Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»
Кафедра программного обеспечения информационных технологий

Л.В. БОЧКАРЁВА, Е.В. ШОСТАК

ПРИКЛАДНЫЕ И ИНТЕГРИРОВАННЫЕ ПАКЕТЫ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ
для студентов специальности
«Программное обеспечение информационных технологий»
дневной формы обучения

Минск 2004

УДК 681.3.061(075.8)
ББК 32.973 я 73
Б 86

Р е ц е н з е н т:
старший научный сотрудник ОИПИ НАН Беларуси,
кандидат технических наук А.А. Несенчук

Бочкарёва Л.В.

Б 86 Прикладные и интегрированные пакеты: Учебно-метод. пособие для студ. спец. «Программное обеспечение информационных технологий» дневн. формы обуч. / Л.В. Бочкарёва, Е.В. Шостак. – Мн.: БГУИР, 2004. – 35 с.

ISBN 985-444-634-4

Рассмотрены создание и применение объектов СУБД Access: таблиц, форм, запросов и отчетов. Изложены основы языка Visual Basic for Applications. Предложены способы разработки приложений на Visual Basic for Applications и лабораторные работы по курсу “Прикладные и интегрированные пакеты”. Материалы, использованные в пособии, были разработаны в ОИПИ НАН Беларуси.

УДК 681.3.061(075.8)
ББК 32.973 я 73

ISBN 985-444-634-4

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ В СУБД ACCESS

Приложение Microsoft Access является мощной и высокопроизводительной 32-разрядной системой управления реляционными базами данных (СУРБД). Access предназначен для создания настольных БД и приложений БД архитектуры клиент/сервер, работающих под управлением операционной системы Windows. Access является компонентом MS Office и совместим с такими его продуктами, как Excel и Word.

Одна из важных причин популярности Access заключается в том, что, являясь по сути настольной БД, Access вобрал в себя многие возможности СУРБД архитектуры клиент/сервер, называемых SQL БД. Начиная с Access 97, вместо макросов в нем используется Visual Basic for Applications (VBA). VBA достаточно прост и имеет большой набор средств для искушенных разработчиков.

Access осуществляет следующие четыре основные функции, что позволяет классифицировать его как полную, функционально законченную СУБД.

1. *Организация данных*. Функция включает в себя создание таблиц данных и управление ими. Для работы с данными выделен специальный режим – режим таблиц.

2. *Связывание таблиц и обеспечение доступа к данным*. Access позволяет связывать таблицы по совпадающим значениям полей. Для этого используются запросы.

3. *Добавление и изменение данных*. Для этого в Access используются формы, которые позволяют разработчикам контролировать представление данных.

4.*Представление данных*. Для этого в Access применяются отчеты.

База данных в Access – это совокупность структурированных и взаимосвязанных данных и методов, обеспечивающих добавление, изменение, выборку и отображение данных. Основной объект БД – это таблица. Каждая таблица имеет пять свойств:

- *Описание* (необязательный комментарий).
- *Условие на значение.* Требование к данным, вводящимся в запись, необходимое для целостности и непротиворечивости данных.
- *Сообщение об ошибке.* Текст, выводимый в случае неверно введенных данных.
- *Фильтр.* Подмножество данных, выводящихся после применения фильтра.
- *Порядок сортировки.* Порядок следования записей в таблице.

В бланке таблицы можно установить следующие свойства полей таблицы:

1. Имя поля.
2. Тип данных.
3. Описание.
4. Ключевое поле.

Эти свойства заполняются в верхней части бланка. Есть еще ряд свойств, которые зависят от типа данных и являются необязательными (размер поля, формат, число десятичных знаков и др.).

1.1. Способы создания таблиц

В состав Access входит ряд средств для создания объектов БД. В частности, таблицы можно создавать с помощью ***мастера таблиц***. В этом случае таблица строится на основе имеющихся образцов, т.е. выбираются поля из таблицы-образца, назначаются ключевые поля, указываются связи с другими таблицами, если они существуют. ***Мастер таблиц*** – это средство быстрого создания таблиц, однако такой способ имеет ограниченные возможности.

Второй путь создания таблиц – это ***режим таблиц***. Access открывает пустую таблицу, имеющую по умолчанию 20 полей и 30 записей. В эту таблицу следует ввести информацию. ***Режим таблиц*** имеет ограниченное применение, так как поля таблицы не имеют содержательных имен, Access не всегда может верно определить тип данных, такая таблица не может содержать объекты OLE. Следовательно, таблицу приходится корректировать. ***Режим таблицы*** используется для просмотра, поиска, редактирования, дополнения и удаления данных. Оптимальным для создания таблиц считается способ при помощи ***конструктора.*** В ***режиме конструктора*** характеристики каждого поля таблицы изображаются в формате, похожем на формат электронной таблицы. В этом режиме в окне таблицы отображается список ее полей и вкладки свойств полей. Можно изменять значения свойств полей и таблицы в целом, задавать формат вывода значений полей и т.д. В окне свойств таблицы задается ее описание, условия на значения полей, текст сообщений об ошибках. В ***режиме конструктора*** выдается информация об индексации таблицы. Поле или поля таблицы, образующие первичный ключ, помечаются маркером ключевого поля. При построении БД связь между таблицами обеспечивается посредством ключевых полей. В большинстве случаев связывают ключевое поле одной таблицы (главной) с полем внешнего ключа, имеющего то же имя, другой таблицы (связанной). Между таблицами могут образовываться четыре вида отношений: один_к_одному, один_ко_многим, многие_к_одному, многие_ко_многим. Access обеспечивает ссылочную целостность данных.

1.2. Формы

Формы позволяют создавать пользовательский интерфейс для таблиц баз данных. Хотя для выполнения тех же функций можно использовать режим таблиц, формы предоставляют преимущества для представления данных в упорядоченном и привлекательном виде. Формы создаются из набора отдельных элементов, называемых элементами управления, или управляющими объектами. Форма состоит из окна, в котором размещаются элементы следующих типов: динамические, отображающие данные из таблиц, и статические – метки и логотипы.

Содержание и вид формы зависят от того, в каком приложении для работы с БД она используется. Приложения можно разделить на три основные категории.

1. *Управление транзакциями.* Такие приложения выполняют функции добавления новых записей в таблицы или изменения существующих записей. Эти приложения требуют наличия доступа «Для записи» к таблицам БД, которые присоединены к форме.

2. *Доступ к данным.* Приложения, входящие в эту категорию, предназначены для представления такой информации, как диаграммы, отчеты, статистические сведения, таблицы или отдельные элементы данных, но не позволяют пользователю добавлять или редактировать данные. Указанные приложения для работы с данными требуют наличия доступа «Для чтения» к таблицам БД, присоединенным к форме.

3. *Администрирование БД.* Приложения этой категории выполняют административные функции, такие, как создание БД или таблиц, разграничение прав доступа пользователей к объектам БД, обеспечение безопасности с помощью шифрования, периодическое уплотнение БД, а также операции резервного копирования. Приложения для администрирования БД требуют наличия полного доступа ко всем объектам, содержащимся в БД.

Форма строится на основе запросов и таблиц. Главная форма может использовать единственную таблицу в качестве источника данных и, кроме этого, включать подчиненные формы, опирающиеся на другие таблицы.

Создание формы при помощи ***мастера форм***. Таким образом можно создавать как одну форму, так и с подчиненными. Сначала выбирается главная таблица и ее поля, которые будут входить в родительскую форму, затем – подчиненная и ее поля. После чего указывается, по полям какой формы будет выполняться представление и параметры представления, например, «ленточное». Созданную ***мастером*** форму можно корректировать в режиме конструктора. Форма делится на три области: заголовок, область данных и примечание.

Мастер форм использует всего несколько элементов управления для создания форм. Имеется еще целый ряд элементов управления, которые становятся доступными в режиме конструктора и выполнения. Для создания многотабличной формы необходимо создавать запрос, который будет источником данных для формы. Запрос соединит данные из таблиц в унифицированный источник данных для использования формой. При создании формы без помощи ***мастера форм*** Access предоставляет по умолчанию пустую форму, к которой добавляются элементы управления.

Макет формы может быть трех видов.

1. *Простая форма* позволяет отображать на экране одновременно только одну запись.

2. *Ленточная форма* отображает столько записей, сколько помещается в текущем окне.

3. *Табличная форма* выводит записи в виде таблицы.

Для сокращения времени и ошибок при вводе информации в формах используются такие элементы управления, как списки и поля со списками. Источником информации для списков являются таблицы и запросы.

1.3. Запросы

Запросы являются важным инструментом в любой СУБД. Они используются для выделения, обновления и добавления новых записей в таблицы. Чаще всего запросы используются для выделения специфических групп записей, удовлетворяющих определенному критерию. Кроме того, их можно применять для получения данных из различных таблиц, обеспечивая единое представление связанных элементов данных. В Access существует четыре типа запросов для различных целей.

1. *Запросы на выборку* отображают данные из одной или нескольких таблиц в виде таблицы.

2. *Перекрестные запросы* собирают данные из одной или нескольких таблиц в формате, похожем на формат электронной таблицы. Эти запросы используются для анализа данных и создания диаграмм, основанных на суммарных значениях числовых величин из некоторого множества записей.

3. *Запросы на изменение* используются для создания новых таблиц из результатов запроса и для внесения изменений в данные существующих таблиц. С их помощью можно добавлять или удалять записи из таблицы и изменять записи согласно выражениям, задаваемым в режиме конструктора запроса.

4. *Запросы с параметрами* – это такие запросы, свойства которых изменяются пользователем при каждом новом запуске. Данный тип запросов не является обособленным, т.е. параметр можно добавить к запросу любого типа.

Простейший тип запроса – это запрос на выборку. Для его создания выбирается таблица, по которой строится запрос, а затем поля из нее. Если не задано условие отбора, запрос возвращает все записи таблицы. Результат запроса представляется в виде таблицы или результирующего набора записей, которые Access интерпретирует как динамический набор, допускающий обновление и хранящийся в оперативной памяти. После завершения конструирования запроса его следует сохранить в файл БД.

Выполнение запросов невозможно без связывания таблиц. Access поддерживает четыре типа их соединений.

