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

Н.Д. Адаменко

Основы программирования на VBA

Методические рекомендации по выполнению лабораторных работ

Витебск
УО «ВГУ им. П.М. Машерова»
2009

УДК 004.42(075)
ББК 32.973.26-018я73
А28

Печатается по решению научно-методического совета учреждения образования «Витебский государственный университет имени П.М. Машерова». Протокол № 6 от 24.06.2009 г.

Автор: доцент кафедры информатики и ИТ УО «ВГУ им П.М. Машерова», кандидат педагогических наук **Н.Д. Адаменко**

Р е ц е н з е н т:
заведующий кафедрой информатики УО «ВГТУ», кандидат технических наук *В.Е. Казаков*

Адаменко, Н.Д.

А28 Основы программирования на VBA : методические рекомендации по выполнению лабораторных работ / Н.Д. Адаменко. – Витебск : УО «ВГУ им. П.М Машерова», 2009. – 50 с.

Учебное издание направлено на формирование умений разрабатывать профессиональные офисные приложения на основе применения языка программирования VBA. Рассмотрены вопросы создания пользовательского интерфейса, а также автоматизации работы в среде приложений MS Excel и MS Access.

УДК 004.42(075)
ББК 32.973.26-018я73

С О Д Е Р Ж А Н И Е

В В Е Д Е Н И Е

Данное учебное издание «Основы программирования на VBA» служит для поддержки цикла дисциплин, изучаемых студентами специальности «Прикладная математика».

Цель изучения дисциплины «Основы программирования на VBA» заключается в формировании умений создавать профессиональные приложения в офисных средах.

Учебное издание способствует освоению объектных моделей MS Excel и MS Access, в нем рассмотрены вопросы создания пользовательского интерфейса, а также автоматизации работы в среде этих приложений.

Включает семь лабораторных работ, выполнение которых позволяет освоить приемы создания офисных приложений на основе Excel, Access и VBA.

Материалы, представленные в учебном издании, прошли успешную экспериментальную проверку на математическом факультете УО «ВГУ имени П.М. Машерова».

ЛАБОРАТОРНАЯ РАБОТА № 1
Использование макрокоманд в MS Excel

Цель работы: освоение автоматизации разработки таблиц с использованием транслятора MacroRecorder.

ОСНОВНЫЕ СВЕДЕНИЯ

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

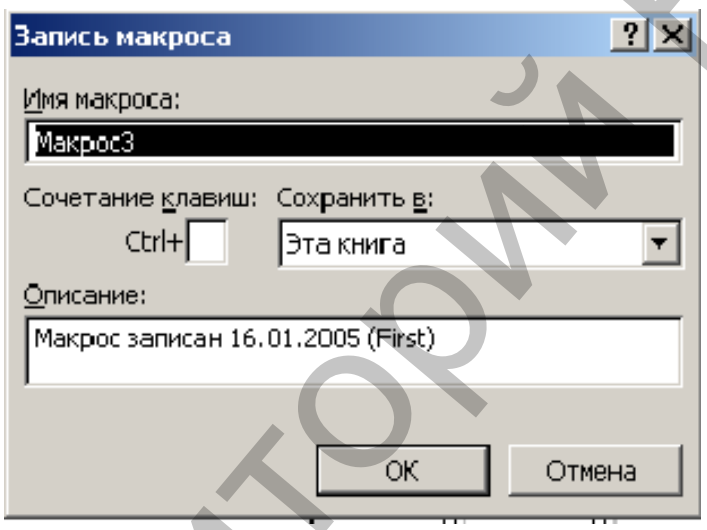

Рис. 1.

Для активизации MacroRecorder следует выбрать команду **Сервис/Макрос/Начать запись.** В результате выполнения команды на экран будет выведено диалоговое окно **Запись макроса** (см. **Ошибка! Источник ссылки не найден.**).

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

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

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

После щелчка на кнопке **ОК** начинается запись макроса. В окне Excel появится панель инструментов записи макроса, на которой рас-

положены кнопки **Остановить запись** и **Выполнить макрос.**

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

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

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

Для назначения макрокоманды кнопке следует выполнить следующие действия:

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

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

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

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

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

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как активизировать MacroRecorder?
2. Какие варианты хранения макросов существуют?
3. Какой способ хранения макроса позволяет сделать его доступным для других рабочих книг?
4. Какими способами макрос может быть запущен на выполнение?
5. Как назначить макросу кнопку на панели инструментов?
6. Как изменить вид кнопки, назначенной для макроса?
7. Как отредактировать макрос?

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Установить средний или низкий уровень безопасности (**Сервис/Параметры/Безопасность**) и создать макрос **Ведомость** для формирования таблицы, приведенной на рис. 2. В случае неявки студента в столбец оценок заносится значение «н/я». Столбцы F-J носят вспомогательный характер и служат для вычисления количества каждой из оценок. Поэтому после создания таблицы их следует скрыть **(Формат/Столбец/Скрыть).** Диапазону F6:F19 присвоить имя «отлично», диапазону G6:G19 – «хорошо» и т.д.).

2. Сохранить макрос в личной папке.

3. Макросу назначить сочетание клавиш **Ctrl+v.**

4. Назначить макросу значок из списка и затем его отредактировать **(Изменить значок для кнопки).**

5. Внести в макрос изменения (Выполнить вертикальное и горизонтальное выравнивание всех заголовков таблицы по центру, обвести жирной линией шапку таблицы (№ п.п, ФИО…), залить ее желтым цветом.

6. Запустить макрос разными способами, создав копии таблицы на втором и третьем рабочем листах для разных дисциплин.

7. На четвертом рабочем листе создать списки фамилий двух групп. Номера групп занести в ячейки **A1** и **C1**. Списки фамилий – в столбцы **A** и **C**, номера зачетных книжек – в столбцы **B** и **D** соответственно.

- Отредактировать текст макроса: отменить вывод строки состояния, защитить рабочий лист 4 паролем.

8. Заполнить таблицы: группу, дисциплину, и т.д. Предусмотреть автоматическое заполнение списка студентов и номеров зачеток в зависимости от номера группы по нажатию кнопки «заполнить список студентов» на панели инструментов.

	A	B	C	D	E	F	G	H	I	J
1	**Экзаменационная ведомость**									
2										
3	**Группа №**		**Дисциплина**							
4										
5	№ п.п	ФИО	№ за-четки	Оценка	Подпись экзаменатора	5	4	3	2	неяв-ки
6						=ЕСЛИ(D6=5;1;0)	…	…	…	…
7						=ЕСЛИ(D7=5;1;0)				
8						=ЕСЛИ(D8=5;1;0)				
9						=ЕСЛИ(D9=5;1;0)				
10						=ЕСЛИ(D10=5;1;0)				
11						=ЕСЛИ(D11=5;1;0)				
12						=ЕСЛИ(D12=5;1;0)				
13						=ЕСЛИ(D13=5;1;0)				
14						=ЕСЛИ(D14=5;1;0)				
15						=ЕСЛИ(D15=5;1;0)				
16						=ЕСЛИ(D16=5;1;0)				
17						=ЕСЛИ(D17=5;1;0)				
18						=ЕСЛИ(D18=5;1;0)				
19						=ЕСЛИ(D19=5;1;0)				
20										
21	Отлично	=СУММ(отлично)								
22	Хорошо	=СУММ(хорошо)								
23	Удовл.	=СУММ(удовлетворительно)								
24	Неудовл.	=СУММ(неудовлетворительно)								
25	Неявка	=СУММ(неявки)								
25	ИТОГО	=СУММ(C21:C25)								

Рис. 2.

ЛАБОРАТОРНАЯ РАБОТА № 2
Редактор VBA

Цель работы: изучение структуры редактора VBA; освоение приемов работы с окнами редактирования кода, создания и редактирования форм, просмотра объектов.

ОСНОВНЫЕ СВЕДЕНИЯ

Структура редактора

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

Интерфейс VBA состоит из следующих основных компонентов:

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

Окно проекта

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

В проекте автоматически создается модуль для каждого рабочего листа и для всей книги. Кроме того, модули создаются для каждой пользовательской формы, макросов и классов. По своему предназначению модули делятся на два типа: модули объектов и стандартные модули. Стандартные модули – это основной вид модулей. Большая часть всех процедур и функций, которые пишет программист, размещаются в стандартных модулях. Разумно иметь не один большой стандартный модуль, а некоторое множество относительно небольших модулей. В один модуль следует помещать набор функций, совместно вызывающих друг друга, связанных общей темой, общей функциональной направленностью.

В модулях, связанных с объектами, следует помещать только те процедуры и функции, которые непосредственно обрабатывают события. К модулям объектов относятся модули, связанные с рабочей книгой, рабочими листами, формами, и модули класса.

Структура модуля. Окно проекта и Окно кода

Модули VBA имеют очень простую синтаксическую структуру. В отличие от большинства языков программирования здесь нет большого числа разделов, нет разделения на интерфейс и реализацию. Каждый модуль вне зависимости от его типа имеет всего два раздела:

• Раздел объявлений переменных уровня модуля. Этот раздел идет первым и автоматически отделяется чертой от раздела методов. Всегда можно добавить новое объявление переменной в этот раздел. Любая переменная из раздела объявлений модуля является глобальной по отношению к методам модуля, она доступна в каждом из методов, каждый метод может изменить глобальную переменную.

• Раздел методов модуля. В этом разделе располагаются процедуры Sub и Function. С точки зрения синтаксиса ничего другого кроме процедур и функций в этом разделе быть не может. Каждый метод модуля общедоступен внутри модуля и может быть вызван другими методами модуля.

Область видимости компонент модуля расширяется на весь проект, если компонент объявлен со спецификатором **Public**. Каждый компонент модуля – переменная или метод – может быть снабжен спецификатором области видимости, который имеет два возможных значения – **Public** и **Private**. Если задан спецификатор **Public**, то это означает, что компонент общедоступен в пределах всего проекта. Спецификатор **Private** делает компонент закрытым для других модулей проекта.

Если при объявлении переменных модуля спецификатор области видимости опущен и указано только ключевое слово **Dim**, то такие переменные считаются закрытыми, то есть действует спецификатор **Private**. Для методов спецификатор области видимости можно опускать. В этом случае действует следующее правило. Все методы стандартных модулей имеют по умолчанию спецификатор **Public** и являются доступными во всем проекте. Методы модулей – классов и модулей, связанных с объектами, по умолчанию являются закрытыми и имеют статус **Private**.

Формы создаются командой **Insert/UserForm**, модули класса – командой **Insert/Module**. По мере создания, добавления и удаления файлов из проекта эти изменения отображаются в окне проекта.

