Министерство образования Республики Беларусь
Учреждение образования «Витебский государственный университет имени П.М. Машерова»
Кафедра информатики и информационных технологий

Н.Д. Адаменко, В.В. Шедько, А.В. Осипов

ИНФОРМАЦИОННЫЕ СИСТЕМЫ И СЕТИ

Методические рекомендации

Витебск
ВГУ имени П.М. Машерова
2016

УДК 004.415.2:004.439(075.8)
ББК 32.972я73+32.973.42я73
А28

Печатается по решению научно-методического совета учреждения образования «Витебский государственный университет имени П.М. Машерова». Протокол № 2 от 24.12.2015 г.

Авторы: доцент кафедры информатики и информационных технологий ВГУ имени П.М. Машерова, кандидат педагогических наук **Н.Д. Адаменко;** старшие преподаватели кафедры информатики и информационных технологий ВГУ имени П.М. Машерова **В.В. Шедько, А.В. Осипов**

Р е ц е н з е н т :
кафедра автоматизации технологических процессов УО «ВГТУ»

Адаменко, Н.Д.

А28 Информационные системы и сети : методические рекомендации / Н.Д. Адаменко, В.В. Шедько, А.В. Осипов. – Витебск : ВГУ имени П.М. Машерова, 2016. – 50 с.

Учебное издание содержит основной материал по предмету «Информационные системы и сети», вопросы и индивидуальные задания для лабораторных занятий и самостоятельной работы и краткие теоретические сведения.

Методические рекомендации предназначены для обучения дисциплине «Информационные системы и сети» студентов специальности «Математика и информатика» и могут быть использованы для обучения дисциплине «Библиотечное сайтостроение» студентов специальности «Библиотечно-информационная деятельность (информатизация)».

УДК 004.415.2:004.439(075.8)
ББК 32.972я73+32.973.42я73

СОДЕРЖАНИЕ

1. ОФИСНОЕ ПРОГРАММИРОВАНИЕ

Данный раздел посвящен усвоению основных понятий и приобретению навыков офисного программирования.

1.1. ИСПОЛЬЗОВАНИЕ МАКРОКОМАНД В MS EXCEL

ЛАБОРАТОРНАЯ РАБОТА № 1

ОСНОВНЫЕ СВЕДЕНИЯ

MacroRecorder – это транслятор, создающий программу на языке VBA, которая является результатом перевода на язык VBA действий пользователя с момента запуска транслятора до окончания записи макроса.

Для активизации MacroRecorder следует выбрать команду **Сервис/Макрос/Начать запись.** В результате выполнения команды на экран будет выведено диалоговое окно **Запись макроса.**

В диалоговом окне следует задать имя макроса, а также можно указать комбинацию клавиш для запуска его на выполнение. Последнее необходимо в тех случаях, когда макрос часто используется.

В поле **Сохранить в** выбирается книга, в которой будет сохранен макрос. Если выбрана **Личная книга макросов**, то макрос будет сохранен в специальной скрытой книге. Макросы, записанные в этой книге доступны для других рабочих книг. Личная книга макросов может быть отображена с помощью команды **Окно/Отобразить.**

Если выбран параметр **Эта книга**, то макрос будет сохранен на новом листе модуля в активной рабочей книге.

После щелчка на кнопке **ОК** начинается запись макроса. В окне Excel появится панель инструментов записи макроса, на которой расположены кнопки **Остановить запись** и **Выполнить макрос.**

Перед тем как записать макрос, необходимо тщательно спланировать шаги и команды, которые он будет выполнять. Если при записи макроса была допущена ошибка, ее исправление будет также записано.

После создания таблицы следует завершить создание макроса, нажав кнопку **Остановить запись** на панели инструментов записи макроса.

Для запуска макроса можно назначить кнопку, рисованный объект или элемент управления графического объекта на листе.

Для назначения макрокоманды кнопке следует выполнить следующие действия:

- Выбрать в меню **Сервис** команду **Настройка**, затем в окне **Настройка** – выбрать вкладку **Команды**.
- Выбрать в списке **Категории** команду **Макросы**, в списке команд этой категории выбрать команду **Настраиваемая кнопка** и перетащить ее на панель инструментов **Стандартная**.
- Не закрывая окно **Настройка**, для назначения вновь созданной кнопке макроса щелкнуть правой кнопкой мыши по настраиваемой кнопке и в контекстном меню выбрать команду **Назначить макрос**.
- В окне **Назначить макрос** в поле **Имя макроса** ввести нужное имя макроса и щелкнуть на кнопке ОК.
- Используя команду **Выбрать значок для кнопки** в контекстном меню выбрать значок для созданной кнопки.
- Закрыть окно **Настройка.**
- С помощью редактора кнопок можно изменить вид кнопки, назначенной для макроса (**Сервис/Настройки**, выбрать назначенную для макроса кнопку, щелкнуть на кнопке **Изменить значок для кнопки**)

Для запуска макроса можно установить указатель на пункт **Макрос** в меню **Сервис** и выбрать команду **Макросы**. В поле **Имя макроса** следует ввести имя макроса, который нужно выполнить, и нажать кнопку **Выполнить**. Для запуска макроса также можно воспользоваться назначенным сочетанием клавиш или созданной для запуска кнопкой.

Прервать выполнение макроса можно нажатием кнопки **Esc.**

Для изменения макроса необходимо выполнить команду **Сервис/ Макрос** и выбрать команду **Макросы**. Выбрать нужный макрос и нажать кнопку **Изменить.**

Изменив макрос, следует вернуться в окно Excel, выбрав в меню **Файл** окна **Visual basic** команду **Закрыть, и вернуться в Microsoft Excel.**

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Установить средний или низкий уровень безопасности (**Сервис/Параметры/Безопасность**) и создать макрос **Ведомость** для формирования таблицы, приведенной на рисунке. В случае неявки студента в столбец оценок заносится значение “н/я”. Столбцы F-J носят вспомогательный характер и служат для вычисления количества каждой из оценок. Поэтому после создания таблицы их следует скрыть (**Формат/Столбец/Скрыть).** Диапазону F6:F19 присвоить имя “отлично”, диапазону G6:G19 – “хорошо” и т.д.).
2. Сохранить макрос в личной папке.
3. Макросу назначить сочетание клавиш **Ctrl+a.**
4. Назначить макросу значок из списка и затем его отредактировать **(Изменить значок для кнопки).**
5. Внести в макрос изменения (Выполнить вертикальное и горизонтальное выравнивание всех заголовков таблицы по центру, обвести жирной линией шапку таблицы (№ п.п, ФИО…), залить ее желтым цветом.
6. Запустить макрос разными способами, создав копии таблицы на втором и третьем рабочем листах для разных дисциплин.
7. На четвертом рабочем листе создать списки фамилий двух групп. Номера групп занести в ячейки A1 и C1. Списки фамилий – в столбцы A и C, номера зачетных книжек – в столбцы B и D соответственно.
8. Отредактировать текст макроса: Отменить вывод строки состояния, защитить структуру рабочей книги, защитить рабочий лист4 паролем.
9. Заполнить таблицы: группу, дисциплину, и т. д. Предусмотреть автоматическое заполнение списка студентов и номеров зачеток в зависимости от номера группы по нажатию кнопки “заполнить список студентов” на рабочем листе.

	A	B	C	D	E	F	G	H	I	J
1	**Экзаменационная ведомость**									
2										
3	**Группа №**		**Дисциплина**							
4										
5	№ п.п	ФИО	№ зачетки	Оценка	Подпись экзаменатора	5	4	3	2	неявки
6						=ЕСЛИ(D6=5;1;0)	…	…	…	…

20										
21	От-лично	=СУММ(отлично)								
22	Хо-рошо	=СУММ(хорошо)								
23	Удовл.	=СУММ(удовлетвори-тельно)								
24	Не-удовл.	=СУММ(неудовлетвори-тельно)								
25	Неяв-ка	=СУММ(неявки)								
25	ИТО-ГО	=СУММ(С21:С25)								

1.2. ПРОГРАММИРОВАНИЕ НА VBA

ЛАБОРАТОРНАЯ РАБОТА № 2

ОСНОВНЫЕ СВЕДЕНИЯ

Структура редактора Редактор VBA активизируется командой **Сервис/Макрос/Редактор Visual Basic**. Возвратиться из редактора VBA в рабочую книгу можно нажатием кнопки **Вид/ Microsoft Excel.**

Интерфейс VBA состоит из следующих основных компонентов:

- окно проекта,
- окно свойств,
- окно редактирования кода, окно форм,
- меню и панели инструментов.

Окно проекта Окно проекта в редакторе VBA активизируется выбором команды **Вид/Окно проекта (View/ Project Explorer).** В окне проекта представлена иерархическая структура файлов форм и модулей текущего проекта.

В проекте автоматически создается модуль для каждого рабочего листа и для всей книги. Кроме того, модули создаются для каждой пользовательской формы, макросов и классов. По своему предназначению модули делятся на два типа: модули объектов и стандартные модули. К стандартным модулям относятся те, которые содержат макросы. К модулям объектов относятся модули, связанные с рабочей книгой, рабочими листами, формами, и модули класса.

Формы создаются командой **Insert/UserForm**, модули класса – командой **Insert/Module**). По мере создания, добавления и удаления файлов из проекта эти изменения отображаются в окне проекта. Удаление файла из окна проекта производится выбором значка файла с последующим выполнением команды **File/Remove...** В окне проекта выводится проект всех открытых рабочих книг.

Окно для редактирования кода

Перемещение указателя на значок файла в окне проекта и выполнение двойного щелчка кнопкой мыши открывает окно редактора кода для соответствующего модуля.

Окно редактирования кода служит в качестве редактора для ввода и изменения кода процедур приложения. Код внутри модуля организован в виде отдельных разде-

лов для каждого объекта, программируемого в модуле. В окне редактирования доступны два режима представления кода: просмотр отдельной процедуры и всего модуля. Переключение режимов работы окна редактирования кода осуществляется выбором одной из двух кнопок в нижнем левом углу окна редактирования кода, либо установкой или снятием флажка **Просмотр всего модуля** (**Default to Full Module View**) вкладки **Editor** диалогового окна **Options**, отображаемого на экране командой **Tools/Options**.

Два раскрывающихся списка в верхней части окна редактора кода облегчают ориентацию в процедурах. Левый раскрывающийся список позволяет выбрать управляющий элемент или форму, а правый – содержит список событий, допустимых для выбранного в левом списке объекта. При выборе элемента управления в форме посредством двойного щелчка или перемещении указателя на элемент управления и нажатии кнопки **View/Code** открывается окно редактирования кода как раз в том месте, где располагается процедура, связанная с этим элементом управления. Обратный переход от процедуры к объекту управления быстрее всего осуществить нажатием кнопки **View Object.**

При написании кода редактор сам предлагает пользователю список компонентов, логически завершающих вводимую пользователем инструкцию. Например, набирая код **Range("A1").**

после ввода точки на экране отобразится список компонентов, которые логически завершают данную инструкцию. Двойной щелчок на выбранном элементе из этого списка или нажатие клавиши <**Tab**> вставляет выбранное имя в код программы.

Автоматическое отображение списка компонентов происходит только при установленном флажке **Список компонентов** (**Auto list members**) вкладки **Editor**, окна **Options**, отображаемого на экране после выбора команды **Tools/Options**.

Список компонентов можно выводить на экран нажатием комбинации клавиш <**Ctrl>+<J**>, при этом список отображается как при установленном, так и при снятом флажке **Список компонентов**.

Редактор автоматически отображает на экране сведения о процедурах, функциях, свойствах и методах после набора их имени.

Для автоматического отображение на экране сведений о процедурах, функциях, свойствах и методах после ввода их имени следует выполнить команду **Tools/Options** активизировать вкладку **Editor** и установить флажок **Auto Quick Info.** Всплывающую подсказку можно выводить на экран нажатием комбинации клавиш <**Ctrl>+<I**>.

Редактор кода также производит автоматическую проверку синтаксиса набранной строки кода сразу после нажатия клавиши <**Enter**>. Если после набора строки и нажатия клавиши <**Enter**> строка выделяется красным цветом, это указывает на наличие синтаксической ошибки в набранной строке. Кроме того, если установлен флажок **Auto Sintax Check** вкладки **Editor,** помимо выделения красным цветом фрагмента кода с синтаксической ошибкой, на экране отображается диалоговое окно, поясняющее, какая возможная ошибка произошла. Если расположить курсор на ключевом слове языка VBA, имени процедуры, функции, свойства или метода и нажать клавишу <**F1**>, то на экране появится окно со справочной информацией об этой функции.

Окно редактирования форм (UserForm)

Форма в проект добавляется с помощью команды **Insert/ Form** или нажатием кнопки соответствующей кнопки.

Используя панель инструментов **Панель элементов** из незаполненной формы, можно сконструировать любое требуемое для приложения диалоговое окно. Для облегчения размещения и выравнивания элементов управления используется сетка. Активизировать ее можно с помощью вкладки **General** диалогового окна, вызываемого командой **Tools/Options**, там же устанавливается шаг сетки. Кроме того, команды меню

Format автоматизируют и облегчают процесс выравнивания элементов управления по их взаимному местоположению, так и по размерам.

Окно **Просмотр объектов (Object Browser)**

Окно **Просмотр объектов** вызывается командой **View/Object Browser** или нажатием соответствующей кнопки. В этом окне приведен список всех объектов, которые имеются в системе и которые можно использовать при создании проекта, и их методов.

Окно **Просмотр объектов** состоит из трех основных частей:

1. Раскрывающегося списка **Project/Library** в левом верхнем углу окна. В этом раскрывающемся списке можно выбрать различные проекты и библиотеки объектов. В частности, библиотеки объектов Excel, VBA, Office и VBAProject; (объекты пользовательского проекта). Выбор в списке строки **All Libraries** отображает список объектов всех библиотек.

2. Списка **Классы** (**Classes**). После выбора из раскрывающегося списка (**Project/ Library**) просматриваемой библиотеки, например VBA, все классы объектов выбранной библиотеки выводятся в списке **Classes**

3. Списка **Компоненты** (**Members**). После выбора класса из списка **Классы** просматриваемой библиотеки все компоненты выбранного класса выводятся в списке Компоненты (**Member)s**. При выделении строки в этом списке в нижней части окна **Просмотр объектов** приводится дополнительная информация о выбранном компоненте. Кроме того, если нажать на кнопку <**Help>**, расположенную на панели инструментов в правой верхней части окна **Просмотр объектов**, то на экране отобразится окно Справочник Visual Basic с подробной информацией о выделенном компоненте.

Вывод значений свойств и переменных

Одним из наиболее удобных средств режима отладки является возможность узнать текущее значение переменных и свойств. Для этого достаточно поставить указатель мыши на имени свойства или переменной. Это вызовет появление всплывающей подсказки с текущим значением переменной свойства. Для установки режима вывода всплывающей подсказки с текущими значениями данных должен быть установлен флажок **Подсказки значений данных** (**Auto Data Tips**) диалогового окна, вызываемого командой **Tools, Options**.

Другим способом отслеживания текущих значений данных является использование диалогового окна **Контрольные значения** (**Quick Watch**), отображаемого на экране с помощью команды **View/ Quick Watch.** Диалоговое окно **Quick Watch** применяется для одновременного отображения текущих значений нескольких переменных. Команда **Debug/Add Watch** позволяет добавить новые контрольные значения в диалоговом окне **Quick Watch.**

Удаление контрольного значения из диалогового окна производится посредством его выделения и нажатия клавиши <**Delete**>.

Окно Локальные переменные (**Locals Window**), отображаемое на экране командой **View/Locals Window**, выводит значения всех переменных текущей процедуры, а не только специально выбранных.

Окно **Проверка** (**Immediate Window**), отображаемое на экране командой **View/Immediate Window**, предоставляет пользователю возможность:

- Набирать и вычислять отдельные инструкции VBA. Для этого достаточно в окне **Immediate Window** ввести соответствующую инструкцию и нажать клавишу <**Enter**>. Единственным ограничением на использование инструкции является то, что она должна быть набрана в одну строку. Например,

```
s=0:  For i=1 to  5: s=s+i^2:  Next i :  MsgBox s
```

- Определять текущие значения переменных и свойств. Для этого в окне **Immediate Window** надо набрать вопросительный знак, имя переменной или свойства и нажать

клавишу <**Enter**>. Например, ?x

- Устанавливать новые текущие значения переменных. Для этого в окне **Immediate Window** надо набрать имя переменной, знак равенства и новое значение переменной:x=10

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какой командой активизируется окно VBA?
2. Какую команду необходимо выполнить для создания формы?
3. Как удалить файл из окна проекта?
4. Как установить режим просмотра отдельной процедуры в окне редактирования кода?
5. Какой элемент окна редактирования кода позволяет выбрать управляющий элемент или форму?
6. Какое сочетание клавиш обеспечивает вывод списка компонентов, логически завершающих вводимую пользователем инструкцию, если флажок **Auto list members** не установлен?
7. Какую команду следует выполнить для отображения сведений о процедурах, функциях, свойствах и методах после ввода их имени?
8. Какая команда открывает окно просмотра компонентов выбранного класса?
9. Какими способами можно отследить текущие значения данных в процессе выполнения процедуры?
10. Для чего служить окно **Immediate Window?**

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Рассчитать сумму, которая должна быть возвращена банку при следующих исходных данных:

- Размер полученной ссуды;
- Дата выдачи ссуды;
- Дата возврата ссуды;
- Процентная ставка банка.

