МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационных систем и технологий

ВВЕДЕНИЕ В НАПРАВЛЕНИЕ ПОДГОТОВКИ И ПЛАНИРОВАНИЕ ПРОФЕССИОНАЛЬНОЙ КАРЬЕРЫ

Методические указания к лабораторным работам
для студентов направления подготовки 02.03.03

Курск 2020

УДК 004

Составители: А.В. Киселев

Рецензент
Кандидат технических наук, доцент *О.О. Яночкина*

Введение в направление подготовки и планирование профессиональной карьеры: методические указания к лабораторным работам для студентов направлений подготовки 02.03.03/ Юго-Зап. гос. ун-т; сост.; А.В. Киселев. – Курск, 2020. - 63 с.: - ил. 28 , табл. 4.– Библиогр.: с. 63

Содержат сведения по вопросам применения современных программных средств решения профессиональных задач.

Предназначены для студентов направления подготовки 02.03.03 очной формы обучения.

Методические указания соответствуют рабочей программе дисциплины «Введение в направление подготовки и планирование профессиональной карьеры».

Текст печатается в авторской редакции

Подписано в печать 18.08.2020. Формат 60∗84 1/16.
Усл. печ. л.3,3. Уч.-изд. л. 3,1. Тираж 50 экз. Заказ 245. Бесплатно.
Юго-Западный государственный университет.
305040 Курск, ул. 50 лет Октября, 94.

Лабораторная работа 1
Обучение искусственной нейронной сети

Цель работы - практическое освоение студентами технологии разработки нейронных сетей персептронного типа.

Программа «Нейросимулятор» позволяет создавать и применять нейронные сети персептронного типа.

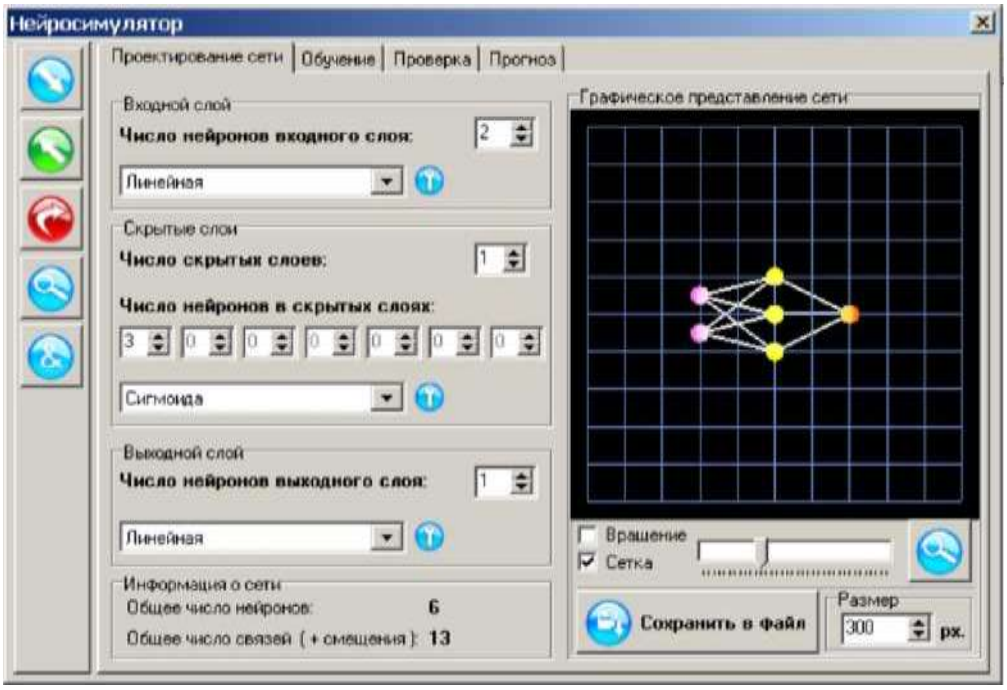


Рис. 1.1. Рабочее окно программы «Нейросимулятор»

Программа «Нейросимулятор» может работать в четырех режимах: «Проектирование сети», «Обучение», «Проверка» и «Прогноз». Переход из одного режима в другой осуществляется путем нажатия соответствующих закладок, расположенных в верхней части рабочего окна (см. рис. 1.1).

В режиме «Проектирование сети» имеется возможность изменять количество входных и выходных нейронов, количество скрытых слоев, количество нейронов в скрытых слоях, виды и параметры активационных функций. Причем, всякое изменение структуры сети автоматически отражается на ее графическом представлении.

Попробуйте поменять структуру персептрона - добавить один или два нейрона в скрытый слой, изменить количество входных и выходных нейронов. Обратите внимание на возможность

запоминания внесенных изменений с помощью клавиш, расположенных в верхнем левом углу рабочего окна: «Сохранить проект» и «Загрузить проект». Сохранение текущих состояний вашего проекта рекомендуется периодически производить на всех стадиях его выполнения.

После экспериментов с изменением структуры сети, верните ее в первоначальное состояние, нажав кнопку «Сбросить настройки». Убедившись, что ваш персептрон имеет два входа и один выход, переведите программу в режим «Обучение», нажав соответствующую закладку в верхней части рабочего окна. Теперь Ваша цель состоит в том, чтобы создать множество обучающих примеров. В нашем случае - это таблица умножения, которой нужно обучить персептрон. Нажимая несколько раз кнопку «Добавить обучающий пример» (на кнопке изображен знак «+»), введите цифры, как показано на рис. 2. Это и есть множество обучающих примеров, в данном случае - таблица умножения на 2 и на 3.

Обратите внимание, что в вашей таблице умножения не хватает одного примера: 2x5 = 10. Этот пример «забыт» специально. После того, как нейросеть будет обучена, мы спросим ее, сколько будет 2 x 5 и проверим, таким образом, ее способность к обобщению.

Обратите внимание на то, что «по молчанию» в программе «Нейросимулятор» (см. рис. 1.2) выбран алгоритм обратного распространения ошибки, что коэффициент скорости обучения установлен 0,08, а количество эпох обучения - 300, что начальные значения синаптических весов (инициализация весов) заданы случайно по стандартному закону распределения.

Нажав клавишу «Обучить сеть», вы запустите процесс обучения персептрона методом обратного распространения ошибки, который будет отображаться в виде графика зависимости максимальной (синий цвет) и среднеквадратичной (красный цвет) ошибок обучения от числа эпох обучения (см. рис. 1.3).

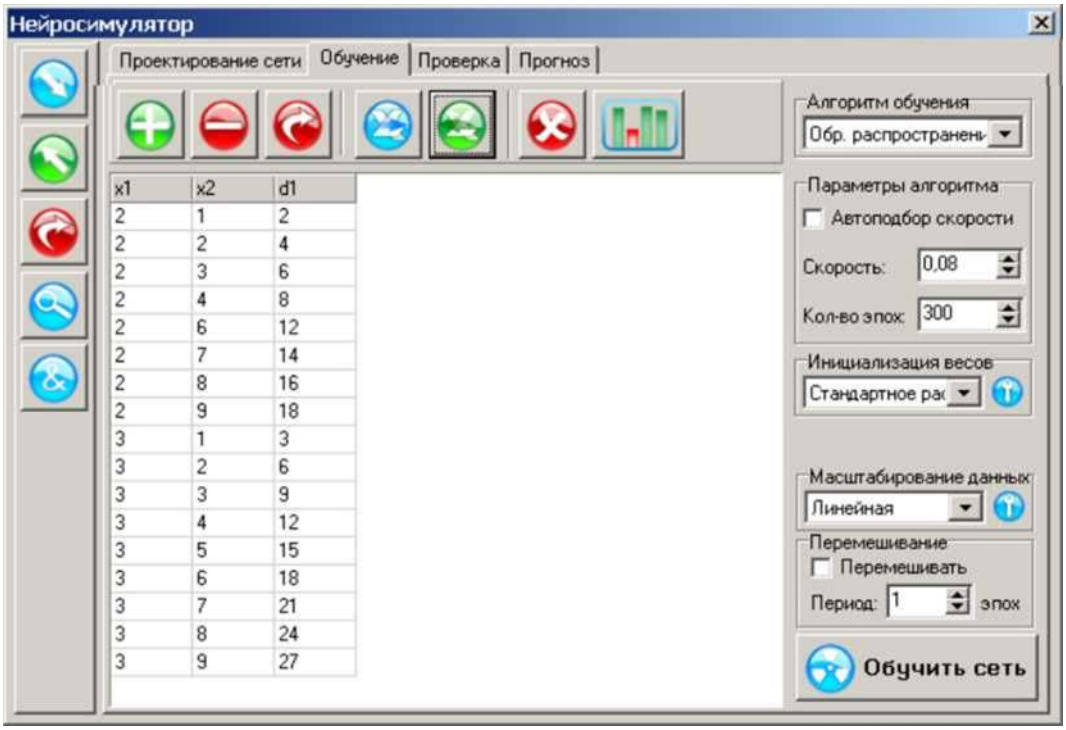


Рис. 1.2. Введено обучающее множество примеров - таблица умножения на 2 и на 3

Рис. 1.3. Графическое отображение процесса обучения сети

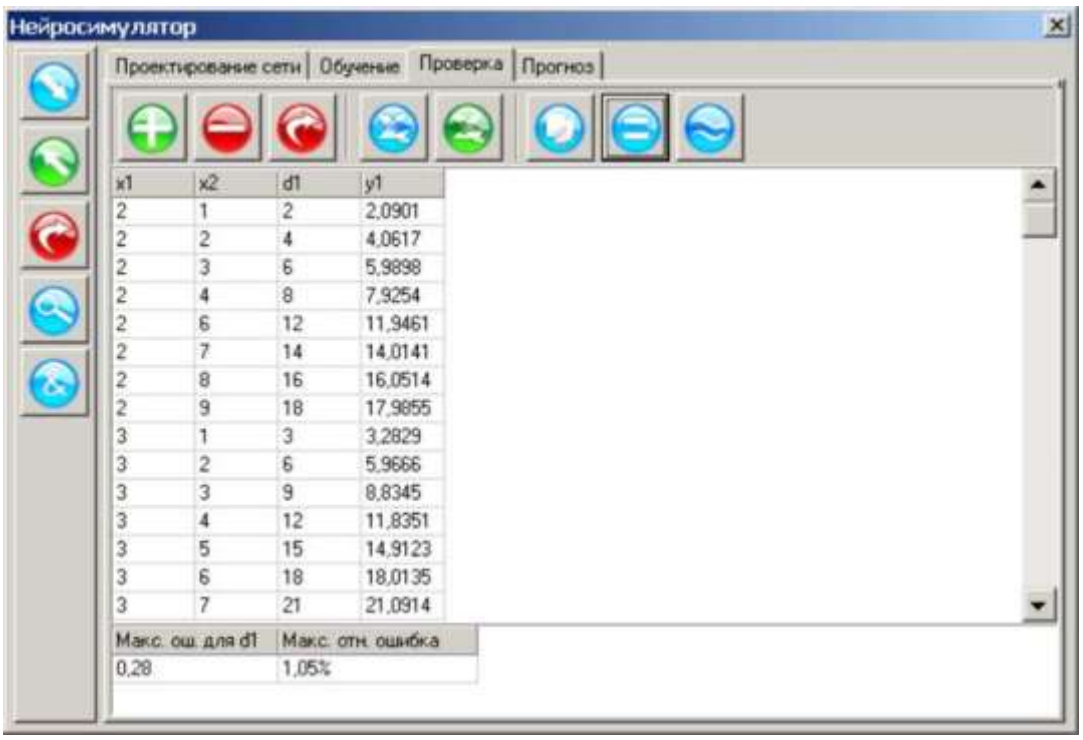


Рис. 1.4. В режиме «Проверка» можно сравнить желаемые выходы сети dlc его действительными выходами y1

После выполнения 300 эпох обучение прекращается, однако вы его можете продолжить, нажимая клавишу «Учимся», причем при каждом нажатии этой клавиши процесс обучения продляется еще на 300 эпох. Обратите внимание, что нужное количество эпох обучения (например, 1000 вместо 300) можно задать до начала процесса обучения.

Если ошибка обучения не будет снижаться, нажмите кнопку «Закрыть» и начните процесс обучения заново. Добившись снижения максимальной ошибки обучения, например, до 0,01. закройте графическое изображение (нажатием клавиши «Закрыть») и переведите программу в режим «Проверка». Нажмите кнопку «Скопировать из обучающих примеров», а затем - кнопку «Вычислить». В появившемся окне (рис. 1.4) вы можете сравнить желаемые выходы сети (dl) с тем. что насчитал обученный вами персептрон (yl). Например, как видно из первой строки, вместо того, чтобы выдать2x1 = 2, персептрон ответил: 2x1 = 2,0901. Ошибка обучения e_Lна этом примере составила: |2,0902-2| = 0,0902, что в процентах (относительно максимального значения выхода сети max(d1) = 27 составляет:$\frac{0,0902}{27} \times 100\% = 1,003\%$.

Аналогично можно подсчитать ошибку обучения на втором примере: $\frac{|4{,}0617-4|}{27} \times 100\% = 0{,}002\%$, затем на третьем и т.д., а затем выбрать максимальную по всем примерам относительную ошибку обучения персептрона. Значение этой ошибки можно увидеть в нижней части рабочего окна: 1,05%.

После того, как Вы нашли максимальную ошибку обучения персептрона, нажмите клавишу «Сбросить примеры», а затем - клавишу «Добавить строку» введите пример, которого не было в обучающем множестве: 2x5 = 10. после чего нажмите кнопку «Вычислить».

Ответ изображен на рис. 1.5. Ошибка обобщения ε_T составила 0,93%.

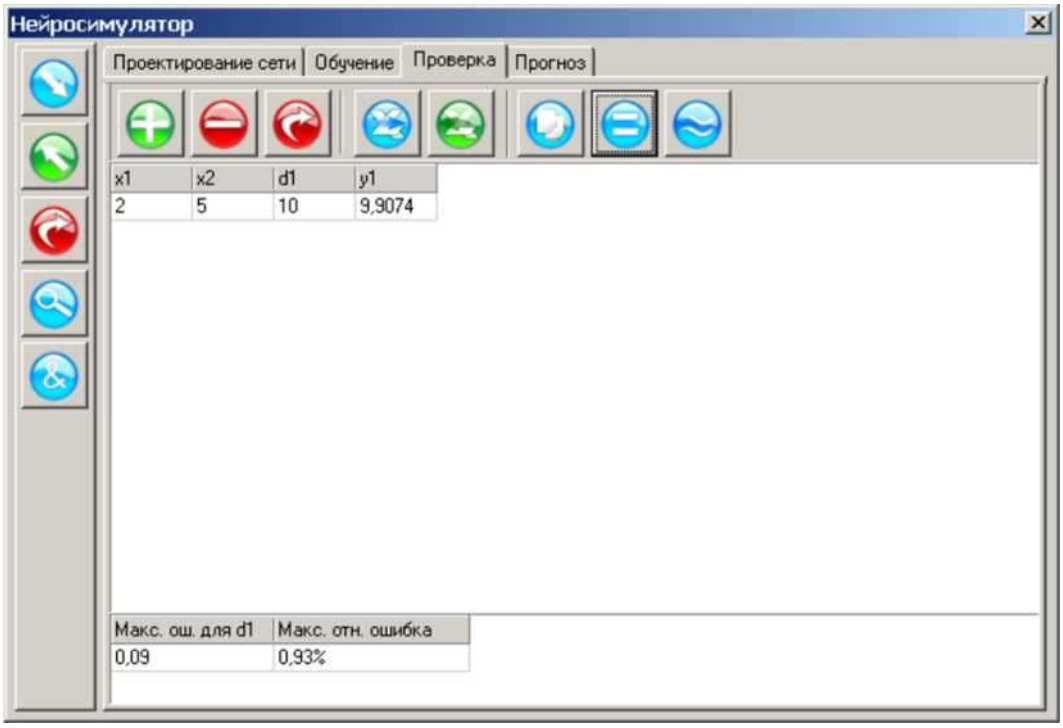


Рис. 1.5. Сеть ответила, что дважды пять будет примерно 9,9074

Более хороших результатов можно добиться, воспользовавшись следствием из теоремы Арнольда - Колмогорова - Хехт-Нильсена (формулы (3.23), (3.24) [1]). Подставляя в формулу (3.23) [1] число входных нейронов N_x= 2, число выходных нейронов N_v= 1 и количество обучающих примеров Q=18, получим оценку необходимого количества синаптических связей нейросети:

3,48 <N_w<41.

Применение формулы (3.24) [1] дает оценку необходимого количества скрытых нейронов:

$1{,}2 < N < 13{,}7$.