Внутреннее соединение обычно используется при создании запросов на выборку (по образцу). Результирующее множество запроса содержит записи одной таблицы, имеющие совпадающие значения в связанных полях другой таблицы. Соединения основываются на уникальном значении поля первичного ключа в одной таблице и значении поля внешнего ключа в другой таблице, если они связаны отношением один_ко_многим. Если в таблице со стороны «многие» искомые записи отсутствуют, то соответствующие записи со стороны «один» в результирующее множество не включаются.

Внешнее соединение используется для создания новой таблицы, содержащей записи, исключая повторяющиеся, связанные поля которых совпадают. Внешнее соединение позволяет вывести данные одной из таблиц независимо от того, имеются ли соответствующие записи в другой таблице.

Рекурсивное соединение связывает данные в одной таблице. Создание этого типа соединения выполняется путем добавления в запрос копии таблицы и связывания полей идентичных таблиц.

Соединение по отношению связывает данные некоторым отношением, исключая равенство.

1.4. Отчеты

Конечным продуктом большинства приложений БД является отчет. Он представляет собой специальный тип непрерывных форм, предназначенных для печати. Для построения отчета Access комбинирует данные в таблицах, запросах и формах. Создаваемые в Access отчеты могут иметь шесть типов макетов.

1. *Отчет в одну колонку* представляет собой один длинный столбец текста, содержащий значения всех полей каждой записи таблицы или запроса. Этот отчет неэкономичный с точки зрения траты бумаги.

2. *Ленточный отчет,* в котором для каждого поля таблицы или запроса выделяется столбец, а значения всех полей записи выводятся по строкам.

3. *Многоколоночный отчет* строится из отчета в одну колонку при использовании колонок “газетного” типа. Этот отчет более экономичный, чем одноколоночный, но неудобный для восприятия информации.

4. *Групповой (итоговый) отчет* – в нем объединяются данные для групп записей, а в конце указываются итоговые значения.

5. *Почтовые наклейки* – это специальный тип многоколоночного отчета, предназначенного для печати имен и адресов в группах. Каждая группа полей образует ячейку в сетке.

6. *Несвязанные отчеты,* в которых содержатся подчиненные отчеты, основанные на несвязанных источниках данных.

Первые четыре типа отчетов называются связанными с источниками данных. Основной отчет несвязанного отчета не использует в качестве источника данных таблицы и запросы, однако подчиненные запросы, входящие в него, должны опираться на источник данных.

Отчеты можно строить при помощи ***мастера отчетов***. Преимущество от его использования заключается в том, что “с нуля” создается базовый отчет, который затем можно изменять и дополнять. Для построения отчетов имеется ряд элементов управления, которые не используются ***мастером отчетов***. Корректировка отчета выполняется в режиме конструктора отчетов. Существуют более сложные отчеты, которые не удается создавать при помощи ***мастера отчетов***. Например, если требуется применять особые формы сортировки или группировки. Для включения в отчет подчиненных отчетов необходимо начинать с пустого отчета, а не созданного ***мастером отчетов***. Построение сложных отчетов выполняется в режиме конструктора и во многом совпадает с созданием сложных форм.

2. ВВЕДЕНИЕ В VISUAL BASIC FOR APPLICATIONS

Сегодня Visual Basic for Applications (VBA) применяется во всех наиболее популярных пакетах корпорации Microsoft: Excel, Project, Word, Mail, Power-Point, Access. В Visual Basic применяется объектный подход, что делает этот язык при разработке программ более понятным и легким. Visual Basic облегчает создание макропроцедур и совместно с OLE-технологией позволяет создавать “макропрограммы”, объединяющие и координирующие усилия многих мощных прикладных программ.

Для создания диалоговых окон, разрабатываемых приложений в VBA, используются формы. ***Редактор форм*** является одним из основных инструментов визуального программирования. Форма в проект добавляется с помощью команды ***Вставка, Форма***. В результате на экран выводится незаполненная форма с панелью инструментов ***Панель элементов***. Используя панель инструментов ***Панель элементов,*** из незаполненной формы можно сконструировать любое требуемое для приложения диалоговое окно. Размеры формы и расположенных на ней элементов управления можно изменять.

2.1. Данные VBA

Типы данных относятся к самым фундаментальным понятиям любого языка программирования. Тип данных определяет множество допустимых значений, которые может принимать указанная переменная.

В VBA имеются основные типы данных, приведенные в табл. 2.1.

Таблица 2.1

Тип данных	Размер(байт)
Byte (байт)	1
Boolean (логический)	2 (True или False)
Integer (целое число)	2
Long (длинное целое число)	4
Single (число с плавающей запятой обычной точности)	4
Double (число с плавающей запятой двойной точности)	8
Currency (денежный)	8
Decimal (масштабируемое целое число	14
Object (объект)	4
String (строка переменной длины)	10+длина строки
String (строка постоянной длины)	длина строки
Variant (числовые подтипы)	16
Variant (строковые подтипы)	22+длина строки
Тип данных, определяемый пользователем (с помощью ключевого слова Type)	Объем определяется элементами

Описание типа каждой переменной делает программу надежнее и, кроме того, ускоряет ее работу, т.к. в VBA не требуется тратить время на распознавание типа неописанной переменной при каждом обращении к ней.

Синтаксис:

Dim [WithEvents] ИмяПеременной [([Индексы])] [As [New] Tun] _
[, [WithEvents] ИмяПеременной [([Индексы])] [As [New] Tun]]…

В табл. 2.2 приведено описание аргументов, использованных с инструкцией ***Dim***

Таблица 2.2

Имя аргумента	Комментарий
WithEvents	Ключевое слово, указывающее, что аргумент ИмяПеременной является именем объектной переменной, которая используется при отклике на события
ИмяПеременной	Имя переменной, удовлетворяющее стандартным правилам именования переменных
Индексы	Размерности переменной массива
New	Ключевое слово, включающее возможность неявного создания объекта
Тип	Тип данных переменной. Для каждой описываемой переменной следует использовать отдельное предложение As Тип

Переменные, описанные с помощью ключевого слова ***Dim*** на уровне модуля, доступны для всех процедур в данном модуле. Переменные, описанные на уровне процедуры, доступны только в данной процедуре. Инструкция ***Dim*** предназначена для описания типа данных переменной на уровне модуля или процедуры. Например, следующая инструкция описывает переменную с типом ***Integer:***

Dim N As Integer.

Инструкция ***Dim*** предназначена также для описания объектного типа переменных. Далее приводится описание переменной для нового экземпляра рабочего листа

Dim X As New Worksheet.

Если при описании объектной переменной не используется ключевое слово ***New***, то для использования объекта, на который ссылается переменная, существующий объект должен быть присвоен переменной с помощью инструкции ***Set.*** Если тип данных или тип объекта не задан, и в модуле отсутствует инструкция ***DefTun,*** по умолчанию переменная получает тип ***Variant.***

Как и в других языках программирования, в VBA можно использовать массивы. Примеры объявления массивов приведены ниже.

Dim B(3,3) As Single

Dim A(12) As Integer.

Первая строка объявляет двухмерный массив 3*3 (матрицу), состоящий из действительных чисел. Вторая строка объявляет одномерный массив (вектор)

из 12 целых чисел, причем по умолчанию первый элемент массива будет ***A(0)***, а последний ***A(11)*** . В этом случае 0 – базовый индекс. Можно изменить базовый индекс, написав в начале листа модуля инструкцию ***Option Base 1.*** После этого индексы массивов ***A*** и ***B*** будут начинаться с 1. Другим способом изменения базового индекса является использование ключевого слова ***To*** при объявлении массива:

Dim B (1 To 3, 1 To 3) As Single
Dim A (1 To 12) As Integer.

Удобным способом определения одномерных массивов является функция ***Array***, преобразующая список элементов, разделенных запятыми, в вектор из этих значений, и присваивающая их переменной тип ***Variant*** . Например,

Dim A As Variant
A = Array (10,20,30)
B = A(2).

Наряду с массивами существует еще один способ создания структурного типа – тип, определенный пользователем, или запись. ***Запись*** – это совокупность нескольких элементов, каждый из которых может иметь свой тип. Элемент записи называется полем.

2.2. Элементы управления VBA

В объектно-ориентированном программировании данные и код, который манипулирует этими данными, объединены в структуре, называемой объект. Общие примеры объектов Visual Basic – это таблицы, области ячеек, командные кнопки, модули. Программный объект обладает определенными свойствами и методами. Свойства – это видимые характеристики объекта, а методы – операции преобразования этих данных. Видимыми характеристиками называются данные, которые могут быть доступны вне объекта. Свойствами считаются данные, которыми объект манипулирует или которые позволяют контролировать, как объект выглядит или как он себя ведет.

Элемент управления ***ListBox*** (список) создается с помощью кнопки ***Список*** (ListBox). Он применяется для хранения списка значений, которые затем будут использоваться в тексте программы. ***ListBox*** имеет целый ряд свойств, среди которых есть свойство ***RowSource.*** Оно устанавливает диапазон, содержащий элементы списка.

Элемент управления ***ComboBox*** (поле со списком) создается с помощью кнопки ***Поле со списком*** (ComboBox). ***ComboBox*** применяется для хранения списка значений. Он сочетает в себе функциональные возможности списка ***ListBox*** и поля. В отличие от ***ListBox*** в элементе управления ***ComboBox*** отображается только один элемент списка. Кроме того, у него отсутствует режим выделения нескольких элементов списка, но он позволяет вводить значение, используя поле ввода, как это делает элемент управления ***TextBox.***

В табл. 2.3 приведены методы и свойства элементов управления VBA.

