Московский государственный технический университет
гражданской авиации
Факультет прикладной математики и вычислительной техники
Кафедра прикладной математики

Коновалов В.М.

Прикладное программное обеспечение

ПОСОБИЕ

к выполнению лабораторных работ

для студентов IV курса
специальности 230401
дневного обучения

Москва – 2010

Данное пособие издается в соответствии с рабочей программой учебной дисциплины «Прикладное программное обеспечение» по Учебному плану для студентов IV курса специальности 230401 дневного обучения, утвержденному в 2007 г. Рассмотрено и рекомендовано к изданию на заседании кафедры и методического совета 17.11.2009 г.

СОДЕРЖАНИЕ

Предисловие

Данная методическая разработка определяет содержание и последовательность выполнения студентами лабораторного практикума. Основная задача состоит в приобретении практических навыков работы в инструментальных системах Visual Basic for Applications (VBA) – для разработки **интерактивных** пользовательских приложений на базе MS Office, и Visual Studio .NET – для разработки приложений любых типов: консольных, Windows-приложений, Web-приложений, Web-сервисов и др.

VBA – это инструмент **офисного** программирования. Но в Office многие задачи, связанные с автоматизацией действий пользователя, можно решать без всякого программирования. Тем не менее, класс задач, решение которых может быть получено "руками", ограничен, и все что нельзя сделать руками, можно сделать программно. Для программирования используется VBA, являющийся расширением ядра языка VB (Visual Basic) объектами, определяющими офисные приложения (т.е. входящие в состав пакета MS Office).

Подобно другим средствам, VBA позволяет создавать полностью автоматические программные продукты, которые можно использовать, в частности, при оформлении документов (подготовка текстов), при вычислениях и анализе данных таблиц (электронных таблиц), при манипулировании данными, содержащимися в таблицах базы данных. Перечисленные функции по обработке данных реализуются офисными прикладными программами: Word, Excel, Access и др. VBA, в этом случае, является инструментом, расширяющим их функциональные возможности.

При решении задач с помощью VBA, все создаваемые программные компоненты объединяются в одно целое, называемое **проектом**. Проекты VBA выполняются совместно с другими приложениями. Офисное приложение, в котором разрабатывается и выполняется проект VBA, называется **основным**. Например, можно создать проект VBA, который работает вместе с Microsoft Excel. В этом случае Excel является основным приложением. Не используя основное приложение, нельзя построить приложение VBA. В этом - основное различие между VBA и "чистым" языком программирования Visual Basic (VB), с помощью которого можно создать **полностью** самостоятельное приложение. VBA учитывает специфические особенности основного приложения. Поэтому при разработке функционально ориентированных приложений, для которых можно явно определить основное приложение, предпочтение следует отдавать VBA, а не VB, что **снизит трудоемкость** программирования.

Для реализации цели обучения в рамках настоящего лабораторного практикума достаточно ограничиться основными приложениями, в качестве которых можно использовать офисные прикладные программы MS Excel, MS Word, MS Access, выполняющие функции по работе с вычислениями, текстами и хранимыми данными в базе данных.

Лабораторная работа 1. Интегрированные инструментальные среды Visual Basic for Applications (VBA) и Visual Studio .NET

Продолжительность работы – **4 часа.**

Цель работы: изучить состав, структуру, назначение элементов управления, освоить приемы работы в средах VBA и Visual Studio .NET

Этапы работы:

1. Знакомство с инструментальной средой VBA.

1.1. Выбор основного приложения.

На начальном этапе создания проекта необходимо выбрать **основное** (офисное) приложение для VBA-проекта. Если разрабатываемое Вами приложение (проект) будет применяться для настройки и автоматизации процесса или задачи в некотором офисном приложении, то, очевидно, следует использовать **это** приложение. В общем случае выбрать основное приложение не столь просто, как кажется. При выборе следует руководствоваться следующими принципами:

- в качестве основного приложения следует выбрать программу, которая будет использоваться для запуска проекта;
- в качестве основного приложения следует выбрать программу, в которой можно выполнить большую часть требуемых операций. Например, если требуется работать с диаграммами, производить вычисления, а также формировать документы, то лучше использовать в качестве основного приложения Excel, а не Word.

Разработка приложения в системе VBA выполняется практически полностью в редакторе VBA. Поэтому инструментальной средой является среда редактора VBA. В приложениях Excel и Word эти среды практически идентичны. В Access имеется специфика в структуре среды при сохранении всех ее инструментальных возможностей. Первоначальное знакомство с инструментальной средой можно начать, например, используя в качестве основного приложения Excel.

1.2. Запуск редактора VBA.

Для запуска редактора VBA выбрать в основном приложении (в Excel, в данном случае) команду **Сервис\Макрос\Редактор Visual Basic**. Сразу отметим, что практически всегда имеется альтернативная возможность отработки команды с помощью соответствующей пиктограммы на панели инструментов в верху экрана. Откроется окно редактора VBA (рис. 1).

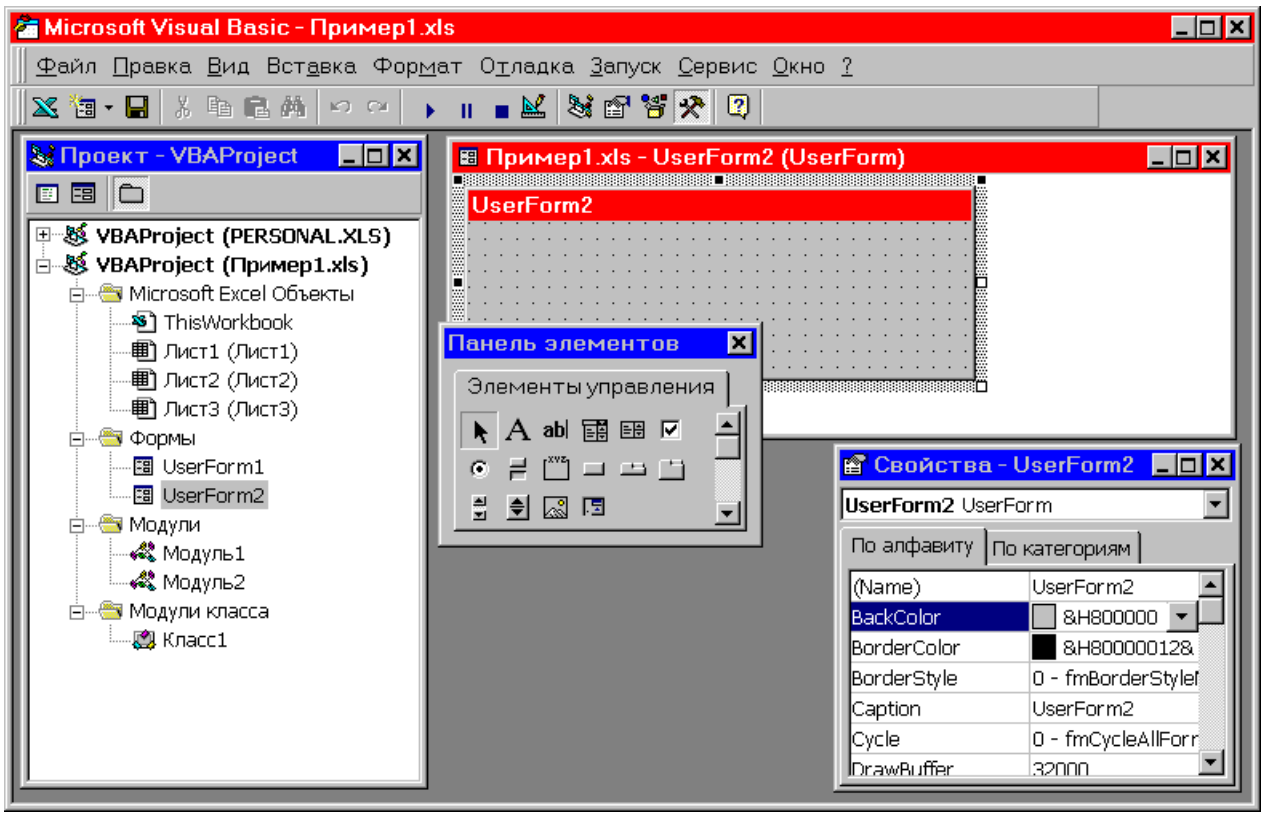

Рис. 1.

Теперь следует изучить возможности по конфигурированию среды, а так же назначение и способы работы с ее компонентами. Конфигурирование выполняется с помощью команд основного меню редактора, в частности, путем отработки команд меню **Вид** и **Вставка**. В результате в окне редактора могут быть активизированы различные компоненты – рабочие окна, панели инструментов, в той конфигурации, какая необходима для эффективной работы.

2. Работа с окном проекта.

Окно проекта – это специальное окно редактора VBA (рис. 2), в котором выводятся все элементы проекта в виде иерархической структуры, включающей все формы, модули кода и объекты основного приложения, например, рабочие листы книги Excel. В окне проекта выводится проект VBA каждого открытого в основном приложении документа.

- Для отображения окна проекта отработайте команду **Вид\Окно проекта**;
- Для скрытия окна проекта можно использовать кнопку **Закрыть** окна проекта, либо в контекстном меню окна проекта отработать команду **Скрыть**;
- Для закрепления окна проекта отработайте команду **Закрепить** из контекстного меню окна проекта;

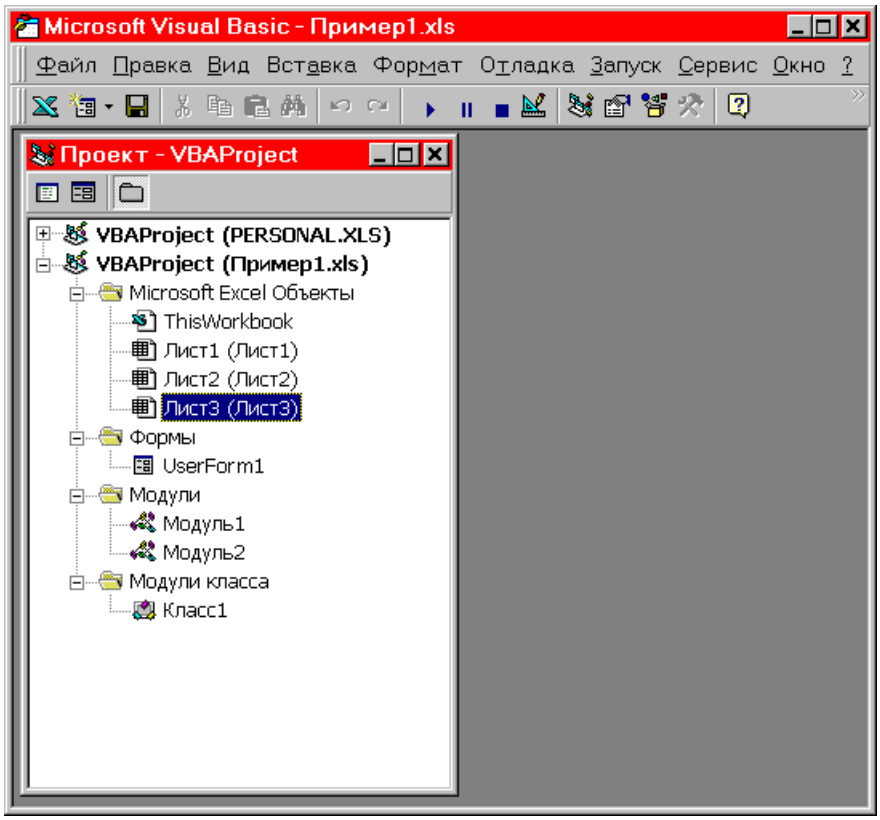

Рис. 2.

- Для перемещения закрепленного окна проекта щелкните по заголовку окна проекта, а затем перетащите окно к верхнему, нижнему, правому или левому краю окна редактора VBA.

В окне проекта можно несколькими способами выбрать требуемый объект. Для выбора и редактирования объекта, выведенного в окне проекта:

- Дважды щелкните по имени объекта;
- Выберите объект, вызовите контекстное меню (щелкнув правой кнопкой мыши по имени объекта), а затем отработайте либо команду **Объект**, либо **Программа**, в зависимости от того, что требуется редактировать. В контекстном меню доступна только та команда, которую можно выполнить. Например, команда **Объект** недоступна для модулей. Для некоторых объектов доступны обе команды. Такими объектами обычно являются документы приложений: документ Word, рабочая книга Excel и т.п., которые имеют как код, так и интерфейсные элементы.

Если требуемый объект не содержится в окне проекта, добавьте его в проект, воспользовавшись командами из меню **Вставка** основной линейки меню редактора, либо командами подменю **Вставить** контекстного меню окна проекта.

- Добавьте в проект форму, отработав команду **Вставка\UserForm**. После выбора выводятся относящиеся к объекту инструменты. В данном случае на экране отображается набор инструментов формы.

3. Работа с окном свойств.

Окно свойств (рис. 3) используется для просмотра и задания свойств объектов проекта. С его помощью можно установить свойства и просмотреть их значения для любого компонента проекта и объекта основного приложения.

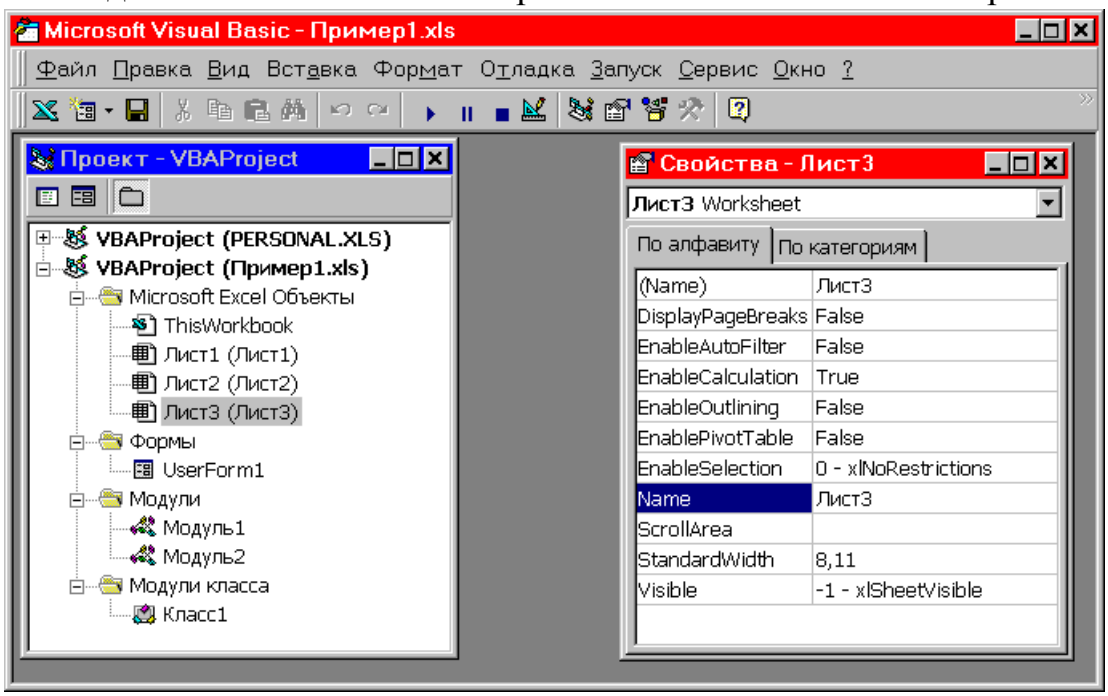

Рис. 3.

- Для отображения окна свойств отработайте команду **Вид\Окно свойств** или нажмите клавишу <F4>;
- Для скрытия окна свойств можно использовать кнопку **Закрыть** окна свойств, либо в контекстном меню окна свойств отработать команду **Скрыть**;
- Для закрепления окна свойств отработайте команду **Закрепить** из контекстного меню окна свойств;
- Для перемещения закрепленного окна свойств щелкните по заголовку окна свойств, а затем перетащите окно к верхнему, нижнему, правому или левому краю окна редактора VBA.

В окне свойств отображаются только свойства текущего объекта. Для вывода свойств объекта:

- Выбрать объект из списка, расположенного вверху окна свойств;
- Выбрать объект в окне проекта и, если окно свойств скрыто, вывести его, нажав клавишу <F4>.

Чтобы изменить значение свойства, следует выбрать требуемое свойство в левом столбце, а затем задать нужную величину в правом столбце окна свойств.

4. Работа с окном модуля.

Разрабатываемые в VBA-проекте программы можно хранить в виде взаимосвязанных процедур в модуле проекта. Различают **стандартные** модули и **модули класса**. Код, используемый для создания объектов, их свойств и методов, содержится в модуле класса. В качестве имени модуля класса обычно

используется имя созданного объекта. В стандартном модуле могут храниться любые процедуры, разрабатываемые пользователем для автоматизации работы приложения.

Чтобы добавить модуль в проект VBA:

1. Выбрать проект, в который требуется добавить модуль, в окне проекта.
2. Отработать в меню редактора VBA (альтернатива – контекстное меню окна проекта) команду **Вставка\Модуль** или **Вставка\Модуль класса**, если в проект добавляется модуль класса. Выводится пустое окно добавляемого модуля (рис. 4).

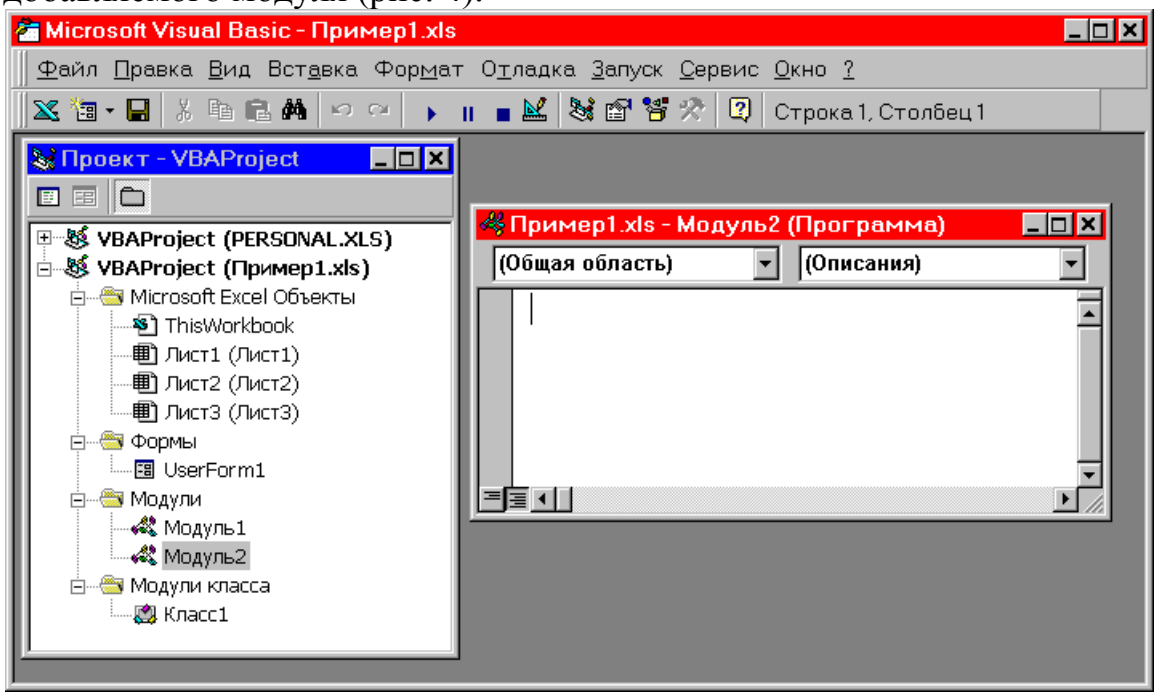

Рис. 4.

3. Вывести окно свойств, нажав клавишу <F4>, и задать имя модуля.

Обратите внимание на два списка, расположенные под строкой заголовка окна модуля. В первом списке выводятся все объекты модуля, а во втором – список процедур, связанных с выбранным объектом.

Для отображения окна модуля можно выполнить одно из следующих действий:

- Дважды щелкнуть в окне проекта по имени требуемого объекта приложения, который может содержать код, например, по названию модуля или модуля класса;
- Дважды щелкнуть по требуемой форме или по любому элементу управления в форме;
- Отработать команду редактора VBA **Вид\Программа** для предварительно выделенного в окне проекта требуемого объекта.
- Отработать в меню основного приложения команду для редактирования макроса. Например, в основном приложении Excel это будет команда **Сервис\Макрос\Макросы\Изменить**.

4.1. Редактирование кода в окне модуля.

Обычно модуль содержит несколько функций или подпрограмм, оформленных в виде процедур. В окне модуля можно либо вывести все процедуры, с отделением текста одной от текста другой горизонтальной линией, либо отобразить только одну процедуру модуля, нажав расположенную в нижнем левом углу кнопку **Представление полного модуля** или кнопку **Представление процедуры**, соответственно.

Код вводится непосредственно в окно модуля, так же как текст в любом текстовом редакторе (рис. 5).

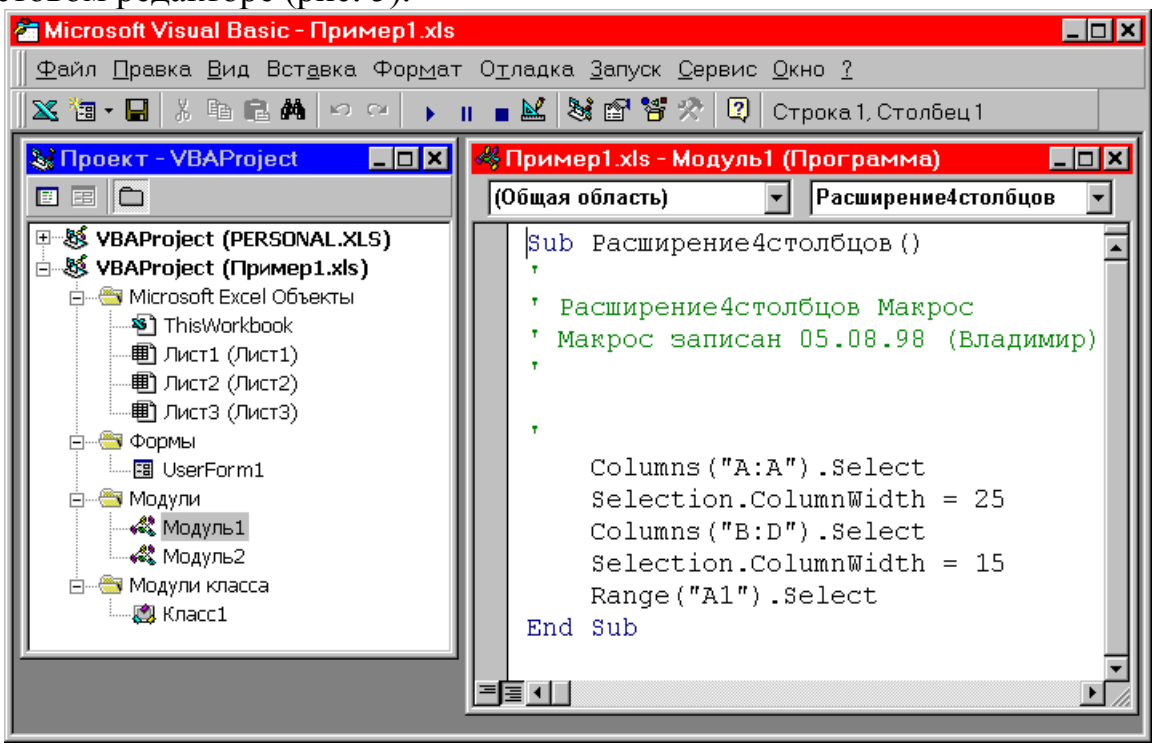

Рис. 5.

Для внесения изменений можно использовать типовые команды редактирования, такие как **Вырезать**, **Копировать**, **Вставить** либо из меню **Правка**, либо из контекстного меню окна модуля, отображаемого при щелчке правой кнопкой мыши в области окна модуля.

4.2. Добавление процедур в проект.