Удаление файла из окна проекта производится выбором значка файла с последующим выполнением команды **File/Remove...**

Окно для редактирования кода

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

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

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

том управления. Обратный переход от процедуры к объекту управления быстрее всего осуществить нажатием кнопки **View Object.**

При написании кода редактор сам предлагает пользователю список компонентов, логически завершающих вводимую пользователем инструкцию. Например, набирая код

Range("A1").,

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

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

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

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

Для автоматического отображение на экране сведений о процедурах, функциях, свойствах и методах после ввода их имени следует выполнить команду **Tools/Options** активизировать вкладку **Editor** и установить флажок **Auto Quick Info.**

Эту же всплывающую подсказку можно также выводить на экран нажатием комбинации клавиш <**Ctrl>+<I**>.

Редактор кода также производит автоматическую проверку синтаксиса набранной строки кода сразу после нажатия клавиши <**Enter**>. Если после набора строки и нажатия клавиши <**Enter**> строка выделяется красным цветом, это указывает на наличие синтаксической ошибки в набранной строке. Кроме того, если установлен флажок **Auto Sintax Check** вкладки **Editor,** помимо выделения красным цветом фрагмента кода с синтаксической ошибкой**,** на экране отображается диалоговое окно, поясняющее, какая ошибка произошла.

Если расположить курсор на ключевом слове языка VBA, имени процедуры, функции, свойства или метода и нажать клавишу <**F1**>, то на экране появится окно со справочной информацией об этой функции. В справке, как правило, имеется пример использования кода.

Окно редактирования форм (UserForm)

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

Используя панель инструментов **Панель элементов** из незаполненной формы, можно сконструировать любое требуемое для прило-

жения диалоговое окно. Для облегчения размещения и выравнивания элементов управления используется сетка. Активизировать ее можно с помощью вкладки **General** диалогового окна, вызываемого командой **Tools/Options**, там же устанавливается шаг сетки. Кроме того, команды меню **Format** автоматизируют и облегчают процесс выравнивания элементов управления по их взаимному местоположению, так и по размерам.

Окно Просмотр объектов (Object Browser)

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

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

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

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

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

Отладка программ

Пошаговое выполнение программ

Для выполнения программы в пошаговом режиме используются четыре команды:

1. Команда **Отладка, Шаг с заходом (Debug/Step Into**), осуществляет последовательную шаг за шагом отладку всей программы, включая процедуры, вызываемые в программе.

2. Команда **Отладка, Шаг с обходом (Debug/Step Over**) осуществляет последовательную шаг за шагом отладку всей программы, исключая процедуры, т.е. если встречается процедура, то она выполняется целиком, а не пошагово.

3. Команда **Отладка, Шаг с выходом (Debug/Step Out**) завершает выполнение текущей процедуры и останавливает процесс пошаговой отладки на следующей после вызвавшей ее инструкции программы.

4. Команда **Отладка, Выполнить до текущей позиции (Debug/Run to Cursor)** выполняет программу до инструкции, на которой установлен курсор.

Точка останова

Точка останова устанавливается или снимается с помощью команды **Debug/Toggle Breakpoint**. На листе модуля точки останова выделяются полосой кирпичного цвета и кругом того же цвета.

В одном проекте может быть несколько точек останова. Точки останова предназначены для приостановки выполнения программы. Все инструкции, расположенные выше, между и ниже точек останова, выполняются в обычном режиме.

Одновременно снять все точки останова можно, выполнив команду **Debug/Clear All Breakpoints.**

Вывод значений свойств и переменных

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

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

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

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

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

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

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

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

?x

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

x=10

КОНТРОЛЬНЫЕ ВОПРОСЫ

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

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

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

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

1. Присвойте листу рабочей книги имя **Расчет ссуды** и создайте на нем следующую таблицу:

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

Рис. 3.

2. Создайте диалоговое окно **Расчет ссуды** (см. рис. 4).
3. Создайте процедуры, которые обеспечивают:

a) проверку всех данных, вводимых в диалоговое окно (правильность ввода дат, числовых данных, наличие в полях всех необходимых данных для расчета);

b) вычисление возвращаемой суммы по формуле: ***Возвращаемая сумма = Размер ссуды*(1+ставка)^((Дата возврата–Дата выдачи) /365);***

c) передачу данных из диалогового окна в таблицу Excel;

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

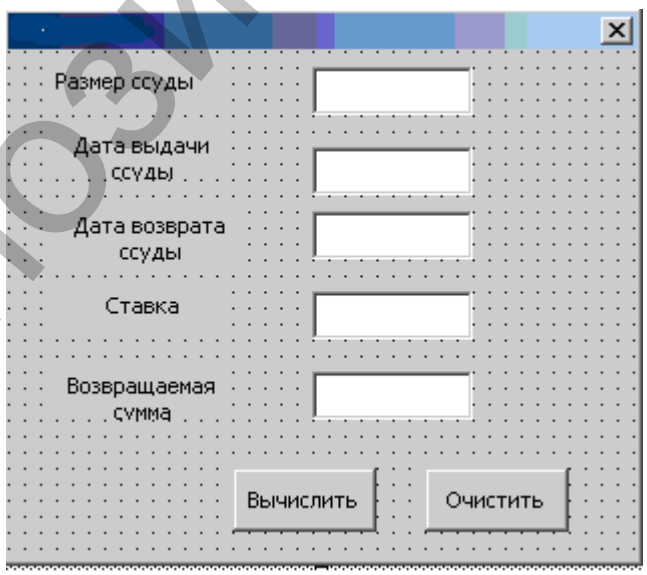

Рис. 4.

4. Создайте на панели инструментов кнопку «Рассчитать ссуду», которая будет выводить на экран диалоговое окно для расчета ссуды

по нажатию кнопки. Для создания процедуры обработки события в контекстном меню кнопки выбрать **Исходный текст.** В окне редактора создать процедуру, открывающую диалоговое окно: ***UserForm1. Show.*** Протестировать работу приложения.

5. Переключите окно редактирование кода в режим просмотра отдельной процедуры. Вернитесь в режим просмотра всего модуля.
6. Перейдите в режим, не предусматривающий вывод списка компонентов для логического завершения вводимых инструкций. Вернитесь в режим отображения этих компонентов.
7. Установите режим всплывающей подсказки с текущим значением переменной.
8. Выведите окно для отображения всех переменных процедуры.
9. В окне Immediate **Window** выполните вычисление:

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

ЛАБОРАТОРНАЯ РАБОТА № 3
Элементы управления «список» и «поле со списком»

Цель работы: освоение способов создания элементов управления пользовательских форм «список» и «поле со списком».

ОСНОВНЫЕ СВЕДЕНИЯ

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

Свойства элемента управления ListBox

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

List	Возвращает элемент списка, стоящий на пересечении указанных строки и столбца. Синтаксис: ***List(Row, Column***)
RowSource	Устанавливает диапазон, содержащий элементы списка
ControlSource	Устанавливает диапазон (ячейку), куда возвращается выбранный элемент из списка
MultiSelect	Устанавливает способ выбора элементов списка. Допустимые значения: • ***fmMultiSelectSingle*** (выбор только одного элемента); • ***fmMultiSelectMulti*** (разрешен выбор нескольких элементов посредством либо щелчка, либо нажатие клавиши <Пробел>); • ***fmMultiSelectExtended*** (разрешено использование клавиши <Shift> при выборе ряда последовательны элементов списка)
Selected	Допустимые значения: ***True*** (если элемент списка выбран) и ***False*** (в противном случае). Используется для определения выделенного текста, когда свойство ***MultiSelect***: имеет значение ***fmMultiSelectMulti*** или ***fmMultiSelectExtended***
Column Widths	Устанавливает ширину столбцов списка. Синтаксис: ***ColumnWidths = String*** ***String*** – строка, устанавливающая ширину столбцов. В следующем примере устанавливается ширина каждого из трех столбцов списка: ***With ListBox1*** ***.ColumnCount = 3*** ***.ColumnWidths= "20,30,30"*** End With
ColumnHeads	Допустимые значения: ***True*** (выводятся заголовки столбцов раскрывающегося списка) и ***False*** (в противном случае)
ListStyle	Допустимые значения: • ***fmListStylePlain*** (выбранный элемент из списка выделяется цветом) • ***fmListStyleOption (***перед каждым элементом в списке располагается флажок и выбор элемента из списка соответствует установке этого флажка)
MatchEntry	Выводит первый подходящий элемент из списка при наборе его имени на клавиатуре. Допустимые значения:

	• ***fmMatchEntryNone*** (режим вывода подходящего элемента в списке отключен); • ***fmMatchEntryFirstLetter*** **(**выводит подходящий **элемент** по набранной первой букве. В этом случае предпочтительно, чтобы элементы списка были упорядочены в алфавитном порядке); • ***fmMatchEntryComlete*** **(**выводит подходящий **элемент** по полному набранному имени)
BoundColumn	Устанавливает тип, возвращаемый свойством ***Value***. А именно, • если свойство ***BoundColumn*** равно 0, то свойство ***Value*** возвращает индекс выбранной строки, т.е. в этом случае оно действует как свойство ***ListIndex*;** • если свойство ***BoundColumn*** принимает значение из диапазона от 1 до количества столбцов в списке, то свойство ***Value*** возвращает элемент из выбранной строки, стоящий в столбце, определенном свойством ***BoundColumn***

Методы элемента управления ListBox

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

Заполнение списка

Заполнить список можно одним из следующих способов:

1. Поэлементно, если список состоит из одной колонки:

```
With ListBox1
.AddItem (“Июнь”)
.AddItem (“Июль”)
 .AddItem (“Август”)
 End With
```

2. Массивом, если список состоит из одной колонки:

```
With ListBox1
.List = Array("Июнь", "Июль" ,"Август")
.ListIndex =1
 End With
```

3. Из диапазона (в примере **A1:B4**), в который предварительно введены элементы списка. Результат выбора (индекс выбранной строки) выводится в ячейку (**C1**).

```
With ListBox1
.ColumnCount = 2
.RowSource = "A1:B4"
.ControlSource= "C1"
.BoundColumn = 0
End With
```

4. Поэлементно, если список состоит из нескольких колонок, например двух:

```
With ListBox1
.ColumnCount = 2
.AddItem "Июнь"
.List(0,1)= "Сессия"
.AddItem "Июль"
.List(1,1)= "Каникулы"
.AddItem "Август"
 .List(2,1)= "Каникулы"
End With
```

5. Массивом, если список состоит из нескольких колонок, например, двух:

```
ListBox1 .ColumnCount = 2
Dim A(2,1) As String
A(0,0) = "Июнь"
A(0,1) = ""Сессия
A(1,0) = "Июль"
A(1,1)="Каникулы"
A(2,0) = "Август"
A(2,1) ="Каникулы
With ListBox1
ListBox1 .List = A
End With
```

Выбор нескольких элементов из списка

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

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

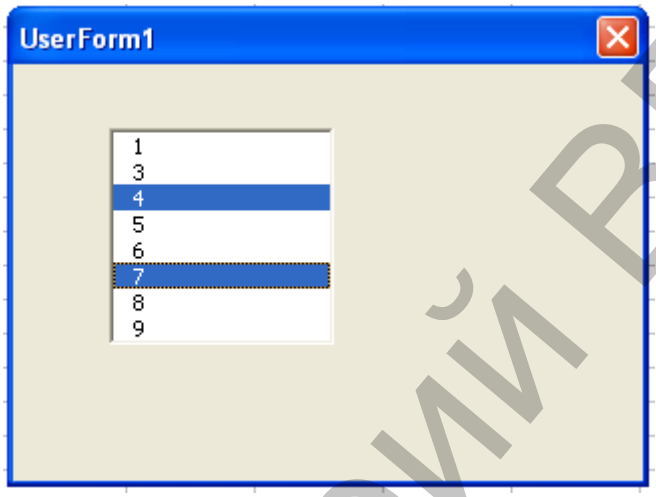

Рис. 5.