Таблица 2.3

Имя	Комментарий
Visible	Допустимые значения: True (элемент управления отображается во время выполнения программы) и False (в противном случае)
Height, Width	Устанавливают геометрические размеры объекта (высоту и ширину)
Left, Top	Устанавливают координаты верхнего левого угла элемента управления, определяющие его местоположение в форме
BorderStyle	Устанавливает тип границы. Допустимые значения: ▪ FmBorderStyleSingle (граница в виде контура) ▪ FmBorderStyleNone (граница невидима)
Picture (создание картинки)	Внедряет картинку на элемент управления. Например, на поверхности кнопки картинка отображается с помощью следующей инструкции: CommandButton1.Picture = LoadPiture(“c:\my_doc\Круг.bmp”) Функция LoadPicture (ПолноеИмяФайла) считывает графическое изображение. Аргумент ПолноеИмяФайла указывает полное имя графического файла
Picture (удаление картинки)	После того как картинка создана на элементе управления, иногда возникает необходимость ее удалить. Это достигается присвоением свойству Picture значения LoadPicture(“”)
Move	Изменяет положение и размер объекта

Наиболее часто используемый объект – это пользовательская форма ***UserForm***, предоставляющая возможность создавать диалоговые окна разрабатываемых приложений. Она служит базой пользовательского диалогового окна, на которой в зависимости от решаемой задачи размещают требуемые элементы управления. Приведенные выше свойства и методы принадлежат объекту ***UserForm.***

Элемент управления ***TextBox*** (поле) создается с помощью кнопки ***Поле*** (TextBox) панели элементов управления. В основном ***TextBox*** используется для ввода текста, который затем используется в программе, или для вывода результатов расчетов программы. Текст, введенный в поле, обычно в программе преобразуется либо в числа, либо в формулы. В табл. 2.4 показаны основные свойства элемента управления ***TextBox***.

Таблица 2.4

Имя	Комментарий
1	2
Text	Возвращает текст, содержащийся в поле
Visible	Допустимые значения: True (поле отображается во время выполнения программы) и False (в противном случае)
Enabled	Допустимые значения: True (можно вносить изменения в содержание поля) и False (в противном случае)

Окончание табл. 2.4

1	2
Multiline	Допустимые значения: True (устанавливается многострочный режим ввода текста в поле) и False (однострочный режим)
WordWrap	Допустимые значения: True (устанавливается режим автоматического переноса) и False (в противном случае)
AutoSize	Допустимые значения: True (устанавливается режим автоматического изменения размера поля так, чтобы весь вводимый текст помещался в нем) и False (устанавливает фиксированный размер поля)
ScrollBars	Устанавливает режим отображения в поле полос прокрутки
SelLenght, SelStart SelText	Эти свойства характеризуют выделенный в поле фрагмент текста (длина, начало и сам фрагмент текста соответственно)
MaxLength	Устанавливает максимальное допустимое количество вводимых в поле символов. Если это свойство равно 0, то нет ограничений на вводимое количество символов
PasswordChar	Устанавливает символ, отображаемый при вводе пароля. Если это свойство определено, то вместо вводимых символов в поле будет отображаться установленный символ

Элемент управления ***Label*** (надпись) создается с помощью кнопки ***Надпись*** (Label) панели элементов управления. В основном ***Label*** используется для отображения надписей, например заголовков, не имеющих свойства Caption. Надпись не может быть изменена пользователем, но код программы во время ее выполнения может управлять текстом надписи. Основные свойства элемента управления ***Label*** приведены в табл. 2.5.

Таблица 2.5

Имя	Комментарий
Caption	Возвращает текст, отображаемый в надписи
Visible	Допустимые значения: True (поле отображается во время выполнения программы) и False (в противном случае)
Multiline	Допустимые значения: True (устанавливается многострочный режим ввода текста) и False (однострочный режим)
WordWrap	Допустимые значения: True (устанавливается режим автоматического переноса) и False (в противном случае)
AutoSize	Допустимые значения: True (устанавливается режим автоматического изменения размера поля так, чтобы весь вводимый текст помещался в нем) и False (устанавливается фиксированный размер поля)

Элемент управления ***CommandButton*** (кнопка) создается с помощью кнопки ***Кнопка*** (CommandButton). ***CommandButton*** в основном используется для инициирования выполнения некоторых действий, вызываемых нажатием кнопки. Например, запуск программы или остановка ее выполнения, печать результатов и т.д. Основные свойства элемента управления ***CommandButton*** собраны в табл. 2.6.

Таблица 2.6

Имя	Комментарий
Caption	Возвращает текст, отображаемый на кнопке
Cancel	Допустимые значения: True (устанавливаются отменяющие функции для кнопки, т.е. нажатие клавиши <Esc> приводит к тем же результатам, что и нажатие кнопки) и False (в противном случае)
Visible	Допустимые значения: True (кнопка отображается во время выполнения программы) и False (в противном случае)
Enabled	Допустимые значения: True (запрещено нажатие кнопки пользователем) и False (в противном случае)
Accelerator	Назначает клавишу, при нажатии которой одновременно с клавишей <Alt> происходит запуск действий, связанных с кнопкой. Например, CommandButton1.Accelerator = "C"
Picture	Внедряет на поверхность кнопки картинку. Например, CommandButton1.Picture = LoadPicture("c:\ Круг.bmp") Функция LoadPicture считывает графическое изображение
Default	Задает кнопку по умолчанию, т.е. устанавливает ту кнопку, для которой действия, связанные с ней, будут выполняться при нажатии клавиши <Enter>

2.3. Файлы последовательного и произвольного доступов

В Access VBA существует два основных типа дисковых файлов – последовательного и произвольного доступа. Файлы первого типа читаются и пишутся последовательно от начала файла до его конца. Для того чтобы получить доступ к определенной части информации в последовательном файле, необходимо читать последовательно от начала файла запись за записью, пока не будет найдена требуемая информация. Последовательный файл можно либо читать, либо писать.

Любой файл, который читается в память целиком, должен быть последовательным. Большинство текстовых файлов, так же как и файлов программ, являются последовательными. Данные в текстовых файлах VBA записаны в форме строк символов ANSI. Последовательный файл VBA пригоден для чтения в

текстовом процессоре. Последовательные файлы не эффективны для хранения числовых величин, так как числа в них хранятся в символьном формате.

Открытие файла связано с подключением к нему файлового числа. ***Файловые числа*** – это небольшие целые числа, которые связываются с файлом при его открытии. Команды чтения и записи используют файловые числа для указания файла, в который надо писать или из которого надо читать. Для открытия последовательного файла используется оператор ***Open,*** имеющий следующий формат: ***Open имя_ файла For режим As файловое_число.***

Аргумент **имя_ файла** – это строка, содержащая имя и путь к требуемому файлу. Аргумент **режим** определяет тип файла, который необходимо открыть, и способ открытия файла. Режим должен принимать одно из следующих значений: ***Input, Output, Append. Input*** открывает файл для чтения, ***Output*** – для записи. Оба режима открывают файл с его начала. ***Append*** открывает файл с конца для добавления новых записей. Попытки открыть несуществующий файл в режиме ***Input*** приведут к ошибке, в режимах ***Output*** или ***Append*** – к созданию файла. Аргумент ***файловое_число*** определяет число, которое присваивается файлу. Обычно первому файлу присваивают 1, второму – 2 и т.д. Если файл закрывается, то освободившееся файловое число может быть использовано повторно. Файловое число должно быть уникальным в каждом файле.

Закрытие файла выполняется с помощью оператора ***Close***. В качестве аргумента оператора ***Close*** используется ***файловое_число,*** указывающее на то, какой файл следует закрыть. Если опустить аргумент ***файловое_число***, то будут закрыты все открытые файлы.

Для записи данных в открытый файл можно использовать оператор ***Print***. Если поместить в строку для печати несколько элементов, разделенных запятыми, то они будут напечатаны в блоках, заполненных пробелами и разделенных фиксированными табулостопами через каждые 14 символов. Если вместо запятых использовать символ (;), то напечатанные элементы не будут содержать пробелов. Если разделить символом (;) числа в операторе ***Print***, то напечатанная строка будет содержать по два пробела между числами. Пробелы появляются в результате преобразования каждого числа в строку текста с добавлением к нему ведущего и конечного пробелов. Для управления печатью используется функция ***Format().*** Она получает два аргумента, а возвращает отформатированную строку. Первый аргумент – это переменная, содержащая число, которое нужно напечатать, а второй – строка форматирования, определяющая форму, в которой нужно напечатать указанное число. Например,

Format(a, “#,###.00”).

Символы (#) используются в качестве цифровых заполнителей, (,) указывают на то, что каждые три символа отделяются запятой, (.) указывает положение десятичной точки, а (0) – позиции обязательного символа.

Для печати в дисковый файл после зарезервированного слова ***Print*** необходимо записать символ (#), файловое число и запятую:

Print #файловое_число, список_аргументов,

где ***файловое_число*** определяет открытый файл, а ***список_аргумен*****тов** содержит то, что необходимо напечатать.

При использовании оператора ***Print*** данные форматируются в пригодную для чтения форму, подобную текстовому документу. Однако если предполагается читать эти данные программами на Visual Basic, то их следует записывать оператором ***Write***. В отличие от оператора ***Print*** оператор ***Write*** сохраняет существовавшие между данными кавычки и разделяет выводимые данные запятыми, что упрощает для Visual Basic определение конца одного элемента данных и начала другого. При работе с оператором ***Write*** вместо конкретного файлового числа используется функция ***FreeFile,*** позволяющая определить свободное файловое число. Оно сохраняется в переменной ***Fnum***, которая затем используется везде, где требуется файловое число.

Для чтения данных из файла применяются операторы ***Input*** и ***LineInput***. Первым оператором каждого из них являются символ (#) и файловое число, затем следует запятая, отделяющая указатель файла, подлежащего чтению. Далее следуют аргументы – имена переменных, которым присваиваются прочитанные из файла значения. В операторе ***Input*** переменные, следующие после запятой, определяют порядок и количество данных, которые будут прочитаны из файла. Пробелы и табуляторы, расположенные в начале строки, игнорируются. Если переменная должна принимать числовые значения, то Visual Basic пытается прочитать из файла число. Первый непустой символ рассматривается как начало числа, его чтение продолжается до тех пор, пока не встретится какой-нибудь символ, который не может быть частью числа. Если переменная является строковой, то Visual Basic читает ее, начиная с первого непустого символа, и прекращает чтение, если встречает запятую или конец строки. Если первый непустой символ является символом (”), то первым символом, прочитанным в строку, будет первый символ после символа (”) и чтение будет продолжено до тех пор, пока не встретится второй символ (”) или конец строки, т.е. будут прочитаны все символы, заключенные в кавычки, включая запятые.