1. Открыть окно модуля, в которое требуется добавить процедуру.
2. Отработать команду **Вставка\Процедура**. Выводится диалоговое окно **Вставка процедуры**.
3. Ввести имя процедуры в поле **Имя**.
4. Выбрать переключатель, задающий тип добавляемой процедуры: подпрограмма, функция или свойство.
5. Задать общую или личную область определения процедуры, выбрав переключатель **Общая (Public)** или **Личная (Private),** соответственно.
6. Если все локальные переменные являются статическими, установить флажок **Все локальные переменные считать статическими**.

7. Нажать кнопку **OK**. В окне модуля выводится пустая процедура с заданным именем.

4.3. Поиск процедур в окне модуля.

По мере увеличения размера модуля становится затруднительным найти требуемую процедуру.

- **Процедуры**. Чтобы быстро найти процедуру, которая находится в текущем модуле, следует выбрать в списке, в верхнем левом углу окна модуля, элемент, соответствующий названию требуемой процедуры.
- **Процедуры обработки события**. Чтобы быстро найти процедуру обработки события для текущего объекта, следует выбрать название события в списке, находящемся в правом верхнем углу окна модуля. Имя объекта при этом устанавливается в списке в левом верхнем углу окна модуля.
- **Описание процедуры**. Чтобы быстро найти описание процедуры следует щелкнуть правой кнопкой мыши по имени требуемой процедуры в любом месте кода и отработать команду **Описание** контекстного меню.
- **Место последней правки**. Чтобы возвратиться в позицию последней правки, следует отработать команду Вернуться к последней позиции контекстного меню окна модуля.

5. Работа с панелью элементов.

Панель элементов позволяет разместить элементы управления на форме. Хотя окно с инструментами отображается автоматически при создании формы, может понадобиться скрыть его, чтобы освободить место на экране. Для вывода панели элементов на экран следует отработать команду **Вид\Панель элементов**.

Кроме стандартного набора инструментов (рис. 6) можно использовать дополнительные инструменты.

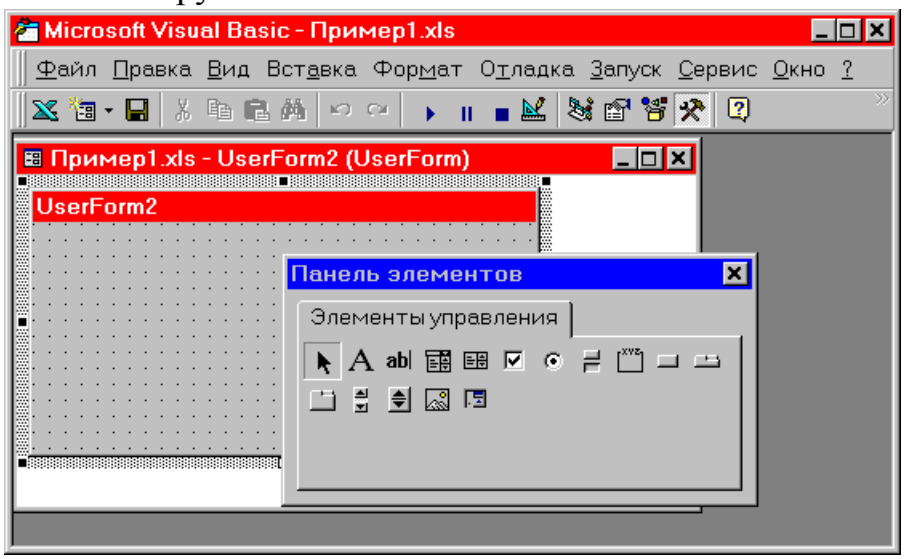

Рис. 6.

Для просмотра набора дополнительных инструментов и переноса их на панель элементов:

1. Открыть требуемую форму.
2. Отработать команду **Сервис\Дополнительные элементы**. Выводится диалоговое окно **Дополнительные элементы**.
3. Сбросить флажок **Только выделенные объекты** для вывода всех доступных дополнительных элементов.
4. Выбрать требуемый элемент управления, установив флажок напротив его имени.
5. Нажать кнопку **OK**. Выбранный элемент выводится на панели элементов.

Разместив на панели элементов все необходимые инструменты, можно приступать к размещению элементов управления на форме (рис. 7).

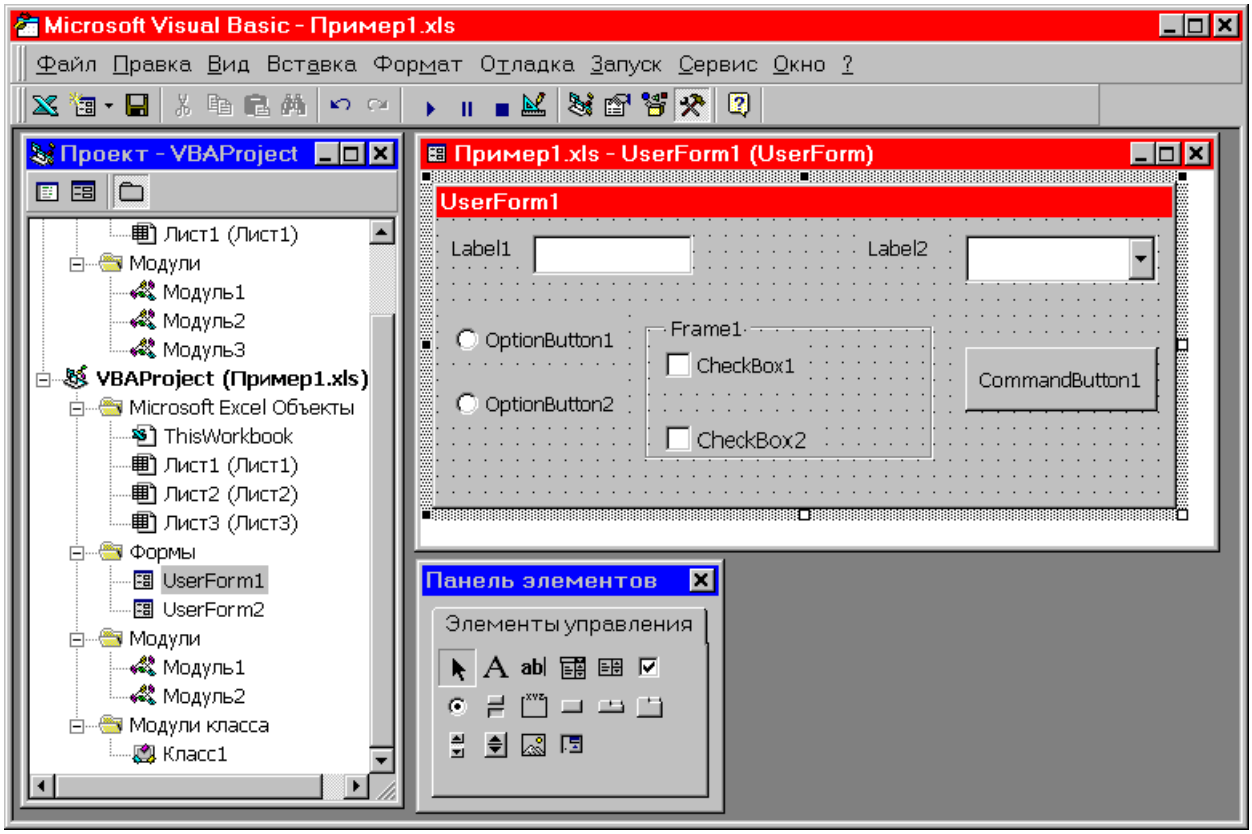

Рис. 7.

Для этого:

1. Выбрать форму, в которую требуется добавить элементы управления.
2. Нажать на панели элементов кнопку требуемого элемента, а затем щелкнуть по любому месту формы. В форме выводится элемент управления.
3. Отбуксировать элемент управления в требуемую позицию.
4. При необходимости изменить размеры элемента управления, перетащив его рамку. То же самое можно сделать и с формой.

6. Работа с формой.

При разработке формы часто необходимо посмотреть на ее внешний вид. Для этого необходимо открыть форму и либо нажать клавишу <F5>, либо

отработать команду **Запуск\Запуск подпрограммы/формы**. Для возврата в окно редактора просто закройте окно формы.

6.1. Выравнивание элементов управления и задание их размеров.

Можно выровнять элементы управления в форме, чтобы не создавалось ощущения, что они размещены беспорядочно. Для выравнивания элементов управления и изменения их размеров используется множество команд меню **Формат** редактора VBA. Предварительно элемент, над которым осуществляются операции, должен быть выделен.

Обычно при форматировании форм требуется выбрать несколько элементов управления для одновременного выполнения над ними одних и тех же операций. Для выделения нескольких объектов щелкните по первому элементу, а затем, удерживая нажатой клавишу <Shift>, выделите другие объекты. Чтобы отменить выбор одного из выделенных объектов, повторно щелкните по этому элементу, удерживая нажатой клавишу <Shift>.

6.2. Задание порядка перехода от одного элемента управления к другому.

Порядок перехода определяет последовательность, в которой активизируются объекты при нажатии клавиши <Tab>. Необходимо определить такой порядок перехода, при котором упрощалась бы работа с формой. Например, при задании новой информации требуется задать перемещение сначала ко всем полям ввода данных, а затем на кнопку, которая сохраняет запись и выводит новую запись.

Для задания последовательности перехода от одного элемента управления к другому:

1. Открыть форму, дважды щелкнув по ее имени в окне проекта.
2. Если элементы управления, для которых требуется задать порядок перехода, находятся в рамке, выбрать рамку.
3. Выбрать команду **Вид\Последовательность** перехода. Появится диалоговое окно **Последовательность перехода** (рис. 8).

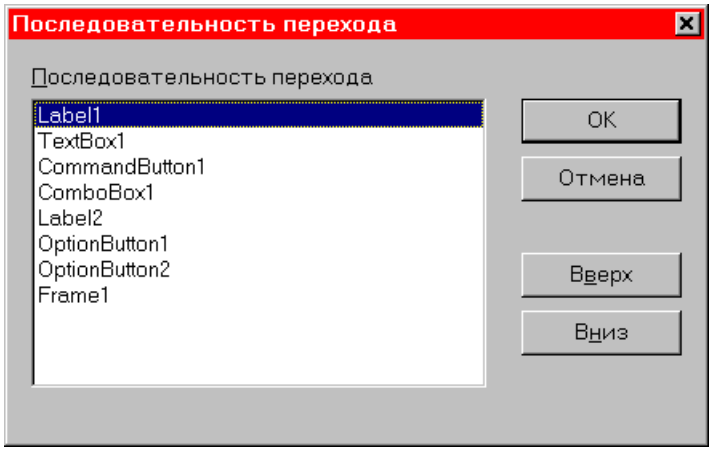

Рис. 8.

4. Для задания порядка перехода расставить требуемым образом названия элементов управления в списке, выделив имя объекта и нажав кнопку **Вверх** или **Вниз**.

5. Нажать кнопку **OK** для сохранения последовательности перехода.

6. Убедиться в правильности заданной последовательности перехода, нажимая клавишу <Tab>.

6.3. Создание процедур для элементов управления в форме.

После размещения, выравнивания и задания размеров элементов управления, определения порядка перехода между ними форма может выглядеть достаточно удачно. Однако все это не имеет значения, если с элементами управления не связано никакого кода.

Примеры ситуаций, в которых обязательно требуется создать код для объекта на форме:

- Запуск выполнения определенной задачи при нажатии кнопки **OK**.
- Закрытие диалогового окна при нажатии кнопки **Отмена**.
- Запоминание выделенного элемента в списке.
- Отмена доступа к элементу управления, например, если выбран переключатель, который запрещает изменение значения в этом элементе.

Для создания процедуры обработки события, связанного с элементом управления на форме:

1. Дважды щелкнуть по элементу управления на форме. Появится окно модуля для выбранного объекта.

2. Выбрать событие, для которого требуется создать процедуру обработки, в списке в верхнем правом углу окна модуля (рис.9).

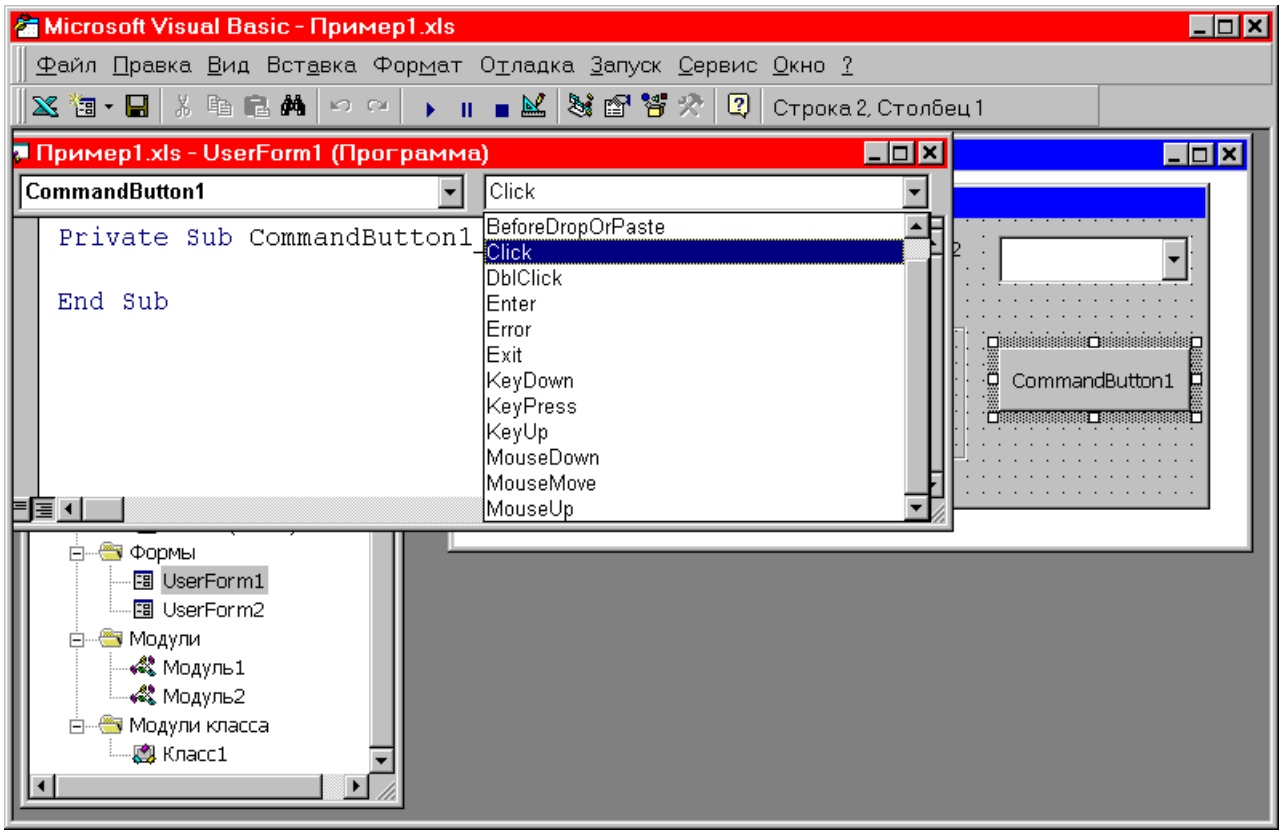

Рис. 9.

3. Ввести текст процедуры обработки в шаблоне процедуры для данного события, который формируется автоматически сразу после выбора события из списка.

4. Если требуется выполнить аналогичную операцию с другим объектом формы, выбрать этот объект управления из списка в верхнем левом углу окна модуля.

7. Импорт и экспорт файлов.

Редактор VBA обеспечивает возможность импорта и экспорта компонентов приложения. Это дает возможность использовать имеющиеся данные в других приложениях и проектах Visual Basic.

Если имеется компонент проекта VBA, который необходимо включить в текущий проект, то следует сначала экспортировать компонент из исходного проекта, а затем импортировать его в текущий проект. Компоненты приложения Visual Basic сохраняются при экспорте в файлах, имеющих стандартные расширения: frm – для формы; cls – для модуля класса; bas – для модуля программы.

Чтобы импортировать файл:

1. Выбрать в окне редактора VBA команду **Файл\Импорт файла**, либо в контекстном меню окна проекта отработать команду **Импорт файла.** Выводится диалоговое окно **Импорт файла** (рис.10).

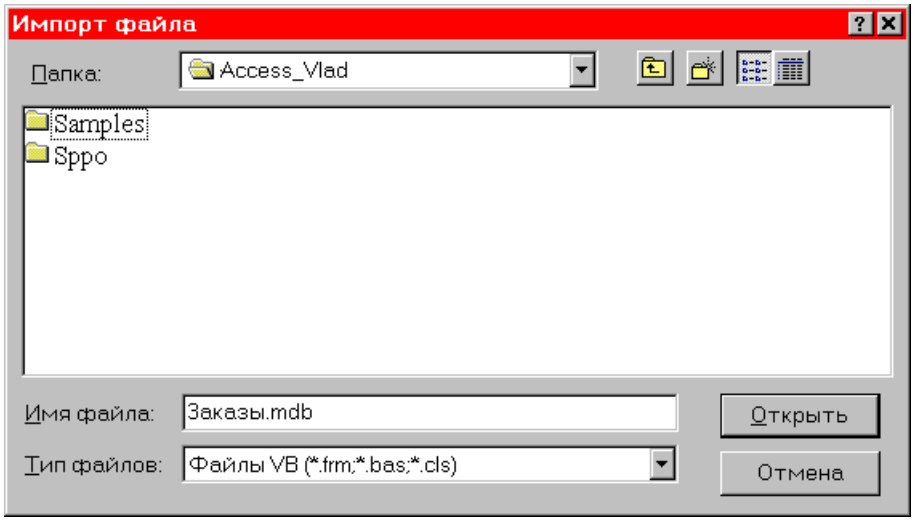

Рис.10.

2. Найти требуемый для импорта файл, выделить его, нажать **OK**.

Поскольку каждый компонент проекта VBA хранится после своего создания в основном приложении, то необходимо выполнить экспорт формы или модуля в соответствующий файл, чтобы использовать их в других проектах или приложениях. Чтобы экспортировать компонент проекта:

1. Выбрать экспортируемый компонент в окне проекта.

2. Выбрать команду **Файл\Экспорт файла**, либо в контекстном меню окна проекта отработать команду **Экспорт файла.** Выводится диалоговое окно **Экспорт файла** (рис.11).

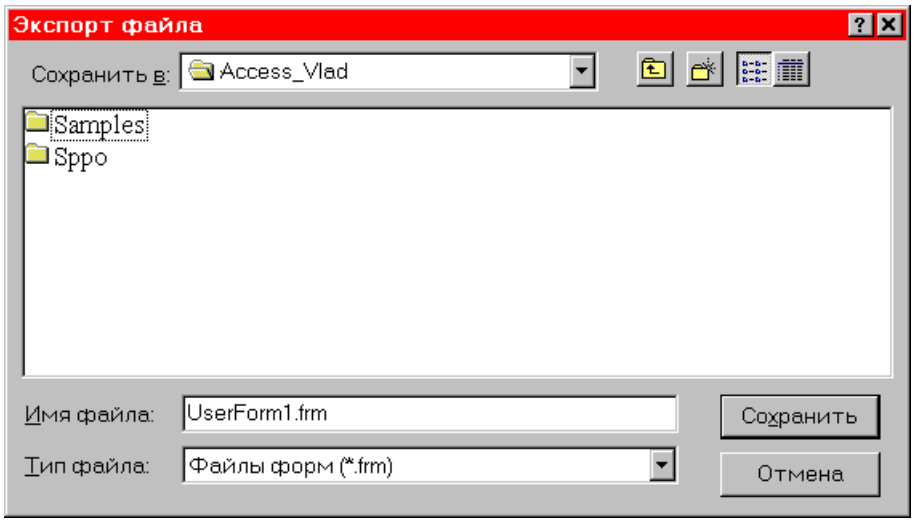

Рис.11.

3. Раскрыть требуемую папку, ввести имя файла, а затем нажать **OK**.

8. Переключение в основное приложение.

В процессе разработки проекта часто приходится переключаться в основное приложение, не закрывая проект. Для выполнения данной операции можно использовать следующие альтернативы:

- В окне проекта щелкните правой кнопкой мыши по объекту, который необходимо просмотреть, например, по листу книги Excel, а затем отработайте команду **Объект** появившегося контекстного меню.
- Отработайте команду **Вид\Объект** в меню редактора VBA.
- Отработайте в меню **Вид** редактора VBA последнюю команду, названием которой обычно является имя основного приложения.
- Выберите на панели задач Windows основное приложение.

После переключения в основное приложение может понадобиться возврат в редактор VBA. Для этого:

- Отработайте команду **Сервис\Макрос\Редактор Visual Basic**.
- Выберите на панели задач Windows приложение Microsoft Visual Basic.

9. Запуск форм, процедур и макросов. Сохранение изменений.

До получения окончательной версии разрабатываемого приложения потребуется в процессе отладки неоднократно запускать на выполнение формы, процедуры и макросы, находясь в среде разработки – в среде редактора VBA.
Для запуска:

- **Процедуры**. Установить курсор в любом месте процедуры, а затем отработать команду **Запуск\Запуск подпрограммы/формы** или нажать клавишу <F5>.
- **Макросы**. Для запуска макроса из окна редактора VBA выбрать команду **Сервис\Макросы**, далее указать имя требуемого макроса и нажать кнопку **Выполнить**.

• **Формы**. Активизировать форму (сделать окно формы активным) или любой ее объект, а затем отработать команду **Запуск\Запуск подпрограммы/формы** или нажать клавишу <F5>.

Код, формы и модули, созданные с помощью VBA, связаны с основным приложением и с документом этого приложения. Поэтому при сохранении проектаVBA записывается также документ основного приложения.

Для сохранения документа в редакторе VBA, выберите команду **Файл\Сохранить**. При этом записываются изменения, внесенные как в документ, так и в проект VBA.

10. Настройка инструментальной среды VBA.

С помощью диалогового окна **Параметры** можно установить параметры работы редактора VBA. Для вывода окна **Параметры** отработайте команду **Сервис\Параметры** (рис.12).

На вкладке **Редактор** произведите настройку параметров редактора, если настройка по умолчанию не удовлетворяет.

На вкладке **Формат** можно настроить параметры текста, что позволяет выводить фрагменты кода различным цветом, различным типом шрифта.

На вкладке **Общие** находятся элементы управления настройкой сетки, а также настройкой режимов обработки ошибок и компиляции.

На вкладке **Закрепление** можно установить фиксированное расположение отдельных окон в редакторе.

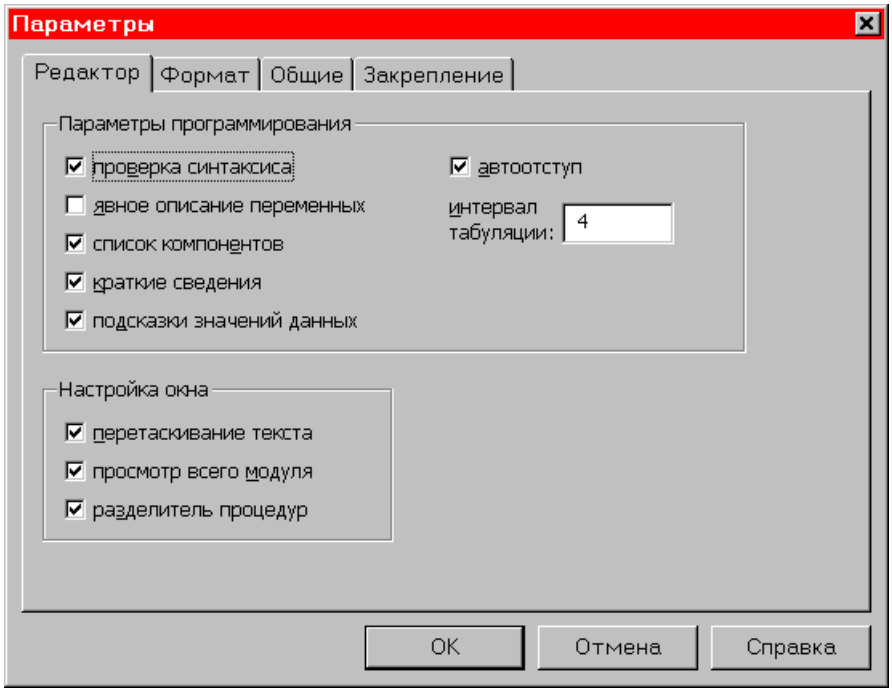

Рис. 12.