Как видим, интервал допустимых значений N получился весьма широким. Выберем значение N посередине полученного интервала:

$$N = \frac{1{,}2+13{,}7}{2} \approx 7.$$

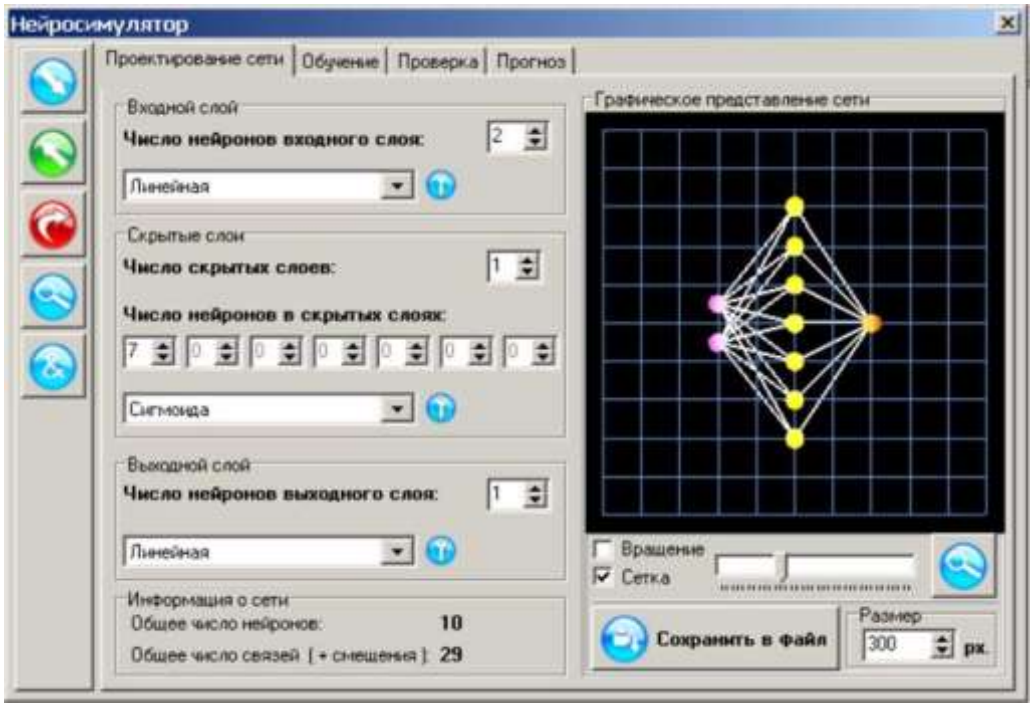


Рис. 1.6. Структура сети с семью нейронами скрытого слоя

Рис. 1.7. Графическое отображение процесса обучения сети с семью нейронами скрытого слоя

Вернувшись в режим «Проектирование сети», увеличим количество нейронов скрытого слоя до семи (рис. 1.6). Затем, выполняя обучение сети (рис. 1.7) и проверку ее прогнозирующих свойств, получаем значительно более хороший результат, приведенный на рис. 1.8 и рис. 1.9, из которых видно, что ошибка обучения сетиε_Lснизилась до 0,33%, а ошибка обобщения ε_Tснизилась до 0,11%.

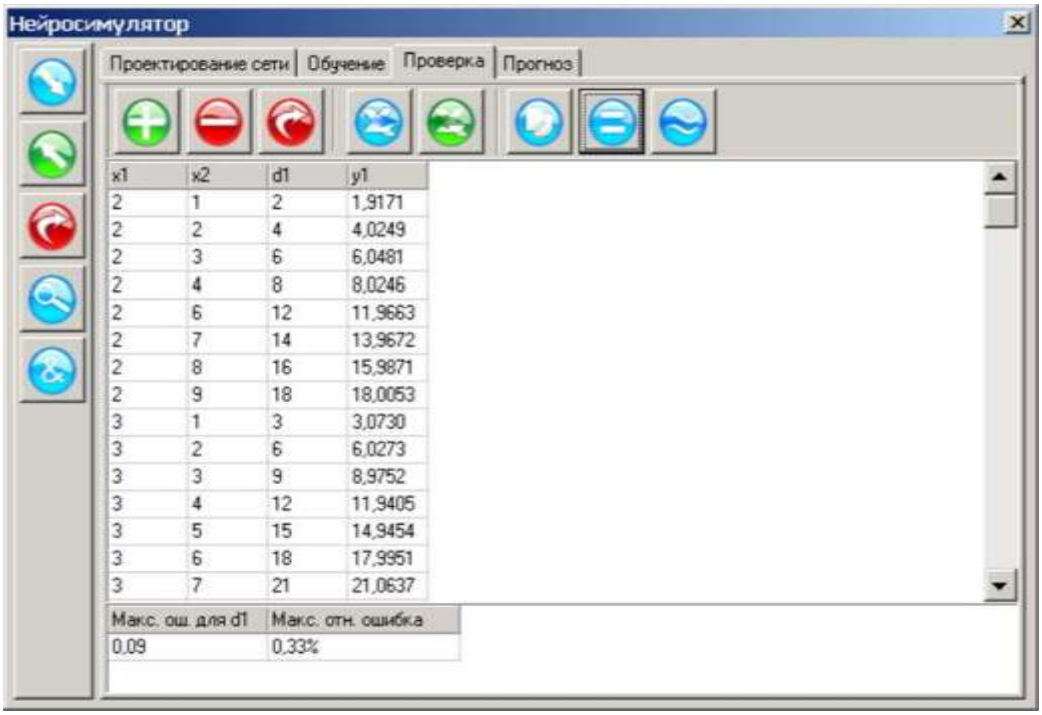


Рис. 1.8. Результат работы сети с 7-ю нейронами скрытого слоя. Ошибка обучения составила: ε_L= 0,33%

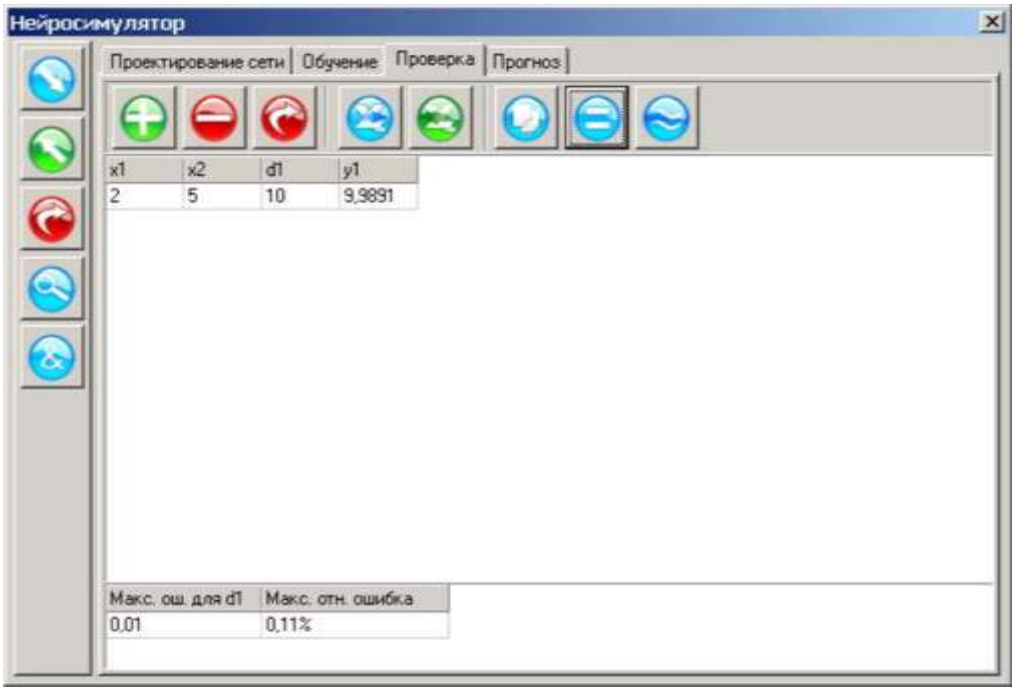
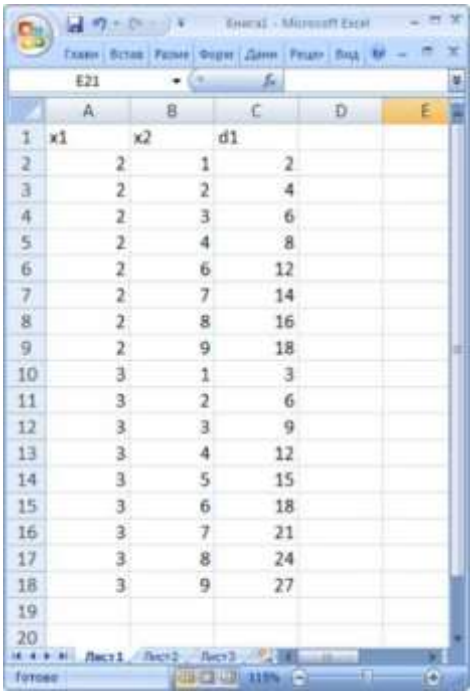


Рис. 1.9. Результат работы сети с 7-ю нейронами скрытого слоя. Ошибка обобщения составила: ε_T = 0,11%

И в заключение, нажмите кнопку «Вычислить и округлить». Результат будет следующим: 2x5 = 10,0. Как помните, этого примера не было в обучающем множестве примеров. Персептрон сам «догадался», сколько будет дважды пять, блестяще проявив свои обобщающие свойства, которые он унаследовал от своего прототипа - человеческого мозга [1].

В программе «Нейросимулятор» предусмотрена возможность обмена данными с приложением MicrosoftExcel. Для этого в режимах «Обучение», «Проверка» и «Прогноз» имеются кнопки «Загрузить из Excel-файла» и «Отправить в Excel-файл». Благодаря этому, имеется возможность использовать редактор MicrosoftExcel, в частности - готовить в нем обучающие и тестирующие примеры, а также обрабатывать результаты нейропрогнозов, используя графические возможности MicrosoftExcel.

Например, множество обучающих примеров таблицы умножения, приведенное на рис. 1.2, можно было предварительно подготовить в редакторе MicrosoftExcel(см. рис. 1.10), а затем загрузить его в программу «Нейрсимулятор» в режиме «Обучение» нажав клавишу «Загрузить из Excel-файла».



	A	B	C	D	E
1	x1	x2	d1		
2	2	1	2		
3	2	2	4		
4	2	3	6		
5	2	4	8		
6	2	6	12		
7	2	7	14		
8	2	8	16		
9	2	9	18		
10	3	1	3		
11	3	2	6		
12	3	3	9		
13	3	4	12		
14	3	5	15		
15	3	6	18		
16	3	7	21		
17	3	8	24		
18	3	9	27		
19					
20					

Рис. 1.10. Множество обучающих примеров таблицы умножения на 2 и на 3, подготовленное в редакторе MicrosoftExcel

Режим «Прогноз» отличается от режима «Проверка» тем, что в нем не требуется указывать желаемый выходные сигналы персептрона. Соответственно, не вычисляется и разница между желаемыми и действительными значениями (погрешность). В этом режиме персептрон выполняет прогнозирование - вычисляет выходные значения, используя вводимые пользователем компоненты входного вектора X.

Более подробную инструкцию по работе с программой «Нейросимулятор», а также примеры ее применения для решения широкого круга практических задач вы найдете в книгах [1] и [2].

Контрольные вопросы

1. В каких режимах может работать программа «Нейросимулятор»?
2. Какие алгоритмы обучения реализованы в программе «Нейросимулятор»?
3. Как определить максимальную ошибку обучения персептрона?
4. Как определяется оптимальное количество нейронов в скрытом слое?

5. В чем заключается отличие режима «Прогноз» от режима «Проверка»?

Лабораторная работа 2
Основы разработки баз данных

Знакомство с Access. Создание таблиц

Теоретический материал.

База данных (БД) - упорядоченная совокупность данных, предназначенных для хранения, накопления и обработки с помощью ЭВМ. Для создания и ведения баз данных (их обновления, обеспечения доступа по запросам и выдачи данных по ним пользователю) используется набор языковых и программных средств, называемых *системой управления базами данных (СУБД).*

К объектам базы данных Access относятся:

1. *Таблицы* - предназначены для упорядоченного хранения данных.
2. *Запросы* - предназначены для поиска, извлечения данных и выполнения вычислений.
3. *Формы* - предназначены для удобного просмотра, изменения и добавления данных в таблицах.
4. *Отчеты* - используются для анализа и печати данных.
5. *Страницы доступа к данным* - предназначены для просмотра, ввода, обновления и анализа данных через сеть или из любого места компьютера.
6. *Макросы* - используются для выполнения часто встречающегося набора макрокоманд, осуществляющих обработку данных.
7. *Модули* - предназначены для описания инструкций и процедур на языке VBA.

Основным объектом базы данных является таблица, которая состоит из записей (строк) и полей (столбцов). На пересечении записи и поля образуется ячейка, в которой содержатся данные.

Каждому полю таблицы присваивается *уникальное имя*, которое не может содержать более 64 символов. В каждом поле содержатся данные одного типа.

Таблица 1. Типы данных

Тип	**Описание**

Текстовый	Используется для хранения символьных или числовых данных, не требующих вычислений. В свойстве *Размер поля* задается максимальное количество символов, которые могут быть введены в данное поле. По умолчанию размер устанавливается в 50 знаков. Максимальное количество символов, которые могут содержаться в текстовом поле, - 255
Поле МЕМО	Предназначено для ввода текстовой информации, по объему превышающей 255 символов; может содержать до 65 536 символов
Числовой	Предназначен для хранения числовых данных, используемых в математических расчетах. На вкладках *Общие* и *Подстановка* можно установить свойства числового поля, среди которых *Размер поля, Формат поля, Число десятичных знаков*
Дата/Время	Используется для представления даты и времени. Выбор конкретного формата даты или времени устанавливается в свойстве *Формат даты*
Денежный	Предназначен для хранения данных, точность представления которых колеблется от 1 до 4 знаков после запятой. Целая часть может содержать до 15 десятичных знаков
Счетчик	Предназначен для автоматической вставки уникальных последовательных (увеличивающихся на 1) или случайных чисел в качестве номера новой записи. Номер, присвоенный записи, не может быть удален или изменен. Поля с этим типом данных используются в качестве ключевых полей таблицы
Логический	Предназначен для хранения одного из двух значений, интерпретируемых как «Да / Нет», «Истина / Ложь», «Вкл. / Выкл.»

Продолжение таблицы 1

Тип	**Описание**

Поле объекта OLE	Содержит данные, созданные в других программах, которые используют протокол OLE. Это могут быть, например, документы Word, электронные таблицы Excel, рисунки, звуковые и видеозаписи и др. Объекты OLE связываются с базой данных Access или внедряются в нее. Сортировать, группировать и индексировать поля объектов OLE нельзя
Гиперссылка	Специальный тип, предназначенный для хранения гиперссылок
Мастер подстановок	Предназначен для автоматического определения поля. С его помощью будет создано поле со списком, из которого можно выбирать данные, содержащиеся в другой таблице или в наборе постоянных значений

Пример создания базы данных.

1. Запустите Microsoft Access 2007.
2. Создайте новую базу данных
3. Задайте имя новой базы данных - «Записная книжка.accdb».
4. На вкладке ленты *Создание* в панели инструментов *Таблицы* нажмите на кнопку *Конструктор таблиц.*
5. Введите имена полей и укажите типы данных, к которым они относятся:

Имя поля	Тип данных
№ п/п	Счетчик
Фамилия	Текстовый
Имя	Текстовый
Адрес	Текстовый
Индекс	Числовой
Телефон	Текстовый
Хобби	Текстовый
Эл. почта	Гиперссылка

6. Выйдите из режима Конструктора, предварительно сохранив таблицу под именем *«Друзья»*; ключевые поля не задавайте.
7. Откройте таблицу *«Друзья»* двойным щелчком мыши и заполните в ней 10 строк.
8. Добавьте поля *«Отчество»* и *«Дата рождения»,* для этого:

1) установите курсор на поле, перед которым нужно вставить новый столбец;
2) выполните команду: вкладка ленты *Режим таблицы* - панель инструментов *Поля и столбцы - Вставить;*
3) щелкнув два раза на *Поле1,* переименуйте его в *«Отчество»,* а *Поле 2 - «Дата рождения».*
9. Перейдите в режим Конструктора командой: вкладка ленты *Главная - Режимы - Конструктор.*
10. Для поля *«Дата рождения»* установите тип данных *Дата / время*; в свойствах поля выберите *Краткий формат даты.*
11. Отформатируйте таблицу следующим образом:
4) цвет сетки - темно-красный;
5) цвет фона - голубой;
6) цвет текста - темно-красный, размер - 12 пт, начертание - курсив.
12. Переименуйте поле *«Хобби»* в *«Увлечения».*
13. Удалите запись под номером 8.
14. Измените размер ячеек так, чтобы были видны все данные. Для этого достаточно два раза щелкнуть левой кнопкой мыши на границе полей.
15. Расположите поля в следующем порядке: *«№п/п», «Фамилия», «Имя», «Отчество», «Телефон», «Дата рождения», «Увлечения», «Адрес», «Индекс», «Фото», «Эл почта».*
16. Заполните пустые ячейки таблицы.
17. В режиме Конструктора добавьте поле *«Семейное положение»*, в котором будет содержаться фиксированный набор значений - замужем, не замужем, женат, не женат. Для создания раскрывающегося списка будем использовать *Мастер подстановок:*
7) установите тип данных *Мастер подстановок;*
8) в появившемся диалоговом окне выберите строку *«Будет введен фиксированный набор значений»* и нажмите кнопку *Далее*,
9) число столбцов - 1;
10) введите данные списка - замужем, не замужем, женат, не женат;
11) нажмите кнопку *Готово.*

18. С помощью раскрывающегося списка заполните новый столбец. Поскольку таблица получилась широкая, то при заполнении данного столбца возникают некоторые неудобства: не видно фамилии человека, для которого заполняется поле *«Семейное положение»*. Чтобы фамилия была постоянно видна при заполнении таблицы, необходимо воспользоваться командой *Закрепить столбцы* из контекстного меню поля *«Фамилия»*.