1. Присвойте листу рабочей книги имя **Расчет ссуды** и создайте на нем следующую таблицу:
2. Создайте диалоговое окно **Расчет ссуды** (см. Рисунок 2).

	A	B
1	**Расчет краткосрочной ссуды**	
2	Размер ссуды	
3	Дата выдачи ссуды	
4	Дата возврата ссуды	
5	Процентная ставка	
6	Возвращаемая сумма	
7		

Рисунок 1.

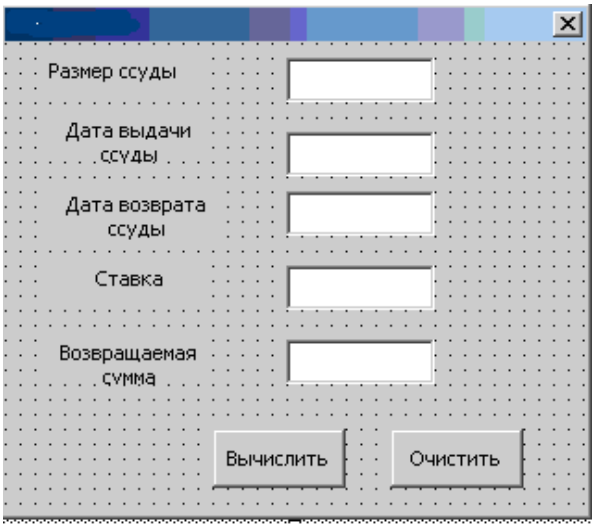


Рисунок 2.

3. Создайте процедуры, которые обеспечивают:

a) проверку всех данных, вводимых в диалоговое окно (правильность ввода дат, числовых данных, наличие в полях всех необходимых данных для расчета);
b) вычисление возвращаемой суммы по формуле:

Возвращаемая сумма = Размер ссуды*(1+ставка)^((Дата возврата-Дата выдачи) /365);

c) передачу данных из диалогового окна в таблицу Excel;
d) очистку полей диалогового окна и полученных данных в таблице на рабочем листе (Полям формы присвоить значения пустых строк, к диапазону рассчитываемых данных рабочего листа применить метод **Clear**.

4. Создайте на рабочем листе кнопку "Рассчитать ссуду", которая будет выводить на экран диалоговое окно для расчета ссуды по нажатию кнопки. Для создания процедуры обработки события в контекстном меню кнопки выбрать **Исходный текст.** В окне редактора создать процедуру, открывающую диалоговое окно: ***DialogSheets("Ссуда"). Show.*** Протестировать работу приложения.

5. Переключите окно редактирование кода в режим просмотра отдельной процедуры. Вернитесь в режим просмотра всего модуля.

6. Перейдите в режим, не предусматривающий вывод списка компонентов для логического завершения вводимых инструкций. Вернитесь в режим отображения этих компонентов.

8. Установите режим всплывающей подсказки с текущим значением переменной.

9. Выведите окно для отображения всех переменных процедуры.

10. В окне **Immediate Window** выполните вычисление:

s=0: For i=1 to 5; s=s+i^2: Next i : MsgBox s

1.3. СОЗДАНИЕ ПРИЛОЖЕНИЙ

ЛАБОРАТОРНАЯ РАБОТА № 3

ОСНОВНЫЕ СВЕДЕНИЯ

Элемент управления **ListBox** (список) создается с помощью кнопки **Список.** Элемент управления **ListBox** применяется для хранения списка значений. Из списка пользователь может выбрать одно или несколько значений, которые в последующем будут использоваться в тексте ***программы***.

Элемент управления ***ListBox*** (список) создается с помощью кнопки ***Список.*** Элемент управления ***ListBox*** применяется для хранения списка значений. Из списка пользователь может выбрать одно или несколько значений, которые в дальнейшем будут использоваться в тексте программы.

Свойства элемента управления ListBox

ListIndex	Возвращает номер текущего элемента списка. Нумерация элементов списка начинается с нуля
ListCount	Возвращает число элементов списка
Topindex	Возвращает элемент списка с наибольшим номером
ColumnCount	Устанавливает число столбцов в списке
TextCount	Устанавливает столбец в списке, элемент которого возвращается свойством ***Text***
Enabled	Допустимые значения: true (запрещен выбор значений из списка пользователем) и false (в противном случае)
Text	Возвращает выбранный в списке элемент

List	Возвращает элемент списка, стоящий на пересечении указанных строки и столбца. Синтаксис: ***List(Row, Column)***
RowSource	Устанавливает диапазон, содержащий элементы списка
ControlSource	Устанавливает диапазон (ячейку), куда возвращается выбранный элемент из списка
MultiSelect	Устанавливает способ выбора элементов списка. Допустимые значения: • ***fmMultiSelectSingle***(выбор только одного элемента); • ***fmMultiSelectMulti*** (разрешен выбор нескольких элементов посредством либо щелчка, либо нажатие клавиши <Пробел>), • ***fmMultiSelectExtended*** (разрешено использование клавиши <Shift> при выборе ряда последовательны элементов списка)
Selected	Допустимые значения: ***True***(если элемент списка выбран) и ***False*** (в противном случае). Используется для определения выделенного текста, когда свойство ***MultiSelect***: имеет значение ***fmMultiSelect-Multi*** или ***fmMultiSelectExtended***
Column Widths	Устанавливает ширину столбцов списка. Синтаксис: ***ColumnWidths = String*** ***String*** – строка, устанавливающая ширину столбце. ***With ListBox1*** ***.ColumnCount = 3*** ***.ColumnWidths= "20,30,30"*** End With
ColumnHeads	Допустимые значения: ***True*** (выводятся заголовки столбцов раскрывающегося списка) и ***False*** (в против ном случае)
ListStyle	Допустимые значения: • ***fmListStylePlain*** (выбранный элемент из списка выделяется цветом)
	• ***fmListStyleOption (***перед каждым элементом в списке располагается флажок и выбор элемента из списка соответствует установке этого флажка):
MatchEntry	Выводит первый подходящий элемент из списка при наборе его имени на клавиатуре. Допустимые значения: • f***mMatchEntryNone***(режим вывода подходящего элемента в списке отключен); • ***fmMatchEntryFirstLetter*** **(выводит подходящий элемент** по набранной первой букве. В этом случае, предпочтительно, чтобы элементы списка были бы упорядочены в алфавитном порядке) • ***fmMatchEntryComlete*** **(выводит подходящий элемент** по полному набранному имени)
BoundColumn	Устанавливает тип, возвращаемый свойством ***Value***. А именно, • если свойство ***BoundColumn*** равно ***0***, то свойство ***Value*** возвращает индекс выбранной строки, т. е. в этом случае оно действует как свойство ***ListIndex*** • если свойство ***BoundColumn*** принимает значение из диапазона от 1 до количества столбцов в списке, то свойство ***Value*** возвращает элемент из выбранной строки, стоящий в столбце, определенном свойством ***BoundColumn***

Методы элемента управления ListBox

Clear	Удаляет все элементы из списка
RemoveItem	Удаляет и списка элемент с указанным номером. **Синтаксис** ***RemoveItem (index***) **index** – номер удаляемого из списка элемента
AddItem	Добавляет элемент в список. Синтаксис**: *AddItem*** ([***item ,varIndex***]]) ***Item*** – элемент(строковое выражение, добавляемое в список; ***varIndex*** – номер добавляемого элемента;

Заполнение списка

Заполнить список можно одним из следующих способов:

1. Поэлементно, если список состоит из одной колонки
2. Массивом, если список состоит из одной колонки
3. Из диапазона, в который предварительно введены элементы списка. Результат выбора (индекс выбранной строки) выводится в ячейку.

Выбор нескольких элементов из списка

Свойство **MultiSelect** позволяет устанавливать режим, при котором допустим выбор нескольких элементов из списка. Свойство **Selected** предоставляет возможность проверить, выбран ли элемент с указанным индексом.

Рассмотрим пример использования этих свойств при вычислении среднего значения выбранных в списке элементов.

```
With ListBox1
.List = Array(1,  3,  4,  5,  6,  7,  8,  9)
.ListIndex =  0
.MultiSelect = .fmMultiSelectMulti
Среднее = 0
 n = 0
For i =  0  To  .ListCount
     If  .Selected(i)  = True  Then
n= n +  1
Среднее = Среднее +  . List (i)
          End If
    Next i
End With
Среднее = Среднее / n
```

Поле со списком

Элемент управления **ComboBox** сочетает в себе функциональные возможности списка **ListBox** и поля **TextBox**. В отличие от поля **ListBox** в элементе управления **ComboBox** отображается только один элемент списка. Кроме того, у него отсутствует режим выделения нескольких элементов. Также как в поле **TextBox,** в поле **ComboBox** можно вводить значение. Свойства объекта **ComboBox**, такие как **ListIndex**, **ListCount**, **Enabled**, **List**, методы **Clear, RemoveItem** и **AfddItem** аналогичны соответствующим свойствам и методам списка **ListBox**. Элемент управления **ComboBox** имеет ряд уникальных свойств.

DropButtonStyle	Устанавливает вид раскрывающегося списка. Допустимые значения: ***fmDropButtonStylePlain*** (кнопка без символов; ***fmDropButtonStyleArrowDisplays*** (кнопка со стрелкой); ***fmDropButtonStyleEllipsis*** (кнопка с эллипсом); ***fmDropButtonStyleReduce*** (кнопка с линией)
ListRows	Устанавливает число элементов, отображаемых в раскрывающемся списке
MatchRequired	Допустимые значения: ***True*** (в поле ввода раскрывающегося списка нельзя ввести значения, отличные от перечисленных в списке, т.е. в поле со спискам 1 отключается функция поля ввода) и ***False*** (в противном случае)
MatchFound	Допустимые значения: **True**(среди элементов раскрывающегося списка имеется элемент, совпадающий с вводимым в поле ввода списка) и **False** (в противном случае)

Значения, введенные в поле со списком, обычно, передаются в таблицы на рабочих листах. Для определения номера строки, в которую следует передать очередное значение, можно воспользоваться свойством **CurrentRegion** (максимальная область, окруженная комбинацией пустых строк и столбцов и содержащая заданный диапазон) и вычислить количество строк в этой области.

Присвоим, например, переменной x значение числа строк в текущем диапазоне, содержащем ячейку A1:

X= Range(“A1”). CurrentRegion.Rows.Count

Согласованная работа двух списков

Часто бывает необходимо обеспечить согласованную работу двух списков. Например, некоторое издательство сотрудничает с определенными магазинами в разных городах. В один список вводится список городов. При выборе элемента из этого списка во втором списке отображаются только магазины, находящиеся в этом городе.

Для реализации этой задачи список магазинов будем хранить в массиве, элементы которого имеют тип **Variant**, а значением переменной типа **Variant** могут быть данные любых типов, в том числе и массивы.

В результате выполнения следующего кода список городов создается из элементов массива **Город,** а список магазинов из массива с индексом, зависящим от индекса города в списке **Город.**

```
Private Sub UserForm2_Init()
Dim Город As Variant
Город= Array("Минск", "Витебск")
UserForm2.ListBox1.List = Город
UserForm2.ListBox1.ListIndex = 0
UserForm2.Show
End Sub
Private Sub ListBox1_Click()
Dim Магазин (2) As Variant
Магазин(0) = Array("Дом книги", "Мир", "Техническая книга")
```

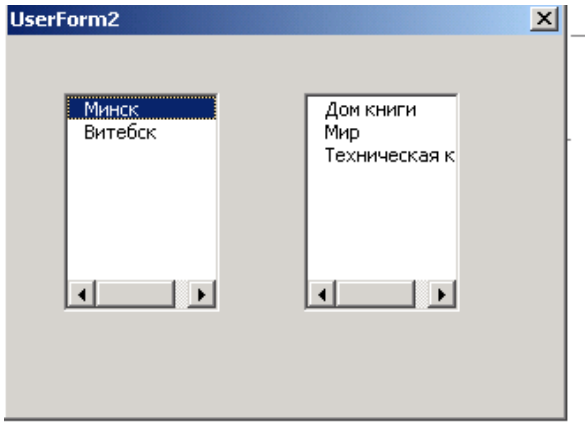


Рисунок 1.

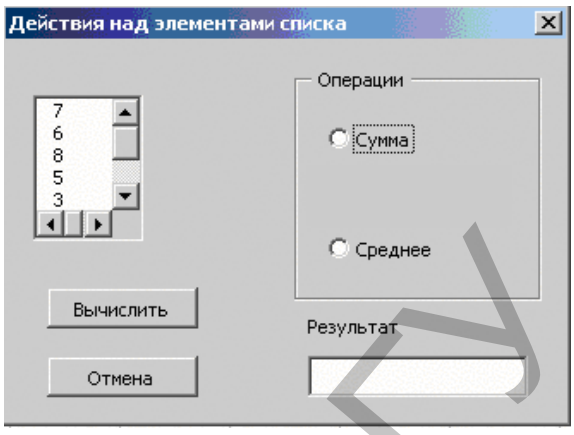


Рисунок 3.

Магазин(1) = Array("Глобус", "Светач", "Книги")
UserForm2.ListBox2.Clear
ListBox2.List = Магазин(ListBox1.ListIndex)
End Sub

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие элементы управления форм служат для создания списков?
2. Какое свойство элемента управления **ListBox** возвращает номер текущего элемента списка?
3. Какое свойство элемента управления **ListBox** устанавливает число столбцов в списке?
4. Какими способами список может быть заполнен элементами?
5. Какое свойство элемента управления **ListBox** устанавливает возможность выбора одного или нескольких элементов из списка?
6. В чем заключаются отличия элементов управления **ListBox** и **ComboBox**?

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Задание 1.Создать приложение, которое позволит выбрать несколько чисел, выводимых в списке в диалоговом окне **Действия над элементами списка (**Рисунок 1**.**).

Для разработки приложения необходимо:

1. Создать диалоговое окно **Действия над элементами списка** (см. рис.2).
2. Создать процедуру инициализации диалогового окна, предусматривающую:

- заполнение списка целыми числами с использованием массива;
- возможность выбора нескольких элементов из списка;
- активизацию первого элемента списка (Свойство **ListIndex**);
- первоначальный выбор переключателя "**Сумма**" при инициализации диалогового окна;

Разработать процедуру, запускаемую по нажатию на кнопку ***Вычислить***. Процедура должна определять, какой из переключателей (***Сумма***, или ***Среднее***) выбран, и выполнять необходимое действие над выбранными в списке элементами. Найденное число должно выводиться в поле ***Результат***.

Предусмотреть закрытие диалогового окна при нажатии на кнопку ***Отмена***.

Задание 2. Создать приложение для нахождениясуммарной нагрузки преподавателей, выбранных из списка в диалоговом окне, приведенном на **Ошибка! Источник ссылки не найден.**.

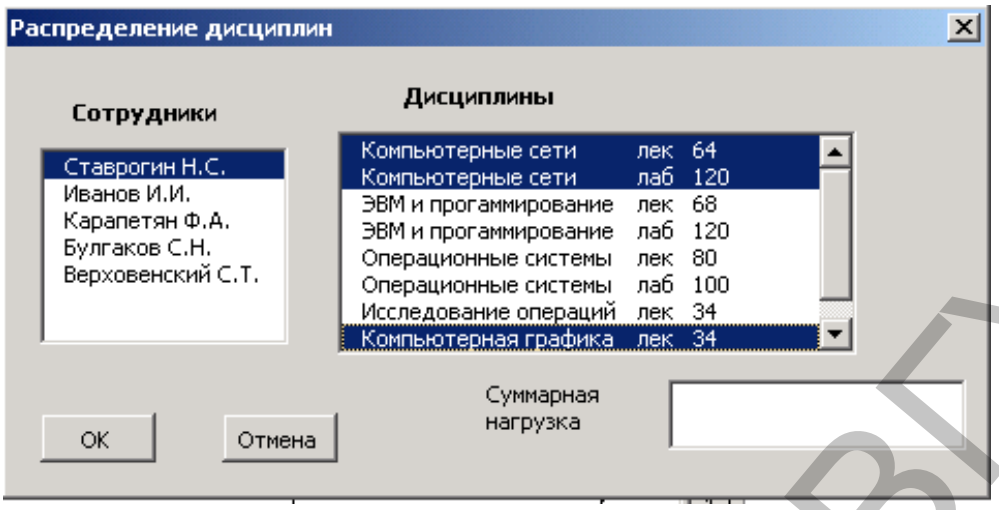


Рисунок 4.

Для создания приложения необходимо:

1. На листе **Преподаватели** в диапазон A1:A10 ввести список преподавателей.
2. На рабочем листе **Дисциплины** создать список дисциплин (см. табл.1).

