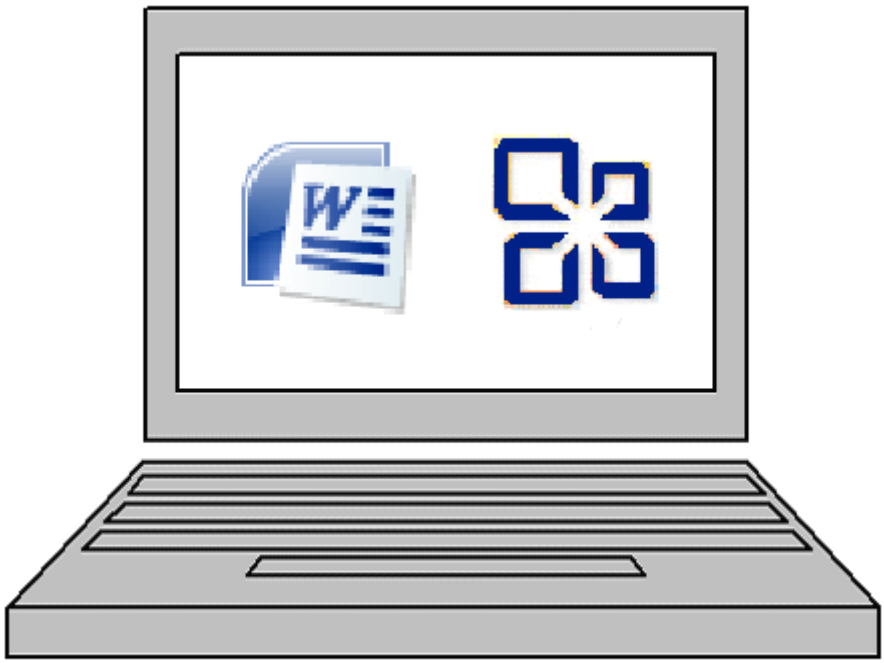
В.М. Пестриков
А.Н. Маслобоев

Основы программирования в Microsoft Word на Visual Basic for Applications

Учебное пособие



Санкт-Петербург
2010

Министерство образования и науки
Российской Федерации

Государственное образовательное учреждение
высшего профессионального образования

«Санкт-Петербургский государственный
технологический
университет растительных полимеров»

В.М. Пестриков
А.Н. Маслобоев

Основы программирования в Microsoft Word на Visual Basic For Applications

Учебное пособие

Санкт-Петербург
2010

ББК 32.97я7
П 286
УДК 681.3 (075)

Пестриков В.М., Маслобоев А.Н. Основы программирования в Microsoft Word на Visual Basic For Applications: учебное пособие /ГОУВПО СПб ГТУ РП. — СПб., 2010. — 83 с.: ил. 29.

В настоящем учебном пособии на конкретных примерах рассматривается основы объектно-ориентированного и визуального программирования в Microsoft Word с помощью инструментальной системы Visual Basic For Applications. Рассмотрены все этапы разработки проекта от постановки задачи до тестирования и отладки созданного приложения.

Пособие предназначено для студентов факультета МАП специальностей 150405 (170400) “Машины и оборудование лесного комплекса” и 240801 (170500) “Машины и аппараты химических производств”, проходящих обучение по дисциплине «Информатика», на II курсе (IV семестр; 32 ч) и III курсе (V семестр; 51 ч).

Рецензенты:
доцент кафедры информационных технологий, кандидат технических наук Петров Г.А. (СПб ГУСЭ);

кандидат технических наук С. Г. Рыбаков (СКБ «Турбина»).

Рекомендовано к изданию Редакционно-издательским советом университета в качестве учебного пособия.

Оглавление

Введение

Настоящее учебное пособие посвящено основам программирования в системе VBA применительно к Microsoft Word. Важным преимуществом рассматриваемой системы программирования является следующее: для того, чтобы начать с ней работать, в большинстве случаев пользователю не нужно инсталлировать на свой компьютер какой-либо новый программный продукт, как это имеет место в случае с такими системами программирования, как Delphi, Visual C++, Visual Basic и другими.

Если на компьютере у пользователя установлен такой программный пакет, как Microsoft Office (а он имеется в нашей стране у 99% пользователей персональных компьютеров), то он уже является обладателем современной эффективной и мощной системы программирования, которая называется VBA. VBA — это англоязычное сокращение полного названия данного программного продукта: Visual Basic For Applications, т.е. Visual Basic для приложений. Эта система интегрирована в программный пакет Microsoft Office и автоматически устанавливается на компьютер вместе с Microsoft Office при установке данного пакета.

Диапазон возможностей данной системы программирования очень широк: от программирования в диалоговом режиме без построения интерфейса (с использованием макрорекордера) до создания различных приложений с оригинальным пользовательским интерфейсом.

Пользователь может создавать простые программы (называемые макрокомандами или макросами) в системе VBA в диалоговом режиме, даже не владея языком Visual Basic. При записи макрокоманды с помощью макрорекордера система VBA автоматически генерирует программный код, который записывается в виде отдельной подпрограммы (процедуры). Пользователь может проанализировать программный код и затем использовать полученные знания уже при написании собственного программного кода.

Для более продвинутых пользователей, уже овладевших основами языка Visual Basic, система предоставляет готовый набор элементов управления для самостоятельной разработки пользовательского интерфейса приложения. Более того, пользователь может в случае необходимости сам создавать новые элементы управления и использовать их в своей программе.

В то же время система программирования VBA сохраняет преемственность с более ранними системами программирования. От пользователя, имеющего опыт работы в QBasic или QuickBasic, потребуется только овладение основными понятиями объектно-ориентированного программирования. Освоение же синтаксиса VBA потребует лишь минимальных затрат времени и усилий ввиду его схожести с синтаксисом QBasic и QuickBasic.

Это наглядно демонстрируют рис. В.1 и В.2. Оба примера показывают программный код, используемый для решения одной и той же задачи –

поиска корней квадратного уравнения вида $ax^2+bx+c=0$. На рис. В.1 показано решение данной задачи в системе программирования QuickBasic, а на рис. В.2. показан программный код для решения аналогичной задачи в системе программирования Visual Basic For Applications.