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

Поле со списком

Элемент управления **ComboBox** сочетает в себе функциональные возможности списка **ListBox** и поля **TextBox**. В отличие от поля **ListBox** в элементе управления **ComboBox** отображается только один элемент списка. Кроме того, у него отсутствует режим выделения нескольких элементов. Также как в поле **TextBox,** в поле **ComboBox** можно вводить значение. Свойства объекта **ComboBox**, такие как

ListIndex, **ListCount**, **Enabled**, **List**, методы **Clear, RemoveItem** и **AfddItem** аналогичны соответствующим свойствам и методам списка **ListBox**. Элемент управления **ComboBox** имеет ряд уникальных свойств. Например,

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

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

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

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

Согласованная работа двух списков

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

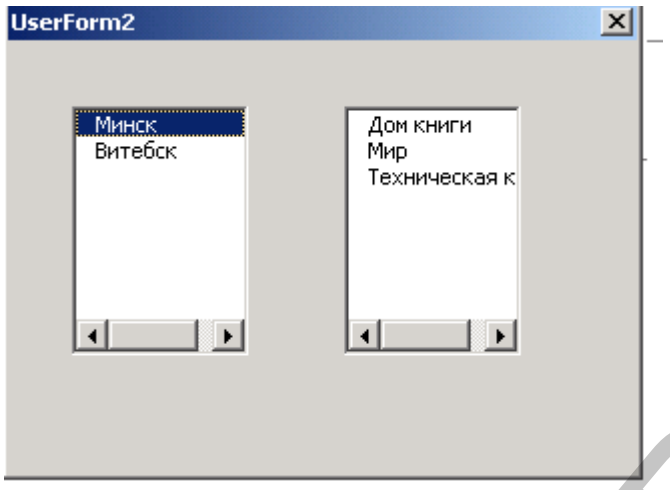

Рис. 6.