Следовательно, операторы ***Write*** и ***Input*** спроектированы для совместной работы в целях записи данных в файл и последующего их чтения. Оператор **Print** спроектирован для создания текстового файла, подлежащего чтению пользователем.

Оператор ***Line Input*** является дополнением оператора ***Input***. Оператор ***Line Input*** присваивает единичному строковому аргументу все символы, обнаруженные в единичной строке файла. Строковому аргументу в виде переменной присваиваются все символы, включая начальные пробелы, запятые и кавычки. Оператор ***Line Input*** применяется для чтения в программу точной копии содержимого строки файла. Единственными символами, не пересылаемыми оператором ***Line Input***, являются символы возврата каретки и перевода строки, рассматриваемые как конец строки.

Кроме описанных выше файлов последовательного доступа, в Access VBA существует файлы произвольного доступа. Они основаны на записях постоянной длины, что позволяет читать и писать информацию произвольно в любом порядке. Файл произвольного доступа можно читать и писать одновременно.

Большинство программ работы с базами данных используют файлы произвольного доступа, так как они позволяют напрямую прочесть любую запись в

базе данных без предварительного чтения всех предшествующих ей записей файла. После чтения записи программа базы данных может изменить эту запись и записать ее на то же место на диске, не разрушая другие записи файла. Ограничением файлов произвольного доступа является фиксированная длина записи. Однако это ограничение позволяет VBA быстро обнаружить запись на диске. Если бы записи не имели фиксированной длины, то пришлось бы поддерживать отдельный индекс, определяющий, где кончается одна запись и начинается другая.

Каждая запись в файле произвольного доступа является независимой от всех других его записей. Длина записи определяется в операторе ***Open***, а тип данных, указанный пользователем, определяет ее содержимое. Файлы произвольного доступа используются в основном для хранения двоичных представлений чисел, а не текстов. Применение двоичных чисел экономит пространство файла по сравнению с символьным представлением тех же чисел. Кроме того, двоичные числа загружаются в память компьютера значительно быстрее, так как не требуют преобразования перед использованием.

Открытие файла произвольного доступа выполняется оператором ***Open***, где аргумент режима всегда равен ***Random***, и в конце оператора находится аргумент ***Len=длина_записи***. Формат оператора ***Open*** имеет следующий вид:

Open имя_файла For Random As файловое_число Len=длина_записи,

где ***имя_файла*** – строка, содержащая имя и путь файла, подлежащего открытию; ***файловое_ число*** – число, которое необходимо присвоить файлу; ***длина_записи*** – длина записи произвольного доступа в байтах.

Оптимальная длина записи должна либо быть кратной размеру дискового сектора, либо делить указанный размер нацело. Большинство дисковых накопителей имеют секторы размером 512 или 1024 байт; следовательно, оптимальная длина записи должна быть степенью 2. Такая длина является целесообразной, поскольку данные записываются и читаются секторами одинаковой длины.

Следующие операторы открывают файл произвольного доступа для чтения или записи

FileNum1=FreeFile

Open ”myfile.txt” For RandomAs FileNum1 Len=256.

Этот блок кода открывает файл произвольного доступа ***myfile.txt*** с записями длиной 256 байт. Функция ***FreeFile*** выбирает свободное файловое число.

Закрытие файлов произвольного доступа осуществляется оператором ***Close:***

Close # файловое_число,

где ***файловое_число*** – файловое число, указанное при открытии этого файла.

Кроме объявления типа, определяемого пользователем, оператор ***Type*** применяется также для объявления содержимого переменных типа запись, используемых для работы с файлами произвольного доступа. В этом случае при использовании оператора ***Type*** не выбирается тип ***Variant*** и учитывается, что все строковые переменные должны иметь постоянную длину. Максимальная длина записи, определенной оператором ***Type***, должна быть не больше длины записи, объявленной оператором ***Open***. Если в операторе ***Type*** использовать

строки переменной длины, то для указания длины строки потребуются два дополнительных байта на каждую строку. Если использовать переменные типа ***Variant,*** то также нужно будет добавить два байта к байтам, содержащим значение переменной. Если используются строки переменной длины и переменные типа ***Variant*** в типе, который должен быть сохранен в записи, то общая длина типа всегда должна быть не больше длины записи. В противном случае произойдет ошибка. Поскольку невозможно быть уверенным в том, что общая длина переменных в записи будет соответствовать данному условию, длина каждой переменной обычно объявляется в типе. Чтобы узнать длину типа, определяемого пользователем, необходимо либо сложить длины всех переменных, либо объявить переменную указанного типа, а затем применить к ней функцию ***Len().***

В отличие от файлов последовательного доступа для сохранения данных в записях файлов произвольного доступа операторы ***Print*** и ***Write*** не применяются. Вместо них используется оператор ***Put***. Он имеет три аргумента – файловое число, номер записи и переменную типа запись

Put #файловое_число, номер_записи, переменная.

Следующий оператор ***Put*** сохраняет содержимое ***theDB(1)*** в третьей записи файла, открытого с файловым числом 1:

Put #1, 3, theDB(1).

Если опустить номер записи в операторе ***Put,*** то будет использован номер, следующий за тем, который был задействован в последнем из предыдущих операторов ***Get, Put, Seek.***

Для выборки записи из дискового файла и загрузки ее в переменную типа запись применяется оператор ***Get,*** который имеет такой же синтаксис, как и оператор ***Put.***

Get #файловое_число, номер_записи, переменная.

Файлы произвольного доступа предоставляют следующую возможность для реорганизации данных: использование индекса, что позволит реорганизовывать индекс, а не сам файл. Для создания индекса вначале создается массив целых чисел, в котором каждый элемент содержит свой номер записи файла произвольного доступа. Затем этот массив используется для получения доступа к записи файла. Такая организация хранения записи называется индексированием записи.

Например, если переменная ***I*** содержит номер требуемой записи, а переменная типа запись называется ***RecVar,*** то обычный доступ к файлу обеспечивается следующим оператором ***Get:***

Get #fileNum, I, RecVar.

Если массив индексов называется ***theIndex(),*** то метод доступа изменяется таким образом: ***Get #fileNum, theIndex(I), RecVar.***

Первоначально каждый элемент массива ***theIndex()*** содержит свой собственный номер. Результат использования такого массива для доступа к записям файла идентичен прямому доступу к записям. Предположим, потребовалось упорядочить записи в алфавитном порядке, для чего необходимо поменять местами некоторые записи, что выполняется путем изменения соответствующих

значений в массиве индексов. В результате доступ к записям файла будет осуществляться в соответствии с указанным порядком индексов, а не в порядке их физического размещения на диске. Индексированные записи удобны для группирования и повторного использования удаленных записей. Для удаления записи ее номер перемещается в конец массива индексов, а затем оставшиеся индексы перемещаются на один элемент вверх. Следующий индексный номер определяется на границе использованных и свободных записей. Если необходимо создать новую запись, то сначала следует проверить, имеет ли массив индексов записи, пригодные для повторного использования. Индексы можно сохранять в отдельном последовательном файле или размещать в виде нескольких записей в начале файла произвольного доступа.

2.4. Создание и применение процедур VBA

Каждая процедура начинается с оператора объявления процедуры ***Sub*** и заканчивается оператором ***End Sub***:

Sub имя_процедуры (аргументы)

…

тело процедуры

…

End Sub

Оператор объявления процедуры присваивает ей имя, отмечает ее начало и перечисляет аргументы, которые передаются процедуре при вызове из программы. Оператор ***End Sub*** отмечает конец процедуры. Все, что расположено между этими двумя операторами, называется телом процедуры и реализует возложенную на процедуру задачу. Список аргументов обеспечивает связь между вызывающей и вызываемой процедурами. Хотя те или иные переменные, объявленные глобальными, доступны обеим процедурам, некоторые специфические переменные должны передаваться и приниматься в качестве списка аргументов.

Существует четыре типа процедур:

1. Общие процедуры.
2. Командные процедуры.
3. Процедуры обработки событий.
4. Функции.

Общие процедуры – это стандартные процедуры Visual Basic, которые, в общем, ничего не изменяют вне своего тела. Они вычисляют переменные, отображают документы, передают сообщения другим программам и манипулируют дисковыми файлами.

Командные процедуры расширяют возможности прикладных программ на Visual Basic. В частности, они расширяют возможности Excel, касающиеся рабочих папок или их содержимого. Поэтому функционирование таких процедур аналогично выполнению директив меню Excel. Эти процедуры представляют собой записанные действия с таблицей. Командные процедуры обычно не получают никаких аргументов. Если им нужны какие-либо данные от пользователя, они выводят диалоговое окно.

Процедуры обработки событий связаны с конкретными событиями и выполняются, когда эти события происходят. Событие – это нажатие командной кнопки, выполнение директивы меню, открытие или закрытие таблицы Excel, изменение содержимого диалогового окна и т.п. Данные процедуры могут быть также и командными или общими процедурами.

Подключение процедуры к событию, например, к командной кнопке, делает ее обрабатывающей событие. В данном случае процедура будет обрабатывать событие, связанное с нажатием командной кнопки. Чтобы процедура обработки событий запускалась на выполнение всякий раз, когда пользователь активизирует таблицу, необходимо приравнять имя процедуры к свойству ***OnSheetActivate*** конкретного объекта таблицы. Например, оператор ***Worksheets("Sheet1").OnSheetActivate="InitIt"*** запускает процедуру ***InitIt*** каждый раз, когда активизируется ***Sheet1***. Если используется объект ***Application***, процедура будет выполняться всегда, когда активизируется таблица. Аналогично можно использовать свойство ***OnSheetDeactivate*** для выполнения процедуры всякий раз, когда деактивизируется таблица, т.е. она закрывается или активизируется другая таблица.