Создание связей между таблицами

Пример создания связей между таблицами:

1. Запустите Microsoft Access 2007.
2. Создадим базу данных «Фирма». Сотрудники данной организации работают с клиентами и выполняют их заказы.

Если все сведения поместить в одной таблице, то она станет очень неудобной для работы. В ней начнутся повторы данных. Всякий раз, когда сотрудник Иванов будет работать с какой-либо фирмой, придется прописывать данные о сотруднике и клиенте заново, в результате чего можно допустить множество ошибок. Чтобы уменьшить число ошибок, можно исходную таблицу разбить на несколько таблиц и установить связи между ними. Это будет более рационально, чем прежде.

Таким образом, необходимо создать 3 таблицы: *Сотрудники, Клиенты* и *Заказы.*

Таблица 2. Сотрудники

Имя поля	Тип данных
Код сотрудника	Счетчик
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый
Должность	Текстовый
Телефон	Текстовый
Адрес	Текстовый
Дата рождения	Дата/Время
Заработная плата	Денежный
Фото	Объект OLE
Эл. почта	Гиперссылка

Таблица 3. Клиенты

Имя поля	Тип данных
Код клиента	Счетчик
Название компании	Текстовый
Адрес	Текстовый
Номер телефона	Текстовый
Факс	Числовой
Адрес электронной почты	Гиперссылка
Заметки	Поле MEMO

Таблица 4. Заказы

Имя поля	Тип данных
Код заказа	Счетчик
Код клиента	Числовой
Код сотрудника	Числовой
Дата размещения	Дата/Время
Дата исполнения	Дата/Время
Сумма	Денежный
Отметка о выполнении	Логический

3. Отдельные таблицы, содержащие информацию по определенной теме, необходимо связать в единую структуру базы данных. Для связывания таблиц следует задать *ключевые поля.* Ключ состоит из одного или нескольких полей, значения которых *однозначно* определяют каждую запись в таблице. Наиболее подходящим в качестве ключевого поля является *«Счетчик»,* так как значения в данном поле являются уникальными (т. е. исключают повторы).
4. Откройте таблицу *Сотрудники* в режиме Конструктора.
5. Нажмите правой кнопкой мыши на поле *Код сотрудника* и в появившемся контекстном меню выберите команду *Ключевое поле.* Если в таблице необходимо установить несколько ключевых полей, то выделить их можно, удерживая клавишу *Ctrl.*
6. Для таблицы *Клиенты* установите ключевое поле *Код клиента*, а для таблицы *Заказы Код заказа.*
7. Таблица *Заказы* содержит поля *Код сотрудника* и *Код клиента.* При их заполнении могут возникнуть некоторые трудности, так как не всегда удается запомнить все предприятия, с которыми работает фирма, и всех сотрудников с номером кода. Для удобства молено создать раскрывающиеся списки с помощью *Мастера подстановок.*
8. Откройте таблицу *Заказы* в режиме Конструктора.
9. Для поля *Код сотрудника* выберите тип данных *Мастер подстановок.*
10. В появившемся окне выберите команду «Объект «*столбец подстановки» будет использовать значения из таблицы или запроса*» и щелкните на *кнопке Далее* .
11. В списке таблиц выберите таблицу *Сотрудники* и щелкните на кнопке *Далее.*
12. В списке *Доступные поля* выберите поле *Код сотрудника* и щелкните на кнопке со стрелкой, чтобы ввести поле в список *Выбранные поля.* Таким же образом добавьте поля *Фамилия* и *Имя* и щелкните на кнопке *Далее.*

13. Выберите порядок сортировки списка по полю *Фамилия.*
14. В следующем диалоговом окне задайте необходимую ширину столбцов раскрывающегося списка.
15. Установите флажок *Скрыть ключевой столбе»* и нажмите кнопку *Далее*.
16. На последнем шаге *Мастера подстановок* замените при необходимости надпись для поля подстановок и щелкните на кнопке *Готово.*
17. Аналогичным образом создайте раскрывающийся список для поля *Код клиента*.
18. После создания ключевых полей можно приступить к созданию связей. Существует несколько типов отношений между таблицами:

1) при отношении *«один-к-одному»* каждой записи ключевого поля в первой таблице соответствует только одна запись в связанном поле другой таблицы, и наоборот. Отношения такого типа используются не очень часто. Иногда их можно использовать для разделения таблиц, содержащих много полей, для отделения части таблицы по соображениям безопасности;

2) при отношении *«один-ко-многим»* каждой записи в первой таблице соответствует несколько записей во второй, но запись во второй таблице не может иметь более одной связанной записи в первой таблице;

3) при отношении *«многие-ко-многим»* одной записи в первой таблице могут соответствовать несколько записей во второй таблице, а одной записи во второй таблице могут соответствовать несколько записей в первой.

19. Закройте все открытые таблицы, так как создавать или изменять связи между открытыми таблицами нельзя.
20. Выполните команду: вкладка ленты *Работа с базами данных - Схема данных*
21. Если ранее никаких связей между таблицами базы не было, то при открытии окна *Схема данных* одновременно открывается окно *Добавление таблицы*, в котором выберите таблицы *Сотрудники, Клиенты* и *Заказы.*
22. Если связи между таблицами уже были заданы, то для добавления в схему данных новой таблицы щелкните

правой кнопкой мыши на схеме данных и в контекстном меню выберите пункт *Добавить таблицу.*

23. Установите связь между таблицами *Сотрудники* и *Заказы*, для этого выберите поле *Код сотрудника* в таблице *Сотрудники* и перенесите его на соответствующее поле в таблице *Заказы.*
24. После перетаскивания откроется диалоговое окно *Изменение связей* (рис. 2.1), в котором включите флажок *Обеспечение условия целостности.* Это позволит предотвратить случаи удаления записей из одной таблицы, при которых связанные с ними данные других таблиц останутся без связи.

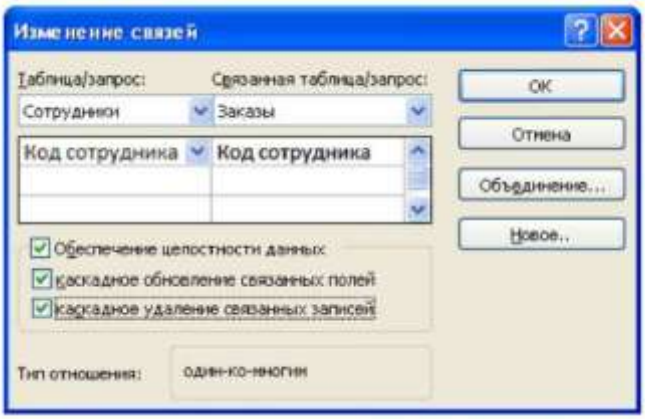


Рисунок 2.1 - Создание связи между таблицами

25. 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.
26. Параметры связи молено изменить, нажав на кнопку *Объединение.*
27. После установления всех необходимых параметров нажмите кнопку *ОК.*
28. Связь между таблицами *Клиенты* и *Заказы* установите самостоятельно.
29. В результате должна получиться схема данных, представленная на рис. 2.2.

Рисунок 2.2 - Схема данных

В приведенном примере используются связи «один-ко-многим». На схеме данных они отображаются в виде соединительных линий со специальными значками около таблиц. Связь «один-ко-многим» помечается «1» вблизи главной таблицы (имеющей первичный ключ) и «со» вблизи подчиненной таблицы (имеющей внешний ключ). Связь «один-к-одному» помечается двумя «1» (оба поля таблиц имеют первичные ключи). Неопределенная связь не имеет никаких знаков. Если установлено объединение, то его направление отмечается стрелкой на конце соединительной линии (ни одно из объединенных полей не является ключевым и не имеет уникального индекса).

30. В таблицу *Сотрудники* внесите данные о семи работниках.
31. В таблицу *Клиенты* внесите данные о десяти предприятиях, с которыми работает данная фирма.
32. В таблице *Заказы* оформите несколько заявок, поступивших на фирму.

Отбор данных с помощью запросов

Запросы являются основным средством просмотра, отбора, изменения и анализа информации, которая содержится в одной или нескольких таблицах базы данных.

Существуют различные виды запросов, но наиболее распространенными являются *запросы на выборку*, с них и начнем наше знакомство.

Пример выполнения отбора данных с помощью запросов:

1. Откройте базу данных *«Фирма»,* созданную ранее.
2. Выполните команду: вкладка ленты *Создание - Мастер запросов - Простой запрос.*
3. В появившемся диалоговом окне (рис. 2.3) укажите таблицу *Сотрудники* и выберите поля *Фамилия, Имя, Телефон.* Нажмите кнопку *Далее.*

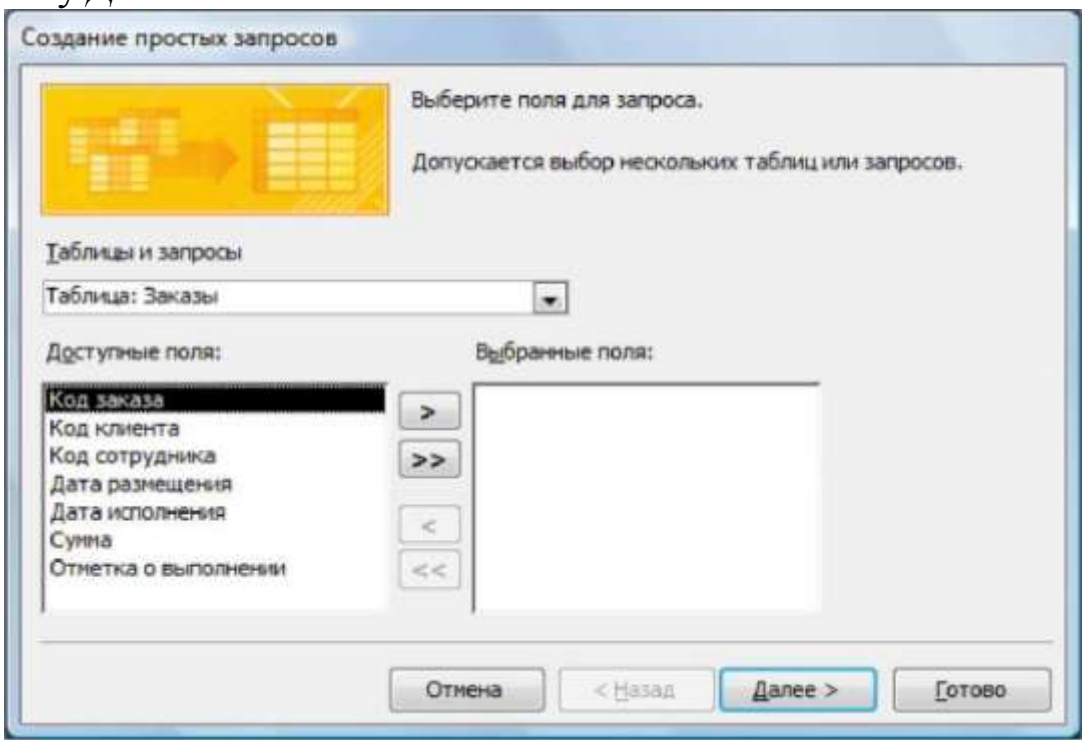


Рисунок 2.3 - Создание простого запроса

4. Введите имя запроса - *Телефоны* - и нажмите кнопку *Готово.* Перед вами появится запрос, в котором можно просмотреть телефоны сотрудников.
5. Следующий запрос попробуйте создать с помощью Конструктора, для этого выполните команду: вкладка ленты *Создание - Конструктор запросов.*
6. В диалоговом окне *Добавление таблиц* выберите таблицу *Клиенты* и щелкните на кнопке *Добавить,* а затем - на кнопке *Закрыть.*
7. Чтобы перенести нужные поля в бланк запроса, необходимо по ним дважды щелкнуть левой кнопкой мыши (рис. 2.4).

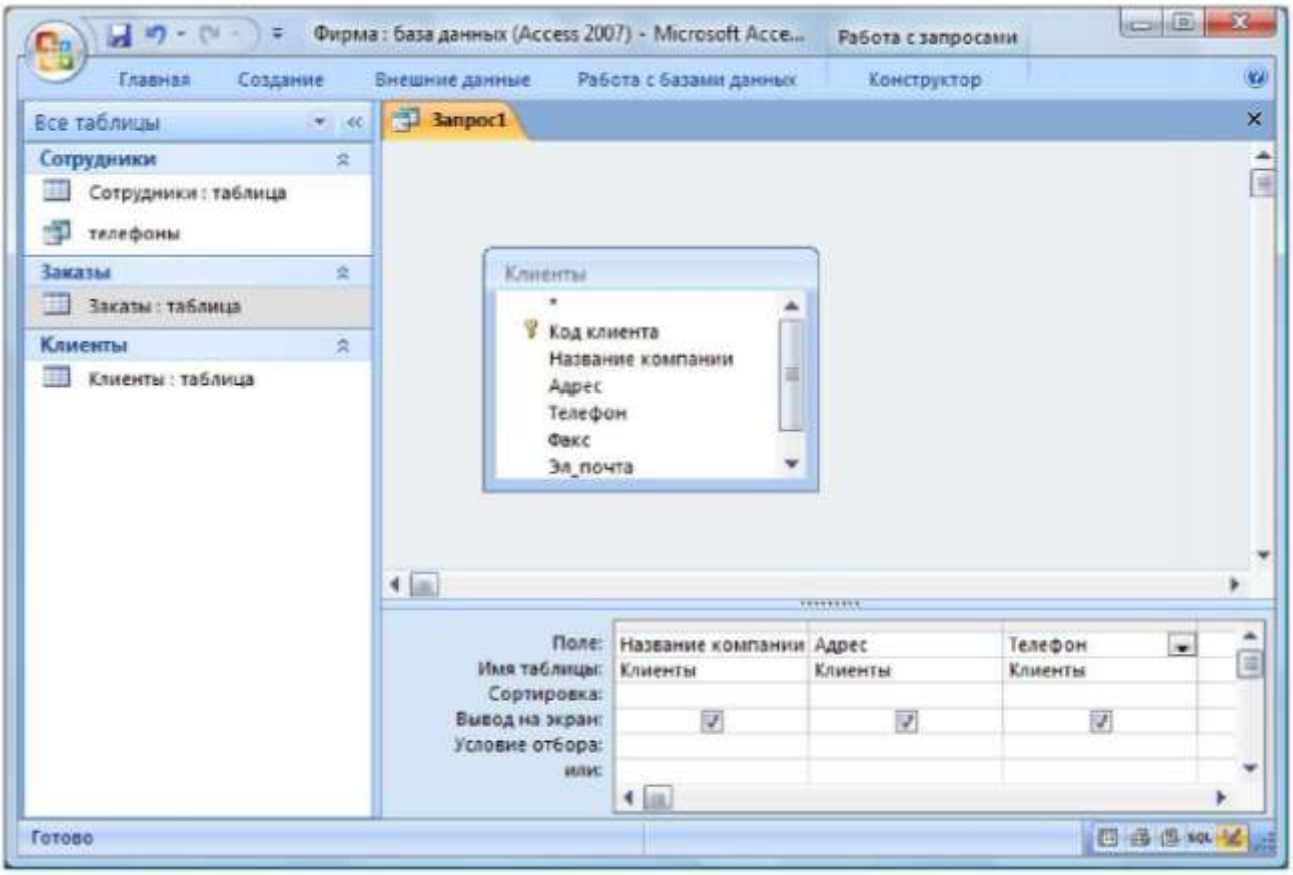


Рисунок 2.4 - Создание запроса в режиме Конструктора

8. Чтобы отсортировать записи в поле *Название компании* в алфавитном порядке, необходимо в раскрывающемся списке строки *Сортировка* выбрать пункт *по возрастанию.*
9. Сохраните запрос с именем *«Адреса клиентов».*
10. Самостоятельно создайте запрос *«Дни рождения»,* в котором можно будет просмотреть дни рождения сотрудников.
11. Допустим, мы хотим узнать, у кого из сотрудников день рождения в текущем месяце, например в апреле. Для этого откройте запрос в режиме Конструктора.
12. В строке *Условие отбора* для поля *«Дата рождения»* введите значение **.04.**. В данной записи * означают, что дата и год рождения могут быть любыми, а месяц 4-м (т. е. апрель). После этого окно запроса должно выглядеть так, как оно представлено на рис. 2.5.