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

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

```
Private Sub UserForm2_Init()
Dim Город As Variant
Город= Array("Минск", "Витебск")
UserForm2.ListBox1.List = Город
UserForm2.ListBox1.ListIndex = 0
UserForm2.Show
End Sub
Private Sub ListBox1_Click()
Dim Магазин (2) As Variant
Магазин(0) = Array("Дом книги", "Мир", "Техническая книга")
Магазин(1) = Array("Глобус", "Светач", "Книги")
UserForm2.ListBox2.Clear
ListBox2.List = Магазин(ListBox1.ListIndex)
End Sub
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие элементы управления форм служат для создания списков?
2. Какое свойство элемента управления **ListBox** возвращает номер текущего элемента списка?
3. Какое свойство элемента управления **ListBox** устанавливает число столбцов в списке?
4. Какими способами список может быть заполнен элементами?

5. Какое свойство элемента управления **ListBox** устанавливает возможность выбора одного или нескольких элементов из списка?
6. Какое свойство элемента управления **ListBox** возвращает номер текущего элемента списка?
7. Какое свойство элемента управления **ListBox** позволяет вывести первый подходящий элемент из списка при наборе его имени на клавиатуре?
8. Какой метод элемента управления **ListBox** служит для добавления элемента в список?
9. В чем заключаются отличия элементов управления **ListBox** и **ComboBox**?
10. Как отключить функцию поля ввода в поле со списком?

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Задание 1.

Создать приложение, которое позволит выбрать несколько чисел, выводимых в списке в диалоговом окне **Действия над элементами списка** (Ошибка! Источник ссылки не найден. **7**).

В группе **Операции** следует установить один из переключателей: **Сумма** или **Среднее**, чтобы указать, какая операция будет выполняться над выбранными числами. Нажатие кнопки **Вычислить** должно привести к выполнению операции и выводу результата в поле **Результат**.

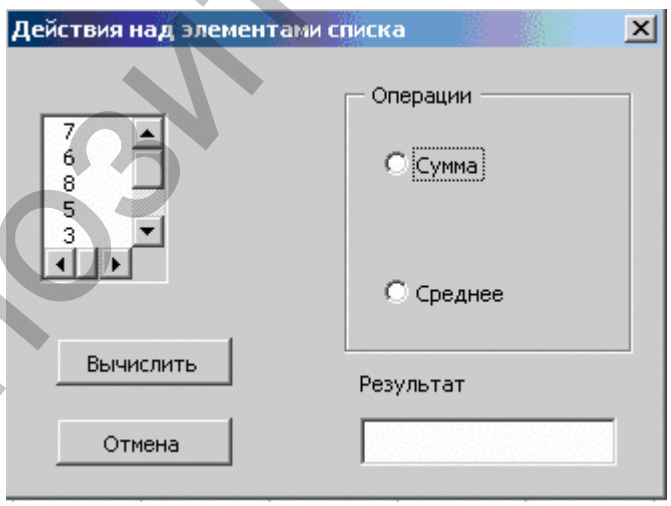

Рис. 7.

Для разработки приложения необходимо:

1. Создать диалоговое окно **Действия над элементами списка** (см. рис. 7).

2.Создать процедуру инициализации диалогового окна, предусматривающую:

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

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

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

Задание 2. Создать форму для распределения дисциплин между преподавателями кафедры. Список преподавателей приведен на рабочем листе **Преподаватели**; список дисциплин – на рабочем листе **Дисциплины**. Список распределения дисциплин вывести на рабочий лист **Распределение нагрузки**.

Заполнение списка на листе **Распределение нагрузки** выполнить с помощью диалогового окна **«Распределение дисциплин»** (см. **Ошибка! Источник ссылки не найден.** 8)

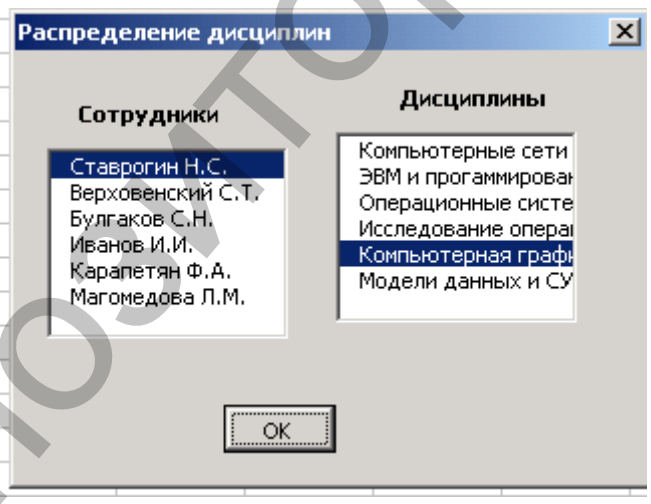

Рис. 8.

При нажатии кнопки **OK** выбранные из списков данные (сотрудник и дисциплина) вводятся в очередную строку списка, расположенного на листе **Распределение нагрузки**.

Для реализации поставленной задачи необходимо:

1. На листе **Преподаватели** в диапазон A1:A10 ввести список преподавателей.

2. На листе **Дисциплины** ввести список дисциплин.

3. На листе **Распределение нагрузки** в ячейку A1 ввести строку

«Преподаватели», а в ячейку B1 – «Дисциплины».

4. Создать форму (рис. 8).

5. Для открытия окна формы создать кнопку на панели инструментов **Стандартная.**

6. Разработать процедуру обработки нажатия кнопки **OK** формы. В процедуре предусмотреть:

- Проверку, были ли выбраны данные из списков (значение свойства ***ListIndex*** элемента управления ***ListBox1*** должно быть неотрицательным).
- Определение первой свободной строки в списке рабочего листа **Распределение нагрузки.**
- Активизацию рабочего листа **Распределение нагрузки** с **помощью** метода **Activate.**
- **Предусмотреть** возможность ошибочного выбора для преподавателя одной и той же дисциплины повторно.

Задание 3. Создать приложение для нахождения суммарной нагрузки преподавателя, выбранного из списка (диалоговое окно, приведенном на **Ошибка! Источник ссылки не найден.** 9).

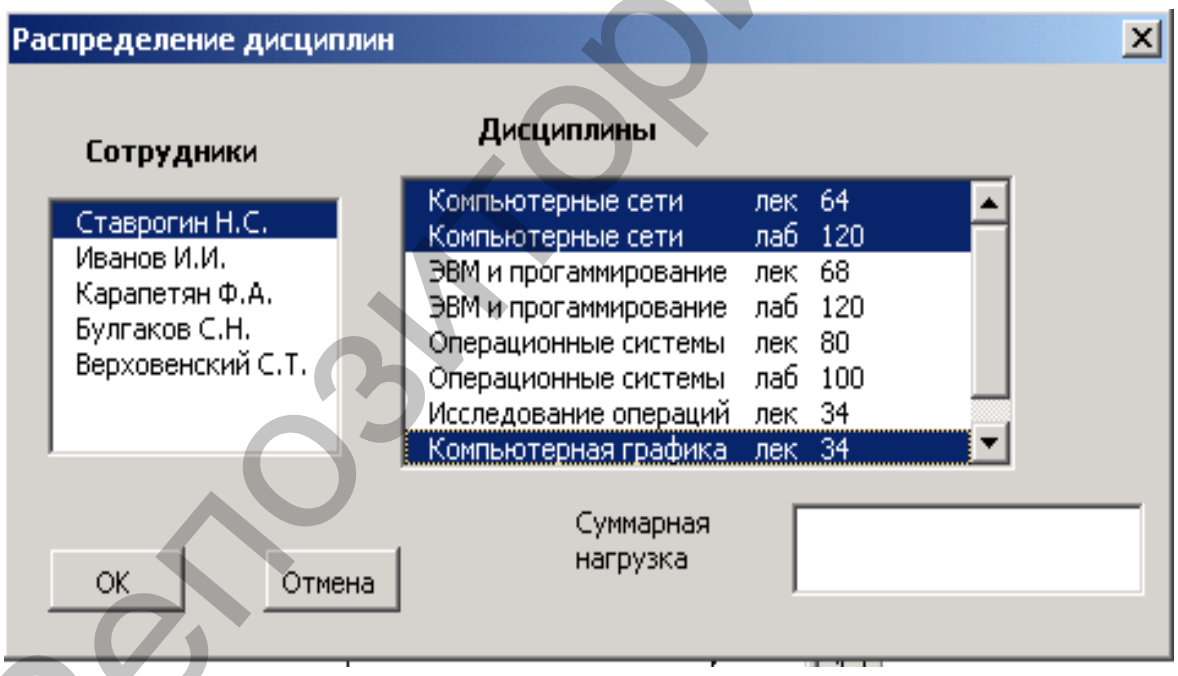

Рис. 9.

Для создания приложения необходимо:

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

Таблица 1

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

3. На панели **инструментов** создать кнопку **Расчет нагрузки** для открытия диалогового окна.

4. Создать диалоговое окно.

5. Разработать процедуру инициализации формы (заполнение списков). Списки должны заполняться из диапазонов соответствующих рабочих листов с помощью свойства ***RowSource*** элемента управления ***ListBox***. Поскольку списки могут иметь переменный размер, сначала необходимо выполнить идентификацию диапазона с помощью свойства **CurrentRegion,** а затем определить адрес диапазона с помощью свойства ***Address*:**

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

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

Обеспечить возможность выбора нескольких элементов из списка дисциплин.

6. Для отображения окна формы в процедуре используется метод ***Show (UserForm1.Show).***

7. Разработать процедуру, **запускаемую** по нажатию на кнопку **OK** формы**.** Процедура должна выполнять суммирование часов по выбранным для преподавателя дисциплинам. Найденная сумма должна выводиться в поле **Результат**.

Предусмотреть:

- запись нагрузки преподавателя (ФИО, дисциплина, вид занятий количество часов) в таблицу рабочего листа **Нагрузка.** Необходимо определить номер первой незаполненной строки и записать в ее ячейки данные из полей формы;

- отображение полученной суммарной нагрузки в столбце **B** на рабочем листе **Преподаватели;**
- назначение клавише <***Enter***> функции кнопки ***OK*** (значение свойства ***Default = True***);
- назначение клавише <***Esc***> функции кнопки ***Отмена*** (значение свойства ***Cancel*** = ***True***.);
- проверку, являются ли значения в столбце нагрузки числами;
- проверку выделения хотя бы одного элемента списка в каждом списке;
- проверку ошибочного повторного выбора пользователем одной и той же дисциплины и вида занятий для преподавателя.
- проверку ошибочного выбора для преподавателя дисциплины, все часы по которой уже были распределены.

Обеспечить очистку полученных результатов при нажатии кнопки **Отмена**.

Задание 4. Разработать форму с двумя согласованными списками: преподавателей и дисциплин, которые они ведут (см. рис. 10)**.**

Массивы, содержащие списки преподавателей и дисциплин, которые они ведут, сформировать из диапазонов на рабочем листе **Распределение нагрузки (**см. задание 2**).** Дополнить таблицу кодами преподавателей (см. столбец **C**).

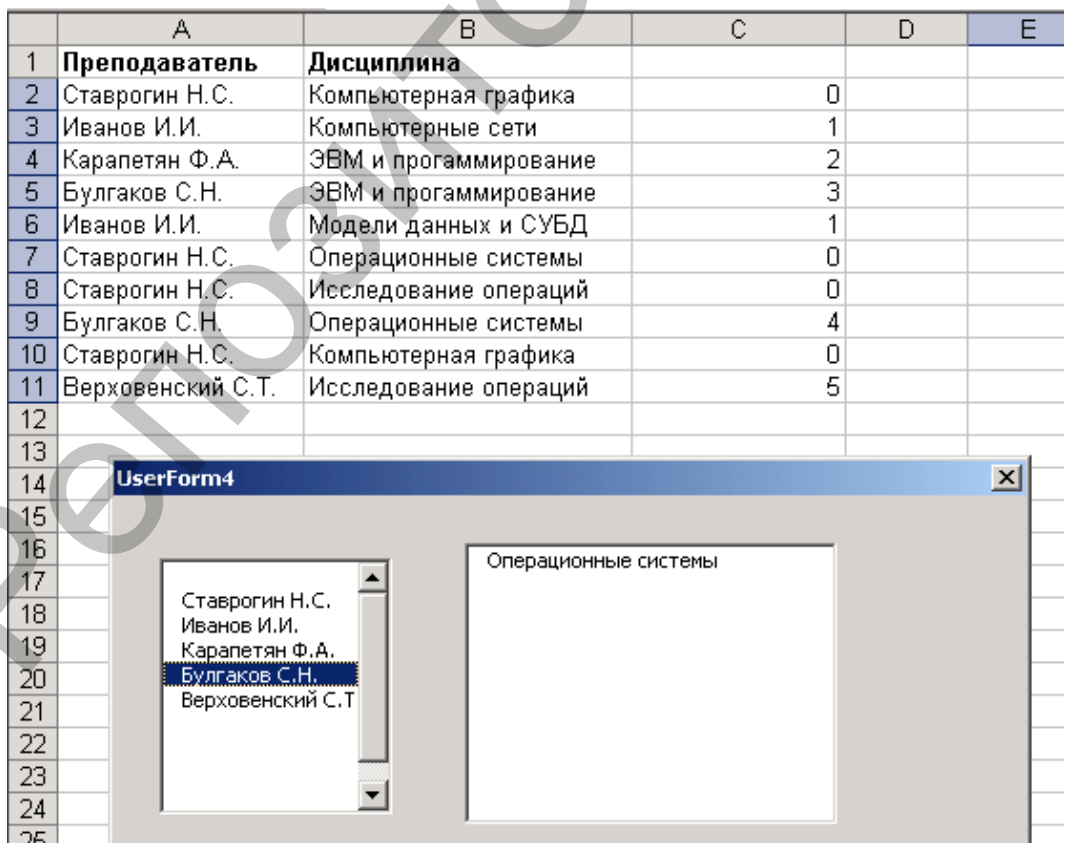

Рис. 10.

Задание 5. Создать приложение для составления графика отпусков сотрудников. В результате работы приложения должна создаваться таблица на рабочем листе **График отпусков** (рис.11**).**

	A	B	C	D
1	**График отпусков**			
2	*Голицын*	27.03.2005	03.05.2005	
3	*Воронцов*	01.02.2005	12.02.2005	
4	*Шлапаков*	27.07.2005	03.08.2005	
5	*Никонова*	28.07.2005	24.08.2005	
6	*Казанцева*	29.07.2005	25.08.2005	
7	*Командина*	30.07.2005	26.08.2005	
8				
9				
10				

Рис. 11.

Данные для создания таблицы вводятся с помощью диалогового окна, приведенного на рис. 12.

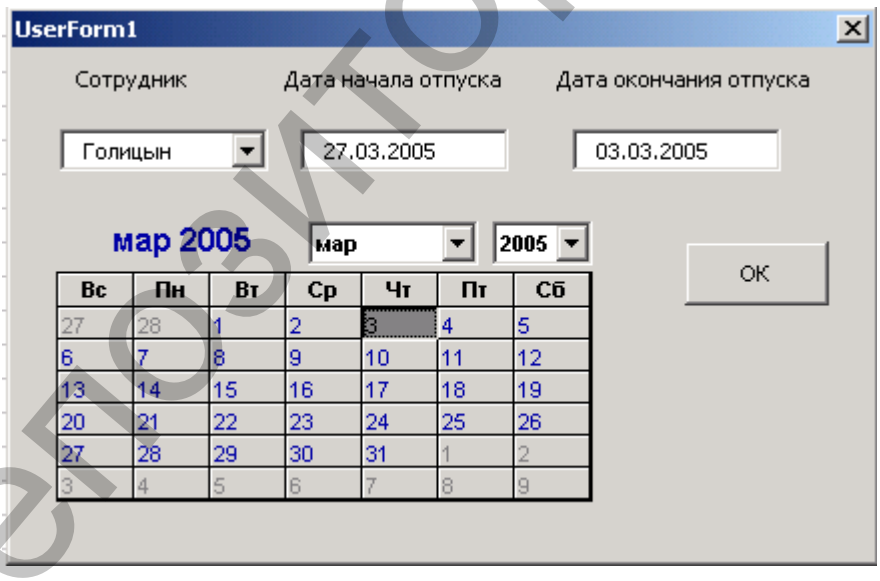

Рис. 12.

Даты начала и окончания отпуска задавать с помощью элемента управления **Календарь**. При инициализации формы создать список преподавателей (список из задания 2), установить текущую дату календаря (**Calendar.Value = Now**). Для определения поля ввода, в которое следует вводить дату (дату начала или дату окончания отпуска) по щелчку на дате в календаре, необходимо проанализировать собы-

тие **Enter**, которое возникает при получении фокуса элементом управления. Переменной, определенной на уровне модуля, можно присвоить два разных значения (например, 1 или 2) в зависимости от того, в каком поле находится фокус. То есть следует создать процедуры, связанные с событием **Enter** обоих элементов управления, и в этих процедурах присвоить переменной необходимые значения. Значения этих переменных затем анализируются при щелчке по дате в календаре, и значение передается в нужный элемент управления.