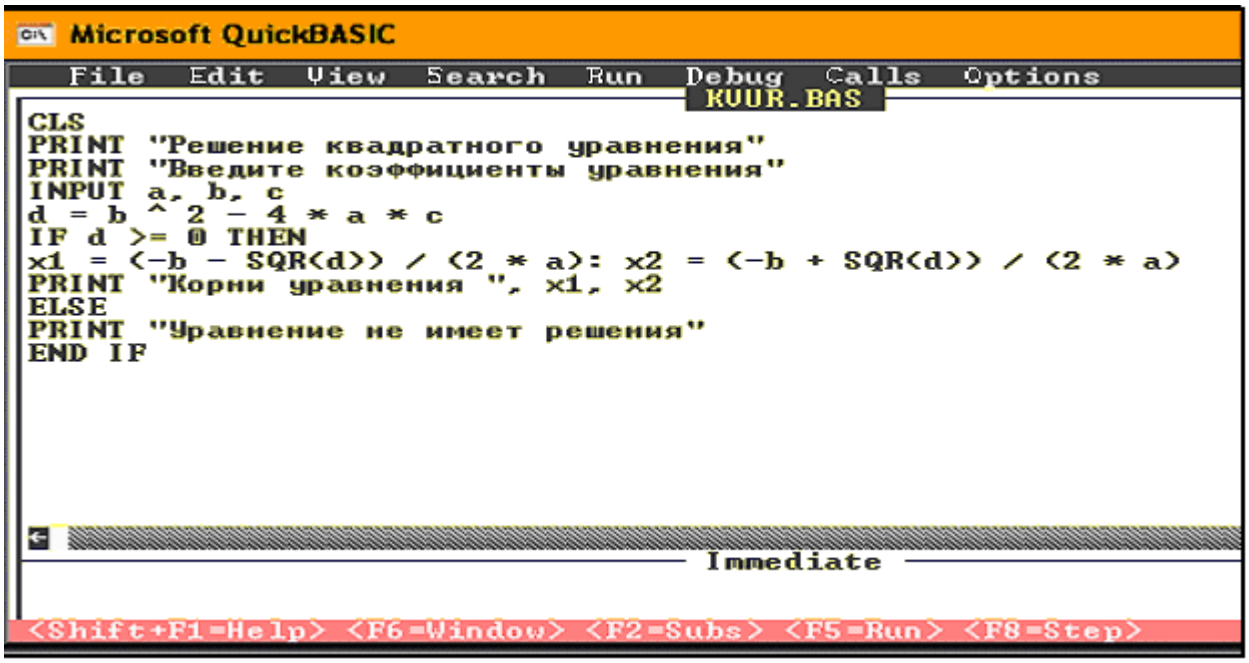


Рис. В.1. Экран системы программирования QuickBasic. Поиск корней квадратного уравнения

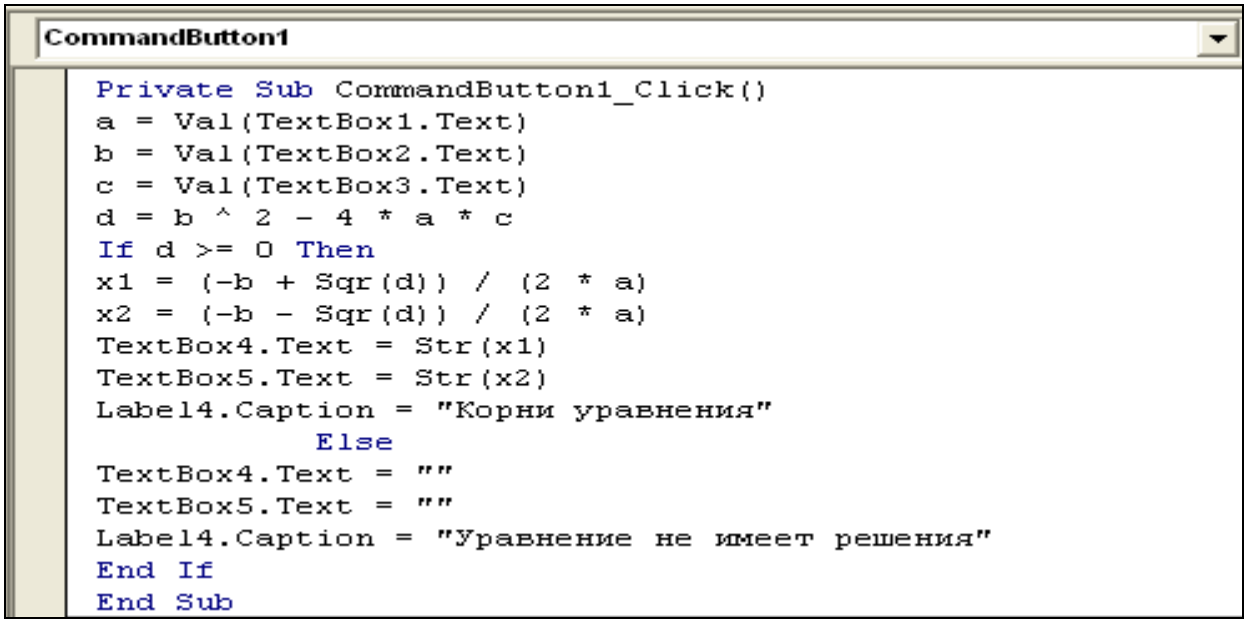


Рис. В.2. Экран системы программирования Visual Basic For Applications. Поиск корней квадратного уравнения

Как видно по приведенным иллюстрациям, операторы, используемые для анализа задачи и вычислений, полностью совпадают. Отличия имеются лишь в тех операторах, которые осуществляют ввод исходных данных и вывод полученных результатов, что обусловлено различиями между принципами алгоритмического программирования, применяемого в QuickBasic, и объектно-ориентированного программирования в VBA.

Необходимость создания этого учебного пособия обусловлена следующими обстоятельствами. Важной составной частью деятельности любого технического специалиста является создание, оформление, обработка различного рода деловой документации. В настоящее время основная часть документооборота осуществляется в документах, созданных в программе Microsoft Word. Microsoft Word фактически стала стандартной программой, используемой при работе с деловой документацией.

При этом актуальность приобретает вопрос о том, как сделать работу инженера, технического специалиста более производительной, эффективной, по возможности освободить его от рутинных операций по работе с текстом документов или облегчить их выполнение.

При решении этого вопроса незаменимым помощником может стать встроенная в Word система программирования VBA. Она позволяет, в частности, автоматизировать процесс создания типовых документов (отчетов, служебных записок и т.п.). Такие документы могут создаваться на основе шаблонов, дополненных необходимым программным кодом. В диалоговом режиме с помощью созданной в VBA формы пользователь может определить конкретную конфигурацию будущего документа.

Использование VBA позволяет обрабатывать не только документ в целом, но и работать с отдельными его частями, фрагментами. Примерами таких задач является автоматическая перекодировка русского текста, ошибочно набранного в английской раскладке клавиатуры (или обратная задача), приведение текста к заданному виду в соответствии с имеющимися стандартами и другие им подобные.

Помогает применение VBA и решению такой важной задачи, как корректное импортирование данных из различных баз данных в документы Microsoft Word. Но успешное решение всех перечисленных и многих других задач возможно только при том условии, что пользователь владеет базовыми навыками программирования в системе VBA применительно к Word.

Целью учебного пособия является обучение пользователей основам программирования в системе VBA на конкретных примерах, причем подробно рассматриваются все этапы работы над приложением, начиная с запуска системы программирования и заканчивая тестированием уже готового приложения. Проработка теоретического материала и овладение практическими навыками объектно-ориентированного и визуального программирования на основе приведенных примеров позволит студентам в дальнейшем успешно и эффективно применять полученные знания в своей профессиональной деятельности.

Глава 1. Работа в VBA

1.1. Основные сведения о языке Visual Basic и системе программирования VBA

Как видно по названию системы VBA (Visual Basic For Applications) , в ее основе лежит язык программирования BASIC. Этот язык имеет достаточно долгую и интересную историю развития, и перед тем, как мы приступим к дальнейшему изложению материала, необходимо сказать о ней несколько слов.

Язык BASIC в своем первоначальном варианте был разработан в середине 60-х гг. XX в. преподавателями Дартмутского колледжа в Канаде Джоном Кемени и Томасом Куртцем как язык для обучения основам программирования для начинающих. Название языка тоже представляет собой сокращение и расшифровывается следующим образом: Beginners All-purpose Symbolic Instruction Code. Перевод этого названия на русский язык: многоцелевой язык символьных инструкций для начинающих.

