf("OP"); // Результат: 14 (счёт с 0)
string b = a.Substring(3, 5); // Результат: DEFGH

```

Если требуется добавить в строку специальные символы, это можно сделать с помощью так называемых escape-последовательностей, начинающихся с обратного слэша:

Escape-последовательность	Действие
\"	Кавычка
\\	Обратная косая черта
\n	Новая строка
\r	Возврат каретки
\t	Горизонтальная табуляция

## 5.5. Более эффективная работа со строками

Строки типа `string` представляют собой неизменяемые объекты: после того, как строка инициализирована, изменить её уже нельзя. Рассмотрим для примера следующий код:

```
string s = "Hello, ";  
s += "world!";
```

Здесь компилятор создаёт в памяти строковый объект и инициализирует его строкой «*Hello,* », а затем создаёт другой строковый объект и инициализирует его значением первого объекта и новой строкой «*world!*», а затем заменяет значение переменной `s` на новый объект. В результате строка `s` содержит именно то, что хотел программист, однако в памяти остаётся и изначальный объект со строкой «*Hello,* ». Конечно, со временем сборщик мусора уничтожит этот бесхозный объект, однако если в программе идёт интенсивная работа со строками, то таких бесхозных объектов может оказаться очень много. Как правило, это негативно сказывается на производительности программы и объеме потребляемой ею памяти.

Чтобы компилятор не создавал каждый раз новый строковый объект, разработчики языка `C#` ввели другой строковый класс: `StringBuilder`. Приведённый выше пример с использованием этого класса будет выглядеть следующим образом:

```
StringBuilder s = new StringBuilder("Hello, ");  
s.Append("world!");
```

Конечно, визуально этот код выглядит более сложным, зато при активном использовании строк в программе он будет гораздо

эффективнее. Помимо добавления строки к существующему объекту (Append) класс `StringBuilder` имеет ещё ряд полезных методов:

- `Insert` вставляет указанный текст в нужную позицию исходной строки
- `Remove` удаляет часть строки
- `Replace` заменяет указанный текст в строке на другой

Если нужно преобразовать объект `StringBuilder` в обычную строку, то для этого можно использовать метод `ToString()`:

```
StringBuilder s = new StringBuilder("Яблоко");  
string a = s.ToString();
```

## 5.6. Элемент управления `ListBox`

Элемент управления `ListBox` представляет собой список, элементы которого выбираются при помощи клавиатуры или мыши. Список элементов задается свойством `Items`. `Items` – это элемент, который имеет свои свойства и свои методы. Методы `Add`, `RemoveAt` и `Insert` используются для добавления, удаления и вставки элементов.

Объект `Items` хранит объекты, находящиеся в списке. Объект может быть любым классом – данные класса преобразуются для отображения в строковое представление методом `ToString()`. В нашем случае в качестве объекта будут выступать строки. Однако, поскольку объект `Items` хранит объекты, *приведённые* к типу `object`, перед использованием необходимо *привести* их обратно к изначальному типу, в нашем случае `string`:

```
string a = (string)listBox1.Items[0];
```

Для определения номера выделенного элемента используется свойство `SelectedIndex`.

## 5.7. Порядок выполнения индивидуального задания

Задание: написать программу подсчета числа слов в произвольной строке. В качестве разделителя между словами используется пробел. Для ввода строк использовать `ListBox`. Строки вводятся на этапе проектирования формы, используя окно свойств. Вывод результата организовать в метку `Label`.

Панель диалога будет иметь вид:

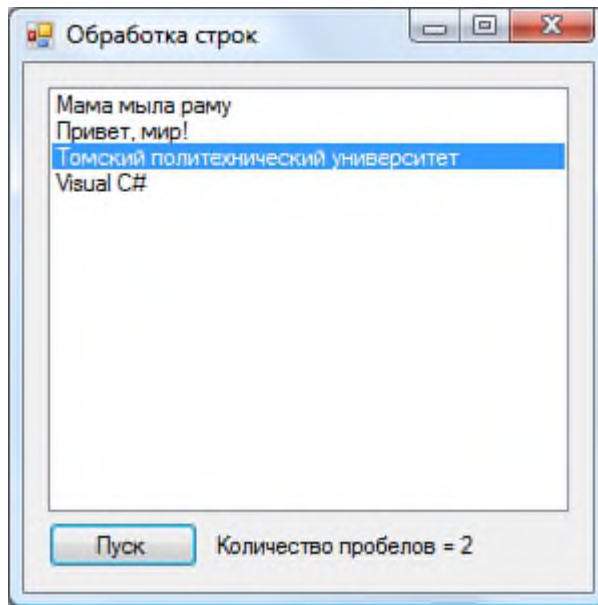


Рис. 5.1. Окно программы обработки строк

Текст обработчика нажатия кнопки «Пуск» приведен ниже.