С помощью диалогового окна **VBAProject – свойства проекта** задаются свойства проекта. Войти в это диалоговое окно можно, отработав команду **Сервис\Свойства VBAProject** (рис.13).

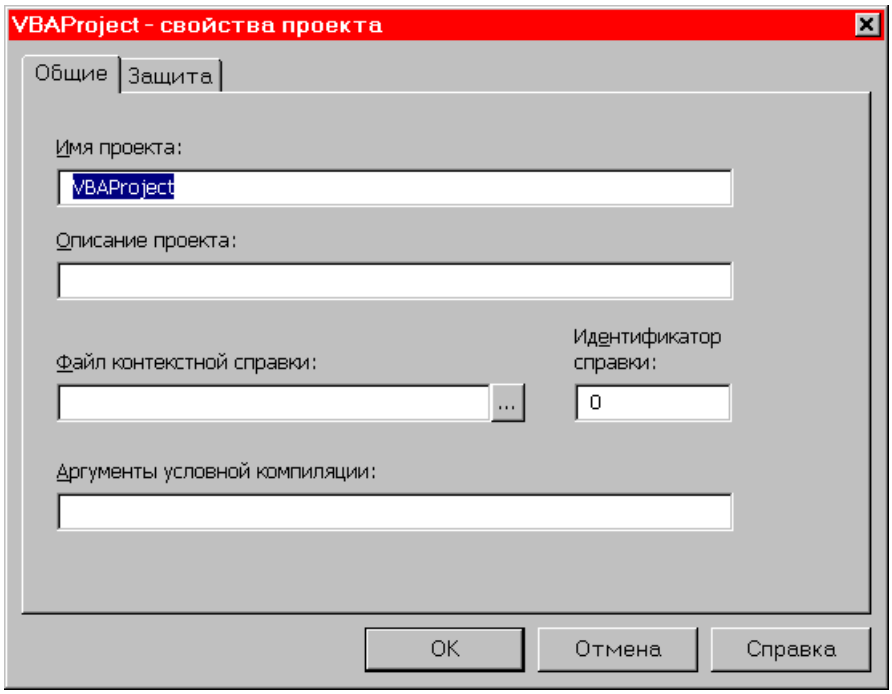

Рис. 13.

Здесь наряду с общими характеристиками проекта можно установить параметры доступа к проекту из основного приложения. Защита основана на необходимости введения пароля. Если параметр **Блокировать проект для просмотра** установлен, то пользователь должен ввести правильный пароль, чтобы просмотреть или изменить проект.

11. Знакомство с инструментальной средой Visual Studio .NET

Имена нынешнего поколения продуктов от Microsoft сопровождаются окончанием .Net (читается Dot Net), отражающим реалии современного глобального информационного мира, в котором **коммуникативная** составляющая любых программных продуктов начинает играть определяющую роль.

Инструментальная среда **Visual Studio .NET** базируется на разработанной корпорацией Microsoft платформе **.NET Framework** и представляет собой универсальный инструмент, с помощью которого можно создавать самые разнообразные приложения:

- Консольные приложения, позволяющие выполнять вывод на "консоль", то есть в окно командного процессора;
- Windows-приложения, использующие интерфейс Windows, включая формы с элементами управления на них;
- Web-приложения, представляющие собой web-страницы, которые могут просматриваться любым web-браузером;
- Web-службы (сервисы), представляющие собой распределенные приложения, которые позволяют обмениваться по Интернету

практически любыми данными с использованием единого синтаксиса XML, независимо от того, какой язык программирования применялся при создании web-службы и на какой системе она размещена.
Такие приложения могут создаваться с использованием **разных** языков программирования, известных как языки группы .NET, в частности, поставляемых Microsoft вместе с Visual Studio .NET:

- Visual Basic .NET;
- Visual C# .NET (читается Си-шарп);
- Visual J# .NET (читается Джей-шарп);
- Visual C++ .NET;
- ASP.NET (для создания web-приложений).

Для работы с этими языками Visual Studio .NET предоставляет один и тот же **интерфейс IDE** (Integrated Development Environment) - интегрированную среду разработки. Здесь слово «интегрированная» означает, что в составе среды имеются все инструменты, необходимые для написания, редактирования, компиляции, отладки и запуска программ.

В программных продуктах **.NET** за этим именем стоит вполне конкретное содержание, которое предполагает, в частности, наличие открытых стандартов коммуникации, переход от создания монолитных приложений к созданию компонентов, допускающих повторное использование в разных средах и приложениях. Возможность повторного использования уже созданных компонентов и легкость расширения их функциональности - все это непременные атрибуты новых технологий. Важную роль в этих технологиях играет язык XML, ставший стандартом обмена сообщениями в сетях.

Сущность разработанных Майкрософт технологий поясняет схема (рис.14). Кратко рассмотрим архитектуру .NET Framework, а также место инструментальной среды Visual Studio .NET и разных систем программирования в этой архитектуре.
Основным элементом .NET Framework является **общеязыковая среда выполнения приложений** (Common Language Runtime, или **CLR**), обеспечивающая выполнение следующих процессов:

- компиляцию кода по мере вызова тех или иных компонентов программы;
- распределение памяти для кэширования откомпилированного кода и размещения данных;
- управление потоками вычислений и удаленное взаимодействие программ;
- строгую проверку типов данных и другие виды проверки точности кода, что гарантирует безопасность и надежность выполнения программ.

Таким образом, основным принципом работы CLR является управление программным кодом. Именно поэтому код, который выполняется в .NET Framework, называют ***управляемым*** кодом (managed code), а код, который выполняется на компьютере, минуя CLR, - ***неуправляемым*** (unmanaged). Более того, данные, с которыми работает управляемый код,

также находятся под полным контролем CLR и поэтому называются ***управляемыми данными*** (managed data).

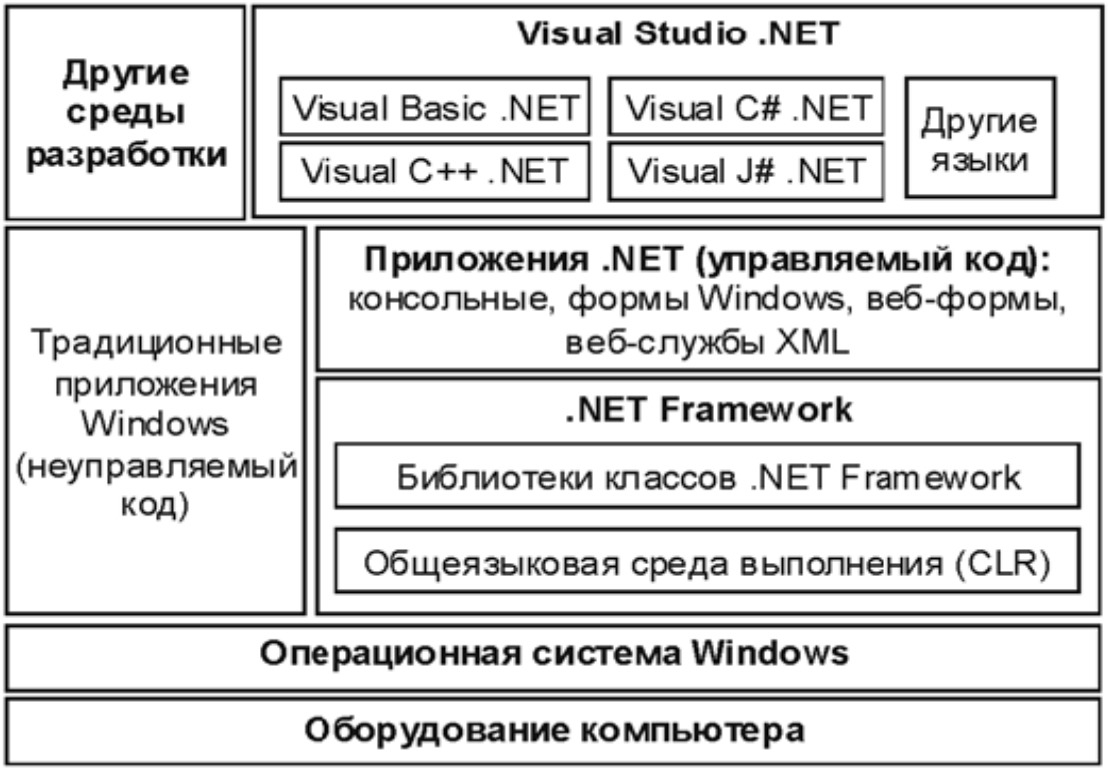

Рис14. Visual Studio .NET. и .NET Framework.

Другой основной компонент .NET Framework - общая для всех языков программирования **библиотека классов -** FCL (Framework Class Library). Ее наличие позволяет разработчикам использовать единую систему типов данных и вызываемых функций (точнее, программных объектов, их свойств и методов). Соответственно, большая часть функциональности программы, которая ранее реализовывалась за счет функций и процедур конкретного языка программирования, теперь обеспечивается использованием библиотеки классов. Например, чтобы вычислить квадратный корень в предыдущих версиях Visual Basic, программисту нужно было воспользоваться конструкцией вида:

X=SQR (y)

В Visual Basic .NET аналогичный оператор будет выглядеть иначе:

X=System.Math.Sqrt (y)

В этой конструкции уже будет задействован один из стандартных классов библиотеки .NET Framework, относящийся к так называемому ***пространству имен*** (namespace) System.Math. В .NET Framework пространством имен называется некоторая область, в которой определены названия, свойства и методы используемых классов (как системных, так и создаваемых разработчиком программы).

Существенным преимуществом данного подхода является возможность использовать **одни и те же** классы в программах**,** написанных на **разных** языках программирования: и на тех, которые разработаны корпорацией Майкрософт, и на языках **сторонних производителей**. Более того, разработчики могут создавать свои собственные библиотеки классов и использовать их в дальнейшей работе. А межъязыковое взаимодействие и общая среда разработки позволяют, например, в Visual Basic .NET-приложениях использовать компоненты, написанные на других языках .NET

Как видно из схемы, Visual Studio .NET - это **единая** среда разработки приложений, как традиционных, так и работающих в среде выполнения .NET Framework. И если первые весьма жестко привязаны к особенностям интерфейса прикладного программирования в операционной системе Windows (Win32 API), то для вторых CLR «экранирует» эти особенности. Такой подход делает написанные для общеязыковой среды программы легко переносимыми на компьютеры, работающие не под управлением Windows. Примерами такого переноса являются **проект Portable .NET** для Linux и других Unix-подобных систем или **проект Rotor** для Free BSD и Mac OS X.

Таким образом, совместное использование Visual Studio .NET и .NET Framework предоставляет в распоряжение разработчиков один из самых мощных на сегодняшний день инструментов для создания приложений.

Система объектно-ориентированного программирования Visual Basic .NET является составной частью единой среды разработки приложений Visual Studio .NET и в полной мере реализует объектный стиль разработки программных проектов. Для программиста, владеющего основами этого стиля, не столь важно, на каком **конкретном языке** программирования или в какой среде ему необходимо разработать тот или иной программный проект – на любом языке он будет создавать программный продукт требуемого качества. Тем не менее, у каждого программиста есть свои предпочтения, свой любимый язык и среда разработки.

Вашей задачей теперь, в рамках данной лабораторной работы, является знакомство с интерфейсом IDE и приобретение практических навыков работы в Visual Studio .NET, ориентируясь, в определенном смысле, на методику, изложенную выше для среды IDE VBA.

Для самостоятельного изучения можно использовать любой, доступный для Вас, выпуск Visual Studio .NET из линейки программных продуктов корпорации Microsoft: Visual Studio 2003, 2005, 2008, 2010.

Некоторые компоненты Visual Studio NET в варианте Express Edition (например, Visual Basic 2005 Express Edition) сейчас распространяются Майкрософт бесплатно; их дистрибутивы доступны для «скачивания» с сайтов корпорации:

http://www.microsoft.com/rus/express/
http://www.microsoft.com/ru/ru/default.aspx
http://msdn.microsoft.com/ru-ru/vstudio/default.aspx

Следует отметить, что в различных выпусках Visual Studio .NET различаются как среда **разработки**, так и среда **выполнения** приложений (версия 2003 базируется на .NET Framework 1.1, версия 2005 - на .NET Framework 2.0 и 3.0, версия 2008 - на .NET Framework 3.5, версия 2010 - на .NET Framework 4.0). Различия необходимо учитывать и в процессе разработки программ с помощью данных систем, и при использовании программ в дальнейшем.
Кроме того, в Visual Studio .NET включена электронная справочная система (так называемая библиотека **MSDN**). Русскоязычная справка размещена в Интернете по адресу: http://msdn.microsoft.com/ru-ru/library/default.aspx

Лабораторная работа 2. Разработка макросов в Excel.

Продолжительность работы – **8 часов.**
Цель работы: освоить технику разработки макросов; научиться пользоваться ими для повышения производительности приложений.

Этапы работы:
1. Повышение производительности приложений с помощью макросов.

Большинству пользователей Excel приходится часто выполнять повторяющиеся задачи – например, вводить серии заголовков в финансовых отчетах или увеличивать ширину нескольких столбцов в книге. Подобные действия отнимают время, поэтому имеет смысл записать их в виде макроса.
Макросом называется именованная последовательность команд, которая синтаксически оформляется в виде процедуры Sub.

Прежде, чем использовать макрос, следует убедиться, что в основном приложении нет встроенного средства для решения нужной задачи. Например, если часто приходится выделять полужирным начертанием текст заголовков столбцов и увеличивать в них размер шрифта, то можно записать макрос для автоматического форматирования заголовков. Но на самом деле значительно быстрее будет воспользоваться командой **Формат\Стиль**, которая накладывает на ячейки стиль заголовка и позволяет добиться того же эффекта форматирования. Другими словами, не пользуйтесь макросами, если только их применение не оправдывается сложностью записываемых команд.
Это предостережение вовсе не означает, что не следует пользоваться макросами для автоматизации работы – дело обстоит как раз наоборот. Но перед тем, как приступить к записи макроса, стоит все же ознакомиться с возможностями основного приложения в части встроенных средств автоматизации и знать, когда использовать готовое средство, а когда для максимальной эффективности следует приступить к созданию макроса.

Используемый в Excel язык макросов Visual Basic обладает достаточными возможностями для решения многих нетривиальных задач, например, для связи

с другими приложениями Windows. Однако самыми полезными макросами нередко оказываются те, которые заменяют всего четыре или пять простых команд.
Хотя макросы можно создать "из ничего" в среде Visual Basic, лучшим способом изучения макросов все же является запись и редактирование.

2. Автоматическая запись макроса.

Самый простой способ создать процедуру макроса – это записать ее. Этот метод пригоден для Excel, Word, PowerPoint. В Access макросы создаются с помощью собственного языка макросов, а не создаются автоматически. Более того, макросы в Access не являются процедурами VBA.
Прежде чем записать макрос, требуется продумать свои действия:

1. Какие условия должны выполняться при запуске макроса:
   - Должен ли быть открыт некоторый файл?
   - Должен ли курсор находиться в определенном положении или должна быть активной определенная ячейка?
   - Требуется ли включить определенный режим работы приложения?
2. Какие действия должен выполнять макрос:
   Рекомендуется перед записью потренироваться и выполнить пробный заход. Во многих случаях не играет роли, что используется при записи – мышь, клавиатура или команды меню, но в записи будет меньше ошибок, если заранее продумать последовательность своих действий.
3. Тщательно продумайте, что необходимо проделать при завершении работы макроса:
   - Требуется ли установить курсор в некоторую позицию или сделать активной некоторую ячейку?
   - Требуется ли закрыть открытый файл или вернуть к исходным любые измененные установки?

Хороший макрос характеризуется тем, что по завершении своей работы приложение имеет то же состояние, что и до его запуска, конечно, если назначение макроса не состоит в том, чтобы изменить состояние.

Процедуру автоматической записи макроса рассмотрим на основе вполне конкретного примера.
По умолчанию ячейка Excel имеет ширину в 8 символов, однако, для нескольких первых столбцов листа этого часто недостаточно, поскольку пользователи обычно вводят в них длинные строки с именами клиентов, названиями компаний или географических регионов.
Запишем макрос, который задает для первого столбца листа ширину в 25 символов, а для трех следующих столбцов – по 15 символов. Для записи макроса воспользуемся командой **Сервис\Макрос\Начать запись,** а имя макроса будет задано в окне диалога **Запись макроса.**
Создание макроса, автоматически меняющего ширину столбца, происходит следующим образом.

1. Вывести на экран лист, на котором будет записываться макрос. Он сохранится в отдельном модуле активной книги, и его можно выполнять на любом листе этой книги до тех пор, пока она открыта.
2. Выполнить команду **Сервис\Макрос\Начать запись**. Откроется диалоговое окно **Запись макроса** (рис. 1).

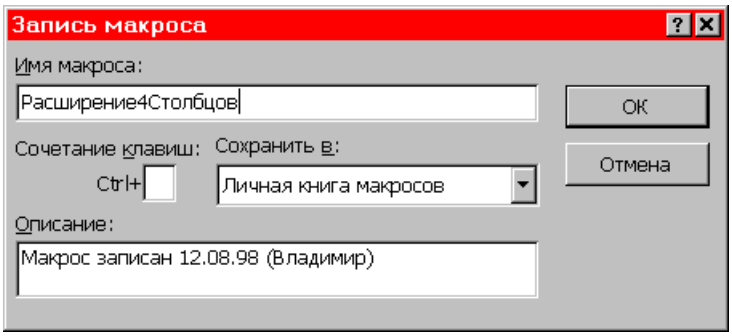

Рис. 1.

3. Ввести в поле **Имя макроса** текст *Расширение4столбцов* (или любое другое) и нажать кнопку **OK**. Позднее это имя пригодится для выполнения и редактирования макроса (имена макросов не должны содержать пробелов или знаков препинания). Excel закрывает окно диалога **Запись макроса,** помещает на отдельную панель инструментов кнопку **Остановить запись** и начинает запись макроса. С этого момента любое выделение ячеек или выполнение команды Excel будет сохранено в макросе (после завершения записи вернуться в рабочий режим можно с помощью кнопки **Остановить запись**).
4. Начать выполнение действий. Для рассматриваемого примера следует выделить первый столбец щелчком на его заголовке – букве A, а затем выполнить команду **Формат\Столбец\Ширина**. Откроется окно диалога, в котором предлагается ввести новую ширину столбца (рис. 2).

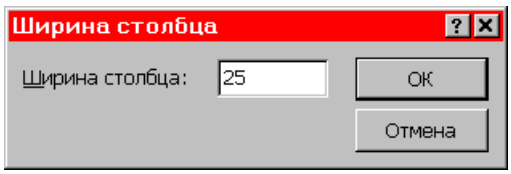

Рис. 2.

5. Ввести в текстовое поле значение 25 и нажать кнопку **OK**.
6. Выделить столбцы B, C, и D, перетаскивая указатель мыши по их заголовкам, и снова выполнить команду **Ширина,** введя в текстовое поле значение 15. Нажать копку **OK**.
7. Последним шагом при записи макроса всегда должно быть приведение экрана к состоянию, позволяющему продолжить работу с минимальными усилиями. Поэтому, выделим ячейку A1, чтобы вернуть курсор, например, в начало листа.
8. Завершить ввод макроса кнопкой **Остановить запись** на небольшой панели инструментов, появившейся во время записи макроса. Запись прекращается

и Excel сохраняет макрос в модуле. В отличие от предыдущих версий, Excel 97 не хранит макросы на отдельных листах.

9. Выполнить команду **Файл\Сохранить** для сохранения книги с макросом на диске.

3. Выполнение макроса.

Чтобы сделать автоматизацию работы более гибкой, Excel предоставляет пользователю три способа выполнения макроса.

- Двойной щелчок на имени макроса в окне диалога **Макрос**.
- Нажатие сочетания клавиш (если оно было присвоено макросу).
- Щелчок кнопки макроса на панели инструментов (если для него была создана кнопка).

3.1. Выполнение макроса из окна диалога Макрос.

Выполнение макроса *Расширение4столбцов* из окна диалога макрос производится в следующей последовательности.

1. Вывести на экран лист, на котором, должен выполняться макрос. Если при записи был использован *Лист1* текущей книги, следует щелкнуть на ярлычке *Лист2*, чтобы потренироваться на пустом листе.
2. Выполнить команду **Сервис\Макрос\Макросы.** Открывается окно диалога **Макрос** (рис. 3).

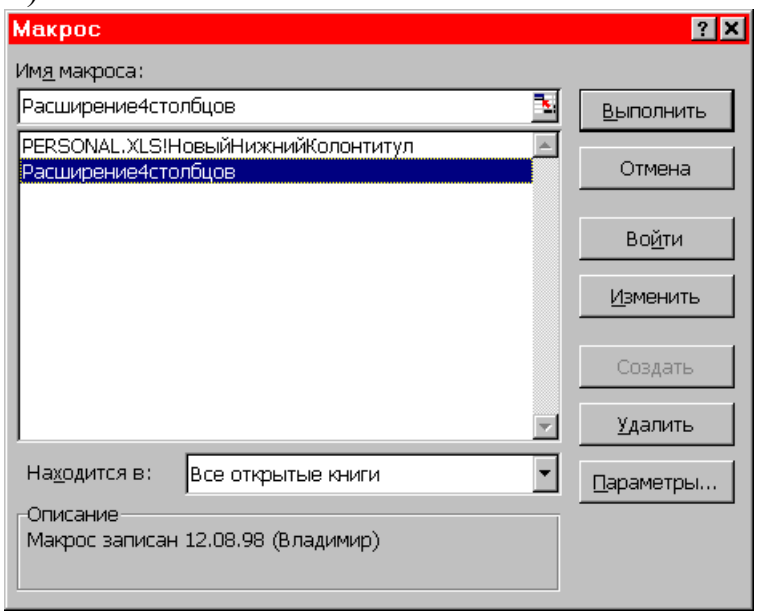

Рис. 3.

В нем перечислены все макросы текущей книги, а вместе с ними – макросы из других открытых книг или из личной книги макросов.

3. Выделить имя макроса, который требуется выполнить, нажать кнопку **Выполнить**. Excel моментально выполняет макрос и форматирует столбцы так, как необходимо (рис. 4).

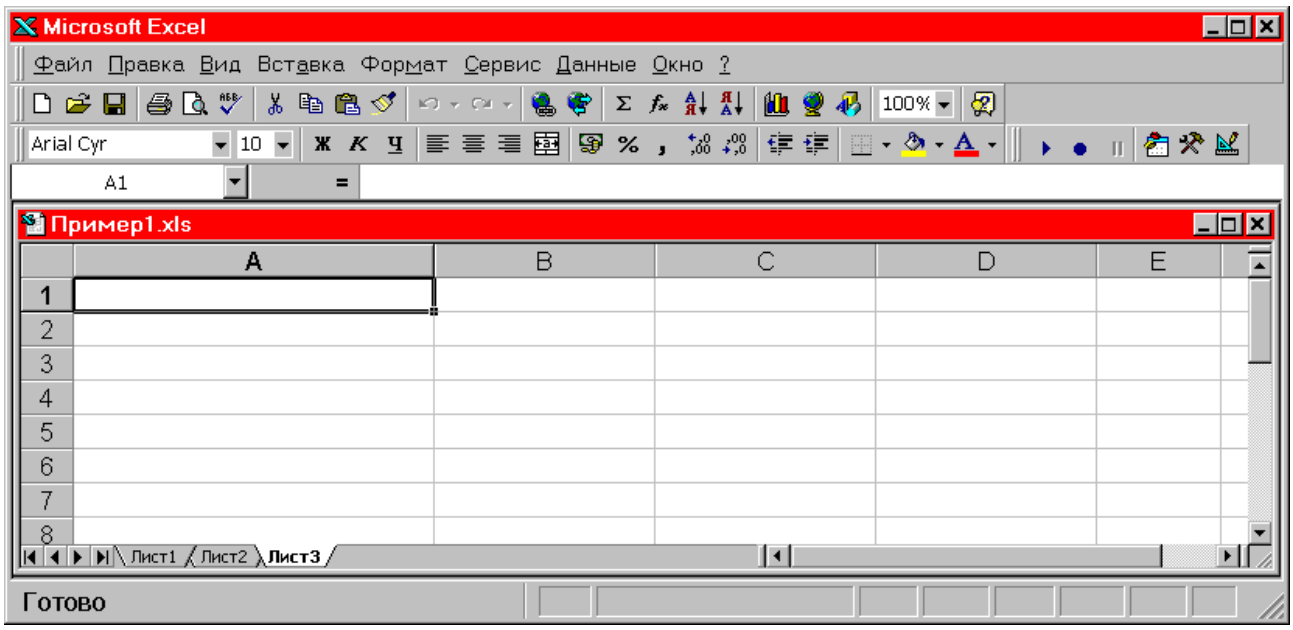

Рис. 4.