Таблица 1

Дисциплины	**Вид занятий**	**Часы**
Компьютерные сети	лек	64
Компьютерные сети	лаб	120
ЭВМ и программирование	лек	68
ЭВМ и программирование	лаб	120
Операционные системы	лек	80
Операционные системы	лаб	100
Исследование операций	лек	34
Компьютерная графика	лек	34
Модели данных и СУБД	лек	34

3. На панели **инструментов** создать кнопку **Расчет нагрузки** для открытия диалогового окна.
4. Создать диалоговое окно.
5. **Разработать** процедуру инициализации формы (заполнение списков). Списки должны заполняться из диапазонов соответствующих рабочих листов с помощью свойства ***RowSource*** элемента управления ***ListBox***. Поскольку списки могут иметь переменный размер, сначала необходимо выполнить идентификацию диапазона с помощью свойства **CurrentRegion,** а затем определить адрес диапазона с помощью свойства ***Address*****:**

   Set r =Workrksheets("Сотрудники").Range("a1").CurrentRegion
   UserForm1.ListBox1.RowSource = r. Address (RowAbsolute:=False, ColumnAbsolute:=False, external:=True)

   Значению параметра ***external*** присваивается значение ***True,*** в том случае, когда диапазон, содержащий список, находится на неактивном рабочем листе.

   Обеспечить возможность выбора нескольких элементов из списка дисциплин.
6. Для отображения окна формы используется метод ***Show (UserForm.Show).***
7. Разработать процедуру, **запускаемую** по нажатию на кнопку **OK** формы**.** Процедура должна выполнять суммирование часов для выбранных для преподавателя дисциплин. Найденная сумма должна выводиться в поле **Результат**.

Предусмотреть:

- запись нагрузки преподавателя (ФИО, дисциплина, вид занятий, количество часов) в таблицу рабочего листа **Нагрузка.**. Необходимо определить номер первой незаполненной строки и записать в ее ячейки данные из полей формы **(**найти текущий диапазон с помощью свойства ***CurrentRegion***, вычислить количество строк в диапазоне: ***<имя диапазона>.Rows.Count)***.
- отображение полученной суммарной нагрузки в столбце **В** на рабочем листе **Преподаватели**;
- назначение клавише <***Enter***> функции кнопки ***OK*** (значение свойства ***Default = True***);
- назначение клавише <***Esc***> функции кнопки ***Отмена*** (значение свойства ***Cancel = True***.);
- проверку, являются ли значения в столбце нагрузки числами;
- проверку выделения хотя бы одного элемента в каждом списке.

8. **Обеспечить** очистку полученных результатов при нажатии на кнопку **Отмена.**

Задание 3. Создать форму для распределения дисциплин между преподавателями кафедры. Список преподавателей приведен на рабочем листе **Преподаватели**, список дисциплин – на рабочем листе **Дисциплины**, Распределение дисциплин - на рабочем листе **Распределение дисциплин**.

Заполнение списка на листе **Распределение дисциплин** выполнить с помощью диалогового окна “**Распределение дисциплин” (**см. **Ошибка! Источник ссылки не найден.**)При нажатии кнопки **OK** выбранные из списков данные (сотрудник и

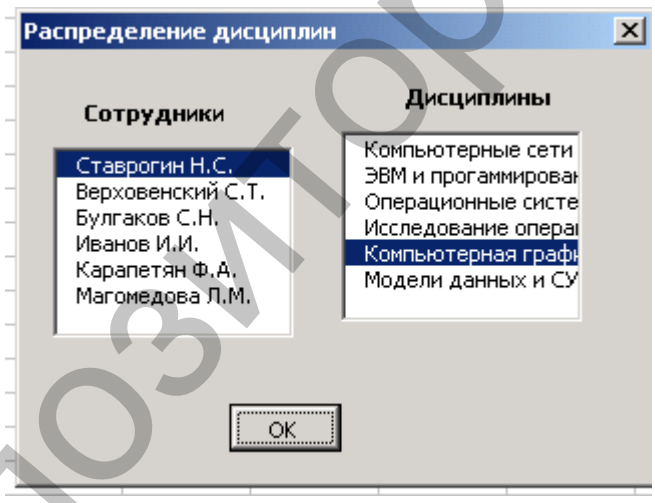


Рисунок 5.

дисциплина) вводятся в очередную строку списка, расположенного на листе **Распределение нагрузки..** Предусмотреть обработку ситуации, когда пользователь повторно выбирает одну и ту же дисциплину для одного и того же преподавателя

Для реализации поставленной задачи необходимо:

9. На листе **Преподаватели** в диапазон A1:A10 ввести список преподавателей.
10. На листе **Дисциплины** ввести список дисциплин.
11. На листе **Распределение дисциплин** в ячейку A1 ввести строку «Преподаватели”, а в ячейку B1 – “Дисциплины”
12. Создать форму (рисунок 4).
13. Для открытия окна формы создать кнопку на панели инструментов **Стандартная.**
14. Разработать процедуру обработки нажатия кнопки **OK** формы. В процедуре предусмотреть:

- Проверку, были ли выбраны данные из списков (значение свойства ***ListIndex*** элемента управления ***ListBox1*** должно быть неотрицательным).
- Определение первой свободной строки в списке рабочего листа **Распределение дисциплин (**найти текущий диапазон с помощью свойства ***CurrentRegion***, вычислить количество строк в диапазоне: ***<имя диапазона>.Rows.Count)***
- Активизацию рабочего листа **Распределение дисциплин** с помощью метода **Activate.**
- Ввод в нужные ячейки значений из списков.

2. МОДЕЛИ И БАЗЫ ДАННЫХ

Цель: данный раздел посвящен усвоению основных приемов проектирования, создания и обработки баз данных.

2.1. ЭТАПЫ ПРОЕКТИРОВАНИЯ БАЗ ДАННЫХ. НАЗНАЧЕНИЕ И ВОЗМОЖНОСТИ MS ACCESS.

ЛАБОРАТОРНАЯ РАБОТА № 4

ОСНОВНЫЕ СВЕДЕНИЯ

Этапы разработки базы данных. Формирование исходного отношения.

Проектирование базы данных начинается с определения всех объектов, сведения о которых будут включены в базу и определения их атрибутов. Затем атрибуты сводятся в одну таблицу.

Рассмотрим пример работы предприятия, занимающегося прокатом компакт дисков. Для того чтобы знать, кто какой диск взял, когда должен возвратить, сколько дисков каждого наименования осталось на складе, предприятию необходима БД. На первом этапе следует определить и свести в одну таблицу атрибуты базы данных.

Прокат дисков (Диск, Стиль, Залоговая стоимость, Количество, Наличие, Дата_выдачи, На_сколько_дней, Возврат, Сумма оплаты; ФИО, Шифр, Адрес, №телефона, №паспорта). Атрибут **Шифр** служит для однозначной идентификации клиента проката.

Очевидно, что исходное отношение содержит избыточное дублирование данных. Явная избыточность заключается в том, что строки с данными о дисках, выдаваемых неоднократно, будут повторяться соответствующее число раз, данные о клиентах проката – адрес, номер паспорта также будут повторяться для каждого диска. Если изменится залоговая стоимость или номер телефона клиента проката, этот факт должен быть отражен во всех записях. В противном случае будет иметь место противоречие в данных, что является примером аномалии редактирования, обусловленной явной избыточностью данных в отношении.

Средством исключения избыточности данных является их нормализация. Пользователю базы данных удобнее иметь несколько таблиц, связанных между собой. Основные цели нормализации БД:

- обеспечение быстрого доступа к данным;
- исключение избыточного дублирования данных;
- обеспечение целостности данных с тем, чтобы, чтобы при изменении одних объектов автоматически происходило изменение всех связанных с ними объектов.

Метод нормализации отношений основан на фундаментальном в теории реляционных БД понятии зависимости между атрибутами отношений.

Виды зависимостей между атрибутами.

Основные виды зависимостей между атрибутами отношений: функциональные,

многозначные, транзитивные.
Понятие функциональной зависимости является базовым, т.к. на его основе формулируются определения всех остальных видов зависимостей.
Атрибут B функционально зависит от атрибута A, если каждому значению A соответствует в точности одно значение: A→B.

Это означает, что во всех кортежах с одинаковым значением атрибута A атрибут B будет иметь также одно и то же значение. Примеры функциональных зависимостей в отношении **Прокат дисков:** Диск → Залоговая стоимость.
Атрибуты A и B могут быть составными, то есть состоять из двух и более атрибутов. Набор атрибутов аргументов, однозначно определяющих набор атрибутов функций, называется ключом отношения. В рассматриваемом примере ключ составной, в него входят атрибуты: **Диск**; **Шифр**; **Дата_выдачи.**
Частичной функциональной зависимостью называют зависимость неключевого атрибута от части составного ключа. Так атрибуты **Количество**, Наличие зависят только от атрибута **Диск.**
В случае полной функциональная зависимости неключевые атрибуты находятся в зависимости от всего составного ключа. В рассматриваемом примере в полной функциональной зависимости от ключа находится атрибут Возврат.Атрибут C зависит от атрибута A транзитивно, если для атрибутов A,B,C выполняется A→B и B→C, но обратная зависимость отсутствует. Между атрибутами может быть многозначная зависимость. В отношении R атрибут B многозначно зависит от атрибута A, если каждому значению A соответствует множество значений B, не связанных с др. атрибутами из R. Многозначные зависимости могут быть «один-ко-многим»(1:M), многие к одному(M:1) или многие ко многим(M:M).

Нормальные формы

Теория нормализации оперирует 5 нормальными формами таблиц. Эти формы предназначены для уменьшения избыточности информации. Каждая последующая форма должна удовлетворять требованиям предыдущей и некоторым дополнительным условиям. Основной операции при проектировании базы данных методом нормализации является проекция, служащая для разложения одного отношения на несколько отношений.

Первая нормальная форма.

Отношение находится в первой нормальной форме, если все ее атрибуты являются простыми, т.е. имеют единственное значение. Исходное отношение строится таким образом, чтобы оно было в первой нормальной форме.

Вторая нормальная форма.

Отношение находится во второй нормальной форме, если оно находится в первой нормальной форме, и каждый неключевой атрибут функционально полно зависит от первичного ключа (составного).
Исходное отношение **Прокат дисков** имеет составной ключ. Неключевые атрибуты **Стиль**, **Залоговая стоимость, Количество, Наличие** находятся в функциональной зависимости от части составного ключа - атрибута **Диск**, а атрибуты **Адрес; №телефона; №паспорта** находятся в зависимости от атрибута **Шифр**, который также является частью составного ключа. Поэтому необходимо выделить данные о клиентах и дисках в отдельные таблицы, то есть таблицу необходимо разбить на три таблицы. Первая таблица будет содержать данные о клиентах, а вторая – о дисках на складе, третья – о выдаче дисков клиентам**.**

Отношение **Клиенты**

ФИО	Шифр	Адрес	№телефона	№паспорта

Отношение **Прокат**

Шифр	Диск	Дата_выдачи	На сколько дней	Возврат	Сумма оплаты

Отношение **Склад**

Диск	Стиль	Залоговая стоимость	Количество	Наличие

Для связывания отношений **Склад** и **Прокат** используется атрибут **Диск**. Между отношениями будет установлена связь типа (1:M).

Отношения **Клиент и Прокат** связаны атрибутом **Шифр.**

Все отношения, имеющие простой ключ, автоматически удовлетворяют требованию второй нормальной формы.

Третья нормальная форма.

Понятие 3-ей норм формы основывается на понятии нетранзитивной зависимости. (Транзитивная зависимость наблюдается в том случае, если один из 2-х неключевых атрибутов зависит от ключа, а другой неключевой атрибут зависит от первого неключевого атрибута). Отношение находится в третьей нормальной форме, если оно находится во второй нормальной форме и каждый неключевой атрибут нетранзитивно зависит от первичного ключа.

Сведение отношения к третьей нормальной форме предполагает разделение отношения с целью помещения в отдельную таблицу атрибутов, которые не зависят от ключа, а зависят от другого неключевого атрибута. В рассматриваемом примере транзитивные зависимости отсутствуют, поэтому процесс проектирования завершается на втором этапе. На практике в большинстве случаев бывает достаточно построения схем отношений, приведенных к третьей нормальной форме.

Архитектура MS Access

Основными объектами MS Access являются таблицы, запросы, формы, отчеты, макросы и модули. В отличие от других СУБД, где термин БД обычно относится только к файлам, в которых хранятся данные, в Access БД включает в себя все объекты, связанные с хранимыми данными, в том числе и те, которые используются для автоматизации работы с базой данных.

Определение основных объектов БД.

Таблица Объект, используется для хранения данных. Каждая таблица содержит информацию о предметах или субъектах определенного типа. Для каждой таблицы можно определить первичный ключ (одно или несколько полей, имеющих уникальные для каждой записи значения) и один или несколько индексов, ускоряющих доступ к данным.

Запрос Объект, позволяющий пользователю получить нужные данные из одной или нескольких таблиц. Для определения запроса можно использовать бланк QBE или написать инструкцию SQL. Можно создать запросы на выборку, обновление, удаление или добавление данных. С помощью запросов можно создавать новые таблицы, используя данные из одной или нескольких существующих таблиц.

Форма Объект, предназначенный в основном для ввода данных, отображения их на экране или управления работой приложения. Формы можно использовать для того, чтобы реализовать требования пользователя к представлению данных таблиц или наборов записей запросов. С помощью форм можно в ответ на некоторое событие запустить макрос или процедуру VBA.

Отчет Объект, предназначенный для форматирования, вычисления итогов и печати выбранных данных.

Макрос Объект, представляющий собой структурированное описание одного или нескольких действий, которые должен выполнить Access в ответ на некоторое событие. Например, можно определить макрос, который при выборе некоторого элемента в основной форме открывает другую форму. С помощью макроса можно осуществлять проверку значения некоторого поля при

изменении его содержимого. В макрос можно включить дополнительные условия для выполнения или пропуска тех или иных указанных в нем действий. Макросы можно использовать для открытия таблиц, выполнения запросов, просмотра или печати отчетов.

Модуль Объект, содержащий программы на языке Visual Basic для приложений. Модули могут быть независимыми объектами, содержащими процедуры, вызываемые из любого места приложения, или непосредственно привязанными к формам или отчетам для реакции на те или иные события.

В стартовом окне базы данных все перечисленные объекты представлены вкладками. Кроме вкладок окно содержит три командные кнопки: ***Открыть, Конструктор*** **и** ***Создать.***

Кнопка ***Открыть*** открывает избранный объект. Если открыта таблица, то ее можно просмотреть, добавить новые записи, изменить данные, введенные ранее. Если открыта форма, можно просматривать и редактирование данные с помощью формы. Если выбрана вкладка **Запросы**, режим открытия позволяет просматривать и в некоторых случаях редактировать данные, отобранные запросом. Если выбран запрос на изменение, нажатие этой кнопки запускает его.

Кнопка ***Конструктор*** открывает структуру выбранного объекта и позволяет исправлять ее. Этот режим, как правило, является инструментом разработчика, а не пользователя.

Кнопка ***Создать*** служит для создания новых объектов. Этот элемент также предназначен для проектировщиков БД. Создание объектов может быть выполнено разными способами: вручную, автоматически или с помощью **Мастера.**

КОНТРОЛЬНЫЕ ВОПРОСЫ.

1. С чего начинается проектирование базы данных?
2. Перечислите основные цели нормализации базы данных.
3. Какие существуют виды зависимостей между атрибутами отношений?
4. В чем заключается сущность функциональной зависимости между атрибутами отношений? Приведите примеры функциональных зависимостей атрибутов, входящих в отношение **Прокат дисков.**
5. Дайте определение ключа отношения.
6. Приведите примеры полной и неполной функциональной зависимостей.
7. Какая зависимость между атрибутами называется транзитивной?
8. Какие виды многозначной зависимости Вам известны?
9. С какой целью выполняется нормализация отношений?
10. В чем заключается сущность метода нормализации?

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ.

Выполнить проектирование базы данных; привести ее к третьей нормальной форме. Исходные данные находятся в приложении 1.

2.2. СОЗДАНИЕ И УПРАВЛЕНИЕ БД В MS ACCESS.

ЛАБОРАТОРНАЯ РАБОТА № 5

Таблицы При первом открытии окна БД Access всегда активизирует вкладку **Таблицы** и выводит на экран список таблиц БД. Создание таблицы заключается в задании ее полей и определении их свойств. Существуют разные способы создания новой таблицы, отличающиеся уровнем автоматизации:

- Импорт таблицы из другой базы;
- Использование **Мастера таблиц**;

- В режиме **Связь с таблицами** в случае, если таблица находится на удаленном сервере и ее нельзя импортировать целиком.
- В режиме таблицы;
- В режиме **Конструктор.**

Таблицы БД не являются самостоятельными документами. Документ - вся БД, которой соответствует файл на диске.

Следует иметь в виду, что все изменения, вносимые в таблицу, сохраняются автоматически в режиме реального времени, т.е. при работе с таблицей происходит ее непрерывное сохранение, данные, введенные для одного поля немедленно записывается на диск.

На рисунке 1 представлена таблица БД.