```
Private Sub Calendar_Click()
If M = 1 Then
 DateBegin.Text = Calendar.Value
 Else
 DateEnd.Text = Calendar.Value
End If
End Sub
```

Фамилии сотрудников вводятся в поле со списком, при этом следует обработать ситуацию повторного ввода фамилии. В этом случае необходимо выдать сообщение о том, что фамилия уже присутствует в списке и не вводить ее в таблицу.

После ввода данных очередного сотрудника при нажатии на кнопку **OK** происходит запись очередной строки в таблицу **График отпусков** рабочего листа**.** Необходимо определить номер первой незаполненной строки и записать в ее ячейки данные из полей формы.

Таблица **График отпусков** может не содержать ни одной записи, в этом случае первая запись вводится в диапазон, левым верхним углом которого является ячейка A2.

ЛАБОРАТОРНАЯ РАБОТА № 4
Разработка приложений в среде MS Excel

Цель работы: освоение способов автоматизации разработки приложений в среде MS Excel

ПОСТАНОВКА ЗАДАЧИ

Необходимо разработать приложение для оформления подписки на газеты и журналы, результатом работы которого должен быть комплект заполненных бланков на подписку. Образцы незаполненного и заполненного бланков приведены на рис. 13 и 14.

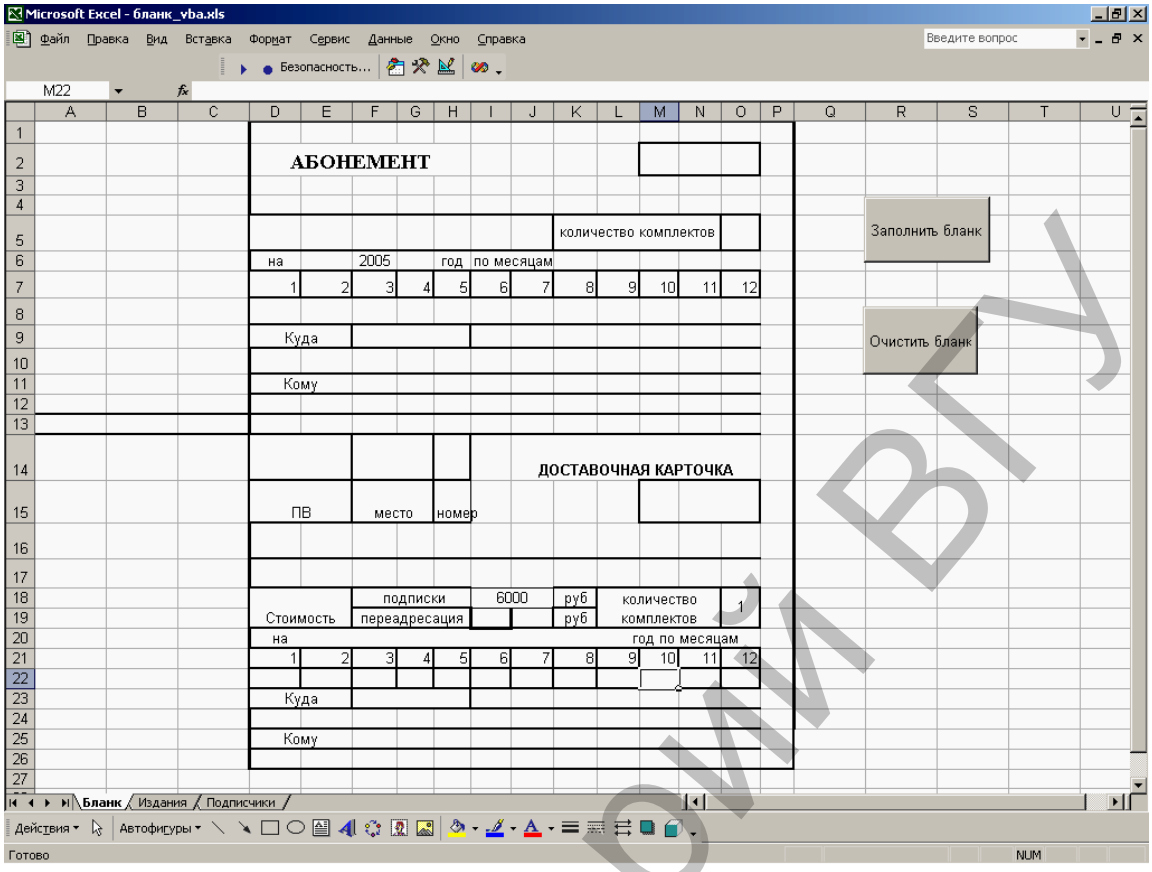

Рис. 13.

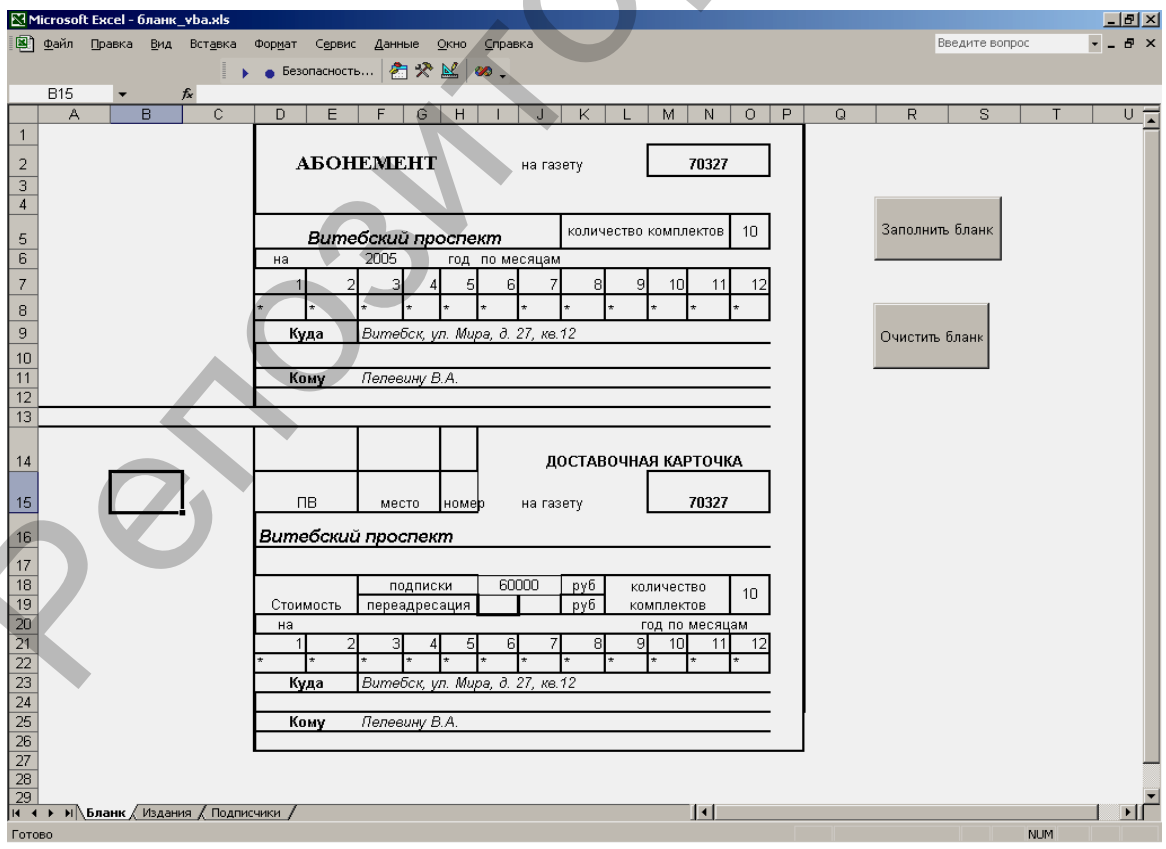

Рис. 14.

При оформлении подписки и заполнении бланков используются две таблицы. Первая таблица содержит список периодических изданий и необходимые сведения об изданиях. Таблица находится на рабочем листе **Издания.** Образец таблицы приведен на рис. 15.

	A	B	C	D
1	**Тип**	**Индекс**	**Навание**	**Цена на 2005**
2	г	70327	АИФ	500
3	г	34325	Витебский проспект	150
4	г	76777	Московский комсомолец	300
5	г	32000	Витебский курьер	350
6	ж	73217	Компьютер пресс	500
7	ж	67987	Звезда	1000
8				
9				

Рис. 15.

Вторая таблица содержит сведения о подписчиках. Таблица находится на рабочем листе **Подписчики**. Образец таблицы приведен на рис. 16.

	A	B	C	D
1	**ФИО**	**Индекс**	**Адрес1**	**Адрес2**
2	Пелевин В.А.	210000	Витебск	ул. Мира, д.27,кв 12
3	Ерофеев В.А.	210007	Витебск	ул. Горького, д.27,кв 12
4				

Рис. 16.

Работа приложения будет заключаться в следующем. Проводящий подписку сотрудник будет открывать рабочий лист с чистым бланком (см. рис. 13), содержащим кнопки **Заполнить бланк** для вызова процедуры заполнения бланка и **Очистить бланк**. Процедура очистки бланка удаляет только те сведения, которые отсутствуют в чистом бланке. Неизменяемую часть бланка следует защитить при создании бланка.

При нажатии на кнопку **Заполнить бланк** открывается диалоговое окно **Подписка на газеты и журналы**, в котором с помощью раскрывающихся списков можно выбрать фамилию подписчика и издание, а также ввести данные о периоде подписке, количестве комплектов и т.д. Образец формы приведен на рис. 17.

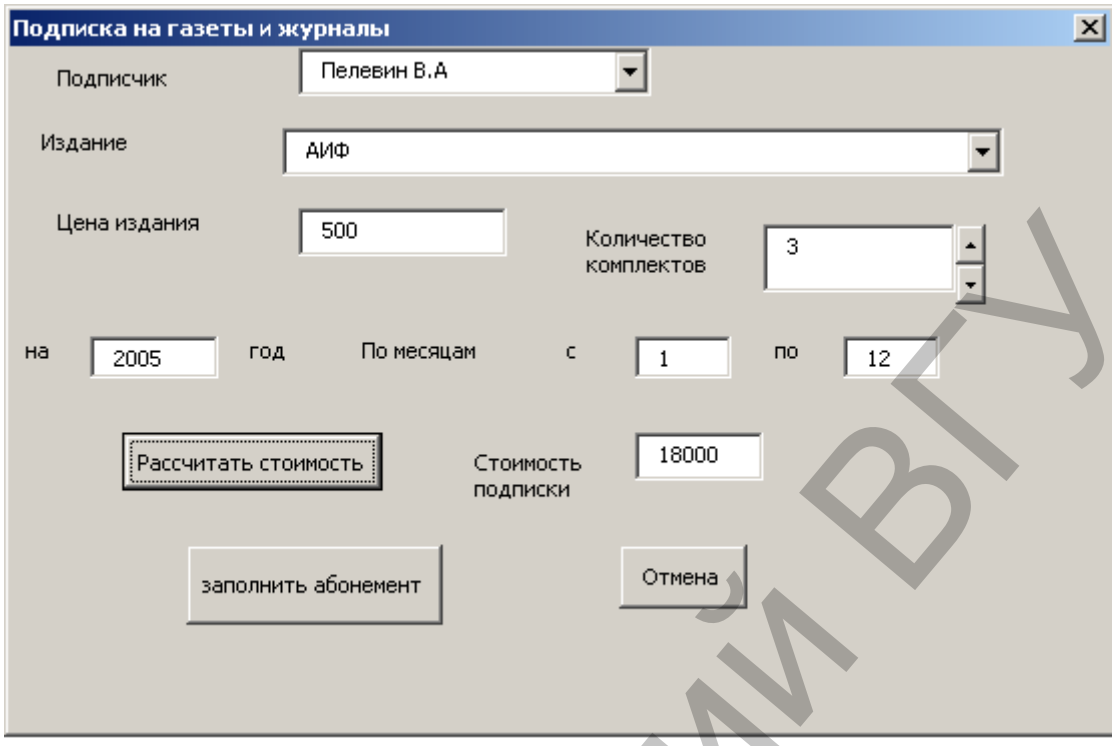

Рис. 17.

Фамилии подписчика и название издания выбираются из раскрывающихся списков. Цена издания отображается автоматически после выбора издания. Год издания также может быть отображен автоматически на основании данных таблицы **Издания.**

Период подписки может быть задан программой по умолчанию (с 1 по 12 месяц), и при необходимости изменен при вводе данных в бланк. Количество комплектов также может быть задано по умолчанию (1).

После ввода всех необходимых данных для расчета стоимости, с помощью кнопки **Рассчитать стоимость** вычисляется общая стоимость подписки.

Затем данные передаются в карточку на рабочем листе **Бланк.**

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Создать бланк, приведенный на рис. 17. Установить нужные размеры строк и столбцов, подобрать необходимые шрифты, установить рамки для ячеек и групп ячеек в соответствии с образцом. Отменить отображение сетки. Выполнить защиту всех ячеек, кроме тех, в которые будут вводиться данные (Сначала отменить защиту изменяемых ячеек, затем задать защиту листа). Рабочему листу присвоить имя **Бланк.**

2. Ввести данные об изданиях по образцу, приведенному на рис. 15. Рабочему листу присвоить имя **Издания.**

3. Ввести данные о подписчиках (не менее 5 подписчиков) по образцу, приведенному на рис. 16. Рабочему листу присвоить имя **Подписчики.**

4. Спроектировать диалоговое окно **Подписка на газеты и журналы (**см. рис. 17). Рядом с полем **Количество комплектов** поместите регулятор (элемент управления **ScrollBar**).

Для полей со списком задайте свойства:

- **ListRows** = 5 (максимальные размеры отображаемого участка списка);
- **Style** = **fmStyleDropDownList (**разрешает выбирать элемент только из списка).

5. На рабочем листе **Бланк** создать кнопку **Заполнить бланк.** К кнопке привязать вызов процедуры открытия и инициализации диалогового окна **Подписка на газеты и журналы.**

6. Разработать процедуру инициализации и открытия диалогового окна **Подписка на газеты и журналы.** В процедуре предусмотреть:

- Определение размеров заполненных списков на рабочих листах **Издания** и **Подписчики**. Эти размеры (количество записей) необходимы для задания свойства **RowSource**, устанавливающего интервал ячеек, содержащих исходные данные для полей со списком. Для задания значения этого свойства используется оператор ***UserForm1.ComboBox1.RowSource = "Подписчики!a2:a" & Last***, где **Last** – количество записей в таблице **Подписчики**.

Для вычисления количества записей организовать цикл, условием завершения которого будет нахождение первой незаполненной строки в таблице.

Задать пустое значение элемента списка с помощью свойства **ListIndex** до выбора значения из списка в поле **Подписчик *(UserForm1.ComboBox1.ListIndex = -1,*** а в поле **Издание** отобразить первый элемент списка (**ListIndex =0)**.

- Заполнение значений периода подписки по умолчанию(с 1 по 12 месяц), количества комплектов (1), значения года подписки (из таблицы **Издания**).
- Вывод диалогового окно на экран: ***UserForm1.Show.***

7. Занести цену в поле **Цена (**найти номер элемента в списке с помощью свойства **ListIndex,** а затем по найденному номеру найти соответствующее значение цены в таблице **Издания).** Привязать к событию изменения значения поля, содержащего название издания ***ComboBox2_Change().***

8. Предусмотреть присваивание значения поля ***Количество комплектов*** (поле ***TextBox5)*** с помощью счетчика (***SpinButton***). Для этого создать процедуру:

Private Sub SpinButton1_Change()
TextBox5.Value = SpinButton1.Value
End Sub

Выполнить проверку ввода в поле ***Количество комплектов.*** Значение должно находиться в интервале от 1 до 100.

9. К кнопке диалогового окна **Рассчитать стоимость** привязать процедуру расчета стоимости подписки с учетом цены, периода подписки и количества комплектов.

10. К кнопке диалогового окна **Заполнить абонемент** привязать вызов процедуры заполнения бланка абонемента.

В процедуре предусмотреть следующие действия:

- Активировать рабочий лист **Бланк (Worksheets("Бланк").Activate).**
- Вывести на бланк данные о названии издания с помощью его номера (свойство **ListIndex)** в поле **Издание** диалогового окна; о типе издания («на газету», «на журнал» и т.д. в зависимости от значения поля **тип** таблицы **Издания),** индексе издания. Оформить заполнение в виде процедуры. Предусмотреть возможность вывода названия издания в две строки в том случае, когда длина строки названия превышает максимальное значение поля вывода.
- Заполнить поле **количество комплектов** из диалогового окна.
- Заполнить поле **год** (из таблицы **Издания).**
- Организовать цикл для заполнения периода подписки символами **("*").**
- Заполнить поле **стоимость подписки.**
- Заполнить **адрес** и **ФИО** подписчика.

11. К кнопке **Отменить** диалогового окна привязать процедуру очистки полей окна. На рабочем листе **Бланк** создать кнопку **Очистить бланк.** Создать процедуру, очищающую все незащищенные ячейки бланка.

ЛАБОРАТОРНАЯ РАБОТА № 5
Применение VBA для создания приложений, работающих с базами данных (БД)

Цель работы: освоение способов применения языка VBA для работы с базами данных.

ОСНОВНЫЕ СВЕДЕНИЯ

Извлечение данных из БД

С целью упрощения доступа к БД были разработаны специальные протоколы DAO, ADO, ODBC.

Рассмотрим применение протокола ADO для работы с базами данных MS Access. Технология ADO предоставляет разработчику универсальный метод извлечения данных из разнообразных источников, не ограничиваясь БД. Технология ADO позволяет:

- Создавать соединение с источником данных.
- Создавать объект, реализующий SQL-команду.
- Выполнять SQL-команду.
- Сохранять результат выполнения SQL- команды в кэше.
- Конструировать виртуальное представления кэша, чтобы пользователь мог сортировать, фильтровать данные, перемещаться по ним.
- Редактировать данные.
- Обновлять источник данных в соответствии с внесенными изменениями.

В объектную модель **ADO** входят следующие объекты:

Connection – среда, в которой выполняется обмен данными с источниками данных. Соединение с источником данных необходимо создать до выполнения любых операций.

Command – способ управления источником данных.

Parameter – переменные компоненты объекта **Command,** которые уточняют способ выполнения команд**.**

Recordset – локальный кэш для данных, считанных из источника.

Field – столбец таблицы данных **Recordset.**

Error – инкапсулирует данные о сгенерированной ошибке.

Property – определяет объекты **Connection**, **Command, Recordset**.

Создание ссылки на библиотеку ADO

Прежде чем применять технологию ADO, должна быть инсталлирована библиотека ADO. Она инсталлируется вместе с MS Windows и может функционировать в любой среде разработки, в которой поддерживается взаимодействие с объектами ActiveX, включая Visual C++, Visual Basic.

Если библиотека ADO инсталлирована, то ее объекты можно использовать после установки ссылки на эту библиотеку. Для этого следует:

- Выбрать команду **Tools|References**.
- Установить флажок **Microsoft ActiveX Data Objects 2.0 Library.**

Объект Connection. Установка подключения к БД

Объект **Connection** представляет собой среду, в которой выполняется обмен данными с источником данных. Приложение может об-

ращаться к БД любых поставщиков (SyBase, Oracle, Informix и др.) с помощью объекта **Connection**. Если необходимо ссылаться на наборы в текущей БД, в которой хранится программный код, следует использовать объект ***CurrentProject.Connection.***

Соединение необходимо создать до любых операций с данными. После прекращения обмена данными с источником соединение должно быть закрыто. Для установки подключения к источнику данных применяется метод **Open** объекта ***Connection,*** а для закрытия – метод **Close.**

Свойство **Provider** объекта **Connection** является текстовой строкой, задающей тип провайдера**,** который будет применен для подключения**.**

Свойство **ConnectionString** объекта **Connection** указывает способ подключения к источнику данных. При использовании Jet-провайдера строка подключения состоит из полного имени файла. При использовании провайдера **ODBC** строка подключения может задавать имя источника данных **DSN**.

Объект RecordSet и его создание

После установления соединения с источником данных можно создать объект **Recordset.** Этот объект представляет собой локальный кэш для исходной таблицы **БД** или результирующего набора считанных из источника записей, возвращаемых в результате запроса. Объект **Recordset** – это виртуальный набор записей из БД, позволяющий управлять данными в БД на уровне записей. Он используется для добавления, изменения или удаления записей из отдельной таблицы базы данных.

На уровне полей управление данными осуществляется объектом **Fields**, причем свойство **Fields** возвращает семейство всех полей данного набора записей.

Рассмотрим пример, в котором сначала создается соединение с именем **cn** и задаются его свойства. Затем создается набор данных с именем **RS**. Источником набора данных **RS** является запрос, созданный на основе таблицы ***Клиенты.*** Значение поля **ФИО** из первой строки запроса передается в ячейку **A2** активного листа активной рабочей книги MS Excel.