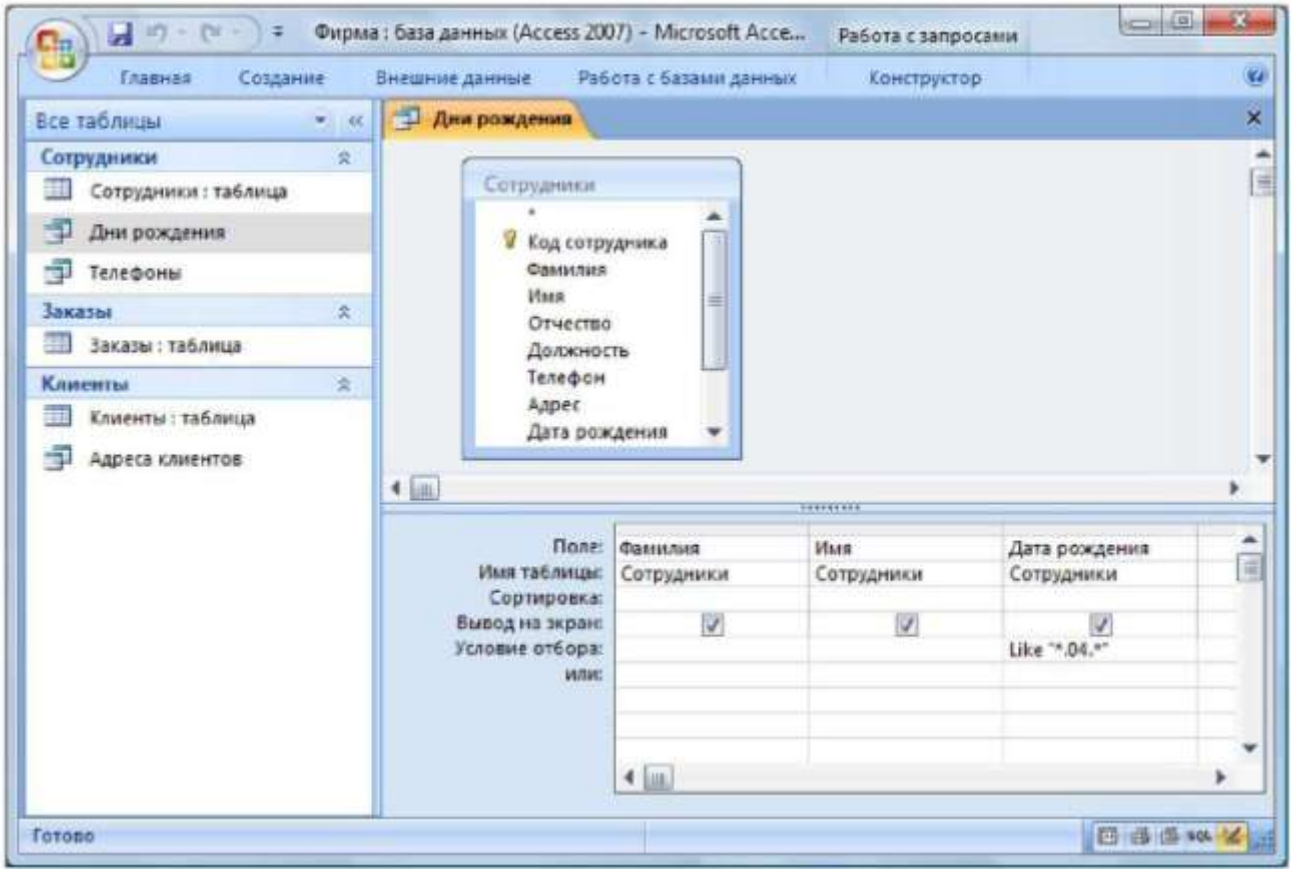


Рисунок 2.5 - Создание запроса

13. Закройте Конструктор и просмотрите полученный результат. Если в запросе *Дни рождения* нет ни одной записи, значит, в таблице *Сотрудники* нет ни одного человека, родившегося в апреле. Добавьте в таблицу *Сотрудники* несколько человек, родившихся в апреле, и посмотрите, как изменится запрос. Запросы автоматически обновляются при каждом открытии.
14. Если нам нужно узнать, кто из сотрудников родился в мае, то придется создать новый запрос или изменить условие в существующем запросе *Дни рождения.* Данная процедура является неудобной и занимает много времени. Если приходится часто выполнять запрос, но каждый раз с новыми значениями условий используют *запрос с параметром.* При запуске такого запроса на экран

выводится диалоговое окно для ввода значения в качестве условия отбора. Чтобы создать запрос с параметром, пользователю необходимо ввести текст сообщения в строке *Условие отбора* бланка запроса (рис. 2.6).

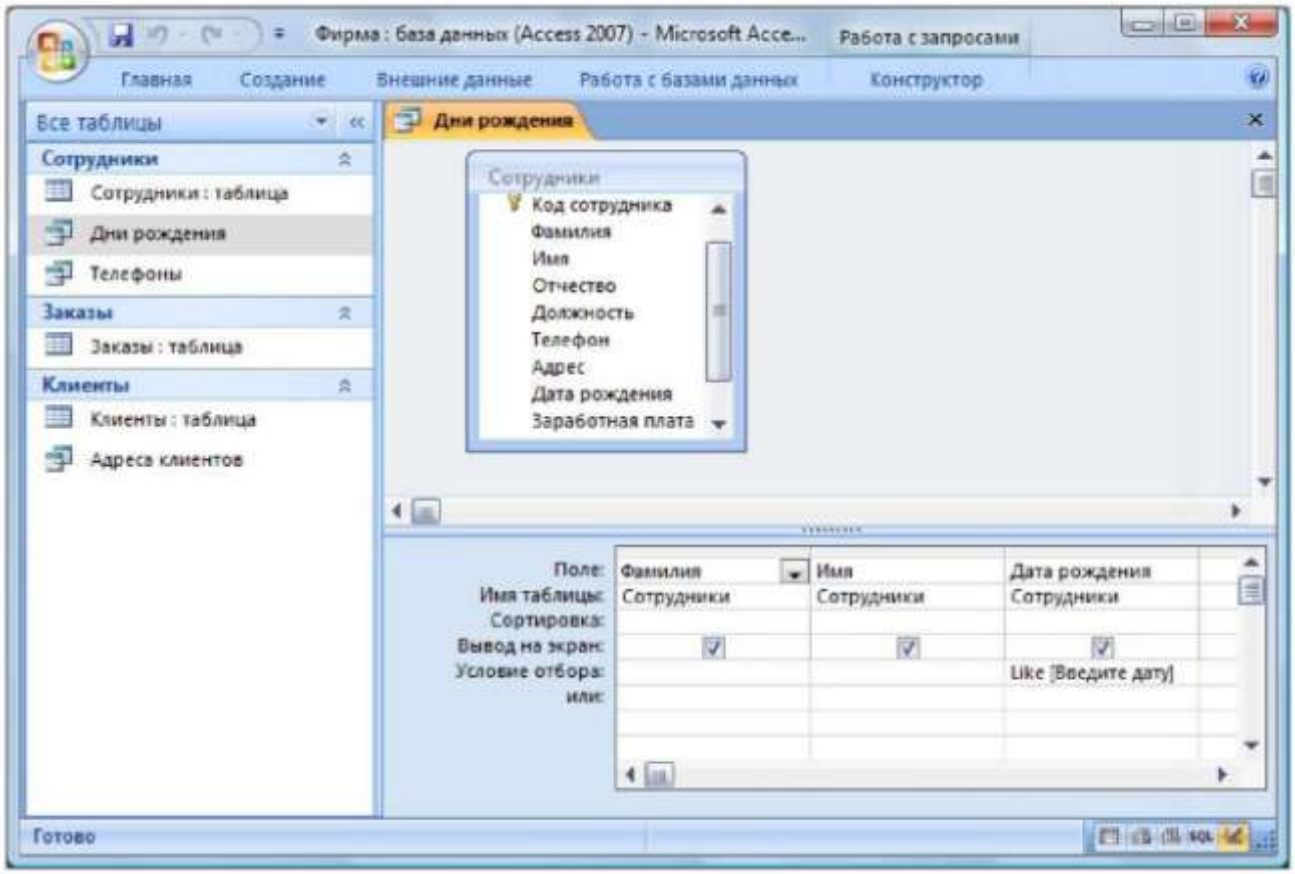


Рисунок 2.6 - Создание запроса с параметром

15. Запись *Like [Введите дату]* означает, что при открытии запроса появится диалоговое окно (рис. 2.7) с текстом *«Введите дату»* и полем для ввода условия отбора. Если ввести условие *.04.*, то в запросе появится список сотрудников, родившихся в апреле. Запустите запрос еще раз и введите значение * *05.* *, посмотрите, как изменился запрос.

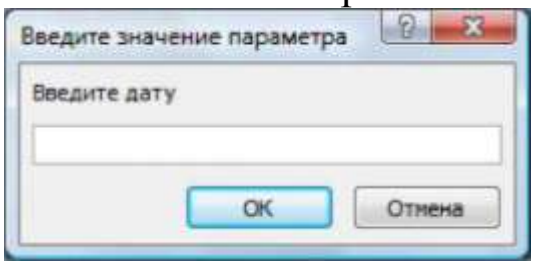


Рисунок 2.7 - Окно для ввода условия отбора

16. Измените запрос *«Телефоны»* так, чтобы при его запуске выводилось диалоговое окно с сообщением *«Введите фамилию».* Поскольку в запросе нужно вывести конкретную фамилию, в условии отбора слово *Like* писать не надо.

17. Измените запрос *«Телефоны»* так, чтобы при его запуске запрашивались не только фамилия, но и имя сотрудника.
18. Самостоятельно создайте запрос *«Выполненные заказы»*, содержащий следующие сведения: фамилия и имя сотрудника, название компании, с которой он работает, отметка о выполнении и сумма заказа. Данные запроса возьмите из нескольких таблиц.
19. В условии отбора для логического поля *Отметка о выполнении* введите Да, чтобы в запросе отображались только выполненные заказы.
20. Сделайте так, чтобы столбец *Отметка о выполнении* не выводился на экран.
21. Создайте запрос *Сумма заказа*, в котором будут отображаться заказы на сумму более *50000 р.*
22. Измените запрос, чтобы сумма заказа была от *20000 до 50000 р.* Для данных запросов в условии отбора можно использовать операторы сравнения и логические операторы *And., Or, Not* и др.
23. Иногда в запросах требуется произвести некоторые вычисления, например посчитать подоходный налог *13 %* для каждой сделки. Для этого откройте запрос *Сумма заказа* в режиме Конструктора.
24. В пустом столбце бланка запроса щелкните правой кнопкой мыши на ячейке *Поле* и в появившемся контекстном меню выберите команду *Построить.* Перед вами появится окно *Построитель выражении* (рис. 2.8), который состоит из трех областей: поля выражения, кнопок операторов и элементов выражения. Сверху располагается поле выражения, в котором оно и создается. Вводимые в это поле элементы выбираются в двух других областях окна Построителя.
25. В левом списке откройте папку *Запросы* и выделите запрос *Сумма заказа.* В среднем списке выделите поле *Сумма* и нажмите кнопку *Вставить.*

Идентификатор этого поля появится в поле выражения Построителя.

26. Щелкните на кнопке * и введите 0,13 (см. рис. 2.8). Таким образом, мы посчитаем подоходный налог 13 %.

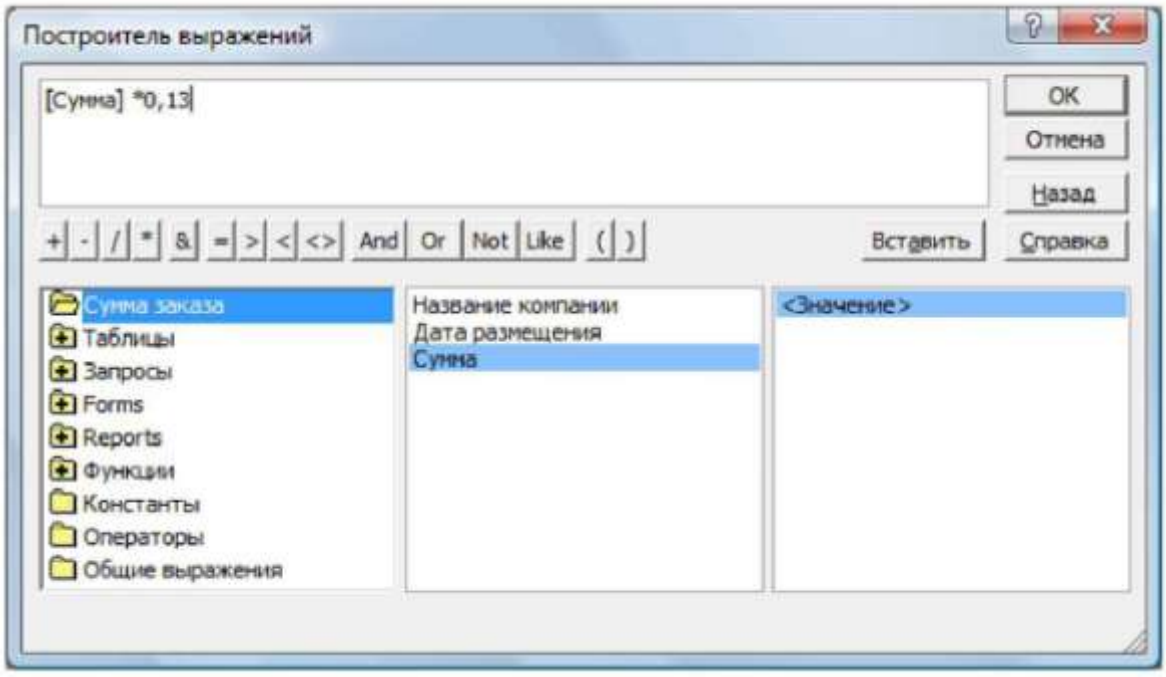


Рисунок 2.8 - Построитель выражений

27. Нажмите кнопку *ОК,* после чего в ячейке свойства *Поле* появится значение *«Выражение 1: [Сумма] *0,13 ».*
28. Замените *Выражение1* на *Налог* и закройте Конструктор.
29. Откройте запрос и посмотрите, что у вас получилось.
30. Используя *Построитель выражений*, добавьте в запрос *Сумма заказа* поле *Прибыль*, в котором будет вычисляться доход от заказа (т. е. сумма минус налог).
31. Создайте запрос *Менеджеры*, с помощью которого в таблице *Сотрудники* найдите всех менеджеров фирмы.

Использование форм в базе данных

Формы - это объекты базы данных, предназначенные для просмотра данных из таблиц и запросов, для ввода данных в базу, корректирования существующих данных и выполнения заданных действий. Форма может содержать графики, рисунки и другие внедренные объекты.

Можно вносить данные в таблицы и без помощи каких-либо форм. Но существует несколько причин, которые делают формы незаменимым средством ввода данных в базу:

1) при работе с формами ограничен доступ к таблицам (самому ценному в базе данных);
2) разные люди могут иметь разные права доступа к информации, хранящейся в базе. Для ввода данных им предоставляются разные формы, хотя данные из форм могут поступать в одну таблицу;
3) вводить данные в форму легче, чем в таблицу, и удобнее, так как в окне формы отображается, как правило, одна запись таблицы;
4) в большинстве случаев информация для баз данных берется из бумажных бланков (анкет, счетов, накладных, справок и т. д.). Экранные формы можно сделать точной копией бумажных бланков, благодаря этому уменьшается количество ошибок при вводе и снижается утомляемость персонала.