```
private void button1_Click(object sender, EventArgs e)
{
    // Получаем номер выделенной строки
    int index = listBox1.SelectedIndex;
    // Считываем строку в переменную str
    string str = (string)listBox1.Items[index];
    // Узнаем количество символов в строке
    int len = str.Length;
    // Считаем, что количество пробелов равно 0
    int count = 0;
    // Устанавливаем счетчик символов в 0
    int i = 0;
    // Организуем цикл перебора всех символов в строке
    while (i < len)
    {
        // Если нашли пробел, то увеличиваем
        // счетчик пробелов на 1
        if (str[i] == ' ')
            count++;
        i++;
    }
    // Так как слов на единицу больше, чем пробелов, увеличим
    // счётчик на 1
    count++;
    label1.Text = "Количество слов = " + count.ToString();
}
```

### Индивидуальные задания

Во всех заданиях исходные данные вводить с помощью `ListBox`. Строки вводятся на этапе проектирования формы, используя окно свойств.

Вывод результата организовать в метку Label1.

**Замечание.** Большинство вариантов заданий достаточно сложны для начинающих.

Если выбранный Вами вариант индивидуального задания покажется слишком сложным, можно выбрать любой из вариантов 1–4, которые очень просты. В этом случае при защите выполненной работы будьте готовы объяснить описанный в обработчике алгоритм выполнения задания и продемонстрировать понимание работы программы.

1. Дана строка, состоящая из групп нулей и единиц. Посчитать количество нулей и единиц.
2. Посчитать в строке количество слов.
3. Найти количество знаков препинания в исходной строке.
4. Дана строка символов. Вывести на экран цифры, содержащиеся в строке.
5. Дана строка символов, состоящая из произвольных десятичных цифр, разделенных пробелами. Вывести количество четных чисел в этой строке.
6. Дана строка символов. Вывести на экран количество строчных русских букв, входящих в эту строку.
7. Дана строка символов. Вывести на экран только строчные русские буквы, входящие в эту строку.
8. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. В каждом слове заменить первую букву на прописную.
9. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Удалить первую букву в каждом слове.
10. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Поменять местами i- и j-ю буквы. Для ввода i и j на форме добавить свои поля ввода.
11. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Поменять местами первую и последнюю буквы каждого слова.
12. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Заменить все буквы латинского алфавита на знак «+».
13. Дана строка символов, содержащая некоторый текст на русском языке. Заменить все большие буквы «А» на символ «\*».
14. Дана строка символов, содержащая некоторый текст.

Разработать программу, которая определяет, является ли данный текст палиндромом, т. е. читается ли он слева направо так же, как и справа налево (например, «А роза упала на лапу Азора»).

15. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Сформировать новую строку, состоящую из чисел длин слов в исходной строке.

## **ЛАБОРАТОРНАЯ РАБОТА №6**

### **ЗНАКОМСТВО С ИНТЕРФЕЙСОМ И ОСНОВНЫМИ ВОЗМОЖНОСТЯМИ МАТЕМАТИЧЕСКОГО ПАКЕТА MATHCAD**

**Цель лабораторной работы:** изучить интерфейс, основные возможности системы Mathcad и способы работы в Mathcad. Научиться определять переменную и функцию.

#### **6.1. Основные понятия, средства и элементы интерфейса**

**Замечание.** Ниже все иллюстрации, названия элементов интерфейса и команды меню приведены для одной из предыдущих версий MathCad. В настоящее время наряду с версией MathCad 15 можно использовать более современные версии Mathcad Prime 6 или Mathcad Prime 7. Между этими версиями нет принципиальных различий, которые бы мешали пониманию приемов работы в новых версиях. На своем компьютере можно получить доступ к этим версиям через удалённый рабочий стол на сервере ТПУ по адресу [var.tpu.ru](http://var.tpu.ru).

MathCad — это интегрированная система, которая предоставляет пользователю широкий набор средств для проведения разнообразных математических расчетов и преобразований, а также для создания документов, оформляющих результаты их выполнения. Пользователь получает возможность просто и наглядно в привычной для математика форме вводить математические выражения и получать результат вычислений. Представлены следующие средства:

- численные математические методы для решения алгебраических и дифференциальных уравнений, систем таких уравнений;
- методы численного дифференцирования и интегрирования;
- дискретные интегральные преобразования;
- разнообразные матричные, векторные, статистические, трансцендентные и другие функции;
- некоторые возможности символьной математики: символьная алгебра, решение уравнений, действия с матрицами, интегральные преобразования, символьные вычисления и упрощения.

Система позволяет строить графики в декартовых и полярных координатах, трехмерные гистограммы, векторные поля и карты линий



уровня, возможно создание анимационных клипов и импорт графики из других приложений. Система предоставляет также возможность динамического обмена данными с другими программами.

Перечисленные выше возможности делают MathCad незаменимым помощником для студентов при выполнении заданий по тем учебным дисциплинам, где требуется проведение вычислений различной степени сложности, а также при проведении студентами научных исследований.

MathCad имеет стандартный оконный интерфейс Windows. Среди панелей инструментальных кнопок особый интерес вызывает панель МАТЕМАТИКА, предоставляющая доступ к панелям, которые называются палитрами (рис. 6.1).

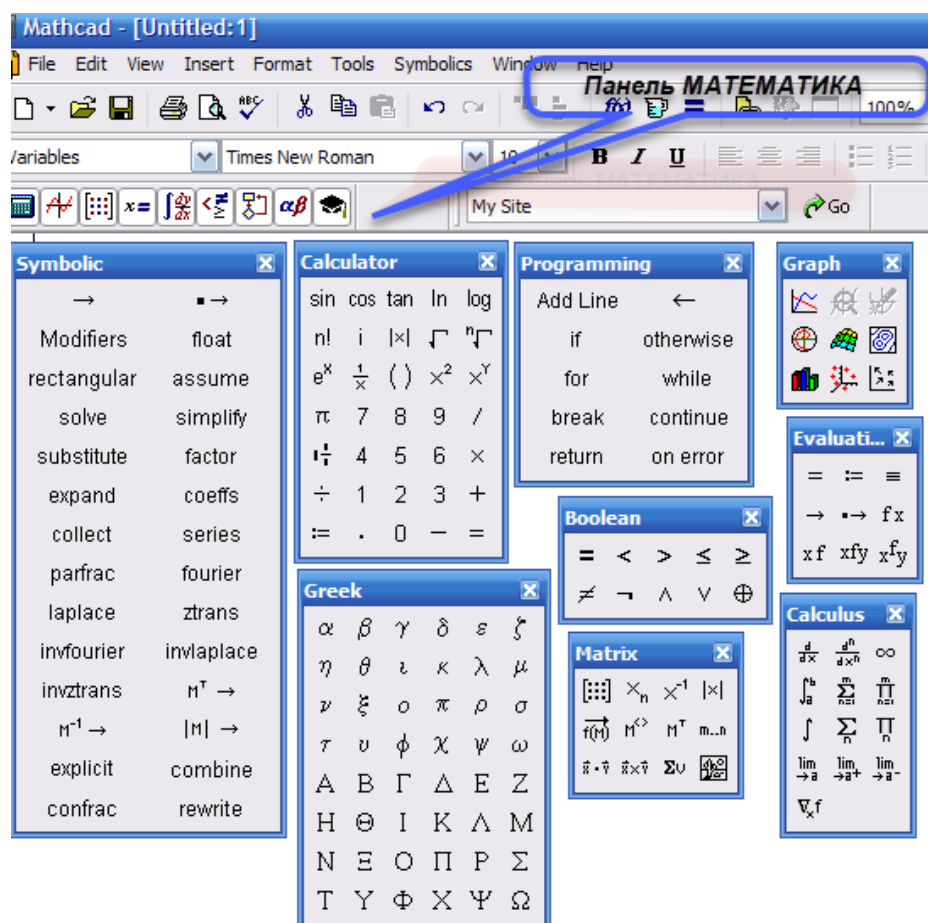


Рис. 6.1. Палитры, доступные из панели МАТЕМАТИКА

- Палитра Calculator служит для вставки основных математических операций.
- Палитра Graph служит для вставки графика в документ.
- Палитра Matrix служит для вставки матрицы, для работы с матрицами и матричными операциями.
- Палитра Evaluation представляет операторы вычисления.
- Палитра Calculus представляет операторы интегрирования, дифференцирования, суммирования.

- Палитра Boolean представляет булевы операторы и предназначена для вставки логических или булевых операций.
- Палитра Programming служит для программирования средствами MathCad.
- Палитра Greek представляет греческие символы.
- Палитра Symbolic служит для вставки символьных операторов.

В рабочей области окна можно вводить математические выражения, текстовые поля и элементы программирования.

**Курсор ввода (визир).** Чтобы отметить место, куда вводить формулу (или текст), используется крестообразный курсор ввода + («щелкнуть» указателем мыши в нужном месте либо передвинуть его клавишами-стрелками клавиатуры). При вводе на месте курсора появляется вертикальная и горизонтальная линия ввода синего цвета, отмечающее место редактирования в данный момент. Для ввода используется клавиатура или палитра.

**Простые вычисления.** MathCad можно использовать как калькулятор для вычисления выражений. Для этого достаточно ввести выражение и ввести знак равно = с клавиатуры или палитры Calculator или Evaluation. Действия происходят с числами или числовыми выражениями. Более сложные вычисления производятся с помощью переменных.

**Переменные.** Чтобы использовать переменную, необходимо её определить. Для этого нужно ввести имя переменной и присвоить ей значение. Для присваивания значений переменным используется оператор присваивания :=. Чтобы ввести оператор присваивания, нужно набрать имя переменной и нажать клавишу двоеточия :. Имя переменной не может начинаться с цифры, символа подчеркивания, штриха или процента. Все буквы имени должны иметь один стиль и шрифт. Имена не могут совпадать с именами встроенных функций, констант и размерностей.

**Ввод уравнений.** Чтобы записать уравнение, между левой и правой частями уравнения нужно поставить знак равенства с помощью нажатия на клавиатуре Ctrl + = или использовать палитру Boolean.

**Функции.** В MathCad используются встроенные функции и функции, заданные пользователем. Имена встроенных функций можно вводить с клавиатуры или вызывать с панели инструментов (f(x)), некоторые можно ввести с палитры Calculator. Функции, заданные пользователем, записываются в обычной для математика форме. Например,  $g(x,y) := x + 2 \cdot y$ . Эта запись называется определением функции. При определении используется оператор присваивания. Переменные, в выражении для функции должны быть определены раньше. «Раньше» в MathCad означает положение выше или левее текущего в рабочей области окна. Для вычисления функции при определенных значениях переменных необходимо до вычисления функции присвоить значения переменным или подставить в определение функции значения и после функции поставить

знак =. Например,  $g(1,1)=3$ .

### Индивидуальные задания

Ниже приведены 15 вариантов заданий. Вычислите функцию заданного вам варианта при  $x = 1$ .

1. а)  $f(x) = \frac{3}{5}x^5 - \frac{1}{2x^4} - \frac{2}{\sqrt[4]{x^3}} + 7$ ; б)  $f(x) = \frac{e^x - \sin x}{\cos x + \sqrt{x}}$ ;

в)  $f(x) = \sqrt[4]{x^2 + \ln x}$ .

2. а)  $f(x) = \frac{2}{3}x^3 - \frac{1}{3x^9} + \frac{5}{\sqrt[5]{x^3}} - 6$ ; б)  $f(x) = (1 - x^2)(\operatorname{tg} x + 3^x)$ ;

в)  $f(x) = e^{\sin 5x - 3}$ .

3. а)  $f(x) = \frac{4}{5}x^5 - \frac{1}{6x^6} + \frac{7}{\sqrt[7]{x^3}} + 2$ ; б)  $f(x) = \frac{\ln x - \operatorname{tg} x}{7^x - 5}$ ;

в)  $f(x) = \sqrt{x^5 + \sin 5x}$ .

4. а)  $f(x) = 3x^2 - \frac{1}{7x^7} + \frac{3}{\sqrt[3]{x^2}} + 1$ ; б)  $f(x) = \frac{5^x - \ln x}{\cos x - 3}$ ;

в)  $f(x) = \arcsin(5x^3 + 1)$ .

5. а)  $f(x) = 4x^5 - \frac{7}{4x^4} - \frac{3}{\sqrt[3]{x^2}} + 2$ ; б)  $f(x) = \frac{\sin x - \cos x}{3^x - \ln x}$ ;

в)  $f(x) = \cos(2x^2 + 3)$ .

6. а)  $f(x) = 2x^5 + \frac{4}{5x^5} - \frac{2}{\sqrt{x}} + 3$ ; б)  $f(x) = (x^2 - 3)(\sin x + 5^x)$ ;

в)  $f(x) = e^{\sin 7x + 3}$ .

7. а)  $f(x) = 6x^5 - \frac{5}{3x^3} + \frac{6}{\sqrt[4]{x^3}} + 2$ ; б)  $f(x) = \frac{3^x + \cos x}{\ln x - \sqrt{x}}$ ;

в)  $f(x) = \arctg \sqrt{x^2 + 1}$ .

8. а)  $f(x) = 3x^4 - \frac{5}{6x^6} - \frac{2}{\sqrt{x^3}}$ ; б)  $f(x) = (e^x + \operatorname{tg} x)(\ln x - 2)$ ;

в)  $f(x) = \operatorname{tg}(3^x - 5)$ .

$$9. a) f(x) = 5x^3 - \frac{3}{4x^4} - 7\sqrt[5]{x^3} - 2; \quad б) f(x) = \frac{6^x - \cos x}{\operatorname{tg} x + \sqrt{x^3}};$$

$$в) f(x) = e^{2x} + 3x \cdot \operatorname{tg} 2x.$$

$$10. a) f(x) = 4x^5 - \frac{3}{x^3} - \frac{2}{\sqrt[5]{x^3}} - 1; \quad б) f(x) = (\ln x + \operatorname{tg} x)(\sqrt{x} - e^x);$$

$$в) f(x) = (\operatorname{tg} 3x)^5.$$

$$11. a) f(x) = \frac{2}{7}x^7 - \frac{1}{4x^4} + \frac{5}{3\sqrt{x}} + 3; \quad б) f(x) = \frac{\operatorname{tg} x + \sqrt{x}}{3^x - \sin x};$$

$$в) f(x) = \ln(x^3 + e^x - 2).$$

$$12. a) f(x) = 4x^2 + \frac{3}{5x^3} - \frac{6}{\sqrt[3]{x^4}}; \quad б) f(x) = (\sin x + 2^x)(\ln x - e^x);$$

$$в) f(x) = e^{\operatorname{tg}(x^2-5)}.$$

$$13. a) f(x) = \frac{2}{5}x^5 - \frac{3}{4x^4} - 6\sqrt[3]{x} - 4; \quad б) f(x) = \frac{\ln x - e^x}{\operatorname{ctg} x - \cos x};$$

$$в) f(x) = \arcsin \sqrt{3^x - \frac{2}{x}}.$$

$$14. a) f(x) = \frac{5}{3}x^3 - \frac{2}{3x^6} + \frac{1}{\sqrt{x}}; \quad б) f(x) = (2 + \sqrt{x})(\operatorname{ctg} x - e^x);$$

$$в) f(x) = (3^x - \ln x)^2.$$

$$15. a) f(x) = \frac{7}{9}x^9 - \frac{4}{3x^3} - 5\sqrt[5]{x^3} - 1; \quad б) f(x) = \frac{\operatorname{ctg} x - 3}{5^x + \sqrt{x}};$$

$$в) f(x) = \ln(\ln x - \sin x).$$

## ЛАБОРАТОРНАЯ РАБОТА №7 ПОСТРОЕНИЕ ГРАФИКА ТАБУЛИРОВАННОЙ ФУНКЦИИ

**Цель лабораторной работы:** приобрести начальные навыки простых вычислений и построения графиков в Mathcad.

### 7.1. Повторяющиеся вычисления, дискретные переменные, построение графиков

Часто возникает необходимость повторять одинаковые вычисления по одним и тем же формулам, но при разных значениях переменных. Такие вычисления называются *повторяющимися*. Для повторяющихся

вычислений в MathCad используется специальный тип переменных - *дискретные аргументы*.

Дискретный аргумент (дискретная переменная, ранжированная переменная) - это переменная, значением которой является совокупность значений в заданном диапазоне с заданным шагом изменения значений. Если в каком-то выражении есть дискретная переменная, то MathCad вычисляет выражение столько раз, сколько значений содержит дискретная переменная, т. е. проводит повторяющиеся вычисления.

Чтобы определить дискретную переменную для последующего использования ее в расчетах, нужно:

- определить обычную переменную (См. лабораторную работу № 6); значение этой переменной должно быть равным началу выбранного числового диапазона
- установить маркер ввода сразу за этим определением
- ввести запятую и затем следующее число из диапазона; тем самым автоматически задается шаг изменения значений
- нажать клавишу “точка с запятой”; это приведет к появлению на экране двух точек подряд (..)
- набрать сразу за этими точками значение верхней границы диапазона
- щелкнуть на свободном месте мышкой;

Например, если определить  $y := 0, 3..15$ , то значениями  $y$  будут 0, 3, 6, 9, 12, 15.

После определения дискретной переменной ее можно вывести в окне MathCad, набрав имя переменной со знаком =. Вместо знака = появится табличка со всеми значениями дискретной переменной из диапазона.

Можно использовать значения такой переменной для вычислений без промежуточного вывода их на экран. Для этого следует набрать вычисляемое выражение с дискретной переменной и поставить в конце набора знак =. Тогда появится таблица со всеми значениями выражения, вычисленными для этой дискретной переменной. Дальнейшее расширение возможностей расчетов связано с определением функций.

Чтобы определить функцию, нужно сначала набрать ее имя с аргументом в круглых скобках, затем символ двоеточия. В процессе набора двоеточие отобразится знаком присваивания. В появившемся поле ввода следует набрать выражение, с помощью которого вычисляется значение функции. Для получения набора значений функции следует набрать ее имя с именем дискретного аргумента в круглых скобках и завершить ввод знаком равенства. Чтобы получить результат для одного конкретного аргумента, после имени функции в скобках указывают значение этого аргумента.

MathCad позволяет строить разнообразные графики с помощью пунктов меню Graphics (Графика) или палитры Graph. Чтобы построить график, нужно:

- определить дискретную переменную (например,  $x$ ) в заданном диапазоне с требуемым шагом
- определить функцию (например,  $f(x) = x^2$ )
- щелкнуть кнопкой мыши на свободном месте, где предполагается разместить график
- выбрать и выполнить пункт меню **Insert/Graph/ X-Y Plot** (Декартов график) из меню Graphics или использовать палитру Graph; MathCad создаст пустой график (рис 6.2, а).

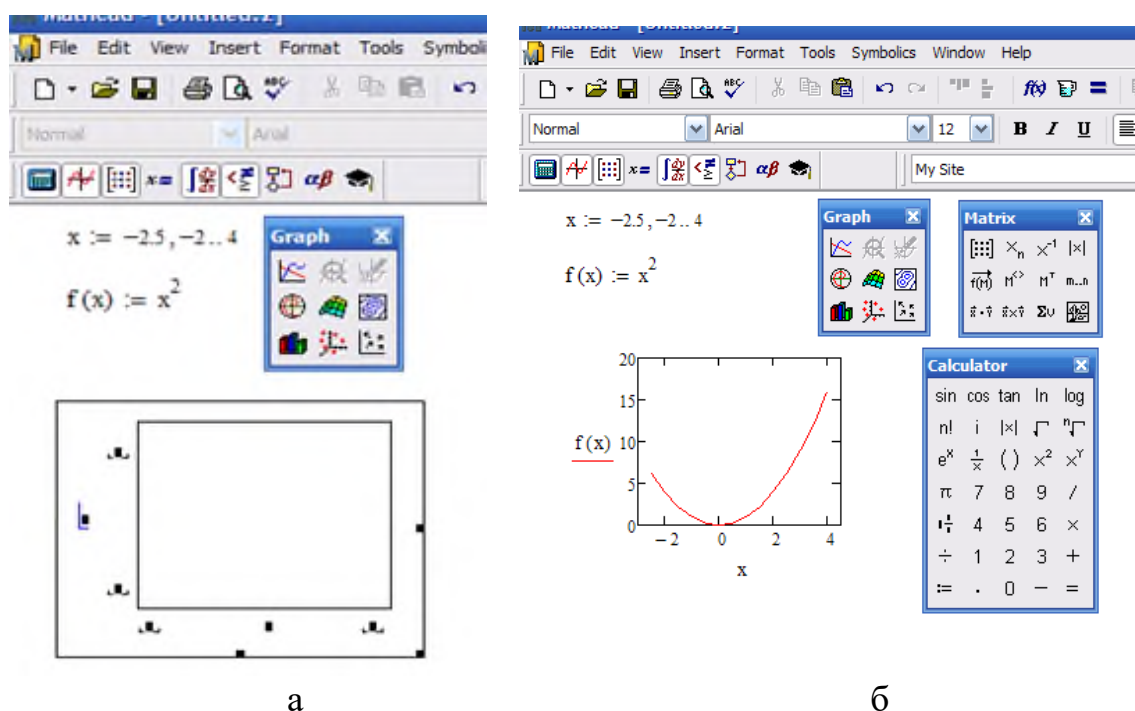


Рис. 6.2. Пример построения графика: а – задание функции, ее аргументов и вызов шаблона для построения графика, б – иллюстрация результата

В этом графике под осью абсцисс создается поле ввода, в которое нужно ввести имя переменной  $x$ . Затем нужно щелкнуть кнопкой мыши напротив середины оси ординат и в появившемся поле ввода указать имя функции  $f(x)$ .

Оставшиеся поля предназначены для ввода границ на осях - максимального и минимального значений, откладываемых на оси. Если эти поля не заполнять, то MathCad автоматически заполнит их при создании графика. После заполнения полей нужно щелкнуть кнопкой мыши вне области графика. MathCad построит график автоматически.

## 7.2. Порядок выполнения индивидуального задания

1. Определить дискретную переменную.
2. Определить функцию этой переменной.

3. «Щелкнуть» мышью там, где нужно создать график.
4. Выбрать X-Y Plot из меню Insert/Graph, появится пустой график с 6 пустыми полями, которые нужно заполнить.
5. Пустое поле в середине горизонтальной оси предназначено для независимой переменной. Введите туда дискретную переменную.
6. Пустое поле в середине вертикальной оси предназначено для функции, график которой нужно построить. Введите туда имя функции дискретной переменной, находящейся на горизонтальной оси.
7. Другие 4 поля предназначены для указания диапазонов и заполняются по умолчанию или вручную.
8. Для отображения графика следует щелкнуть мышью вне его поля или нажать F9.
9. Чтобы представить несколько зависимостей на одном графике, введите первую переменную по оси ординат с запятой в конце. Ниже появится пустое поле для второй переменной (выражения), введите вторую переменную с запятой в конце, ниже появится третье поле и т.д. (до 16 графиков).

### Индивидуальные задания

Ниже приведены 15 вариантов заданий. Постройте график функции на интервале  $x$  от  $-10$  до  $10$ .

1.  $f(x) = x^3 - 9x^2 + 24x - 15$ .
2.  $f(x) = -x^3 - 12x^2 - 45x + 51$ .
3.  $f(x) = x^3 - 3x + 2$ .
4.  $f(x) = -x^3 + 9x^2 - 24x + 21$ .
5.  $f(x) = x^3 + 3x^2 - 2$ .
6.  $f(x) = -x^3 - 3x^2 - 1$ .
7.  $f(x) = x^3 - 9x^2 + 24x - 12$ .
8.  $f(x) = -x^3 + 9x^2 - 24x + 15$ .
9.  $f(x) = x^3 - 12x^2 + 45x - 45$ .
10.  $f(x) = -x^3 + 3x - 7$ .
11.  $f(x) = x^3 + 6x^2 + 9x + 3$ .
12.  $f(x) = -x^3 - 9x^2 - 24x - 18$ .
13.  $f(x) = x^3 - 3x^2 + 9$ .
14.  $f(x) = -x^3 - 6x^2 - 9x - 6$ .
15.  $f(x) = x^3 - 6x^2 + 9x + 2$ .

## ЛАБОРАТОРНАЯ РАБОТА №8 РЕШЕНИЕ СИСТЕМЫ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ В MATHCAD

*Цель лабораторной работы:* научиться пользоваться средствами математического пакета MathCad для решения систем линейных алгебраических уравнений

### 8.1. Массивы: векторы и матрицы

MathCad позволяет решать линейные системы уравнений, которые в матричной форме имеют вид  $A \cdot x = b$  (где  $A$  - матрица коэффициентов при неизвестном векторе  $x$ , а  $b$  - вектор-столбец, составленный из свободных членов в уравнениях). Для представления системы уравнений в такой форме используется специальное понятие **массива**. В MathCad используются следующие типы массивов:

- одномерный массив, вектор – это столбец чисел;
- двумерный массив, матрица – прямоугольная таблица чисел.

Существует несколько способов создания массивов (рис. 8.1). Один из наиболее простых способов: ввести имя массива (например,  $a$ ) и знак присваивания ( $:=$ ), затем на палитре Matrix выбрать первый инструмент (Insert Matrix), указать количество строк и столбцов, подтвердить свой выбор (т.е. нажать ОК), ввести все элементы массива вручную.