```
Dim cn As ADODB.Connection
Set cn = New ADODB.Connection
cn.Provider = "Microsoft.Jet.OLEDB.4.0"
cn.ConnectionString = "e:\db4.mdb
cn.Open
Dim RS As New ADODB.Recordset
RS.Source = "Select ФИО FROM Клиенты"
```

Set RS.ActiveConnection = cn
RS.Open
Rs.MoveFirst
Range("A2").Value = RS.Fields("ФИО").Value
RS.Close
cn.Close

Имя соединения можно указать как параметр метода **Open,** например**:**

Rs.Open "Клиенты", cn
или
Call Rs.Open ("Клиенты", cn)

Объекты **Recordset** используются для обработки данных в базе данных на уровне ***записи***. При работе с объектами доступа к данным почти все операции выполняются с помощью объектов **Recordset**. Каждый объект **Recordset** состоит из записей (строк) и полей (столбцов).

Свойства объекта RecordSet

Свойство	Описание
AbsolutePosition	Порядковый номер текущей записи в наборе записей
ActiveConnection	Подключение, при помощи которого был создан данный набор записей
BOF	Возвращает значение **True**, если указатель текущей записи находится перед первой записью набора записей и значение **False** – в противном случае
BookMark	Закладка, уникальный идентификатор текущей записи набора записей
EOF	Возвращает значение **True**, если указатель текущей записи находится после последней записи набора записей и значение **False –** в противном случае
Fields	Семейство полей
RecordCount	Число записей в наборе
Source	Команда или запрос, создавший данный набор записей
NoMatch	Возвращает значение **True**, если нужная запись не найдена и значение **False –** в противном случае

Методы объекта RecordSet

Если необходимо создать новую запись в наборе, перейти к следующей записи в наборе записей, отредактировать запись применяется соответствующий метод объекта **RecordSet** или объектной переменной, указывающей на объект. Многие методы имеют аргументы, указывающие, как они должны действовать на объекты. Наиболее часто применяются методы, позволяющие создать набор записей, а затем найти, обновить, вставить или удалить строку из набора данных.

Чтобы **создать** набор записей, надо сначала объявить объектную переменную типа **RecordSet**, затем присвоить свойству ***ActiveConnection*** значение соединения, с помощью которого был создан данный набор, открыть набор записей, применив метод **RecordSet.Open** к текущей базе данных, указав имя таблицы или запроса**.**

Например, опишем объектную переменную ***Nabor*** типа **RecordSet** и присвоим ей значение таблицы ***Склад,*** находящейся в текущей БД:

Dim Nabor As New ADODB.Recordset
Set Nabor.ActiveConnection = имя соединения
Call Nabor.Open("Склад", имя соединения, adOpenDynamic, adLockOptimistic)

Третий параметр метода **Open** определяет тип курсора. Запись ***adOpenDynamic*** означает, что данные могут добавляться, модифицироваться и удаляться, причем все изменения, вносимые в это время в таблицу другими пользователями, будут динамически отслеживаться.

Аргумент ***adLockOptimistic*** определяет тип блокировки записей. Оптимистический уровень блокировки означает, что запись будет блокирована одним пользователем, другим пользователям будет отказано в доступе, предполагающем внесение изменений в тот момент, когда вы вносите свои изменения.

После открытия набора записей для перехода к определенной записи можно использовать один из методов **Move:(MoveFirst, MoveLast, MoveNext, MovePrevious, Move n).**

Для перехода к определенной строке в наборе записей, удовлетворяющей заданным условиям, используется метод **Find** с заданием строковой переменной, содержащей условие поиска. Например, для того, чтобы найти первую запись набора записей **Tbl,** в которой значение поля **Залог** превышает 2000 необходимо задать инструкцию:

Tbl.Find "Залог < 2000"

Для **удаления** записи из набора необходимо переместиться к ней и применить метод **Delete:**

Tbl.Delete.

Для удаления всех записей применяется метод ***Delete*** с аргументом ***adAffectAll***:

Call Tbl.Delete(adAffectAll).

Для обновления содержимого строк в наборе записей необходимо перейти к первой обновляемой строке, по имени обратиться к любому полю записи и изменить его значение. Поля записи относятся к семейству **Fields**. Поэтому ссылки на поля могут иметь вид: ***имя набора данных.Fields("имя поля")***

или ***имя набора данных! [имя поля]***.

Перед переходом к другой строке следует использовать метод **Update**, для сохранения изменений. Например, для увеличения значения поля **Залог** на 10% для найденной записи необходимо вычислить новое значение поля, а затем сохранить запись.

Tbl![Залог] = Tbl ![Залог] *1.1

Tbl.Update

Для присвоения полю значения используется запись:

Tbl![Залог].Value = 2500

Для вставки новой строки в набор записей используется метод **AddNew**. Затем устанавливаются значения полей в новой строке. После установки значений полей необходимо применить метод **Update** для сохранения новой записи.

Ошибки и их обработка

При добавлении процедуры обработки ошибок следует учитывать способ передачи управления процедуре при возникновении ошибки. Первым этапом передачи управления является подключение обработчика ошибок путем включения некоторой формы инструкции **On Error** в процедуру. Инструкция **On Error** передает управление процедуре обработки события возникновения данной ошибки. Если инструкция **On Error** отсутствует, то при возникновении ошибки Visual Basic прерывает выполнение программы и выводит сообщение об ошибке.

При возникновении ошибки в процедуре с подключенным обработчиком ошибок Visual Basic не выводит обычного сообщения об ошибке, а передает управление в обработчик. В обработчике ошибок может определяться тип ошибки и осуществляться произвольная обработка.

Существуеть три формы инструкции **On Error**: **On Error GoTo метка, On Error GoTo 0** и **On Error Resume Next.** Инструкция **On Error GoTo метка** подключает процедуру обработки ошибок, начиная с той строки, на которой она находится. Подключить обработчик следует перед первой строкой, которая может привести к возникновению ошибки.

Строка, заданная в качестве аргумента **метка** должна быть первой строкой процедуры обработки ошибок.

Инструкция Resume

Инструкция **Resume** передает управление обратно в процедуру из обработчика ошибок. Эту инструкцию следует включать в обработчик, если требуется передача управления в определенную точку процедуры. Однако инструкция **Resume** не является обязательной. После завершения работы обработчика процедура может быть также завершена.

Существует три формы инструкции **Resume**. Инструкции **Resume** или **Resume** 0 возвращают управление строке, при выполнении которой произошла ошибка. Инструкция **Resume Next** возвращает управление строке, непосредственно следующей за строкой, вызвавшей ошибку. Инструкция **Resume метка** передает управление строке, заданной в качестве аргумента метка. Аргумент метка может указывать номер строки или метку.

Объект Err

Объект **Err** является объектом **Visual Basic**. При возникновении ошибки Visual Basic информация по ней записывается в объект **Err.** Этот объект может содержать сведения только по одной ошибке. При возникновении новой ошибки информация объекта **Err** обновляется.

Свойства и методы объекта **Err** применяются для получения информации по отдельной ошибке. Свойство **Number** является стандартным свойством объекта **Err**; оно возвращает идентификационный номер возникшей ошибки.

Пример 1. Составить процедуру ввода данных в таблицу **Клиенты** с обработкой ошибок.