Этот язык получил широчайшее распространение во всем мире, и практика его использования показала, что его не только можно успешно использовать для обучения азам программирования (в том числе людей, вообще до этого не знакомых или малознакомых с вычислительной техникой), но и применять его для решения серьезных профессиональных задач в области программирования.

В 80-е гг. XX в. в связи с началом широкого внедрения в различные области человеческой деятельности персональных компьютеров (ПК) возник вопрос о разработке новых версий языка BASIC для ПК. Эта потребность была реализована путем создания специальных систем программирования для ПК на основе языка BASIC. В то время большинство персональных компьютеров работали под управлением операционной системы (ОС) MS DOS. Поэтому широкое распространение получили системы программирования QBASIC и QuickBasic, разработанные фирмой Microsoft на базе ОС MS DOS.

В 90-е гг. XX в. в связи с тем, что ведущей операционной системой для ПК стала ОС Windows, появилась потребность в массовой разработке программных продуктов на платформе Windows (эти программные продукты для краткости часто называют приложениями Windows).

Процесс разработки Windows-приложений в значительной степени облегчился, когда той же фирмой Microsoft была создана новая система программирования на основе языка Basic, работающая под управлением ОС Windows и предназначенная для разработки приложений в этой среде. Эта система получила название Visual Basic. Следует отметить, что появление Visual Basic привело не просто к механическому переносу языка на новую платформу, но ознаменовало собой новый этап в развитии программирования

на Basic — переход от алгоритмического программирования к объектно-ориентированному и визуальному программированию.

При алгоритмическом программировании в основе программы лежит алгоритм, т.е. последовательность четких, однозначных указаний, разработанных для решения поставленной задачи. Алгоритм задает жесткую схему работы программы, все операторы (инструкции), содержащиеся в программе, последовательно выполняются в раз и навсегда установленном порядке. Такой механизм работы программы когда-то был единственно возможным, но на современном этапе развития информационных технологий он является недостаточно гибким для решения многих возникающих задач.

Иначе обстоит дело при работе с системой объектно-ориентированного программирования. Приложение, разработанное в такой системе, содержит ряд объектов, причем в процессе взаимодействия с приложением пользователь может работать со всеми этими объектами, а может использовать в своих целях только часть из них.

Важной особенностью такой системы является также следующее: если при традиционном алгоритмическом программировании создание каждого нового объекта программы (экранной кнопки, текстового окна, надписи, переключателя, пункта меню) осуществляется путем написания программного кода, задающего тип объекта и его основные характеристики, то в системе объектно-ориентированного программирования существует набор заготовок, используемых для создания новых объектов. В Visual Basic для обозначения этого набора употребляется термин «панель элементов».

Для того чтобы создать новый объект, достаточно перетащить мышью нужную заготовку с панели элементов в окно, в котором создается интерфейс будущего приложения (в Visual Basic это окно называется пользовательской формой). В указанном месте на форме создается новый объект.

Таким образом, создается пользовательский интерфейс приложения, т.е. совокупность средств взаимодействия между приложением и пользователем. Создаваемые объекты являются элементами пользовательского интерфейса, а вышеописанный способ называется визуальным проектированием пользовательского интерфейса приложения.

Каждый из имеющихся в приложении объектов обладает определенным набором свойств. Примерами свойств являются геометрические размеры объекта, положение объекта на экране, цвет объекта, содержание текста объекта (если таковой имеется), характеристики шрифта, которым написан этот текст, и др. Начальная настройка свойств объекта, как правило, производится в диалоговом режиме. В ходе работы программы свойства объекта могут неоднократно изменяться.

Каждый из объектов может реагировать на определенные внешние воздействия, которые называются событиями. События могут быть вызваны действиями пользователя (нажатие какой-либо клавиши на клавиатуре ПК, наведение мыши на объект, нажатие одной из кнопок мыши, перетаскивание объекта с помощью мыши и т.д.), а могут быть инициированы самой программой. Реакция каждого объекта на определенное событие, приводящее

к изменению его свойств, описывается отдельной подпрограммой (процедурой), входящей в основной программный модуль.

Таким образом, создание приложения в системе объектно-ориентированного программирования происходит в три этапа (создание пользовательского интерфейса приложения, настройка свойств объектов, написание процедур), причем непосредственно написанием программного кода разработчик приложения должен заниматься только на последнем, заключительном этапе работы. Эта технология разработки программного обеспечения на сегодняшний день является передовой и широко используется при разработке Windows-приложений.

Следующим шагом в развитии современных инструментальных систем (систем программирования) стала интеграция системы объектно-ориентированного и визуального программирования Visual Basic с офисным программным пакетом Microsoft Office. Этот шаг не только обеспечил пользователя готовой системой программирования при установке пакета Microsoft Office, но и предоставил ему ряд новых возможностей.

Эти возможности связаны с внутренней структурой самого пакета Microsoft Office. Данный пакет предназначен для решения разнообразных задач по обработке информации: от создания простых текстовых документов и электронных таблиц до полной автоматизации работы с документами с использованием систем управления базами данных. Настройка программ, входящих в пакет Microsoft Office, может производиться пользователем различными способами: как в диалоговом режиме, так и с использованием различных программных средств, важнейшим из которых является система программирования VBA.

Система VBA позволяет создавать приложения в любой из программ, входящих в состав пакета Microsoft Office, причем используемая версия языка Visual Basic в последних версиях пакета является общей для всех программ. VBA позволяет работать с большим количеством различных объектов, к которым относятся не только элементы графического интерфейса пользователя (экранные кнопки, текстовые поля, переключатели, флажки), но и документы, создаваемые различными приложениями Microsoft Office, а также различные элементы этих документов.

Например, система VBA может работать с такими объектами программы Microsoft Word, как само приложение Word, документ, окно документа, абзац, таблица, шаблон документа, стиль форматирования, шрифт документа. Важным преимуществом VBA является возможность объединять различные программы пакета Microsoft Office для решения практически любых задач по комплексной обработке информации.

Из всего вышеизложенного следует, что изучение системы программирования VBA является важным, а во многих случаях и необходимым компонентом в освоении современных информационных технологий будущими техническими специалистами.

1.2. Создание макроса в диалоговом режиме

Как уже было указано во вводной части пособия, средства программного пакета Microsoft Office позволяют создавать простые программы без написания программного кода. Для того чтобы создать такую программу, пользователю следует один раз выполнить последовательность команд и инструкций, входящих в нее, и сохранить ее под определенным именем. Такой набор команд и инструкций в терминологии Microsoft Office называется макрокомандой или макросом, а средство, используемое для выполнения этой операции – макрорекордером. Созданный один раз макрос впоследствии можно многократно выполнять.

Возможность создания и использования макросов существует в различных приложениях Microsoft Office: Microsoft Word, Microsoft Excel, Microsoft Access, Microsoft PowerPoint. В данном пособии мы рассмотрим процесс разработки макроса на примере программы Word (процессы создания макросов в различных приложениях Microsoft Office имеют некоторые несущественные отличия, но общие принципы работы с макросами во всех приложениях Microsoft Office остаются неизменными).

Создание макроса начнем с того, что любым доступным в ОС Windows способом запустим программу Word. Далее в этой программе следует открыть меню **Сервис**, в нем вложенное подменю **Макрос**, а в этом подменю запустить на выполнение команду «**Начать запись…**» (рис. 1.1).