К матрицам, содержащим числовые значения, можно применять различные алгебраические действия сложение, вычитание, умножение, а также специальные векторные и матричные операторы. Для этих операций можно использовать палитры Calculator и Matrix.

В тех случаях, когда в расчетах нужно обращаться к отдельным элементам массива (вектора или матрицы), используют понятие нижнего индекса. Чтобы указать значение нижнего индекса, нужно после набора имени массива нажать клавишу левой скобки [. Затем следует ввести в пустое поле ввода значение нижнего индекса компоненты вектора, либо через запятую указать два значения индексов, соответствующих нужному элементу матрицы. При указании значений нижнего индекса следует иметь в виду, что по умолчанию нумерация элементов массивов в MathCad начинается с нуля.

Кроме того, MathCad дает возможность выделять из матрицы отдельный столбец. Для этого используется верхний индекс, с помощью которого указывается номер выделяемого столбца. Чтобы это сделать, нужно после набора имени матрицы нажать одновременно клавиши CTRL и 6. (Такого же результата можно достичь с помощью палитры Matrix). Это приведет к появлению вверху имени матрицы в угловых скобках



пустого поля ввода, в котором указывается номер нужного столбца. После нажатия знака равенства на экране появится столбец значений.

Если необходимо выделить значения строки матрицы, ее предварительно следует транспонировать, а затем использовать верхний индекс для выделения того столбца, в который отразилась нужная строка.

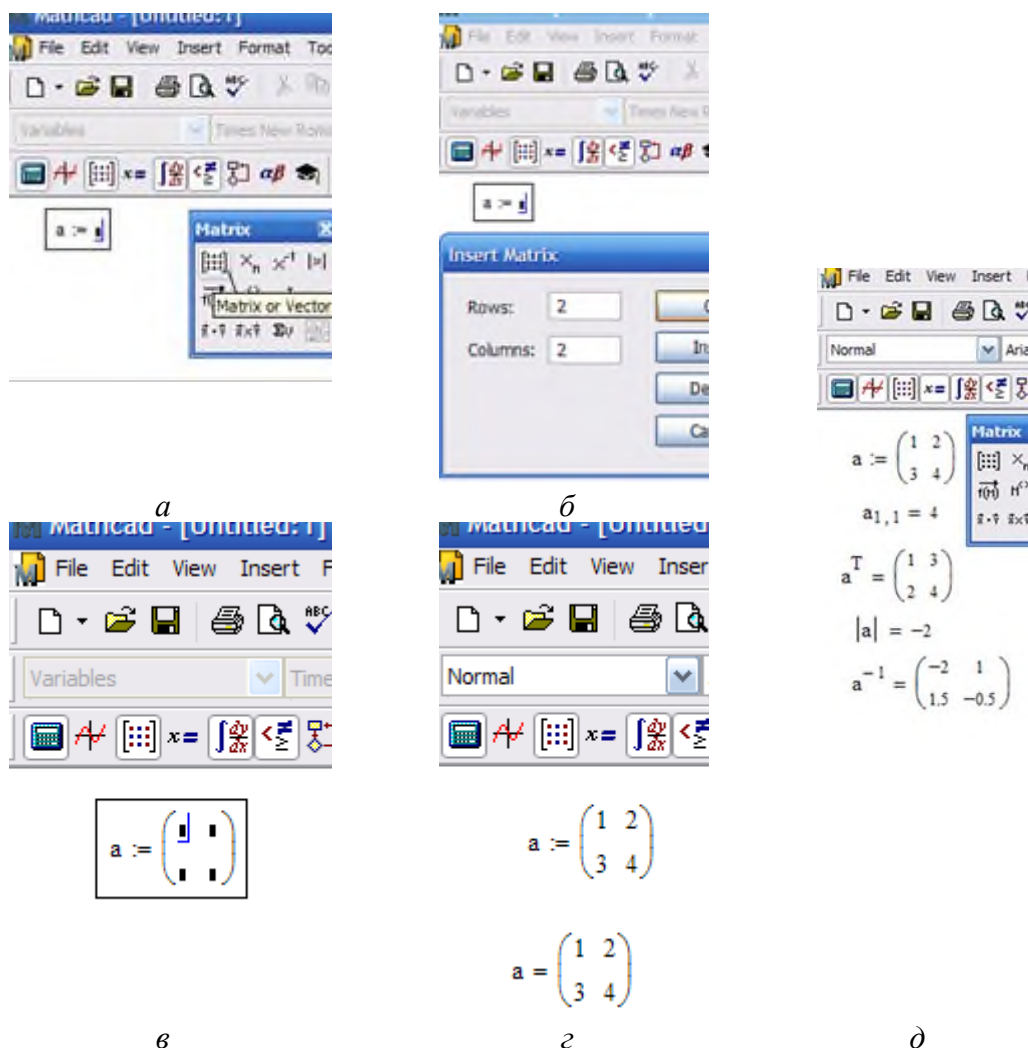


Рис. 8.1. Пример ввода матрицы и проведение операций с ней: а – создание матрицы с помощью палитры Matrix; б – создание шаблона матрицы; в – пустая матрица перед заполнением; г – представление матрицы; д – результаты выполненных операций с матрицей

## 8.2. Решение системы уравнений

В MathCad систему уравнений можно решить двумя способами:

- используя обращение матрицы  $A$ ; решение в этом случае получается в виде вектора-столбца  $x$  и вычисляется по формуле  $x = A^{-1} \cdot b$ ; здесь  $A^{-1}$  – обратная матрица;
- применяя специальную функцию ***lsolve***( $A$ ,  $b$ ); решение получается также в виде вектора-столбца.

При этом следует обращать внимание на то, чтобы детерминант матрицы  $A$  не равнялся нулю. Такая матрица называется вырожденной. Значение детерминанта в сомнительных случаях всегда можно быстро определить, используя соответствующую кнопку с палитры матричных и векторных операций. В случаях, когда детерминант матрицы почти равен нулю, матрица почти вырождена, и имеет большое число обусловленности. Такие случаи также непригодны для решения системы указанными способами. Число обусловленности матрицы можно определить, используя одну из функций  $\text{cond1}(A)$ ,  $\text{cond2}(A)$ ,  $\text{conde}(A)$ ,  $\text{condi}(A)$ .

На рис. 8.2. показан пример решения системы уравнений обращением матрицы.

**Решение системы уравнений обращением матрицы**

$$3x + 6y = 9$$

Исходная система уравнений

$$2x + 0.54y = 4$$

Создание матрицы коэффициентов и вектора правой части

$$A := \begin{pmatrix} 3 & 6 \\ 2 & 0.54 \end{pmatrix} \qquad b := \begin{pmatrix} 9 \\ 4 \end{pmatrix}$$

Решение системы

$$z := A^{-1} \cdot b$$

$$z = \begin{pmatrix} 1.844 \\ 0.578 \end{pmatrix} \qquad \begin{array}{l} \text{Искомое значение } x \\ \text{Искомое значение } y \end{array}$$

Проверка решения

$$A \cdot z = \begin{pmatrix} 9 \\ 4 \end{pmatrix} \qquad +$$

**Решение системы уравнений использованием функции *lsolve***

$$\text{lsolve}(A, b) = \begin{pmatrix} 1.844 \\ 0.578 \end{pmatrix} \qquad \begin{array}{l} \text{Искомое значение } x \\ \text{Искомое значение } y \end{array}$$

Рис. 8.2. Пример решения системы уравнений

## Индивидуальные задания

Решите систему линейных уравнений своего варианта двумя способами и сделайте проверку.

$$1. \begin{cases} x + 2y - z = 5, \\ 2x - y + 5z = -7, \\ 5x - y + 2z = -4. \end{cases} \quad 2. \begin{cases} 2x + 3y - 5z = 1, \\ 3x + 4y - 3z = 2, \\ x - 3y + 7z = 5. \end{cases} \quad 3. \begin{cases} 7x - 3y + z = 5, \\ x + 2y - z = -4, \\ 3x + y - z = -3. \end{cases}$$

$$4. \begin{cases} 5x + y + 6z = -3, \\ 4x + 3y - z = 2, \\ x + 2y - 5z = 3. \end{cases} \quad 5. \begin{cases} 5x - 3y + z = -3, \\ 3x - y + 2z = 1, \\ x + 5y + z = 1. \end{cases} \quad 6. \begin{cases} 8x + 2y - 7z = 3, \\ x - 3y + 5z = 3, \\ 5x - 2y + 4z = 7. \end{cases}$$

$$7. \begin{cases} 3x - 4y + z = 5, \\ 2x - y + 3z = 1, \\ x + 5y - z = 3. \end{cases} \quad 8. \begin{cases} 7x - y + 2z = 5, \\ 2x + y - 3z = -7, \\ x - 5y + z = 7. \end{cases} \quad 9. \begin{cases} x - 4y - z = -3, \\ 3x + 7y + z = -1, \\ 2x + 3y - z = -4. \end{cases}$$

$$10. \begin{cases} x + y + z = 3, \\ 3x - 2y + z = 2, \\ 5x + 2y - 7z = 0. \end{cases} \quad 11. \begin{cases} x - 5y + z = 1, \\ 3x + y - 2z = -7, \\ 2x + 7y + z = 0. \end{cases} \quad 12. \begin{cases} 3x - 4y + 7z = -1, \\ x + 7y + 2z = 0, \\ 2x - 3y + z = 3. \end{cases}$$

$$13. \begin{cases} 5x - 3y + z = 9, \\ 3x - 7y + 6z = 0, \\ x + 2y + z = 1. \end{cases} \quad 14. \begin{cases} x + 2y + 5z = -1, \\ 5x + y - 3z = 5, \\ 7x - 4y - 3z = -5. \end{cases} \quad 15. \begin{cases} x - y + 7z = -3, \\ 2x + y - 5z = 0, \\ 3x + 2y - 5z = 1. \end{cases}$$

## ЛАБОРАТОРНАЯ РАБОТА №9 СИМВОЛЬНАЯ МАТЕМАТИКА. ВЫЧИСЛЕНИЕ ПРОИЗВОДНЫХ ПЕРВОГО И ВЫСШИХ ПОРЯДКОВ В MATHCAD

**Цель лабораторной работы:** научиться пользоваться средствами математического пакета MathCad для аналитических вычислений.

### 9.1. Символьные вычисления. Вычисление производных

MathCad позволяет выполнять не только численные расчеты, но и символьные (аналитические) вычисления. Это такие расчеты, где результатами преобразования выражений в ходе вычислений являются также выражения, представленные в символьном виде. При этом может быть задана желаемая форма преобразованных выражений. Первоначальное выражение можно разложить на множители, проинтегрировать, дифференцировать, разложить в ряд, реализовать некоторые интегральные преобразования и выполнить иные

математические операции, например, матричные. Все символьные вычисления можно осуществить через пункт меню **Symbolics**. При выполнении символьных вычислений нужно иметь ввиду, что :

- многие вычисления могут быть выполнены только численно;
- в ряде случаев получаются такие длинные ответы, что удобнее использовать соответствующие численные расчеты с последующим графическим представлением этих ответов;
- когда уравнение имеет несколько решений, MathCad иногда возвращает одно из них и запрашивает, показать ли весь результат, хранящийся в буфере обмена; при нажатии **OK** MathCad показывает вектор решений и слово Root;
- символьные и численные расчеты выполняются двумя разными процессорами, между которыми можно организовать взаимодействие для сложных расчетов. Например, упростить сначала символьные выражения перед их использованием в численных вычислениях.

**Вычисление первой и высших производных.** Чтобы получить первую производную от заданной функции, следует:

- вызвать оператор производной, щелкнув мышкой на нужной кнопке на палитре Calculus
- заполнить поля ввода для функции и переменной, по которой выполняется дифференцирование
- заключить выражение в выделяющую рамку
- нажать клавиши **Ctrl + .** (или на палитре Evaluation нажать клавишу  $\rightarrow$ ) и нажать клавишу **Enter**.

Это приведет к появлению результата под функцией.

Получить первую производную от заданной функции можно и не используя оператор производной. Для этого нужно набрать функцию, выделить переменную, по которой выполняется дифференцирование (поставить рядом с ней маркер ввода) и выполнить пункт меню **Symbolics/Variable/Differentiate**. Результат появится под функцией.

Чтобы получить высшую производную ( $n$ -ю производную) от заданной функции, следует:

- вызвать оператор  $n$ -ой производной, щелкнув мышкой на нужной кнопке на палитре Calculus
- заполнить поля ввода для функции и переменной, по которой выполняется дифференцирование
- заключить выражение в выделяющую рамку
- нажать клавиши **Ctrl + .** (или на палитре Evaluation нажать клавишу  $\rightarrow$ ), и нажать клавишу **Enter**.

Результат появится под функцией.

### **Индивидуальные задания**

Найдите первую и вторую производные функций своего варианта.

$$1. a) f(x) = \frac{3}{5}x^5 - \frac{1}{2x^4} - \frac{2}{\sqrt[4]{x^3}} + 7; \quad б) f(x) = \frac{e^x - \sin x}{\cos x + \sqrt{x}};$$

$$в) f(x) = \sqrt[4]{x^2 + \ln x}.$$

$$2. a) f(x) = \frac{2}{3}x^3 - \frac{1}{3x^9} + \frac{5}{\sqrt[5]{x^3}} - 6; \quad б) f(x) = (1 - x^2)(\operatorname{tg} x + 3^x);$$

$$в) f(x) = e^{\sin 5x - 3}.$$

$$3. a) f(x) = \frac{4}{5}x^5 - \frac{1}{6x^6} + \frac{7}{\sqrt[7]{x^3}} + 2; \quad б) f(x) = \frac{\ln x - \operatorname{tg} x}{7^x - 5};$$

$$в) f(x) = \sqrt{x^5 + \sin 5x}.$$

$$4. a) f(x) = 3x^2 - \frac{1}{7x^7} + \frac{3}{\sqrt[3]{x^2}} + 1; \quad б) f(x) = \frac{5^x - \ln x}{\cos x - 3};$$

$$в) f(x) = \arcsin(5x^3 + 1).$$

$$5. a) f(x) = 4x^5 - \frac{7}{4x^4} - \frac{3}{\sqrt[3]{x^2}} + 2; \quad б) f(x) = \frac{\sin x - \cos x}{3^x - \ln x};$$

$$в) f(x) = \cos(2x^2 + 3).$$

$$6. a) f(x) = 2x^5 + \frac{4}{5x^5} - \frac{2}{\sqrt{x}} + 3; \quad б) f(x) = (x^2 - 3)(\sin x + 5^x);$$

$$в) f(x) = e^{\sin 7x + 3}.$$

$$7. a) f(x) = 6x^5 - \frac{5}{3x^3} + \frac{6}{\sqrt[4]{x^3}} + 2; \quad б) f(x) = \frac{3^x + \cos x}{\ln x - \sqrt{x}};$$

$$в) f(x) = \operatorname{arctg} \sqrt{x^2 + 1}.$$

$$8. a) f(x) = 3x^4 - \frac{5}{6x^6} - \frac{2}{\sqrt{x^3}}; \quad б) f(x) = (e^x + \operatorname{tg} x)(\ln x - 2);$$

$$в) f(x) = \operatorname{tg}(3^x - 5).$$

$$9. a) f(x) = 5x^3 - \frac{3}{4x^4} - 7\sqrt[5]{x^3} - 2; \quad б) f(x) = \frac{6^x - \cos x}{\operatorname{tg} x + \sqrt{x^3}};$$

$$в) f(x) = e^{2x} + 3x \cdot \operatorname{tg} 2x.$$

$$10. a) f(x) = 4x^5 - \frac{3}{x^3} - \frac{2}{\sqrt[5]{x^3}} - 1; \quad б) f(x) = (\ln x + \operatorname{tg} x)(\sqrt{x} - e^x);$$

$$в) f(x) = (\operatorname{tg} 3x)^5.$$

$$11. a) f(x) = \frac{2}{7}x^7 - \frac{1}{4x^4} + \frac{5}{3\sqrt{x}} + 3; \quad б) f(x) = \frac{\operatorname{tg} x + \sqrt{x}}{3^x - \sin x};$$

$$в) f(x) = \ln(x^3 + e^x - 2).$$

$$12. a) f(x) = 4x^2 + \frac{3}{5x^3} - \frac{6}{\sqrt[3]{x^4}}; \quad б) f(x) = (\sin x + 2^x)(\ln x - e^x);$$

$$в) f(x) = e^{\operatorname{tg}(x^2-5)}.$$

$$13. a) f(x) = \frac{2}{5}x^5 - \frac{3}{4x^4} - 6\sqrt[3]{x} - 4; \quad б) f(x) = \frac{\ln x - e^x}{\operatorname{ctg} x - \cos x};$$

$$в) f(x) = \arcsin \sqrt{3^x - \frac{2}{x}}.$$

$$14. a) f(x) = \frac{5}{3}x^3 - \frac{2}{3x^6} + \frac{1}{\sqrt{x}}; \quad б) f(x) = (2 + \sqrt{x})(\operatorname{ctg} x - e^x);$$

$$в) f(x) = (3^x - \ln x)^2.$$

$$15. a) f(x) = \frac{7}{9}x^9 - \frac{4}{3x^3} - 5\sqrt[5]{x^3} - 1; \quad б) f(x) = \frac{\operatorname{ctg} x - 3}{5^x + \sqrt{x}};$$

$$в) f(x) = \ln(\ln x - \sin x).$$

**ЛАБОРАТОРНАЯ РАБОТА №10**  
**СИМВОЛЬНАЯ МАТЕМАТИКА. ВЫЧИСЛЕНИЕ**  
**НЕОПРЕДЕЛЁННЫХ И ОПРЕДЕЛЁННЫХ ИНТЕГРАЛОВ В**  
**MATHCAD**

*Цель лабораторной работы:* научиться пользоваться средствами

математического пакета MathCad для аналитических вычислений.

## 10.1. Символьные вычисления. Вычисление интегралов

### Вычисление неопределенных и определенных интегралов.

Для вычисления символьного неопределенного интеграла следует:

- вызвать оператор интегрирования, щелкнув мышкой на нужной кнопке на палитре Calculus;
- заполнить поля ввода для подинтегрального выражения и переменной интегрирования;
- заключить выражение в выделяющую рамку;
- нажать клавиши **Ctrl** + **.** (или на палитре Evaluation нажать клавишу  $\rightarrow$ ), и нажать клавишу **Enter**.

Результат появится ниже интеграла.

Получить интеграл от заданной функции можно и не используя оператор интеграла. Для этого нужно набрать функцию, выделить переменную, по которой выполняется интегрирование (поставить рядом с ней маркер ввода) и выполнить пункт меню **Symbolics/Variable/Integrate**. Результат появится под функцией.

Для вычисления символьного определенного интеграла следует:

- вызвать оператор интегрирования, щелкнув мышкой на нужной кнопке на палитре Calculus;
- заполнить поля ввода для подинтегрального выражения, переменной интегрирования и пределов;
- заключить выражение в выделяющую рамку;
- нажать клавиши **Ctrl** + **.** (или на палитре Evaluation нажать клавишу  $\rightarrow$ ), и нажать клавишу **Enter**.

Результат появится ниже интеграла.

### Индивидуальные задания

Найдите интегралы указанных функций своего варианта.

1.  $f(x) = x^3 - 9x^2 + 24x - 15$ .

2.  $f(x) = -x^3 - 12x^2 - 45x + 51$ .

3.  $f(x) = x^3 - 3x + 2$ .

4.  $f(x) = -x^3 + 9x^2 - 24x + 21$ .

5.  $f(x) = x^3 + 3x^2 - 2$ .

6.  $f(x) = -x^3 - 3x^2 - 1$ .

7.  $f(x) = x^3 - 9x^2 + 24x - 12$ .

8.  $f(x) = -x^3 + 9x^2 - 24x + 15$ .

9.  $f(x) = x^3 - 12x^2 + 45x - 45$ .

10.  $f(x) = -x^3 + 3x - 7$ .

11.  $f(x) = x^3 + 6x^2 + 9x + 3$ .
12.  $f(x) = -x^3 - 9x^2 - 24x - 18$ .
13.  $f(x) = x^3 - 3x^2 + 9$ .
14.  $f(x) = -x^3 - 6x^2 - 9x - 6$ .
15.  $f(x) = x^3 - 6x^2 + 9x + 2$ .

## **ЛАБОРАТОРНАЯ РАБОТА №11 СОЗДАНИЕ И ЗАПОЛНЕНИЕ ОДНОТАБЛИЧНОЙ БАЗЫ ДАННЫХ В MS ACCESS.**

*Цель лабораторной работы:* изучить принципы построения баз данных, освоить правила создания и редактирования таблиц в СУБД ACCESS.

### **11.1. Основные сведения о технологии баз данных**

С самого начала развития вычислительной техники образовалось два основных направления ее использования:

- вычислительное применение вычислительной техники для выполнения численных расчетов
- информационное использование средств вычислительной техники в автоматизированных информационных системах.

Информационные системы ориентированы на хранение, выбор и модификацию постоянно существующих данных.

Примеры информационных систем:

- банковские системы,
- системы резервирования авиационных или железнодорожных билетов,
- системы резервирования мест в гостиницах и т.д.

Основные операции информационных систем (например, упорядочивание информации по различным признакам, быстрое извлечение выборки с произвольным сочетанием признаков) возможно выполнить, только если данные структурированы.

*Структуризация* – это представление данных в соответствии с определенными соглашениями.

Проблемы использования неструктурированных данных легко пояснить следующим примером.

Данные, содержащие сведения о людях (номер личного дела, фамилию, имя, отчество и год рождения), записанные произвольно в текстовом файле, являются неструктурированными. Пусть есть следующие записи: Личное дело N 16493, Сергеев Петр Михайлович, дата рождения 1 января 1976 г; Л/д. N 16593. Петрова Анна Владимировна,



дата рожд. 15 марта 1975 г; N личн. дела 16693, д.р. 14.04.78, Анохин Андрей Борисович.

Легко убедиться, что сложно организовать поиск необходимых данных, хранящихся в таком виде, а упорядочить подобную информацию практически почти невозможно.

Чтобы автоматизировать поиск и систематизировать эти данные, необходимо выработать определенные соглашения о способах представления данных, например такие соглашения: дату рождения нужно записывать одинаково для каждого человека, она должна иметь одинаковую длину и определенное место среди остальной информации. Эти же замечания справедливы и для остальных данных (номер личного дела, фамилия, имя, отчество). Структурированные данные имеют следующий вид

<b>N личного дела</b>	<b>Фамилия</b>	<b>Имя</b>	<b>Отчество</b>	<b>Дата рождения</b>
16493	Сергеев	Петр	Михайлович	01.01.76
16393	Петрова	Анна	Владимировна	15.03.75
16693	Анохин	Андрей	Борисович	14.04.76

При создании новой информационной технологии баз данных были сформулированы основные стандартные требования к организации данных:

- концентрация данных в одном месте и создание постоянно обновляемой модели предметной области (ПО) (предметная область – это часть реального мира, моделируемая какими-либо средствами);.
- максимально возможная независимость прикладных программ от данных.

Единый блок данных для каждой ПО называется базой данных (БД). Единую управляющую программу для манипулирования данными каждой ПО на физическом уровне называют системой управления базами данных – СУБД.

## **11.2. Модели данных**

Любая БД строится в соответствии с некоторым набором данных, в рамках которого представляется информация о реальном мире. Основные понятия и способы организации таких наборов данных, используемые при моделировании объектов реального мира и их взаимосвязей, можно назвать моделью данных.

В процессе развития технологии баз данных в качестве основных рассматривались три модели данных:

- сетевая,
- иерархическая
- реляционная.

В настоящее время наиболее распространенной является реляционная модель, которая поддерживается подавляющим большинством СУБД.

Предварительно рассмотрим несколько важных для реляционной модели данных понятий:

**тип данных** – полностью эквивалентно понятию типа данных в языках программирования. Под типом данных понимается множество допустимых значений данных, относящихся к этому типу, и набор операций, разрешенных над данными этого типа. Обычно в современных реляционных БД используются данные символьных, числовых и специализированных типов данных (денежные типы, типы даты, времени и др.).

**домен** – соответствует понятию множества в математике. Наиболее правильной интуитивной трактовкой понятия домена является понимание домена как допустимого множества значений данного типа.

**кортеж** – это последовательность, составленная из элементов доменов. Кортеж содержит по одному элементу из каждого домена, причем порядок расположения этих элементов строго задан.

**отношение** можно определить как множество, элементами которого являются кортежи.

В приведенном рисунке домены **Номер**, **Количество** и **Добавка к окладу** – множества целых чисел, домен **Должность** – множество строк и домен **Оклад** – множество вещественных чисел. Множество кортежей и является отношением, а вся полученная таблица является графическим представлением нового множества. Чаще всего эта таблица отождествляется с отношением, т.е. о ней говорят, что это и есть отношение.

Имена доменов, на базе которых получают отношение, называются его атрибутами.

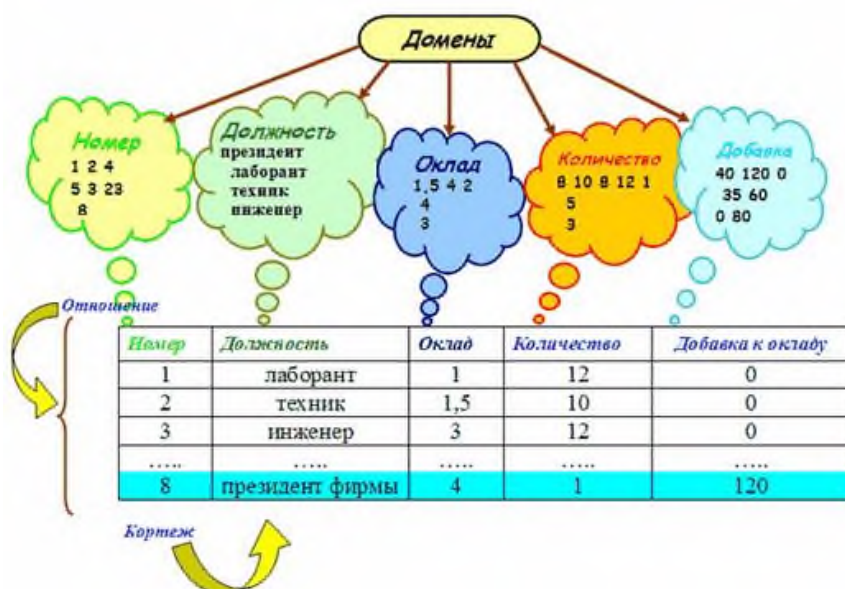


Рис. 10.1. Графическое представление отношения

Таким образом, теперь можно дать следующее определение реляционной модели данных: это совокупность основных понятий и способов организации данных, используемых для моделирования предметной области, которая основана на отношениях.