3. ЛАБОРАТОРНЫЕ РАБОТЫ

ЛАБОРАТОРНАЯ РАБОТА №1

РАЗРАБОТКА СХЕМЫ БАЗЫ ДАННЫХ ACCESS, СОЗДАНИЕ ТАБЛИЦ

Цель работы:

- научиться строить схемы баз данных;
- научиться строить таблицы различными средствами Access;
- получить навыки работы с данными таблиц: фильтрация, сортировка записей.

Задание:

Разработать схему базы данных по предложенной тематике. Создать такие объекты БД, как таблицы, и наполнить содержанием БД. В процессе построения таблиц использовать различные средства Access: ***мастер таблиц*** и ***конструктор***.

Порядок выполнения работы

1. Изучить теоретическую часть лабораторной работы.
2. Создать требуемое приложение.
3. Оформить отчет по выполненной работе.

ЛАБОРАТОРНАЯ РАБОТА №2

ПОСТРОЕНИЕ ФОРМ

Цель работы:

- научиться строить формы различными средствами Access;
- научиться работать с данными форм: добавление, изменение, удаление записей;
- изучить элементы управления, предназначенные для создания форм.

Задание:

Разработать формы для построения пользовательского интерфейса таблиц, созданных в лабораторной работе №1. Для эффективной работы с формами использовать разнообразные элементы управления.

Порядок выполнения работы

1. Изучить теоретическую часть лабораторной работы.
2. Создать требуемое приложение.
3. Оформить отчет по выполненной работе.

ЛАБОРАТОРНАЯ РАБОТА №3

ВЫПОЛНЕНИЕ ЗАПРОСОВ, ПОСТРОЕНИЕ ОТЧЕТОВ

Цель работы:

- научиться строить запросы различных типов;
- изучить различные виды связывания таблиц для выполнения запросов;
- научиться строить отчеты различных типов.

Задание:

Для таблиц, построенных в лабораторных работах № 1,2, выполнить различные типы запросов: на выборку, перекрестный, на изменение, с параметром. Оформить и распечатать построенную БД в виде отчетов различных типов.

Порядок выполнения работы

1. Изучить теоретическую часть лабораторной работы.
2. Создать требуемое приложение.
3. Оформить отчет по выполненной работе.

ЛАБОРАТОРНАЯ РАБОТА №4

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ КОМПЬЮТЕРНОЙ ИГРЫ

Цель работы:

- научиться создавать из формы диалоговое окно;
- программировать контроль ввода информации в поля;
- научиться управлять вводом и выводом данных в поля;
- получить навыки работы с функцией генератора случайных чисел;
- научиться запускать программы на выполнение.

Задание:

Разработать приложение для игры «В двенадцать». Последовательно до трех раз бросается игральная кость. Игрок может бросить игральную кость один раз или, оценив результат первого броска, – выполнить его во второй раз, или, помня результаты двух предыдущих попыток, – бросить кость в третий раз. Все выпавшие очки суммируются. Если сумма 12 и более очков, то игрок проигрывает и игра заканчивается. Если сумма менее 12 очков, в игру вступает компьютер. Компьютеру известно только то, что игрок набрал менее 12 очков. Компьютер, как и игрок, может совершить до трех попыток бросания игральной кости. Если он наберет 12 и более очков, то проигрывает. Если компьютер набирает менее 12 очков, очки, набранные игроком и компьютером, сравниваются. Побеждает тот, у кого большая сумма очков. При равенстве очков – ничья.

Рекомендации по выполнению задания

В качестве примера рассмотрим построение простейшей компьютерной игры «Орел и решка». В игре участвуют игрок и компьютер. Игрок единожды вносит в банк определенную сумму денег. Игра состоит из последовательности шагов, которая может быть бесконечной. На очередном шаге игрок загадывает либо орел, либо решку. Компьютер "бросает монету". Если выпадает значение, загаданное игроком, то банк увеличивается на единицу, в противном случае – уменьшается на единицу. Игра заканчивается либо по желанию игрока, либо, когда величина банка становится 0 или больше 10000. Игрок забирает себе содержимое банка. В приложении отслеживаются максимальные и минимальные суммы, которые были в банке в течение игры. Для решения поставленной задачи необходимо выполнить следующие действия:

1. Выберите команду ***Сервис, Макрос, Редактор Visual Basic*,** после чего откроется окно редактора Visual Basic.

2. Выберите команду ***Вставить User Form***.

3. Используя панель элементов и окно свойств, заполните пользовательскую форму элементами управления, создав требуемое диалоговое окно приложения.

Для написания кода программы, связанного с пользовательской формой, достаточно дважды щелкнуть кнопкой ***Бросание монеты*** по форме (рис. 4.1). Откроется редактор кода на листе модуля ***UserForm1.***

```
Dim Банк As Long
Dim Партия As Long
        Dim Номермаксимум As Long
Dim Номерминимум As Long
Dim Максимум As Long
Dim Минимум As Long
Private Sub CommandButton1_Click()
Партия = Партия + 1
TextBox1.Enabled = False
If IsNumeric(TextBox1.Text) = False   Then
MsgBox "Введите ставку",  _
vbExclamation, "Орел и решка"
   TextBox1.Enabled = True
   TextBox1.SetFocus
   Exit Sub
End If
Банк = CLng(TextBox1.Text)
If Банк > 10000 Or Банк <= 0 Then
MsgBox "Ставка должна быть в диапазо-
не[1,10000]",  _
   vbExclamation, "Орел и решка"
    TextBox1.Enabled = True
    TextBox1.SetFocus
     Exit Sub
End If
Randomize
Монета = Int(2 * Rnd)
If OptionButton4.Value = True Then
 If Монета = 0 Then
 Банк = Банк - 1
 TextBox1.Text = CStr(Банк)
End If
If Монета = 1 Then
 Банк = Банк + 1
TextBox1.Text = CStr(Банк)
  End If
End If
If OptionButton5.Value = True Then
If Монета = 1 Then
   Банк = Банк - 1
   TextBox1.Text = CStr(Банк)
End If
  If Монета = 0 Then
     Банк = Банк + 1
   TextBox1.Text = CStr(Банк)
   End If
End If
 TextBox2.Text = CStr(Партия)
If Банк > Максимум Then
   Максимум = Банк
   Номермаксимум = Партия
  TextBox3.Text = CStr(Максимум)
TextBox5.Text = CStr(Номермаксимум)
    End If
If Банк < Минимум Then
   Минимум = Банк
   Номерминимум = Партия
   TextBox4.Text = CStr(Минимум)
TextBox6.Text = CStr(Номерминимум)
End If
End Sub
Private Sub CommandButton2_Click()
    UserForm1.Hide
End Sub
Private Sub UserForm_Initialize()
  Максимум = 0
  Минимум = 10000
   Партия = 0
   TextBox1.Enabled = True
   TextBox2.Enabled = False
   TextBox3.Enabled = False
   TextBox4.Enabled = False
   TextBox5.Enabled = False
   TextBox6.Enabled = False
  OptionButton4.Value = True
End Sub
```

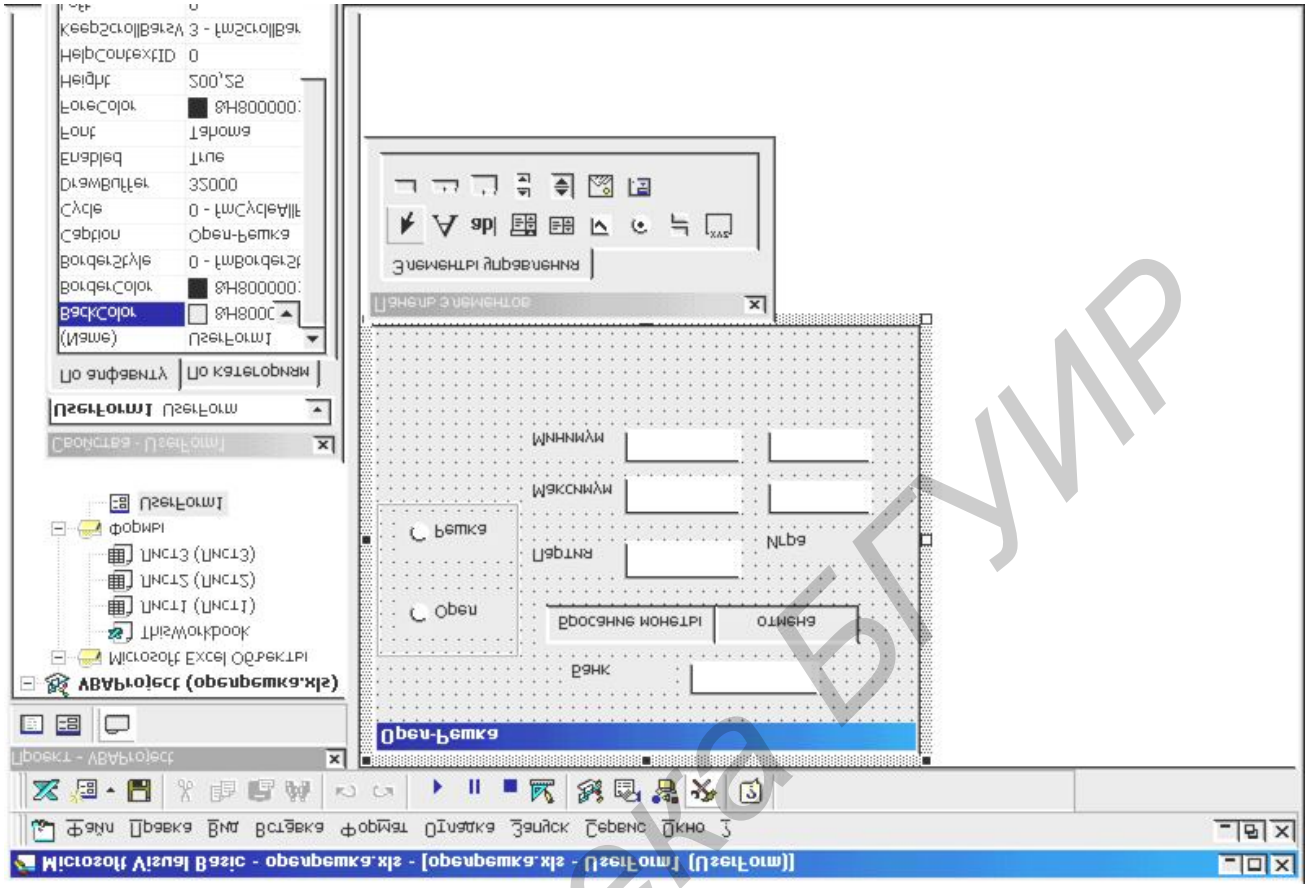


Рис. 4.1. Пользовательская форма

Порядок выполнения работы

1. Изучить теоретическую часть лабораторной работы.
2. Создать требуемое приложение.
3. Оформить отчет по выполненной работе.