Для повторного выполнения макроса на другом листе текущей книги, следует открыть лист и повторить шаги 2 и 3.

Excel не позволяет отменить результаты выполнения макроса командой **Отменить (Undo),** поэтому пред тем, как вносить в лист изменения, следует убедиться в правильности выбора. В качестве меры безопасности следует сохранять книгу **перед** выполнением незнакомых макросов. Тогда можно не сохранять книгу с изменениями, внесенными макросом и которые не удовлетворяют Вас, а открыть прежний вариант.

3.2. Назначение макросу сочетания клавиш.

Макросу можно назначить определенное сочетание клавиш, нажатие которых осуществляет вызов макроса. Для того, чтобы назначить макросу *Расширение4столбцов* сочетание клавиш **Ctrl+Shift+W**, следует произвести следующие действия.

1. Выполнить команду **Сервис\Макрос\Макросы**. Откроется окно диалога **Макрос**.
2. Выбрать из списка макрос *Расширение4столбцов* и нажать кнопку **Параметры**, чтобы перейти к настройке параметров макроса. Откроется окно диалога **Параметры макроса** (рис. 5).
3. Удерживая нажатой клавишу **Shift**, нажать **W**. Название сочетания клавиш заменяется на **Ctrl+Shift+W** – именно оно будет использоваться для запуска макроса на выполнение.

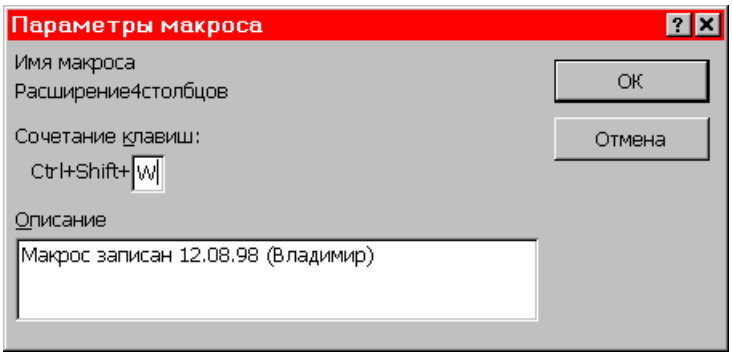

Рис. 5.

Здесь отметим, что при вводе в поле "Ctrl+" буквы нижнего регистра, Excel добавит к ней только префикс Ctrl. Поскольку сочетание клавиши Ctrl с буквой уже используется в некоторых командах Excel (например, Ctrl+W закрывает окно с текущей книгой), то для избежания конфликтов следует всегда пользоваться сочетанием Ctrl+Shift+буква.

4. Нажать кнопку **OK**, чтобы присвоить сочетание клавиш макросу, и закрыть окно диалога Макрос.

5. Воспользуйтесь новой возможностью запуска макроса – с помощью сочетания клавиш. Щелкните на ярлычке *Лист3* текущей книги, и нажмите клавиши **Ctrl+Shift+W**.

4. Редактирование макроса

По умолчанию Excel сохраняет макросы в модуле с именем *Модуль1*. Его можно вывести на экран, чтобы просмотреть команды языка Visual Basic, из которых состоит макрос, снабдить его комментариями относительно использования или отредактировать для изменения поведения или повышения эффективности. Для редактирования макроса:

1. Выполнить команду **Сервис\Макрос\Макросы**. Откроется окно диалога **Макрос**.

2. Нажать кнопку **Изменить**, чтобы открыть макрос в редакторе VBA.

3. Макросы Excel хранятся, как уже говорилось, в виде подпрограмм Sub в модуле. По умолчанию комментарии выводятся зелеными буквами, ключевые слова Visual Basic – синими, а все прочие команды и величины – черными.

4. Установите курсор в необходимую позицию и выполните обычное редактирование кода в окне модуля (см. п. 4.1 лабораторной работы 1).

5. Выполнить команду Сохранить, чтобы записать изменения в коде макроса на диск. Редактирование макроса закончено.

6. Выполнить команду **Файл\Закрыть и вернуться в Microsoft Excel**. Работа редактора VBA завершается и на экране снова появляется рабочая область Excel.

7. Проверьте работу отредактированного макроса. Макрос должен работать в соответствии с изменениями. Если Excel выводит сообщение об ошибке или результаты выполнения оказываются неожиданными, снова перейдите в редактор VBA и тщательно проверьте свою работу.

Чтобы удалить макрос из книги, выполните команду **Сервис\Макрос\Макросы**, выберите его из списка и нажмите кнопку **Удалить**.

5. Использование личной книги макросов.

Теперь, когда мы обзавелись кое-каким опытом работы с макросами, давайте изучим последний пример, в котором команда **Параметры страницы** используется для настройки верхних и нижних колонтитулов листа перед печатью. Макрос записывается с помощью приемов, рассмотренных выше, но на этот раз он будет сохранен в личной книге макросов, чтобы им можно было пользоваться во всех книгах Excel.

Личная книга макросов хранится в папке XLStart и автоматически загружается при запуске Excel. Обычно она используется для хранения макросов, но при желании в нее можно заносить и обычные листы. Поскольку для записи или выполнения макросов видеть ее необязательно, она остается скрытой, если только не открыть ее командой **Окно\Показать**. Личная книга макросов не существует до момента сохранения в ней первого макроса.

Стандартные верхние колонтитулы Excel содержат имя листа, а нижние – текущий номер страницы. Запишем макрос, который удаляет верхний колонтитул и создает нестандартный нижний колонтитул со словами *Конфиденциально*, текущей датой и номером страницы. Тогда, каждый раз, когда потребуется распечатать лист с указанием таких параметров, достаточно выполнить макрос.

Так как макрос будет доступен во всех книгах, его запись можно произвести на произвольном листе.

Для создания макроса в личной книге:

1. Выполнить команду **Сервис\Макрос\Начать запись**.
2. Ввести в качестве имени макроса строку *НовыйНижнийКолонтитул* или другое значение по усмотрению (без пробелов и знаков препинания).
3. Выбрать из списка **Сохранить в** значение *Личная книга макросов*
4. Начать запись макроса кнопкой **ОК**. Excel закрывает окно диалога, отображает панель инструментов и приступает к записи макроса. Все команды, отдаваемые с настоящего момента, будут записаны в макросе и сохранены в личной книге.
5. Выполнить команду **Файл\Параметры страницы** и перейти на вкладку **Колонтитулы**. Откроется окно диалога (рис. 6.), в котором изображаются текущие колонтитулы.
6. Щелкнуть на раскрывающемся списке **Верхний колонтитул**, прокрутить его в начало и выбрать значение *Нет*. Excel удалит верхний колонтитул из текущего листа.
7. Щелкнуть на раскрывающемся списке **Нижний колонтитул**, прокрутить его ближе к концу и выбрать строку формата *Конфиденциально*, *Дата*, *Страница 1*, где *Дата* будет совпадать с текущей датой. Эта строка определяет формат

нового колонтитула, который будет печататься на листах. Если Вы предпочитаете другой вариант, выберите его.

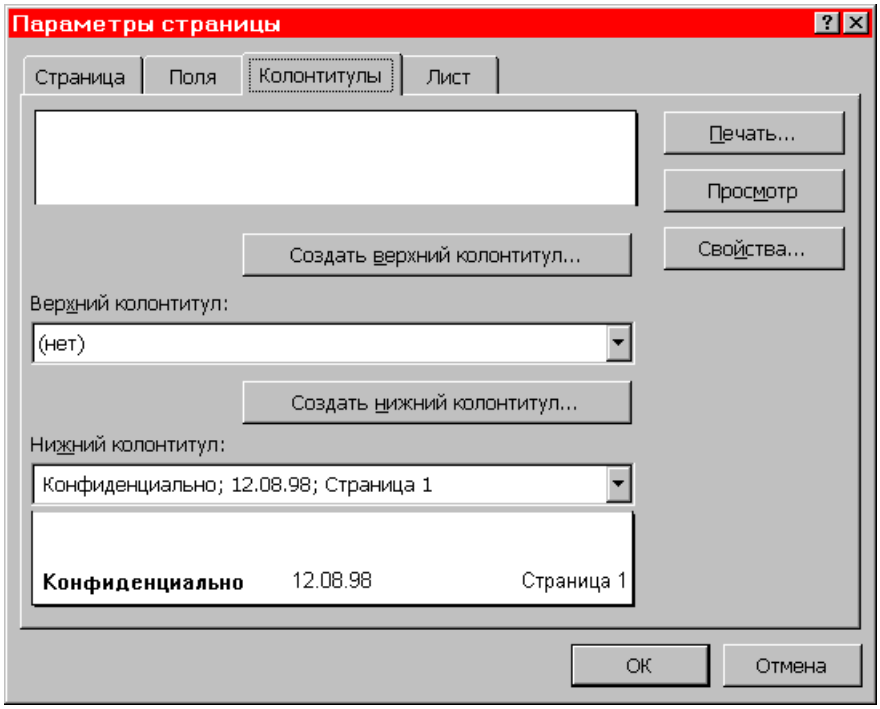

Рис. 6.

8. Нажать кнопку **ОК** для подтверждения остальных параметров окна **Параметры страницы** и затем щелкнуть кнопку **Остановить запись** для прекращения записи макроса и сохранения его в личной книге макросов.

Для проверки работы макроса *НовыйНижнийКолонтитул*:

1. Вывести на экран лист, на котором должен выполняться макрос. Не пользуйтесь листом, на котором макрос был записан, поскольку в нем уже записан нестандартный нижний колонтитул. Можно даже закрыть текущую книгу и открыть другую. Это поможет убедиться в том, что макрос сохранен не в той книге, где он создавался.

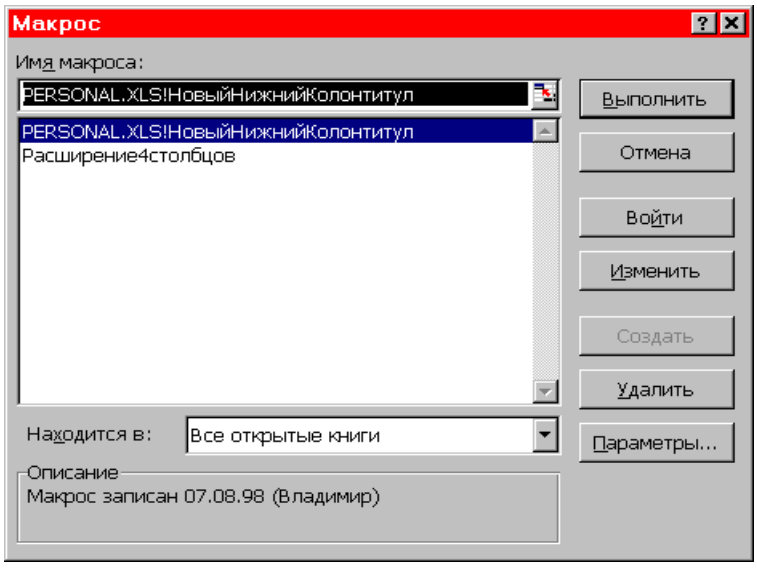

Рис.7.

2. Выполнить команду **Сервис\Макрос\Макросы**. Откроется окно диалога **Макрос**. В нем будут перечислены все макросы текущей книги, а вместе с ними - макросы из других открытых книг, а также из личной книги макросов (рис. 7).

3. Выполнить двойной щелчок на имени *НовыйНижнийКолонтитул*. Во время выполнения команд макроса лист будет слегка мигать. Это происходит потому, что Excel воспроизводит все параметры страницы, а не только те, которые связаны с оформлением нижнего колонтитула. Для ускорения работы макроса можно исключить из его кода команды, связанные с переустановкой параметров, которые Excel устанавливает по умолчанию. Для этого удалите соответствующие операторы Visual Basic из кода макроса *НовыйНижнийКолонтитул* в личной книге.

4. После выполнения макроса, занести на лист пару значений и нажать кнопку **Предварительный просмотр** на стандартной панели инструментов. Excel запрещает режим предварительного просмотра, если на листе нет данных. На экране появится изображение первой страницы текущей книги .

5. Изменить масштаб изображения, если это нобходимо для рассмотрения.

6. После завершения выйти из режима предварительного просмотра кнопкой **Закрыть** и сохранить лист, если нужно. Макросом *НовыйНижнийКолонтитул* можно пользоваться во всех книгах.

Данный макрос невозможно сохранить командой **Файл\сохранить**. Вместо этого при выходе из Excel Вам будет предложено записать изменения в личной книге макросов.

6. Назначение макросу кнопки на панели инструментов.

Макросы становятся более доступными, если для их запуска имеется кнопка на панели инструментов. Для того чтобы воспользоваться этой услугой, необходимо предварительно записать макрос и присвоить ему имя. Если требуется, чтобы кнопка постоянно присутствовала на панели инструментов, макрос должен быть записан в личную книгу. Для добавления кнопки макроса на панель инструментов:

1. Записать макрос и присвоить ему имя.

2. Открыть окно диалога **Настройка** командой **Вид\Панели инструментов\Настройка**.

3. Перейти на вкладку **Команды** в окне **Настройка**. В левой части вкладки находится список с категориями команд, а справа от него отображаются кнопки, связанные с выделенной в списке категорией.

4. Выбрать в списке *Категории* значение *Макросы*. В списке *Команды* появится нестандартная кнопка (рис.8). Для рассмотренного выше макроса *НовыйНижнийКолонтитул* назначим желтую кнопку с улыбающейся рожицей.

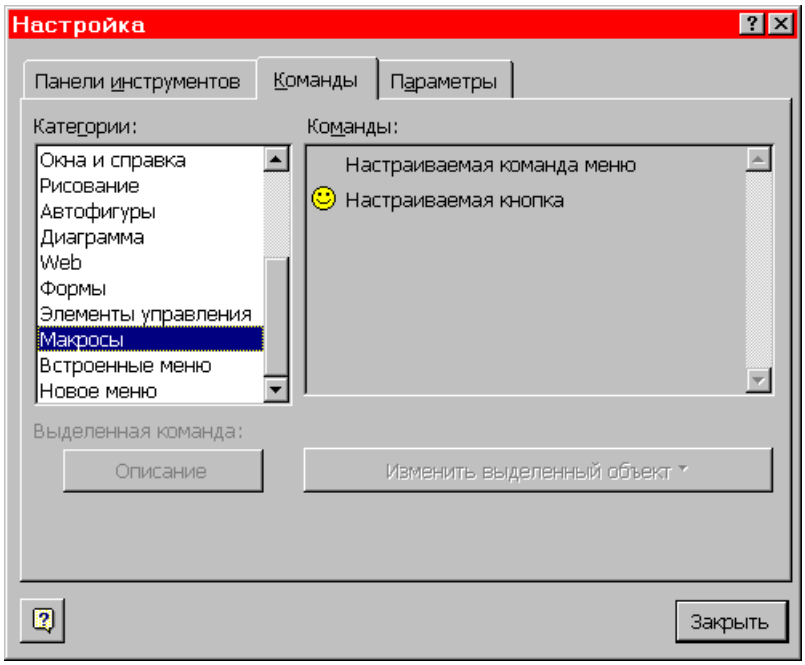

Рис.8.

5. Щелкнуть на желтой кнопке и перетащить ее на одну из панелей инструментов. Во время перетаскивания кнопки появляется вертикальная черта, которая облегчает ее позиционирование на панели.

6. Пока кнопка на панели инструментов остается выделенной, нажать в окне диалога **Настройка** кнопку **Изменить выделенный объект**.

7. Выполнить в открывшемся подменю команду **Выбрать значок для кнопки**. На экране появляется коллекция нестандартных кнопок, из которой можно выбрать любую.

8. После завершения настройки закрыть окно диалога **Настройка**.

9. Нажать созданную кнопку – откроется окно диалога **Назначить макрос** (рис.9).

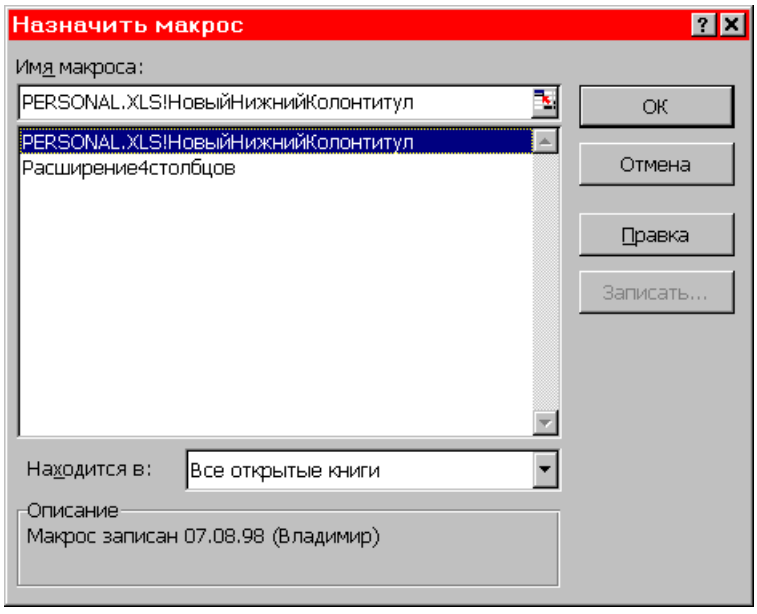

Рис. 9.

Это окно позволяет закрепить макрос за нестандартной кнопкой панели инструментов. Оно выводится при первом нажатии данной кнопки.
10. Выбрать из списка макрос, назначаемый кнопке, и щелкнуть кнопку **OK**. Окно диалога закрывается, а Excel связывает созданную кнопку с макросом.
11. Проверить работу кнопки макроса на новом листе. Если ей был присвоен макрос *НовыйНижнийКолонтитул*, перейти в режим предварительного просмотра и убедиться в правильности работы макроса, запускаемого нажатием назначенной кнопки на панели инструментов.

Сделаем одно важное замечание. Во многих случаях записанные макросы выполняют не совсем те действия, которые требуются. Основная причина заключается в том, что информация в командах макроса в точности соответствует введенной при записи. Например, в коде записанного макроса указано именно то имя файла, который был открыт при записи, либо задано значение параметра, установленное на момент записи в диалоговом окне. По этой причине часто необходимо отредактировать записанные макросы. Тем не менее, запись макроса позволяет существенно сократить время на создание приложений и является эффективным способом освоить VBA.

После ознакомления с вопросами, относящимися к технологии создания макросов в MS Excel, **Вашей задачей** теперь является:
1. Подготовка объекта автоматизации - алгоритма вычислений на таблице (одной или нескольких) MS Excel, в соответствии с вариантом заданий. Метод вычислений (соответственно, реализующий этот метод вычислительный алгоритм) с применением таблиц MS Excel выбирается самостоятельно.
2. Запись макроса, автоматизирующего работу пользователя с таблицами.
3. Редактирование процедуры макроса с целью внесения в нее изменений, которые не могут быть воспроизведены в режиме автоматической записи.
4. Разработка пользовательского интерфейса интерактивного документа Excel.
5. Оформление результатов в формате исполняемого файла – **lr2.xls** и файла-отчета по лабораторной работе – **lr2Отчет.doc**.

Варианты заданий для разработки макросов в MS Excel.

Вариант 1.
Взять матрицу M x N. Определить сумму положительных элементов каждой строки и поместить на место элементов главной диагонали. Вывести результирующую матрицу рядом с исходной. M = 5, N = 5.

Вариант 2.
Взять матрицу M x N. Определить сумму отрицательных элементов каждой строки и поместить на место первого элемента соответствующей строки.

Вывести результирующую матрицу рядом с исходной. M=4, N=6.

Вариант 3.

Взять матрицу M x N. Определить сумму четных элементов каждой строки и поместить на место последнего элемента соответствующей строки. Вывести результирующую матрицу под исходной матрицей. M=6, N=8.

Вариант 4.

Взять матрицу M x N. Определить произведение элементов кратных 3 в каждой строке и поместить на место элемента, номер которого в строке совпадает с номером строки. Вывести результирующую матрицу рядом с исходной. M=3, N=6.

Вариант 5.

Взять одномерный массив из L элементов. Отсортировать его по возрастанию. Вывести результирующий массив под исходным. L = 30.

Вариант 6.

Взять одномерный массив из L элементов. Определить количество различных элементов и поместить их в начале массива. Вывести результирующий массив под исходным. L=20.

Вариант 7.

Взять одномерный массив из L элементов. Поместить в начало массива отрицательные элементы, в середину – положительные, а в конец – нулевые. Вывести результирующий массив под исходным. L=25.

Вариант 8.

Взять матрицу M x N. Определить строку с максимальной суммой элементов и выделить ее цветом. M = 8, N = 6.

Вариант 9.

Взять матрицу M x N. Определить сумму элементов каждого столбца. Вывести результирующую матрицу–строку под исходной. M=4. N=6.

Вариант 10.

Взять матрицу M x N. Определить максимальный элемент каждой строки. Выделить их цветом. M=4, N=6.

Вариант 11.

Взять матрицу M x N. Определить минимальный элемент каждого столбца и выделить их цветом. M = 4, N = 5.

Вариант 12.

Взять матрицу M x N. Вычеркнуть строку с заданным номером. Вывести результирующую матрицу рядом с исходной. M =5, N=6.

Вариант 13.

Взять матрицу M x N. Вычеркнуть столбец с заданным номером. Вывести результирующую матрицу рядом с исходной. Вывести номер вычеркнутого столбца. M=6, N=7.

Вариант 14

Взять матрицу M x N. Определить строку с максимальной суммой положительных элементов. Выделить ее цветом. M = 6, N = 4.

Вариант 15

Взять матрицу M x N. Определить суммы элементов над и под главной диагональю. Вывести полученные суммы справа и снизу, соответственно, относительно исходной матрицы. . M = 6, N = 6.

Вариант 16

Взять матрицу M x N. Определить в каждой строке произведение элементов, кратных 4. Поместить их на место соответствующих элементов побочной диагонали. Вывести результирующую матрицу рядом с исходной. M = 5, N = 5.

Вариант 17

Взять матрицу M x N. Расставить по возрастанию элементы заданной строки. Вывести результирующую матрицу рядом с исходной. M=6, N=5.

Вариант 18

Взять матрицу M x N, Преобразовать ее в симметричную по строке M/2. Вывести результат рядом с исходной матрицей. M=6, N=7.

Вариант 19

Взять матрицу M x N. Определить максимальный элемент над главной диагональю. Выделить его цветом. M=6, N=6.

Вариант 20

Взять матрицу M x N. Обнулить элементы с четной суммой индексов. Вывести результирующую матрицу рядом с исходной. M=7, N=4.

Вариант 21

Взять матрицу M x N. Определить элементы, сумма индексов которых кратна трем. Выделить их цветом. M=4, N=7.

Вариант 22

Взять матрицу M x N. Определить сумму элементов над побочной диагональю. Результат-сумму вывести над исходной матрицей. M=8 N=8.

Вариант 23

Взять матрицу M x N. Вычеркнуть строки с отрицательной суммой элементов. Вывести результирующую матрицу рядом с исходной. M = 7, N = 5.

Вариант 24

Взять матрицу M x N. Вычеркнуть столбцы с четными номерами. Вывести результирующую матрицу рядом с исходной. M = 5, N = 8.

Вариант 25

Взять матрицу M x N. Определить количество ненулевых элементов каждого столбца. Результат-строку вывести под исходной матрицей. M=5, N=7.