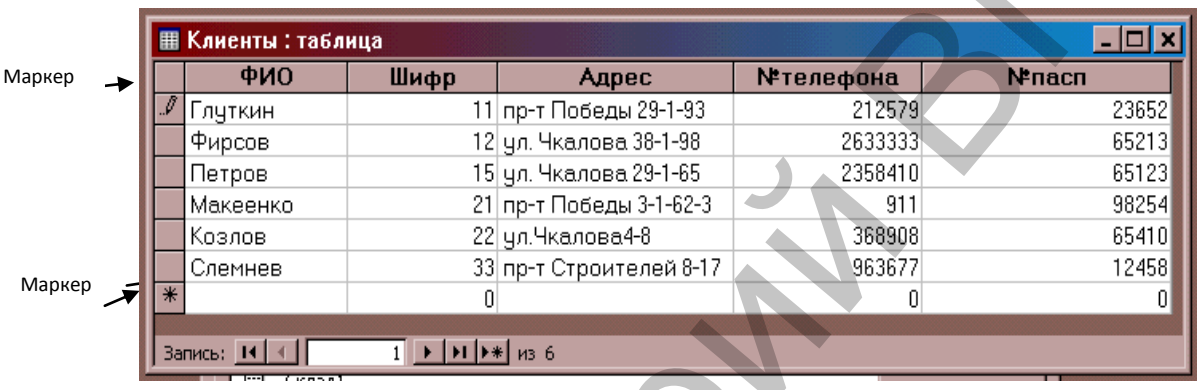


Рисунок 1.

Каждая запись имеет слева кнопку – маркер записи. Щелчок на маркере выделяет запись. Щелчок правой кнопкой на выделенной записи открывает контекстное меню, содержащее перечень возможных операций над записью.

Щелчок по маркеру таблицы (в верхнем левом углу) выделяет всю таблицу, щелчок правой кнопкой по выделенной таблице открывает контекстное меню, относящееся ко всей таблице.

Щелчок на заголовке столбца таблицы выделяет весь столбец, а щелчок правой кнопкой на выделенном столбце открывает контекстное меню, в котором присутствуют команды сортировки таблицы по возрастанию или убыванию значений столбца, скрытие столбца (скрытый столбец можно вернуть двойным щелчком на границе столбцов в месте его скрытия) и др.

Создание таблицы БД в режиме Конструктор.

В окне СУБД MS Access, открывающемся после загрузки приложения включить переключатель **Новая база данных.**

1. Ввести имя файла базы данных, указать место его сохранения и щелкнуть по кнопке **Создать.**
2. В следующем окне выбрать вкладку **Таблица** и щелкнуть по кнопке **Создать.**
3. В открывшемся диалоговом окне щелкнуть по кнопке **Конструктор.**
4. В окне Конструктора задать атрибуты отношения и типы данных, задать маски для ввода данных, значение по умолчанию, условия на значения для контроля данных в процессе их ввода, тексты сообщений в случае ошибочного ввода данных. Ввести описания полей. указать ключ, сохранить таблицу. Редактирование структуры таблицы осуществляется в режиме **Конструктор**. Для вставки атрибута следует установить указатель на маркер следующего атрибута и нажать **Insert**. Для удаления – выделить атрибут и нажать **Delete**. Для перемещения атрибутов -

выделить атрибут, установив курсор мыши на маркер записи и перетащить мышью к месту вставки.

Создание таблицы в режиме таблицы

1. При создании таблицы в режиме таблицы выводится пустая таблица, поля которой не определены и имеют названия Поле1, Поле 2 и т.д. Для переименования полей необходимо установить курсор в любую ячейку столбца поля, выполнить команду **Формат/Переименовать** столбец и ввести имя столбца**.**

Ввод данных в таблицу

1. Для ввода данных следует перейти в режим таблицы щелчком по кнопке **Вид** на панели инструментов и заполнить таблицу значениями.
2. При необходимости можно упорядочить записи в таблицах. Для этого необходимо выделить столбец, по которому выполняется сортировка, и щелкнуть по кнопке **Сортировать по** ... на панели инструментов.

Создание схемы данных

Преимущество БД обеспечивается работой с группами взаимосвязанных таблиц. Для создания связей между таблицами используется диалоговое окно **Схема данных**. Окно открывается командой **Сервис/Схема** данных или кнопкой **Схема данных** на панели инструментов. Если связи между таблицами еще не заданы, то одновременно с окном **Схема** данных открывается окно **Добавление таблицы**, в котором выбирают нужные таблицы для включения в структуру связей. Если связи уже заданы, для включения в схему новой таблицы необходимо щелкнуть правой кнопкой мыши на значке **Схема данных** и в контекстном меню выбрать **Добавить таблицу.**

Связи между атрибутами устанавливаются перетаскиванием имен атрибутов из одной таблицы в другую на связанный атрибут. После перетаскивания в диалоговом окне **Связи** могут быть заданы свойства образующейся связи. Включение флажка **Обеспечение условия целостности данных** исключает возможность удаления записей из таблицы, после которого связанные с ними данные других таблиц останутся без связей.

Флажки **Каскадное обновление связанных полей, и Каскадное удаление связанных записей** обеспечивают одновременное обновление или удаление данных во всех подчиненных таблицах при их изменении в главной таблице.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ.

1. Создать базу данных “Прокат компакт-дисков”. БД содержит информацию о том, кто какой диск взял, когда должен возвратить, сколько дисков каждого наименования осталось на складе. БД включает следующие отношения:

Отношение **Клиенты**

ФИО	Шифр	Адрес	№телефона	№паспорта

Отношение **Склад**

Диск	Стиль	Залоговая стои-мость	Количество	Наличие

Отношение **Прокат**

Шифр	Диск	Дата_выдачи	На_сколько_дней	Возврат

Структуру таблицы **Прокат** создать в режиме **Конструктор**, таблицы **Склад** – в режиме **Таблица**, структуру таблицы **Клиенты** – с помощью **Мастера таблиц**, недостающие поля – в режиме **Конструктор**. Сохранить таблицы. Типы данных должны соответствовать смыслу атрибутов. Атрибут **Возврат** таблицы **Прокат** имеет тип данных **логический**. Атрибут Наличие таблицы **Склад** – числовой.

Для всех таблиц в режиме **Конструктор** задать маски для ввода данных, условия на значения для контроля данных в процессе их ввода, тексты сообщений в случае ошибочного ввода данных, значения по умолчанию (в поле **Дата выдачи** значение по

умолчанию =Date() – функция возвращает текущую дату). Ввести описания полей. Задать ключевые атрибуты в отношениях **Прокат** (**Шифр**), **Клиенты** (**Шифр**) и **Склад** (**Диск**). Если необходимо переопределить ключевые атрибуты, следует удалить ключ и определить новый.

Переопределить ключ таблицы **Прокат** – добавить в ключ атрибуты **Диск, Дата_выдачи.**

2. Заполнить таблицы данными (не менее 10 записей). Значения полей **Диск** в таблице **Прокат** заполнить путем подстановки данных из таблицы **Склад**. (В режиме Конструктор выбрать вкладку **Подстановка**, Тип элемента управления**: Поле со списком**, источник строк – **Склад**).
3. Отредактировать структуру таблиц. Дополнить отношение **Склад** атрибутом **Обложка диска**, тип: **объект OLE**. (В режиме **Конструктор** добавить поле, в режиме **Таблица** вставить объект - точечный рисунок или рисунок из Clipart). Отсортировать таблицу **Прокат** в алфавитном порядке фамилий, таблицу **Склад** – в порядке возрастания залоговой стоимости.
4. Создать схему данных для созданной базы данных. Включить флажки **Каскадное обновление связанных полей, Каскадное удаление связанных записей.**

2.3. ИНФОРМАЦИОННЫЕ ВОЗМОЖНОСТИ MS ACCESS И ИХ РЕАЛИЗАЦИЯ

ЛАБОРАТОРНАЯ РАБОТА № 6

ОСНОВНЫЕ СВЕДЕНИЯ

При использовании БД, доступ пользователя к базовым таблицам обычно бывает закрыт. Обращение к информации БД, как правило, осуществляется с помощью запросов. Запрос позволяет отобрать необходимые данные из одной или нескольких связанных таблиц. Для одной и той же таблицы может быть создано много разных запросов, каждый из которых позволяет извлекать из таблицы ту информацию, которая в данный момент необходима. Информация в таблице – ответе на запрос может быть упорядочена, отфильтрована, объединена или обработана каким-либо другим образом. При этом исходные таблицы не изменяются. В запросах могут выполняться итоговые вычисления. Наиболее часто используются запросы на выборку. В результате такого запроса создается таблица, содержащая только данные, отвечающие условию запроса. Проще всего сформировать запрос с помощью бланка QBE – запроса по образцу.

Запрос на выборку

Для создания запроса в режиме **Конструктор** необходимо выполнить следующие действия:

1. Открыть вкладку **Запрос** диалогового окна **База данных**, и щелкнуть по кнопке **Создать**;
2. В диалоговом окне **Новый запрос** выбрать пункт **Конструктор**;
3. В диалоговом окне **Добавление таблицы** выбрать нужные таблицы и занести их в верхнюю половину бланка запроса щелчком по кнопке **Добавить таблицу** на панели инструментов. В диалоговом окне **Добавление** таблицы присутствуют три вкладки: **Таблицы, Запросы, Запросы и таблицы**. Это значит, что запрос можно создавать не только на основе таблицы, но и на основе уже созданного запроса.

В верхней части бланка запроса по образцу находятся списки полей таблиц, на которых основывается запрос. В нижней части таблицы определяется структура запроса, то есть таблицы, в которой будут содержаться данные, получаемые в результате ответа на запрос. Эта часть бланка формируется следующим образом:

1. Строка Поле заполняется перетаскиванием названий полей из верхней части бланка.

2. Строка **Имя таблицы** формируется автоматически при перетаскивании поля.
3. Щелчок по строке **Сортировка** открывает кнопку открывающегося списка, содержащего способ упорядочения данных.
4. Флажки в строке **Вывод на экран** обеспечивают отображение данных столбца.
5. В строку **Условие отбора** заносятся условия включения записи в результирующую таблицу. По каждому полю может быть создано свое условие отбора.
6. Запуск запроса осуществляется щелчком по кнопке **Запуск**(!) на панели инструментов. После этого формируется результирующая таблица.
7. Для выхода из результирующей таблицы и возврата в режим создания запроса следует еще раз щелкнуть по кнопке **Запуск.**

Для полей могут быть заданы свойства, например, изменен формат выводимых данных. Для задания свойств можно щелкнуть правой кнопкой мыши в любой ячейке столбца или нажать кнопку **свойства** на панели инструментов.

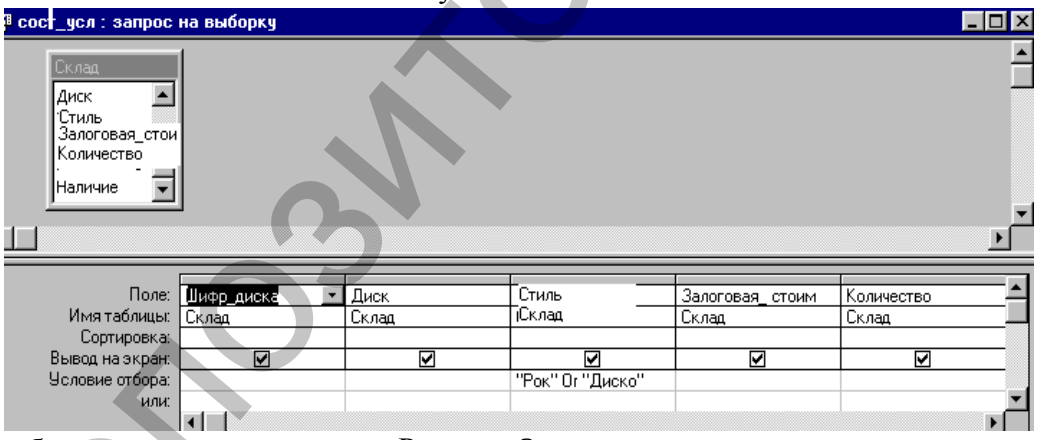
Ввод условий отбора

Для отбора записей с конкретным значением поля, это значение вводится в ячейку **Условие отбора**. Текстовое значение вводится без кавычек, они добавляются автоматически. Если необходимо задать несколько условий для одного поля, они вводятся в строку **Условие отбора** с использованием знаков логических операций – AND, OR. Условия, в которых используется логическая операция ИЛИ, могут быть введены двумя способами:

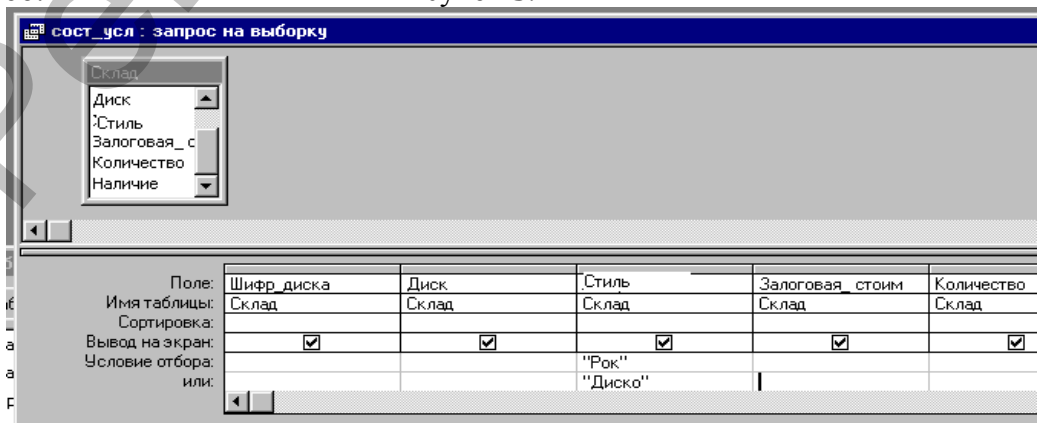
1) в поле **Условие отбора** вводится составное условие со знаком операции **OR**;
2) в поле **Условие отбора** вводится первое условие, остальные условия вводятся в строку **Или.**

Например, для отбора из таблицы **Склад** записей, относящихся к стилям **Рок** и **Диско,** условия отбора могут быть заданы так:

1 способ: Рисунок 2.



2 способ: Рисунок 3.



Кроме общепринятых логических операций, Access предоставляет три специальных оператора, предназначенных для отбора данных:

BETWEEN – определяет диапазон значений. Например, выражение BETWEEN 100 And 200 означает то же, что >=100 And <=200.

Например, для отображения дисков, взятых клиентами, не ранее 5-декабря-2000 и не позднее 5-декабря-2000 использует условие: Between #05.12.00# And #25.12.00# (Значения типа Дата и время вводятся в символах – "#").

IN – задает используемый список значений. Так, выражение IN("A","B","C") означает то же, что "A" Or "B" Or "C".

Например, для отбора записей относящихся к стилям Рок из таблицы склад в строку Условие отбора можно записать условие: In("Рок").

LIKE – используется для поиска образцов в текстовых полях. В образец поиска можно включать символы шаблона:

- "?" – один символ в данной позиции и "*"- любое количество (включая нулевое) символов в данной позиции.
- ! – исключение сиволов, указанных после подстановочного знака.
- "#" означает, что в данной позиции должна стоять цифра. (Полный список подстановочных знаков приведен в разделе справочной системы "Использование подстановочных знаков для поиска значений").

Допустимый диапазон символов для данной позиции должен быть заключен в квадратные скобки, восклицательный знак используется для указания исключений. Выражение Like "?[d-h]a[0-9*]" проверяет наличие произвольного символа в первой позиции, буквы от *d* до *h* – во второй позиции, буквы *a* - в третьей позиции, цифры – в четвертой позиции, и произвольного числа и набора символов в конце строки. Выражение Like "?[!d-h]a[0-9*]" отличается от предыдущего тем, что во второй позиции могут быть любые символы, исключая диапазон от d до h.

Например, для отбора записей, значение поля **Стиль** у которых начинается на буквы "Д", или "К" можно ввести в строку Условие отбора: Like "Д*" or Like "К*"

Вычисляемые поля

Результирующие таблицы запросов могут содержать вычисляемые поля. Использование вычисляемых полей позволяет экономить память. Те данные, которые могут быть вычислены на основе имеющейся в таблицах информации, нет необходимости хранить в базовых таблицах. Для выполнения вычислений вводится выражение, результат вычисления которого заносится в новое поле запроса. Выражения могут включать арифметические операции, встроенные функции Access. Выражения для вычисляемых полей могут быть созданы с использованием построителя выражений. Для создания вычисляемого поля необходимо выполнить следующие действия:

- установить курсор в строку **поле***;*
- щелкнуть по кнопке **Построить** на панели инструментов;

ввести выражение для вычисления значения поля в область ввода окна **Построитель выражений.** Выражение может быть построено с помощью знаков операций, перечня имен полей, списка функций, представленных в окне построителя. Пример выражения для вычисления количества просроченных дней возврата дисков приведен на рисунке 4. В выражении используется встроенная функция Date(), возвращающая текущую дату. Идентификаторы полей вводятся в квадратных скобках. Полный идентификатор поля записывается следующим образом: [Имя таблицы].[Имя поля].