```
Sub InputR()
Dim Nabor As New ADODB.Recordset
Dim Temp As Variant
Dim Fiel As Field
Call Nabor.Open("Клиенты", CurrentProject.Connection,, adOpenDynamic, adLockOptimistic)
Do While True
 Nabor.AddNew' создает новую запись
 For Each Fiel In Nabor.Fields
 On Error GoTo Обработка_ ошибок
A: Temp = InputBox("введите значение поля" & Fiel.Name & "(Q=выход):") ', Fiel.Name)
 If (Temp = "Q") Then Exit Do
 Fiel.Value = Temp
 Next
```

```
Nabor.Update 'сохраняет запись
Loop
Set Nabor = Nothing
Обработка_ ошибок:
 If Err.Number <> 0 Then
 If Err.Number = 2001 Then
 MsgBox "Ошибка типа": Resume A
 Else
 msg = "ошибка" & Str(Err.Number) & Chr(13) & Err.Description
 MsgBox msg
 End If
End If
End Sub
```

Процедура может быть вызвана из стартовой кнопочной формы базы данных, для этого к событию “Нажатие кнопки” следует привязать макрос, содержащий команду “**ЗапускПрограммы**”. Однако следует иметь в виду, что эта макрокоманда вызывает процедуру **Function** Visual Basic. Имя функции вводится в поле **Имя функции** в разделе **Аргументы макрокоманды**.

Для выполнения процедуры **Sub** или процедуры обработки события, написанной на языке Visual Basic, следует создать процедуру ***Function***, вызывающую процедуру ***Sub*** или процедуру обработки события. После этого становится возможным вызов этой процедуры ***Function*** с помощью макрокоманды **ЗапускПрограммы**.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислите компоненты, входящие в объектную модель ADO.
2. Раскройте назначение каждого из компонентов, входящие в объектную модель ADO.
3. Как установить ссылку на необходимый объект ADO?
4. Какое свойство объекта **Connection** позволяет указать путь к источнику данных?
5. Какой аргумент метода **Open** позволяет выполнять любые операции над записями динамического набора записей?
6. Какой метод объекта **RecordSet** позволяет осуществлять поиск записей, удовлетворяющих определенным условиям?
7. Для чего используется инструкция **Resume**?
8. Какие формы инструкции **Resume** существуют?

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Создать процедуры для ввода данных в таблицы БД «Прокат дисков». Подключить обработчики ошибок. Предусмотреть возможность повторного ввода пользователем одного и того же диска в таблицу **Склад.** То есть, когда вводится новая запись, необходимо проверить, не дублирует ли поле **Диск** те одноименные поля, которые уже были введены в таблицу в предшествующих записях. Тестирование может быть выполнено следующим образом: для каждой новой записи подсчитывается количество совпадений введенного значения с полем **Диск** уже введенных записей. Если это число <>0, то программа извещает пользователя о наличии дубликата, если количество совпадений =0, то процедура вводит значение в БД.

Для подсчета количества записей, отвечающих заданному условию можно использовать функцию **DCount.**

Функция ***Dcount*** возвращает число записей в заданном наборе записей. Синтаксис: ***Dcount (выражение,набор[условие]),*** где

Выражение – определяет поле, для которого производится подсчет значений. Допускается использовать в аргументе ***Выражение*** имя поля в таблице или элемента управления в форме константы, а также имя встроенной или определяемой пользователем функции.

Набор – строковое выражение, которое определяет набор записей, образующих подмножество. Может представлять имя таблицы или запроса.

Условие – необязательное строковое выражение, ограничивающее диапазон данных, для которых рассчитывается число значений. Аргумент **условие** является эквивалентом предложения **Where** в инструкциях SQL. Если аргумент **условие** опущен, ***Dcount*** выполняет расчеты над полем, заданным в аргументе ***выражение*** для всего набора записей.

Присвоим переменной **T** значение равное количеству записей, совпадающих с переменной **Temp,** значением которой является название диска, введенное пользователем:

T = DCount("Диск", "Склад", "Диск=" & "'" & Temp & "'")

2. Создать процедуры для изменения значения поля **№_телефона** таблицы **Клиенты** по следующему правилу: если первая цифра равна 6 – заменить ее на 4, если первая цифра равна 5 – заменить ее на 9.

3. Создать процедуру ввода в одну из таблиц базы данных, разработанной в соответствии с индивидуальным заданием [2].

ЛАБОРАТОРНАЯ РАБОТА № 6
Поиск информации в таблицах базы данных

Цель работы: Применение языка VBA для поиска информации в таблицах базы данных.

ОСНОВНЫЕ СВЕДЕНИЯ

Поиск записей

Существует 2 основных способа поиска записей. Можно создать запрос SQL или построить символьную строку, содержащую запрос на языке SQL. А можно открыть набор данных и, используя процедуру VBA, найти нужные записи.

Пример 1.

Создадим процедуру для поиска произвольных, заданных пользователем данных в таблице **Клиенты**.

```
Sub FindR()
Dim cn As New ADODB.Connection
Dim Nabor As New ADODB.Recordset
Dim Temp As Variant
Dim Fiel As Variant
Dim D As Variant
Dim H As Variant
Set cn = CurrentProject.Connection
Set Nabor.ActiveConnection = cn
 Call Nabor.Open("Клиенты", cn)
Nabor.MoveFirst
 Temp = InputBox("Введите данные для поиска (Q=Выход):",
"Поиск_ данных")
 If (Temp = "Q") Then Exit Sub
Do While (Nabor.EOF = False)
 For Each Fiel In Nabor.Fields
 If (Fiel.Value = Temp) Then
 D= Fiel.Value
 H = Nabor![ФИО].Value
 Exit Do
 End If
 Next
 Nabor.MoveNext
Loop
Nabor.Close
Set Nabor = Nothing
MsgBox "Найдено:" & D & "для клиента" & H
End Sub
```

Поиск с использованием SQL – запроса.

Процедура позволяет вывести шифры клиентов из таблицы **Прокат,** у которых находится заданный оператором диск**.**

Пример 2.