Вариант 26

Взять матрицу M x N. Транспонировать. Вывести результирующую матрицу рядом с исходной. M = 7, N = 4.

Лабораторная работа 3. Работа с макросами при автоматизации процедур подготовки документов в Word.

Продолжительность работы – **4 часа**.
Цель работы: изучить особенности работы с макросами в Word; научиться пользоваться ими для повышения производительности обработки документов.

Этапы работы:

1. Автоматизация Word.

Принципиальных отличий в работе с макросами в Word, по сравнению с Excel нет. Отличия связаны, в основном, со спецификой собственно основного приложения. Как известно, Word – прежде всего мощный текстовый процессор, имеющий собственный пользовательский интерфейс. Поэтому различия относятся, по сути, к элементам интерфейса, а не к макросам, как таковым. Макрос в Word содержит последовательность действий Word и позволяет выполнить их всего одной командой, как и в Excel. Создавая макрос, Вы фактически добавляете в Word новую команду. Кроме того, пользователь может настроить по-своему усмотрению и другие элементы Word – панели инструментов, меню и сочетания клавиш, добавляя команды вызова макросов.

2. Запись макросов.

При записи макроса сохраняется последовательность действий Word. Позднее она воспроизводится выполнением процедуры, сохраненной в модуле макроса. Например, бывает нужно запомнить положение курсора в документе, чтобы в последствии вернуться к нему. Для этого приходится каждый раз выполнять команду Вставка\Закладка, указывать имя закладки и нажимать кнопку Добавить. Запись макроса может упростить подобные задачи.
В Word запись макроса производится следующим образом.
1. Выполнить команду **Сервис\Макрос\Начать запись** или сделать двойной щелчок на индикаторе **ЗАП** в строке состояния. Word откроет окно диалога **Запись макроса** (рис. 1).
2. Выполнить одно из следующих действий:

- Указать имя макроса в текстовом поле **Имя макроса**. По умолчанию Word задает что-нибудь типа *Макрос1*, но лучше указать конкретное имя.
- Дать краткую характеристику макросу в поле **Описание**. По умолчанию описание состоит только из имени автора и даты записи.
- Назначить макросу кнопку на панели инструментов или сочетание клавиш. Для этого нажмите, соответственно, кнопку **Панели** или **Клавиши**. В любом случае следуйте инструкциям в появляющихся окнах диалога. Назначение макроса какому – либо из этих элементов интерфейса упрощает выполнение макроса. Но если макрос не был связан ни с каким элементом интерфейса, его можно вызвать на выполнение из окна диалога **Макрос**.

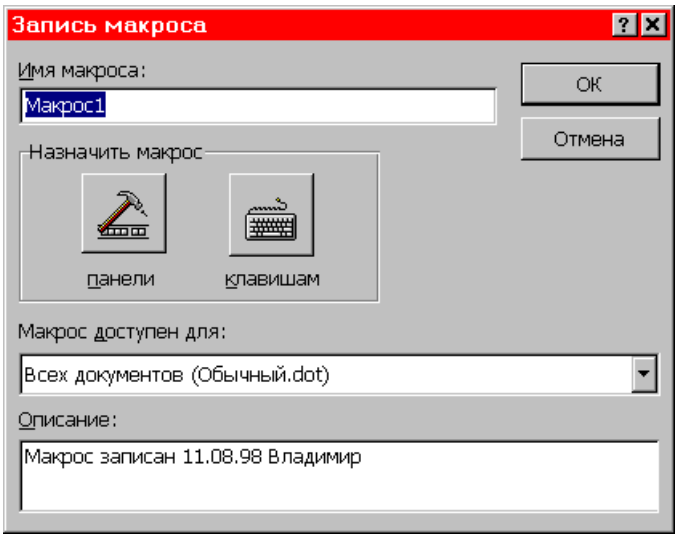

Рис. 1

По умолчанию макрос сохраняется в шаблоне *Обычный* (*Normal*) и поэтому доступен при работе с любым документом. Если к документу присоединен не шаблон *Обычный*, а какой – то другой, Вы можете выбрать для сохранения макроса имя шаблона в списке **Макрос доступен для**. В этом случае макрос будет доступен только при работе с документами, созданными на базе этого шаблона. Кроме того, можно выбрать в списке **Макрос доступен для** имя текущего документа, и тогда макрос будет доступен только при работе с этим конкретным документом.

3. Нажать кнопку **OK**, чтобы начать запись макроса. Во время записи Word выполняет следующие действия:

- Выводит рядом с указателем символ кассеты.
- Выделяет полужирным начертанием индикатор ЗАП в строке состояния.
- Отображает на экране панель инструментов записи макроса.

4. Выполнить все действия, которые должны быть зафиксированы в макросе. Следует учитывать, что в Word не запоминаются манипуляции мышью при выполнении операций выделения текста или перемещения курсора в окне документа. Поэтому, приходится выполнять способы редактирования текста с помощью клавиатуры. При необходимости временно приостановить запись макроса, используйте кнопку **Пауза** на панели инструментов записи макроса.

5. После завершения записи команд нажмите кнопку. Остановить запись на панели инструментов записи макроса или сделайте двойной щелчок мышью на индикаторе **ЗАП**. Панель инструментов записи макроса исчезнет, а записанный макрос сохранится.

3. Работа с макросами.

Для выполнения операций с макросами – редактирование описания, удаление или изменение содержимого макроса, надо проделать следующее.

1. Выполнить команду **Сервис\Макрос\Макросы** или нажать **Alt+F8**.

2. Выбрать макрос из списка **Имя**. Если макрос не появляется в списке, проследить за тем, чтобы в списке **Макросы из** было выбрано значение *Активных шаблонов*. В этом случае Word перечислит макросы из всех шаблонов, присоединенных к документу.

3. Выполнить какие – либо из следующих действий:

- Чтобы изменить описание макроса, достаточно отредактировать текст в поле **Описание** окна диалога **Макрос**.
- Чтобы удалить макрос, нажмите кнопку **Удалить**.
- Чтобы изменить содержимое макроса, нажмите кнопку **Изменить**. Word запускает редактор Visual Basic, в котором макрос представлен в форме процедуры VBA.
- Отредактируйте код макроса, создавая тем самым более сложный макрос, который нельзя получить в процессе автоматической записи.

4. Выполнение макросов.

Если макросу был назначен один из элементов интерфейса – кнопка на панели инструментов, сочетание клавиш или команда меню, то запуск макроса на выполнение сводится к воздействию на соответствующий интерфейсный элемент. Есть два способа назначить макросу любой из вышеперечисленных элементов интерфейса:

- при записи макроса - нажатием кнопки **Панели** или **Клавиши** в окне диалога **Запись макроса**, о чем уже говорилось.
- после записи макроса - командой **Сервис\Настройка**.

Здесь отметим, что изменение меню путем ввода новой команды, связанной с макросом, возможно при использовании второго способа.

Вместе с тем, любой макрос (даже если ему ничего не назначено) вызывается на выполнение следующим образом:

1. Выполнить команду **Сервис\Макрос\Макросы**. Откроется окно диалога **Макрос** (рис.2).
2. Выбрать из списка **Макросы из** либо значение *Активных шаблонов*, либо имя шаблона, в котором хранится требуемый макрос.
3. Выделить макрос в списке **Имя**. Если у него имеется описание, оно появляется в поле **Описание**.
4. Нажать кнопку **Выполнить**.

Из окна диалога **Макрос** можно выполнить любой встроенный макрос Word. Для этого следует выбрать из списка **Макросы из** значение *Команд Word,* выделить в списке **Имя** нужный макрос и нажать кнопку **Выполнить**.

5. Примеры макросов.

Познакомимся с некоторыми примерами, которые дают общее представление о задачах Word, поддающихся быстрой автоматизации.

Рассмотрим пару макросов. Первый из них позволяет запомнить текущее положение курсора (или выделенный фрагмент) с помощью сочетания клавиш **Ctrl+Shift+S**; он называется *СохранитьМесто*. Второй макрос возвращает

курсор в сохраненную позицию (или восстанавливает выделенный фрагмент) после нажатия клавиш **Ctrl+Shift+R**; макрос называется *ВернутьсяНаМесто*.

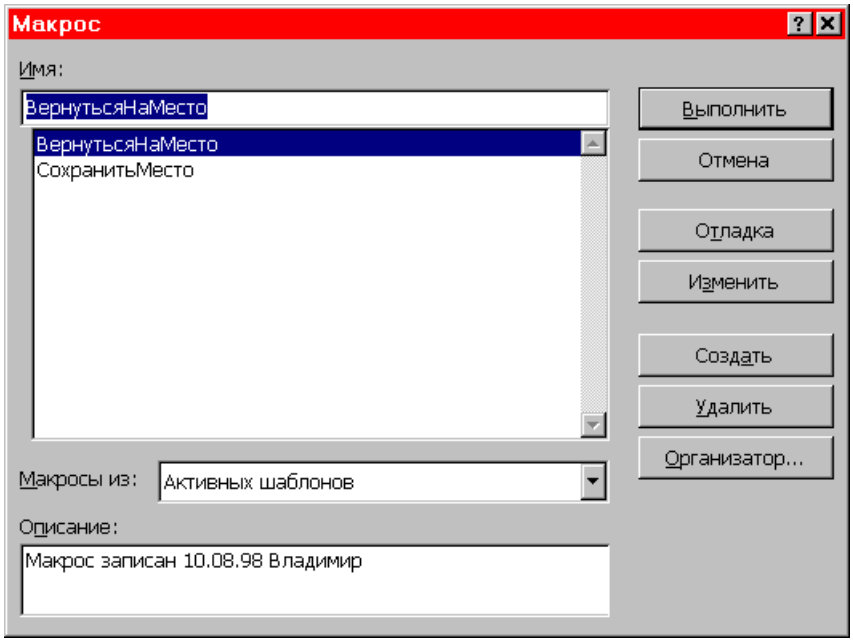

Рис. 2.

Макрос *СохранитьМесто* создается следующим образом:

1. Начать запись макроса – сделать двойной щелчок на индикаторе **ЗАП** в строке состояния и задать имя *СохранитьМесто* в окне диалога **Запись макроса**.
2. Чтобы назначить сочетание клавиш, надо открыть окно диалога **Настройка** щелчком на кнопке **Клавиши**. Убедиться, что курсор находится в текстовом поле **Новое сочетание клавиш** и нажать **Ctrl+Shift+S**. Щелкнуть кнопку **Назначить**, после чего – кнопку **Закрыть**. Word возвращается в документ; все готово к записи макроса.
3. Выполнить команду **Вставка\Закладка**.
4. Задать в поле **Имя Закладки** значение *Пометка* и нажать копку **Добавить**.
5. Остановить запись макроса кнопкой **Остановить запись** на панели инструментов записи макроса.

Макрос *ВернутьсяНаМесто* создается следующим образом:

1. Начать запись макроса – сделать двойной щелчок на индикаторе **ЗАП** в строке состояния и задать имя *ВернутьсяНаМесто* в окне диалога **Запись макроса**.
2. Чтобы назначить сочетание клавиш, надо открыть окно диалога **Настройка** щелчком на кнопке **Клавиши**. Убедиться, что курсор находится в текстовом поле **Новое сочетание клавиш** и нажать **Ctrl+Shift+R**. Щелкнуть кнопку **Назначить**, после чего – кнопку **Закрыть**. Word возвращается в документ; все готово к записи макроса.
3. Выполнить команду **Вставка\Закладка**.
4. Выбрать в списке **Имя закладки** значение *Пометка*, нажать кнопку **Перейти** и кнопку **Закрыть**.

5. Остановить запись макроса кнопкой **Остановить запись** на панели инструментов записи макроса.

Теперь можно проверить работу созданных макросов. Установите курсор в любом месте документа (или выделите фрагмент текста) и нажмите **Ctrl+Shift+S**. Word запомнит текущую позицию курсора или выделенный фрагмент. Переместите курсор в любое место того же документа и выполните действия (какие хотите) по редактированию документа. Когда вы захотите вернуться к прежнему месту, нажмите **Ctrl+Shift+R**. Word немедленно перемещает курсор в исходную позицию (или восстанавливает исходный выделенный фрагмент).

Еще несколько примеров макросов, которые могут пригодиться в работе.

- Запишите пару макросов для установки и снятия флажка **Перенос по границе окна**. Этот флажок бывает полезно установить во время редактирования в режиме электронного документа, чтобы в окне помещалось максимальное количество текста, но периодически возникает необходимость быстро снять его, чтобы увидеть истинное положение разрывов строк и шрифты в печатной копии документа.

Для создания первого макроса начните запись, выполните команду **Сервис\Параметры**, перейдите на вкладку **Вид** и установите флажок *Перенос по границе окна*. Второй макрос записывается точно также, только вместо того, чтобы установить флажок, снимите его.

- Запишите пару макросов для переключения между синим и белым фоном окна. Синий фон снижает нагрузку на зрение, но на нем бывает сложно разглядеть некоторые объекты (например, темные линии, нарисованные при помощи панели инструментов рисования).

При записи первого макроса выполните команду **Сервис\Параметры**, перейдите на вкладку **Общие** и установите флажок *Белый текст на синем фоне*. Второй макрос записывается точно также, только установка флажка заменяется его снятием.

- Запишите макрос для подготовки документа к печати. Этот макрос может пригодиться, если Вы пишите и редактируете документ с одним набором атрибутов форматирования (например, одинарными межстрочными интервалами и легко читаемым шрифтом типа Courier размером 14 пунктов), но для печати хотите пользоваться другим (к примеру, двойными межстрочными интервалами и мелким пропорциональным шрифтом типа Times New Roman размером 10 пунктов). Во время записи макроса выполните все действия, предшествующие печати документа (например, измените атрибуты формата стилей в документе). Выполняйте макрос непосредственно перед печатью.

После ознакомления с особенностями, относящимися к технологии создания макросов в MS Word, **Вашей задачей** теперь является:

1. Подготовка объекта автоматизации - алгоритма обработки данных документа MS Word, в соответствии с вариантом заданий. Метод обработки (соответственно, реализующий этот метод алгоритм обработки данных) выбирается самостоятельно.
2. Запись макроса, автоматизирующего работу пользователя по подготовке документа Word.
3. Редактирование процедуры макроса с целью внесения в нее изменений, которые не могут быть воспроизведены в режиме автоматической записи.
4. Разработка пользовательского интерфейса интерактивного документа Word.
5. Оформление результатов в формате исполняемого файла – **lr3.doc** и файла-отчета по лабораторной работе – **lr3Отчет.doc**.

Варианты заданий для разработки макросов в MS Word.

Вариант 1.
Взять текст, состоящий из M слов по N символов, разделенных пробелом. Переставить слова в тексте так, чтобы каждое следующее слово начиналось с той буквы, на которую закончилось предыдущее. Первое слово оставить на месте. $M \geq 6$, $N \geq 8$.

Вариант 2.
Взять последовательность латинских букв длиной L символов. Расставить их по алфавиту. $L \geq 18$.

Вариант 3.
Взять текст из M слов по N символов, разделенных несколькими пробелами. Сжать текст, оставив между словами по одному пробелу. $M \geq 8$, $N \geq 6$.

Вариант 4.
Взять последовательность из L символов, содержащую повторяющиеся сочетания символов "XX". Определить количество повторяющихся сочетаний "XX" и заменить их символом "!". $L \geq 30$.

Вариант 5.
Взять текст из L символов. Определить количество слов, содержащих более 4-х символов. Слова разделяются одним пробелом. $L \geq 40$.

Вариант 6.
Взять текст, состоящий из M слов по N символов. В словах с четным номером изменить порядок букв на обратный. $M \geq 20$, $N \geq 6$.

Вариант 7.
Взять строку, состоящую из L символов. Преобразовать её в последовательность символов алфавита в следующем формате: АББВВВГГГГДДДДДЕЕЕЕЕЕЕЖ… .$L \geq 16$.

Вариант 8.

Взять строку, состоящую из L символов. Построить последовательность из символов, используя таблицу, каждая запись которой содержит символ и количество его повторений. Например, А – 3 раза, Б – 5 раз, С – 6 раз: АААББББССССС. L ≥ 10.

Вариант 9.

Взять текст, состоящий из M слов, разделенных одним пробелом. Определить количество слов и количество букв в каждом слове. M ≥ 30.

Вариант 10.

Взять текст, состоящий из M слов, разделенных одним пробелом. Определить количество слов, в которых символ "е" встречается более чем 2 раза. M ≥ 30.

Вариант 11.

Взять последовательность из L символов. Определить частоту повторения каждого символа. L ≥ 30.

Вариант 12.

Взять M слов по N символов. В начале каждого слова записан номер из двух символов. Расставить слова в порядке возрастания номеров. M ≥ 20, N ≥ 5.

Вариант 13.

Взять текст длиной L символов Определить всевозможные сочетания из 2-х символов и подсчитать их количество. L ≥ 30.

Вариант 14.

Взять текст из M слов, разделенных пробелами (одним и более). Определить количество слов, содержащих более четырех символов. M ≥30.

Вариант 15.

Взять текст из M слов, разделенных пробелом. Определить количество повторений буквы Л в каждом слове. M ≥30.

Вариант 16.

Взять текст из M слов, разделенных пробелом. Расставить слова в соответствии с латинским алфавитом. M ≥30.

Вариант 17.

Взять текст из L символов. Определить количество одинаковых символов и частоту их повторений. L ≥40.

Вариант 18.

Взять текст из M слов по N символов. Расставить слова в алфавитном порядке. M ≥ 10, N ≥ 5.

Вариант 19.

Взять текст, состоящий из слов, разделенных некоторым количеством пробелов. Определить количество букв "р" в каждом слове.

Вариант 20.

Взять текст, состоящий из M слов по N символов. Определить количество гласных букв в каждом слове. M ≥ 10, N ≥5.

Вариант 21.
Взять текст, состоящий из M слов по N символов. Определить количество различных букв в каждом слове. $M \geq 20$, $N \geq 5$.

Вариант 22.
Взять текст, состоящий из M слов по N символов. Удалить слова, содержащие более двух символов «о». $M \geq 30$, $N \geq 6$.

Вариант 23.
Взять текст, состоящий из M слов по N символов. Изменить порядок букв в словах на противоположный. $M \geq 10$, $N \geq 5$.

Вариант 24.
Взять текст, состоящий из M слов по N символов. Найти слова, порядок букв в которых является обратным по отношению к первому слову. $M \geq 40$, $N \geq 5$.

Вариант 25.
Взять строку, состоящую из M слов по N символов. Осуществить кольцевой сдвиг каждого слова влево на три символа. $M \geq 10$, $N \geq 5$.

Вариант 26.
Взять текст из M слов, разделенных пробелом; определить количество слов в которых символ "a" встречается более 3-х раз. $M \geq 20$.

Лабораторная работа 4. Автоматизация обработки данных в Access.

Продолжительность работы – **8 часов**.
Цель работы: изучить особенности работы с макросами в Access; научиться пользоваться ими для повышения производительности работы с данными базы.

Этапы работы:

1. Автоматизация приложений в Access.

В Microsoft Access многие действия выполняются с помощью макросов. Вместе с тем, для решения задач автоматизации требуется программирование на языке VBA. Выбор между созданием макроса или разработкой программы VBA обычно определяется требуемыми действиями.

В каких случаях следует создавать макрос?

Макрос является удобным средством выполнения простых задач, таких как открытие и закрытие форм, вывод на экран и скрытие панелей инструментов или запуск отчетов. Действия, связывающие различные объекты базы данных, выполняются легко и просто, поскольку пользователь не должен запоминать правила синтаксиса - все аргументы, требуемые каждой макрокомандой, выводятся в нижней половине окна макроса.

В дополнение к упомянутым простым действиям, макросы необходимо использовать для выполнения следующих задач.

- Определение общих назначенных клавиш.
- Выполнение макрокоманды или набора макрокоманд при открытии базы данных. Однако определенные действия, которые должны производиться при открытии базы данных, например, открытие формы, могут быть заданы в диалоговом окне Параметры запуска.

В каких случаях следует создавать программу VBA?

Программы Visual Basic используют вместо макросов в случаях, когда необходимо:

- Упростить управление базой данных. Поскольку макросы являются объектами, существующими отдельно от использующих их форм и отчетов, поддержание базы данных, в которой реакция на события в формах и отчетах определяется многими макросами, становится достаточно затруднительным. В отличие от этого, процедуры обработки события Visual Basic являются встроенными в описания соответствующих форм и отчетов. При переносе формы или отчета из одной базы данных в другую встроенные процедуры обработки события автоматически переносятся вместе с формой или отчетом.

- Создавать собственные специализированные функции. В Microsoft Access определен ряд встроенных функций, например, функция IPmt, которая рассчитывает проценты по платежам. Пользователь имеет возможность использовать для проведения расчетов встроенные функции без необходимости разрабатывать сложные выражения. Однако язык VBA позволяет пользователям создавать собственные функции как для решения задач, выходящих за рамки возможных для встроенных функций, так и для замены сложных выражений, содержащих встроенные функции. Кроме того, создаваемые пользователем функции используются для выполнения одинаковых операций над разными объектами.
- Скрывать сообщения об ошибках. Стандартные сообщения об ошибках Microsoft Access, выводящиеся при возникновении нештатных ситуаций во время работы пользователя с базой данных, могут оказаться малопонятными для пользователя, в особенности, для не имеющего большого опыта работы с Microsoft Access. Средства VBA позволяют перехватывать ошибку при ее возникновении и либо выводить собственное сообщение об ошибке, либо предпринимать определенные действия.
- Создавать или обрабатывать объекты. В большинстве случаев удобнее создавать или изменять объекты в режиме конструктора. Однако в некоторых ситуациях приходится работать с описанием объекта в программе. Средства VBA позволяют выполнять обработку любых объектов в базе данных и самой базы данных.

• Выполнять действия на системном уровне. Выполнение в макросе макрокоманды ЗапускПриложения (RunApp) позволяет запускать из собственного приложения другое приложение, работающее в среде Windows или MS-DOS, однако, это практически все, что можно сделать вне Microsoft Access из макроса. Средства Visual Basic позволяют проверять существование файлов, использовать механизм программирования объектов или динамического обмена данными (DDE) для связи с другими приложениями, работающими под управлением Windows, например, Microsoft Excel, а также вызывать функции из библиотек динамической компоновки (DLL) Windows.

• Обрабатывать записи по одной. Инструкции VBA позволяют перебирать наборы записей по одной и выполнять определенные действия над отдельной записью. В отличие от этого, макросы позволяют работать только с целым набором записей.

• Передавать аргументы в специальные процедуры VBA. Пользователь имеет возможность задать в окне макроса значения аргументов макрокоманды при создании макроса, однако, невозможно изменить значения этих аргументов при выполнении макроса. В отличие от макросов, в программах Visual Basic допускается передача аргументов в программу при запуске программы или использование в качестве аргументов значений переменных. Поэтому использование программ VBA дает более широкие возможности работы с данными.

2. Создание макроса.

Макросом в Access называют набор из одной или нескольких макрокоманд, выполняющих определенные операции, такие как открытие форм или печать отчетов. Макросы особенно полезны при создании небольших персональных приложений или создании прототипов больших приложений. Но даже если Вы считаете, что готовы сразу перейти к VBA, лучше сначала изучить макрокоманды. Почти все из них придется использовать в VBA. Поэтому изучение макросов является прекрасным введением в программирование в Access в целом.

Подготовка базы данных. В качестве полигона для рассмотрения примера создания макроса будем использовать учебную базу данных "Студенты и занятия", которую можно создать в автоматическом режиме с помощью мастера. Для этого воспользуйтесь командой **Файл\Создать базу данных…**. В появившемся окне диалога **Создание** (рис.1) выбрать вкладку **Базы данных,** установить указатель на нужный значок шаблона базы данных и дважды щелкнуть кнопкой мыши. После этого указать имя и каталог для создаваемой базы данных. Чтобы начать создание новой базы данных, нажать кнопку **OK**.