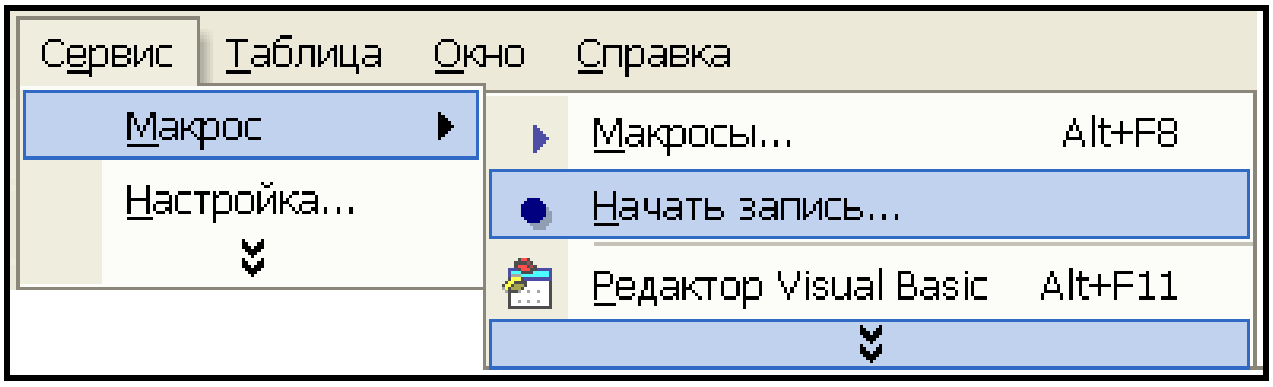


Рис. 1.1. Начало создания макроса в программе Microsoft Word

В программе Microsoft Word существует и другой способ выполнить те же действия — достаточно дважды щелкнуть мышью на индикаторе **ЗАП**, который находится в строке состояния, расположенной в нижней части окна приложения. По умолчанию индикатор **ЗАП** закрашен бледно-серым цветом, т.е. он является неактивным. После двойного щелчка мышью индикатор окрашивается в черный цвет. Это изменение цвета говорит об активизации данного индикатора и включении макрорекордера.

Выполнение указанной последовательности команд или двойной щелчок мышью на индикаторе **ЗАП** открывает диалоговое окно «**Запись**

макроса», в котором пользователю предлагается присвоить имя создаваемому макросу. Изображение данного окна приведено на рис. 1.2.

По умолчанию первый создаваемый пользователем макрос получает имя **Макрос1**, следующий **Макрос2** и т.д. Понятно, что такие имена макросов малоинформативны, поэтому лучше в поле «**Имя макроса»**, находящееся в верхней части окна, ввести имя, которое бы отражало содержание данного макроса. Например, если мы хотим создать макрокоманду, которая выделяет слово, на котором установлен курсор, одновременно полужирным шрифтом и курсивом, то можно дать ей имя **Жирн_Курс**. Обратите внимание на то обстоятельство, что в имени макроса недопустимы пробелы. В том случае, если имя макроса состоит из двух или более слов (как в нашем примере), пробелы нужно заменять символом подчеркивания.

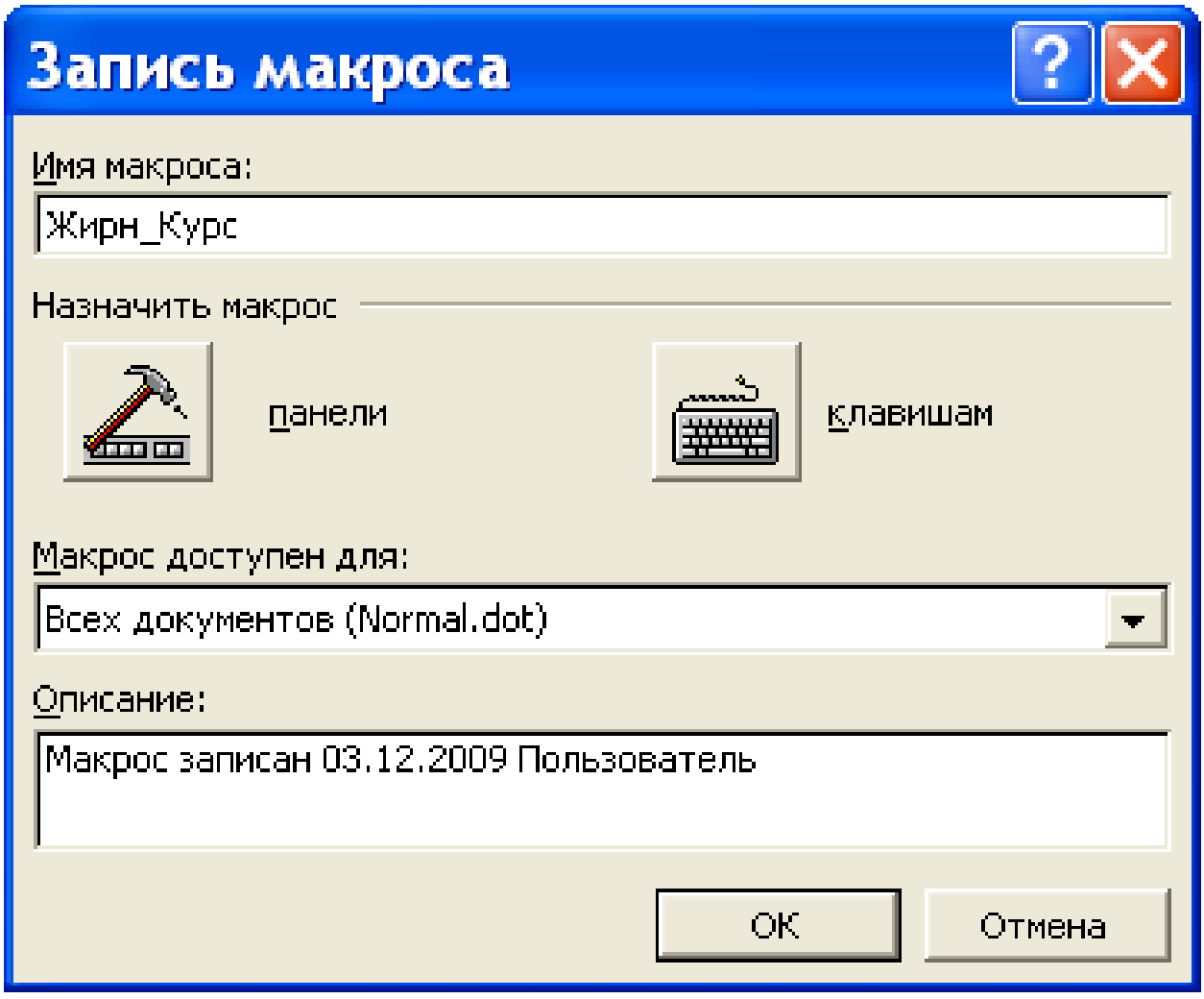


Рис. 1.2. Диалоговое окно «**Запись макроса**»

В этом диалоговом окне (в его центральной части) пользователю также предлагается назначить макрос панели (т.е. создать кнопку на панели инструментов для запуска макроса) или назначить макросу комбинацию клавиш, которая его выполняет. Выберем вариант с назначением комбинации

клавиш. При нажатии экранной кнопки с изображением клавиатуры открывается еще одно дополнительное диалоговое окно «**Настройка клавиатуры**», внешний вид которого показан на рис.1.3.