Создавать формы можно на основе нескольких таблиц или запросов с помощью Мастера, используя средство автоформы, «вручную» в режиме Конструктора, сохраняя таблицу или запрос как форму. Созданную любым способом форму можно затем изменять в режиме Конструктора. Рассмотрим некоторые из перечисленных способов.

1. Выполните команду: вкладка ленты *Создание* - панель инструментов *Формы* - *Другие формы* - *Мастер форм.*
2. В диалоговом окне *Создание форм* выберите таблицы (запросы) и поля, которые будут помещены в форму. Щелкните по кнопке *Далее.*
3. В следующих диалоговых окнах мастера выберите внешний вид формы, стиль,

задайте имя формы. Щелкните по кнопке *Готово.*

4. С помощью Мастера создайте формы *Сотрудники, Клиенты, Заказы, Менеджеры.*
5. Откройте форму *Сотрудники* в режиме Конструктора. Этот режим предназначен для создания и редактирования форм.
6. Разместите элементы в удобном для вас порядке, измените размер и цвет текста.
7. В заголовок формы добавьте текст *Сотрудники фирмы.*
8. В примечание формы добавьте объект *Кнопка* (вкладка ленты *Конструктор* - панель инструментов *Элементы управления).*
9. После того как вы «нарисуете» кнопку указателем, на экране появится диалоговое окно *Создание кнопок* (рис. 2.9).

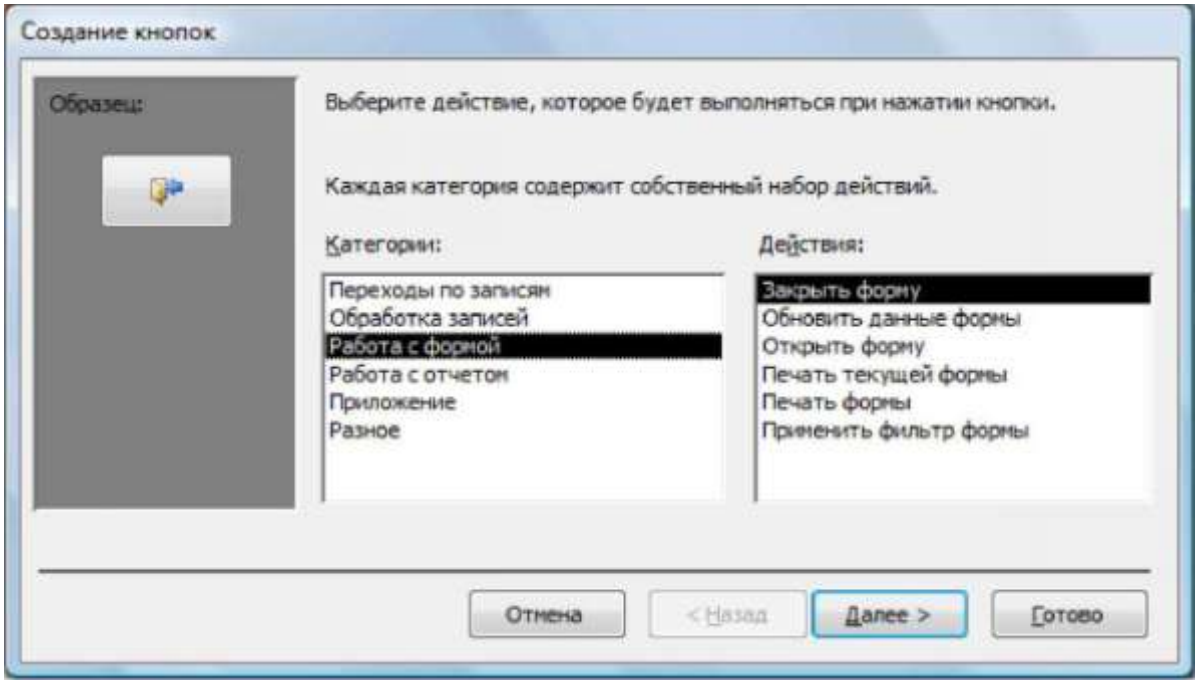


Рисунок 2.9 - Создание кнопок на форме

10. В категории *Работа с формой* выберите действие *Закрытие формы* и нажмите кнопку *Далее.*

11. Выберите рисунок или текст, который будет размещаться на кнопке.
12. В последнем диалоговом окне *Мастера кнопок* задайте имя кнопки и нажмите *Готово.*
13. *Мастер кнопок* написал для данной кнопки процедуру на языке Microsoft Visual Basic. Просмотреть процедуру обработки события можно с помощью команды *Обработка событий* контекстного меню кнопки.
14. Самостоятельно создайте кнопки *Выход из приложения, Поиск записи, Удаление записи.*
15. Иногда на форме требуется разместить несколько страниц, содержащих данные из различных источников, справочную или вспомогательную информацию. Для этой цели можно использовать набор вкладок.
16. Создайте пустую форму.
17. Для добавления к форме набора вкладок щелкните по кнопке *Вкладка* на панели инструментов *Элементы управления.* Сначала добавятся только две вкладки с формальными именами *Вкладка 1* и *Вкладка 2.*
18. Добавьте еще одну вкладку.
19. Переименуйте ярлычки вкладок так, чтобы на них отображались названия данных, которые будут в них располагаться: *Сотрудники, Менеджеры, Помощь.*
20. Перейдите на вкладку *Сотрудники* и перетащите на нее мышкой из базы данных форму *Сотрудники.*
21. Аналогичным образом поместите форму *Менеджеры* на вкладку *Менеджеры.*

22. На вкладку *Помощь* поместите некоторые советы по работе с базой данных.
23. Данную форму сохраните с именем *Сотрудники фирмы.*
24. В *Microsoft Access* можно создавать кнопочные формы. Они содержат только кнопки и предназначены для выбора основных действий в базе данных. Для создания кнопочной формы необходимо на вкладке ленты *Работа с базами данных* выбрать команду *Диспетчер кнопочных форм.*
25. Если кнопочной формы в базе данных нет, то будет выведен запрос на подтверждение ее создания. Нажмите *Да* в диалоговом окне подтверждения.
26. Перед вами *появятся Диспетчер кнопочных форм*, в котором щелкните по кнопке *Создать.*
27. В диалоговом окне *Создание* (рис. 2.10) введите имя новой кнопочной формы и нажмите *ОК.*

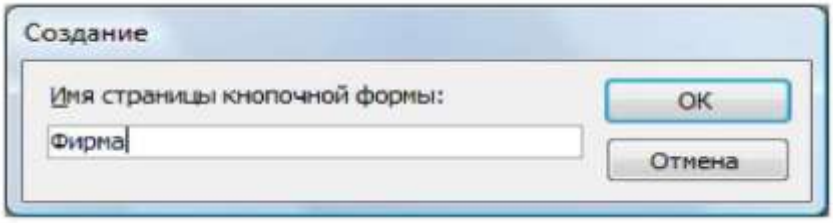


Рисунок 2.10 - Задание имени кнопочной формы

28. Имя новой кнопочной формы добавится в список *Страницы кнопочной формы окна Диспетчер кнопочных форм* (рис. 2.11). Выделите имя новой кнопочной формы и щелкните по кнопке *Изменить.*

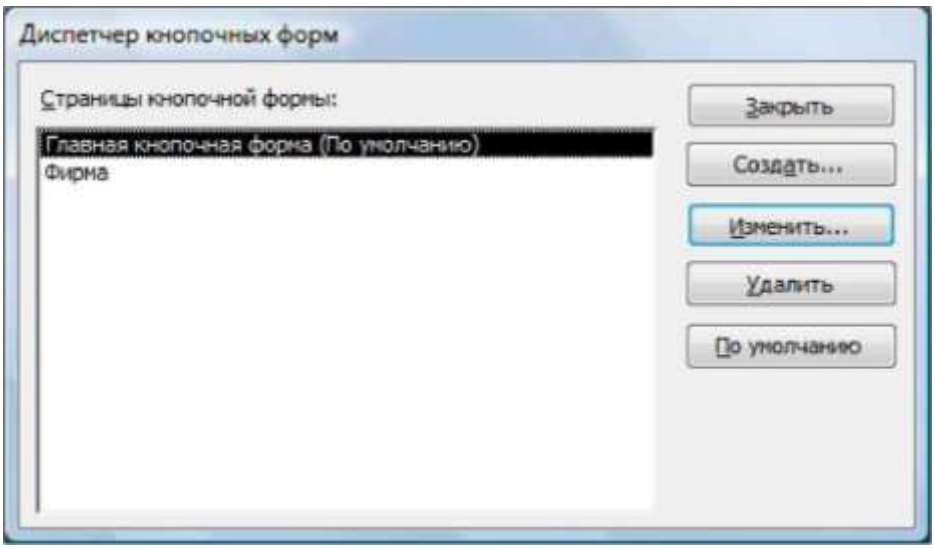


Рисунок 2.11 - Диспетчер кнопочных форм

29. В диалоговом окне *Изменение страницы кнопочной формы* щелкните по кнопке *Создать*. Появится диалоговое окно *Изменение элемента кнопочной формы* (рис. *2*.12).

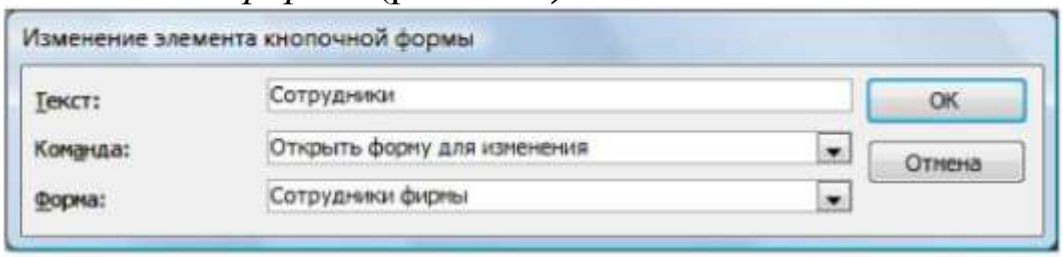


Рисунок 2.12 - Создание кнопок на форме

30. В поле *Текст* введите текст подписи для первой кнопки кнопочной формы, а затем выберите команду из раскрывающегося списка в поле *Команда.* В поле *Форма* выберите форму, для которой будет выполняться данная команда.
31. Аналогичным образом добавьте кнопки *Клиенты, Заказы, Выход.*
32. В диалоговом *окно Диспетчер кнопочных форм* выберите имя вашей кнопочной формы и щелкните по кнопке *По умолчанию.*

Рядом с названием кнопочной формы появится надпись *«(по умолчанию)».*

33. Чтобы закончить создание кнопочной формы, щелкните по кнопке *Закрыть*.
34. В результате должна получиться форма, представленная на рис. 2.13.

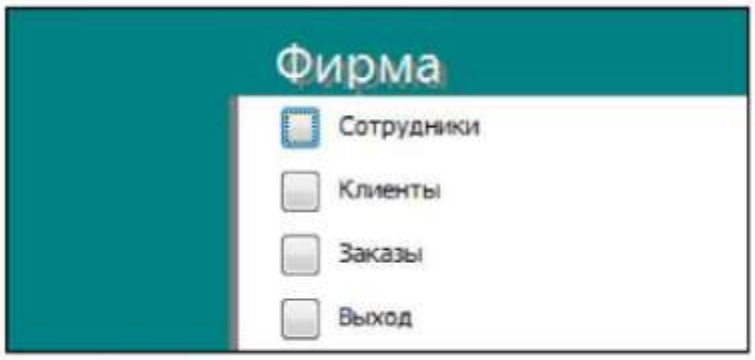


Рисунок 2.13 - Главная кнопочная форма

35. Добавьте в форму какой-нибудь рисунок.
36. Для того чтобы главная кнопочная форма появлялась на экране при запуске приложения, необходимо в главном меню нажать на кнопку *Параметры Access* (рис. 2.14). Для текущей базы данных установите форму просмотра- «кнопочная форма».

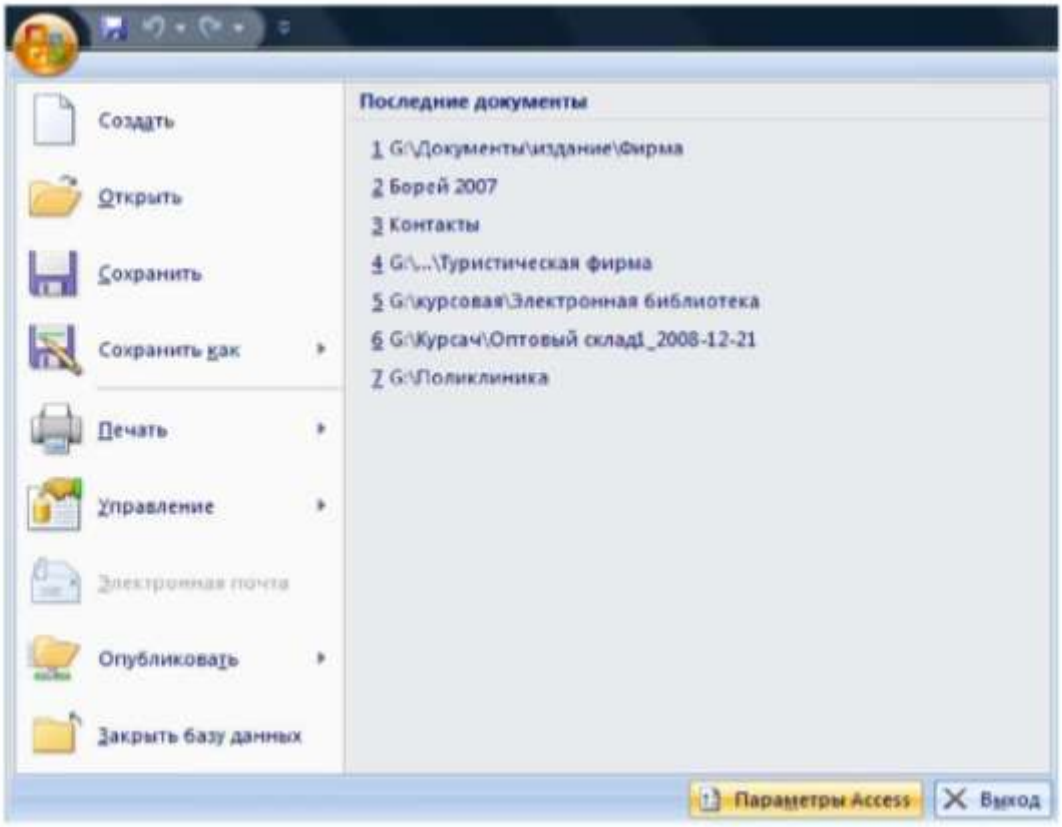


Рисунок 2.14 - Задание параметров Access

Создание отчетов

Отчеты предназначены дня вывода информации на печать. Часто данные в них располагаются в табличной форме. В отличие от распечаток таблиц или запросов отчет дает более широкие возможности сортировки и группировки данных, он предоставляет возможность добавлять итоговые значения, а также поясняющие надписи, колонтитулы, номера страниц, стили и различные графические элементы.

Создавать отчеты в базе данных Access можно несколькими способами:

1) с помощью Мастера отчетов;
2) на основе таблиц или запросов;
3) в режиме Конструктора.

Пример создания отчета:

1. В окне базы данных выполните команду: вкладка ленты Создание - панель инструментов Отчеты - Мастер отчетов.

2. Выберите из списка таблицу (или запрос), которая будет использована как источник данных (например, запрос Адреса клиентов).

3. В появившемся диалоговом окне Создание отчетов (рис. 2.15) переместите все доступные поля в область «выбранные поля»

.