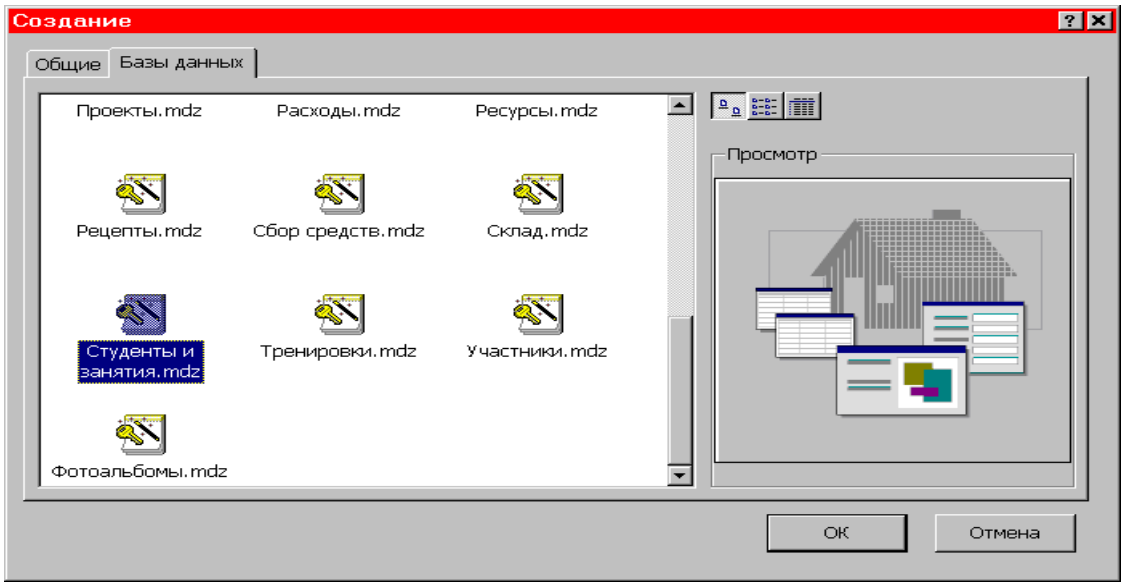

Рис.1.

После ответов на вопросы мастера в режиме диалога, создается новая база данных окно которой показано на рис.2.

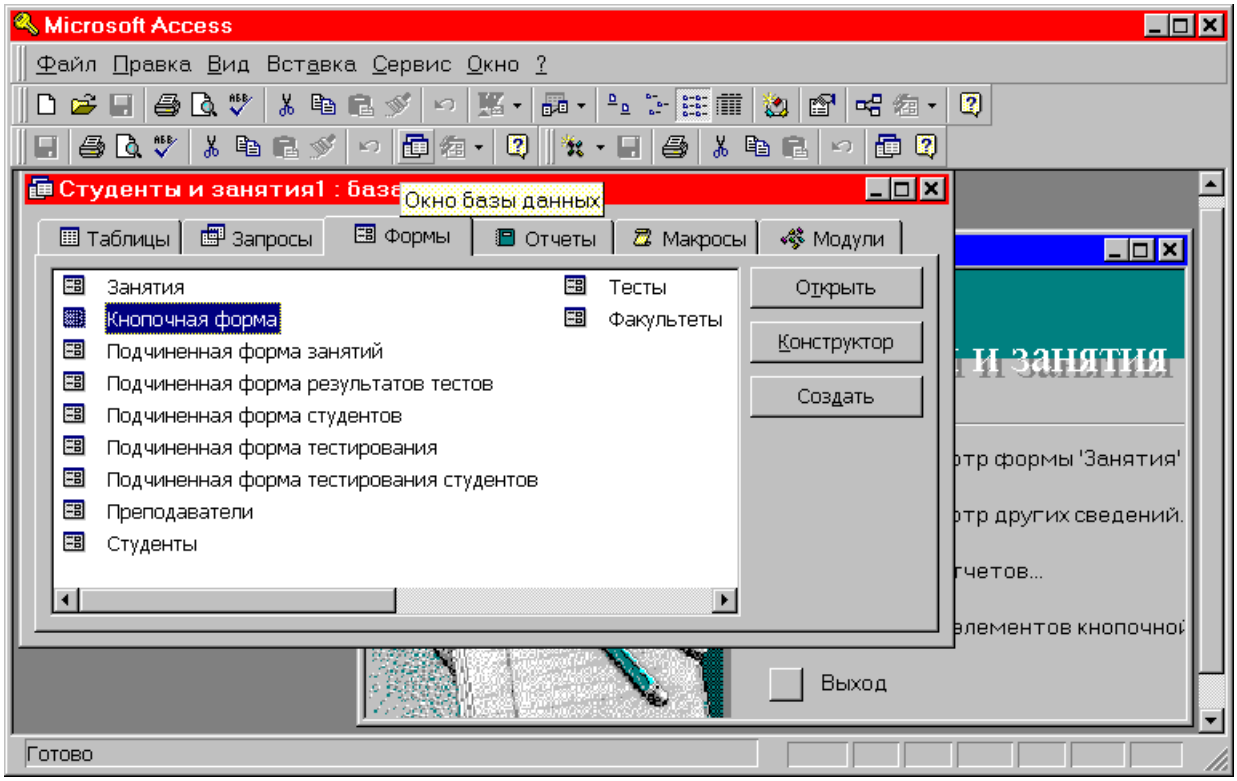

Рис.2.

Щелкнув по вкладке **Макросы** в окне базы, можно увидеть, что рассматриваемая база данных не содержит каких-либо разработанных ранее макросов.

Выберем объект, который в последующем свяжем с создаваемым макросом. Пусть это будет форма, выводящая сведения о приложении "Студенты и занятия". Сведения о приложении обычно выводятся в диалоговом окне (форме), которое может быть вызвано различными путями, в зависимости от способа реализации вызова – например, с помощью созданной на панели инструментов кнопки запуска макроса, или специальной команды меню Справка.

Таким образом, последовательно решим две возникшие задачи:

- создание свободной формы, выводящей сведения о приложении.
- создание макроса, открывающего эту форму.

Подготовка формы. Создать форму можно несколькими способами, но в данном случае лучше всего начать с кнопки **Создать** на вкладке **Формы** окна базы данных. Откроется окно диалога **Новая форма** (рис.3).

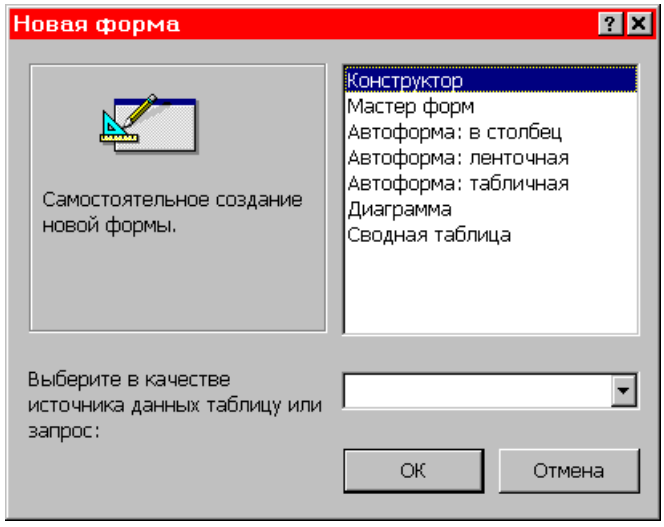

Рис.3.

Из верхнего списка в окне диалога выбрать значение *Конструктор*. Поскольку создаваемая форма не связана ни с какими другими объектами базы, то список в нижней части диалогового окна не используется. Нажать кнопку **OK**. Access откроет окно формы в режиме конструктора (рис. 4). Если панель элементов не появилась автоматически, ее можно вызвать, используя команду **Вид\Панель элементов** основного меню (альтернатива – контекстное меню окна формы). При необходимости, на экран можно вывести окно свойств формы и производить необходимые действия по модификации формы путем установки значений свойств прямо в окне свойств. Для вывода сведений о приложении в справочном окне, достаточно разместить информацию в области данных формы –области серого цвета с нанесенной сеткой, расположенную в центральной части окна конструктора формы

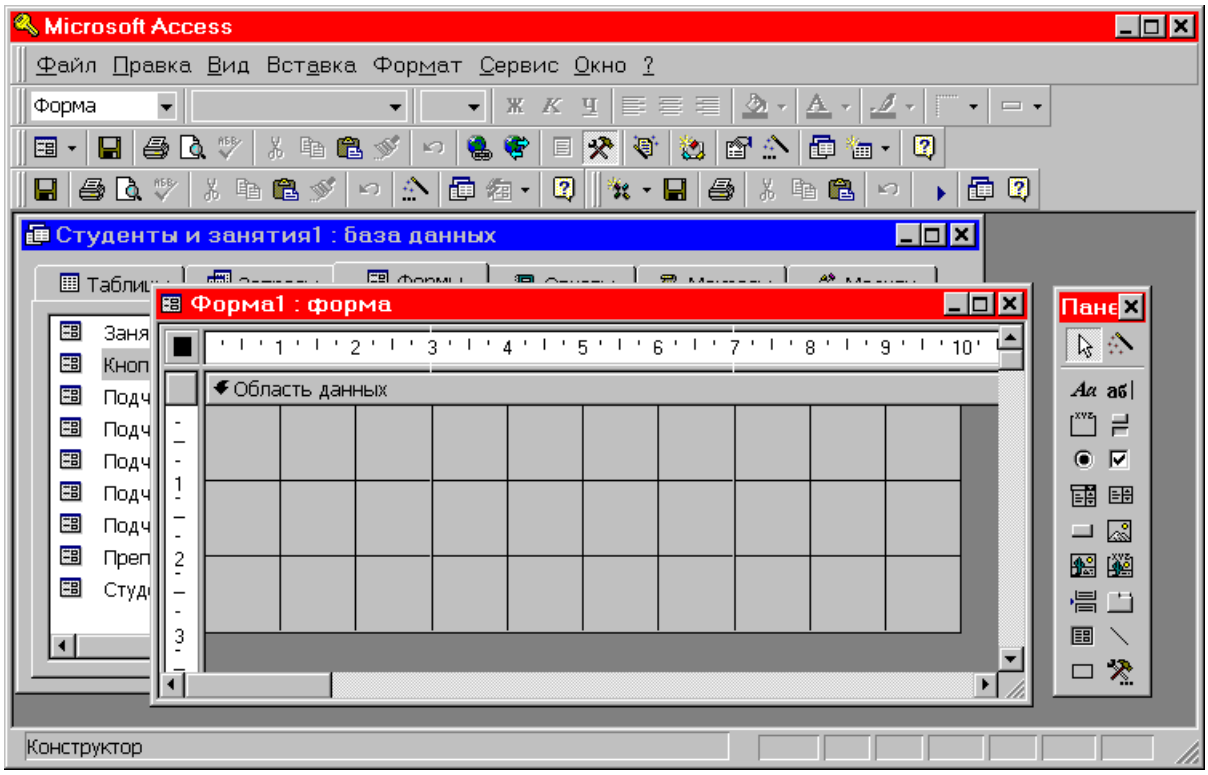

Рис.4.

Теперь можно выполнить редактирование формы: создать надписи, в которых будут отображаться название приложения, назначение приложения; создать свободную рамку объекта для вывода эмблемы (рисунка); создать кнопку для закрытия справочного окна после просмотра. В числе прочих свойств формы, в окне свойств устанавливаются следующие значения:

Свойство	**Значение**
Подпись (Caption)	О программе "Студенты и занятия"
Режим по умолчанию (DefaultView)	Простая форма
Допустимые режимы (ViewsAllowed)	Форма
Разрешить изменение (AllowEdits)	Нет
Разрешить удаление (AllowDeletions)	Нет
Разрешить добавление (AllowAdditions)	Нет
Полосы прокрутки (ScrollBars)	Отсутствуют
Область выделения (RecordSelectors)	Нет
Поле номера записи (NavigationButtons)	Нет
Выравнивание по центру (AutoCenter)	Да
Тип границы (BorderStyle)	Окно диалога
Кнопки размеров окна (MinMaxButtons)	Отсутствуют
Контекстные меню (ShortcutMenu)	Нет

При создании кнопки, закрывающей окно справки, необходимо задать значения ее свойств и определить процедуру обработки события **Нажатие кнопки (OnClick).** Для этого воспользуемся панелью элементов и окном свойств для созданного элемента **Кнопка** (CommandButton), устанавливая для него, в числе прочих, следующие свойства:

Свойство	**Значение**
Имя (Name)	OK
Подпись (Caption)	OK
Всплывающая подсказка (ControlTipText)	Закрывает окно диалога "ОПрограмме"
Нажатие кнопки (OnClick)	[процедура обработки событий]

В качестве процедуры обработки события **Нажатие кнопки** OK достаточно единственной инструкции, определяющей использование метода Close объекта DoCmd:

```
Private Sub OK_Click()
'Закрывает окно справки
   DoCmd.Close
End Sub
```

Признаком хорошего стиля программирования является включение в процедуру средств обработки возможных ошибок. Это существенно повышает работоспособность приложения в любых ситуациях. В рассматриваемом простейшем случае код с использованием обработчика ошибок может выглядеть так:

```
Private Sub OK_Click()
'Обработка ошибок
On Error GoTo Err_OK_Click

'Закрывает окно справки
DoCmd.Close

Exit_OK_Click:
Exit Sub

Err_OK_Click:
   MsgBox Err.Description
   Resume Exit_OK_Click

End Sub
```

На этом действия по подготовке формы, с которой будет связан макрос, можно завершить. При сохранении формы в файле текущей базы данных следует ввести имя файла "ОПрограмме". Теперь на вкладке **Формы** окна базы данных можно увидеть новый объект базы – форму "ОПрограмме" (рис.5).

3. Запись макроса.

Для записи макроса, открывающего созданную форму "ОПрограмме", потребуется выполнить следующую последовательность действий:

1. В окне базы данных выбрать вкладку **Макросы.**

2. Нажать кнопку создать. Откроется окно нового макроса (рис.6). Верхняя часть окна используется для определения нового макроса, а нижняя – для ввода значений аргументов макрокоманд, включенных в макрос.

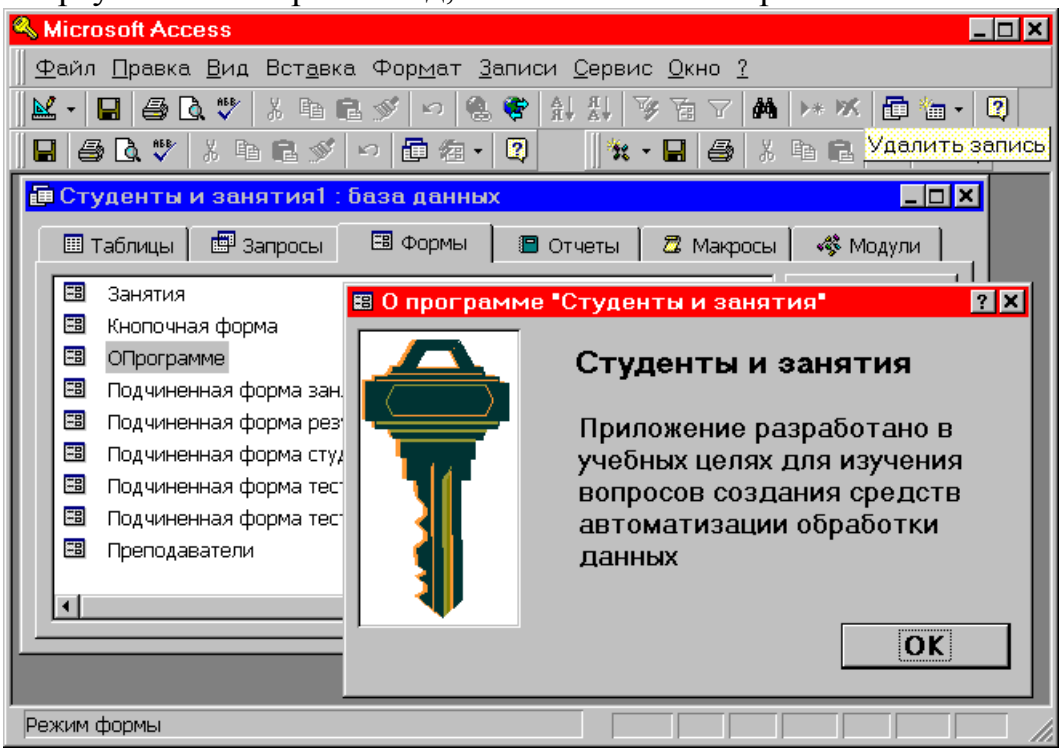

Рис. 5.

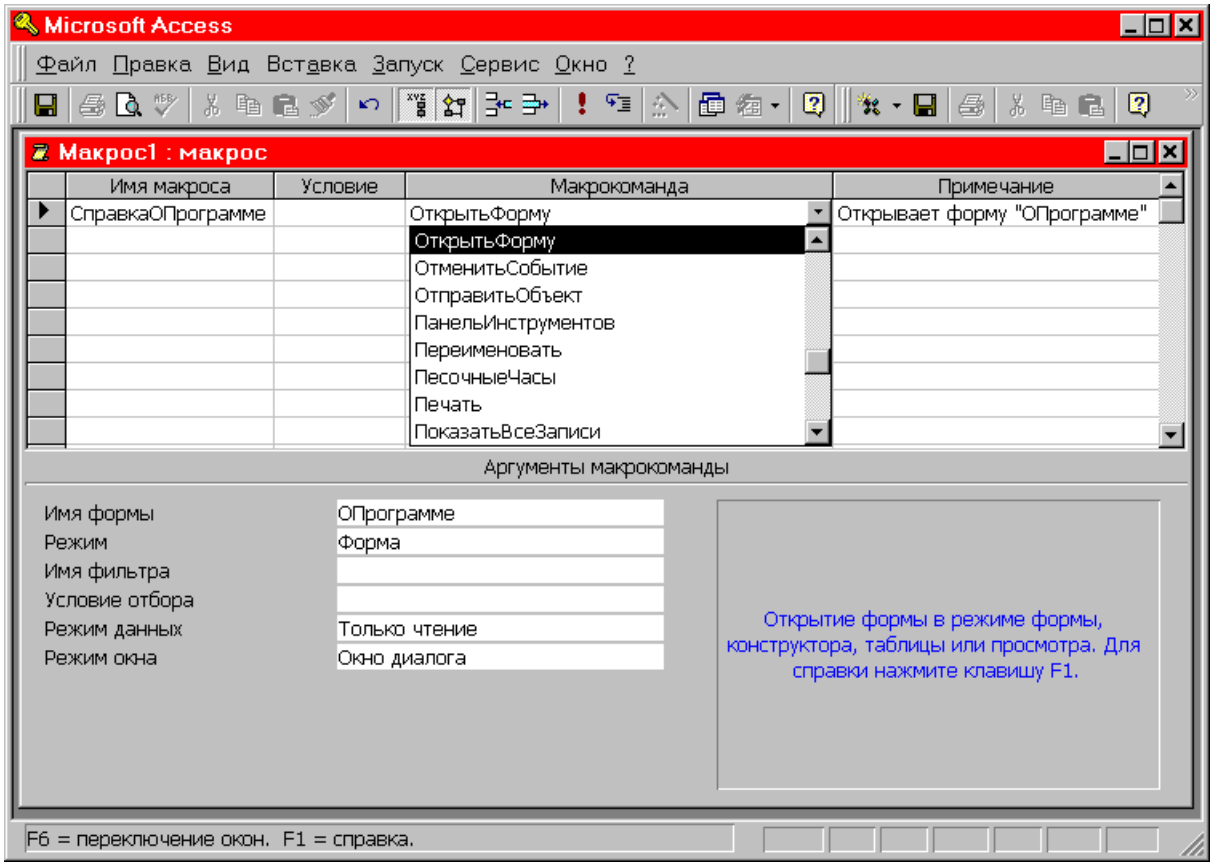

Рис.6.

В верхней части окна присутствуют по крайней мере два столбца с заголовками

Макрокоманда и **Примечание.** Если столбцы с заголовками **Имя макроса** и **Условие** не появились, их можно увидеть, нажав кнопки **Имена макросов** и **Условия** на панели инструментов.

Обратите внимание на область в правой нижней части окна макроса, в которой выводится краткая справка. Содержание сообщения меняется в зависимости от положения курсора в верхней части окна макроса.

3. Щелкнуть мышью в ячейке столбца **Макрокоманда.** В правом конце ячейки появится кнопка списка макрокоманд, которые можно включить в макрос. Имеется возможность выбора одной из более, чем 50 макрокоманд списка.

4. Раскрыть список макрокоманд и выбрать имя подходящей макрокоманды.

5. Ввести текст комментария к макрокоманде. Столбец **Примечание** особенно полезен для документирования сложных макросов, содержащих большое число макрокоманд. Комментарии в этом столбце можно размещать на пустых строках, не содержащих макрокоманд.

6.В нижней половине окна, при необходимости, указать аргументы макрокоманды.

7. Для добавления в макрос других макрокоманд перейти на следующую строку в столбце **Макрокоманда** и повторить шаги с 3 по 6.

8. Задать имя макросу.

Все используемые в рассматриваемом примере установки в окне макроса отображены на рис 6. Записанный макрос необходимо сохранить. Для этого используется команда **Файл\Сохранить**. В окне диалога **Сохранение** ввести имя макроса *СправкаОПрограмме*. Далее нажать кнопку **ОК**. Теперь на вкладке **Макросы** окна базы данных можно увидеть имя вновь созданного объекта базы.

Существует полезный прием быстрого создания макроса, выполняющего действия над конкретным объектом базы данных. Для этого надо выбрать объект в окне базы и переместить его с помощью мыши в ячейку макрокоманды в окне макроса. Например, для того чтобы создать макрос открывающий форму, переместите мышью значок формы в ячейку макрокоманды. Для этого расположите окно базы данных и окно макроса рядом на экране с помощью команды **Сверху вниз** или **Слева направо** из меню **Окно**. Затем выберите в окне базы вкладку объектов нужного типа, выберите объект и переместите его в ячейку макрокоманды. При переносе объекта в макрос добавляется макрокоманда, открывающая объект. Остается лишь, при необходимости, отредактировать установки в окне макроса, отличные от устанавливаемых по умолчанию системой. В том случае, если переносимым в ячейку макрокоманды объектом является макрос, создается макрокоманда, запускающая этот макрос.

Следующие рекомендации оказываются полезными при указании аргументов макрокоманды:

- В большинстве случаев существует возможность выбора значения аргумента макрокоманды из раскрывающегося списка.

• Рекомендуется вводить значения аргументов в порядке их расположения в бланке, поскольку возможные значения конкретного аргумента могут определяться значениями ранее указанных аргументов.

• Если макрокоманда была введена путем переноса объекта базы данных из окна базы данных, то правильный набор значений аргументов задается автоматически.

• Для некоторых аргументов макрокоманд допускается задание выражения с предшествующим знаком равенства (=). Кнопка построителя выражений (…), появляющаяся справа от ячейки соответствующего аргумента, позволяет автоматизировать создание выражений, являющихся значениями аргументов.

В некоторых случаях требуется выполнить макрокоманду или серию макрокоманд только при выполнении некоторых условий. В таких случаях условия позволяют определить порядок передачи управления между макрокомандами в макросе. Условие задается с помощью логического выражения. В зависимости от значения логического выражения управление передается разным макрокомандам.

Примеры записи условных выражений:

Выражение	**Условие выполнения макрокоманды**
Город="Москва"	Поле "Город" в форме, из которой запускается макрос, содержит значение "Москва".
[Дата] Between #2-фев-98# And #2-мар-98#	Поле "Дата" в форме, из которой запускается макрос содержит значение не раньше 2-фев-98 и не позже 2-мар-95.
Forms!Товары!На складе<5	Значение поля "На складе" в форме "Товары" меньше 5.
IsNull([Имя])	Поле "Имя" в форме, из которой запускается макрос, содержит пустое (Null) значение. Данное выражение эквивалентно следующему: [Имя] Is Null.
[Страна]="Литва" And Forms![Сумма продаж]![Объем заказов]>100	Одновременное выполнение двух условий. Поле "Страна" в форме, из которой запускается макрос, содержит значение "Литва", а значение поля "Объем заказов" в форме "Сумма продаж" превышает 100.