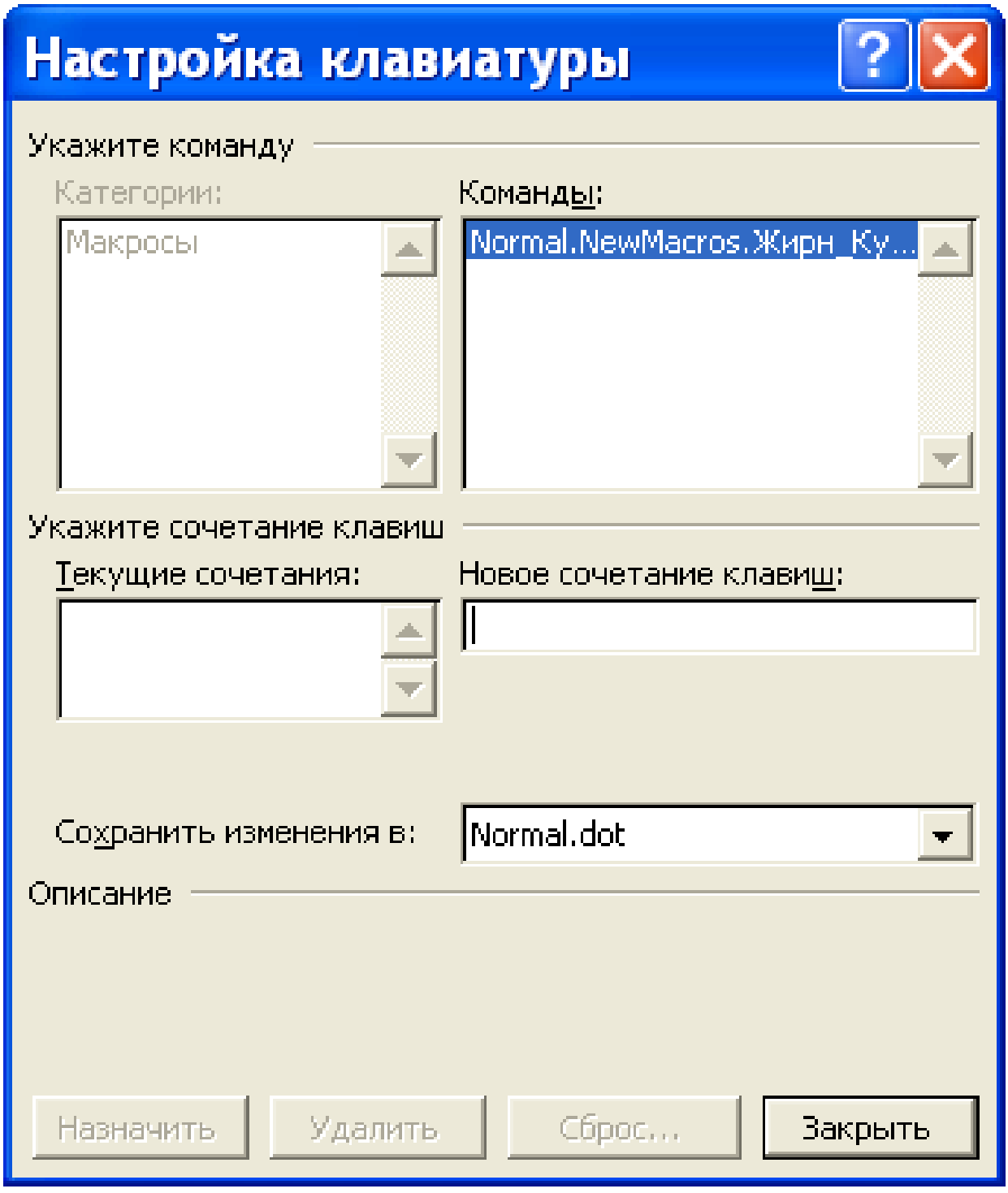


Рис. 1.3. Диалоговое окно «**Настройка клавиатуры**» до ввода комбинации клавиш

Сразу после открытия окна поле **Новое сочетание клавиш** является пустым. В этом поле следует указать то сочетание клавиш, которое запускает макрос на выполнение. Необходимо ввести комбинацию из двух

клавиш, одной из которых может быть только специальная клавиша **Ctrl** или **Alt**, а другой – любая алфавитно-цифровая клавиша. Следует обратить внимание на то, что для заполнения поля **Новое сочетание клавиш** не нужно вручную набирать на клавиатуре название специальной клавиши. Достаточно просто нажать клавишу **Ctrl** или **Alt** и, удерживая эту клавишу в нажатом состоянии, другой рукой нажать алфавитно-цифровую клавишу. Когда пользователь отпустит обе клавиши, поле **Новое сочетание клавиш** будет заполнено.

После ввода комбинации клавиш в нижней части окна активизируется кнопка «**Назначить**», которая до этого была неактивной. Щелчком по кнопке «**Назначить**» за данным макросом окончательно закрепляется введенное сочетание клавиш. В примере, показанном на рис. 1.4, такой комбинацией является сочетание клавиш **Alt + ю**. Затем следует нажать на экранную кнопку «**Закрыть**», после чего начинается непосредственно процесс записи команд, входящих в макрос.

В программе Microsoft Word в том, что начался процесс записи макроса, можно убедиться по трем признакам. Во-первых, в строке состояния, расположенной в нижней части экрана, становится активным индикатор **ЗАП** (активный индикатор окрашивается в черный цвет).

Во-вторых, изменяется внешний вид указателя мыши. Рядом со стрелкой указателя мыши появляется изображение магнитофонной кассеты. Теперь необходимо выполнить все действия, которые должны войти в состав макрокоманды. При выполнении этих действий нужно обратить внимание на следующее обстоятельство: действия, выполненные в окне документа с помощью мыши, не записываются. Для записи таких действий, как перемещение текстового курсора, а также выделение, копирование и перемещение текста, необходимо использовать не мышь, а клавиатуру. Эти действия удобнее всего выполнять в специальном режиме выделения, имеющемся в программе Word.

Например, в нашем примере (выделение заданного слова полужирным шрифтом и курсивом) следует выполнить следующие действия:

1. Нажать клавишу **F8** для активизации режима выделения. Режим выделения можно также активизировать двойным щелчком мыши на индикаторе **ВДЛ**, находящемся в строке состояния программы Word. Затем нажатием клавиши **F8** выделить текущее слово. Нажать клавишу **Esc** для выхода из режима выделения.

2. Комбинацией клавиш **Ctrl+B** (если активным языком является английский) или **Ctrl+И** (если активным языком является русский) задать полужирный шрифт для выделенного слова.

3. Комбинацией клавиш **Ctrl+I** (если активным языком является английский) или **Ctrl+Ш** (если активным языком является русский) задать курсивное начертание для выделенного слова.