```
Sub FindR(Pole As String, Znach As String)
Dim cn As New ADODB.Connection
Dim Nabor As New ADODB.Recordset
Dim SQl As String
SQl = "Select * From Прокат Where "& Pole & "=" & "'" & Znach & "'"
Set cn = CurrentProject.Connection
Set Nabor.ActiveConnection = cn
 Call Nabor.Open(SQl, cn)
 Nabor.MoveFirst
 While Not Nabor.EOF
MsgBox Nabor!Шифр
 Nabor.MoveNext
 Wend
End Sub

Sub Findtest()
Call FindR("Диск", "ДДТ")
End Sub
```

Для того чтобы сделать программу более универсальной будем принимать от пользователя значение поля поиска и искомого значения. С помощью процедуры можно вывести список клиентов, взявших заданный диск, или список клиентов, взявших диск на заданное число дней.

Пример 3.

```
Sub FindR(Pole As String, Znach As Variant)
Dim cn As New ADODB.Connection
Dim Nabor As New ADODB.Recordset
Dim SQl As String
If IsNumeric(Znach) Then
 SQl = "Select * From Прокат Where " & Pole & "=" & Znach
 Else
 SQl = "Select * From Прокат Where " & Pole & "=" & "'" & Znach & "'"
End If
Set cn = CurrentProject.Connection
Set Nabor.ActiveConnection = cn
 Call Nabor.Open(SQl, cn)
 Nabor.MoveFirst
```

```
While Not Nabor.EOF
MsgBox "Шифр =" & Str(Nabor(0))
Nabor.MoveNext
Wend
End Sub

Sub Findtest()
Dim Fiel As String
Dim Znachenie As Variant
Fiel = InputBox("В каком поле ищем?", "Ввод поля", "Диск")
Znachenie = InputBox("Что ищем?", "Ввод значения поля", "Диск")
Call FindR(Fiel, Znachenie)
End Sub
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие способы поиска информации используются при создании приложений, работающих с базами данных?

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Создать процедуру для поиска любых, заданных пользователем данных, содержащихся в таблице **Склад** БД “Прокат дисков” без использования SQL-запроса.

2. Создать процедуру с использованием SQL-запроса для поиска любых, заданных пользователем данных, в одной из таблиц базы данных, разработанной в соответствии с индивидуальным заданием [2] .

ЛАБОРАТОРНАЯ РАБОТА № 7
Взаимодействие приложений MS Office

Цель работы: освоение способов взаимодействия приложений MS Office.

ОСНОВНЫЕ СВЕДЕНИЯ

Вывод набора записей базы данных на рабочий лист MS Excel

Для вывода набора записей на рабочий лист можно воспользоваться методом

CopyFromRecordset(Data, MaxRows, MaxColumns)

- ***Data*** – обязательный параметр, задающий копируемый набор записей;
- ***MaxRows*** – необязательный параметр, определяющий максимальное количество копируемых записей;
- ***MaxColumns*** **–** необязательный параметр, определяющий максимальное количество копируемых полей.

Например, для вывода набора записей **rs** на рабочий лист1 рабочей книги **NewBook** используется команда:

NewBook.Sheets(1).Range("A2"). CopyFromRecordset rs

Предварительно следует выполнить определение объекта **NewBook** типа WorkBook и создать объект **NewBook** с помощью метода **Add:**

Set NewBook = WorkBooks.Add

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какая процедура используется для вывода набора записей БД на рабочий лист?
2. Перечислите параметры метода ***CopyFromRecordset*** и их назначение**.**

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Задание 1.

1. Создать базу данных, содержащую таблицы:

Таблица 2

Дисциплины			
Дисциплина	**Вид занятий**	**Часы**	**Всего подгрупп**
Исследование операций	лек	34	1
Компьютерная графика	лек	34	1
Компьютерные сети	лаб	120	3
Компьютерные сети	лек	64	1
Модели данных и СУБД	лек	34	1
Операционные системы	лаб	100	3
Операционные системы	лек	80	1
ЭВМ и прогаммирование	лаб	120	2
ЭВМ и прогаммирование	лек	68	1

Таблица 2 содержит сведения о том, какие дисциплины и в каком объеме должны быть проведены кафедрой.

Таблица 3

Преподаватели	
Код	**ФИО**
1	Ставрогин Н.В.
2	Иванов И.И.
3	Карапетян Ф.А.
4	Булгаков С.Н.
5	Верховенский С.Т.

Таблица 3 содержит сведения о преподавателях.

Таблица 4

Распределение дисциплин			
Код	**Дисциплина**	**Вид занятий**	**количество подгрупп**
1	Исследование операций	лек	
1	Компьютерная графика	лек	
1	Компьютерные сети	лек	
2	Операционные системы	лаб	
2	ЭВМ и прогаммирование	лек	
3	ЭВМ и прогаммирование	лаб	
4	Компьютерная графика	лек	
4	Модели данных и СУБД	лек	
5	Исследование операций	лек	
5	Операционные системы	лаб	

Таблица 4 содержит данные о том, какие дисциплины и какие виды занятий могут быть проведены каждым преподавателем.

2. Создать запрос для определения нагрузки каждого преподавателя в часах. Результат запроса должен представлять собой следующую таблицу:

Таблица 5

Запрос1						
Дисциплина	**Всего подгрупп**	**ФИО**	**Вид занятий**	**Часы**	**Количество подгрупп**	**Нагрузка**
Компьютерная графика	1	Ставрогин Н.С.	лек	34	1	34
Компьютерные сети	1	Ставрогин Н.С.	лек	64	1	64
ЭВМ и прогаммирование	1	Иванов И.И.	лек	68	1	68
ЭВМ и прогаммирование	2	Карапетян Ф.А.	лаб	120	2	240
Модели данных и СУБД	1	Булгаков С.Н.	лек	34	1	34
Операционные системы	3	Верховенский С.Т.	лаб	100	1	100
Исследование операций	1	Ставрогин Н.С.	лек	34	1	34
Операционные системы	3	Иванов И.И.	лаб	100	2	200
Компьютерная графика	1	Булгаков С.Н.	лек	34	1	34
Исследование операций	1	Верховенский С.Т.	лек	34	1	34

3. Создать стартовую форму, содержащую кнопки:

a. для открытия форм, предназначенных для заполнения исходных таблиц;

b. для запуска запроса, представленного в табл. 5;

c. для вывода набора записей запроса на рабочий лист;

d. Для открытия рабочей книги MS Excel.

Для вывода набора записей на рабочий лист создать процедуру, в которой необходимо:

- Открыть набор данных, содержащий результат запроса.
- Создать новую рабочую книгу с помощью метода **Add** (для работы с объектами установить ссылку на библиотеку MS Excel 10.0 Objects Library).
- В первую строку таблицы на рабочем листе ввести заголовок: **Нагрузка:**
- Вторая строка таблицы должна содержать заголовки полей, источником которых являются последовательно считанные имена полей набора записей:

For i = 0 To rs.Fields.Count - 1
Nagr.Sheets(1).Range("A1").Offset(0, i).Value = rs.Fields(i).Name
Next

(***Nagr*** – имя рабочей книги, ***rs*** – набор данных, источником которого является результат запроса).

- Затем, начиная со второй строки вывести набор данных.
- Задать ширину столбцов таблицы, достаточную для размещения данных.
- Вычислить итоговую нагрузку в столбце «нагрузка».

4. Предусмотреть сохранение рабочей книги в файле с помощью метода ***SaveAs***:

Nagr.SaveAs ("d:\nagr")

5. Закрыть рабочую книгу: ***Nagr.Close.***

Задание 2

Дополнить стартовую форму приложения двумя кнопками:

a. сортировка по дисциплине;

b. сортировка по ФИО преподавателя.

Первая кнопка предназначена для сортировки данных таблицы Нагрузка в алфавитном порядке дисциплин. Вторая кнопка предназначена для сортировки данных таблицы в алфавитном порядке ФИО преподавателей. Столбец, по которому выполняется сортировка, должен быть первым столбцом таблицы. Для сортировки используется метод **Sort**.

Предусмотреть получение промежуточных итогов по сортируемому полю (метод **Subtotal**).

Задание 3

Вывести отчеты, предусмотренные вариантом индивидуального задания [2], на рабочий лист MS Excel. Предусмотреть отображение заголовка отчета, названий полей, итогов, а также сохранение рабочей книги.

Л И Т Е Р А Т У Р А

1. Уокенбах Дж. Профессиональное программирование на VBA в Excel 2002. : пер. с англ. – М.: Издательский дом «Вильямс». 2003. – 784 с.
2. Адаменко Н.Д. Практикум по СУБД MS Access. – Витебск: ВГУ им. П.М. Машерова, 2001. – 91 с.
3. Гарнаев А.Ю. MS Excel 2002: разработка приложений. – СПб.: БХФ-Петербург, 2004. – 768 с.
4. Гарнаев А.Ю. Самоучитель VBA. – СПб.: БХФ-Петербург, 2003. – 512 с.
5. Кузьменко В.Г. Базы данных в Visual Basic и VBA. Самоучитель. – М. ООО «Бином-Пресс», 2004. – 416 с.
6. Хомоненко А.Д., Цыганков В.М., Мальцев М.Г. Базы данных: учебник для высших учебных заведений / под ред. проф. А.Д. Хомоненко. – СПб.: КОРОНА принт, 2000. – 416 с.

Учебное издание

АДАМЕНКО Наталья Дмитриевна

ОСНОВЫ ПРОГРАММИРОВАНИЯ НА VBA

Методические рекомендации по выполнению лабораторных работ

Технический редактор *А.И. Матеюн*

Корректор *В.В. Жойдик*

Компьютерный дизайн *Г.В. Разбоева*

Подписано в печать 2009. Формат 60x84 $^{1}/_{16}$. Бумага офсетная. Гарнитура Таймс. Ризография.

Усл. печ. л. 2,91. Уч.-изд. л. 2,01. Тираж экз. Заказ .

Издатель и полиграфическое исполнение – учреждение образования

«Витебский государственный университет им. П.М. Машерова».

ЛИ № 02330 / 0494385 от 16.03.2009.

Отпечатано на ризографе учреждения образования

«Витебский государственный университет им. П.М. Машерова».

210038, г. Витебск, Московский проспект, 33.

Н.Д. Адаменко

О С Н О В Ы ПРОГРАММИРОВАНИЯ НА VBA

Витебск 2009