ЛАБОРАТОРНАЯ РАБОТА №5

РАБОТА СО СПИСКАМИ И ЭЛЕМЕНТАМИ УПРАВЛЕНИЯ

Цель работы:

- научиться заполнять списки;
- научиться выбирать несколько элементов из списка;
- выполнять специфицированную операцию над выбранными элементами из списка;
- научиться работать с объектами, их свойствами и методами;
- выполнять заданные действия, используя методы и свойства элементов управления.

Задание 1

Составьте программу нахождения среднего балла студентов, выбранных из списка в диалоговом окне ***Средний балл*** (рис.5.1). Список заполните из данных, введенных из диапазона на рабочем листе с помощью свойства ***RowSource*** объекта ***ListBox.*** Фамилии студентов пусть будут расположены в столбце ***A***, а их оценки в столбце ***B*** диапазона с данными о студентах. Создайте в программе обработчик ошибок, который будет проверять, являются ли данные из второго столбца списка с оценками студентов числами. Если хотя бы одно из этих данных не является числовым, программа должна проинформировать об этом пользователя с просьбой исправить эту некорректность.

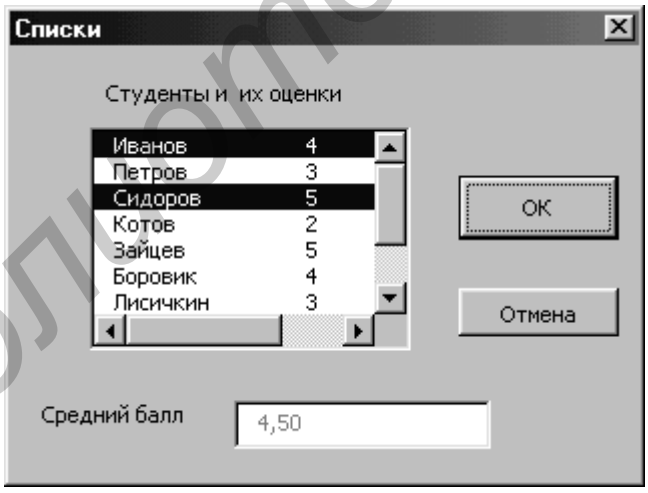


Рис. 5.1. Диалоговое окно ***Средний балл***

Рекомендации по выполнению задания

В качестве примера рассмотрим текст программы, в которой выполняются действия с элементами списка, приведенными на рис. 5.2.

```
Private Sub CommandButton1_Click()
Dim i As Integer
Dim n As Integer
Dim Сумма As Double
```

```
Dim Произведение As Double
Dim Среднее As Double
Dim Результат As Double
If OptionButton1.Value = True Then
Сумма = 0
With ListBox1
For i = 0 To .ListCount - 1
   If .Selected(i) = True Then
   Сумма = Сумма + .List(i)
    End If
 Next i
 End With
 Результат = Сумма
 End If
If OptionButton2.Value = True Then
   Произведение = 1
  With ListBox1
For i = 0 To .ListCount - 1
   If .Selected(i) = True Then
   Произведение = Произведение * .List(i)
End If
Next i
End With
Результат = Произведение
End If
If OptionButton3.Value = True Then
 Среднее = 0
 n = 0
  With ListBox1
For i = 0 To .ListCount - 1
   If .Selected(i) = True Then
      n = n + 1
      Среднее = Среднее + .List(i)
   End If
 Next i
End With
Результат = Среднее / n
End If
TextBox1.Text = CStr(Format(Результат,
"Fixed"))
End Sub
Private Sub CommandButton2_Click()
UserForm1.Hide
End Sub
Private Sub UserForm_Initialize()
With ListBox1
   .List = Array(1, 3, 4, 5, 6, 7, 8, 10)
   .ListIndex = 0
   .MultiSelect = fmMultiSelectMulti
 End With
With OptionButton1
   .Value = True
  .ControlTipText = "Сумма выбранных эле-
ментов"
 End With
OptionButton2.ControlTipText = _
"Произведение выбранных элементов"
OptionButton3.ControlTipText = _
 "Среднее значение выбранных элементов"
TextBox1.Enabled = False
With CommandButton1
  .Default = True
   .ControlTipText = "Нахождение результата"
 End With
CommandButton2.Cancel = True
UserForm1.Caption = "Списки"
  UserForm1.Show
End Sub
```

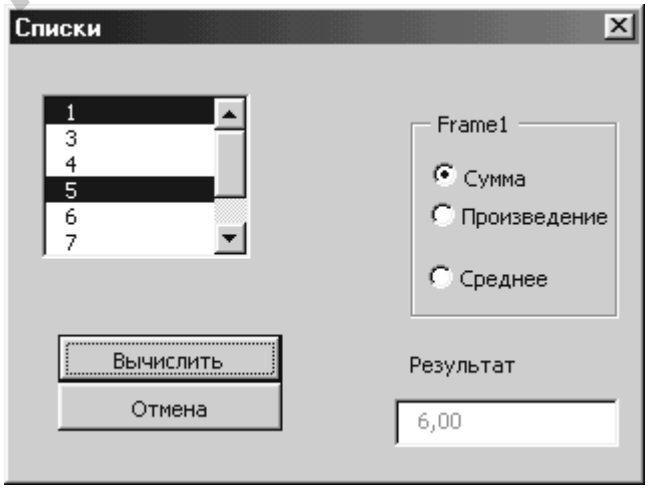


Рис. 5.2. Диалоговое окно ***Списки***

Задание 2

Создать программу, позволяющую реализовывать действия мячика, обозначенные на кнопках формы (рис. 5.3). Для этого используйте следующие методы и свойства элементов управления**:** ***Move, Visible, Height, Width, Left, Top, Picture, BorderStyle, PictureSizeMode.***

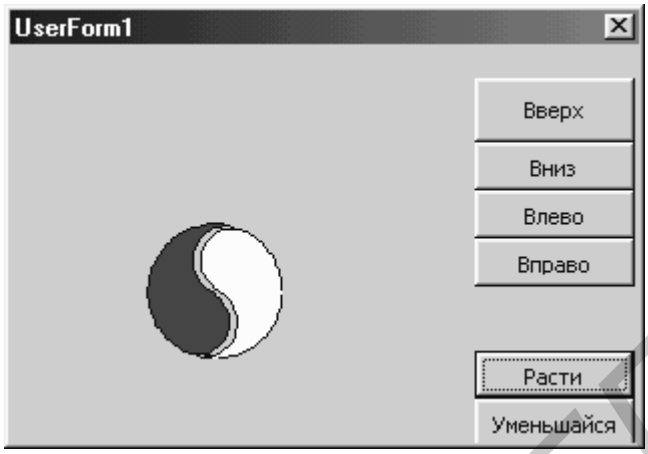


Рис. 5.3. Пользовательская форма

Рекомендации по выполнению задания

Если расположить указатель мыши на мячике и перемещать указатель мыши при нажатой левой кнопке по поверхности диалогового окна, то мячик должен передвигаться вслед за указателем мыши.

Используйте событие ***MouseMove***.

Событие	Событие происходит
MouseDoun	При нажатии кнопки мыши
MouseUp	При отпускании кнопки мыши
MouseMove	При перемещении указателя мыши

Порядок выполнения работы

1. Изучить теоретическую часть лабораторной работы.
2. Создать требуемые приложения.
3. Оформить отчет по выполненной работе.

ЛАБОРАТОРНАЯ РАБОТА №6

РАЗРАБОТКА ПРИЛОЖЕНИЙ С ЭЛЕМЕНТАМИ АВТОМАТИЗАЦИИ

Цель работы:

- научиться работать с объектами, их свойствами и методами;
- научиться автоматизировать заданные действия, используя методы и свойства элементов управления.

Задание

Требуется разработать приложение, автоматизирующее составление графика дежурств на неделю.

Пусть имеется трое работников (дежурных): Иванов, Петров, Сидоров. Дежурство ведется в две смены по 12 часов.

Рекомендации по выполнению задания

В редакторе форм создать диалоговое окно ***Магазин*** (рис.6.1).

В графическом редакторе, например Paint, создать файл ball.bmp с изображением красного шара (маркера), которым будет помечаться выбранный дежурный.

График дежурств составляется следующим образом:

1. Указатель перемещается на надпись из группы ***Компаньоны*** и выбирается компаньон, который будет дежурить. Программа информирует пользователя о выборе кандидата в дежурные отображением красного маркера рядом с его фамилией.

2. Для большей наглядности надписи, соответствующие первым сменам, изображены на белом фоне, а надписи, соответствующие вторым сменам, – на желтом фоне.

3. Ввод фамилии дежурного в смену выполняется путем перемещения указателя на надпись выбранной смены. Программа автоматически вставляет фамилию дежурного в надпись смены (рис. 6.2).

4. Удаление дежурного из смены выполняется указанием на пустую надпись в группе ***Компаньоны*** а затем выбором надписи смены, из которой требуется удалить фамилию дежурного.

5. Подсчет числа рабочих смен каждого из дежурных осуществляется нажатием на кнопку ***ОК***.

Порядок выполнения работы

1. Изучить теоретическую часть лабораторной работы.
2. Создать требуемое приложение.
3. Оформить отчет по выполненной работе.