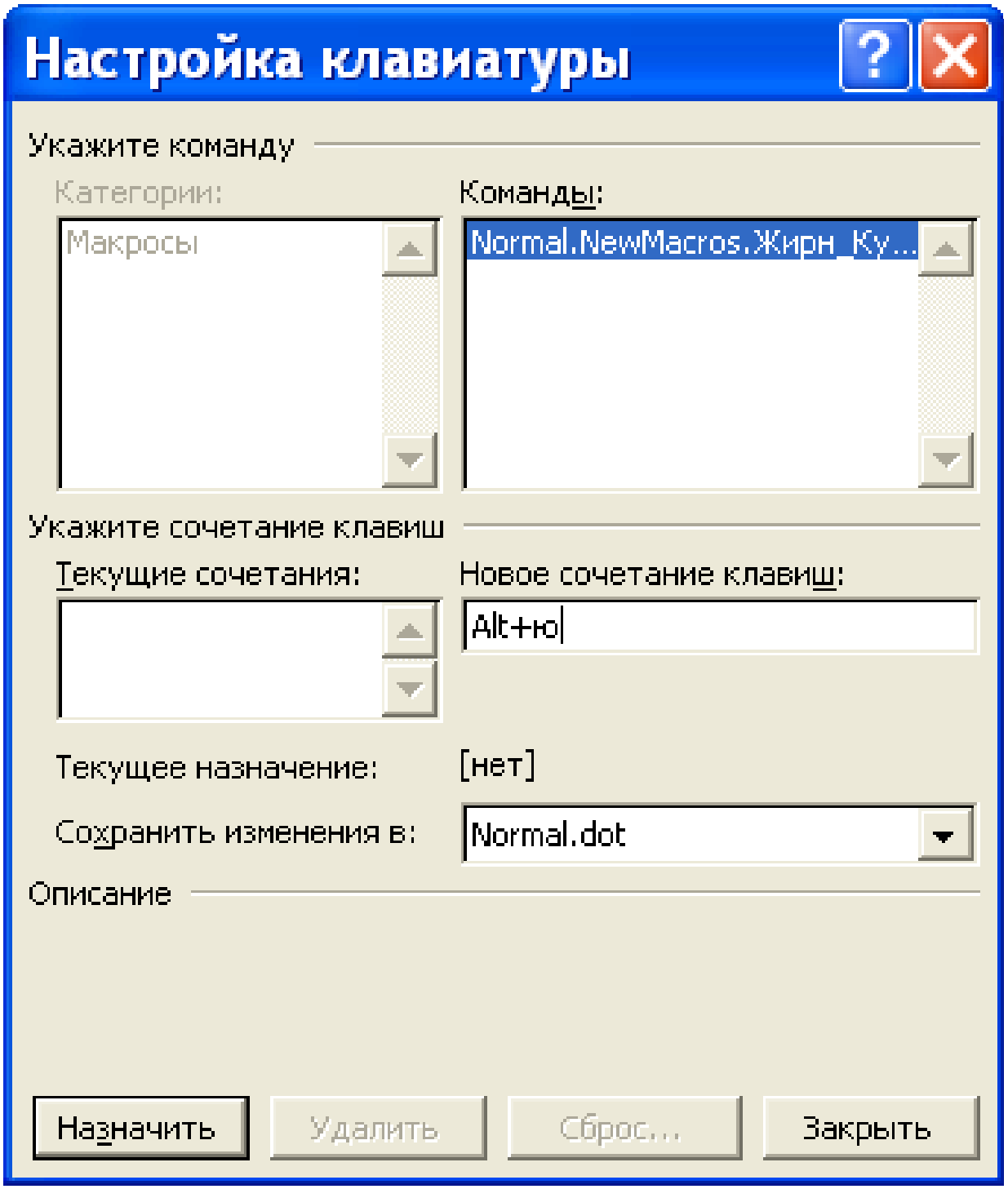


Рис. 1.4. Диалоговое окно «**Настройка клавиатуры**» после ввода комбинации клавиш

Третьим признаком того, что идет процесс записи, является появление в окне приложения дополнительной панели, используемой для управления процессом записи. Она содержит всего две кнопки: кнопка «**Пауза**» используется в том случае, если пользователю нужно выполнить какие-либо действия, не входящие в макрос. Для возобновления записи макроса нужно повторно нажать на кнопку «**Пауза**». Кнопка «**Остановить запись**»

применяется в том случае, если все действия, входящие в макрос, выполнены, и нужно завершить его запись. После нажатия на кнопку «**Остановить запись**» процесс создания макроса завершен.

В дальнейшем готовый макрос можно запускать на выполнение двумя способами. Первый способ — это запуск макроса через меню. Для этого нужно открыть в приложении меню **Сервис**, в нем подменю **Макрос**, а в этом подменю щелкнуть мышью команду **Макросы…**. В открывшемся диалоговом окне нужно выбрать из списка название макроса, а затем щелкнуть экранную кнопку «**Выполнить**». Понятно, что описанный выше способ является недостаточно быстрым, поэтому гораздо удобнее воспользоваться вторым способом, т.е. нажать ту комбинацию из двух клавиш, которая была ранее задана в диалоговом окне «**Настройка клавиатуры**». Для приведенного нами примера нажатие комбинации клавиш **Ctrl+ю** будет выделять в тексте документа текущее слово полужирным и курсивным шрифтом.

Возможна ситуация, когда при повторном открытии документа правильно записанный макрос не выполняется. Для того чтобы решить данную проблему, следует воспользоваться последовательностью команд **Сервис → Макрос → Безопасность.** Она открывает диалоговое окно, в котором можно выбрать уровень безопасности для данного документа. Следует установить средний уровень безопасности, который позволяет пользователю при открытии документа самостоятельно решить вопрос о том, можно ли запускать на выполнение имеющиеся в документе макросы.

Если пользователь удовлетворен полученным результатом, то в дальнейшем он может многократно использовать созданный им макрос для выполнения требуемой операции. Но если появляется необходимость каким-либо образом видоизменить имеющийся макрос, то произвести желаемые изменения в диалоговом режиме (как при создании макроса) пользователь не сможет. Для внесения изменений следует снова использовать в меню последовательность команд **Сервис → Макрос → Макросы…** . Выполнение этой последовательности открывает уже знакомое нам диалоговое окно **«Макрос»**. В данном диалоговом окне нужно щелкнуть мышью экранную кнопку «**Изменить**». Щелчок по экранной кнопке открывает окно для редактирования программного кода в Visual Basic.

Но такого рода редактирование требует знания как синтаксиса языка Visual Basic, так и основных принципов объектно-ориентированного программирования. Причем такие знания помогут пользователю не только откорректировать уже существующий макрос, но и создать новую программу с оригинальным пользовательским интерфейсом. Изучению этих вопросов и посвящены следующие разделы.

1.3. Экран системы VBA и его основные элементы

Пользователь может приступить к программированию в системе VBA двумя различными способами.

Первый из них заключается в том, что в одном из приложений пакета Microsoft Office создается какой-либо документ, а затем непосредственно в документе конструируется графический интерфейс пользователя, который затем дополняется программным кодом, описывающим реакцию элементов интерфейса (объектов будущей программы) на различные программные события. Для того чтобы начать создание элементов интерфейса, достаточно использовать последовательность команд **Вид** → **Элементы управления**. В результате на экране появляется дополнительная панель «**Элементы управления».** Она содержит ряд экранных кнопок, соответствующих создаваемым объектам.

Однако недостаток такого способа заключается в том, что у пользователя могут возникать трудности, связанные с позиционированием объектов в документе, т.е. размещением объектов в нужном месте рабочей области документа. В ряде случаев просто перетаскивать объекты с помощью мыши по рабочей области документа не получается, и для того, чтобы поместить объект там, где это желательно разработчику программы или изменить положение уже имеющегося объекта, приходится многократно нажимать клавишу **Пробел**.

Поэтому более удобным и рациональным представляется другой способ. Этот второй способ заключается в том, что пользователь открывает среду программирования VBA, создает в этой среде экранную форму, на которой размещает элементы интерфейса, а затем приступает к настройке свойств этих элементов и написанию программного кода. Для того чтобы открыть среду программирования VBA, следует в меню Microsoft Word использовать последовательность команд **Сервис** → **Макрос** → **Редактор Visual Basic** (рис. 1.5).