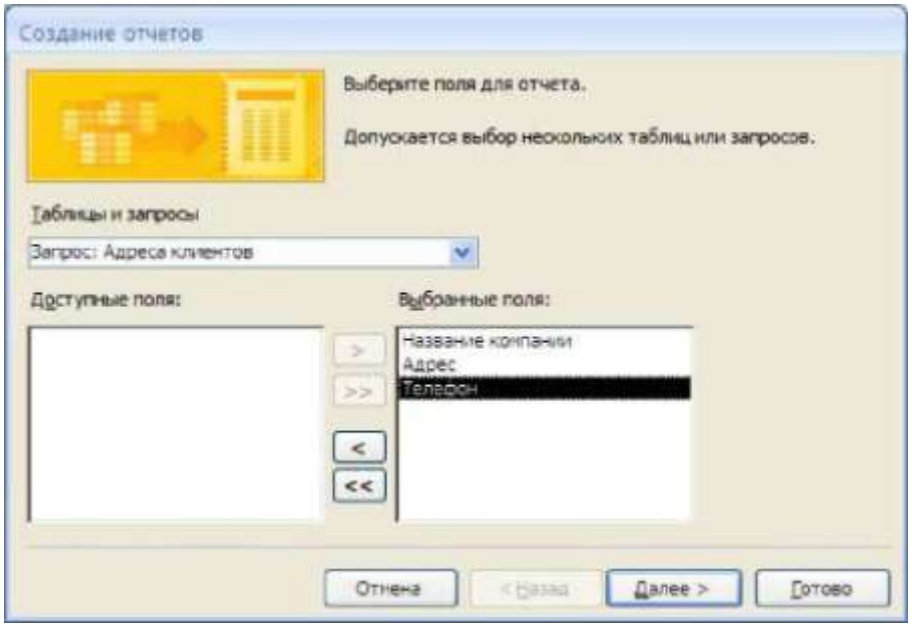


Рисунок 2.15 - Мастер отчетов

1. С помощью *Мастера отчетов* создайте отчет *Дни рождения.* В качестве источника данных используйте таблицу *Сотрудники.*
2. Если требуется напечатать почтовые наклейки, Access предоставляет такую возможность. Для этого выделите таблицу *Клиенты* и выполните команду: вкладка ленты *Создание* - панель инструментов *Отчеты - Наклейки.*
3. В появившемся диалоговом окне (рис. 2.16) укажите размер наклейки, систему единиц, тип наклейки и нажмите кнопку *Далее.*

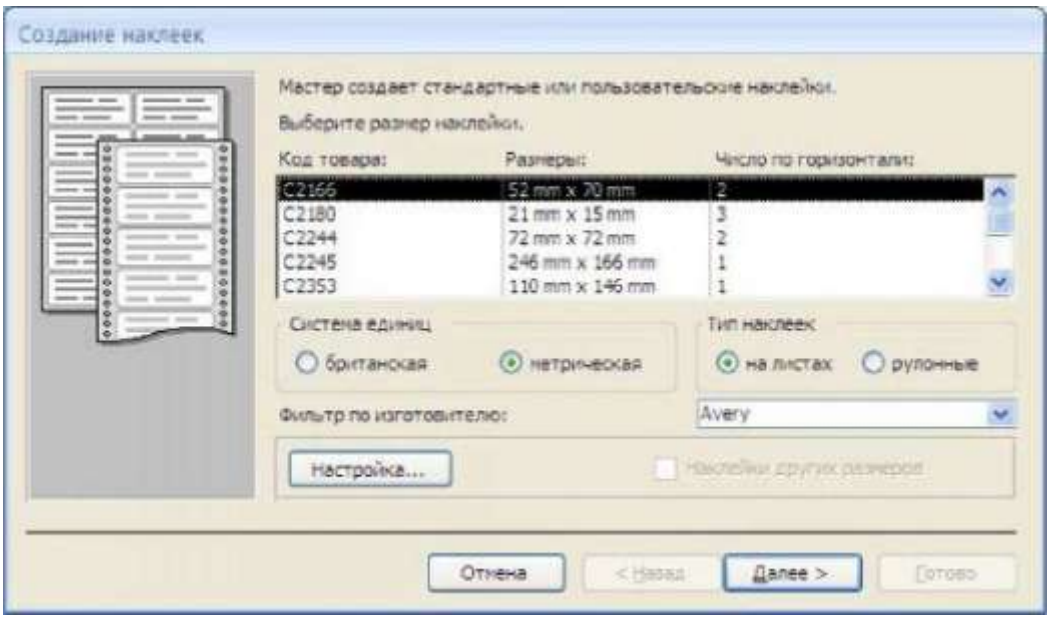


Рисунок 2.16 - Диалоговое окно Создание наклеек

4. На следующем шаге создания отчета установите шрифт, размер, цвет текста и начертание. Нажмите кнопку *Далее.*
5. Выберите поля, которые будут размещаться на наклейке. Например, *Название компании, Адрес, Телефон* и *Факс*. Если на каждой наклейке требуется вывести определенный текст, то введите его в прототип наклейки.
6. При необходимости измените название отчета с наклейками и нажмите кнопку *Готово.*
7. Иногда в отчетах требуется вычислять итоговые значения, среднее, минимальное или максимальное значения, а также проценты. Для этого

запустите *Мастер отчетов* и в качестве источника данных укажите запрос *Сумма заказа.*

11. В диалоговом окне *Мастера*, в котором задается порядок сортировки записей, нажмите кнопку *Итоги* (рис. 2.17).

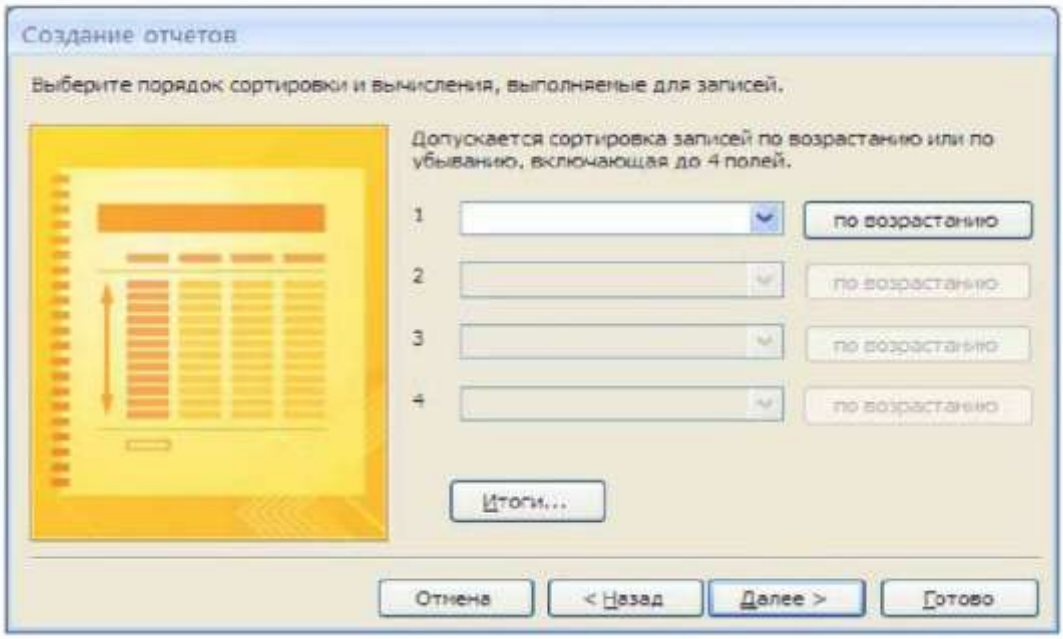


Рисунок 2.17 - Вычисление итоговых значений в отчетах

12. В диалоговом окне *Итоги* (рис. 2.18) для полей *Сумма* и *Налог* установите флажки в столбце *sum*, чтобы посчитать итоговую сумму.

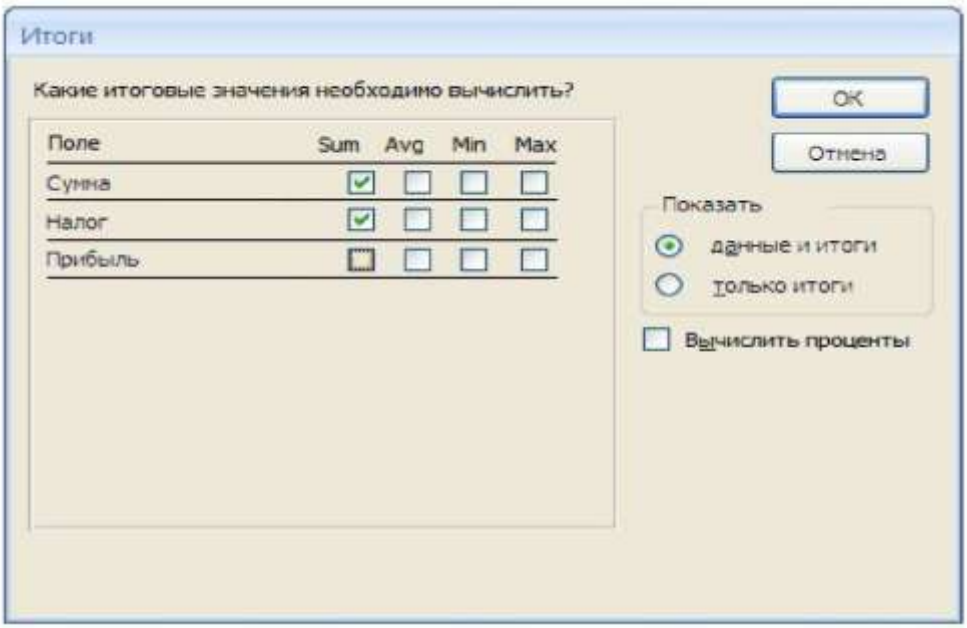


Рисунок 2.18 - Вычисление итоговых значений суммы

13. Далее выполните все шаги Мастера и нажмите кнопку *Готово.*
14. Создайте отчет *Дни рождения*, используя в качестве источника данных запрос *Дни рождения.*
15. Составьте отчет *Выполненные заказы*, в котором будут данные о компании и сумме заказа. Вычислите итоговую сумму, среднее значение (Avg) и максимальную сумму для каждой фирмы.

Задания для выполнения лабораторной работы - по вариантам.

Варианты для самостоятельного выполнения лабораторной работы

Вариант 1

Разработайте базу данных *«Электронная библиотека»*, состоящую из трех таблиц со следующей структурой:

Книги - шифр книги (ключевое поле), автор, название, год издания, количество экземпляров.

Читатели - читательский билет (ключевое поле), фамилия, имя, отчество, адрес.

Выданные книги - шифр книги, читательский билет, дата выдачи, дата возвращения, дата фактического возвращения.

1. Установите связи между таблицами.
2. С помощью запроса отберите все книги, выпущенные с 1990 по 2007 годы.

3. Создайте запрос с параметром для отбора книг определенного автора.

4. Создайте формы для ввода данных, отчеты и главную кнопочную форму.

Вариант 2

Разработайте базу данных *«Продуктовый магазин»*, которая состоит из четырех таблиц со следующей структурой:

Товары - код товара (ключевое поле), наименование товара, количество товара.

Поступление товаров - код товара, дата поступления, цена приобретения товара за единицу, код поставщика.

Продажа товаров - код товара, месяц продажи, проданное количество за месяц, цена продажи товара.

Поставщики - код поставщика (ключевое поле), название поставщика, адрес поставщика, телефон поставщика.

2. Установите связи между таблицами.
3. С помощью запроса отберите товары, цены которых от 100 до 450 руб.
4. Создайте запрос с параметром для отбора товаров, проданных в определенном месяце.
5. Создайте формы для ввода данных, отчеты и главную кнопочную форму.

Вариант 3

Разработайте базу данных *«Сессия»,* состоящую из четырех таблиц со следующей структурой:

Студенты - шифр студента (ключевое поле), фамилия, имя, отчество, курс, группа.

Экзамены - шифр студента, дата, шифр дисциплины, оценка.

Зачеты - шифр студента, дата, шифр дисциплины, зачет.

Дисциплины шифр дисциплины (ключевое поле), название дисциплины, количество часов.

1. Установите связи между таблицами.
2. С помощью запроса отберите студентов, сдавших экзамен на 4 или 5.
3. Создайте запрос с параметром для отбора студентов, получивших или не получивших зачет.
4. Создайте формы для ввода данных, отчеты и главную кнопочную форму.

Вариант 4

Разработайте базу данных *«Оптовый склад»,* состоящую из четырех таблиц со следующей структурой:

Склад - код товара, количество, дата поступления.

Товары - код товара (ключевое поле), название товара, срок хранения.

Заявки - код заявки (ключевое поле), название организации, код товара, требуемое количество.

Отпуск товаров - код заявки (ключевое поле), код товара, отпущенное количество, дата отпуска товара.

1. Установите связи между таблицами.
2. С помощью запроса отберите товары, количество которых от 50 до 200 штук.
3. Создайте запрос с параметром для отбора товаров, поступивших на склад какого-либо числа.
4. Создайте формы для ввода данных, отчеты и главную кнопочную форму.

Вариант 5

Разработайте базу данных «*Абитуриенты*», состоящую из четырех таблиц со следующей структурой:

Анкета - номер абитуриента (ключевое поле), фамилия, имя, отчество, дата рождения, оконченное среднее учебное заведение (название, номер, населенный пункт), дата окончания учебного заведения, наличие красного диплома или золотой / серебряной медали, адрес, телефон, шифр специальности.

Специальности - шифр специальности (ключевое поле), название специальности.

Дисциплины - шифр дисциплины (ключевое поле), название дисциплины.

Вступительные экзамены - номер абитуриента, шифр дисциплины, экзаменационная оценка.

1. Установите связи между таблицами.
2. Составьте запрос для отбора студентов, сдавших экзамены без троек.
3. Создайте запрос с параметром для отбора студентов, поступающих на определенную специальность.

4. Создайте формы для ввода данных, отчеты и главную кнопочную форму.

Вариант 6

Разработайте базу данных *«Транспортные перевозки»*, состоящую из трех таблиц со следующей структурой:

Транспорт - марка автомобиля, государственный номер (ключевое поле), расход топлива.

Заявки - код заявки (ключевое поле), дата заявки, название груза, количество груза, пункт отправления, пункт назначения.

Доставка - № п/п, дата и время отправления, дата и время прибытия, код заявки, государственный номер автомобиля, пройденное расстояние.

1. Установите связи между таблицами.
2. С помощью запроса отберите заявки с количеством груза от 100 до 500 кг.
3. Создайте запрос с параметром для отбора транспорта по марке автомобиля.
4. Создайте формы для ввода данных, отчеты и главную кнопочную форму.

Вариант № 7

Разработайте базу данных *«Прокат спортивного оборудования»,* состоящую из трех таблиц со следующей структурой:

Кпиенты - код клиента (ключевое поле), фамилия, имя, отчество, телефон, адрес, паспортные данные, залог.

Склад - код оборудования (ключевое поле), название, количество, залоговая стоимость, остаток.

Прокат - № п/п, клиент, оборудование, дата выдачи, срок возврата, отметка о возврате, оплата проката.

1. Установите связи между таблицами.
2. Создайте запрос для отбора оборудования с залоговой стоимостью от 10000 до 50000 руб.
3. Создайте запрос с параметром для отбора клиентов, возвративших оборудование.

4. Создайте формы для ввода данных, отчеты и главную кнопочную форму.

Вариант 8

Разработайте базу данных *«Банк»*, состоящую из трех таблиц со следующей структурой:

Клиенты - код клиента (ключевое поле), фамилия, имя, отчество, паспорт, телефон, адрес, заработная плата.

Виды кредитов - код кредита (ключевое поле), название кредита, процентная ставка, условия предоставления.

Предоставленные кредиты - № п/п, клиент, кредит, дата предоставления, срок, дата возврата, сумма, отметка о возврате.

1. Установите связи между таблицами.
2. Создайте запрос для отбора клиентов, взявших кредит от 500 ООО до 1 ООО ООО руб.
3. Создайте запрос с параметром для отбора кредитов по процентной ставке.
4. Создайте формы для ввода данных, отчеты и главную кнопочную форму.

Вариант 9

1. Разработайте базу данных *«Туристическая фирма»,* состоящую из четырех таблиц со следующей структурой:

Клиенты - код клиента (ключевое поле), фамилия, имя, отчество, телефон, адрес, паспорт.

Сотрудники - код сотрудника (ключевое поле), фамилия, имя, отчество, должность, телефон, адрес, паспортные данные.

Туристические маршруты - код маршрута (ключевое поле), название, описание маршрута, страна, стоимость путевки, количество дней, вид транспорта.

«Заказы» код заказа (ключевое поле), клиент, маршрут, сотрудник (менеджер, оформивший заказ), дата, отметка об оплате.

1. Установите связи между таблицами.

2. Создайте запрос для отбора маршрутов со стоимостью от 10000 до 20000 руб.
3. Создайте запрос с параметром для отбора клиентов, выбравших определенный вид маршрута.
4. Создайте формы для ввода данных, отчеты и главную кнопочную форму.

Вариант 10

Разработайте базу данных *«Поликлиника»*, состоящую из четырех таблиц со следующей структурой:

Врачи код врача (ключевое поле), ФИО, должность, специализация, стаж работы, адрес, телефон.

Болезни - № п/п (ключевое поле), название заболевания, рекомендации по лечению, меры профилактики.

Пациенты - код пациента (ключевое поле), ФИО, адрес, телефон, страховой полис, паспорт.

Диагноз - № п/п (ключевое поле), пациент, заболевание, лечащий врач, дата обращения, дата выздоровления.

1. Установите связи между таблицами.
2. С помощью запроса отберите врачей-стоматологов и ортопедов.
3. Создайте запрос с параметром для отбора пациентов с определенным видом заболевания.
4. Создайте формы для ввода данных, отчеты и главную кнопочную форму.