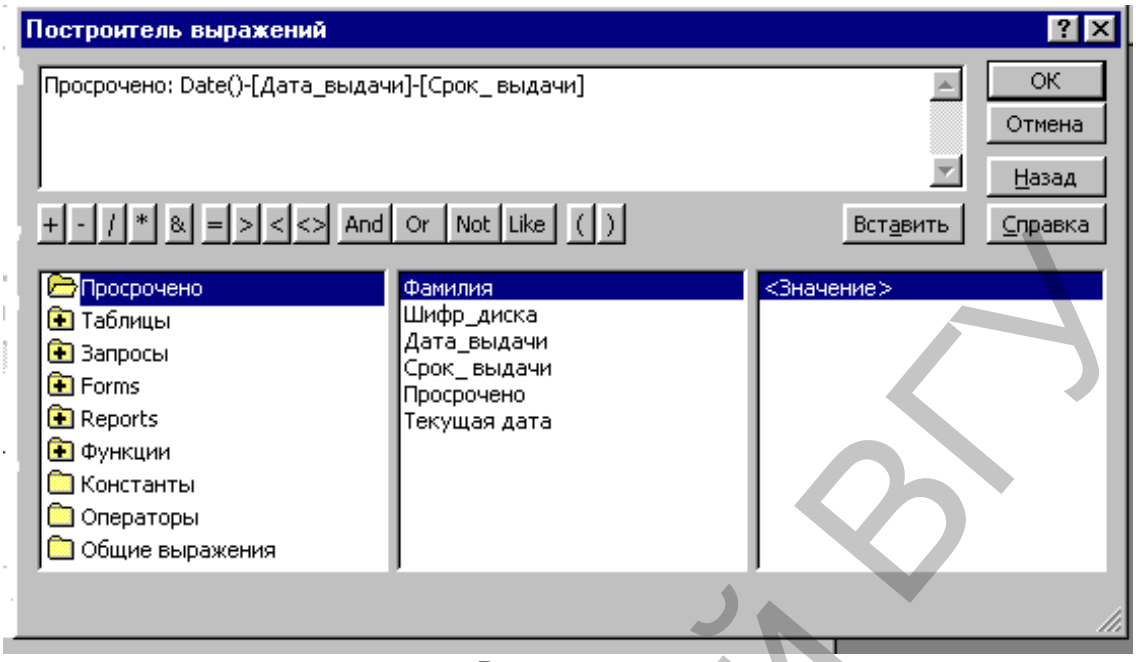


Рисунок 4.

Итоговый запрос

Для вычисления итоговых значений по группам данных используется итоговый запрос. Рассмотрим пример формирования таблицы, содержащей данные о количестве дисков, относящихся к каждому стилю (Исходные данные содержатся в таблице **Склад**). Для получения таблицы следует в режиме конструктора запроса выполнить следующие действия:

- из таблицы **Склад** перетащить в ячейку **Поле** первого столбца бланка идентификатор поля **Стиль***;* в ячейку **Поле** второго столбца – идентификатор поля **Количество***;*
- щелкнуть по кнопке **Групповые операции** (Σ) на панели инструментов. В поле **Количество**, где должно быть выполнено групповое вычисление, раскрыть список и выбрать итоговую функцию **Сумма.**
- Запустить запрос щелчком по кнопке **Запуск (!)**

Для того чтобы в итоговой таблице выводилось количество дисков, относящихся только к одному стилю, используется запрос с параметром. Для создания такого запроса служит оператор - LIKE[]. В квадратных скобках задается текст, обращенный к пользователю. Оператор LIKE вводится в поле **Условие отбора** (см. рисунок 5).

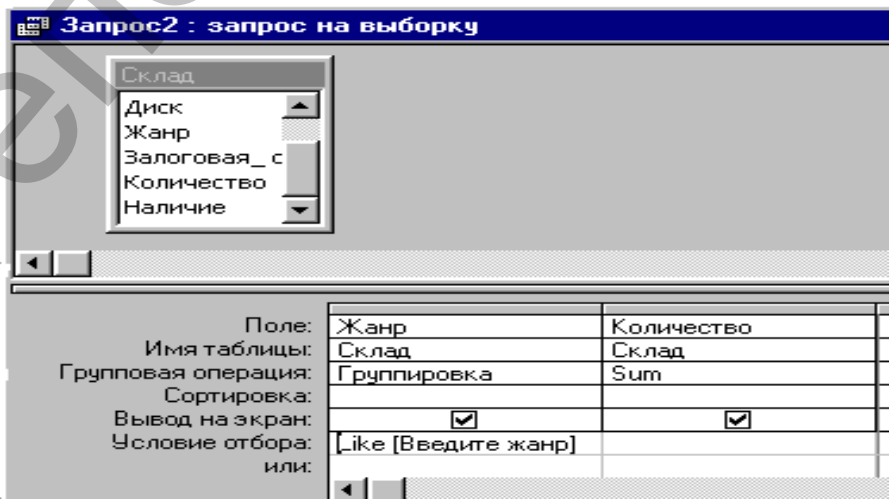


Рисунок 5.

После запуска запроса, выводится окно, в котором задается значение параметра **Стиль.**

Запросы на изменение

К числу запросов на изменение относятся запросы на обновление данных, запросы на добавление и запросы на удаление записей. Запросы на изменения позволяют автоматически создавать новые таблицы или изменять уже имеющиеся. При этом выполняется следующее: создается запрос, с помощью которого отбираются данные из существующих таблиц или создаются новые данные путем вычислений; после запуска запроса создается временная результирующая таблица. Данные этой временной таблицы служат основой для создания новых, или изменения уже существующих таблиц. Запрос на выборку преобразуется в нужный тип запроса (с помощью меню **Запрос** или **кнопки Тип запроса** на панели инструментов).

Запрос на добавление

Запрос на добавление позволяет скопировать данные из одной таблицы в другую, не прибегая к помощи буфера обмена. Например, имеется таблица **Склад1**, имеющая такую же схему, как и таблица **Склад**. Добавим в таблицу **Склад** записи из таблицы **Склад1**. Для этого необходимо выполнить следующие действия:

- На вкладке **Запрос** нажать кнопку **Создать** и выбрать режим **Конструктор.**
- Выбрать из списка таблицу **Склад1** и нажать кнопку **Добавить**. Таблица **Склад1** будет источником данных при выполнении запроса.
- В режиме Конструктора запросов выбрать тип запроса **Добавление**. В окне **Добавление** выбрать из списка имя таблицы, в которую надо добавить данные - **Склад.**
- Определить поля, которые будут добавлены в таблицу **Склад** (*-все поля).
- Нажать кнопку **Запуск.**

Для добавления не всех, а только выборочных записей из таблицы источника в соответствии с некоторым условием, необходимо заполнить строку условие отбора. Перед выполнением запроса можно увидеть данные, которые будут вставлены в таблицу. Для этого достаточно переключить режим просмотра запроса: выбрать в списке **Вид** элемент **Режим таблицы.**

Запрос на обновление

Рассмотрим применение запроса на обновление для изменения значений поля **Шифр (**к имеющимся значениям добавляется 0) . Для создания запроса необходимо выполнить следующее:

- Создать запрос на выборку - для таблицы **Клиент** ввести в бланк запроса поле **Шифр.**
- Преобразовать запрос на выборку в запрос на обновление (команда **Запрос/Обновление**).
- В поле **Шифр** строки **Обновление** ввести: Шифр*10.
- Запустить запрос на обновление.

Запрос на удаление

Создадим запрос на удаление, предназначенный для удаления из таблицы **Прокат** дисков, которые были взяты более 30 дней назад. Для этого следует:

1. Создать новый запрос, содержащий таблицу, из которой необходимо удалить записи.
2. В режиме конструктора запроса выбрать тип запроса - **Удаление.**
3. Переместить символ (*) из списка полей в бланк запроса. (В ячейке **Удаление** в этом столбце появляется значение “Из”, как показано на рисунке 6).
4. Чтобы задать условия отбора удаляемых записей необходимо переместить с помощью мыши в бланк запроса атрибуты, для которых устанавливаются условия

отбора (**Дата_выдачи**). В ячейке **Удаление** в этом поле появляется значение **Условие**.

5. Ввести условие отбора в ячейку **Условие отбора** для полей, перемещенных в бланк запроса.

7. Для удаления записей нажать кнопку **Запуск** (!) на панели инструментов.

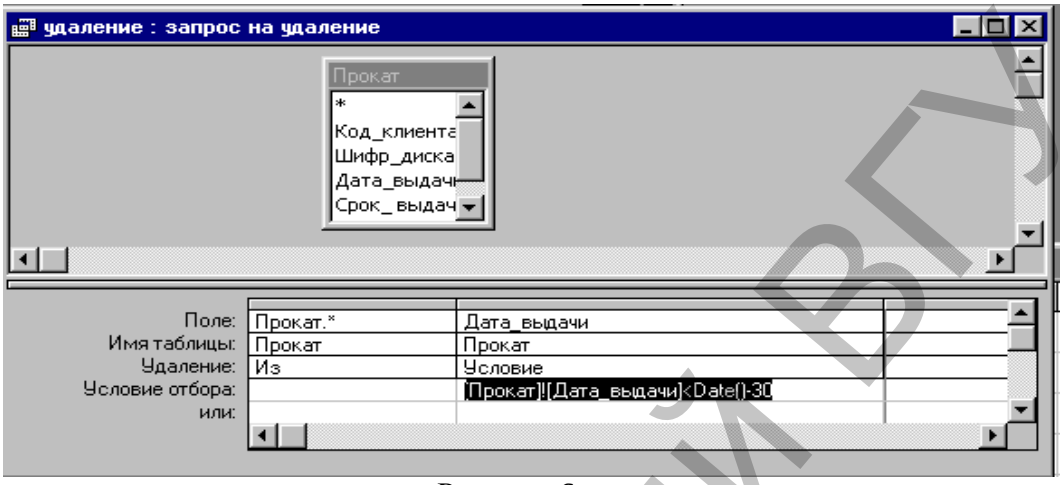


Рисунок 6.

Запрос на создание новой таблицы

Запрос на создание новой таблицы строится на основе запроса на выборку.

Создадим таблицу **Прокат1**, дополнив таблицу **Прокат** двумя полями: **Просрочено_дней и Долг**. Для этого:

1. Создадим запрос на выборку, включив в него все поля таблицы, Прокат и два новых вычисляемых поля: **Просрочено_дней и Долг.**

2. Преобразуем запрос на выборку в запрос на создание таблицы (команда **Запрос/Создание таблицы**).

3. Введем имя таблицы – **Прокат1**.

Перекрестный запрос

Перекрестный запрос – это операция построения таблица для вычисления итоговых значений на основе существующей таблицы. Перекрестный запрос создается в предположении, что исходная таблица содержит необходимые данные для формирования заголовков строк и столбцов новой таблицы.

Создание перекрестного запроса рассмотрим на примере преобразования таблицы **Расписание**.

Фамилия	День недели	№_пары	Аудитория
Семенов	понедельник	1	206
Семенов	понедельник	3	305
Семенов	вторник	3	303
Семенов	среда	3	305
Семенов	пятница	2	306
Иванова	понедельник	1	203
Иванова	четверг	2	202
Иванова	пятница	1	203

Создадим таблицу, данных о количестве пар каждого преподавателя по дням недели:

График работы

Фамилия	**Итого**	**Понед.**	**Вторник**	**Среда**	**Четверг**	**Пятница**
Семенов	**5**	**2**	**1**	**1**		**1**
Иванова	**3**	**1**			**1**	**1**

Перекрестный запрос проще всего создать с помощью Мастера. В окне создания запроса следует выбрать **Перекрестный запрос.** На первом шаге выбирается таблица, на основании которой строится запрос – **Расписание**. На втором шаге следует задать поле, значение которого будет использовано для создания строк таблицы (**Фамилия**). Если поле содержит повторяющиеся значения, то в перекрестном запросе будет выполнена группировка данных.

На третьем шаге необходимо определить поле, значение которого будет использоваться для формирования столбцов и их заголовков (выбираем поле **День недели**). Каждому уникальному значению указанного поля будет соответствовать свой столбец в новой таблице.

Далее выбирается функция, которая должна быть использована для подведения итогов и подсчета количества значений. На последнем шаге запросу присваивается имя.

Фильтры.

Фильтр выполняет ту же функцию, что и запрос – отбор нужных данных. Если запрос – это один из объектов БД, то фильтр – это свойство объекта базы данных, предназначенное для поиска записей. Основные разновидности фильтров:

- Фильтр по выделенному;
- Обычный фильтр;
- Расширенный фильтр.

Для задания фильтра по выделенному следует выделить одно или несколько смежных полей (типы данных в полях должны совпадать) и выполнить команду **Записи/Фильтр/По выделенному** или щелкнуть кнопку **Фильтр по выделенному**. В таблице останутся только те записи, у которых значения полей совпадает с выделенным. Команда **Исключить выделенное** также обеспечивает фильтрацию, но с обратным критерием – скрывает записи, значения полей которых совпадают с выделенными. Команда **Удалить фильтр** не удаляет сам использованный критерий, а только отменяет действие фильтра, делая все записи доступными.

Команда **Изменить фильтр** позволяет отредактировать условия отбора в окне Обычного фильтра. Команда **Изменить фильтр** выводит окно **Запроса по форме.** Если условие отбора для таблицы еще не задано, то в нижней части находится ярлычок вкладки **Найти**. В каждом столбце, следует ввести логическое выражение для ввода составных условий используется ярлычок **Или**. Для выполнения фильтрации следует выполнить команду **Применить фильтр**. Отменяется фильтрация командой **Удалить фильтр.**

Расширенный фильтр вызывается командой **Записи/Фильтр/Расширенный** фильтр. Окно расширенного фильтра напоминает окно создания запроса. Контекстное меню расширенного фильтра содержит команды, позволяющие преобразовывать расширенные фильтры в запросы и наоборот.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Для чего предназначены запросы?
2. Какие виды запросов Вам известны?
3. Каким образом задаются таблицы, из которых берутся данные для запроса?
4. Какими способами вводятся составные условия отбора?
5. Перечислите операторы, предназначенные для отбора данных.
6. Каким образом создается вычисляемое поле?
7. Как создать итоговый запрос?
8. Какие виды запросов относятся к числу запросов на изменение?
9. Для чего предназначен запрос на добавление данных?
10. Что представляет собой перекрестный запрос и как он формируется?

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1 . Создать запросы на выборку в базе данных **Прокат_ дисков.**

a) Вывести список не вернувших диски клиентов проката (в алфавитном порядке), их номера телефонов и названия невозвращенных дисков.

b) Вывести список невозвращенных дисков, взятых более 30 дней назад. Отсортировать список в порядке возрастания срока задолженности.

c) Вывести список имеющихся на складе дисков, относящихся к определенному стилю *(*использовать **Запрос с параметром**).

d) Вывести список дисков, относящихся к заданному стилю (использовать оператор **In**),отсутствующих на складе.

e) вывести список клиентов, взявших диски в заданном интервале времени, использовать оператор **Between.**

f) вывести список клиентов, фамилии которых начинаются на букву «М», третья буква «к» или «р».

2.Создать запросы с вычисляемыми полями:

a) В таблице **Прокат** подсчитать плату за прокат (200 рублей за каждый день проката + залоговая стоимость).

b) Вывести список клиентов проката, просрочивших возврат дисков; количество просроченных дней (текущую дату возвращает функция Date());Название диска; штраф – 10% от залоговой стоимости за каждый день просрочки, и сумму, которую должен доплатить клиент (штраф +доплата за просроченные дни). Имя запроса: **Штрафы.**

4. Создать итоговые запросы:

a) Вывести данные об общем количестве дисков по каждому стилю на складе.

b) Вывести количество дисков по одному из стилей (по запросу пользователя).

c) На основе запроса **Штрафы** создать итоговый запрос для подсчета общей суммы штрафа каждого клиента за все невозвращенные в срок диски.

d) Вывести данные о сумме, внесенной каждым клиентом за какой-либо период.

5.Создать запросы на обновление:

a) Увеличить залоговую стоимость всех дисков на 10 %.

b) Исключить из таблицы **Прокат** всех клиентов, не обращавшихся в прокат в течение 2-х месяцев.

c) Создать таблицу **Склад1**, внести в нее несколько записей. Создать запрос на добавление и добавить данные из таблицы **Склад1** в таблицу **Склад.**

d) Удалить из таблицы **Склад** все записи, относящиеся к заданному стилю, у которых залоговая стоимость ниже 1000 руб.

e) Создать новую таблицу, которая будет содержать следующие данные: название диска, стиль, количество дисков.