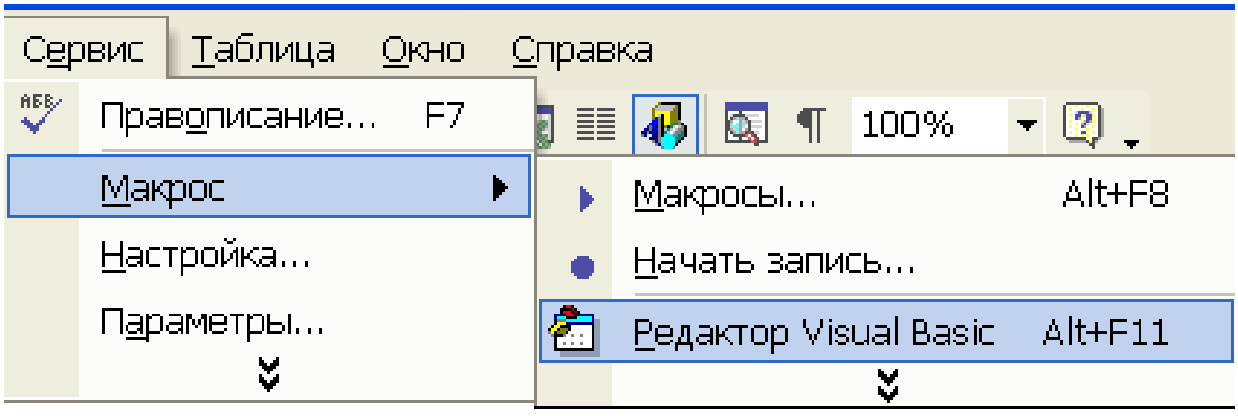


Рис. 1.5. Открытие системы программирования VBA в программе Microsoft Word

В результате выполнения этой последовательности команд на экране компьютера появляется экран системы программирования VBA. Перед тем как приступить к рассмотрению дальнейших действий по созданию приложения в этой системе, рассмотрим основные элементы экрана системы VBA, которые показаны на рис. 1.6.

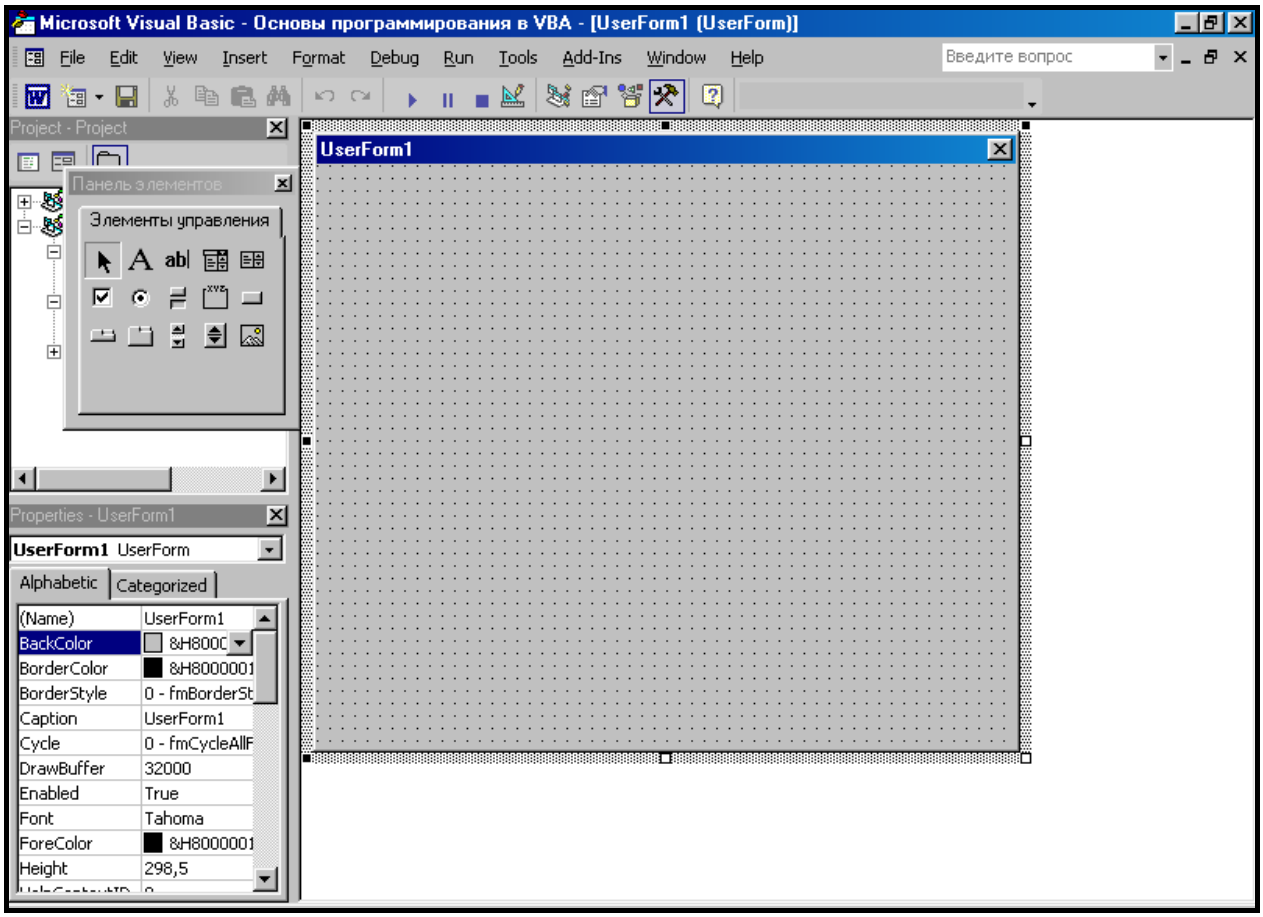


Рис. 1.6. Экран системы программирования VBA

В верхней части экрана располагается строка заголовка, содержащая название среды программирования, в которой работает пользователь – Microsoft Visual Basic. Эта строка также содержит название того документа (файла Microsoft Word или Microsoft Excel), из которого была запущена система программирования VBA.

В правой части строки заголовка, как и у большинства других приложений Windows находятся три экранные кнопки: сворачивающая, разворачивающая (восстанавливающая) и закрывающая. Следует учитывать, что щелчок мышью по закрывающей кнопке не приведет к закрытию документа Microsoft Word или Microsoft Excel, а закроет среду программирования VBA с возвращением в окно этого документа.

Под строкой заголовка располагается строка меню, содержащая ряд ниспадающих меню. Каждое такое меню содержит ряд пунктов, идущих один под другим по вертикали. Каждому пункту соответствует определенная

команда системы программирования. Если эта команда активна (т.е. она может быть выполнена в данный момент времени), то ее название в меню написано черным цветом. Если же команда неактивна (т.е. в данный момент она невыполнима), то ее название в меню написано бледно-серым цветом.

Если справа от названия пункта меню находится многоточие, то данный пункт открывает диалоговое окно, в котором необходимо ввести дополнительные параметры для выполнения соответствующей ему команды. Если же справа от названия пункта меню имеется черная треугольная стрелка, то этот пункт открывает вложенное подменю (меню второго уровня), также содержащее ряд команд.