Условное выражение вводится в ячейку столбца **Условие** в окне макроса. Если условие истинно, выполняется макрокоманда, содержащаяся в данной строке. Для того чтобы выполнить набор макрокоманд при истинности данного условия, следует ввести, начиная со следующей строки, многоточие (…) в

ячейки столбца **Условие** идущих подряд макрокоманд, принадлежащих этому набору.
Для того чтобы временно пропустить макрокоманду, надо ввести значение **False** в соответствующую макрокоманде ячейку условия. Такой прием часто используют при отладке макросов.
При указании условий выполнения макроса, придерживайтесь следующей последовательности действий:
1. В окне макроса нажать кнопку **Условия** на панели инструментов (если столбец **Условие** не виден в окне макроса).
2. Введите условное выражение в ячейку столбца **Условие** той строки, для которой необходимо указать условие. Чтобы создать выражение с помощью построителя выражений, надо щелкнуть правой кнопкой мыши ячейку **Условие** и выбрать команду **Построить…** в появившемся контекстном меню.
3. Ввести в ячейку столбца **Макрокоманда** имя макрокоманды, которая должна выполняться, если условное выражение истинно.
При запуске макроса проверяется значение каждого условного выражения. Если это выражение истинно, выполняется макрокоманда, находящаяся в той же строке и все идущие подряд макрокоманды, у которых в ячейках столбца **Условие** содержится многоточие (**…**). Например, в макросе на рис.7 макрокоманды **Сообщение** и **ОстановитьМакрос** выполняются только в том случае, если поле "КодПоставщика" имеет пустое (**Null**) значение.

	Условие	Макрокоманда
▶		ВыводНаЭкран
	IsNull([КодПоставщика])	Сообщение
	…	ОстановитьМакрос
		ОткрытьФорму

Рис.7.
После этого будут выполнены все макрокоманды, у которых ячейки столбца **Условие** являются пустыми, до следующей макрокоманды с определенным условием, до следующего имени макроса или до конца макроса.
Если условное выражение ложно, Access игнорирует эту макрокоманду и все идущие подряд макрокоманды, у которых в ячейках столбца **Условие** содержится многоточие (**…**), и переходит к ближайшей строке, в которой содержится новое условие или ячейка условий является пустой.

4. Запуск и отладка макроса.

При запуске макроса выполнение макрокоманд начинается с первой строки макроса и продолжается до конца макроса, или, если макрос входит в группу макросов, до начала следующего макроса.
Выполнение макроса может начинаться по команде пользователя, при вызове из другого макроса или процедуры обработки события, а также в ответ на

событие в форме, отчете или элементе управления. Например, можно назначить макросу кнопку в форме, в результате чего макрос будет запускаться при нажатии кнопки. Допускается также создание специальной команды меню или кнопки на панели инструментов, запускающей макрос, определение сочетания клавиш, одновременное нажатие которых запускает макрос, а также автоматический запуск макроса при открытии базы данных.

Запуск макроса пользователем.

Некоторые макросы (как рассмотренный выше в качестве примера) могут быть запущены непосредственно из окна базы данных или окна макроса (прямой запуск), поскольку они не зависят от элементов управления открытой формы или отчета.

Варианты запуска макроса пользователем:

- Чтобы запустить макрос из окна макроса, нажать кнопку **Запуск** на панели инструментов.
- Чтобы запустить макрос из окна базы данных, выбрать вкладку **Макросы**, установить указатель мыши на имя макроса и дважды щелкнуть мышью.
- Чтобы запустить макрос из режима конструктора формы или режима конструктора отчета, выбрать в меню **Сервис** команду **Макрос\Запуск макроса.**
- Чтобы запустить макрос из любого окна Access, выбрать в меню **Сервис** команду **Макрос\Запуск макроса**. Затем выбрать нужный макрос в поле **Имя макроса.**

Обычно прямой запуск макроса используется только при тестировании макросов. После этого создают специальную команду меню, запускающую макрос, или связывают макрос с формой, отчетом или элементом управления и указывают запуск макроса в ответ на событие.

Перед запуском макроса неплохо проверить его работу, выполнив макрокоманды в пошаговом режиме. Чтобы начать пошаговую отладку макроса, перейдите в окно базы данных, на вкладке **Макросы** выделите имя макроса, который необходимо протестировать, и нажмите кнопку **Конструктор**. После открытия окна макроса, нажмите кнопку **По шагам** на панели инструментов, либо выберите команду меню **Запуск\По шагам**.

Теперь, после запуска макроса командой **Запуск** этого же меню, Access будет открывать окно диалога **Пошаговое исполнение макроса** (рис.8) перед выполнением каждого шага. В этом окне Вы увидите имя макроса, название макрокоманды, условие ее выполнения и аргументы макрокоманды.

Если в макросе определено более одной макрокоманды, то после первого шага вновь появится окно диалога **Пошаговое исполнение макроса**, содержащее следующую готовую к исполнению команду. Поскольку в рассматриваемом примере макрос состоит всего из одной макрокоманды, Access возвратит Вас в окно макроса.

Если во время выполнения приложения в каком-нибудь макросе встретится ошибка, то Access сначала выведет окно диалога, объясняющее ее. Затем

происходит переход к окну диалога **Ошибка выполнения макрокоманды** с информацией о макрокоманде, вызвавшей ошибку. В этом состоянии можно нажать только кнопку **Прервать** и отредактировать макрос, чтобы устранить причину ошибки.

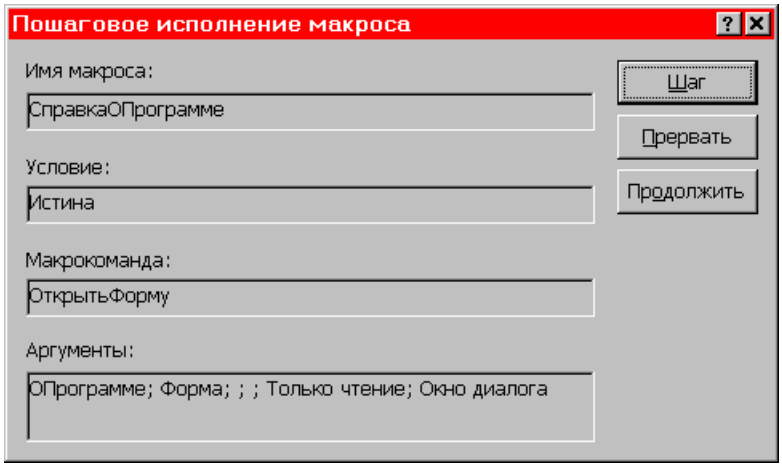

Рис.8.

Завершая отладку в режиме прерывания, вернитесь в окно макроса (если произошел выход из него), и еще раз нажмите кнопку **По шагам**, чтобы отменить пошаговый режим. В противном случае Вы останетесь в режиме пошагового выполнения любого макроса, пока не закроете и не перезапустите Access.

Запуск макроса, входящего в группу макросов.

В большинстве форм, создаваемых для приложения, требуется применение значительного числа макрокоманд. Одни используются для редактирования полей, другие открывают отчеты, третьи реагируют на нажатие командных кнопок. Можно создать макросы для каждой отдельной операции, но в этом случае в приложении накопятся сотни различных макросов.

Существует эффективный способ упростить ситуацию. Он связан с группированием макросов. Группирование макросов можно произвести для каждой формы или отчета. Другой подход состоит в группировании макросов по типу операций. Например, можно создать группу макросов, содержащую все макрокоманды типа **ОткрытьФорму**, используемые в базе данных.

Каждому макросу в группе дается имя, которое заносится в столбец **Имя макроса**. Любой макрос из группы может быть вызван на исполнение. Например, если запуск макроса планируется осуществлять из какого-нибудь другого макроса, то в макрокоманде **ЗапускМакроса** надо использовать в качестве аргумента **Имя макроса** групповое имя запускаемого макроса, используя синтаксис: **имяГруппыМакросов.имяМакроса**.

Имя группы макросов, указанное при ее сохранении, появится в списке макросов в окне базы данных.

При использовании в меню **Сервис** команды **Макрос\Запуск макроса,** поле со списком **Имя макроса** будет содержать в качестве элементов списка как имя

группы, так и имена макросов группы. Запуск конкретного макроса группы в этом случае осуществляется простым выбором его имени из списка.

При запуске макроса, находящегося в группе, из процедуры VBA следует применять метод **RunMacro** объекта **DoCmd**, используя при этом описанный выше синтаксис для ссылки на макрос. Например, в следующей инструкции метод **RunMacro** вызывает макрос "Макрос2":

DoCmd.RunMacro "Макрос2"

В начале группы макросов можно поместить несколько макрокоманд, которые будут выполняться при вызове группы без указания имени макроса.

5. Назначение макросу кнопки в форме.

Для вызова созданного выше макроса "СправкаОПрограмме" назначим кнопку, которую поместим на существующую в приложении "Студенты и занятия" форму "Главная кнопочная форма".

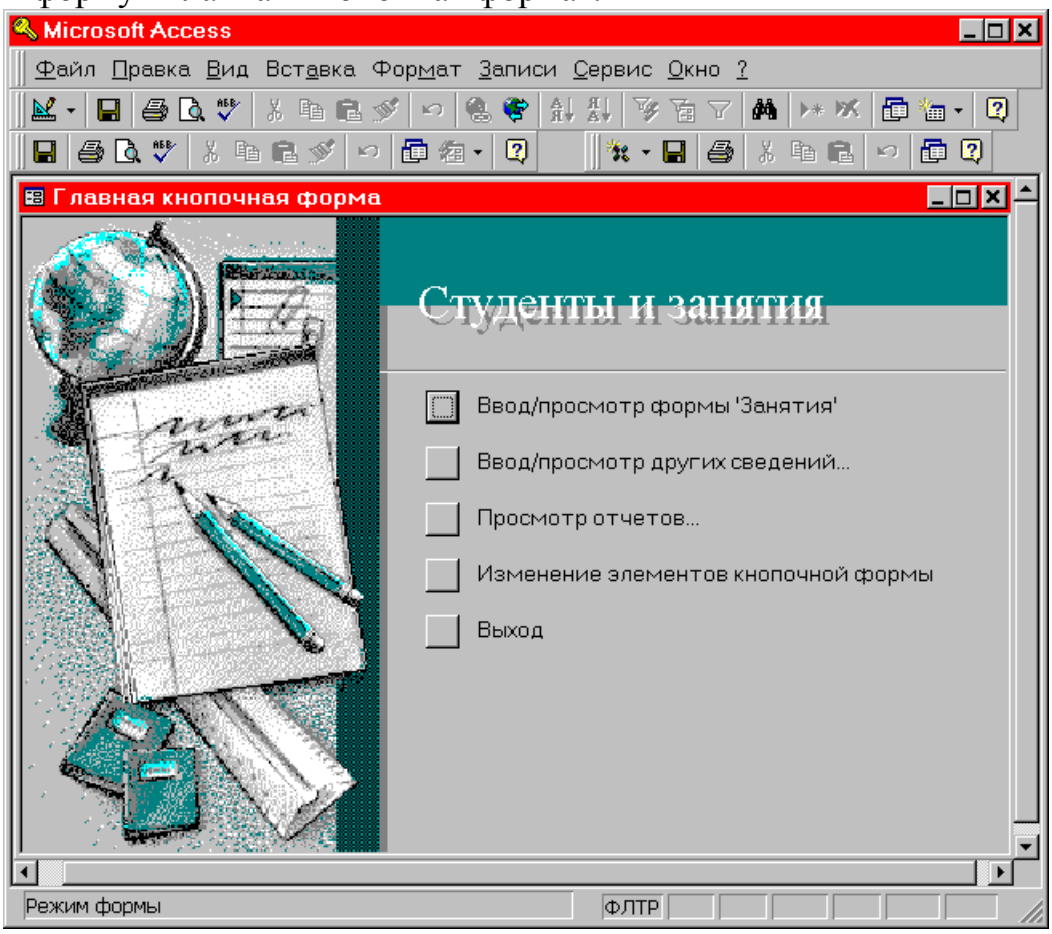

Рис.9.

Для этого надо выполнить следующую последовательность действий:

1. Находясь в окне базы данных, открыть форму "Кнопочная форма" в режиме конструктора.
2. Открыть панель элементов и нажать кнопку **Мастера**, если она уже нажата. Это позволит отключить услуги Мастеров и создать элемент управления в форме вручную.
3. Выбрать элемент управления **Кнопка** на панели элементов.

4. Выбрать в форме место, в которое помещается верхний левый угол кнопки. В нашем случае – ниже имеющейся кнопки "Выход".

5. Убедиться, что кнопка выбрана, и нажать кнопку **Свойства** на панели инструментов, чтобы открыть окно свойств. Привести размеры добавленной в форму кнопки в соответствие с размерами уже имеющихся кнопок, устанавливая значения для свойств **Высота** и **Ширина** в окне свойств. Подпись на кнопке можно удалить, устанавливая значениие свойства **Подпись** пустым.

6. Ввести в ячейку свойства **Нажатие кнопки** (**OnClick**) имя макроса, который должен выполняться при нажатии кнопки. Имя макроса (в данном случае, "СправкаОПрограмме") можно выбрать из раскрывающегося списка в этой же ячейке.

7. Используя панель элементов, разместить на форме справа от добавленной кнопки элемент **Надпись**, содержащий текст "О программе "Студенты и занятия"".

8. Выйти из режима конструктора формы с сохранением внесенных изменений.

Теперь "Главная кнопочная форма" приложения с добавленной кнопкой запуска макроса примет вид (рис.10):

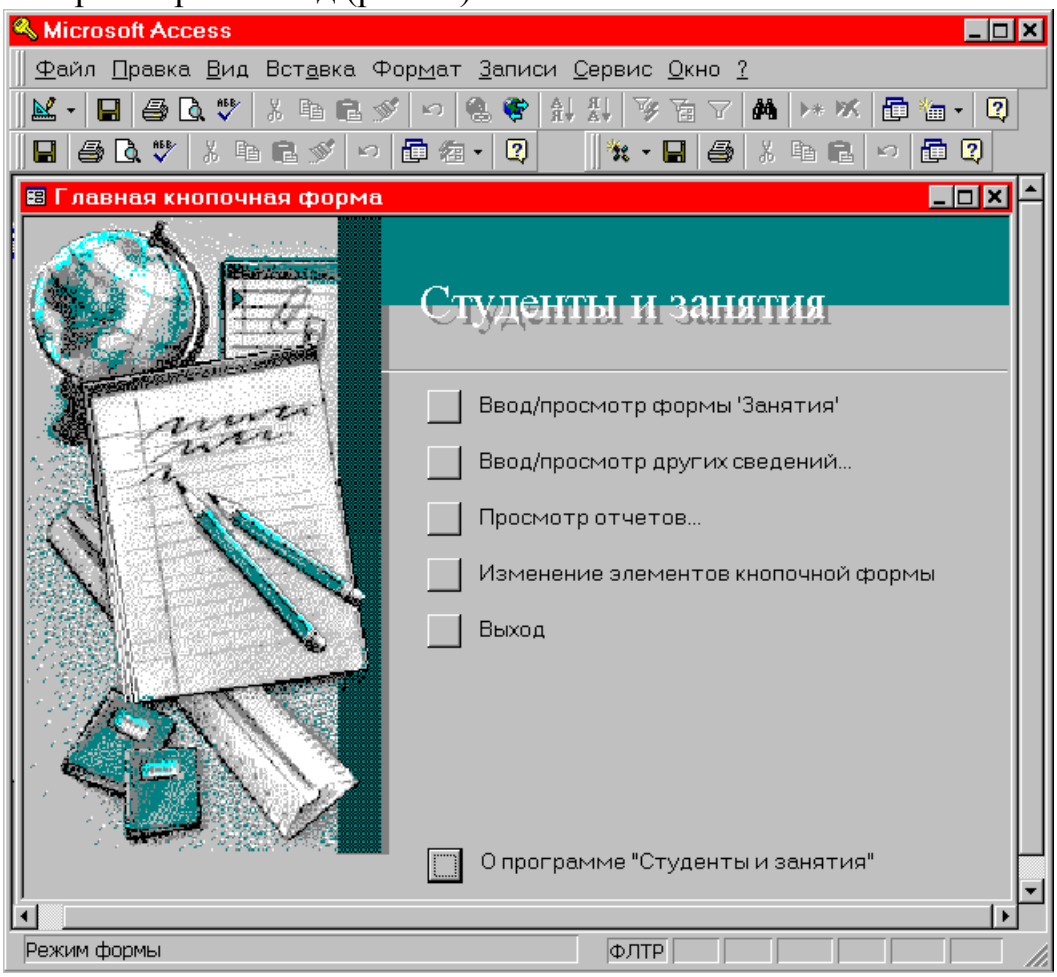

Рис.10.

6. Создание специальной команды меню, запускающей макрос.

В некоторых случаях требуется использовать одни и те же средства в разных частях приложения. Например, желательно выполнять какую–либо

команду независимо от того, с чем Вы работаете в данный момент – с формой, отчетом или чем-нибудь еще. Возможность подобных действий дает процедура вставки в существующее меню требуемой команды или добавление в строку меню нового меню, содержащего требуемую команду. Наконец, можно полностью перепроектировать существующую строку меню.

Для созданного выше макроса "СправкаОПрограмме" встроим команду вызова макроса вменю "**?**" существующей строки меню. Для этого надо выполнить следующую процедуру:

1. В меню **Вид** выбрать команду **Панели инструментов**, а затем подкоманду **Настройка**. Откроется окно диалога Настройка (рис. 11).

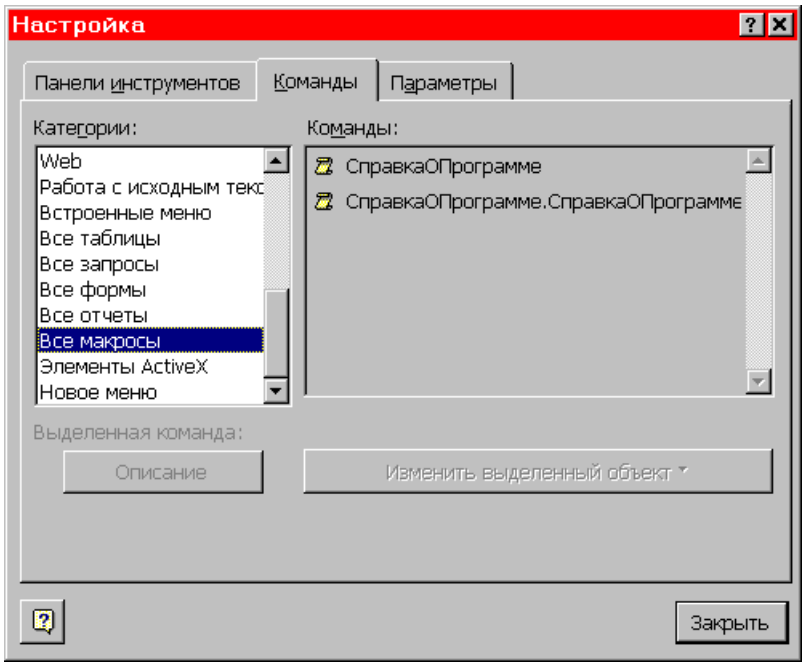

Рис. 11.

2. Выбрать вкладку **Команды**.

3. В поле **Категории** выбрать нужный параметр. В рассматриваемом случае добавляемым элементом является команда меню, запускающая макрос. Поэтому в качестве выбираемого параметра следует установить **Все макросы**.

4. Перетащить мышью нужную команду (СправкаОПрограмме) из поля **Команды** в меню в строке меню (в данном случае – в меню "**?**"). После того, как будет выведен список команд (или пустой бланк, если меню новое), выбрать нужное расположение для встраиваемой команды в меню и отпустить кнопу мыши. Не закрывая окна диалога, можно заменить название команды, например, **СправкаОПрограмме** на **Справка о программе**.

5. Закрыть окно диалога **Настройка**.

Теперь, открывая меню "**?**", в нем можно обнаружить новую команду **Справка о программе.**

Рассмотрим еще один вариант встраивания команды в меню. В отличие от предыдущего, добавим новое меню в существующую строку меню, а после этого в добавленное меню встроим команду вызова макроса. Для этого в поле

Категории в качестве выбираемого параметра следует установить **Новое меню**. Затем перетащить мышью из поля **Команды** элемент **Новое меню** в строку меню (например, после меню "**?**"). После этого будет выведен пустой бланк, так как меню новое. Не закрывая окно диалога **Настройка**, можно заменить стандартное название добавленного меню, например, дать название **Справка**. Далее, повторяя действия по встраиванию команды (шаги с 3 – 5), но только в новое меню, можно получить встроенную команду **Справка о программе** в добавленное меню **Справка** (рис. 12).

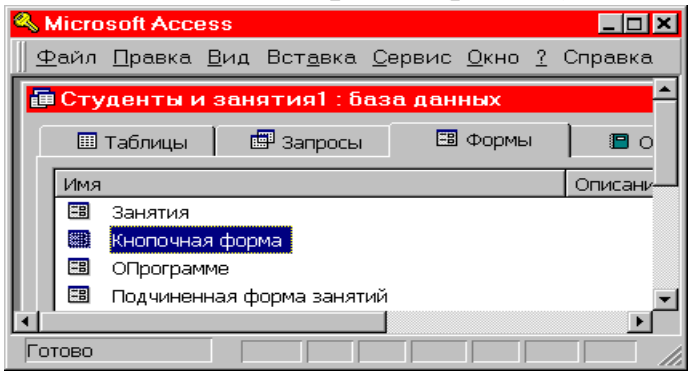

Рис. 12.

7. Создание кнопки панели инструментов, запускающей макрос.

Добавление кнопки на панель инструментов является еще одним удобным средством выполнения некоторой команды, находясь в различных местах приложения. Процедура создания кнопки на панели инструментов во многом идентична рассмотренной выше процедуре добавления команды меню, поскольку использует то же окно диалога **Настройка.** Здесь рассмотрим лишь отличия, связанные с созданием кнопки.

Прежде всего панель инструментов, к которой добавляется кнопка, должна быть видимой. Для этого на вкладке **Панели инструментов** диалогового окна **Настройка** следует установить флажок требуемой панели инструментов (рис.13.). Пусть, к примеру, это будет панель инструментов Visual Basic.

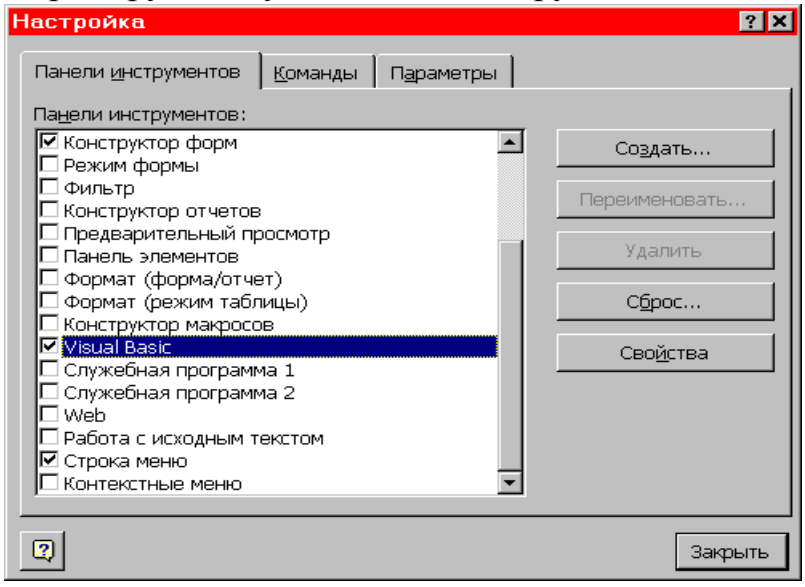

Рис. 13.