Содержание отчета

Перед выполнением лабораторной работы студент обязан ознакомиться с теоретическим материалом по данной теме.

Отчет по лабораторной работе должен содержать:

1) тема и цель работы;
2) условие задания;
3) постановка задачи;
4) ход выполнения работы;
7) результаты выполнения работы;
8) выводы.

Лабораторная работа считается выполненной, если все задания выполнены, получены соответствующие результаты, составлен полный отчет по указанной форме. После выполнения студент допускается к защите лабораторной работы.

На защиту выносятся:

- теоретические сведения по теме данной лабораторной работы;
- результаты выполнения работы;
- контрольные вопросы.

Лабораторная работа считается выполненной, если она выполнена и защищена.

Контрольные вопросы

1. Что относится к объектам базы данных Access?
2. Какие типы данных существуют? Какие из этих типов данных были использованы в ходе выполнения работы?
3. Перечислите виды отношений между таблицами (виды связей). Какие виды связей использовались в ходе выполнения работы?
4. Какими способами можно создавать отчеты в MS Access?
5. Как создать кнопочную форму в MS Access?

Лабораторная работа 3
Основы разработки мобильных приложений

Цель работы: знакомство с инструментами разработки Android-приложений.

Методические указания

Android- операционная система для мобильных устройств: смартфонов, планшетных компьютеров, КПК. В настоящее время именно Android является самой широко используемой операционной системой для мобильных устройств. Это бесплатная операционная система, основанная на Linux с интерфейсом программирования Java.

Платформа Android объединяет операционную систему, построенную на основе ядра ОС Linux, промежуточное программное обеспечение и встроенные мобильные приложения. Разработка и развитие мобильной платформы Android выполняется в рамках проекта AOSP (Android Open Source Project) под управлением OHA (Open Handset Alliance), руководит всем процессом Google.

Android поддерживает фоновое выполнение задач: предоставляет развитую библиотеку элементов пользовательского интерфейса: поддерживает 2D- и 3D-графику. используя OpenGL стандарт: поддерживает доступ к файловой системе и встроенной базе данных SQLite.

Приложение для Android пишется на языке Java, а среду разработки можно выбрать, например, из популярных средств разработки, таких как Android Studio или Eclipse,

Android Studio - среда разработки под Android. Она предоставляет интегрированные инструменты для разработки и отладки. Для отладки приложений используется эмулятор телефона - виртуальная машина, на которой будет запускаться созданное приложение.

Чтобы создать эмулятор телефона, выбираем в меню Android Studio **ПУНКТЫ** SDK Manager | Tools| AVD Manager | Device Definitions.

Далее нужно выбрать нужную версию виртуального устройства и при необходимости отредактировать его параметры.

Далее рассмотрим этапы создания нового проекта. После запуска Android Studio выбираем New Project, появится диалоговое окно мастера.

Поле Application name - имя, которое будет отображаться в заголовке приложения.

Поле Company Domain служит для указания адреса вашего сайта,

Поле Package name формирует специальный Java-пакет на основе вашего имени из предыдущего поля.

Поле Project location позволяет выбрать место на диске для создаваемого проекта.

После нажатия на кнопку Next переходим к следующему окну. Здесь выбираем типы устройств, под которые будем разрабатывать приложение. В большинстве случае мы будет писать для смартфонов и планшетов, поэтому оставляем флажок у первого пункта. Также можно создавать приложения для Android TV. Android Wear и Glass.

Далее необходимо выбрать внешний вид экрана приложения. Выберем, например, вариант Blank Activity. После нажатия кнопки Finish студия формирует проект и создаёт необходимую структуру из различных файлов и папок.

В левой части среды разработки на вкладке Android появится иерархический список из папок, которые относятся к проекту. Вкладка Android содержит две основные папки: app и Gradle Scripts. Первая папка app является отдельным модулем для приложения и содержит все необходимые файлы приложения - код. ресурсы картинок и т.п, Вторая папка служит для различных настроек для управления проектом, Раскрываем папку app. В ней находятся три папки: manifest, java, res.

Папка manifest содержит единственный файл манифеста AndroidManifest.xml. В этом файле должны быть объявлены все активности, службы, приёмники и контент-провайдеры приложения, Также он должен содержать требуемые приложению разрешения. «AndroidManifest.xml» можно рассматривать, как описание для развертывания Andioid-приложения.

Папка java содержит три подпапки - рабочую и для тестов. Рабочая папка имеет название пакета и содержит файлы классов. Сейчас там один класс MainActivity.

Папка res содержит файлы ресурсов, разбитых на отдельные подпапки,

Запуск программы осуществляется выбором команды Run в пункте меню Run. После этого в эмуляторе загрузится разработанная

программа, на экране появится окно приложения с надписью «Hello World!» и заголовком программы.

Порядок выполнения работы

1. Изучить методические указания к практическому занятию №2.
2. Запустить на выполнение Android Studio.
3. Если будет использоваться внешний эмулятор для запуска Android-приложений (например, Blue Stacks или Genvmotion), загрузить его.
4. Создать в среде Android Studio проект, рассмотренный на практическом занятии №2.
5. Изучить содержимое файка AndroidManifest.xml из папки manifest и папку res, содержащую файлы ресурсов проекта,
6. Запустить созданное приложение в эмуляторе Android и наблюдать за появлением этого приложения и результатов его работы в окне приложений эмулятора.
7. Изучить содержимое файлов проекта activity_main.xml и MainActivity.java. Отредактировать их содержимое для получения нового текста на экране и запустить приложение в эмуляторе Android.

Отчет должен содержать:

1. Название и цель работы,
2. Листинг созданного приложения в Android Studio на языке Java.
3. Результаты работы приложения в эмуляторе телефона.
4. Выводы по проделанной работе.

Контрольные вопросы

1. Какой язык программирования используется при создании приложений для ОС Android?
2. В чем заключается преимущество ОС Android среди конкурентов с точки зрения разработчика мобильных приложений?
3. Что такое эмулятор телефона?
4. Как создать эмулятор телефона?
5. Как осуществляется запуск программы в AndroidStudio?

Лабораторная работа 4

Программирование на языке гипертекстовой разметки документов HTML

Цель работы: Изучение языка гипертекстовой разметки HTML. Получение основных навыков создания HTML-документов для internet-сетей.

2. Подготовка к работе

В процессе подготовки к лабораторной работе необходимо изучить:

- методы создания HTML документов;
- рабочее задание и методические указания к его выполнению.

3. World Wide Web, HTML и HTTP

World Wide Web *(Web)* - это глобальная сеть информационных ресурсов. Для того, чтобы сделать эти ресурсы доступными наиболее широкой аудитории, в Web используются три механизма:

1. Единая схема наименования для поиска ресурсов в Web (например, URI).
2. Протоколы для доступа к именованным ресурсам через Web (например, HTTP).
3. Гипертекст для простого перемещения по ресурсам (например, HTML).

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.

URI обычно состоят из трех частей:

1. Схема наименования механизма, используемого для доступа к ресурсу.
2. Имя машины, на которой располагается ресурс.
3. Имя собственно ресурса, зачданное в виде пути.

Рассмотрим URI спецификации четвертой версии HTML на сервере W3C:

http://www.w3.org/TR/PR-html4/cover.html

Этот URI может читаться следующим образом: этот документ можно получить по протоколу HTTP (см. [RFC2068]), он располагается на машине www.w3.org, путь к этому документу - "/TR/PR-html4/cover.html".

Кроме того, в документах в формате HTML можно увидеть схемы "mailto" для электронной почты и "ftp" для протокола FTP. Приведем пример URI, относящийся к почтовому ящику пользователя:

...текст...

Комментарии отправляйте <A href="mailto:joe@someplace.com"> Джо Кулу</A>.

Некоторые URI указывают на местоположение внутри ресурса. Этот тип URI заканчивается символом "#", за которым следует указатель (*идентификатор фрагмента*). Например, следующий URI указывает на фрагмент с именем section_2:

http://somesite.com/html/top.html#section_2

Относительный URI не содержит информации о схеме наименования. Путь в нем указывает на ресурс на машине, на которой находится текущий документ. Относительные URI могут содержать компоненты относительного пути (например, ".." означает один уровень выше в иерархии) и идентификаторы фрагментов.

Относительные URI приводятся к полным URI с помощью базового URI. В качестве примера приведения относительного URI предположим, что у нас имеется базовый URI "http://www.acme.com/support/intro.html". Относительный URI в следующей ссылке:

<A href="suppliers.html">Suppliers</A>

будет преобразован в полный URI "http://www.acme.com/support/suppliers.html", а относительный URI в следующем фрагменте

<IMG src="../icons/logo.gif" alt="logo">

будет преобразован в полный URI "http://www.acme.com/icons/logo.gif".

В HTML URI используются для:

- ссылки на другие документы или ресурсы (элементы A и LINK).
- ссылки на внешние таблицы стилей или скрипты (элементы LINK и SCRIPT).

- включения в страницу изображений, объектов или апплетов (элементы IMG, OBJECT, APPLET и INPUT).
- создания изображений-карт (элементы MAP и AREA).
- отправки форм (FORM).
- создания документов с использованием кадров (элементы FRAME и IFRAME).
- ссылок на внешние источники (элементы Q, BLOCKQUOTE, INS и DEL).
- ссылок на соглашения о метаданных, описывающих документ (элемент HEAD).

Чтобы представить информацию для глобального использования, нужен универсальный язык, который понимали бы все компьютеры. Языком публикации, используемым в World Wide Web, является ***HTML*** (HyperText Markup Language - язык разметки гипертекстов).

HTML дает авторам средства для:

- публикации электронных документов с заголовками, текстом, таблицами, списками, фотографиями и т.д.
- загрузки электронной информации с помощью щелчка мыши на гипертекстовой ссылке.
- разработки форм для выполнения транзакций с удаленными службами, для использования в поиске информации, резервировании, заказе продуктов и т.д.
- включения электронных таблиц, видеоклипов, звуковых фрагментов и других приложений непосредственно в документы.

Язык HTML был разработан Тимом Бернерс-Ли во время его работы в CERN и использовался в браузере Mosaic, разработанным в NCSA. В 1990-х годах он получил широкое распространение благодаря быстрому росту Web. В это время HTML был расширен и дополнен. Важнейшие черты языка были отражены в ряде спецификаций.

HyperText Transfer Protocol (HTTP) - это протокол высокого уровня (а именно, уровня приложений), обеспечивающий передачу данных, требующуюся для распределенных информационных систем гипермедиа. HTTP используется в World Wide Web с 1990 года.

Практические информационные системы требуют большего, чем примитивный поиск, модификация и аннотация данных. HTTP/1.0 предоставляет открытое множество методов, которые могут быть

использованы для указания целей запроса. Они построены на дисциплине ссылок, где для указания ресурса, к которому должен быть применен данный метод, используется URI, в виде местонахождения (URL) или имени (URN). Формат сообщений сходен с форматом Internet Mail или Multipurpose Internet Mail Extensions (MIME-Многоцелевое Расширение Почты Internet).

HTTP/1.0 используется также для коммуникаций между различными пользовательскими браузерами и шлюзами, дающими гипермедиа доступ к существующим Internet протоколам, таким как SMTP, NNTP, FTP, Gopher и WAIS. HTTP/1.0 разработан, чтобы позволять таким шлюзам через proxy серверы, без какой-либо потери передавать данные с помощью упомянутых протоколов более ранних версий.

HTTP основывается на парадигме запросов/ответов. Запрашивающая программа (клиент) устанавливает связь с обслуживающей программой-получателем (сервер) и посылает запрос серверу в следующей форме: метод запроса, URI, версия протокола, за которой следует MIME-подобное сообщение, содержащее управляющую информацию запроса, информацию о клиенте и, может быть, тело сообщения. Сервер отвечает сообщением, содержащим строку статуса (включая версию протокола и код статуса - успех или ошибка), за которой следует MIME-подобное сообщение, включающее в себя информацию о сервере, метаинформацию о содержании ответа, и, вероятно, само тело ответа

В Internet коммуникации обычно основываются на TCP/IP протоколах. Для WWW номер порта по умолчанию - TCP 80, но также могут быть использованы и другие номера портов - это не исключает возможности использовать HTTP в качестве протокола верхнего уровня.

Для большинства приложений сеанс связи открывается клиентом для каждого запроса и закрывается сервером после окончания ответа на запрос. Тем не менее, это не является особенностью протокола. И клиент, и сервер должны иметь возможность закрывать сеанс связи, например, в результате какого-нибудь действия пользователя. В любом случае, разрыв связи, инициированный любой стороной, прерывает текущий запрос, независимо от его статуса.

4. Структура HTML документа

Данные в формате HTML похожи на текстовой файл, за исключением того, что некоторые из символов интерпретируются как разметка. Разметка придает документу некую структуру.

Данные представляют собой иерархию элементов. Каждый элемент имеет имя, атрибуты и несет некую информацию. Большинство элементов представлены в документе в виде начальной метки, указывающей имя и атрибуты. Далее следует собственно содержание элемента. И наконец, заканчивает все это конечная метка.

HTML документ состоит из следующих базисных элементов:

тип документа <HTML></HTML> (начало и конец файла);

заголовок <HEAD></HEAD> (описание документа, например его имя);

имя документа <TITLE></TITLE> (должно быть в заголовке);

тело <BODY></BODY> (содержит публикуемые данные).

Например:

```
<HTML>
    <HEAD>
        <TITLE>Название документа</TITLE>
    </HEAD>
    <BODY>
Информация, содержащаяся в документе.
    </BODY>
</HTML>.
```

Каждый элемент HTML документа начинается с метки, меткой же и заканчивается каждый непустой элемент. Начальные метки выделяются символами < и > , а конечные - символами </ и > .

Имя элемента следует в метке сразу за символом открытия <. Имя начинается с буквы, за которой могут следовать еще 33 буквы, цифры, пробелы или дефисы. В именах игнорируется разница между прописными и строчными буквами.

Начальная метка позволяет вставить между именем и символом > пробелы и атрибуты. Атрибут состоит из имени, символа равенства и значения. Слева и справа от символа равенства можно оставлять пробелы.

Значение атрибута указывается в виде строки, заключенной в одинарные или двойные кавычки.

5. Элементы языка HTML

Документы должны (но не обязательно) содержать элемент HEAD, за которым следует элемент BODY.

5.1 Тэг <HEAD>

Элемент HEAD содержит всю информацию о документе в целом. Однако он не содержит какого-либо текста. Последний должен находиться в элементе BODY. В элементе заголовка HEAD можно использовать лишь строго заданный набор элементов.

Нижеприведенные элементы определяют общие свойства документа. Они должны появляться в элементе HEAD. Порядок следования элементов значения не имеет.

<TITLE> - название элемента.

<ISINDEX> - элемент, посылаемый серверу вместе с документом, предназначенным для информации к поиску.

<LINK> - элемент, определяющий связь этого документа с другими. В , документе может присутствовать несколько элементов LINK.

<BASE> - запись, сделанная на языке URL при фиксировании данного , документа.

<META> - метаинформация.

Например:

```
<HEAD>
    <LINK REV="made" TITLE="Vasia Ivanov" HREF="mailto:vasia@ivanov.ru">
    <META NAME="KeyWords" CONTENT="ключевые слова для поисковых машин">
    <META NAME="Description" CONTENT="краткое описание документа">
    <TITLE>Домашняя страничка Васи Иванова</TITLE>
</HEAD>
```

5.2 Тэг <BODY>

В противоположность элементу HEAD элемент BODY содержит всю ту информацию, из которой собственно и состоит рассматриваемый документ.

Далее рассматриваются элементы, применяемые для представления и форматирования информации в документе.

5.3 Определение структуры текста

Для определения структуры текста используются следующие тэги.

<H?></H?> - заглавие (стандарт определяет 6 уровней).

<H? ALIGN=LEFT|CENTER|RIGHT></H?> - заглавие с выравниванием.

<DIV></DIV> - секция.

<DIV ALIGN=LEFT|RIGHT|CENTER></DIV> - секция с выравниванием.

<BLOCKQUOTE></BLOCKQUOTE> - цитата (обычно выделяется отступом).

<EM></EM> - выделение (обычно изображается курсивом).

<STRONG></STRONG> - дополнительное выделение (обычно изображается жирным шрифтом).

<CITE></CITE> - отсылка, цитата (обычно курсив).

<CODE></CODE> - код (для листингов кода).

<SAMP></SAMP> - пример вывода.

<KBD></KBD> - ввод с клавиатуры.

<VAR></VAR> - переменная.

<BIG></BIG> - большой шрифт.

<SMALL></SMALL> - маленький шрифт.

5.4 Внешний вид

Внешний вид текста определяется с использованием следующих тэгов.