**Реляционная база данных** – это совокупность взаимосвязанных отношений, содержащих всю информацию о предметной области. Каждое отношение отображается таблицей и в компьютере хранится в виде файла записей.

Реляционная БД – это не просто набор таблиц (отношений). Каждая таблица отображает отношение, полученное по определенным правилам из других отношений. Существует строгая система операций (реляционная алгебра), которая позволяет выводить одни отношения из других подобно тому, как выполняются арифметические операции. Применение такой системы дает возможность устранить избыточность информации, а также делить на хранимую и не хранимую (вычисляемую) части. И при необходимости вычислять нужную информацию из хранимой части, что экономит память.

### 11.3. Пример разработки базы данных

Пусть требуется создать базу данных для хранения данных о студентах, преподавателях и изучаемых дисциплинах, используя которую можно, например, получить сведения о студентах, имеющих право получать стипендию. Рассмотрим условную предметную область «Учебный процесс», включающую следующие основные понятия: «студент», «преподаватель», «изучаемая дисциплина», «оценки».

На этапе логического проектирования необходимо выделить основные объекты предметной области, которые требуется моделировать,

дать им имена и описать их атрибуты. Приведем эти объекты в виде списка имен, за которыми в скобках следует список атрибутов.

Можно выделить четыре объекта:

- **Студенты** (*код студента, фамилия, имя, отчество, номер группы, дата рождения, стипендия, оценки*).
- **Дисциплины** (*код дисциплины, название дисциплины*),
- **Оценки** (*код студента, код дисциплины, оценка*),
- **Преподаватели** (*код преподавателя, код дисциплины, фамилия, имя, отчество, дата рождения, телефон, название дисциплины*).

В реляционной базе данных в качестве объектов рассматриваются отношения, которые можно представить в виде таблиц. Таблицы между собой связываются посредством общих полей, одинаковых по форматам и, как правило, по названию, имеющих в обеих таблицах.


Общие поля в таблицах для обеспечения связности данных обозначены в списках атрибутов одинаковым цветом. В таблицах **Студенты** и **Оценки** таким полем будет «Код студента», в таблицах **Дисциплины** и **Оценки** – «Код дисциплины», в таблицах **Преподаватели** и **Дисциплины** – «Код дисциплины». Выбор цифровых кодов вместо фамилий или названий дисциплин обусловлен меньшим размером данных в таких полях: например, число «2» по объему занимаемой памяти значительно меньше слова «математика».

На рисунке представлена схема базы данных, где жирным шрифтом выделены **ключевые поля**.



**Ключом** называют любую функцию от атрибутов отношения, с помощью которой можно однозначно определить конкретный кортеж. Такая функция может быть значением одного из атрибутов (простой ключ), или задаваться алгебраическим выражением, включающим значения нескольких атрибутов (составной ключ). Это означает, что данные в строках каждого из столбцов составного ключа могут повторяться, но комбинация данных каждой строки этих столбцов является **уникальной**. Например, в таблице **Студенты** есть столбцы *Фамилии* и *Год рождения*. В каждом из столбцов есть некоторые повторяющиеся данные, т.е. одинаковые фамилии и одинаковые года рождения. Но если студенты, имеющие одинаковые фамилии, имеют разные года рождения, то эти столбцы можно использовать в качестве

составного ключа.

Как правило, ключ является уникальным, т.е. каждый кортеж определяется значением ключа однозначно, но иногда используют и неуникальные ключи (ключи с повторениями). Обозначение  на схеме соответствует типу связи между таблицами «один-ко-многим». При таком типе связи одной строке таблицы, например, **Студенты** с уникальным значением ключа «Код студента» может соответствовать множество строк таблицы **Оценки** с таким же значением поля «Код студента».

В приведенной схеме каждое отношение представляет собой таблицу, каждая строка которой состоит из полей, значения которых являются значениями атрибутов отношения.

Например, таблица **Студенты** может иметь следующий вид:

Код студента	Фамилия	Имя	Отчество	Номер группы	Дата рождения	Стипендия
16493	Сергеев	Петр	Михайлович	9a51	01.01.87	750
16393	Петрова	Анна	Владимировна	9a52	15.03.86	750
16693	Анохин	Андрей	Борисович	9a52	14.04.87	800
-----	-----	-----	-----	-----	-----	-----

## 11.4. Средства и объекты СУБД

Основные средства СУБД включают:

- средства описания структуры БД;
- средства конструирования экранных форм для ввода данных;
- средства создания запросов для выборки данных при заданных условиях и выполнения операций по их обработке;
- средства создания отчетов;
- языковые средства (макросы, встроенный алгоритмический язык, язык запросов) для реализации нестандартных алгоритмов обработки данных.

Основными объектами любой СУБД являются таблицы, запросы, формы, отчеты, макросы и модули.

**Таблицы** служат для хранения всех данных, имеющихся в БД, и ее структуры (полей, их типов и свойств).

**Запросы** используются для извлечения данных из таблиц и предоставления их пользователю в удобном виде. С помощью запросов данные обрабатывают (упорядочивают, фильтруют, отбирают, изменяют, объединяют, выполняют простейшие вычисления в таблицах). Запросы обеспечивают сохранность данных в таблицах БД и разграничение доступа к различным данным для разных категорий пользователей. Например, для базы данных «Учебный процесс» можно создать запрос для

получения списка отличников по результатам сессии.

**Формы** – это средство для ввода данных. Они используются для заполнения тех полей таблицы, к которым есть доступ пользователям данной категории. В форме можно разместить специальные элементы управления для автоматизации ввода (раскрывающиеся списки, переключатели, флажки и т.п.). Формы особенно удобны для ввода данных с заполненных бланков.

**Отчеты** предназначены для вывода данных на принтер в удобном и наглядном виде. В отчетах данные таблиц и запросов преобразуются в документы.

Макросы и модули предназначены для автоматизации повторяющихся операций при работе с СУБД и создания новых функций путем программирования. Макросы состоят из последовательностей внутренних команд СУБД, модули создаются средствами внешнего языка программирования.

### Выполнение индивидуального задания

Создать и отредактировать простую однотабличную базу данных. Далее приведено описание этапов выполнения и пояснения по их выполнению.

#### 1 этап

- Создайте новую базу данных.
- Создайте таблицу базы данных.
- Определите поля таблицы в соответствии с табл. 1.

Таблица 1. **Teachers**

Имя поля	Тип данных	Размер поля
Код преподавателя	Счетчик	
Фамилия	Текстовый	15
Имя	Текстовый	15
Отчество	Текстовый	15
Дата рождения	Дата/время	Краткий
Должность	Текстовый	9
Дисциплина	Текстовый	11
Телефон	Текстовый	9
Зарплата	Денежный	

- Сохраните созданную таблицу.

#### 2 этап

- Введите ограничения на данные, вводимые в поле Должность; допускается ввод только значений Профессор, Доцент или Ассистент.
- Задайте текст сообщения об ошибке, которое будет появляться на экране при вводе неправильных значений в поле Должность.
- Задайте значение по умолчанию для поля Должность: Доцент.
- Заполните таблицу данными в соответствии с табл. 2 и проверьте реакцию системы на ввод неправильных значений в поле Должность.

Таблица 2.

**Внимание:** вместо Игнатьевой Татьяны Павловны напишите свои

Фамилию, Имя и Отчество!!!

Код преподавателя	Фамилия	Имя	Отчество	Дата рождения	Должность	Дисциплина	Телефон	Зарплата
1	Истомин	Ремир	Евгеньевич	23.10.54	Доцент	Информатика	110-44-68	890р.
2	Миронов	Павел	Юрьевич	25.07.40	Профессор	Экономика	312-21-40	1200р.
3	Гришин	Евгений	Сергеевич	05.12.67	Доцент	Математика	260-23-65	760р.
4	Сергеева	Ольга	Ивановна	12.02.72	Ассистент	Математика	234-85-69	450р.
5	Емец	Татьяна	Ивановна	16.02.51	Доцент	Экономика	166-75-33	890р.
6	Игнатьева	Татьяна	Павловна	30.05.66	Доцент	Информатика	210-36-98	790р.
7	Миронов	Алексей	Николаевич	30.07.48	Доцент	Физика	166-75-33	890р.

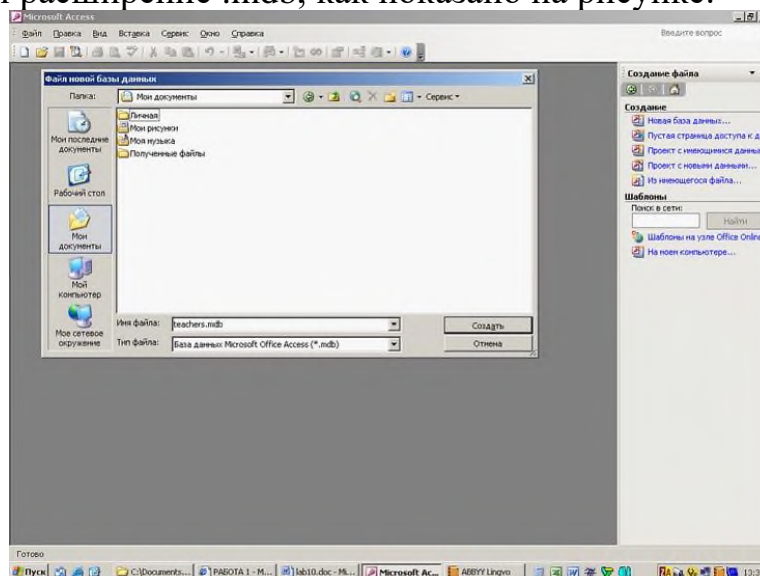
- Измените ширину каждого поля таблицы в соответствии с шириной данных.
- Произведите поиск в таблице преподавателя Миронова.
- Произведите замену данных: измените заработную плату ассистенту Сергеевой с 450 р. на 470 р.
- Произведите сортировку данных в поле Дата рождения по убыванию.
- Произведите фильтрацию данных по полям Должность (Доцент) и Дисциплина (Информатика).

### **Пояснения**

#### **1. Создание новой базы данных и таблицы базы**

##### **Создание новой базы**

- запустите приложение MS ACCESS;
- выполните команду меню Файл/ Создать / Новая база данных;
- выберите размещение базы данных, указав папку, в которой она будет находиться;
- задайте имя базы данных (имя файла) teachers, сохранив предлагаемое приложением расширение .mdb, как показано на рисунке:



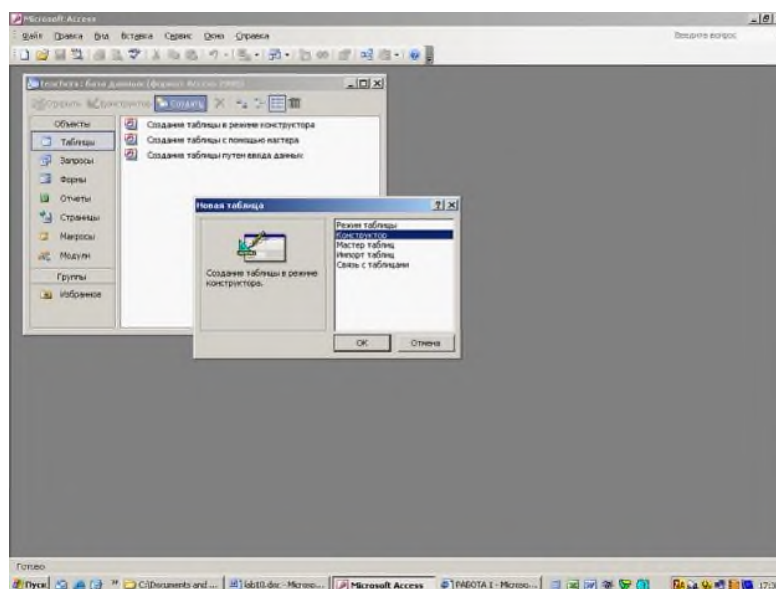
- Нажмите кнопку Создать.

##### **Создание таблицы**

- в окне teachers: база данных выберите вкладку Таблицы и нажмите кнопку Создать;



- в окне Новая таблица выберите пункт Конструктор и нажмите кнопку ОК:

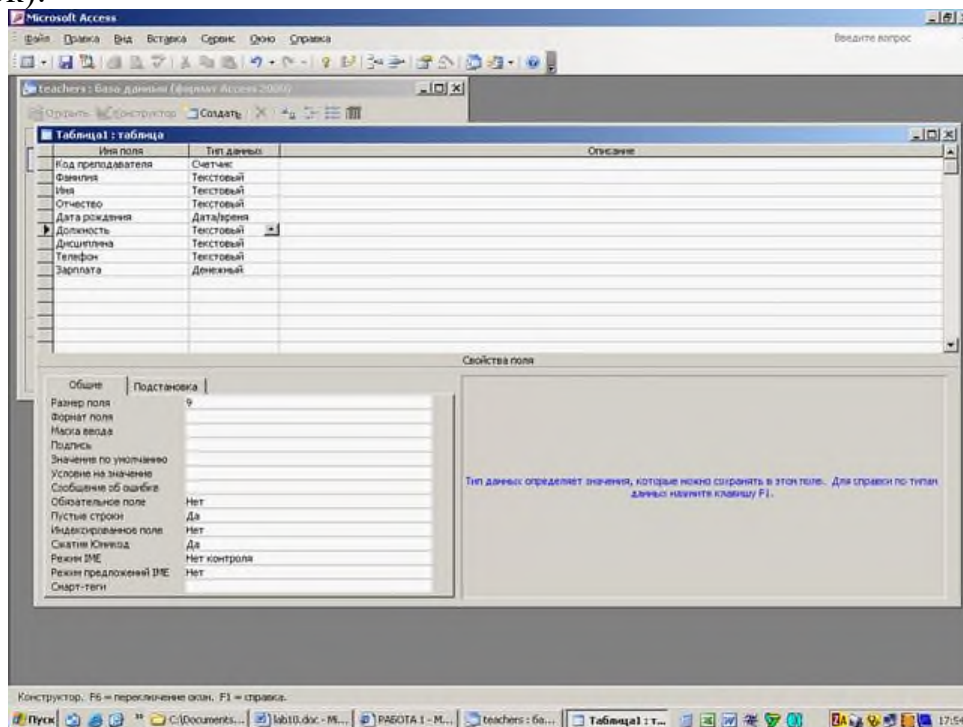


- в открывшемся окне Таблица 1: таблица определите поля таблицы.

## 2. Определение полей и сохранение созданной таблицы

### Определение полей таблицы

- в соответствии с данными табл.1 заполните поля таблицы (см. рисунок):



- на вкладке Общие определите Размер поля.

### Сохранение таблицы

- выполните команду **Файл/ Сохранить**;
- сохраните таблицу под именем **teachers**;

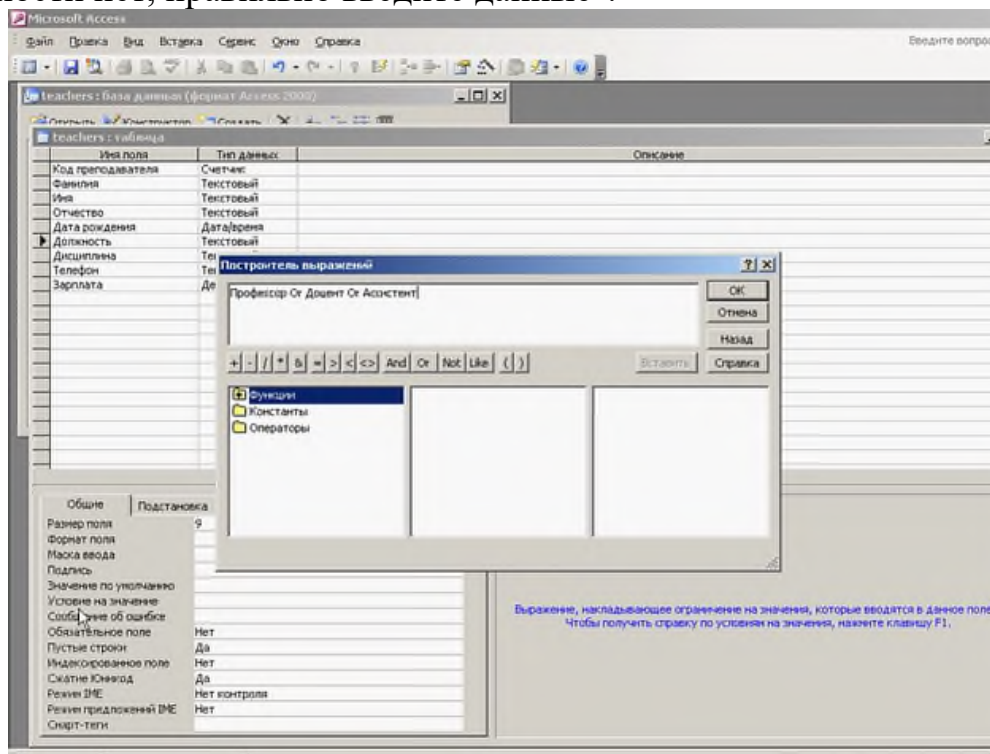


- для однотабличной базы данных ключевое поле задавать не следует.

### 3. Введение ограничений на данные, вводимые в поле Должность

#### Введение ограничений

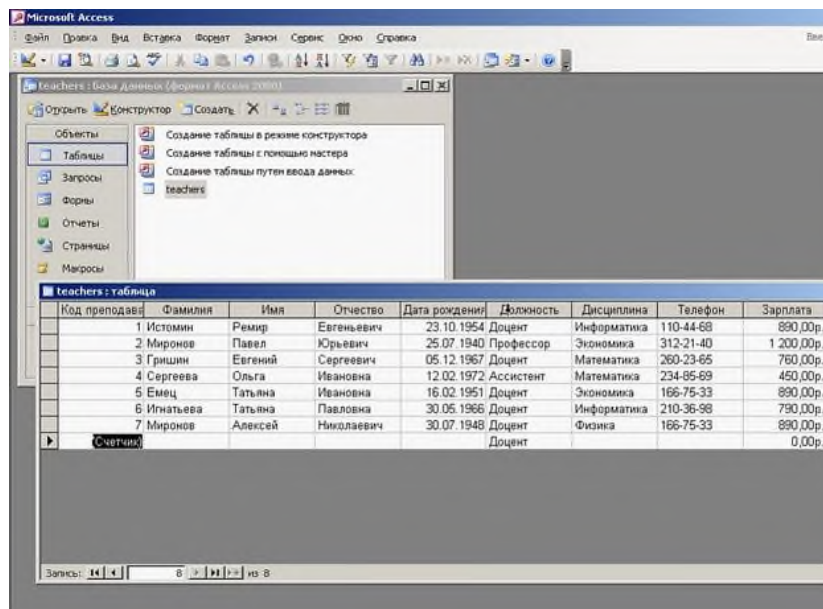
- войдите в режим Конструктор для проектируемой таблицы: в окне teachers: база данных выберите вкладку Таблицы и нажмите кнопку Конструктор;
- в верхней части окна выберите поле Должность;
- в нижней части о
- кна выберите строку параметра Условие на значение;
- нажмите кнопку для определения условий на значение при помощи построителя выражений;
- в окне Построитель выражений введите значение Профессор, затем нажмите кнопку Or (эта кнопка выполняет функцию ИЛИ), введите значение Доцент, снова нажмите Or, введите Ассистент и нажмите кнопку ОК; таким образом задается условие, при котором в поле Должность могут вводиться только указанные значения;
- В строке Сообщение об ошибке (см. рисунок) введите значение "Такой должности нет, правильно введите данные".



- В строке Значение по умолчанию введите значение Доцент. Перейдите в Режим таблицы, выполнив команду Вид / Режим таблицы. На вопрос о сохранении таблицы ответьте Да.

#### Заполнение таблицы

- Заполните таблицу данными, как показано на рисунке ниже.



- Сохраните созданную таблицу.

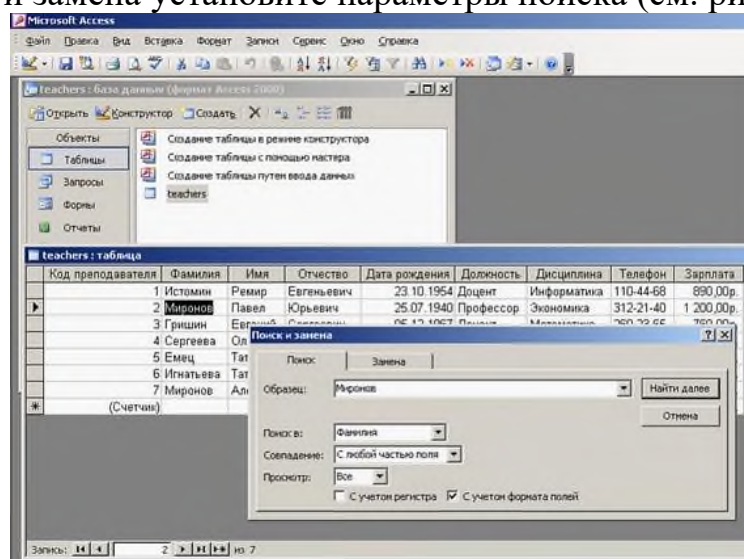
#### 4. Изменение ширины полей таблицы, организация поиска, изменение значений полей

##### *Изменение ширины полей*

- поместите курсор в любой строке поля Код преподавателя;
- выполните команду Формат / Ширина столбца;
- в окне Ширина столбца нажмите кнопку По ширине данных;
- проделайте такие же операции со всеми полями таблицы.

##### *Поиск в таблице (поиск преподавателя Миронова)*

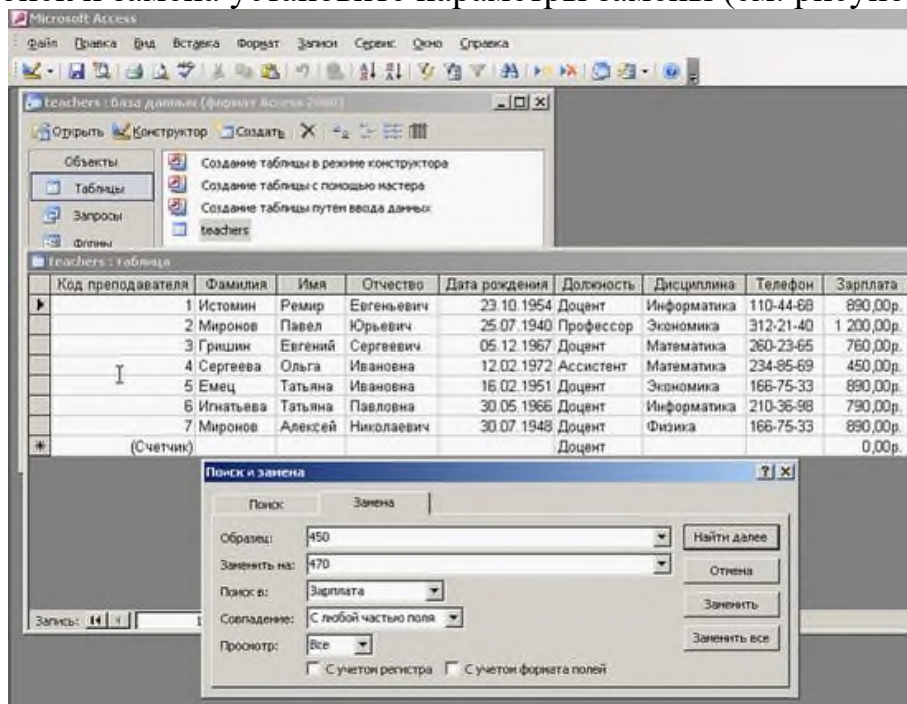
- поместите курсор в первую строку поля Фамилия;
- выполните команду Правка / Найти;
- в окне Поиск и замена установите параметры поиска (см. рисунок)



##### *Изменение значений полей*

Для изменения заработной платы ассистенту Сергеевой:

- поместите курсор в первую строку поля Зарплата;
- выполните команду Правка / Заменить;
- в окне Поиск и замена установите параметры замены (см. рисунок)



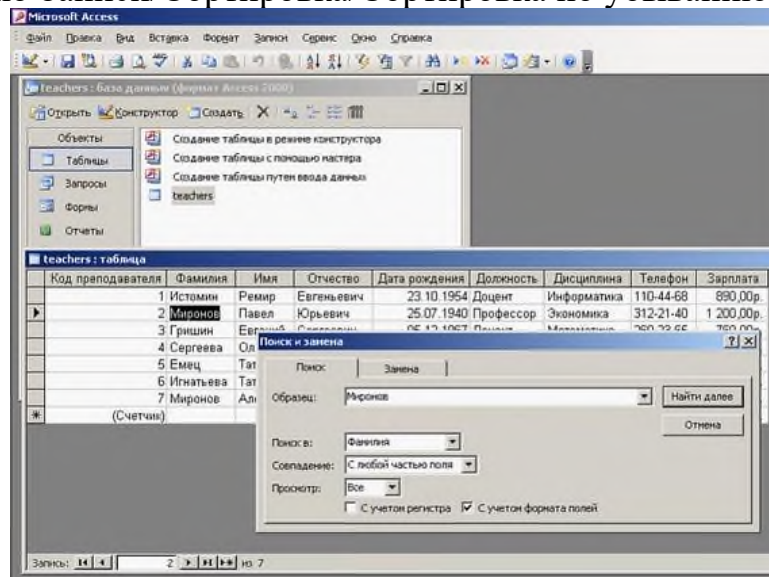
- нажмите кнопку **Заменить**;
- закройте окно **Поиск и замена** для выхода из режима замены.

## 5. Сортировка и фильтрация данных

### Сортировка данных

Сортировка данных в поле Дата рождения по убыванию:

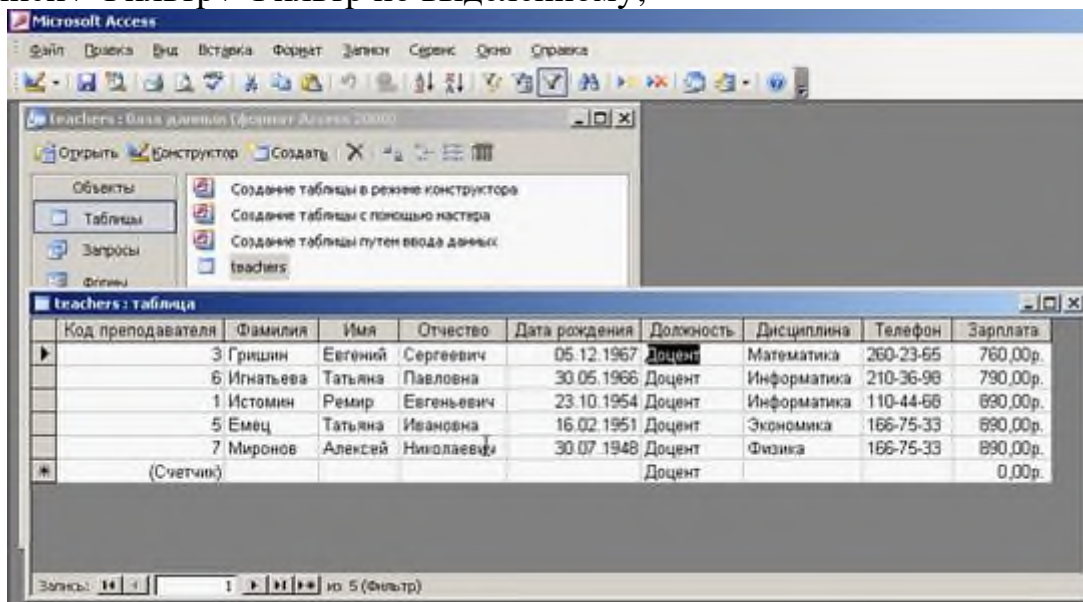
- поместите курсор в любом месте поля Дата рождения и выполните команду меню Записи/Сортировка/Сортировка по убыванию.



### Фильтрация данных

Фильтрация данных по полям **Должность (Доцент)** и **Дисциплина (Информатика)**:

- выделите значение **Доцент** поля **Должность** и выполните команду **Записи / Фильтр / Фильтр по выделенному**;



- для отмены фильтрации выполните команду **Записи / Удалить фильтр**;
- аналогичным образом выполняется фильтрация данных по полю **Дисциплина** (**Информатика**).

Выполните команду **Файл / Предварительный просмотр**.

## ЛАБОРАТОРНАЯ РАБОТА №12 ФОРМИРОВАНИЕ ЗАПРОСОВ НА ВЫБОРКУ

**Цель лабораторной работы:** научиться работать с объектами баз данных в СУБД ACCESS.

### Выполнение индивидуального задания

Далее приведено задание и основные пояснения по выполнению его этапов.

**1.**

- На основе таблицы **teachers** создайте простой запрос на выборку, в котором должны отображаться фамилии, имена, отчества преподавателей и их должность.
- Данные запроса отсортируйте по должностям.
- Сохраните запрос.

**2.**

- Создайте запрос на выборку с параметром;  
в запросе должны отображаться фамилии, имена, отчества преподавателей и преподаваемые ими дисциплины;  
в качестве параметра задайте **Фамилию преподавателя**;



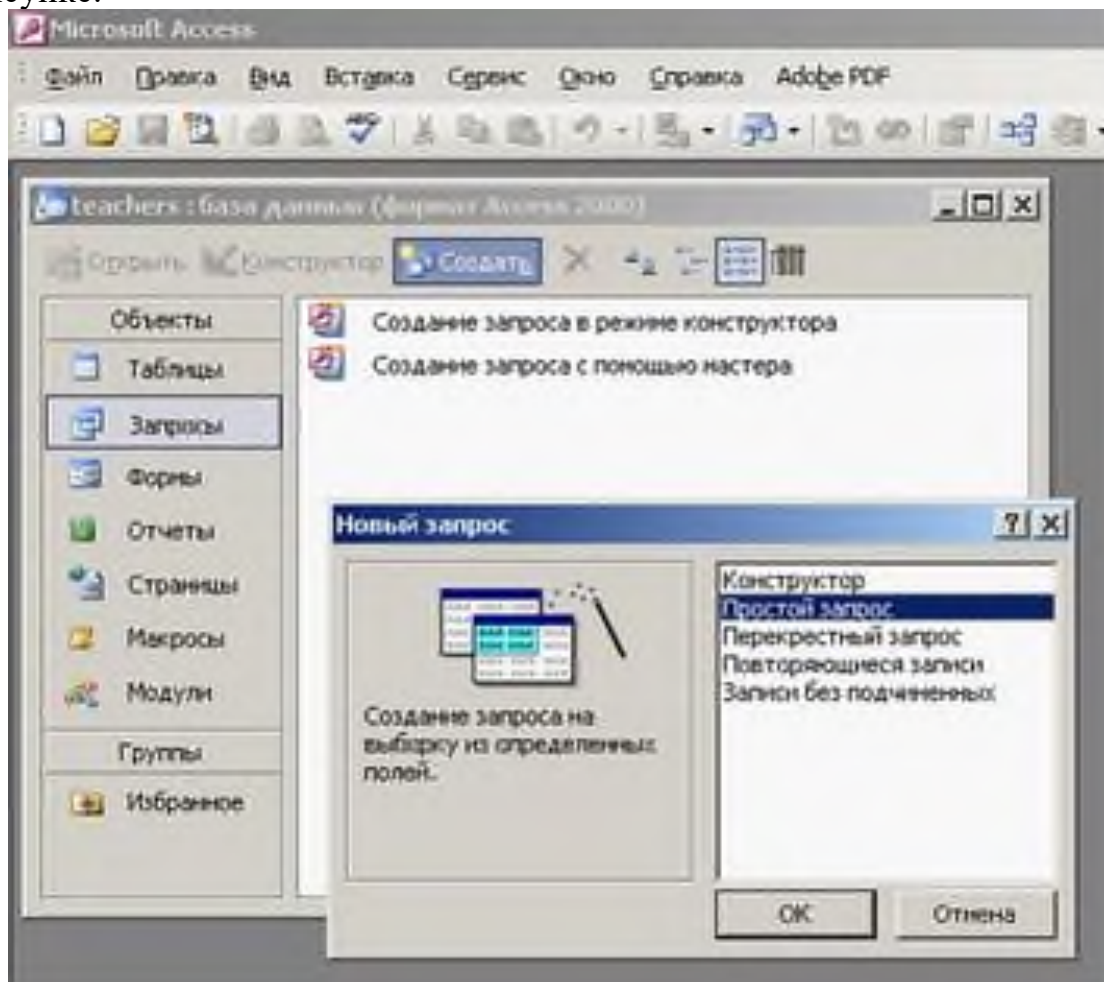
выполните этот запрос для преподавателя Гришина.

### ***Пояснения***

## **1. Создание простого запроса, определение полей данных для запроса**

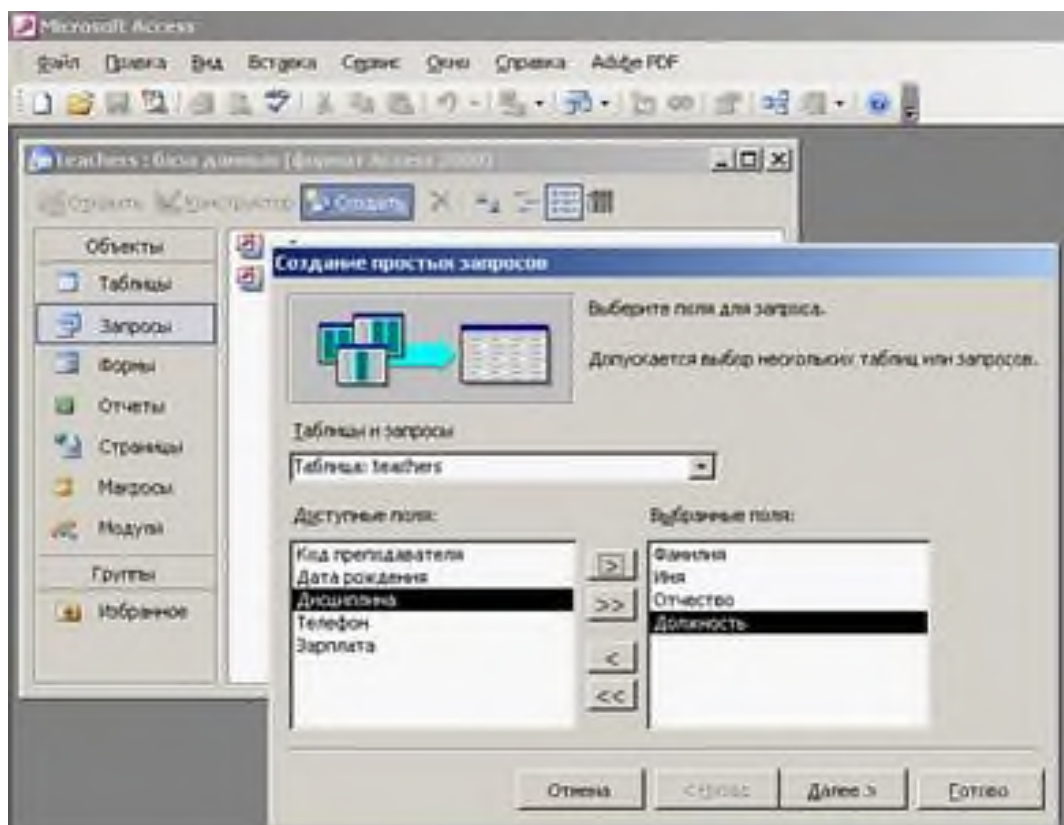
### ***Создание запроса***

- в окне базы данных откройте вкладку Запросы; нажмите кнопку Создать
- в окне Новый запрос выберите Простой запрос, как показано на рисунке:



### ***Выбор полей данных***

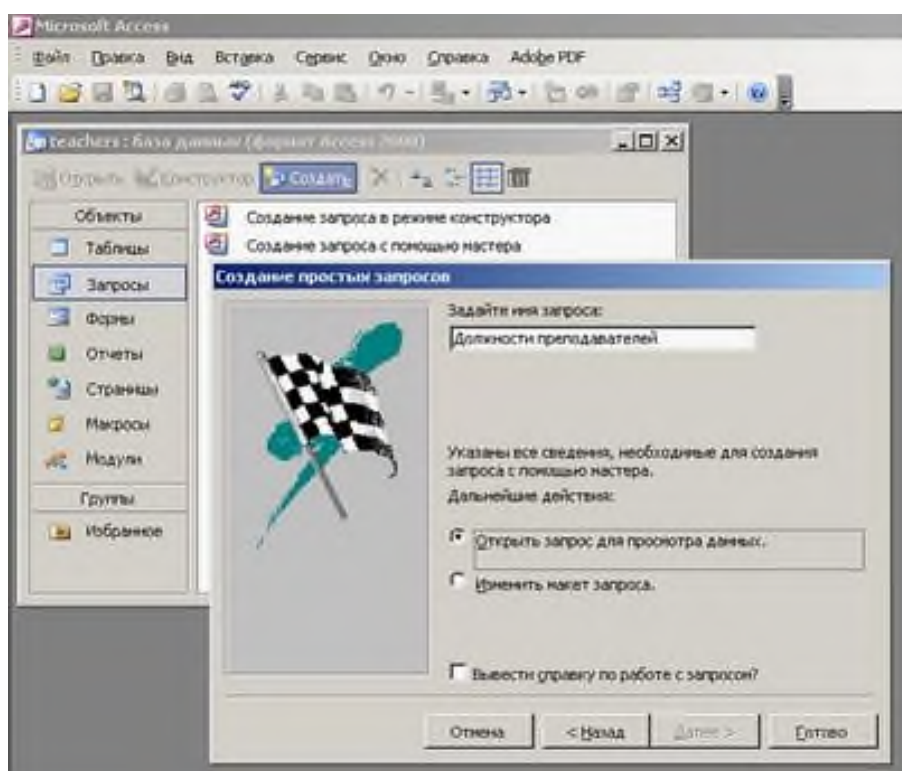
- в появившемся окне в поле Таблицы и запросы выберите таблицу **teachers**;
- в поле Доступные поля выберите последовательно поля: Фамилия, Имя, Отчество, Должность, как показано на рисунке:



**Задание имени и просмотр результатов запроса**

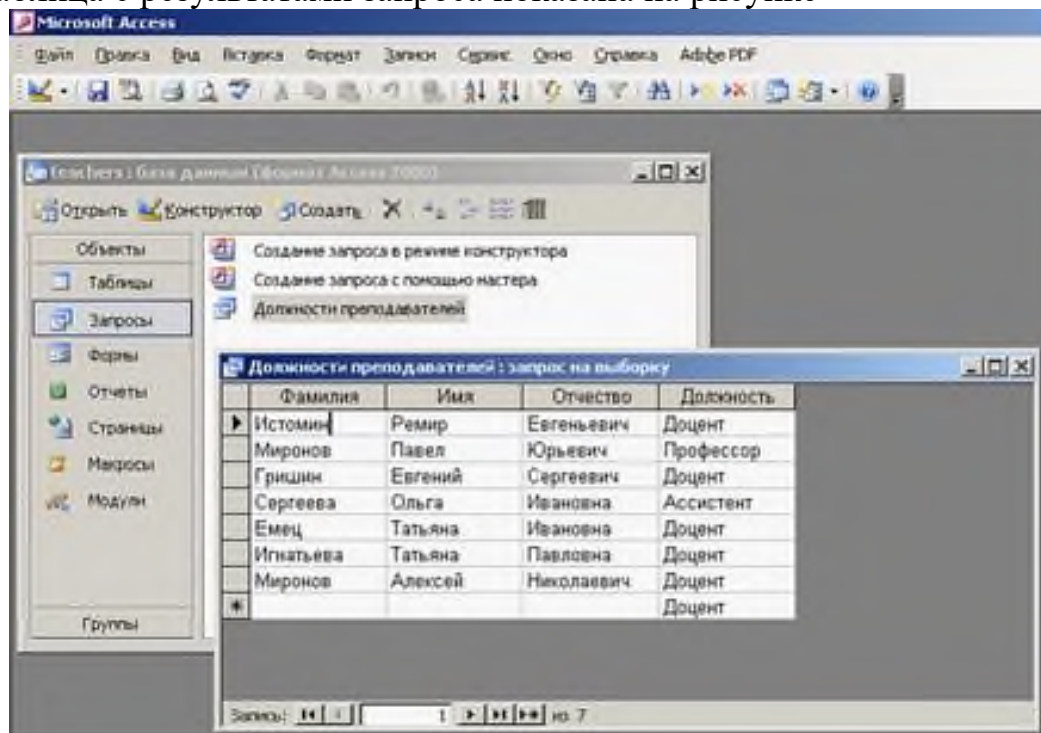
***Задание имени и сохранение запроса***

- в поле **Задать имя запроса** введите новое имя Должности преподавателей, как показано на рисунке ниже;
- нажмите **Готово**;



## Результаты запроса

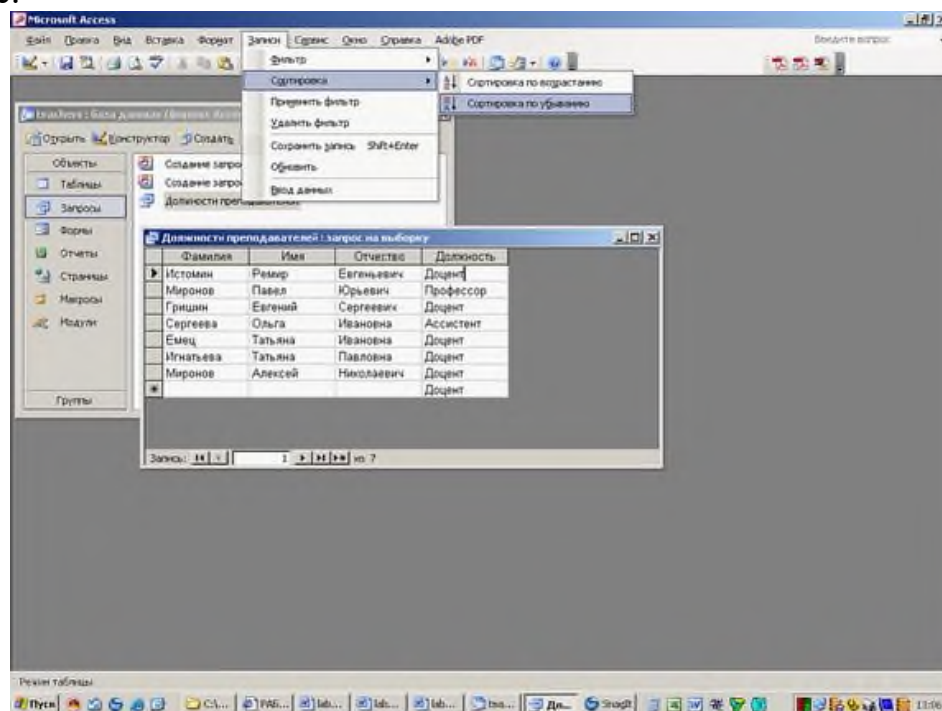
таблица с результатами запроса показана на рисунке



## Сортировка результатов и сохранение файла

### Сортировка результатов

- Поместите курсор в любую строку поля Должность; выполните команду Записи/Сортировка/Сортировка по убыванию, как показано на рисунке:



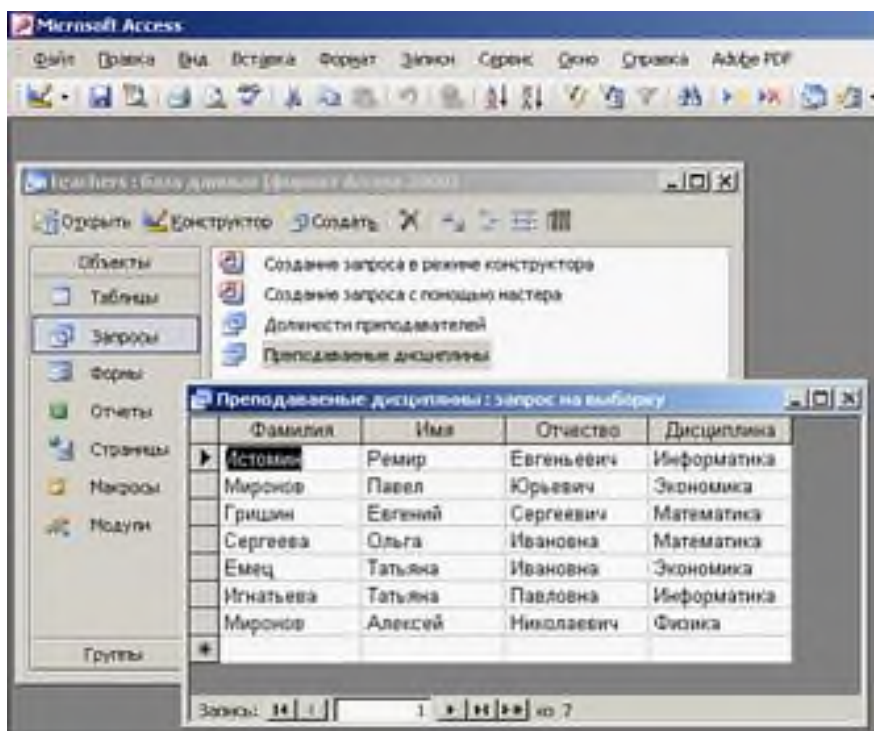
### *Сохранение файла*

- Выполните команду меню Файл/Сохранить.

## **2. Создание запроса на выборку с параметром**

### *Создание простого запроса*

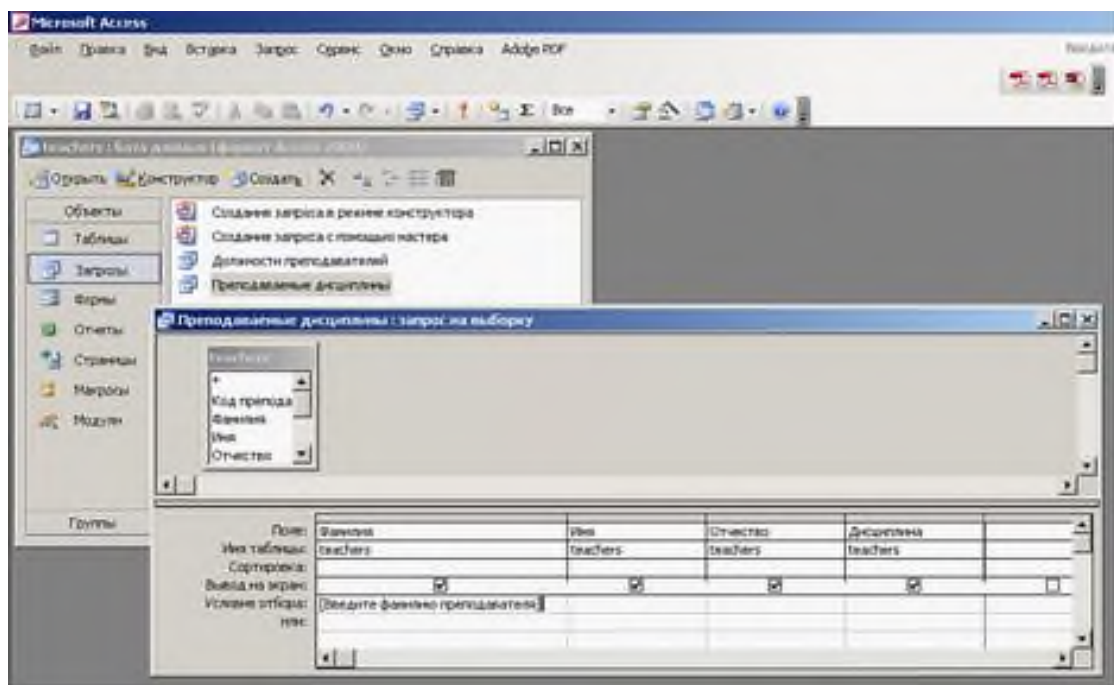
- создайте запрос на выборку для следующих полей таблицы **teachers**: **Фамилия, Имя, Отчество, Дисциплина**; создание запроса аналогично тому, как это делалось в разделе **Создание простого запроса**;
- задайте имя запроса: **Преподаваемые дисциплины**;



### *Задание условий отбора*

- перейдите в режиме конструктора, выполнив команду меню **Вид/Конструктор**;
- в строке параметра **Условия отбора** для поля **Фамилия** введите фразу: **[Введите фамилию преподавателя]**,

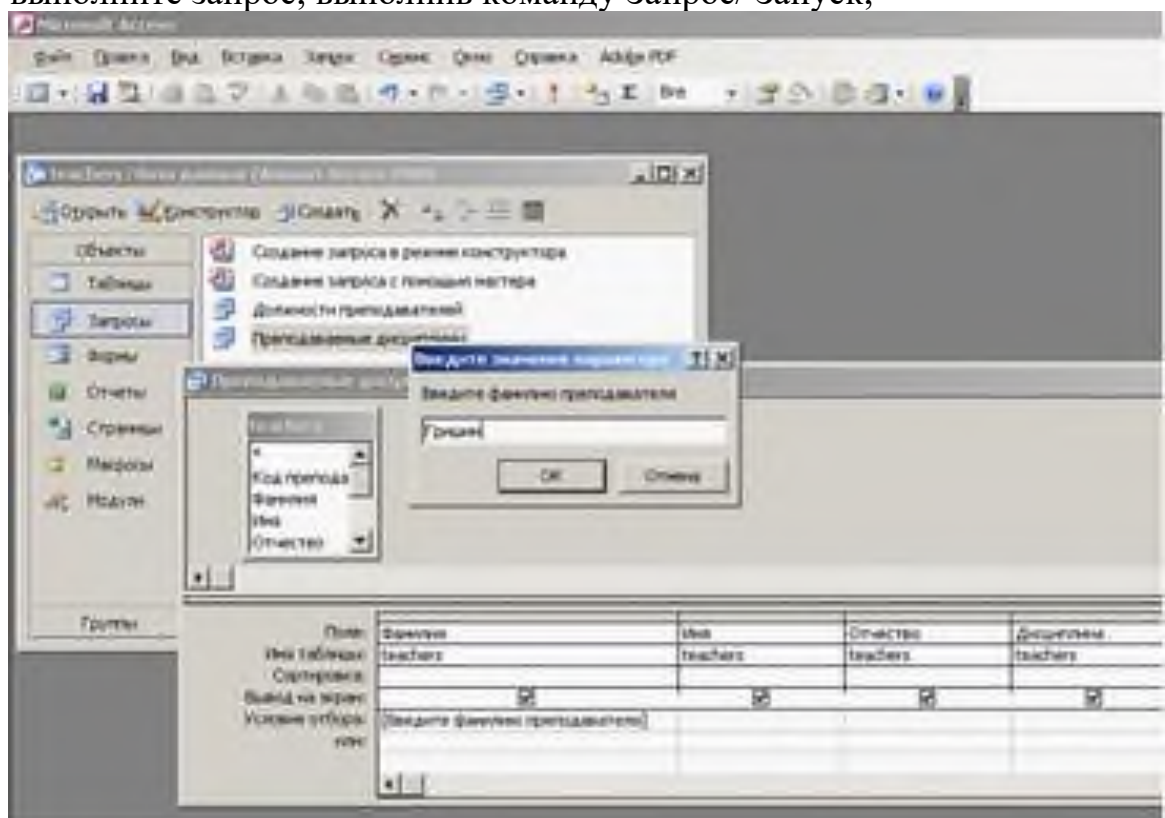




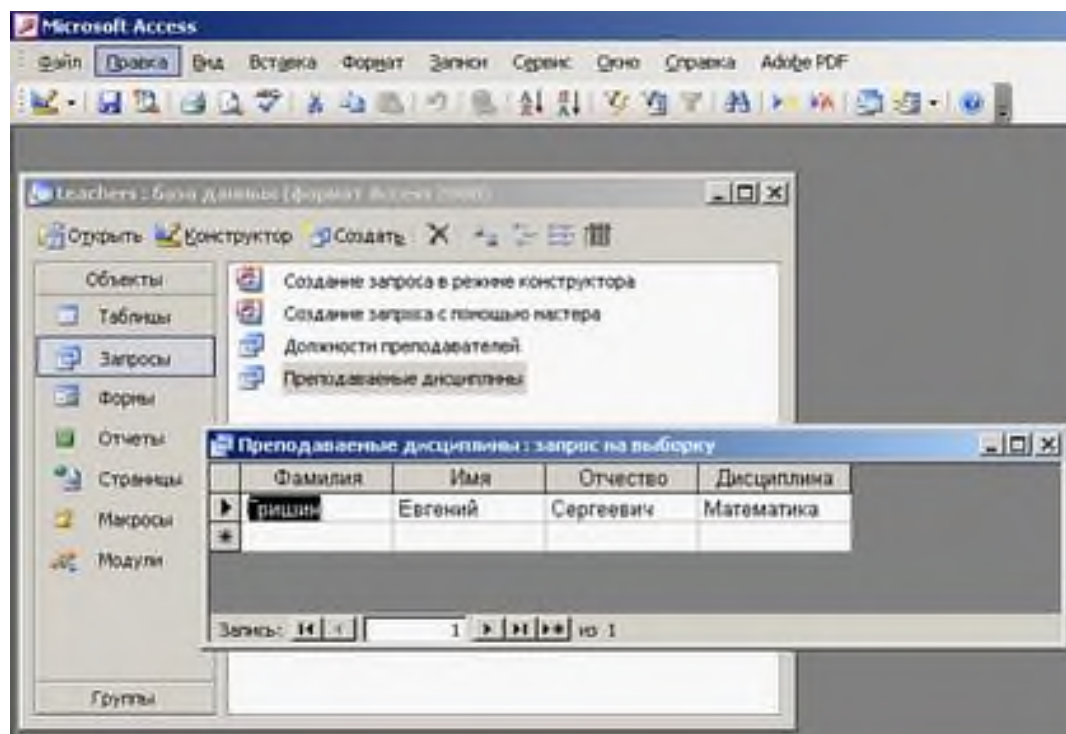
**Выполнение запроса, просмотр результатов и сохранение запроса**

### *Выполнение запроса*

- выполните запрос, выполнив команду Запрос/ Запуск;



## Результаты запроса



- Сохраните запрос. Закройте окно запроса.

## ЛАБОРАТОРНАЯ РАБОТА №13 СОЗДАНИЕ ОТЧЕТА С ГРУППИРОВКОЙ ДАННЫХ ПО ДОЛЖНОСТЯМ

**Цель лабораторной работы:** научиться работать с объектами баз данных в СУБД ACCESS.

### Выполнение индивидуального задания

Далее приведено задание и основные пояснения по выполнению его этапов.

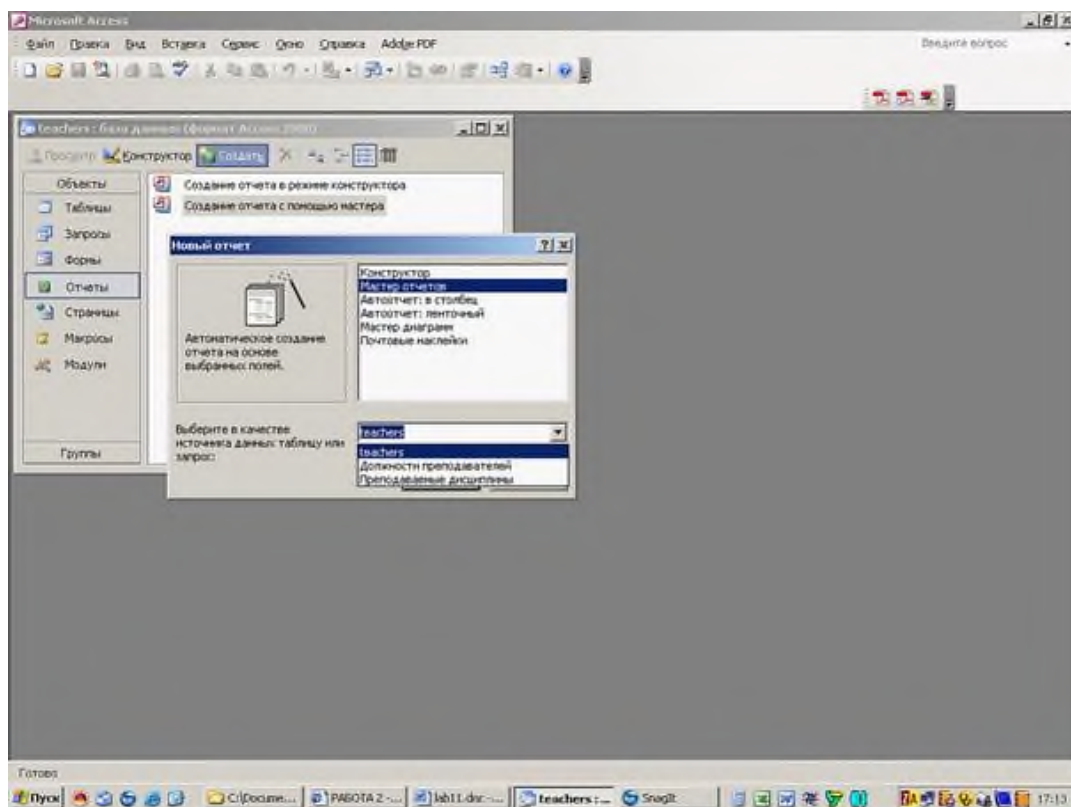
На основе таблицы **teachers** создайте отчет с группировкой данных по должностям.

#### Пояснения

#### 1. Создание отчета с группировкой данных по должностям

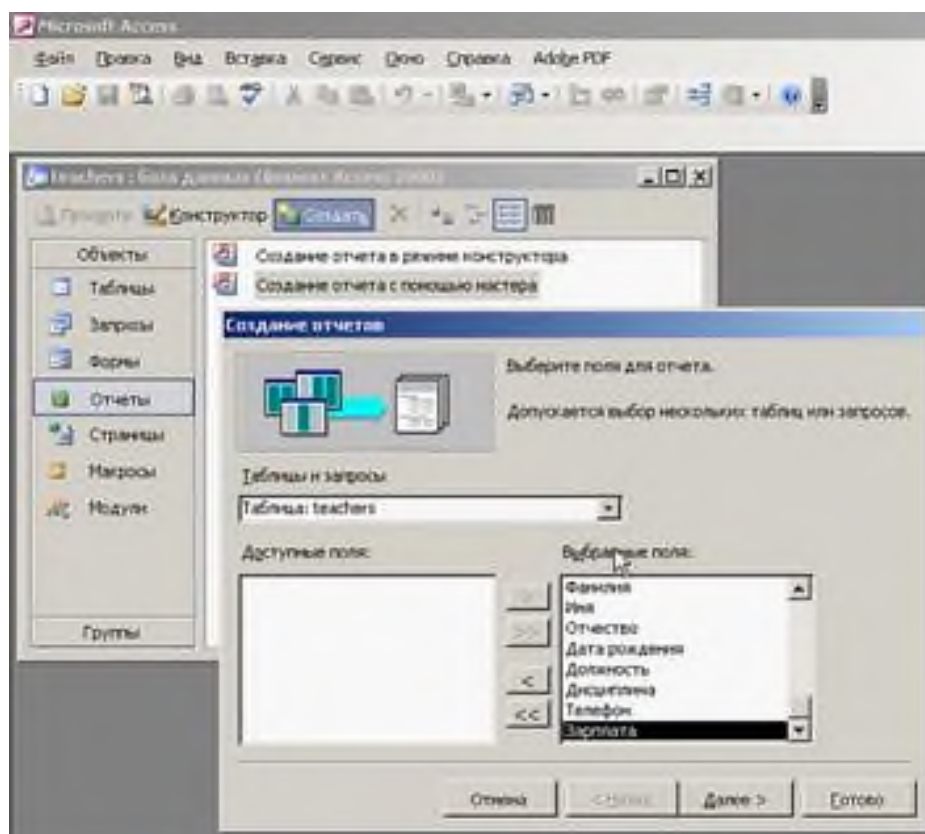
##### Создание отчета

- в окне базы данных **teachers** выполните команду Отчеты/ Создать; отчет создается с помощью Мастера отчетов
- выберите в качестве источника данных таблицу **teachers**, как показано на рисунке ниже.



### ***Выбор полей для отчета***

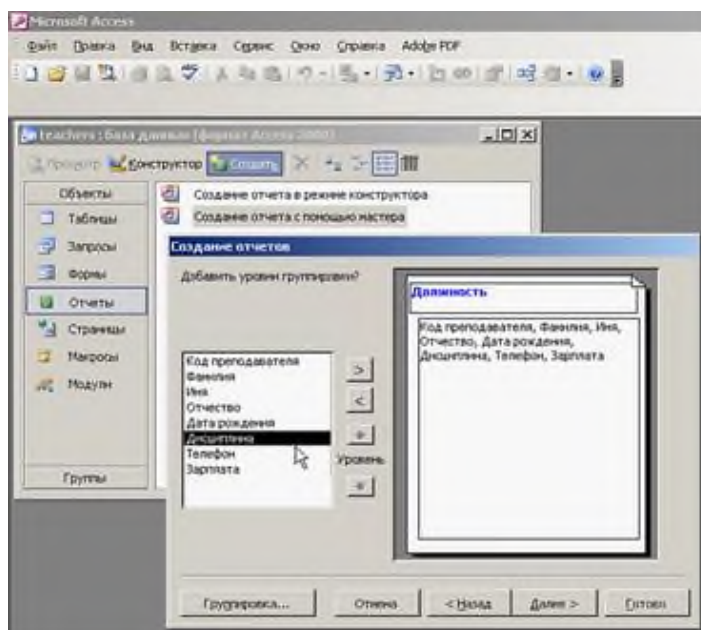
Выберите поля, которые будут присутствовать в отчете (выбираются все поля с помощью кнопки >>) и нажмите Далее.



## 2. Задание имени и просмотр результатов запроса

### *Задание группировки данных по полю*

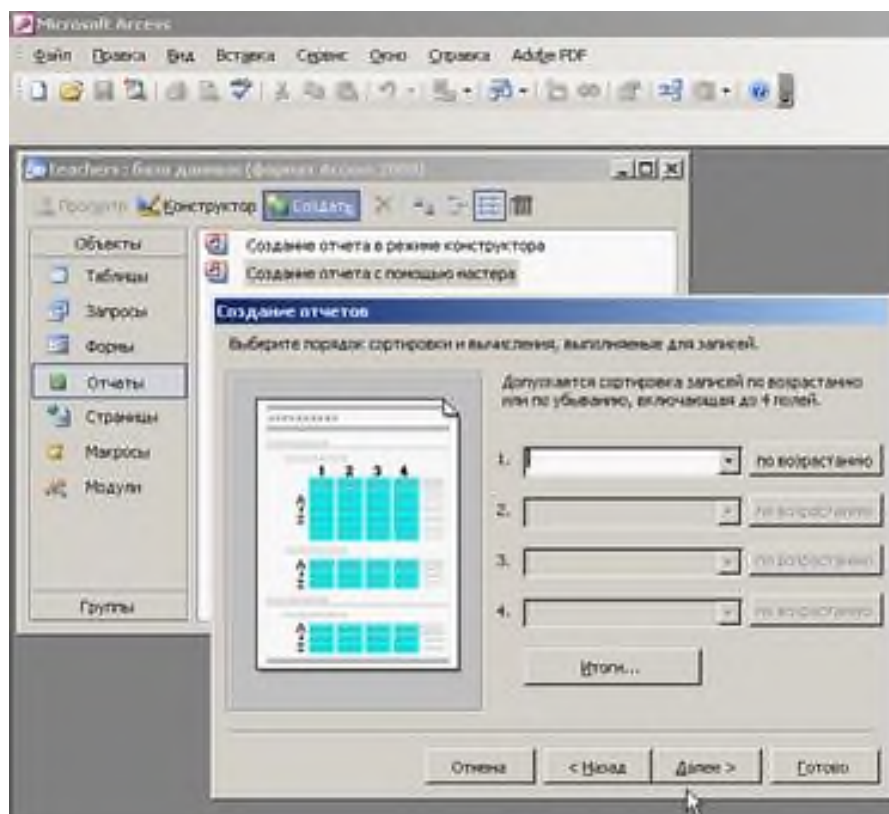
- В окне Создание отчетов присутствует перечень выбранных полей; выделите поле Должность двойным щелчком мыши - таким образом задается группировка данных по должности, результат показан на рисунке:



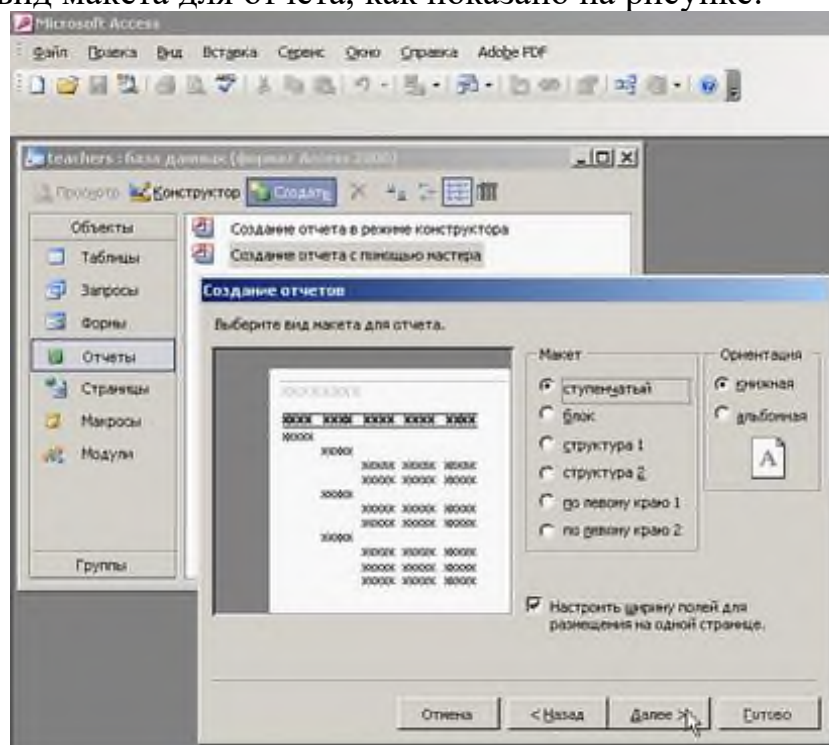
### *Пропуск сортировки и выбор макета отчета*

- Нажмите Далее. Порядок сортировки и вычисления, выполняемые для записей, задавать не будем.





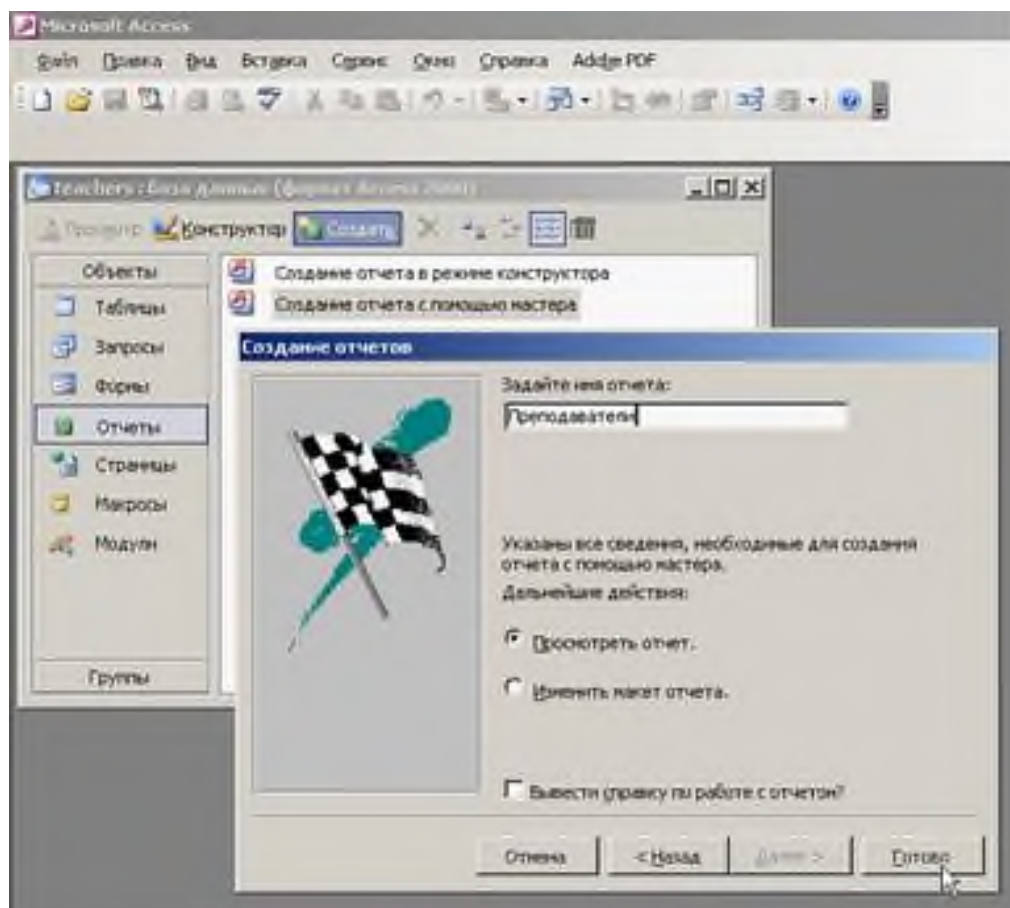
- Выберите вид макета для отчета, как показано на рисунке:



### 3. Задание стиля и просмотр отчета

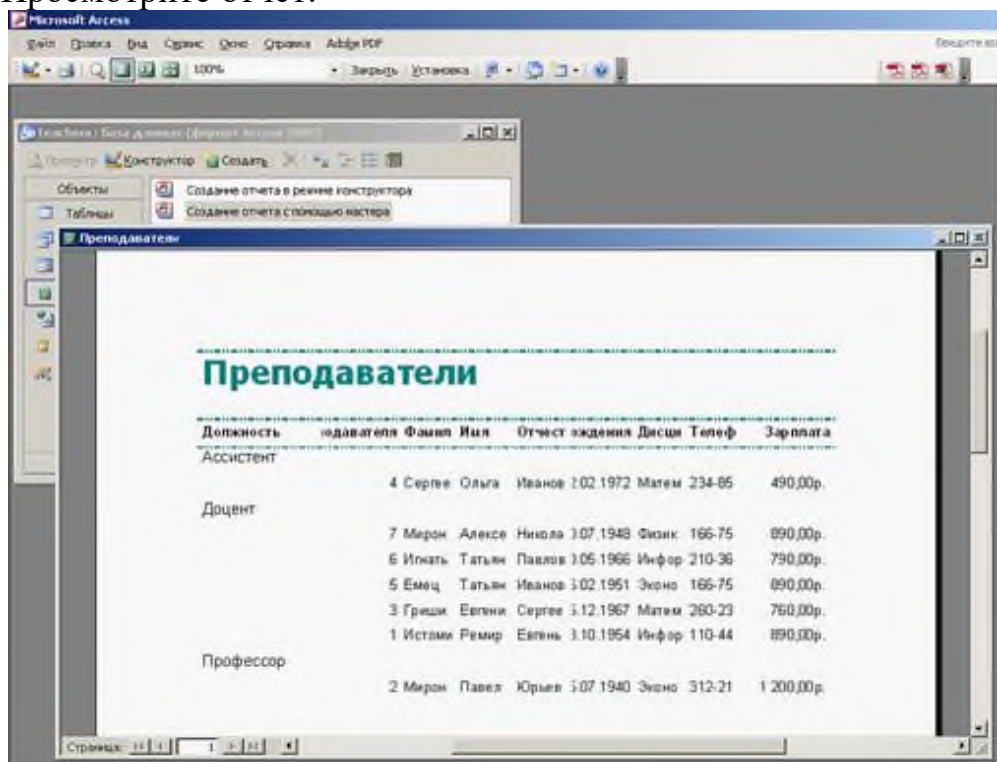
#### Задание стиля и наименование отчета

- Выберите стиль отчета **Обычный**.
- Задайте имя отчета **Преподаватели**



*Просмотр отчета*

Просмотрите отчет:



## ЛАБОРАТОРНАЯ РАБОТА №14 ОСНОВЫ РАЗРАБОТКИ И СОЗДАНИЯ WEB-СТРАНИЦ. ОФОРМЛЕНИЕ ТЕКСТА

*Цель лабораторной работы:* изучить основы языка гипертекстовой разметки HTML, научиться с помощью простейших средств создавать web-страницы, содержащие текст.

### 14.1. Основной инструментарий

Порядок прежде всего, поэтому перед началом работы мы создадим на нашем компьютере отдельную папку для будущей страницы. Создайте папку с номером своей группы в папке Мои документы (либо в папке, указанной преподавателем), а в ней создайте папку по своей фамилии. Например, если Вы учитесь в группе 12345, а Ваша фамилия - Иванов, то нужно создать такое полное имя для папки:

*Мои документы\12345\Иванов\*

Теперь откроем блокнот - notepad (пуск - программы - стандартные - блокнот) и скопируем туда следующий текст:

```
<html>
<head>
<title>Мой первый шаг </title>
</head>
<body>
Здравствуйте, это моя первая страница.
<br>
Добро пожаловать! :)
</body>
</html>
```

Сохраним этот документ в своей папке, присвоив ему имя indexFIO.html. Здесь FIO - Ваши инициалы, написанные латинскими буквами.

Многие спотыкаются на фразе: " Сохраним этот документ, присвоив ему имя indexFIO.html", они говорят, что у них получается сохранить, только как текстовый документ indexFIO.txt, а вот как indexFIO.html - никак. Чтобы в дальнейшем избежать этого недоразумения, приводится это примечание.

Если вы сохраняете документ, через **Файл -> Сохранить**, то он сохраняется как indexFIO.txt. Надо сохранять ваш документ следующим образом: - **Файл -> Сохранить как**  
Дальше вводите имя своего документа, например, indexFIO.html (а не просто indexFIO; расширение .html должно быть обязательно).

Если вы уже сохранили ваш документ, как indexFIO.html, то при внесении изменений в этот документ вы можете уже сохранять их через **Файл -> Сохранить**.

Теперь откроем браузер, допустим, Internet Explorer, и откроем в браузере наш документ (Файл - Открыть - кнопка Обзор - Наш документ (indexFIO.html) ).

Если мы что-то изменим в нашем indexFIO.html документе (в блокноте), то, чтобы посмотреть, как это выглядит в нашем браузере, надо не забывать нажимать в браузере кнопку ОБНОВИТЬ (на клавиатуре - клавиша F5). Если изменений не видно, то это значит, что вы где-то что-то неправильно написали, или забыли сохранить документ.

**Сделайте скриншот открывшегося окна и вставьте его в отчёт. В дальнейшем самостоятельно выбирайте окна для скриншотов, подсказок не будет.**

## 14.2. Что такое тэги?