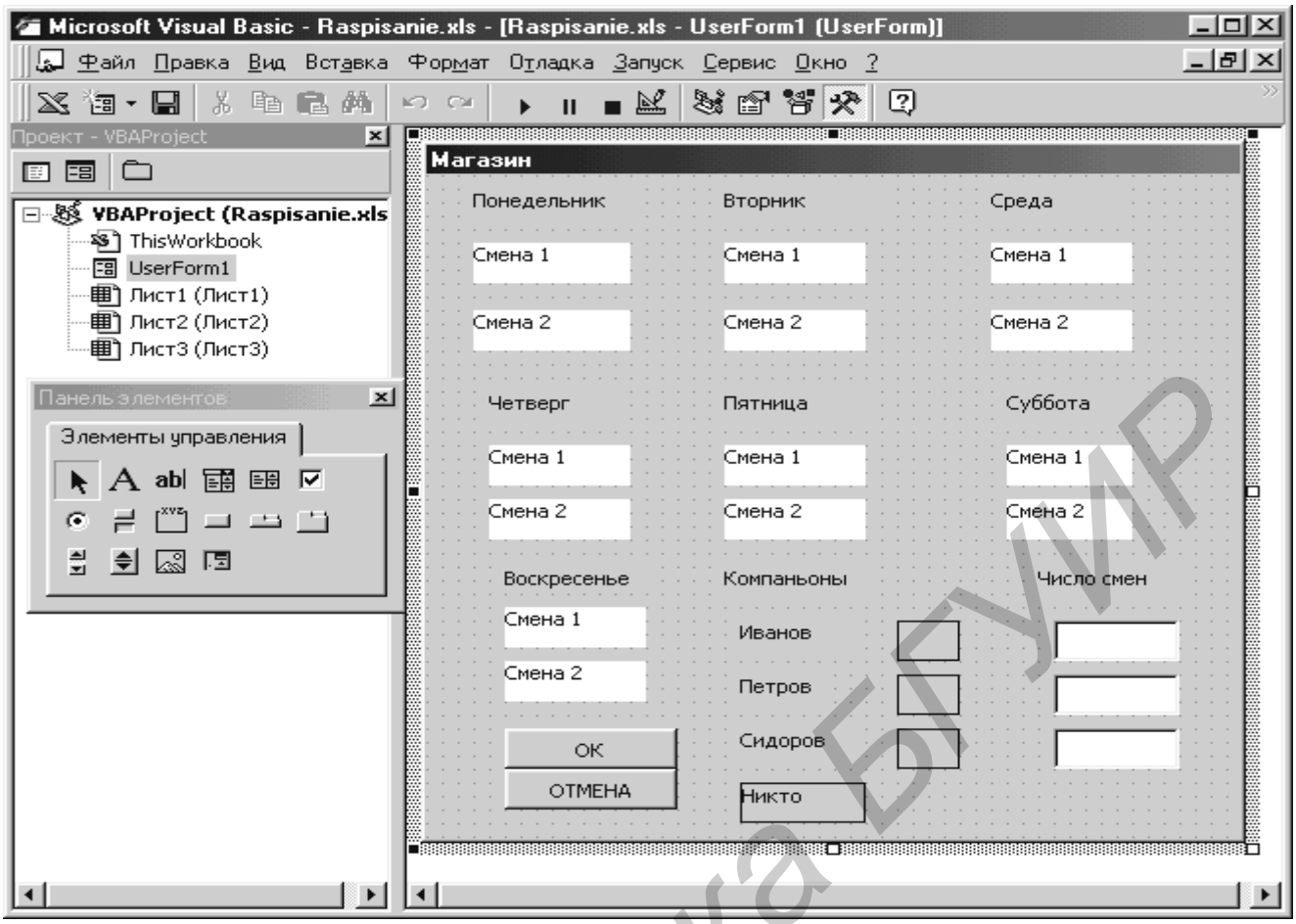


Рис. 6.1. Диалоговое окно ***Магазин***

Рис.6.2. Диалоговое окно ***Магазин***

ЛАБОРАТОРНАЯ РАБОТА №7

РАБОТА С ФАЙЛАМИ ПОСЛЕДОВАТЕЛЬНОГО И ПРОИЗВОЛЬНОГО ДОСТУПА

Цель работы:

- научиться работать с файлами последовательного доступа;
- научиться создавать, читать, писать файлы последовательного доступа;
- научиться работать с файлами произвольного доступа;
- научиться создавать, читать, писать файлы произвольного доступа.

Задание 1

Создать приложение, являющееся примером простейшего текстового редактора, в который можно загрузить из файла уже существующий текст, отредактировать его и записать измененный текст в файл.

Рекомендации по выполнению задания

1. Создать файл в текстовом редакторе **Блокнот**.
2. В редакторе форм создать диалоговое окно ***Файл последовательного доступа*** (рис. 7.1).
3. При разработке программы используйте инструкции ***Open, Print, Close*** и функции ***Input, LOF.*** Имя открываемого файла вводите в переменную ***Имя_Файла.***
4. Добавить в файл последовательного доступа строку: "Мне нравится изучать VBA".
5. Создать в редакторе ***Microsoft Word*** файл с расширением .doc, записать его в поле диалогового окна, проанализировать результат.
6. Создать две процедуры, использующие инструкции **Print** и **Write** для создания файлов последовательного доступа.

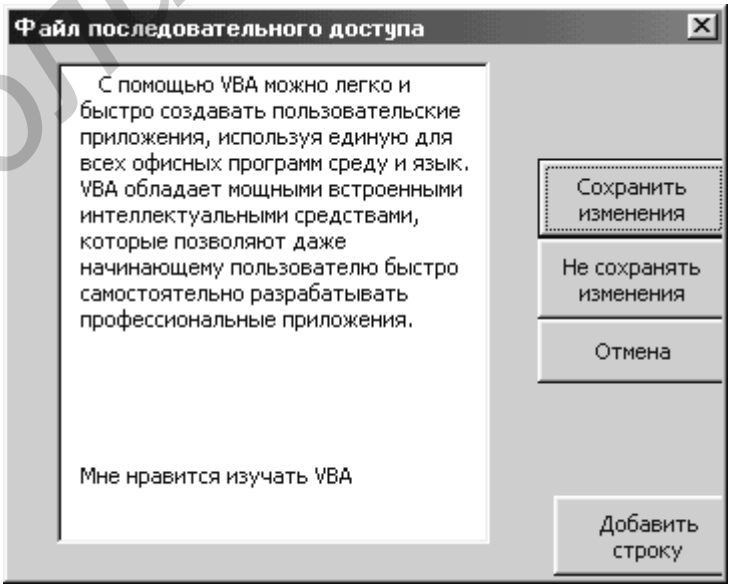


Рис. 7.1. Диалоговое окно ***Файл последовательного доступа***

Задание 2

Разработать приложение, реализующее создание базы данных с помощью файла произвольного доступа. Файл предназначен для хранения информации о студентах: фамилии, имени, группе, адресе. Диалоговое окно ***Информация о студентах*** (рис. 7.2) должно позволять перемещаться от записи к записи, редактировать существующие и создавать новые записи. Переход от записи к записи выполнять при помощи счетчика.

Рекомендации по выполнению задания

1. На листе стандартного модуля наберите описание пользовательского типа:

```
Option Explicit
Public Type Студент
Фамилия As String * 15
Имя As String *   String * 30
End Type
```

2. На листе модуля UserForm1 набрать программу, используя следующие переменные уровня модуля:

```
Dim Студент As Студент
Dim ДлинаЗаписи As Long
Dim ИмяФайла As String
Dim ТекущаяЗапись As Long
Dim ПоследняяЗапись As Long
```

3. Создать две процедуры: ***Показать Запись*** и ***Записать Запись***. Процедура ***Показать Запись*** выводит в поля диалогового окна запись с номером, указанным в переменной ***ТекущаяЗапись***. При ее написании используйте функцию обработки строк ***Trim***. Процедура ***Записать Запись*** записывает в файл из полей диалогового окна запись с номером, указанным в переменной ***ТекущаяЗапись***. Используйте команды ввода/вывода для файлов произвольного доступа ***Get*** и ***Put***.

4. По нажатии кнопки ***Создать файл*** информация из текстовых полей ***Фамилия, Имя, Группа, Адрес*** записывается в файл, т.е. создается файл произвольного доступа. Файл открывать в режиме ***Random.***

```
ДлинаЗаписи=Len(Студент)
Open …
```

5. Кнопка ***Закрыть файл*** закрывает файл по окончании его создания.

6. При нажатии кнопки ***Открыть существующий файл*** происходит открытие файла и считывание первой записи в диалоговое окно при помощи процедуры ***Показать Запись***. В надписи с общим числом записей выводится общее число записей открытого файла.

ДлинаЗаписи=Len(Студент)
Open …
ТекущаяЗапись=1
ПоследняяЗапись=FileLen(ИмяФайла) / ДлинаЗаписи

7. Кнопка ***Новая запись*** создает новую запись в конце файла.

8. Кнопка ***Записать изменения*** считывает из диалогового окна в текущую запись файла информацию при помощи процедуры ***Записать Запись.***

9. ***Счетчик*** устанавливает границы изменения счетчика. Выводит текущее значение счетчика в поле, считывает текущую запись из файла в диалоговое окно при помощи процедуры ***Показать Запись***.

10. Кнопка ***Выход*** закрывает пользовательскую форму.

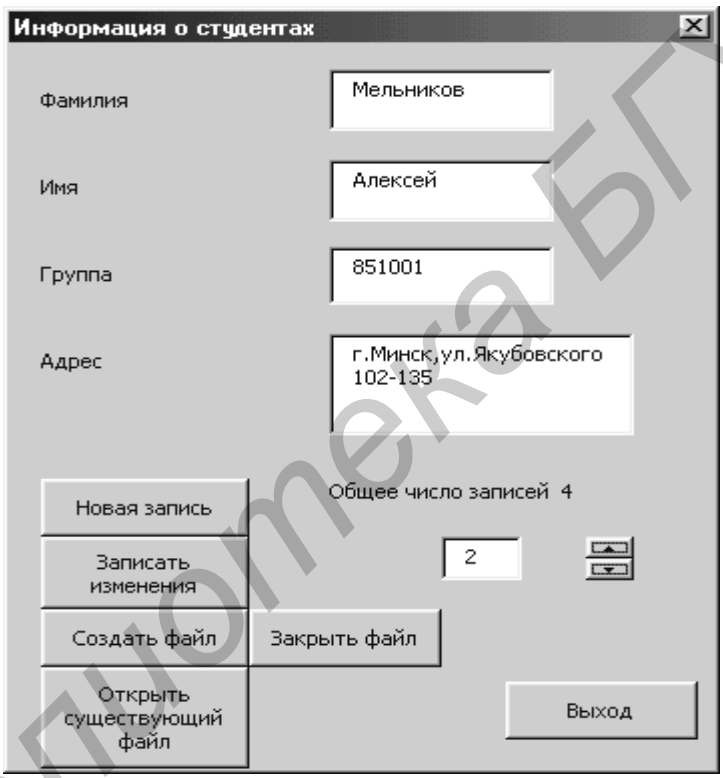


Рис. 7.2. Диалоговое окно ***Информация о студентах***

Порядок выполнения работы

1. Изучить теоретическую часть лабораторной работы.
2. Создать требуемое приложение.
3. Оформить отчет по выполненной работе.