6.Создать таблицу **График работы сотрудников проката** с атрибутами: **Фамилия, День недели, Количество часов**. Заполнить таблицу (каждый сотрудник должен отработать 24 часа в неделю). Создать перекрестный запрос (заголовки строк – фамилии сотрудников, заголовки столбцов – дни недели)**.**

7.Выполнить фильтрацию данных:с использованием фильтра по выделенному вывести только те записи таблицы **Прокат**, у которых значение поля **Возврат** =Ложь; в таблице **Склад** вывести диски, относящиеся к определенному стилю.

a) С помощью обычного фильтра вывести только те записи таблицы **Склад**, у которых значение поля **Количество** > 10 и залоговая стоимость превышает 2000 руб.

b) Через расширенный фильтр вывести записи таблицы **Прокат**, значение поля **На_сколько_ дней** превышает заданное значение. Преобразовать таблицу в запрос.

2.4. ЯЗЫК SQL

ЛАБОРАТОРНАЯ РАБОТА № 7

ОСНОВНЫЕ СВЕДЕНИЯ

Большинство запросов может быть построено в режиме конструктора запросов, однако в некоторых случаях возможностей конструктора может быть недостаточно. В таких случаях используется язык SQL. Запросы, созданные в режиме конструктора, также реализованы на языке SQL. Они могут быть просмотрены с помощью команды **Запрос/ SQL.**

Ядро языка SQL – инструкция SELECT. Она используется для отбора строк и столбцов таблиц базы данных и содержит пять основных предложений. В общем случае синтаксис инструкции SELECT может быть представлен в следующем виде:

SELECT <список полей>
FROM <список таблиц>
[**WHERE** <спецификация выбора строк>]
[**GROUP BY** <спецификация группировки>]
[**HAVING** <спецификация выбора групп>]
[**ORDER BY** <спецификация сортировки>]

Список полей задается полным идентификатором: имя таблицы.имя поля в тех случаях, если имя поля может быть неоднозначным в контексте запроса, т.е. если одно и то же имя встречается в нескольких таблицах, перечисленных в предложении **FROM.**

Имена, содержащие пробелы обязательно заключаются в квадратные скобки, например: [Список поставщиков].[Фамилия поставщика].

Раздел **FROM** определяет таблицы или запросы, служащие источником данных для создания запроса.

Если список предложения **FROM** содержит несколько таблиц или запросов и для них не заданы условия объединения, то для них используется прямое (декартово) произведение всех таблиц. Например:

FROM TABLE A, TABLE B

заставляет Acess считать областью поиска все строки из **TABLE** A, присоединенные к каждой из строк **TABLE B**. В этом случае число записей будет равно числу строк таблицы А, умноженному на число строк табл В.

Раздел **WHERE** задает условия отбора.

Например, имеется БД состоящая из 2-х отношений:

отношение **Заказы**

ISBN	№_заказа	Фамилия	Город	Количество

отношение **Книги**

ISBN	Название_книги	Цена

Вывести список книг, цены которых превышают 2000 руб.

SELECT Книги.ISBN,Книги.Цена
FROM Книги
WHERE Книги.Цена >2000;

Возможности SQL основаны на его способности объединять информацию из нескольких таблиц или запросов, и представлять результаты в виде единого логического набора записей. Для задания типа соединения таблиц единый набор записей, из которого будет выбираться необходимая информация, в предложении **FROM** используется операция **JOIN.** В логический набор записей можно включать только соответствующие строки обеих таблиц. Эта операция называется **INNER JOIN** – внутреннее объединение.

Операция **LEFT JOIN** возвращает все строки из первой таблицы, объединенные с теми строками второй, для которых выполняется условие объединения.
Если во второй таблице таких строк нет, Access возвращает значения **NULL** в строках второй таблицы. Аналогично, операция **RIGHT JOIN** возвращает все строки второй таблицы, объединенные с теми строками первой, для которых выполняется условие объединения.

Пример:

SELECT Заказы.ISBN, Книги.Название, Заказы.N_заказа, Заказы.Фамилия, Заказы.Количество
FROM Заказы INNER JOIN Книги
ON Заказы.ISBN = Книги.ISBN;

Раздел **GROUP BY** используется для создания итоговых запросов.
Синтаксис**: GROUP BY**<имя столбца>
Имя столбца – имя любого столбца из любой из упомянутой в предложении **FROM** таблицы.
Если **GROUP BY** расположено после **WHERE,** создаются группы из строк, выбранных после применения предложения **WHERE**.
При включении предложения **GROUP BY** в инструкцию **SELECT** список полей должен состоять из итоговых функций **SQL (AVG, COUNT, MAX, MIN, SUM** и др.) и из имен столбцов, указанных в предложении **GROUP BY.**
Например: вычислить групповые итоги по каждому покупателю, фамилии которых начинаются на букву от ‘А’ до ‘О’.

SELECT Заказы.Фамилия, Sum(Заказы.Количество) AS Итого
FROM Заказы
WHERE Заказы.Фамилия < 'П'
GROUP BY Заказы.Фамилия;

Раздел задает группы строк, которые включаются в таблицу, определяемую инструкцией **SELECT**.
Условия отбора применяется к столбцам, указанным в предложении **GROUP BY**, к столбцам итоговых функций или к выражениям, содержащим итоговые функции. Если некоторая группа не удовлетворяет условию отбора, она не попадает в набор записей.
Синтаксис: **HAVING** <условие отбора>
Разница между **HAVING** и **WHERE** заключается в том, что условие отбора, заданное в предложении WHERE применяется к отдельным записям, перед их группировкой, а условие отбора предложения **HAVING** применяется к группам строк. Если раздел **GROUP BY** находится перед **HAVING**, условие отбора применяется к каждой из групп, сформированных на основе совпадения значений в заданных столбцах. В случае отсутствия предложения **GROUP BY** условие отбора применяется ко всей таблице определенной инструкцией **SELECT**.

Пример: условие предыдущего примера.

SELECT Заказы.Фамилия, Sum(Заказы.Количество) AS Итого
FROM Заказы
GROUP BY Заказы.Фамилия
HAVING Заказы.Фамилия < 'П';

Раздел **ORDER BY** задает порядок расположения строк, возвращаемых инструкцией **SELECT**.
Синтаксис **ORDER BY** <имя столбца[**ASC[DESC**]]>
В предложении можно указать несколько столбцов. По умолчанию применяется порядок – по возрастанию.

Пример:

SELECT ISBN, Код_заказа, Фамилия, Город, Количество
FROM Заказы
ORDER BY Фамилия;

В инструкции **SELECT** используется предикат **ALL**, **DISTINCT**, **DISTINCTROW**, **TOPn**, которые уточняют окончательный набор записей, выводимых по запросу. По умолчанию действует **ALL**, при котором в набор записей включаются все строки, удовлетворяющие условиям отбора, в т.ч. и дубликаты. **DISTINCT** требует, чтобы возвратились только строки, отличающиеся от всех остальных. Если инструкция **SELECT** содержит предикат **DISTINCTROW**, в набор записей включаются только те, строки, в которых конкатенация первичных ключей во всех таблицах, участвующих в формировании возвращаемых столбцов является уникальной. Этот предикат оказывает влияние только при использовании в запросе операции **JOIN**.

Запросы на удаление, добавление, создание новых таблиц

Запрос на удаление

Синтаксис: **DELETE** [<таблица>.*]
FROM<таблица>
WHERE <условие отбора>

Пример 1: удалить из таблицы **Заказы** все записи, относящиеся к Иванову

DELETE Заказы.*
FROM Заказы
WHERE Фамилия = “Иванов”;

Пример 2. Имеется таблица **Книги1**, со структурой, аналогичной структуре таблицы **Книги**. Удалить из таблицы **Книги** те записи, которые содержатся в таблице **Книги**.

DELETE Книги.*
FROM Книги
WHERE Книги.ISBN IN
(SELECT Книги1.ISBN FROM Книги1);

Запрос на добавление

Синтаксис запроса на добавление нескольких записей:
INSERT INTO<имя таблицы>[(<имя столбца1>,[<имя столбца2>[,…]])]
SELECT [<таблица источник>] <имя столбца1>[,<имя столбца2>[, …]

Если в запросе не указана таблица источник, необходимо предоставить новые значения для всех столбцов.

Пример 1:

Добавить в таблицу **Книги** новую строку со следующими значениями полей: **ISBN** = Ш1, **Цена** 2555 руб.

INSERT INTO Книги (ISBN,Цена)
SELECT ”Ш1” AS ISBN,2555 AS Цена;

Пример 2: Отобрать из таблицы **Книги1** (структура такая же, как у таблицы **Книги**) все записи, у которых цена выше 3000 руб. и добавить их в таблицу **Книги**.

INSERT INTO Книги
SELECT Книги1.*
FROM Книги1
WHERE Цена > 3000;

Запрос на создание таблицы

SELECT<поле1>[<поле 2>[,…]]
INTO <новая таблица>
FROM<источник>

Пример: Создать новую таблицу **Заказы1**, и скопировать в нее фамилии всех заказчиков, заказавших более 10 книг.

SELECT Книги.фамилия
INTO Книги1
FROM Книги
WHERE Количество>10;

Запрос на обновление

Предназначен для создания запроса на обновление записей, который изменяет значения полей на основе заданного условия отбора.

Синтаксис: **UPDATE**<таблица>
SET<новое значение>
WHERE<условие отбора>

Пример: Увеличить на 10 % цены всех книг, цена которых ниже 2000 руб.

UPDATE Книги
SET Цена=Цена*1.1
WHERE Цена>2000;

Перекрестный запрос

Перекрестный запрос – это операция построения таблицы для вычисления итоговых значений на основе существующей таблицы. Перекрестный запрос строится, исходя из предположения, что исходная таблица содержит необходимые данные для формирования заголовков строк и столбцов новой таблицы.

Синтаксис:

TRANSFORM <выражение с итоговой функцией>
инструкция **SELECT**
PIVOT <поле>
[**IN**(<список заголовков столбцов>)]

Выражение с итоговой функцией определяет значения, которые должны появиться в ячейках перекрестной таблицы.

Инструкция **SELECT** содержит раздел **GROUP BY**, в котором задаются атрибуты, используемые как заголовки строк.

PIVOT - выражение задает столбец или выражение, значения которого используются в качестве заголовков столбцов в перекрестном запросе.

Список заголовков столбцов необязательный список значений, заключенных в кавычки и отделенных друг от друга запятыми, определяющий порядок вывода столбцов.

Пример: Создать перекрестный запрос для получения данных о количестве заказанных книг в каждом городе.

TRANSFORM Sum(Заказы.Количество) AS [Значение]
SELECT Заказы.Название_книги, Sum(Заказы.Количество) AS Всего
FROM Заказы
GROUP BY Заказы.Название_книги
PIVOT Заказы.Город;

Таблица, полученная на основе перекрестного запроса:

Название_книги	**Всего**	**Витебск**	**Минск**
MS Excel в бизнесе	3	3	
Базы данных	16	6	10
Учебник по Internet	6		6
Базы данных	24	11	13
Информационные техноло-	1		1

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какая инструкция языка SQL предназначена для отбора данных?
2. В каких случаях имя поля должно задаваться полным идентификатором?
3. Что определяет раздел **FROM**?
4. Для чего служит раздел **WHERE**?
5. Каково назначение операции **INNER JOIN**?
6. Какая операция позволяет создать результирующую таблицу, в которой соединены все строки из первой таблицы и те строки второй, для которых выполняется условие объединения?
7. Какой раздел служит для создания итоговых запросов?
8. Для чего предназначен раздел **HAVING**?
9. Какой раздел задает порядок расположения строк, возвращаемых инструкцией **SELECT**?
10. Для чего служит перекрестный запрос? Каким образом строится перекрестный запрос в режиме SQL?

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ.

Создать запросы на SQL:

1.Вывести список дисков, у которых поля “Портрет исполнителя” не заполнены (IS NULL)**.**

2.Вывести список дисков, находящихся у одного из клиентов проката, даты их выдачи, количество просроченных дней, сумму начисленного штрафа – 10 % от залоговой стоимости, доплату за просроченные дни. (Шифр клиента задается оператором).

3.Начислить общую сумму доплаты за все невозвращенные в срок диски для одного клиента (Шифр клиента задается оператором).

4.Вывести список клиентов проката, фамилии которых начинаются на заданную букву (задается оператором), просрочивших возврат дисков более чем на три дня. Отсортировать список в убывания просроченных дней.

5.Вывести список дисков ни разу не бывших в прокате. Снизить их залоговую стоимость в 2 раза.

6.Вывести данные об общей сумме дохода, полученной от каждого клиента проката (с учетом штрафов и доплат).

7.Вывести данные о сумме дохода, полученной прокатом за каждый из дисков, имеющихся на складе (разность общей суммы оплаты за весь период и залоговой стоимости).

8.Вычислить среднее значение залоговой стоимости дисков по каждому стилю.

9.Найти самый дорогостоящий диск по каждому стилю.

10. Вывести данные о сумме, внесенной каждым клиентом за последний месяц.

11. Создать таблицу **Склад1**, перенести в нее те записи из таблицы Склад, у которых значение поля ”Залоговая стоимость” превышает заданное оператором значение.

12. Удалить из таблицы **Склад** записи, содержащиеся в таблице **Склад1.**

13. Удалить из таблицы **Склад** диски, ни разу не бывшие в прокате.

14. Создать перекрестный запрос для создания таблицы следующей структуры:

Фамилия	**Итого**	**Диско**	**Классика**	**Рок**
Иванов	3	1	2	
…	1		1	
…	2	1	1	
…	1			1

3. ПРОГРАММИРОВАНИЕ В ИНТЕРНЕТ

Цель: данный раздел посвящен изучению приемов создания иерархии классов, выделению общих признаков объектов в базовый класс, организации доступа к элементам базового и производных классов.

3.1. СТРУКТУРА HTML-ДОКУМЕНТА

ЛАБОРАТОРНАЯ РАБОТА № 8

Гипертекстовая страница, написанная на языке HTML, состоит из элементов. Каждый элемент, в свою очередь, состоит из открывающего тэга (пишется <имя_элемента>), содержимого (текста, который может содержать другие элементы) и закрывающего тэга (пишется </имя_элемента>). Некоторые элементы при своем описании допускают отсутствие открывающего или закрывающего тэга, ряд элементов состоят только из одного открывающего тэга.

Рассмотрим структуру HTML-страницы на следующем примере:

```
<HTML>
 <HEAD>
  <TITLE>Моя первая страница</TITLE>
 </HEAD>
 <BODY>
  <P ALIGN=”CENTER”>Всем привет!</P>
 </BODY>
</HTML>
```

HTML-страница состоит из двух частей: элемента **HEAD** (заголовка документа, содержащего служебную информацию, используемую только браузером) и элемента **BODY** (тело документа, содержащего отображаемую в окне браузера информацию).

Элементы заголовка документа

TITLE Текст, который будет отображаться в заголовке окна браузера

META Задает метаданные — информацию о документе: кто является автором документа, в какой кодировке он подготовлен и т.д. (не требует закрытия). Например:

```
<meta name="author" content="Иванов Иван Иванович">
<meta http-equiv="Content-Type"
content="text/html; charset=windows-1251">
```

Элементы тела документа

При отображении HTML-страницы браузером полностью игнорируется разбиение исходного документа на строки и лишние пробелы.

BR Принудительный переход на новую строку (не требует закрытия)

HR Горизонтальная разделительная черта (не требует закрытия)

P Абзац.

BLOCKQUOTE Абзац-цитата

В большинстве открывающих тэгов допускается задание дополнительных атрибутов (параметров), влияющие на отображение содержимого элементов. В общем виде атрибут задается следующим образом:

```
<тэг атрибут1="значение" атрибут2="значение" ...>
```

Так, атрибут **ALIGN** в тэге <P ALIGN=”CENTER”> означает, что содержимое абзаца при отображении будет выровнено по центру строки. При указании других допустимых значений этого атрибута (LEFT, RIGHT или JUSTIFY) содержимое абзаца будет выровнено по левому краю, по правому краю или по ширине.
Другим атрибутом, который может быть указан в элементе **P** (и во многих других элементах), является **TITLE** (не путайте с *элементом TITLE*). Значение этого атрибута определяет всплывающую подсказку, которая отображается браузером при наведении на объект указателя мыши.