```
<html>
<head>
<title>Мой первый шаг </title>
</head>
<body>
Здравствуйте, это моя первая страница.
<br>
Добро пожаловать! :)
</body>
</html> посмотреть
```

Первое, что нужно усвоить: язык "HTML" - это то, что мы сейчас изучаем. Второе, что нужно усвоить: "HTML" не является языком программирования, он предназначен для разметки текстовых документов (т. е. с помощью него мы размечаем текст, таблицы, картинки и т. д. на нашей странице). То, как будут выглядеть картинки, текст и другие элементы на вашей странице, и как они будут располагаться относительно друг друга, определяют метки (tags или тэги).

Пример тэга: <br> (перенос текста на другую строку). Попробуйте вставить еще несколько тэгов <br> в нашем документе перед <добро пожаловать!>. Сохраните. Посмотрите в вашем браузере, что получилось.

Итак, все, что находится между < и > - это тэг. Текст, не находящийся между такими скобками < > - весь виден при просмотре в браузере.

Как мы видим на нашем примере - тэгов много, и они разные. Есть обязательные тэги. Обязательные теги, это такие теги, которые вы всегда



должны прописывать для каждой своей страницы.

```
<html>
```

Этот тэг должен открывать документ. Если есть открывающий тэг, то должен быть и закрывающий:

```
</html>
```

Некоторые тэги, вроде `<br>`, не требуют закрывающего тэга.

Итак, вернемся, к нашему документу.

### 14.3. Обязательные тэги

```
<html>
```

```
<head>
```

```
<title>Мой первый шаг</title>
```

```
</head>
```

```
<body>
```

Здравствуйте, это моя первая страница.

```
<br>
```

Добро пожаловать! :)

```
</body>
```

```
</html>
```

`<head>` `</head>` - голова документа

`<body>` `</body>` - тело документа

Все тэги, расположенные между `<head>` `</head>`, это что-то вроде служебной информации. Например `<title>`- заголовок. Зачем он? Откройте IE (Internet Explore) с нашим документом и устремите свой взгляд на заголовок окна... Увидели?:)

Все тэги, расположенные между `<body>` `</body>` - непосредственное содержание документа. Следующие несколько ступенек будут посвящены именно этим тэгам: мы узнаем, как менять цвет текста, фона, как делать текст крупнее-мельче, поговорим о картинках, таблицах и многом другом. Но прежде, все-таки закончим наш разговор о тэгах, в общем.

`<тэг>` `</тэг>` - не просто тэг, это контейнер - тэг, который может содержать внутри себя другие тэги (и текст).

Обратите внимание:

```
<тэг1><тэг2><тэг3> ... </тэг3></тэг2></тэг1>
```

Только такая очередность закрывающих тэгов верна: тэг, который мы открыли первым - закрываем последним, второй - предпоследним и т.д.

Т.е. следующая очередность нежелательна и не верна, она может привести к ошибкам на вашей страничке:

```
<тэг1><тэг2><тэг3> ... </тэг3></тэг1></тэг2>
```

Так что будьте внимательны, и пишите код своих страничек аккуратно и вдумчиво.

#### 14.4. Изменяем цвет текста

Начнем с малого, легкого, и даже интересного, и постепенно перейдем к сложному, большому, и, возможно, даже нудному, без чего не обходится ни одно обучение. Но не расстраивайтесь заранее. Итак, на этой ступеньке мы будем учиться раскрашивать. Для начала нам будет нужна палитра (таблица цветов, оригинал находится на страничке Артемия Лебедева <https://www.artlebedev.ru/colors/>).

Обязательно сохраните палитру у себя в папке на компьютере, она вам еще не раз пригодится: Файл - Сохранить как. Дело в том, что все обозначения цвета в Html прописываются именно таким способом. Например, белый цвет - ffffff. Но вернемся к нашей страничке. Давайте окрасим слова "Добро пожаловать" в красный.

```
<font color="#CC0000"> Добро пожаловать! :) </font>
```

Color - параметр (атрибут) для тэга font, он отвечает, в данном случае, за цвет заключенного в контейнер текста. Атрибут color, как и другие атрибуты, не принадлежит только одному тэгу, он может быть присвоен и некоторым другим тэгам.

Попробуйте вместо CC0000 подставить другие значения цветов для атрибута color. Обратите внимание, что значению цвета обязательно должен предшествовать значок <решетка> - #.

Кстати, для лучшего усвоения материала этой и следующих работ вам нужно не только исполнять от точки до точки предлагаемые задания, но и пытаться самостоятельно экспериментировать с кодом вашей странички на основе полученных вами знаний.

```
<html>
<head>
<title>Мой первый шаг </title>
</head>
<body>
Здравствуй, это моя первая страница.
<br>
<font color="#CC0000"> Добро пожаловать! :) </font>
</body>
</html> посмотреть
```

Тэг <font></font> - многофункционален. Им может задаваться не только цвет текста в конкретной части документа, но и размер шрифта, и вид шрифта (Arial), но об этом чуть позже.

Цвета в документе можно задавать и в открывающем тэге <body>:  
<body text="#336699">

Это значит, что весь текст страницы будет синим, кроме текста, для которого мы специально прописали `<font></font>`. Если цвет текста в `<body>` не задавать, то по умолчанию он будет черным.

```
<html>
<head>
<title>Мой первый шаг </title>
</head>
<body text="#336699">
Здравствуйте, это моя первая страница.
<br>
<font color="#CC0000"> Добро пожаловать!</font> :)
</body>
</html> посмотреть
```

С цветами для текста мы разобрались, теперь подумаем о фоне.

**Добавьте в текст своего файла indexFIO.html код, меняющий цвет текста (см. фрагменты, приведённые выше). Сохраните изменения.**

#### **14.5. Как изменять цвет фона страницы.**

Здесь вы узнаете, как изменять цвет фона документа, и немного об этике создателя страницы, о вкусе и чувстве меры.

Цвет фона устанавливается в уже знакомом нам тэге `<body>`:

```
<body bgcolor="#000000">
```

Для наглядности мы прописали черный цвет, вы же свой документ можете раскрасить любым другим, например, цветом `bgcolor="#CCFFFF"`. Кстати, если цвет в `<body>` не указывать, то по умолчанию он будет белым, хотя иногда цветом по умолчанию может являться любой другой, кроме белого, поэтому лучше всегда прописывайте цвет фона в `body`, а также цвет основного текста (как его задавать вы уже знаете) и цвета ссылок (с ними мы познакомимся чуть позже).

```
<html>
<head>
<title>Мой первый шаг </title>
</head>
<body text="#336699" bgcolor="#000000">
Здравствуйте, это моя первая страница.
<br>
<font color="#CC0000"> Добро пожаловать!</font> :)
</body>
</html> посмотреть
```

**Добавьте в текст своего файла indexFIO.html код, меняющий чёрный цвет фона на CCFFFF или любой другой (см. фрагмент, приведённый выше). Сохраните изменения.**

Обратите внимание: мы одновременно можем прописать в тэге <body> и цвет текста в документе, и цвет фона (это на всякий случай, если вы еще не поняли - одному тэгу может быть присвоено несколько атрибутов :).

```
<body text="#336699" bgcolor="#000000">
```

Теперь немного о принципах раскрашивания, которые желательно использовать при создании страниц.

Не злоупотребляйте яркими фонами (желтым, красным, салатовым, ну, и черным с яркой смесью текста на нем). Почему? Да, просто пожалейте глаза ваших друзей, знакомых и случайного посетителя. Не следует также употреблять слишком много различных цветов на вашей страничке, это, как и яркий фон, отвлекает от содержания, мешает прочтению и выставляет вас самого не в лучшем свете. Всё, что относится к дизайну создаваемых страниц, – это специальные знания, для изучения которых можно найти много источников.

#### **14.6. Параграфы и DIV. Учимся выравнивать текст**

В этом разделе мы поговорим о параграфах. Параграфы вводятся тэгом:

```
<p></p>
```

С помощью параграфов мы можем центрировать текст:

```
<p align="center">текст</p>
```

С помощью параграфов мы можем выровнять текст по левому краю:

```
<p align="left">текст</p>
```

По правому краю документа:

```
<p align="right">текст</p>
```

По обоим краям документа:

```
<p align="justify">текст</p>
```

Теперь введем параграфы в наш документ и посмотрим наглядно, что получится (в наш пример мы подставили параграф с атрибутом центрирования текста (align="center"), попробуйте подставить в параграф атрибут align с другими значениями: Left, Right, Justify).

Кстати, правильно говорить не выравнивание, а выключка: выключка по левому краю, правому, центру, по обоим краям. Если вы позже будете углубляться в область дизайна, то вам наверняка этот термин встретится. Между прочим, HTML не имеет к дизайну ни малейшего отношения, не считайте, что, изучив HTML, вы станете вдруг дизайнером, это большое заблуждение многих начинающих в области сайтостроительства.

```

<html>
<head>
<title>Мой первый шаг </title>
</head>
<body text="#336699" bgcolor="#000000">
<p align="center">
Здравствуйтесь, это моя первая страница.
<br>
<font color="#CC0000"> Добро пожаловать!</font> :) </p>
</body>
</html>

```

### Посмотрите результат

Ещё одно замечание об HTML и параграфах, которое стоит запомнить: никогда нельзя вводить в документ подобную конструкцию:

```
<p></p>
```

Пустые элементы <p> без какого-либо содержания (других тэгов или текста) могут игнорироваться браузерами. Не забывайте это. Заметьте, что текст в документе, если не задавать параграфы, всегда выравнивается по умолчанию по левому краю. Также запомните, что после закрывающего тега </p> автоматически происходит перенос строки. Но что делать, если вам этот перенос никак не нужен? Есть тэг альтернативный <p align="center">:

```
<center> текст </center>
```

```

<html>
<head>
<title>Мой первый шаг </title>
</head>
<body text="#336699" bgcolor="#000000">
<center>
Здравствуйтесь, это моя первая страница.
<br>
<font color="#CC0000"> Добро пожаловать!</font> :) </center>
</body>
</html>

```

Конечно, тэг <center> хорош, но остался нам он еще с прошлых версий HTML. Пока что этот тэг никто не отменял, и его можно использовать, но это не слишком желательно.

Как же лучше поступить? Лучше использовать тэг <div></div>, одно из назначений которого выравнивание содержимого вашего документа. Все четыре значения атрибута align можно употреблять с <div>:

```
<div align="center"> текст </div>
```

```
<div align="left"> текст </div>
```

```
<div align="right"> текст </div>
```

```
<div align="justify"> текст </div>
```

```
<html>
```

```
<head>
```

```
<title>Мой первый шаг </title>
```

```
</head>
```

```
<body text="#336699" bgcolor="#000000">
```

```
<div align="center">
```

```
Здравствуйте, это моя первая страница.
```

```
<br>
```

```
<font color="#CC0000"> Добро пожаловать!</font> :) </div>
```

```
<p align="justify">
```

Я совсем недавно начал(а) знакомиться с виртуальной жизнью, но мне по давней традиции тоже захотелось создать свою домашнюю страничку для моих новых виртуальных друзей и знакомых, чтобы они могли посмотреть мои фотографии, почитать обо мне, черкнуть пару строчек в мою гостевую книгу. А может и просто случайный посетитель вдруг захочет познакомиться со мной, и у меня появиться еще один виртуальный друг? :)

```
</p>
```

```
</body>
```

```
</html> посмотреть
```

**Добавьте в текст своего файла indexFIO.html код, меняющий содержание текста и его выравнивание (см. фрагмент, приведённый выше). Сохраните изменения.**

Параграф не может содержать в себе другие параграфы, и также тэг <div></div>. Т.е. следующие конструкции будут не верны, и вводить их в документ нельзя:

```
<p align="right">
```

```
<p>текст</p>
<p>текст</p>
<p>текст</p>
</p>
```

и

```
<p align="right">
<div>текст</div>
<p>текст</p>
<div>текст</div>
</p>
```

Однако <div> может содержать в себе параграфы: с помощью него мы можем сгруппировать их, допустим, по правому краю.

```
<div align="right">
<p>текст первого абзаца</p>
<p>текст второго абзаца</p>
<p>текст третьего абзаца</p>
</div>
```

## 14.7. Что такое заголовки и как задавать размер букв

Здесь вы научитесь выделять текст при помощи заголовков и узнаете еще одну функцию тэга <font></font>.

Начнем с заголовков. Существуют шесть уровней заголовков:

```
<H1> текст </H1>
<H2> текст </H2>
<H3> текст </H3>
<H4> текст </H4>
<H5> текст </H5>
<H6> текст </H6>
```

Итак, h1 - самый важный, h6 - наименее важный. Вводя заголовки в текст, вы позволяете вашему посетителю более легко ориентироваться на странице. Заголовками вы определяете структуру внутри текста.

Давайте теперь введем заголовок в наш документ. Очевидно, что фраза "Здравствуйте, это моя первая страница" так и просится, чтобы ее выделили:

```
<html>
<head>
<title>Мой первый шаг </title>
</head>
<body text="#336699" bgcolor="#000000">
<div align="center">
```

```
<H3>Здравствуй, это моя первая страница.</H3>
<br>
<font color="#CC0000"> Добро пожаловать!</font> :) </div>
<p align="justify">
```

Я совсем недавно начал(а) знакомиться с виртуальной жизнью, но мне по давней традиции тоже захотелось создать свою домашнюю страничку для моих новых виртуальных друзей и знакомых, чтобы они могли посмотреть мои фотографии, почитать обо мне, черкнуть пару строчек в мою гостевую книгу. А может и просто случайный посетитель вдруг захочет познакомиться со мной, и у меня появится еще один виртуальный друг? :)

```
</p>
</body>
</html>
```

### Посмотрите результат

В пример мы ввели заголовок третьего уровня `<h3></h3>`. Визуально заголовки отображаются не только более крупным шрифтом, но к тому же и полужирным. Также после закрывающего `</h3>` происходит автоматически перенос на новую строку.

Заголовки предназначены для выделения небольшой части текста (строки, фразы). Но, если вы хотите выделить большой фрагмент текста, то заголовки для этого использовать нельзя. Для этого предназначен атрибут `size` тэга `<font></font>`, который устанавливает желаемый размер шрифта:

```
<font size="+4"> текст </font>
<font size="+3"> текст </font>
<font size="+2"> текст </font>
<font size="+1"> текст </font>
<font size="+0"> текст </font>
<font size="-1"> текст </font>
<font size="-2"> текст </font>
```

Как всегда, применяем новые знания на практике. Давайте, выделим, что мы горим желанием завести еще одного виртуального друга.

```
<html>
<head>
<title>Мой первый шаг </title>
</head>
<body text="#336699" bgcolor="#000000">
<div align="center">
<H3>Здравствуй, это моя первая страница.</H3>
<br>
```



```
<font color="#CC0000"> Добро пожаловать!</font> :) </div>
```

```
<p align="justify">
```

Я совсем недавно начал(а) знакомиться с виртуальной жизнью, но мне по давней традиции тоже захотелось создать свою домашнюю страничку для моих новых виртуальных друзей и знакомых, чтобы они могли посмотреть мои фотографии, почитать обо мне, черкнуть пару строчек в мою гостевую книгу. А может и просто случайный посетитель вдруг захочет познакомиться со мной, и у меня появится <font size="+1">еще один виртуальный друг? :)</font>

```
</p>
```

```
</body>
```

```
</html>
```

### Посмотрите результат

Избегайте гигантомании. Ее симптомы: непомерно большие размеры шрифта, гигантские кнопки на пол экрана, вместо маленьких и милых кнопочек, и толстый (жирный) во всех случаях текст. Поэтому стоит напомнить: стандартный size (по умолчанию) ="+0".

Жалейте глаза своих посетителей, и они к вам потянутся.

**Добавьте в текст своего файла indexFIO.html код, меняющий заголовки и размеры фрагментов текста (см. фрагменты, приведённый выше). Сохраните изменения.**

## 14.8. Курсив, жирный текст, подчеркнутый и другие

Настало время поговорить о том, как определить стиль шрифта. В этом разделе вы узнаете, как выделить текст курсивом, подчеркнуть, перечеркнуть, сделать полужирным, как задавать моноширинный текст, как делать верхний и нижний индексы.

Сначала рассмотрим курсив и полужирный текст:

```
<b> Полужирный текст </b>
```

```
<i> Наклонный текст (курсив) </i>
```

Как видите, все просто. Можете их сами повставлять в тексте, где хочется, для пробы, а в примере мы снова помучаем виртуального друга:

```
<html>
```

```
<head>
```

```
<title>Мой первый шаг </title>
```

```
</head>
```

```
<body text="#336699" bgcolor="#000000">
```

```
<div align="center">
```

```
<H3>Здравствуй, это моя первая страница.</H3>
```

```
<br>
<font color="#CC0000"> Добро пожаловать!</font> :) </div>
<p align="justify">
```

Я совсем недавно начал(а) знакомиться с виртуальной жизнью, но мне по давней традиции тоже захотелось создать свою домашнюю страничку для моих новых виртуальных друзей и знакомых, чтобы они могли посмотреть мои фотографии, почитать обо мне, черкнуть пару строчек в мою гостевую книгу. А может и просто случайный посетитель вдруг захочет познакомиться со мной, и у меня появится **<b>** еще один виртуальный друг? :)</b>

```
</p>
</body>
</html>
```

Добавьте в текст своего файла indexFIO.html код, меняющий начертание слов "ещё один виртуальный друг" (см. фрагмент, приведённый выше). Сохраните изменения.

Теперь пара строк о моноширинном шрифте. Что это такое за шрифт? Это шрифт с символами одинаковой фиксированной ширины, как шрифт у пишущей машинки. А тэг для него следующий:

```
<tt> моноширинный шрифт </tt>
```

Также моноширинным шрифтом отображается текст заключенный в тэг `<pre></pre>`:

```
<pre>
текст  (куча пробелов)  текст
      текст  (куча пробелов)  текст
      текст  (куча пробелов)  текст
</pre>
```

У тэга `<pre>` есть одна замечательная особенность: текст, заключенный в него, выводится с точностью до пробела так, как вы его набили в блокноте. Этот тэг может быть полезен, допустим, для форматирования стихотворений.

К одному фрагменту текста может применяться сразу несколько тэгов:

```
<tt><b><i> текст </i></b></tt>
```

В нашем примере текст моноширинный, полужирный, и выделен курсивом. Не бойтесь использовать различные комбинации тэгов, экспериментируйте, но с умом!).

Подчеркнутый текст вводится при помощи тэга `<u>`:

```
<u> Подчеркнутый текст </u>
```

Тэги `<strike>` и `<s>` представляют текст перечеркнутым шрифтом, можете использовать какой вам угодно из них, принципиальной разницы между ними нет:

~~Перечеркнутый~~

Перечеркнутый

Тэг `<big>` представляется крупным шрифтом, а `<small>` малым шрифтом относительно основного текста:

Малый

Нормальный текст

**БОЛЬШОЙ**

Тэги `Sup` и `Sub` - определяют верхний и нижний индексы. `Sup` - верхний, `Sub` - нижний. Где они могут пригодиться? Ну, например, при написании какой-нибудь формулы -  $H_2O$  (все, что мы выносим со школьной скамьи:).

Верхний индекс `<sup>`  `</sup>`

Нижний индекс `<sub>`  `</sub>`

**Добавьте в текст своего файла `indexFIO.html` тэги, меняющие начертание различных фрагментов текста (подчёркивание, зачёркивание, крупный и малый шрифты. Используя верхний и нижний индексы, напишите формулу воды).). Сохраните изменения.**

## 14.9. Стандартные шрифты

Мы уже с вами ознакомились с атрибутами `size` и `color` для тэга `<font>`. Есть ещё атрибут `face`. С помощью `face` мы можем задать тип шрифта.

Попробуйте ввести следующую конструкцию в наш документ для части текста, чтобы задать шрифт `Arial`:

`<font face="arial">` текст (шрифт `Arial`)`</font>`

Эти типы шрифтов являются стандартными, и обычно должны находиться на компьютере каждого пользователя:

Times;

Times New Roman;

Arial;

Helvetica;

Courier;

Verdana;

Tahoma;

Cosmic Sans;

Garamond

Вы можете безбоязненно использовать любой из них.

В атрибуте face можно указать сразу несколько типов шрифтов:

`<font face="arial, verdana, courier"> текст (шрифт Arial) </font>`

В этом случае если у посетителя не окажется на компьютере шрифта Arial, то текст будет отображен шрифтом Verdana. Если и Verdana нет на компьютере вашего посетителя, то текст будет отображен шрифтом Courier. Т.е. в атрибуте face мы можем задать список разделенных запятыми названий шрифтов, которые браузер вашего посетителя должен попытаться найти у него на компьютере и отобразить в порядке приоритета.

Готовьте отчёт!

## **ЛАБОРАТОРНАЯ РАБОТА №15 ОСНОВЫ РАЗРАБОТКИ И СОЗДАНИЯ WEB-СТРАНИЦ. РАБОТА С ИЗОБРАЖЕНИЯМИ. ССЫЛКИ**

*Цель лабораторной работы:* научиться с помощью простейших средств создавать web-страницы, содержащие текст и иллюстрации.

### **15.1. Что такое путь? Как вставлять картинки**

Запомните, чем меньше объем страницы (сайта), тем быстрее он загрузится в компьютер посетителя вашего сайта.

Также не следует изощряться с анимированными картинками: они отвлекают внимание от содержания странички, а анимированная картинка не к месту смотрится нелепо. Постарайтесь также создавать свои картинки для своего сайта, а не собирать их по бесплатным коллекциям с графикой, пусть ваш сайт будет уникальным.

Для вставки картинки в документ используется тэг:

``

Вместо my.jpg мы можем подставить имя любой картинки (me.gif, main.png). Самое главное понять, что все расположенное между кавычками - ссылка (путь к картинке). В этом примере картинка лежит в том же каталоге (директории, папке), в которой лежит и наш документ. Если картинка лежит в поддиректории (в папке, которая лежит в вашей основной папке), то ссылка на неё будет выглядеть так:

``

Если картинка лежит на уровень выше, а документ находится в поддиректории, то ссылка на неё будет такой:

``

Если картинка лежит на другом сайте, то путь прописывается полностью:

``

Для вашего удобства помещайте картинку в ту же папку, что и документ, тогда путаницы будет меньше. Также обратите внимание на то, что MY.jpg, my.JPG, my.jpg и MY.JPG - это разные имена файлов. Никогда не забывайте, что регистр нужно учитывать.

Создайте в блокноте файл и вставьте в него код, приведённый ниже (в этом фрагменте "имя файла.тип" имя файла-картинки, который вы поместили в папку с документом, например фотографию). Сохраните файл под именем indexFIO\_1.html в своей папке

```
<html>
<head>
<title>Мой первый шаг </title>
</head>
<body text="#336699" bgcolor="#000000">
<div align="center">
<H3>Здравствуйтесь, это моя первая страница.</H3>
<br>
<font color="#CC0000"> Добро пожаловать!</font> :) </div>
<p align="justify">
 Я совсем недавно начал(а)
знакомиться с виртуальной жизнью, но мне по давней традиции
тоже захотелось создать свою домашнюю страничку для моих новых
виртуальных друзей и знакомых, чтобы они могли посмотреть мои
фотографии, почитать обо мне, черкнуть пару строчек в мою
гостевую книгу. А может и просто случайный посетитель вдруг
захочет познакомиться со мной, и у меня появится <b> еще один
виртуальный друг? :)</b>
</p>
</body>
</html>
```

**Посмотрите, что у вас получилось. Не очень симпатично? Было бы лучше, если фон был иного цвета, а текст аккуратно располагался, например, сбоку от картинки. Измените цвет фона на более радостный, например, тот, который был у вас в предыдущей работе.**

Как же сделать так, чтобы текст располагался весь рядом с картинкой, а не только одна его строчка. Все очень просто. Вспомним об атрибуте align, который отвечает за выравнивание. Атрибут align есть и у картинок:

```

```

Это означает, что картинка будет прижата к левому краю экрана, а текст будет обтекать ее справа. Чтобы сделать наоборот (картинка справа, текст слева) надо прописать right:

``

Но это не все: текст может располагаться внизу картинки (это по умолчанию) - (1), посередине - (2), и вверху - (3):

(1) - ``

(2) - ``

(3) - ``

Кроме атрибута `align` для тэга `<img>` можно ввести еще несколько атрибутов:

(1) - ``

(2) - ``

(3) - ``

(4) - ``

(5) - ``

(6) - ``

(1) - атрибут `vspace` - задает расстояние между текстом и рисунком (по вертикали). Расстояние задается в пикселях. Pixel - минимальная единица изображения, точка. Например, разрешение экрана 800x600 - 800 на 600 точек. В нашем примере расстояние равно 10 пикселям.

(2) - атрибут `hspace` - тоже задает расстояние между текстом и рисунком, но по горизонтали. Расстояние задается в пикселях. В нашем примере оно равно 30 пикселям (точкам).

(3) - атрибут `alt` - краткое описание картинки. Если навести курсором мыши на рисунок, и так подержать его (курсор) несколько секунд, появится описание картинки. В нашем случае это будет фраза - "моя фотография". Если параметр `alt` не задавать, описания не будет. Но умные люди говорят, что описание картинкам задавать следует (особенно, если это кнопки), т.к. есть особенные люди, которые бродят по интернету с отключенной графикой. Без `alt` им не будет видно, на что нажимать (если картинка является ссылкой или кнопкой в меню), т.к. картинка не отображается, а при заданном `alt`, можно увидеть надпись, для чего картинка предназначена.

(4) атрибут `width` - ширина самой картинки (в пикселях). Если ширину не задавать специально, то по умолчанию она будет равна реальной ширине картинки (а так вы можете ее сделать или уже, или шире).

(5) - атрибут `height` - высота самой картинки (тоже в пикселях). Так же как в случае с `width` высоту (`height`) картинки можно и не задавать. Правда, те же умные люди говорят, что размеры картинок следует задавать, для тех же особенных людей с отключенной графикой...

(6) - атрибут `border` - рамка вокруг самой картинки (в пикселях). Можно не задавать. Однако, по умолчанию, рамка вокруг картинки есть всегда. И если вы хотите убрать ее, тоставляйте атрибут `border` равным нулю.

Введем следующие атрибуты для нашей картинки:

```

```

Наша картинка будет прижата к левому краю экрана, текст будет обтекать ее справа, расстояние до текста по горизонтали - 30 пикселей, по вертикали - 5 пикселей (чтобы красиво все смотрелось), ну, и если вы наведете на картинку курсор, то выскочит надпись - "моя фотография".