Далее следует перейти на вкладку **Команды** и, поскольку добавляемым элементом является кнопка, запускающая макрос, выбрать параметр **Все макросы**. После этого перетащить мышью нужную команду из поля **Команды** на открытую панель инструментов. На панели появится кнопка со стандартным значком. Его можно изменить, воспользовавшись командой **Изменить выделенный объект\Выбрать значок для кнопки** диалогового окна **Настройка.** На рис.14 для запуска макроса, открывающего форму со справочными данными о приложении "Студенты и занятия", выбран символ 

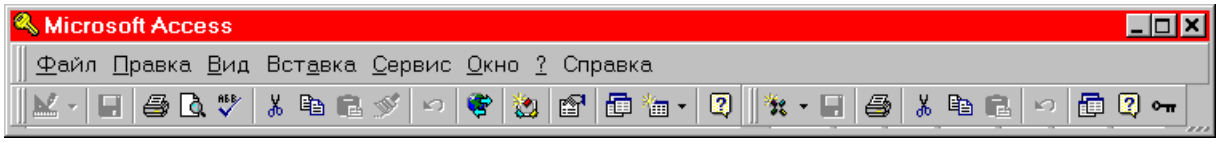

Рис. 14.

8. Назначение сочетания клавиш для запуска макроса.

В Access допускается связывание макрокоманды или набора макрокоманд с конкретной клавишей или сочетанием клавиш с помощью специальной группы макросов **AutoKeys**. После этого при нажатии клавиши или сочетания клавиш Access будет выполнять данную макрокоманду. Действия, которые должны быть выполнены при назначении, следующие:

1. В окне базы данных выбрать вкладку **Макросы**.
2. Нажать кнопку **Создать**.
3. Указать в ячейке столбца **Имя макроса** клавишу или сочетание клавиш, с которыми связывается макрокоманда или набор макрокоманд.

В следующем списке представлены сочетания клавиш, используемые для назначения клавиш в группе макросов **AutoKeys**. Допустимые сочетания клавиш являются подмножеством синтаксиса инструкции Visual Basic **SendKeys.**

Инструкция SendKeys	**Сочетание клавиш**
^A или ^4	CTRL+Любая буква или цифра
{F1}	Любая функциональная клавиша
^{F1}	CTRL+Любая функциональная клавиша
+{F1}	SHIFT+Любая функциональная клавиша
{INSERT}	INS
^{INSERT}	CTRL+INS
+{INSERT}	SHIFT+INS
{DELETE} or {DEL}	DEL
^{DELETE} or ^{DEL}	CTRL+DEL
+{DELETE} or +{DEL}	SHIFT+DEL

Здесь сделаем одно существенное замечание. Если макрокоманда или набор макрокоманд, связывается с сочетанием клавиш, которое уже используется в Access (например, Ctrl+C – сочетание клавиш для команды **Копировать**), то

новое назначение макрокоманд на это сочетание клавиш заменит стандартное назначение команд Access. Об этом не стоит забывать.

4. Ввести в макрос макрокоманду или набор макрокоманд, которые должны выполняться при нажатии указанной клавиши или сочетания клавиш. В нашем случае используется макрокоманда **ЗапускМакроса**, которая будет запускать макрос "СправкаОПрограмме" при нажатии клавиш **SHIFT+INSERT** (рис.15).

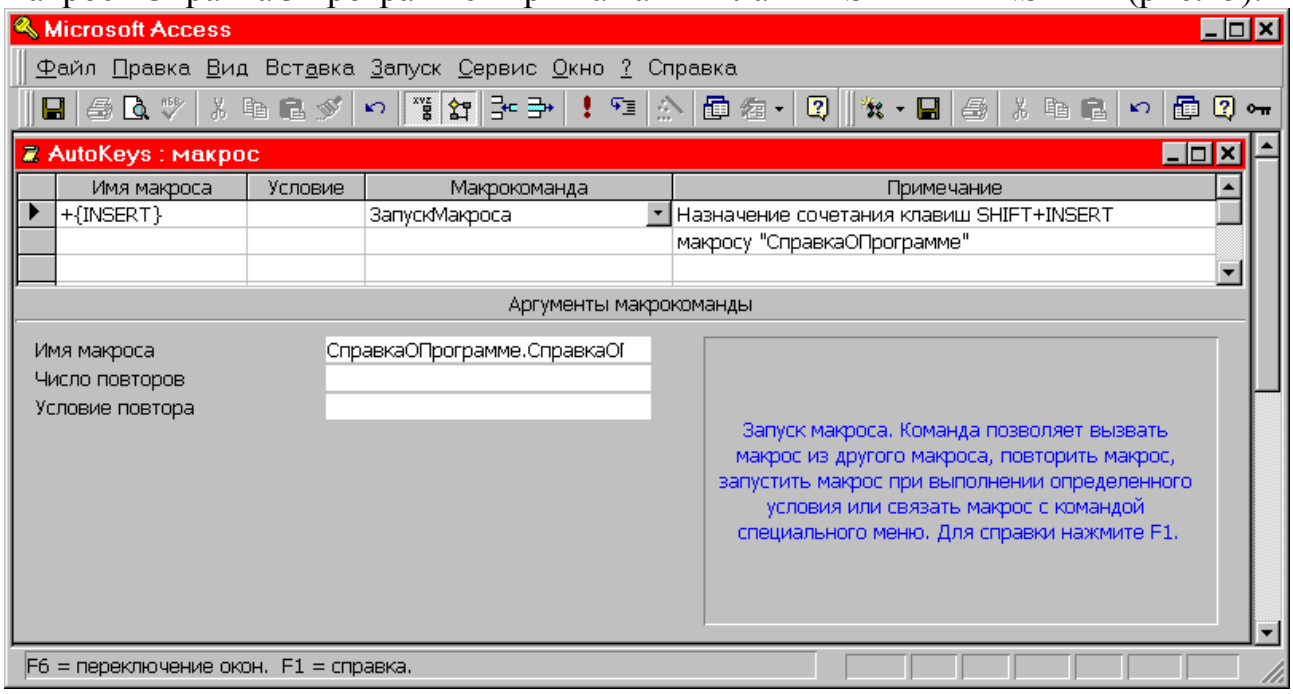

Рис. 15.

5. Повторить, при необходимости, шаги 3 и 4 для каждого нового определения сочетания клавиш.

6. Сохранить группу макросов под именем **AutoKeys**. Новые назначения клавиш вступят в силу сразу после сохранения макроса и будут использоваться при следующем открытии базы данных.

9. Запуск макроса при открытии базы данных.

Специальный макрос с именем **Autoexec** позволяет автоматически выполнить макрокоманду или набор макрокоманд при открытии базы данных. В процессе открытия базы данных Access проводит поиск макроса с этим именем и, если такой макрос существует, автоматически запускает его. Для создания макроса автозапуска:

1. Записать макрос, содержащий макрокоманды, которые требуется автоматически выполнить при открытии базы данных.

2. Сохранить макрос под именем **Autoexec.**

При следующем открытии базы данных Access автоматически запустит этот макрос. Следует учитывать, что кроме макроса **Autoexec,** параметры автозапуска при открытии базы данных можно задать в окне диалога **Параметры запуска**, открываемого из меню **Сервис**. Настройки, устанавливаемые в этом диалоговом окне, могут быть использованы вместо или в дополнение к макросу **Autoexec.** Макрос **Autoexec** запускается после

вступления этих настроек в действие. Поэтому в него не следует включать макрокоманды, которые могут изменить параметры запуска. Например, если в поле **Форма** в диалоговом окне **Параметры запуска** указано имя формы, открываемой при запуске, а в макросе **Autoexec** вызывается макрокоманда **Открыть форму**, то сначала будет открыта форма, указанная в диалоговом окне **Параметры запуска,** а сразу за ней – форма, указанная в макрокоманде **Открыть форму.**

Завершая рассмотрение нашего примера, можно создать макрос **Autoexec,** содержащий макрокоманду открытия формы – справочного окна "О программе "Студенты и занятия", появляющегося при запуске базы данных сразу, после открытия главной кнопочной формы. Теперь на вкладке **Макросы** окна базы данных можно видеть имена трех, рассмотренных выше макросов (рис. 16).

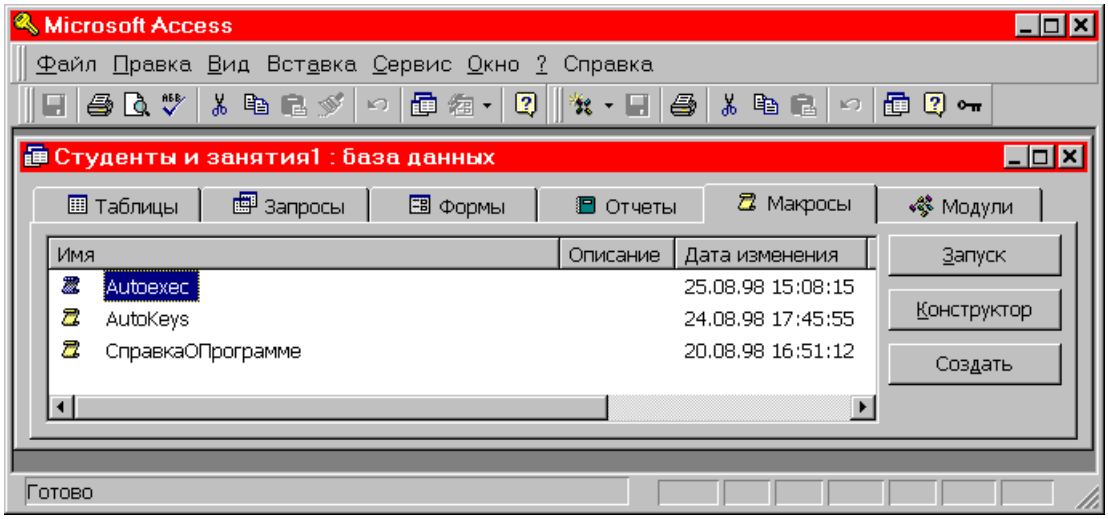

Рис. 16.

При необходимости, создаваемые макросы можно автоматически преобразовать в модули VBA, которые выполняют эквивалентные действия с помощью программ Visual Basic. Допускается преобразование макросов, определенных в форме или отчете, или преобразование общих макросов, не связанных с конкретными формами и отчетами.

Для преобразования макросов в форме или отчете в процедуры VBA следует, находясь в режиме конструктора формы или конструктора отчета, выполнить команду **Сервис\Макрос\Преобразовать макросы.**

Для преобразования общих макросов в процедуры VBA следует выбрать макрос на вкладке **Макросы** окна базы данных. После этого выполнить команду **Файл\Сохранить как / экспорт** и в диалоговом окне **Сохранение** выбрать параметр **В виде модуля Visual Basic**.

После ознакомления с особенностями, относящимися к технологии создания макросов в Access, **Вашей задачей** теперь является:

1. Воспроизведение описанных этапов работы применительно к самостоятельно подготовленному объекту автоматизации – базе данных, в соответствии с вариантом заданий.
2. Оформление результатов в формате исполняемого файла – **lr4.mdb** и файла-отчета по лабораторной работе – **lr4Отчет.doc**.

Варианты заданий для разработки макросов в MS Access.

Вариант 1.

А. Создать файл, содержащий сведения о месячной зарплате рабочих завода. Каждая запись содержит поля – фамилия рабочего, наименование цеха, размер зарплаты за месяц. Количество записей ≥ 10.

Б. Вычислить общую сумму выплат за месяц по цеху X, а также среднемесячный заработок рабочего этого цеха. Напечатать для бухгалтерии ведомость для начисления зарплаты рабочим этого цеха.

Вариант 2.

А. Создать файл, содержащий сведения о количестве изделий, собранных сборщиками цеха за неделю. Каждая запись содержит поля: фамилия сборщика, количество изделий, собранных им ежедневно в течении шестидневной недели. Количество записей ≥ 10.

Б. Написать программу, выдающую на печать следующую информацию: фамилию сборщика и общее количество деталей, собранных им за неделю, фамилию сборщика собравшего наибольшее число изделий, и день, когда он достиг наивысшей производительности труда.

Вариант 3.

А. Создать файл, содержащий сведения о количестве изделий категорий A, B, C, собранных рабочими за месяц. Структура записи: фамилия сборщика, наименование цеха, количество изделий по категориям, собранных рабочим за месяц. Количество записей ≥ 10.

Б. Считая заданными значения расценок S_a, S_b, S_c за выполненную работу по сборке единицы изделия категорий A, B, C соответственно, выдать на печать следующую информацию:

- общее количество изделий категорий A, B, C, собранных рабочим цеха X;
- ведомость заработной платы рабочих цеха X;
- средний размер заработной платы работников этого цеха.

Вариант 4.

А. Создать файл, содержащий сведения о телефонах абонентов. Каждая запись имеет поля:

- фамилия абонента;
- год установки телефона;
- номер телефона.

Количество записей ≥ 10.

Б. Написать программу, выдающую информацию следующего вида:
- по вводимой фамилии абонента выдается номер абонента;
- определяется количество установленных телефонов с XXXX г.

Вариант 5.

А. Создать файл, содержащий сведения об ассортименте игрушек в магазине. Структура записи:
- название игрушки;
- цена;
- количество;
- возрастные границы.

Количество записей ≥ 10.

Б. Написать программу, в результате выполнения которой выдаются следующие сведения:
- названия игрушек, которые подходят детям от 1 до 3-х лет;
- стоимость самой дорогой игрушки;
- названия игрушек, которые по стоимости не выше X руб. и подходят детям от A до B лет. Значения X, A, B вводить на форме.

Вариант 6.

А. Создать файл, содержащий сведения о сдаче студентами 4 курса кафедры “ПМ” сессии. Структура записи: индекс группы, фамилия студента, оценки по пяти предметам, признак участия в общественной работе:
- "1" – активное участие
- "0" – неучастие.

Количество записей ≥ 25.

Б. Написать программу, зачисления студентов группы X на стипендию. Студент, получивший все оценки "5" и активно участвующий в общественной работе, зачисляется на повышенную стипендию (доплата 50%). Не активно участвующий – доплата 25%. Студенты, получившие "4","5", зачисляются на обычную стипендию. Студенты, получившие одну "3", но активно занимающиеся общественной работой, также зачисляются на стипендию, в противном случае зачисление не производится. Индекс группы вводится на форме.

Вариант 7.

А. Создать файл, содержащий сведения о сдаче студентами сессии. Структура записи: индекс группы, фамилия студента, оценки по пяти экзаменам и пяти зачетам ("з" означает зачет, "нз" – незачет). Количество записей ≥ 25.

Б. Написать программу, в результате выполнения которой выдаются следующие сведения:
- фамилии неуспевающих студентов с указанием индексов групп и количества задолженностей;
- средний балл, полученный каждым студентом группы X, и всей группы в целом.

Вариант 8.

А. Создать файл, содержащий сведения о личной коллекции книголюба. Структура записи: шифр книги, автор, название, год издания, местоположение (номер стеллажа, шкафа и т.п.). Количество записей ≥ 20.

Б. Написать программу, выдающую следующую информацию:

- местоположение книги автора X названия Y. Значения X и Y вводятся на форме;
- список книг автора Z ,находящихся в коллекции;
- число книг издания XXXX года, имеющихся в библиотеке.

Вариант 9.

А. Создать файл, содержащий сведения о наличии билетов и рейсах Аэрофлота. Структура записи: номер рейса, пункт назначения, время вылета, время прибытия, количество свободных мест в салоне. Количество записей ≥ 20.

Б. Написать программу, выдающую информацию следующего вида:

- время отправления самолетов в город X.
- наличие свободных мест на рейс в город X с временем отправления Y.

Значения X, Y по запросу вводятся на форме.

Вариант 10.

А. Создать файл, содержащий сведения об ассортименте обуви в магазине фирмы. Структура записи: артикул, наименование, количество, стоимость одной пары. Количество записей ≥ 20. Артикул начинается с буквы Д для дамской обуви, М - для мужской, П - для детской.

Б. Написать программу, выдающую следующую информацию:

- о наличии и стоимости обуви артикула X;
- ассортиментный список дамской обуви с указанием наименования и имеющегося в наличии числа пар каждой модели.

Вариант 11.

А. Создать файл, содержащий сведения о 10 нападающих хоккейных команд «Спартак» и «Динамо». Соответствующие таблицы должны содержать: имена нападающих, число заброшенных ими шайб, сделанных голевых передач, заработанное штрафное время;

Б. Написать программу, которая по данным, извлеченным из таблиц, формирует отчет, содержащий имя, команду, сумму очков (голы + передачи) для шести лучших игроков обеих команд.

Вариант 12.

А. Создать файл, содержащий сведения о том, какие из пяти предлагаемых дисциплин по выбору желает слушать студент. Структура записи: фамилия студента, индекс группы, 5 дисциплин, средний бал успеваемости. Выбираемая дисциплина отмечается символом "1", иначе – пробел. Количество записей ≥ 25.

Б. Написать программу, которая печатает список студентов, желающих прослушать дисциплину X. Если число желающих превысит 8 человек, то отобрать студентов, имеющих более высокий средний бал успеваемости.

Вариант 13.

А. Создать файл, содержащий сведения об отправлении поездов дальнего следования с Ярославского вокзала. Структура записи: номер поезда, станция назначения, время отправления, время в пути, наличие билетов. Количество записей ≥ 20.

Б. Написать программу, которая позволяет получить следующую справочную информацию:

- время отправления поездов в город X во временном интервале отA до B часов;
- наличие билетов на поезд с номером XXX.

Вариант 14.

А. Создать файл, содержащий сведения о сотрудниках института. Структура записи: фамилия работающего, название отдела, год рождения, стаж работы, должность, оклад. Количество записей ≥10.

В. Написать программу, которая позволяет получить следующую информацию:

- список сотрудников пенсионного возраста на сегодняшний день с указанием стажа работы;
- средний стаж сотрудников, работающих в отделе X.

Вариант 15.

А. Создать файл, содержащий сведения о пациентах поликлиники. Структура записи: фамилия пациента, пол, возраст, место проживания (город), диагноз. Количество записей ≥10.

Б. Написать программу, которая позволяет получить следующую справочную информацию:

- количество иногородних пациентов, прибывших в клинику;
- список пациентов, старше X лет с диагнозом Y. Значения X, Y по запросу вводятся на форме.

Вариант 16.

А. Создать базу данных для работы виртуального магазина по продаже продуктов питания.

Б. Разработать объектно-ориентированный интерфейс для взаимодействия с базой данных для покупателя и сотрудников магазина.

Вариант 17.

А. Создать базу данных для работы виртуального магазина по продаже книг и печатной продукции.

Б. Разработать объектно-ориентированный интерфейс для взаимодействия с базой данных для покупателя и сотрудников магазина.

Вариант 18.

А. Создать базу данных для работы виртуального магазина по продаже аудио, видео, CD–продукции.

Б. Разработать объектно-ориентированный интерфейс для взаимодействия с базой данных для покупателя и сотрудников магазина.

Вариант 19.

А. Создать базу данных для работы виртуального магазина по продаже туристических путевок.

Б. Разработать объектно-ориентированный интерфейс для взаимодействия с базой данных для покупателя и сотрудников магазина.

Вариант 20.

А. Создать базу данных для работы виртуального магазина по продаже автомобилей.

Б. Разработать объектно-ориентированный интерфейс для взаимодействия с базой данных для покупателя и сотрудников магазина.

Вариант 21.

А. Создать базу данных для работы виртуального магазина по продаже компьютерной техники.

Б. Разработать объектно-ориентированный интерфейс для взаимодействия с базой данных для покупателя и сотрудников магазина.

Примечание. В вариантах №16-21, при разработке интерфейсных форм, необходимо учесть, чтобы покупатель имел возможность выбрать товар по каталогу, сформировать заказ и получить счет без посещения офиса или торгового зала. Сотрудникам магазина - получить информацию о сделанных заказах.

Вариант 22.

А. Создать базу данных для автоматизированной системы управления работой магазина.

Б. Разработать объектно-ориентированный интерфейс для взаимодействия с базой данных для должностного лица.

Вариант 23.

А. Создать базу данных для автоматизированной системы управления работой гостиницы.

Б. Разработать объектно-ориентированный интерфейс для взаимодействия с базой данных для должностного лица.

Вариант 24.

А. Создать базу данных для автоматизированной системы управления работой школы.

Б. Разработать объектно-ориентированный интерфейс для взаимодействия с базой данных для должностного лица.

Примечание. В вариантах №22-24, при разработке интерфейсных форм, необходимо обеспечить соответствующему должностному лицу возможность формировать структуру и штатное расписание организации, принимать сотрудников на работу, увольнять сотрудников, распределять денежные вознаграждения за отдельные виды работ, составлять график работы, объявлять взыскания и поощрения.

Вариант 25.

А. Создать базу данных для функционирования информационно-справочной системы морского порта.

Б. Разработать объектно-ориентированный интерфейс для взаимодействия с базой данных для пользователя и сотрудника информационно-справочной службы.

Вариант 26.

А. Создать базу данных для функционирования информационно-справочной системы автовокзала.

Б. Разработать объектно-ориентированный интерфейс для взаимодействия с базой данных для пользователя и сотрудника информационно-справочной службы.

Вариант 27.

А. Создать базу данных для функционирования информационно-справочной системы библиотеки.

Б. Разработать объектно-ориентированный интерфейс для взаимодействия с базой данных для пользователя и сотрудника информационно-справочной службы.

Вариант 28.

А. Создать базу данных для функционирования информационно-справочной системы гостиницы.

Б. Разработать объектно-ориентированный интерфейс для взаимодействия с базой данных для пользователя и сотрудника информационно-справочной службы.

Вариант 29.

А. Создать базу данных для функционирования информационно-справочной системы железнодорожного вокзала.

Б. Разработать объектно-ориентированный интерфейс для взаимодействия с базой данных для пользователя и сотрудника информационно-справочной службы.

Вариант 30.

А. Создать базу данных для функционирования информационно-справочной системы аэропорта.

Б. Разработать объектно-ориентированный интерфейс для взаимодействия с базой данных для пользователя и сотрудника информационно-справочной службы.

Примечание. В вариантах №25-30, при разработке интерфейсных форм, необходимо предоставить пользователю возможность многокритериального поиска необходимой информации и формирования заказа, а сотрудникам - возможность наполнения системы новыми данными и управления заказами.

Дополнительная литература.

1. Биллиг В.А.. **Основы офисного программирования и язык VBA**.
http://www.intuit.ru/department/office/vba2000/0/1.html
2. Биллиг В.А. **Основы офисного программирования и документы Excel.**
http://www.intuit.ru/department/office/vbaexcel/
3. Биллиг В.А. **Основы офисного программирования и документы Word**.
http://www.intuit.ru/department/office/vbaword/
4. Заика А.А. **VBA в MS Office 2007**.
http://www.intuit.ru/department/se/vbamsoffice2007/

5. Джонсон Б., Скиба К., Янг М. **Основы Microsoft Visual Studio .NET 2003.** М.: Русская Редакция, 2003.
6. Гарнаев А. **Самоучитель Visual Studio .NET 2003**. СПб.: БХВ-Петербург, 2003.

7. Хальворсон М. **Visual Basic.NET 2003**. Русская версия. М.: ЭКОМ, 2004.
8. Гарнаев А. **Visual Basic .NET. Разработка приложений**. СПб.: БХВ-Петербург, 2002.

9. Джефри Рихтер. **Программирование на платформе Microsoft .NET Framework 2.0 на языке C#. Мастер класс** М.: Издательско-торговый дом "Русская Редакция", 2007. – 656 с
10. Карли Ватсон. **C#.** М.: Издательство "Лори", 2005. – 862 с
11. Павловская Т.А. **C#. Программирование на языке высокого уровня. Учебник для вузов** СПб.: Питер, 2007. – 432 с
12. Троелсен Э. **C# и платформа .NET. Библиотека программиста** СПб.: Питер, 2004. —796 с
13. Биллиг В.А. **Основы программирования на C#**. Интернет-университет информационных технологий - ИНТУИТ.ру, 2006 г., 488 стр.
14. Кариев Ч.А. **Разработка Windows-приложений на основе Visual C#.** Интернет-университет информационных технологий - ИНТУИТ.ру, БИНОМ. Лаборатория знаний, 2007 г., 768 стр.
15. Кудрина Е.В., Огнева М.В., Портенко М.С. **Программирование на языке C#: разработка консольных приложений.** Интернет-университет информационных технологий - ИНТУИТ.ру, 2009 г.

16. Столбовский Д.Н. **Разработка Web-приложений ASP .NET с использованием Visual Studio .NET.**
http://www.intuit.ru/department/internet/aspnetvsnet/