Задание наберите в блокноте и сохраните следующий текст:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Моя первая веб-страница</title>
 </head>
 <body>

  <h1>Заголовок страницы</h1>
  <p>Основной текст.</p>

 </body>
</html>
```

Чтобы посмотреть результат примера в действии, проделайте следующие шаги.
1. В Windows откройте программу Блокнот (Пуск > Выполнить > набрать «notepad» или Пуск > Программы > Стандартные > Блокнот).
2. Наберите или скопируйте код в Блокноте (рис. 1).

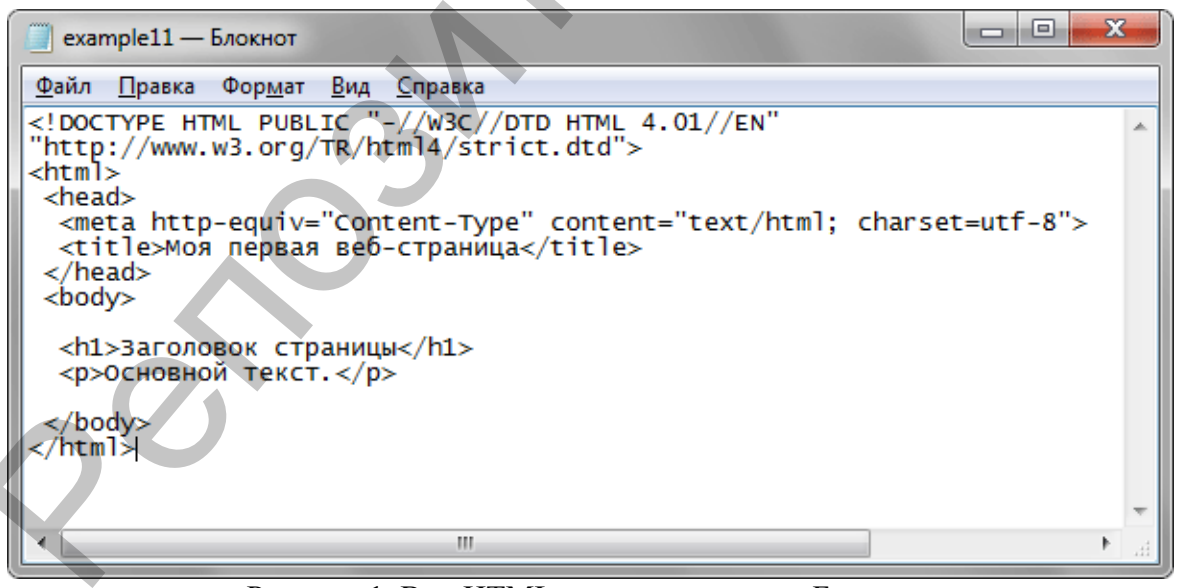
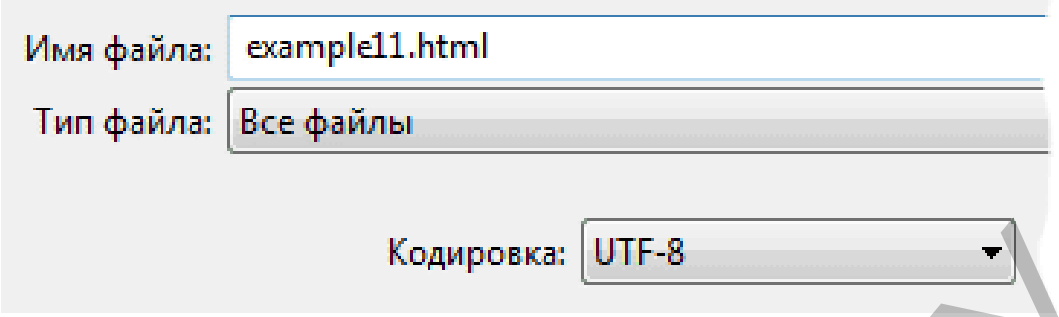


Рисунок 1. Вид HTML-кода в программе Блокнот.

3. Сохраните готовый документ (Файл > Сохранить как...) под именем D:\www\example11.html, при этом обязательно поставьте в диалоговом окне сохранения тип файла: Все файлы и кодировку UTF-8 (рис. 2). Обратите внимание, что расширение у файла должно быть именно html.



Риснок 2. Параметры сохранения файла в Блокноте.

4. Запустите браузер Internet Explorer (Пуск > Выполнить > набрать «iexplore» или Пуск > Программы > Internet Explorer).
5. В браузере выберите пункт меню Файл > Открыть и укажите путь к вашему файлу.
6. Если все сделано правильно, то в браузере вы увидите результат, как показано на рис. 3.

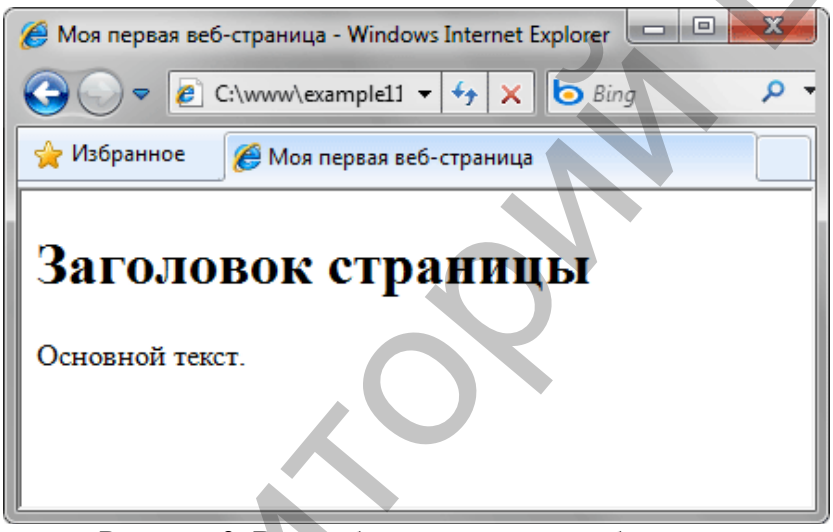


Рисунок 3. Вид веб-страницы в окне браузера.

В случае возникновения каких-либо ошибок проверьте правильность набора кода согласно примеру 1, расширение файла (должно быть html) и путь к документу.

3.2. ФОРМАТИРОВАНИЕ ТЕКСТА ДОКУМЕНТА

Лабораторная работа № 9

Для форматирования текста в языке HTML можно использовать шесть уровней заголовков: от **H1** (наиболее важный) до **H6** (наименее важный). Браузеры обычно отображают более важные заголовки более крупным шрифтом.

Следующие элементы управляют стилем шрифта:

B **полужирный текст;**
I *текст, выделенный курсивом*;
U подчеркнутый текст;
S ~~перечеркнутый текст;~~
TT стиль печатной машинки
BIG текст будет отображаться шрифтом чуть большего размера, чем основной;
SMALL текст будет отображаться шрифтом чуть меньшего размера, чем основной;
SUP верхний индекс
SUB нижний индекс

Элементы стиля шрифта должны корректно вкладываться, например, так

<B><I>полужирный курсив</I></B>

Для изменения гарнитуры, размера или цвета текста следует использовать элемент **FONT**, который имеет следующие атрибуты:

COLOR Задает цвет текста. В качестве значения можно указать или английское название шрифта, или его шестнадцатеричный код. Так, следующие примеры задают для текста зеленый цвет:

<FONT COLOR=”Green”>
<FONT COLOR=”#00FF00”>

FACE Задает имя шрифта (шрифтов), которым желательно отобразить текст, например,

<FONT FACE=”Arial,Helvetica,Tahoma”>

(используется первый найденный в системе шрифт)

SIZE Значение этого атрибута задает размер шрифта. Возможно указание абсолютной величины от 1 до 7 (по умолчанию 3) или относительной, например, следующий тэг

<FONT SIZE=”+1”>

приводит к увеличению размера шрифта текста *внутри элемента* на один шаг.

Гипертекстовые ссылки

Каждый ресурс в Web: документ HTML, изображение, видеоклип, программа и т.д. – имеет адрес, который может быть закодирован с помощью *универсального идентификатора ресурсов (Universal Resource Identifier*, или *URI*). Например, спецификации HTML на сервере W3C соответствует URI
http://www.w3.org/TR/PR-html4/cover.html
Здесь http – наименование протокола (способа доступа к ресурсу), www.w3.org – имя компьютера, на котором располагается ресурс, /TR/PR-html4/ – путь, cover.html – собственно имя ресурса.

Наиболее существенным отличием HTML-документа от большинства других текстов, является возможность включения специальной *гипертекстовой ссылки*, позволяющей осуществить переход к другим ресурсам.

Для создания гипертекстовой ссылки используется элемент **A** с атрибутом **HREF**:

<A HREF="Адрес ресурса">Текст ссылки</A>

Кроме того, в элементе **A** рекомендуется использовать атрибут **TITLE** для определения всплывающей над элементом подсказки.

Текст ссылки при отображении в окне браузера обычно подчеркивается и выделяется цветом. Для задания цвета ссылки следует использовать следующие атрибуты элемента **BODY**:

LINK	цвет текста ссылки
VLINK	цвет текста ссылки на посещенный ресурс
ALINK	цвет текста выделенной ссылки

В атрибуте **HREF** допускается использовать *относительный адрес*, не содержащий названия протокола, имени компьютера и пути (предполагается, что они такие же, как и у текущей страницы). Относительные адреса могут содержать компоненты относительного пути: “.” обозначает текущий уровень, “..” – на один уровень выше, “/” – самый верхний уровень.

Например, если текущая страница имеет абсолютный адрес http://www.w3.org/TR/PR-

html4/cover.html, то относительному адресу intro/intro.html соответствует абсолютный адрес http://www.w3.org/TR/PR-html4/intro/intro.html.
В гипертекстовой ссылке в адресе ресурса может быть указан идентификатор конкретного элемента, или закладка, к которой необходимо осуществить переход. Закладки могут использоваться как в относительных, так и в абсолютных адресах, и должны отделяться от имени ресурса знаком “#”, например, так:

<A HREF=”text.html#label”>

или, если закладка определена в текущем документе, так:

<A HREF=”#label”>

Для определения идентификатора, большинство элементов допускают использование атрибута **ID**, например:

<P ID=”label”>Этот абзац имеет идентификатор label.</P>

Для определения закладки следует использовать элемент **A** с атрибутом **NAME**, например:

<A NAME=”label2”>Это закладка label2.</A>

Имена, указываемые в атрибутах **ID** и **NAME** должны быть уникальными в пределах документа и начинаться с латинской буквы, за которой может следовать любое число латинских букв, цифр, символов подчеркивания (“_”), символов переноса (“–”), точек (“.”) и двоеточий (“:”).

Задание 1:

Для установки выравнивания текста обычно используется тег параграфа <p> с атрибутом align, который определяет способ выравнивания. Также блок текста допустимо выравнивать с помощью тега с аналогичным атрибутом align. Он может принимать следующие значения:

left – выравнивание по левому краю, задается по умолчанию;
right – выравнивание по правому краю;
center – выравнивание по центру;
justify – выравнивание по ширине (одновременно по правому и левому краю). Это значение работает только для текста, длина которого более, чем одна строка.

Атрибут align можно применять как для текста, так и для заголовков (пример 7.4).

Задание 2:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Выравнивание текста</title>
 </head>
 <body>
  <h1 align="center">Как поймать льва?</h1>
  <h2 align="right">Метод перебора</h2>
  <p align="justify">Делим пустыню на ряд элементарных участков, размер
   которых совпадает с габаритными размерами льва, но при этом меньше размера
   клетки. Далее простым перебором определяем участок, в котором находится
лев,
   что автоматически приводит к его поимке.</p>
  <h2 align="right">Метод дихотомии</h2>
  <p align="justify">Делим пустыню на две половины. В одной части - лев, в
   другой его нет. Берем ту половину, в которой находится лев, и снова делим
```

ее пополам. Так повторяем до тех пор, пока лев не окажется пойман.**</p>**
</body>
</html>

Результат данного примера показан на рис. 4.

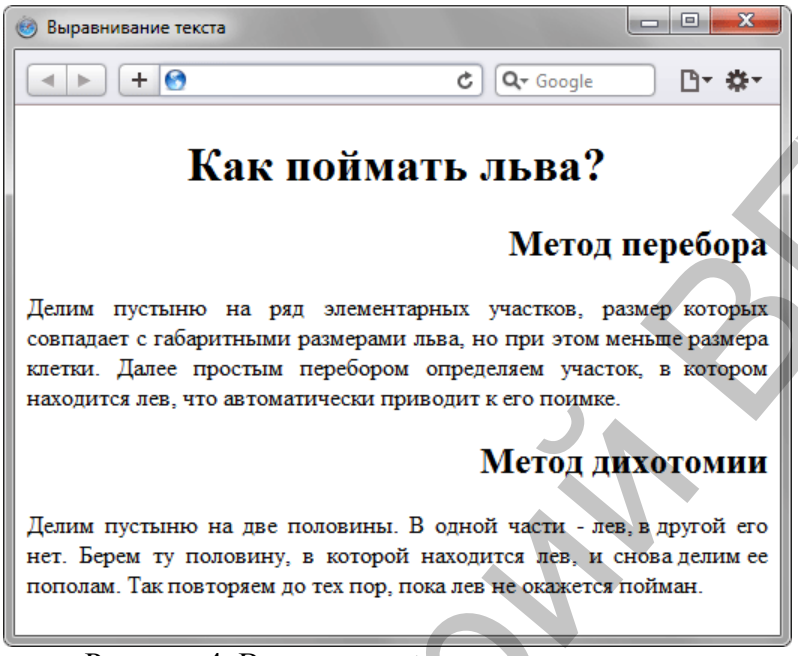


Рисунок 4. Вид текста при его выравнивании.

3.3. РАБОТА С ТАБЛИЦАМИ

Лабораторная работа № 10

Для упорядочения данных: текста, изображений, ссылок и т.п. в HTML используются *таблицы*. Таблицы формируются из необязательного заголовка и последовательности строк, каждая из которых, в свою очередь, состоит из последовательности ячеек.

Рассмотрим пример таблицы и соответствующего ей HTML-кода.

Ячейка 1	Ячейка 2	Ячейка 3
Ячейка 4	Ячейка 5	Ячейка 6

<TABLE BORDER=”1” WIDTH=”100%”>
<TR><TD>Ячейка 1</TD><TD>Ячейка 2</TD><TD>Ячейка 3</TD></TR>
<TR><TD>Ячейка 4</TD><TD>Ячейка 5</TD><TD>Ячейка 6</TD></TR>
</TABLE>

Вся таблица определяется содержимым элемента **TABLE**, внутри которого находится несколько элементов **TR** (**T**able **R**ow). Каждый элемент **TR** определяет одну строку таблицы. В свою очередь, внутри элемента **TR** находится несколько элементов **TD** (**T**able **D**ata), описывающих каждую ячейку таблицы. Закрывающие тэги **</TD>** и **</TR>** можно не указывать.Количество столбцов в таблице определяется количеством ячеек в самой длинной строке. Если в других строках ячеек меньше, то они дополняются пустыми (рамка у пустых ячеек не отображается). Ширина столбца таблицы определяется наиболее широкой ячейкой.

Наиболее часто таблицы используются для размещения текста web-страницы в несколько колонок. Чаще всего в таких случаях задают относительную ширину таблицы и каждой колонки.

Атрибуты элемента TABLE

ALIGN Определяет выравнивание таблицы относительно документа (возможные значения: left, center или right)

BGCOLOR Задает цвет фона таблицы

BORDER Задает толщину внешней рамки таблицы (в пикселах)

CELLSPACING Задает расстояние между ячейками таблицы (в пикселах)

CELLPADDING Задает расстояние между содержимым ячейки и ее рамкой (в пикселах)

WIDTH Ширина таблицы (в пикселах или в процентах от ширины окна)

Атрибуты элемента TR

ALIGN Определяет горизонтальное расположение текста в ячейках строки (возможные значения: left, center, right или justify)

VALIGN Определяет вертикальное расположение текста в ячейках строки (возможные значения: top, middle или bottom)

Атрибуты элемента TD

HEIGHT Задает рекомендуемую высоту ячейки (в пикселах)

WIDTH Задает рекомендуемую ширину ячейки (в пикселах или в процентах от ширины таблицы)

Задание1:

Для добавления таблицы на веб-страницу используется тег <table>. Этот элемент служит контейнером для элементов, определяющих содержимое таблицы. Любая таблица состоит из строк и ячеек, которые задаются соответственно с помощью тегов<tr> и <td>. Таблица должна содержать хотя бы одну ячейку (пример 12.1). Допускается вместо тега <td> использовать тег<th>. Текст в ячейке, оформленной с помощью тега <th>, отображается браузером шрифтом жирного начертания и выравнивается по центру ячейки. В остальном, разницы между ячейками, созданными через теги <td> и <th> нет.