<B></B> - жирный.

<I></I> - курсив.

 - верхний индекс.

 - нижний индекс.

<TT></TT> - печатная машинка (изображается как шрифт фиксированой ширины).

<PRE></PRE> - форматированый (сохранить формат текста как есть).

<PRE WIDTH=?></PRE> - ширина (в символах).

<CENTER></CENTER> - центрировать (как текст, так и графика).

<FONT SIZE=?></FONT> - размер шрифта (от 1 до 7).

<FONT SIZE="+|-?"></FONT> - изменить размер шрифта.

<FONT COLOR="#$$$$$$"></FONT> - цвет шрифта.

<FONT FACE="***"></FONT> - выбор шрифта.

5.5 Ссылки и графика

Следующие тэги используются для организации ссылок и работы с графикой.

<A HREF="URL"></A> - ссылка.

<A HREF="URL#***"></A> - ссылка на закладку (в другом документе).

<A HREF="#***"></A> - ссылка на закладку (в том же документе).

<A HREF="URL" TARGET="***|_blank|_self|_parent|_top"></A> - ссылка на другое окно. Например: <A HREF="http://www.torrentino.com/" TARGET="_blank"> Торрентино (Большое количество разных файлов)</A>

<A NAME="***"></A> - определить закладку.

<IMG SRC="URL"> - графика.

<IMG SRC="URL" ALIGN=TOP|BOTTOM|MIDDLE|LEFT|RIGHT> - графика с выравниванием.

<IMG SRC="URL" ALT="***"> - альтернатива (выводится если картинка не изображается).

<IMG SRC="URL" WIDTH=? HEIGHT=?> - размеры (в точках).

<IMG SRC="URL" BORDER=?> - окантовка (в точках).

<IMG SRC="URL" HSPACE=? VSPACE=?> - отступ (в точках).

5.6 Разделители

Следующие тэги используются в качестве разделителей.

<P></P> параграф (закрывать элемент часто не обязательно).

<P ALIGN=LEFT|CENTER|RIGHT></P> - выравнивание.

 - новая строка (одиночный перевод строки).

<BR CLEAR=LEFT|RIGHT|ALL> - убрать выравнивание.

<HR> - горизонтальный разделитель.

<HR ALIGN=LEFT|RIGHT|CENTER> - выравнивание.

<HR SIZE=?> - толщина (в точках).

<HR WIDTH=?> - ширина (в точках).

<HR WIDTH="%"> - ширина в процентах (в процентах от ширины страницы).

<HR NOSHADE> - сплошная линия (без трехмерных эффектов).

5.7 Списки

Для реализации списков используются следующие тэги.

<UL><LI></UL> - неупорядоченный (<LI> перед каждым элементом).

<UL COMPACT></UL> - компактный.

<UL TYPE=DISC|CIRCLE|SQUARE> - тип метки (для всего списка).

<LI TYPE=DISC|CIRCLE|SQUARE> - (этот и последующие).

<OL><LI></OL> - нумерованый (<LI> перед каждым элементом).

<OL COMPACT></OL> - компактный.

<OL TYPE=A|a|I|i|1> - тип нумерации (для всего списка).

<LI TYPE=A|a|I|i|1> - (этот и следующие).

<OL START=?> - первый номер (для всего списка).

<LI VALUE=?> - (этот и следующие).

<DL><DT><DD></DL> - список определений (<DT>=термин, <DD>=определение).

<DL COMPACT></DL> - компактный.

<MENU><LI></MENU> - меню (<LI> перед каждым элементом).

<MENU COMPACT></MENU> - компактное.

<DIR><LI></DIR> - каталог (<LI>перед каждым элементом).

<DIR COMPACT></DIR> - компактный.

5.8 Фон и цвета

Следующие тэги определяют фон и цвета.

<BODY BACKGROUND="URL"> - фоновая картинка.

<BODY BGCOLOR="#$$$$$$"> - цвет фона.

<BODY TEXT="#$$$$$$"> - цвет текста.

<BODY LINK="#$$$$$$"> - цвет ссылки.

<BODY VLINK="#$$$$$$"> - пройденная ссылка.

<BODY ALINK="#$$$$$$"> - активная ссылка.

5.9 Специальные символы

(обязаны быть в нижнем регистре)

&#?; - специальный символ (где ? это код ISO 8859-1).

< - <.

> - >.

& - &.

" - ".

® - торговая марка TM.

© - copyright.

  - неразделяющий пробел.

5.10 Формы

Для работы с формами используются следующие тэги.

<FORM ACTION="URL" METHOD=GET|POST></FORM> - определить форму.

<INPUT TYPE="TEXT|PASSWORD|CHECKBOX|RADIO|IMAGE|HIDDEN| SUBMIT|RESET"> - поле ввода.

<INPUT NAME="***"> - имя поля.

<INPUT VALUE="***"> - значение поля.

<INPUT CHECKED> - отмечен (checkboxes и radio boxes).

<INPUT SIZE=?> - размер поля (в символах).
<INPUT MAXLENGTH=?> - максимальная длина (в символах).
<SELECT></SELECT> - список вариантов.
<SELECT NAME="***"></SELECT> - имя списка.
<SELECT SIZE=?></SELECT> - число вариантов.
<SELECT MULTIPLE> - множественный выбор (можно выбрать больше одного).
<OPTION> - опция (элемент который может быть выбран).
<OPTION SELECTED> - опция по умолчанию.
<TEXTAREA ROWS=? COLS=?></TEXTAREA> - ввод текста, размер.
<TEXTAREA NAME="***"></TEXTAREA> - имя текста.

5.11 Таблицы

Для организации таблиц используются следующие тэги.
<TABLE></TABLE> - определить таблицу.
<TABLE BORDER=?></TABLE> - окантовка таблицы.
<TABLE CELLSPACING=?> - расстояние между ячейками.
<TABLE CELLPADDING=?> - дополнение ячеек.
<TABLE WIDTH=?> - желаемая ширина (в точках).
<TABLE WIDTH="%"> - ширина в процентах (проценты от ширины страницы).
<TR></TR> - строка таблицы.
<TR ALIGN=LEFT|RIGHT| CENTER|MIDDLE|BOTTOM> - выравнивание.
<TD></TD> - ячейка таблицы (должна быть внутри строки).
<TD ALIGN=LEFT|RIGHT| CENTER|MIDDLE|BOTTOM> - выравнивание.
<TD NOWRAP> - без перевода строки.
<TD COLSPAN=?> - растягивание по колонке.
<TD ROWSPAN=?> - растягивание по строке.
<TD WIDTH=?> - желаемая ширина (в точках).
<TD WIDTH="%"> - ширина в процентах (проценты от ширины страницы).

<TD BGCOLOR="#$$$$$$"> - цвет ячейки.
<TH></TH> - заголовок таблицы (как данные, но жирный шрифт и центровка).
<TH ALIGN=LEFT|RIGHT| CENTER|MIDDLE|BOTTOM> - выравнивание.
<TH NOWRAP> - без перевода строки.
<TH COLSPAN=?> - растягивание по колонке.
<TH ROWSPAN=?> - растягивание по строке.
<TH WIDTH=?> - желаемая ширина (в точках).
<TH WIDTH="%"> - ширина в процентах (проценты ширины таблицы).
<TH BGCOLOR="#$$$$$$"> - цвет ячейки.
<CAPTION></CAPTION> - заглавие таблицы.
<CAPTION ALIGN=TOP|BOTTOM> - выравнивание (сверху/снизу таблицы).

5.12 Фреймы

Для работы с фреймами используются следующие тэги.
<FRAMESET></FRAMESET> - документ с фреймами (вместо <BODY>).
<FRAMESET ROWS=,,,></FRAMESET> - высота строк (точки или %).
<FRAMESET ROWS=*></FRAMESET> - высота строк (* = относительный размер).
<FRAMESET COLS=,,,></FRAMESET> - ширина колонок (точки или %).
<FRAMESET COLS=*></FRAMESET> - ширина колонок (* = относительный размер).
<FRAMESET BORDER=?> - ширина окантовки.
<FRAMESET FRAMEBORDER="yes|no"> - окантовка.
<FRAMESET BORDERCOLOR="#$$$$$$"> - цвет окантовки.
<FRAME> - определить фрейм (содержание отдельного фрейма).
<FRAME SRC="URL"> - документ.
<FRAME NAME="***"|_blank|_self| _parent|_top> - имя фрейма.

<FRAME MARGINWIDTH=?>- ширина границы (правая и левая границы).

<FRAME MARGINHEIGHT=?> - высота границы (верхняя и нижняя границы).

<FRAME SCROLLING="YES|NO|AUTO"> - скроллинг.

<FRAME NORESIZE> - постоянный размер.

<FRAME FRAMEBORDER="yes|no"> - окантовка.

<FRAME BORDERCOLOR="#$$$$$$"> - цвет окантовки.

<NOFRAMES></NOFRAMES> - содержание без фреймов (для просмотрщиков не поддерживающих фреймы).

5.13 Комментарий

<!-- *** --> - тэг комментария (игнорируется браузером).

6. Список используемых терминов и сокращений

CGI (Common Gateway Interface) - интерфейс, позволяющий взаимодействовать программам клиента с программами, запущенными на сервере.

Cookie - порция информации, оставляемая на компьютере WEB-клиента программой, запущенной на стороне WEB-сервера. Применяется для сохранения данных, специфичных для данного клиента, например: имя пользователя, количество посещений сервера, регион пользователя и т.п.

CSS (Cascading Style Sheets) - -язык иерархических стилевых спецификаций. Главная цель CSS - отделить структуру документа от его оформления и позволить автору страницы самому решать, как должен выглядеть тот или иной элемент содержания. CSS не только освобождает от "обязательного" форматирования тех или иных тегов (например, полужирного начертания заголовков), но и добавляет множество новых степеней свободы, о которых раньше не приходилось и мечтать (например, возможность изменения интерлиньяжа - расстояния между строками текста).

FTP (File Transfer Protocol) - протокол передачи файлов, а также программа, его реализующая. Протокол был разработан для передачи файлов между компьютерами, использующими сеть на основе TCP/IP, в том числе и в Internet. Для доступа к некоторой информации посредством FTP на компьютере, с которого осуществляют доступ, должен быть установлен т.н. FTP-клиент, а на другом, соответственно, FTP-сервер. В WEB-практике FTP-доступ используется для доступа к страничкам WEB-сайта, обычно расположенным на сервере провайдера.

JAVA - межплатформенный язык программирования. Программы, написанные на JAVA, запускаются на стороне клиента, используя т.н. виртуальную машину (VM) Java. Применяется для создания динамических страничек, организиции доступа к базам данных посредством Internet и т.п.

JAVAScript - язык программирования, основанный на объектном представлении броузера. Текст программы встроен непосредственно в HTML-документ и интерпретируется самим броузером. Применяется в основном для создания таких эффектов,

как: бегущая строка, рисунки, изменяющие свой вид при подведении курсора и т.д.

Perl - язык программирования. Программы, написанные на Perl, запускаются на стороне сервера. В основном применяется на UNIX-ориентированных WEB-серверах. Применяется для обеспечения доступа к базам данным, создания динамических страничек и т.п.

Script - программа, написанная на каком-либо языке программирования для взаимодействия клиента с сервером. Например: Script на Perl для подсчета количества посещений.

Tag (Тэг) - элемент HTML, представляет из себя текст, заключенный в угловые скобки <>, является активным элементом, изменяющим представление следующей за ним информации. Может иметь некоторые атрибуты. Обычно имеются два тэга - открывающий и закрывающий. Например и - данные тэги описывают текст, находящийся между ними, как полужирный.

Доменное имя (Domain Name) - уникальный идентификатор, который назначается определенному IP-адресу. Доменное имя дает возможность обращаться к компьютеру по имени типа www.company.com, вместо запоминания его числового эквивалента.

Кодовая таблица - таблица соответствий символов и их положений в таблице. Исторически сложилось так, что в России есть несколько несовместимых кодировок, т.е. одинаковые символы имеют различные коды в разных кодировках. Это приводит к тому, что при просмотре страничек не в той кодировке, на которую настроен броузер, экран засоряется непонятными символами. В России распространены следующие кодировки: WIN1251 (Windows), KOI-8 (Unix), CP866(DOS), Macintosh, ISO-8859-5 (Unix).

Фреймы (Frames) - элементы HTML, появившиеся в броузерах версий 3.0. Позволяют разделить страницу на несколько независимых окон и в каждом из них размещать свою собственную WEB-страничку. Возможна ссылка из одного окна в другое. Применяется в основном для организации постоянно находящихся на экране меню, в то время как в другом окне располагается непосредственно сама информация.

7. Задание на лабораторную работу

Создать на языке гипертекстовой разметки HTML пример Вашей персональной домашней странички.

Страница должна содержать:

Заголовок (тэг <HEAD>), включающий в себя:

- название страницы (тэг <TITLE>), содержащее Ваше имя;
- обозначение кодировки страницы (рекомендуется использовать кодировку WINDOWS-1251, тэг <meta http-equiv=Content-Type content="text/html; charset=windows-1251">);
- Описание страницы (тэг <meta name="Description">);
- Описание ключевых слов (тэг <meta name="Keywords">).

Тело страницы (тэг <BODY>), включающее в себя:

- Меню с ссылками внутри документа;
- Текст с краткой информацией о себе;
- Изображение, "обтекаемое" вышеуказанным текстом;
- Ссылку на адрес Вашей электронной почты;
- Ссылки на различные сайты (сайты должны открываться в новом окне);
- Таблицу, содержащую данные о Ваших оценках за прошлый семестр.

Текст на странице должен быть оформлен разными стилями, размерами и шрифтами.

8. Порядок выполнения работы

1. Включить ПЭВМ и запустить любой текстовый редактор (желательно использовать текстовый редактор NOTEPAD.EXE, находящийся в стандартной поставке WINDOWS, так как он позволяет редактировать и сохранять текст в кодировке windows-1251).
2. Написать пример персональной страницы согласно п. 7.
3. Сохранить готовую программу в файл с форматом HTM или HTML.
4. Просмотреть файл-страницу в окне браузера (например Internet Explorer) и при необходимости устранить ошибки.
5. Продемонстрировать страницу преподавателю.

9. Содержание отчета

Отчет должен содержать:

1. Листинг файла-страницы на языке HTML.

2. Описание используемых тэгов.

3. Распечатку окна броузера с загруженной персональной страницей.

Контрольные вопросы

1. Поясните понятие URI.
2. Что из себя представляет язык HTML?
3. Что из себя представляет протокол HTTP?
4. Опишите структуру HTML документов.
5. Назовите элементы языка HTML, применяемые внутри тэга <HEAD>.
6. Назовите элементы языка HTML, применяемые для разметки текста.
7. Назовите элементы языка HTML, применяемые для построения таблиц.
8. Каким образом можно определить цвета текста и фона?

Список литературы

1. Советов, Б. Я. Информационные технологии [Текст]: учебник для прикладного бакалавриата / Б. Я. Советов, В. В. Цехановский. - 6-е изд., перераб. и доп. – М.: Юрайт, 2015. - 263 с.

2. Информационные технологии [Текст]: учебник / Санкт-Петербургский государственный университет экономики и финансов; под ред. В. В. Трофимова. - М.: Юрайт, 2011. - 624 с.

3. Сидоркина, И. Г. Системы искусственного интеллекта [Текст]: учебное пособие / И. Г. Сидоркина. - Москва: КНОРУС, 2016. - 246 с.

4. Гущин, А. Н. Базы данных [Электронный ресурс]: учебно-методическое пособие / А. Н. Гущин. - 2-е изд., испр. и доп. - М.; Берлин: Директ-Медиа, 2015. - 311 с.

5. Савельев, А. О. HTML5. Основы клиентской разработки [Электронный ресурс]: учебное пособие / А. О. Савельев, А. А. Алексеев. - 2-е изд., испр. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 272 с.

6. Соколова, В.В. Разработка мобильных приложений [Электронный ресурс]: учебное пособие / В.В. Соколова; Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский Томский государственный университет». – Томск: Издательство Томского политехнического университета, 2015. – 176 с. Режим доступа: по подписке. – URL: http://biblioclub.ru/index.php?page=book&id=442808 (дата обращения: 05.08.2020). – Библиогр. в кн. – ISBN 978-5-4387-0369-3. – Текст : электронный.

7. Науменко, А.Е. Планирование карьеры: хрестоматия: / А.Е. Науменко. – Тюмень: Тюменский государственный университет, 2016. – 218 с. Режим доступа: по подписке. – URL: http://biblioclub.ru/index.php?page=book&id=574614 (дата обращения: 05.08.2020). – Библиогр. в кн . – ISBN 978-5-400-011832. – Текст : электронный.