ЛАБОРАТОРНАЯ РАБОТА №8

СОЗДАНИЕ БАЗЫ ДАННЫХ

Цель работы:

- разработать приложение для заполнения базы данных.

Задание

Требуется создать пользовательское приложение, позволяющее при помощи диалогового окна заполнять базу данных.

Рекомендации по выполнению задания

1. Предлагается занести в базу данных информацию о туристах. Сведения о туристах заносить в диалоговое окно ***Регистрация туристов*** (рис. 8.1).

2. По нажатии кнопки ***ОК*** данные вводятся в базу данных (рис. 8.2).

3. Для определения первой пустой строки в заполняемой базе данных о туристах следует использовать инструкцию

НомерСтроки=Application.CountA(ActiveSheet.Columns(1))+1,

правая часть которой вычисляет количество непустых ячеек в первом столбце активного рабочего листа. Переменной ***НомерСтроки*** присваивается номер первой непустой строки базы данных.

4. При написании программы используйте объекты ***Range*** и ***Cells*****,** свойство ***Value.***

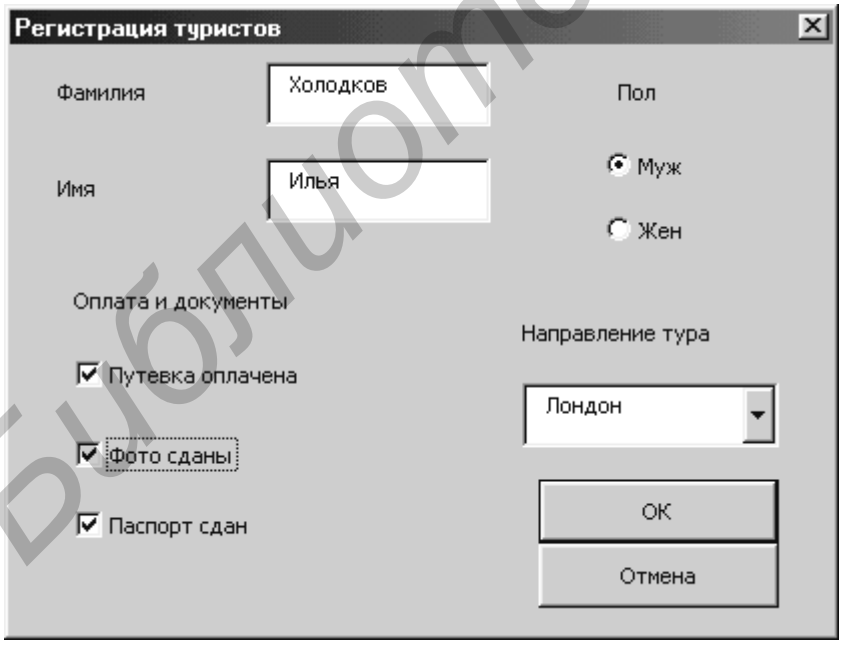
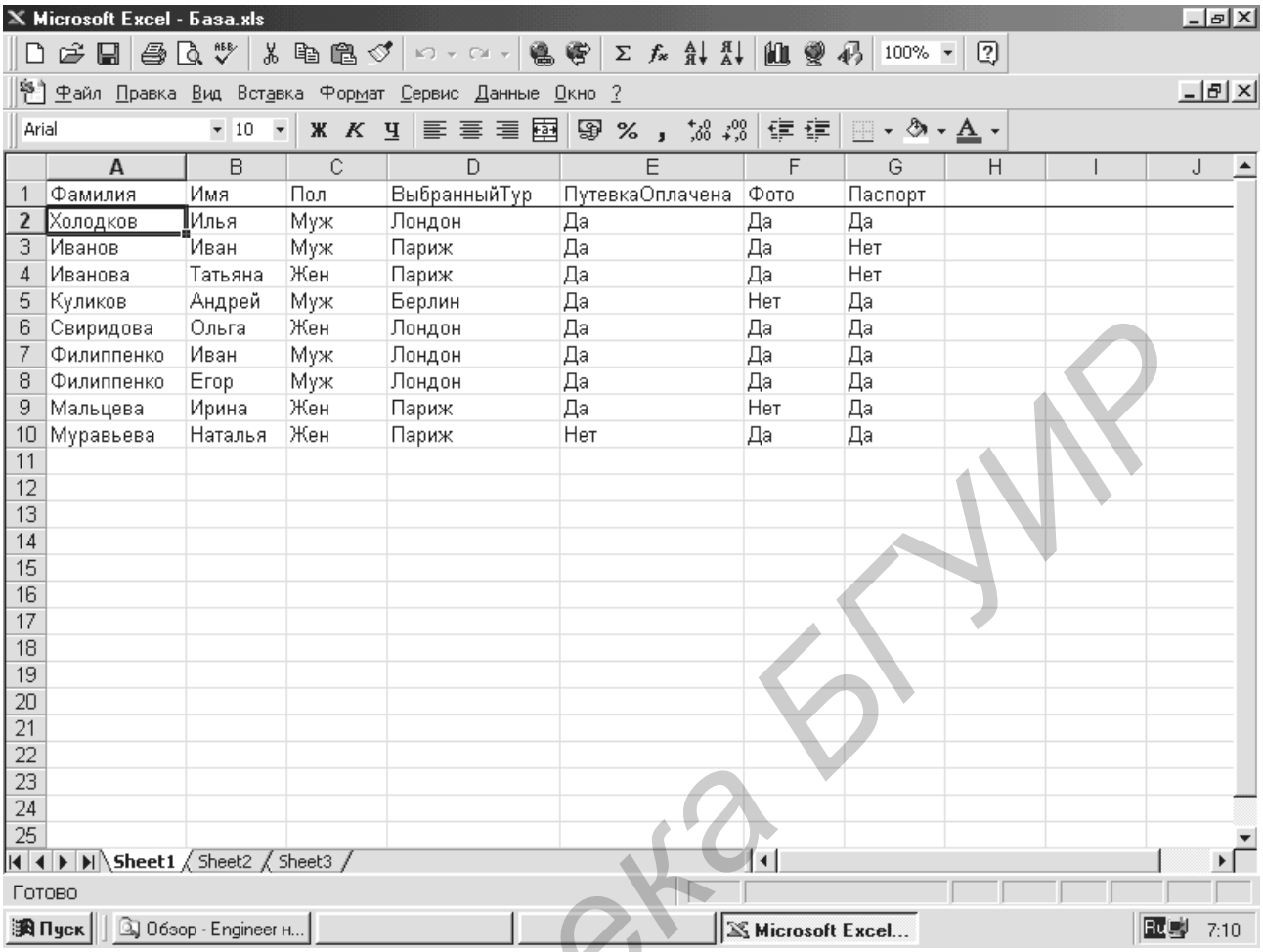


Рис. 8.1. Диалоговое окно ***Регистрация туристов***



	A	B	C	D	E	F	G
1	Фамилия	Имя	Пол	ВыбранныйТур	ПутевкаОплачена	Фото	Паспорт
2	Холодков	Илья	Муж	Лондон	Да	Да	Да
3	Иванов	Иван	Муж	Париж	Да	Да	Нет
4	Иванова	Татьяна	Жен	Париж	Да	Да	Нет
5	Куликов	Андрей	Муж	Берлин	Да	Нет	Да
6	Свиридова	Ольга	Жен	Лондон	Да	Да	Да
7	Филиппенко	Иван	Муж	Лондон	Да	Да	Да
8	Филиппенко	Егор	Муж	Лондон	Да	Да	Да
9	Мальцева	Ирина	Жен	Париж	Да	Нет	Да
10	Муравьева	Наталья	Жен	Париж	Нет	Да	Да

Рис.8.2. База данных туристов

Порядок выполнения работы

1. Изучить теоретическую часть лабораторной работы.
2. Создать требуемое приложение.
3. Оформить отчет по выполненной работе.

ЛИТЕРАТУРА

1. Орвис В. Дж.Visual Basic for Applications на примерах: Пер. с англ. – М.: БИНОМ, 1995. – 512 с.

2. Дженнингс Р. Microsoft Access 97 в подлиннике. В 2 кн.: Пер. с англ. – СПб.: 2000. – 842 с.

3. Гарнаев А.Ю. СамоучительVBA. – СПб.: БХВ – Санкт-Петербург, 2000. – 512 с.

Св.план 2004, поз. 54

Учебное издание

Бочкарёва Лия Валентиновна,
Шостак Елена Викторовна

ПРИКЛАДНЫЕ И ИНТЕГРИРОВАННЫЕ ПАКЕТЫ

Учебно-методическое пособие
для студентов специальности
«Программное обеспечение информационных технологий»
дневной формы обучения

Редактор Е.Н.Батурчик

Подписано в печать 21.04.2004.	Формат 60x84 1/16.	Бумага офсетная.
Гарнитура «Таймс».	Печать ризографическая.	Усл. печ.л.
Уч.-изд. л.1,8.	Тираж 100 экз.	Заказ 621.

Издатель и полиграфическое исполнение
Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
Лицензия ЛП № 156 от 30.12.2002
Лицензия ЛВ № 509 от 03.08.2001
220013, Минск, П.Бровки,6