Задание 2.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Тег TABLE</title>
 </head>
 <body>
  <table border="1" width="100%" cellpadding="5">
   <tr>
    <th>Ячейка 1</th>
    <th>Ячейка 2</th>
   </tr>
   <tr>
    <td>Ячейка 3</td>
     </tr>
 </table>
 </body>
</html>
```

Порядок расположения ячеек и их вид показан на рис. 5.

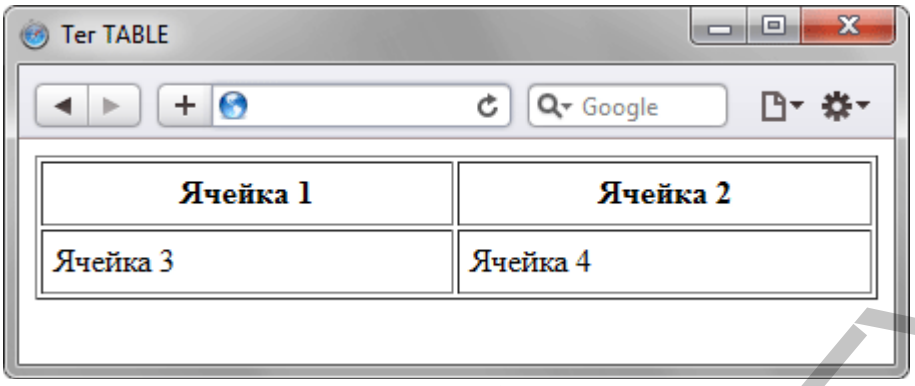


Рисунок 5. Результат создания таблицы с четырьмя ячейками.

Задание 2. Создайте страницу, реализующую календарь на текущий год и месяц, номер которого определяется вашим номером в журнале.

3.4. ГРАФИКА В HTML

ЛАБОРАТОРНАЯ РАБОТА № 11 .

Существует несколько возможностей для включения графических изображений в HTML-страницы. Наиболее часто картинки используются в качестве фона страницы. Для этого используется атрибут **BACKGROUND** элемента **BODY**. Например, тэг

<BODY BACKGROUND=”fon.gif”>

означает использование изображения из файла fon.gif, расположенного в том же каталоге, что и HTML-страница, в качестве фона страницы. Если размер картинки меньше размера окна, то браузер размножит это изображение по горизонтали и/или вертикали.
Для включения отдельного изображения используется элемент **IMG**, состоящий только из открывающего тэга, например:

<IMG SRC=”foto.jpg” ALT=”Это я на занятии”>

Атрибуты элемента IMG

SRC	Этот атрибут задает адрес (URI) изображения. Наиболее распространены изображения в формате GIF, JPEG и PNG
ALT	Задает альтернативный текст — текст, который будет выведен вместо изображения до его полной загрузки или при невозможности загрузки
ALIGN	Задает выравнивание изображения по вертикали (top, middle, bottom) относительно окружающего его текста или по горизонтали (left, right) относительно страницы
WIDTH	Задает рекомендуемую ширину картинки (в пикселах)
HEIGHT	Задает рекомендуемую высоту картинки (в пикселах)
HSPACE	Задает расстояние от окружающего текста по горизонтали
VSPACE	и вертикали
BORDER	задает ширину рамки вокруг изображения

Для включения графических изображений, видео- и аудиофрагментов в текст страницы можно использовать более универсальный элемент **OBJECT**. К сожалению, этот элемент браузеры интерпретируют по-разному.
Пример включения видеофрагмента:

<OBJECT DATA = ”FILE:///C:/Delphi5/Demos/Coolstuf /cool.avi”TYPE=”video/avi”></OBJECT>

Данный элемент требует обязательного закрывающего тэга.

Атрибуты элемента OBJECT

DATA Этот атрибут задает адрес (URI) данных объекта

TYPE Указывает тип объекта, например: application/msword, video/avi, image/jpeg, image/gif

ALIGN, WIDTH, HEIGHT, HSPACE, VSPACE, BORDER имеют тот же смысл, что и у **IMG**

Заметим, что браузер MS Internet Explorer допускает использование в элементе **IMG** не включенного в стандарт HTML атрибута **DYNSRC**. Значением этого атрибута является адрес (URI) видеофрагмента.

Если элемент **IMG** поместить внутри гипертекстовой ссылки, то переход по ней может осуществляться щелчком левой кнопки мыши по картинке.

Задание:

Для примера возьмем файл с рисунком, который называется sample.gif и хранится в папке images корня сайта.

- Если в начале адреса стоит слэш (символ /), это значит, что отсчет идет от корня сайта. Например, адрес сайта — http://baklan.narod.ru, значит, написав путь к изображению как /images/bird.jpg, мы, тем самым говорим серверу, что показать следует файл http://baklan.narod.ru/images/bird.jpg. Учтите, что подобные ссылки со слэшем впереди работают только на веб-сервере, на локальном компьютере они действовать не будут.
- Если перед адресом добавляется упоминание протокола http (http://), то речь идет об абсолютной ссылке. Изображение всегда будет загружаться с указанного адреса в Интернете, даже при сохранении веб-страницы на локальный компьютер.
- Двоеточие со слэшем (../) в начале адреса говорит о том, что и рисунок и веб-страница находятся в разных папках, которые размещены на одном уровне. На рис. 6 показан файл index.html, в который требуется поместить рисунок pic.gif. Тогда относительный путь к изображению из index.html будет ../images/pic.gif. Возможны случаи хранения файлов, что обращение из одного файла к другому превращается в набор двоеточий, например: ../../../images/pic.gif.

Рисунок 6. Пример размещения файлов

- Имя папки в начале пути, без всяких слэшей и двоеточий, сообщает, что и текущий файл и папка с изображением находятся на одном уровне. Как показано на рис. 7, относительный путь к рисунку pic.gif из файла index.html будет images/pic.gif.

Рисунок 7. Пример размещения файлов

В примере показано несколько способов добавления рисунка на веб-страницу.

Пример: вставка изображения в документ

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Добавление рисунков</title>
 </head>
 <body>
  <p><img src="http://webimg.ru/themes/cloverfield/images/ref_collage.gif"
      alt="Это абсолютный адрес размещения изображения"></p>
  <p><img src="/example/images/home.png"
      alt="Адрес размещения изображения относительно корня сайта"></p>
  <p><img src="images/home.png"
      alt="Адрес размещения изображения относительно текущего HTML-
документа"></p>
 </body>
</html>
```

ЛИТЕРАТУРА

1. Таненбаум Э. Компьютерные сети. – 6-е изд. – СПб.: Питер, 2013. – 992 с.: ил.
2. Хомоненко А.Д. Базы данных: учеб. для высш. учеб. заведений / под ред. А.Д. Хомоненко. – 4-е изд., доп. и перераб. – СПб.: КОРОНА принт, 2004. – 736 с
3. Карпова И. П. Базы данных : курс лекций и материалы для практических занятий : учеб. пособие для студентов техн. фак., изучающих автоматизир. информ. системы и системы управления базами данных. – Санкт-Петербург [и др.]: Питер, 2013. – 240 с.
4. Адаменко Н. Д.Основы программирования на VBA: методические рекомендации по выполнению лабораторных работ / М-во образования РБ, УО «Витебский гос. ун-т им. П.М. Машерова», Каф. информатики и информационных технологий. – Витебск: УО «ВГУ им. П. М. Машерова», 2009. – 50 с. – Библиогр.: с. 50.
5. Практикум по СУБД MS ACCESS / ВГУ им. П.М. Машерова; сост. Н. Д. Адаменко. – Витебск: Изд-во ВГУ, 2001. – 92 с. – Библиогр.: с. 76.
6. Адаменко Н.Д. СУБД MS SQL Server: практикум / Н.Д. Адаменко. – Витебск: Изд-во УО «ВГУ им. П.М. Машерова», 2012. – 154 с.
7. Каммингс С. VBA для «чайников»: учеб. пособие / пер. с англ. – 2-е изд. – М.; СПб.; Киев: Диалектика, 2000. – 384 с. : ил. – Предм. указ.: с. 372–378.
8. Кузьменко В. Г. Базы данных в Visual Basic и VBA. – М.: БИНОМ, 2004. – 416 с.: ил. – (Самоучитель). – Предм. указ.: с. 409–412.
9. Мак-Федрис П. Формы, отчеты и запросы в Microsoft Access 2003 / [пер. с англ. и ред. С.А. Храмова]. – Москва [и др.]: Вильямс, 2005. – 415 с.: ил. – (Бизнес-решения). – Предм. указ.: с. 412–415.
10. Шкарина Л. Язык SQL: учебный курс. – СПб.: Питер. – 592 с.
11. Мейер Э. CSS – каскадные таблицы стилей. Подробное руководство; пер. с англ. – 3-е изд. – СПб: Символ-Плюс, 2008. – 576 .
12. Никсон Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript и CSS / [пер. с англ. Н. Вильчинский]. – 2-е изд. – Санкт-Петербург [и др.]: Питер, 2013. – 560 с.: ил., табл.

ПРИЛОЖЕНИЕ

ПРИЛОЖЕНИЕ 1

Варианты заданий по Access

Вариант 1.

1. Спроектируйте базу данных **«Абитуриент»**. Приведите ее к 3 нормальной форме. В БД должна содержаться следующая информация о приемных экзаменах:
   - Фамилия, имя, отчество абитуриента;
   - Пол;
   - Домашний адрес;
   - Специальность (МИ, МФ, ПМ);
   - 4 экзамена и их результаты.
2. Занесите в БД не менее 10 записей.
3. Сформируйте и сохраните следующие запросы:
   - Фамилии абитуриентов с перечнем предметов, по которым получены неудовлетворительные оценки.
   - Фамилии, имена и отчества абитуриентов и соответствующая сумма баллов. Отсортируйте этот список по сумме баллов.
   - Фамилии абитуриентов-женщин, проживающих в районе, который задается пользователем как параметр.
4. Подготовьте форму для внесения сведений об оценках каждого абитуриента.

Вариант 2.

1. Спроектируйте базу данных **«Турист»**. Приведите ее к 3 нормальной форме. В БД должна содержаться следующая информация:
   - Название турбазы;
   - Стоимость 1 дня для данной турбазы;
   - Типы маршрутов для данной турбазы (пеший, конный, горный и т.п.);
   - Название маршрута;
   - Фамилия, имя, отчество туриста;
   - Дата заезда;
2. Занесите в БД не менее 10 записей.
3. Сформируйте и сохраните следующие запросы:
   - Фамилии туристов в алфавитном порядке по данному маршруту.
   - Список туристов для данной турбазы со стоимостью их путевок (отсортировать по стоимости путевок);
   - Список туристов для любой турбазы (она задается пользователем как параметр), проживающих на турбазе в период от начальной до конечной даты (даты задать самостоятельно).
4. Подготовьте формы для внесения сведений о путевках и туристах.

Вариант 3.

1. Спроектируйте базу данных **«Школа»**. Приведите ее к 3 нормальной форме. В БД должна содержаться следующая информация:
   - Фамилия, имя, отчество учителя, премет преподавания, классы;
   - Классные руководители (не все учителя имеют классное руководство);
   - Фамилия, имя ученика;
   - Пол ученика;
   - Год рождения ученика.

2. Занесите в БД не менее 10 записей.
3. Сформируйте и сохраните следующие запросы:
   - Перечень классов с фамилиями их классных руководителей, отсортированный в алфавитном порядке фамилий.
   - Список учителей, работающих в заданном классе, с указанием предметов;
   - Список школьников-мальчиков с указанием их возраста (количество полных лет);
   - Список учеников любого класса (класс задается пользователем как параметр).
4. Подготовьте формы для внесения сведений о школьниках и учителях.

Вариант 4.

1. Спроектируйте базу данных **«Аэропорт»**. Приведите ее к 3 нормальной форме. В БД должна содержаться следующая информация:
   - Номер рейса
   - Тип самолета
   - Объем топливных баков
   - Расход горючего на 1000 км
   - Пункт назначения
   - Расстояние до пункта назначения
   - Дата вылета
   - Время вылета
2. Занесите в БД не менее 10 записей.
3. Сформируйте и сохраните следующие запросы:
   - Информация о рейсах и типах самолетов, вылетающих из аэропорта не позже указанной даты
   - Перечень типов самолетов, которые могут без дозаправки долететь до данного пункта назначения с указанием объема горючего на весь полет
   - Информация о рейсах в данный пункт назначения, который задается пользователем
4. Подготовьте формы для внесения сведений о рейсах и самолетах.

Вариант 5.

1. Спроектируйте базу данных **«Музыка»**. Приведите ее к 3 нормальной форме. В БД должна содержаться следующая информация:
   - Название музыкального альбома
   - Исполнитель
   - Гонорар исполнителя (в % от выручки от продажи альбома)
   - Страна исполнителя
   - Год выхода альбома
   - Цена альбома
   - Сведения по продажам альбома в различных городах за текущий год
2. Занесите в БД не менее 10 записей.
3. Сформируйте и сохраните следующие запросы:
   - Информация об исполнителях и альбомах, вышедших за 2 последних года.
   - Сведения о гонораре исполнителей за текущий год
   - Перечень альбомов исполнителей данной страны (страна задается как параметр)
4. Подготовьте формы для внесения сведений об исполнителях и альбомах.

Вариант 6.

1. Спроектируйте базу данных **«Ресторан»**. Приведите ее к 3 нормальной форме. В БД должна содержаться следующая информация:
   - Название блюда
   - Тип пищи (салат, суп и др.)
   - Описание рецепта
   - Продукты и их количество для 1 порции
   - Число порций
   - Цена 1 порции
2. Занесите в БД не менее 10 записей.
3. Сформируйте и сохраните следующие запросы:
   - Меню с перечнем блюд и их ценами для заданного типа пищи, которое отсортировано по убыванию цены.
   - Перечень продуктов с указанием их количества, необходимых для приготовления всех запланированных блюд.
   - Список блюд, которые невозможно приготовить, если отсутствует данный продукт (продукт задается пользователем).
4. Подготовьте формы для внесения сведений о блюдах ресторана.

Вариант 7.

1. Спроектируйте базу данных **«Занятия»**. Приведите ее к 3 нормальной форме. В БД должна содержаться следующая информация:
   - Фамилия, имя, отчество студента
   - Специальность (МФ, МИ, ПМ)
   - № зачетной книжки
   - Группа
   - Предметы (только лекционные курсы)
   - Преподаватели (Ф.И.О. лекторов)
   - Средний балл за последнюю сессию
2. Занесите в БД не менее 10 записей.
3. Сформируйте и сохраните следующие запросы:
   - Список студентов вашей специальности, имеющих средний балл более 4, упорядоченный по убыванию среднего балла.
   - Ф.И.О. преподавателей с перечнем предметов для данной группы, номер которой вводится как параметр.
   - Список студентов с указанием фамилии, инициалов, № группы и среднего балла с указанием размера стипендии, упорядоченный по группам. Вычислить стипендию следующим образом: средний балл*0,5*мин.зарплату.
4. Подготовьте формы для внесения сведений о студентах и занятиях.

Вариант 8.

1. Спроектируйте базу данных **«Нотариус»**. Приведите ее к 3 нормальной форме. В БД должна содержаться следующая информация:
   - Ф.И.О. владельца имущества
   - Домашний адрес владельца
   - Личное имущество (дом, гараж, автомобиль и т.п.)
   - Цена каждого вида имущества
   - Ф.И.О. наследников каждого владельца и степень их родства

   Предполагается, что каждый владелец составил завещание и выделил каждому наследнику что-то из своего имущества (например, сыну – автомобиль, дочери – дачу и т.д.)
2. Занесите в БД не менее 10 записей.

3. Сформируйте и сохраните следующие запросы:
   - Список владельцев в алфавитном порядке, имеющих дома стоимостью более некоторого значения.
   - Список наследников, которые наследуют какой-то конкретный вид имущества (он вводится как параметр).
   - Перечень всех наследников, степень их родства владельцам, наследуемое имущество, его цена и налог на него (15% стоимости).
4. Подготовьте формы для внесения сведений о владельцах и их имуществе.

Вариант 9.

1. Спроектируйте базу данных **«Кадры»**. Приведите ее к 3 нормальной форме. В БД должна содержаться следующая информация:
   - Фамилия, имя, отчество сотрудника
   - Год рождения
   - Год поступления на работу
   - Подразделение
   - Руководитель подразделения
   - Должность
   - Оклад
2. Занесите в БД не менее 10 записей.
3. Сформируйте и сохраните следующие запросы:
   - Список руководителей подразделений в алфавитном порядке, принявших работников в течение двух последних лет на конкретную должность
   - Список сотрудников с указанием года рождения, должности некоторого подразделения (оно задается как параметр
   - Ведомость на зарплату сотрудников с учетом доплаты за стаж, которая определяется следующим образом: стаж*0,01*оклад.
4. Подготовьте формы для внесения сведений о сотрудниках.

Вариант 10.

1. Спроектируйте базу данных **«Склад»**. Приведите ее к 3 нормальной форме. В БД должна содержаться следующая:
   - Наименование товара
   - Вид товара
   - Цена
   - Дата поступления товара
   - Количество поступившего товара
   - Дата отгрузки товара
   - Количество отгруженного товара
2. Занесите в БД не менее 10 записей.
3. Сформируйте и сохраните следующие запросы:
   - Список товаров с ценами, поступивших в течение предыдущего месяца
   - Список товаров данного вида (вид товара вводится как параметр)
   - Перечень отгруженного товара, отсортированного по видам товара, с указанием стоимости каждого
4. Подготовьте формы для внесения сведений о товарах.

Учебное издание

АДАМЕНКО Наталья Дмитриевна
ШЕДЬКО Василий Викторович
ОСИПОВ Александр Владимирович

ИНФОРМАЦИОННЫЕ СИСТЕМЫ И СЕТИ

Методические рекомендации

Технический редактор *Г.В. Разбоева*
Компьютерный дизайн *Т.Е. Сафранкова*

Подписано в печать .2016. Формат 60x84 $^{1}/_{16}$. Бумага офсетная.
Усл. печ. л. 2,96. Уч.-изд. л. 3,10. Тираж экз. Заказ .

Издатель и полиграфическое исполнение – учреждение образования
«Витебский государственный университет имени П.М. Машерова».

Свидетельство о государственной регистрации в качестве издателя,
изготовителя, распространителя печатных изданий
№ 1/255 от 31.03.2014 г.

Отпечатано на ризографе учреждения образования
«Витебский государственный университет имени П.М. Машерова».
210038, г. Витебск, Московский проспект, 33.