```
<html>
<head>
<title>Мой первый шаг </title>
</head>
<body text="#336699" bgcolor="#000000">
<div align="center">
<H3>Здравствуйте, это моя первая страница.</H3>
<br>
<font color="#CC0000"> Добро пожаловать!</font> :) </div>
<p align="justify">
 Я совсем недавно начал(а) знакомиться с
виртуальной жизнью, но мне по давней традиции тоже захотелось
создать свою домашнюю страничку для моих новых виртуальных
друзей и знакомых, чтобы они могли посмотреть мои фотографии,
почитать обо мне, черкнуть пару строчек в мою гостевую книгу.
<br><br> А может и просто случайный посетитель вдруг захочет
познакомится со мной, и у меня появится <b> еще один виртуальный
друг? :) </b><br><br> На фотографии изображен(а) я. Качество
картинки не очень хорошее, к сожалению, поэтому она не четкая и
разглядеть черты моего лица немного проблематично. Но в целом
заметно, что я вполне ничего:) <br><br> Если ты так тоже думаешь,
то давай как-нибудь встретимся, поболтаем, чайку попьем в
кафешке?:) Кто знает, может быть мы и в реальной жизни станем
друзьями:)
</p>
</body>
</html>
```

Попробуйте прежде, чем двинуться дальше, поподставлять и другие атрибуты и их значения в наш пример: задать разное значение в пикселях для атрибутов и т.д. - это поможет вам лучше усвоить урок, все-таки тема большая и уже не такая простая.

Как же расположить не текст относительно картинки, а саму картинку в центре экрана (справа, слева). Здесь все очень просто, вспомните параграфы (<p></p>) или другие тэги для выравнивания текста, о которых мы говорили, ведь они не только текст

выравнивают (см. наш пример, наша картинка заключена в параграф, как и текст).

Любую картинку можно сделать фоном документа. Все необходимые для этого сведения прописываются в открывающем тэге боди:

```
<body text="#336699" bgcolor="#000000" background="ваш_фон.jpg">
```

Атрибут background указывает на то, где лежит фоновая картинка, в нашем примере он указывает на то, что наша фоновая картинка лежит в той же директории (папке), что и наш документ.

Но зачем оставлять атрибут bgcolor, если есть background? А вдруг фоновая картинка не загрузится (представьте, такое может быть), тогда сами поймете зачем.

Добавьте в текст своего файла indexFIO\_1.html код для обтекания изображения текстом справа (см. фрагменты, приведённые выше). Сохраните изменения.

## 15.2. Ссылка

Наша страничка может состоять из нескольких документов. Один из них главный (indexFIO\_1.html) - он открывается первым и должен обязательно лежать на вашем сайте в интернете. Остальные документы вы можете называть как угодно (photos.html, about\_me.html, my\_pets.html, friends.html, gh516hgd.html). Они все могут лежать в одной директории (папке), а могут в разных.

При помощи ссылок мы связываем эти документы. Так с главной страницы по ссылкам мы можем перейти на страницу с фотографиями, с этой страницы мы можем перейти обратно на главную страницу, или, допустим, на страницу с нашими стихотворениями, и т.д.

Ссылкой на эти другие документы (части нашей странички) может быть текст (фраза, слово), а может быть и картинка. Рассмотрим пока только текстовую ссылку.

Пусть prf.html - документ с вашими фотографиями. Тогда мы можем без зазрения совести фразу "посмотреть мои фотографии" в главном документе indexFIO\_1.html сделать ссылкой на pr.html:

```
<a href="prf.html">посмотреть мои фотографии</a>
```

Тэг <a></a> делает ссылкой заключенную в него картинку или фразу (текст). Принципы прописывания пути мы уже знаем:

(1) - <a href="prf.html">мои фотографии</a>

(2) - <a href="photos/prf.html">мои фотографии</a>

(3) - <a href="http://www.homepage.ru/prf.html">мои фотографии</a>

В случае (1) документ лежит в той же директории (папке), что и



документ, в котором мы ссылаемся на prf.html, в случае (2) документ лежит в поддиректории /photos, в случае (3) мы ссылаемся на сайт <http://www.homepage.ru>, где лежит нужный нам документ.

Первые два примера (1) и (2) - называют относительными путями. (3) - абсолютный, т.е. указанный полностью, включая имя сайта (в нашем случае - <http://www.homepage.ru>). Абсолютный путь мы используем, когда ссылаемся на чужие странички, относительный мы используем, когда ссылаемся на документы внутри нашего сайта.

Для всех ссылок в нашем документе мы можем прописать цвета: link - цвет просто ссылки, alink - цвет активной ссылки (нажатой), vlink - цвет уже посещенной ссылки.

```
<body text="#336699" bgcolor="#000000" link="#339999"
alink="#339999" vlink="#339999">
```

Как и цвет для всего текста в документе, цвета ссылок мы прописываем в <body>. В нашем примере цвета для просто ссылки, активной и посещенной - одинаковые, но они могут быть разными - это на ваше усмотрение.

Итак, пропишем цвета для ссылки и саму ссылку в нашем документе:

```
<html>
<head>
<title>Мой первый шаг </title>
</head>
<body text="#336699" bgcolor="#ccffff" link="#339999"
alink="#339999" vlink="#339999">
<div align="center">
<H3>Здравствуй, это моя первая страница.</H3>
<br>
<font color="#CC0000"> Добро пожаловать!</font> :) </div>
<p align="justify">
 Я совсем недавно начал(а)
знакомиться с виртуальной жизнью, но мне по давней традиции
тоже захотелось создать свою домашнюю страничку для моих новых
виртуальных друзей и знакомых, чтобы они могли <a
href="prf.html">посмотреть мои фотографии</a>, почитать обо мне,
черкнуть пару строчек в мою гостевую книгу. <br><br> А может и
просто случайный посетитель вдруг захочет познакомиться со мной,
и у меня появится <b> еще один виртуальный друг? :) </b><br><br>
На фотографии изображен(а) я. Качество картинки не очень
хорошее, к сожалению, поэтому она не четкая и разглядеть черты
моего лица немного проблематично. Но в целом заметно, что я
вполне ничего:) <br><br> Если ты так тоже думаешь, то давай как-
```

нибудь встретимся, поболтаем, чайку попьем в кафешке?..) Кто знает, может быть мы и в реальной жизни станем друзьями:)

```
</p>
</body>
</html>
```

Как вы помните, мы можем задать различные цвета для разных блоков текста в нашем документе (`<font color=":"></font>`). Для текстовых ссылок мы тоже можем задать разный цвет - это делается при помощи тэга `<font>` и его атрибута `color`:

```
<a href="prf.html"><font color="#CC0000">посмотреть мои
фотографии</font></a>
```

Обратите внимание, `<font color=":"></font>` - прописывается внутри тэга `<a></a>`, если вы пропишите иначе, то у вас не получится задать вашей ссылке цвет отличный от цвета других ссылок в документе.

Создайте новый документ `pr.html` в той же папке, где находится наш главный документ `indexFIO_1.html`. Содержание документа выдумайте сами, у вас для этого уже достаточно знаний. Добавьте в текст своего файла `indexFIO_1.html` код для ссылки на файл `pr.html`. Текст ссылки должны быть слова "посмотреть мои фотографии". (см. фрагменты, приведённые выше). Сохраните изменения.

## **ЛАБОРАТОРНАЯ РАБОТА №16**

### **ОСНОВЫ РАЗРАБОТКИ И СОЗДАНИЯ WEB-СТРАНИЦ. РАБОТА С ТАБЛИЦАМИ**

*Цель лабораторной работы:* научиться с помощью простейших средств создавать web-страницы, содержащие текст и иллюстрации и таблицы.

#### **16.1. Учимся создавать таблицы**

При создании сайтов таблицы используются очень часто. Главное назначение таблиц на странице – удобное расположение объектов на странице (картинок, текстовых блоков) относительно друг друга в ячейках таблиц. При этом границы ячеек делают невидимыми.

Таблица задается тэгом:

```
<table></table>
```

Но это еще не все: таблица состоит из строк и столбцов (ячеек), поэтому надо еще указать и их.

```
<tr></tr> - строчка таблицы
```

```
<td></td> - столбец (ячейка) таблицы
```

Опишем таблицу из двух строк и трех столбцов (ячеек). Для наглядности выделим ячейки таблицы разными цветами. Границы таблицы не заданы, поэтому их не будет видно.

Сначала мы задали строки.:

```

<table>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</table>

```

В нашем примере две строки. В каждой строке по три столбца (ячейки).

Кстати, когда вы будете создавать свой сайт и таблицы, лучше рисуйте все сначала на бумаге, так вам будет легче потом верстать, т.к. вы все наглядно будете видеть на бумаге, и у вас будет меньше вероятности ошибиться.

Теперь нам надо заполнить получившийся каркас:

```

<table>
<tr>
<td>1x1</td>
<td>1x2</td>
<td>1x3</td>
</tr>
<tr>
<td>2x1</td>
<td>2x2</td>
<td>2x3</td>
</tr>
</table>

```

Первая цифра в загадочных надписях - это номер ряда, а вторая номер ячейки (1x2 - первый ряд, вторая ячейка). Это для наглядности. Если посмотреть то, что уже у нас с вами получилось, то это будет выглядеть так:

```

1x1 1x2 1x3
2x1 2x2 2x3

```

Фон задается атрибутом bgcolor="цвет\_фона". Фон можно задать для таблицы в целом, для ряда, для ячейки (в пределах одного ряда). В нашем случае мы задаем фон для каждой ячейки.

```

<table>
<tr>

```

```

<td bgcolor="#FFCC33">1x1</td>
<td bgcolor="#336699">1x2</td>
<td bgcolor="#FFCC33">1x3</td>
</tr>
<tr>
<td bgcolor="#336699">2x1</td>
<td bgcolor="#FFCC33">2x2</td>
<td bgcolor="#336699">2x3</td>
</tr>
</table>

```

**Создайте в блокноте файл и вставьте в него код для пустой html-страницы. Добавьте код для создания таблицы, приведённый выше. Сохраните файл под именем indexFIO\_2.html в своей папке.**

**Посмотрите, что у вас получилось.**

Если вы хотите задать фон для ряда, то атрибут bgcolor мы прописываем для тэга <tr>:

```
<tr bgcolor="#FFCC33">
```

Если вы хотите задать фон для всей таблицы, то атрибут bgcolor мы прописываем для тэга <table>:

```
<table bgcolor="#FFCC33">
```

Однако, если при заданном фоне для всей таблицы, вы задаёте свой фон для ряда или ячейки, то этот ряд или ячейка будут иметь фон отличный от всей таблицы.

Попробуйте самостоятельно задать фон для таблицы и для ряда (это для усвоения материала). Когда все усвоится переходите дальше.

**Добавьте в текст своего файла indexFIO\_2.html код для фона таблицы и для ряда. Сохраните изменения.**

## 16.2. Таблицы, вертикальное выравнивание (valign)

Мы пока не задали высоту и ширину ячейкам нашей таблицы. Вспомните об атрибутах height и width - их можно задать для всей таблицы, для одного ряда, для ячейки (столбца). Высота и ширина могут задаваться как в пикселях, так и процентах. В нашем случае мы зададим ширину и высоту в пикселях для столбцов (ячеек).

```

<table>
<tr>
<td height="35" width="50" bgcolor="#FFCC33"> 1x1 </td>
<td width="50" bgcolor="#336699"> 1x2 </td>
<td width="50" bgcolor="#FFCC33"> 1x3 </td>
</tr>
<tr>

```

```

<td height="35" width="50" bgcolor="#336699"> 2x1 </td>
<td width="50" bgcolor="#FFCC33"> 2x2 </td>
<td width="50" bgcolor="#336699"> 2x3 </td>
</tr>
</table>

```

Замечание: Если в ряду вы задаете для какой-либо ячейки высоту большую, чем для других, то все ячейки (столбцы) вашего ряда станут по высоте равны наибольшей (это же относится к рядам). Ряд – это совокупность всех ячеек (столбцов). Мы задали в нашем примере ширину для каждой ячейки (столбца)).

Можно задать высоту и ширину для всей таблицы, тогда все ячейки (столбцы) и ряды поделят данное им пространство поровну, если не задавать им это пространство персонально (в процентах от общей ширины (высоты) таблицы или пикселях).

Атрибуты height и width можно задать в процентах:

```

<td width="30%"> содержимое ячейки </td>

```

Сумма ширин должна равняться 100%.

Теперь осталось лишь выровнять содержимое (текст) внутри таблицы:

```

<table>
<tr>
<td height="35" width="50" bgcolor="#FFCC33"> <center> 1x1
</center> </td>
<td width="50" bgcolor="#336699"> <center> 1x2 </center> </td>
<td width="50" bgcolor="#FFCC33"> <center>1x3 </center>
</td>
</tr>
<tr>
<td height="35" width="50" bgcolor="#336699"> <center> 2x1
</center> </td>
<td width="50" bgcolor="#FFCC33"> <center> 2x2 </center>
</td>
<td width="50" bgcolor="#336699"> <center> 2x3 </center> </td>
</tr>
</table>

```

Поскольку содержимое каждой ячейки как бы обстановка отдельной комнаты, то и тэги для центрирования текста нужно

прописывать в нашем примере для содержимого (текста) каждой ячейки.

В каждой ячейке могут находиться и картинки, и текст (+тэги и атрибуты применяемые к ним), и даже таблицы (в этом случае они называются - вложенные таблицы). Т.е. тэги, которые мы применяем для форматирования содержимого (контента) - все те же.

Теперь немного о вертикальном выравнивании содержимого таблицы, т.е. о том, как можно сделать так, чтобы содержимое ячейки не только располагалось ровно посередине ее (как по умолчанию), а еще вверху или внизу. Вертикальное выравнивание задается следующим атрибутом - `valign="middle"` (`top`, `bottom`) - содержимое конкретной ячейки будет находиться в середине ячейки (наверху или внизу):

```
<table>
<tr>
<td height="35" width="50" bgcolor="#FFCC33" valign="top">
<center>1x1</center> </td>
<td width="50" bgcolor="#336699"> <center>1x2</center> </td>
<td width="50" bgcolor="#FFCC33" valign="bottom">
<center>1x3</center> </td>
</tr>
<tr>
<td height="35" width="50" bgcolor="#336699" valign="bottom">
<center>2x1</center> </td>
<td width="50" bgcolor="#FFCC33"> <center>2x2</center> </td>
<td width="50" bgcolor="#336699" valign="top"> <center>2x3</center>
</td>
</tr>
</table>
```

Атрибут `valign` прописан не для всех ячеек, а только для тех, где мы захотели, чтобы текст располагался сверху или снизу, чтобы зря не тратить время и усилия.

**Добавьте в текст своего файла `indexFIO_2.html` код для выравнивания текста внутри таблицы и в её отдельных ячейках. Сохраните изменения.**

**Готовьте отчёт!**

## ПРИЛОЖЕНИЕ 1. МЕТОДЫ ДЛЯ РАБОТЫ СО СТРОКАМИ

	Пояснения
<code>Compare()</code>	Сравнивает две строки и возвращает целое число, которое показывает их относительное положение в порядке сортировки. Возвращаемое число будет равно нулю, если значения параметров равны.

Concat()	Соединяет в одну строку две и более строки. При этом разделители не добавляются.
Copy() CopyTo()	Методы Copy и CopyTo служат для копирования строки или подстроки в другую строку или в массив типа Char.
Format()	Форматирует строку, используя строго заданный формат. Для этого заменяет каждый элемент формата в указанной строке текстовым эквивалентом значения соответствующего объекта.
Join()	Конкатенация (соединение) массива строк в единую строку. При конкатенации между элементами массива вставляются разделители. Операция, заданная методом Join, является обратной к операции, заданной методом Split.
Length	Свойство, которое возвращает количество символов в строке.
EndsWith()	Проверяет, заканчивается ли строка определённой последовательностью символов.
Insert()	Вставляет новую строку в уже существующую.
LastIndexOf()	Возвращает индекс последнего вхождения элемента в строку.
PadLeft()	Выравнивает строку по правому краю, пропуская все пробелы или другие специально заданные символы.
PadRight()	Выравнивает строку по левому краю, пропуская все пробелы или другие специально заданные символы.
Remove()	Удаляет заданное число символов из строки.
Replace()	Заменяет подстроку в заданной позиции на новую подстроку.
Split()	Возвращает подстроку, отделённую от основного массива определённым символом. На вход методу Split передается один или несколько символов, интерпретируемых как разделители. Объект string, вызвавший метод, разделяется на подстроки, ограниченные этими разделителями. Из этих подстрок создается массив, возвращаемый в качестве результата метода. Другая реализация позволяет ограничить число элементов возвращаемого массива.
StartsWith()	Определяет, начинается ли строка с определённой последовательности символов.

Substring()	Извлекает подстроку из строки.
ToCharArray()	Копирует символы из строки в массив символов.
ToLower()	Преобразует символы в строке к нижнему регистру.
ToUpper()	Преобразует символы в строке к верхнему регистру.
Trim()	Удаляет все вхождения определённых символов в начале и в конце строки.
TrimEnd()	Удаляет все вхождения определённых символов в конце строки.
TrimStart()	Удаляет все вхождения определённых символов в начале строки.

## ПРИЛОЖЕНИЕ 2. МЕТОДЫ ДЛЯ РАБОТЫ С МАССИВАМИ

Метод или свойство	Пояснения
Concat()	Объединяет две последовательности.
Contains()	Определяет, содержится ли указанный элемент в массиве.
CopyTo()	Копирует все элементы текущего массива в заданный массив.
GetLength()	Получает 32-разрядное целое число, представляющее количество элементов в заданном измерении массива. Примером метода GetLength может служить метод GetLength(0), который возвращает число элементов в первом измерении массива (например, количество строк в двумерном массиве).
Intersect()	Находит пересечение множеств, представленных двумя массивами.
Length	Свойство, которое возвращает целое число, представляющее общее число элементов во всех измерениях массива.
Max()	Возвращает максимальное значение, содержащееся в массиве.
Min()	Возвращает минимальное значение, содержащееся в массиве.
Reverse()	Изменяет порядок элементов массива на противоположный.
Sum()	Вычисляет сумму последовательности числовых значений.



## ЛИТЕРАТУРА

1. Эндрю Троелсен. Язык программирования C# 6.0 и платформа .NET 4.6. М.: Вильямс. 2016 г. 1440 с.
2. Албахари, Дж. C# 7.0. Справочник: Пер. с англ./ Дж. Албахари, Б. Албахари. – 3-е изд. – СПб.: БХВ-Петербург, 2018. – 1024 с.: ил.
3. Дёмин, Антон Юрьевич. Информатика. Лабораторный практикум: учебное пособие для прикладного бакалавриата / А. Ю. Дёмин, В. А. Дорофеев; Национальный исследовательский Томский политехнический университет (ТПУ). — Москва: Юрайт, 2016. — 132 с.: ил.
4. Немировский В.Б., Стоянов А.К. Информатика: учебное пособие / В.Б. Немировский, А.К. Стоянов; Национальный исследовательский Томский политехнический университет (ТПУ). — Томск: Изд-во ТПУ, 2012. — 314 с.
5. Кирьянов Д.В. Mathcad 15/Mathcad Prime 1.0 — СПб.: БХВ-Петербург, 2012. — 432 с.: ил.
6. Бекаревич, Ю.Б. Самоучитель Access 201/ Ю.Б. Бекаревич, Н.В. Пушкина. – СПб.: БХВ-Петербург, 2011. — 432 с.: ил.
7. Гурвиц Г. Л. Microsoft Access 2010. Разработка приложений на реальном примере. - СПб.: БХВ- Петербург, 2010. 496с.: ил.
8. Дакет Дж. HTML и CSS. Разработка и дизайн веб-сайтов. Изд-во «Эксмо», 2017, 480 с.

Учебное издание

НЕМИРОВСКИЙ Виктор Борисович

## ПРАКТИКУМ ПО ИНФОРМАТИКЕ

Учебное пособие

Рецензенты

Компьютерная вёрстка **И.О. Фамилия**  
Дизайн обложки **И.О. Фамилия**

**Зарегистрировано в Издательстве ТПУ**  
**Размещено на корпоративном портале ТПУ**

**в полном соответствии с качеством предоставленного оригинал-макета**



---


Национальный исследовательский Томский политехнический университет

Система менеджмента качества

Издательства Томского политехнического университета сертифицирована

NATIONAL QUALITY ASSURANCE по стандарту BS EN ISO 9001:2008

---

**ИЗДАТЕЛЬСТВО**  **ТПУ**. 634050, г. Томск, пр. Ленина, 30  
Тел./факс: 8(3822)56-35-35, [www.tpu.ru](http://www.tpu.ru)