Под строкой меню располагается панель инструментов (в нерусифицированных версиях VBA она называется **Toolbar**). Эта панель содержит ряд кнопок, которые дублируют основные, наиболее часто употребляемые команды меню. На каждой кнопке имеется пиктограмма – небольшой рисунок, который наглядно поясняет назначение данной кнопки.

Существует и еще один (кроме изучения пиктограмм) способ выяснить функцию какой-либо кнопки. Для того чтобы узнать назначение той или иной кнопки, достаточно подвести к ней указатель мыши и задержать указатель на несколько секунд. Появляется всплывающая подсказка, которая кратко (на английском языке) характеризует назначение данной кнопки.

Например, кнопка с изображением дискеты выполняет функцию аналогичную команде **Save** (**Сохранить**) из меню **File** (**Файл**). Кнопка с изображением ножниц аналогична команде **Cut** (**Вырезать**) из меню **Edit** (**Правка**). Кнопка с синей треугольной стрелкой дублирует команду **Run** (**Запустить на выполнение**) из одноименного меню **Run** и так далее.

Центральную часть экрана в системе программирования VBA занимает основная пользовательская форма (**UserForm**). Данная форма представляет собой окно, в котором размещаются элементы пользовательского интерфейса программы (надписи, экранные кнопки, текстовые окна, флажки, переключатели и другие). Размеры элементов интерфейса, а также самой экранной формы, можно изменять вручную с помощью перетаскивания границы объекта мышью. По краям формы расположены специальные маркеры (небольшие белые квадратики), «ухватившись» указателем мыши за которые, можно изменять размеры формы по вертикали, горизонтали или по диагонали (т.е. одновременно и по вертикали, и по горизонтали). Существует и другой способ изменения размеров формы и ее элементов (с помощью окна свойств) который будет описан ниже.

Для того чтобы разработчику было удобнее позиционировать элементы интерфейса, на форме имеется специальная пунктирная сетка. Эта сетка отчетливо видна, когда программа находится в режиме разработки. Когда пользователь, нажав на треугольную стрелку, запускает программу на выполнение, пунктирная сетка сразу же исчезает и поверхность формы становится однородной. Такое изменение говорит о том, что программа перешла в другой режим – режим выполнения.

Необходимо также отметить следующее: при запуске системы программирования VBA пользовательская форма не появляется автоматически на экране. Для того чтобы создать форму, необходимо воспользоваться меню **Insert** (**Вставка**). В данном меню содержится список объектов, которые может создать система. В списке следует выбрать объект **UserForm** (**Пользовательская форма**) который затем появится в центре экрана.

Слева от основной пользовательской формы находится окно «**Панель элементов**» (в нерусифицированной версии системы – **Toolbox**). Данное окно содержит набор кнопок, используемых для создания элементов интерфейса. На каждой кнопке (как и на кнопках панели инструментов) имеется пиктограмма, а также при наведении на кнопку указателя мыши появляется всплывающая подсказка.

Например, кнопка с изображением прописной буквы **A** применяется для создания объекта «**Надпись**» (**Label**). Если на кнопке изображены буквы **ab** и вертикальный курсор, то с ее помощью можно создать объект «**Текстовое окно**» (**Textbox**). Щелчок на кнопке с изображением серого прямоугольника создает объект «**Экранная кнопка**» (**CommandButton**). Кнопка, на которой изображен кружок с точкой в центре, применяется для создания объекта «**Переключатель**» (**OptionButton**). Кнопка, на которой изображена «галочка», используется для создания объекта «**Флажок**» (**Checkbox**).

Если **Панель элементов** не видна на экране, то следует щелкнуть мышью на любом свободном месте пользовательском формы, и данная панель сразу же появится. **Панель элементов** жестко не закреплена на экране, поэтому в случае необходимости ее можно перемещать по экрану, «ухватившись» мышью за строку заголовка.

Под **Панелью элементов** расположено **Окно свойств** (**Properties**). В данном окне содержится список свойств того объекта, находящегося на пользовательской форме, который является активным в настоящий момент времени. Для того чтобы сделать какой-либо объект на форме активным, достаточно щелкнуть по нему мышью. Если на форме не активизирован не один объект, то данное окно отображает свойства самой пользовательской формы.

В **Окне свойств** содержатся две вкладки, в каждой из которых свойства объекта упорядочены определенным образом. В левой вкладке свойства упорядочены по алфавиту, в правой – по категориям. Примерами свойств объекта являются длина (**Height**) и ширина (**Width**) объекта, расположение объекта относительно левой границы окна (**Left**) и верхней его границы (**Top**), фоновый цвет объекта (**BackColor**) и цвет текста, имеющегося на объекте (**ForeColor**). Указанные свойства присущи большинству объектов, но разные объекты могут отличаться как по набору свойств, так и по значениям этих свойств.

Справа от названия каждого свойства находится поле, в котором содержится значение данного свойства. Ряд свойств, характеризующих геометрические размеры объекта и его положение на экране, имеет численное значение, выраженное в пикселях. Для изменения значений этих объектов следует вводить новое значение с клавиатуры.

Например, если ширина созданной пользователем экранной кнопки равна 50 пикселям, и ее не хватает для размещения того текста, который должен быть на этой кнопке, необходимо сделать следующее: выделить данную кнопку мышью, найти свойство **Width** (ширина данной кнопки) и вместо имеющегося значения 50 ввести значение 100. В результате ширина кнопки увеличится вдвое (до 100 пикселей), и места для размещения текста теперь должно хватить (рис. 1.7).

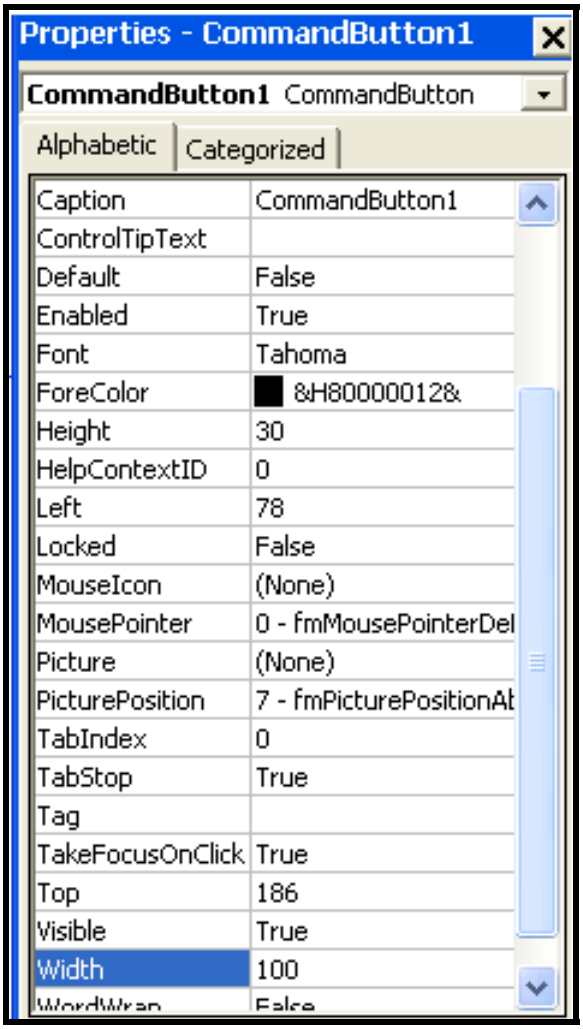


Рис. 1.7. Окно свойств объекта. Настройка ширины экранной кнопки

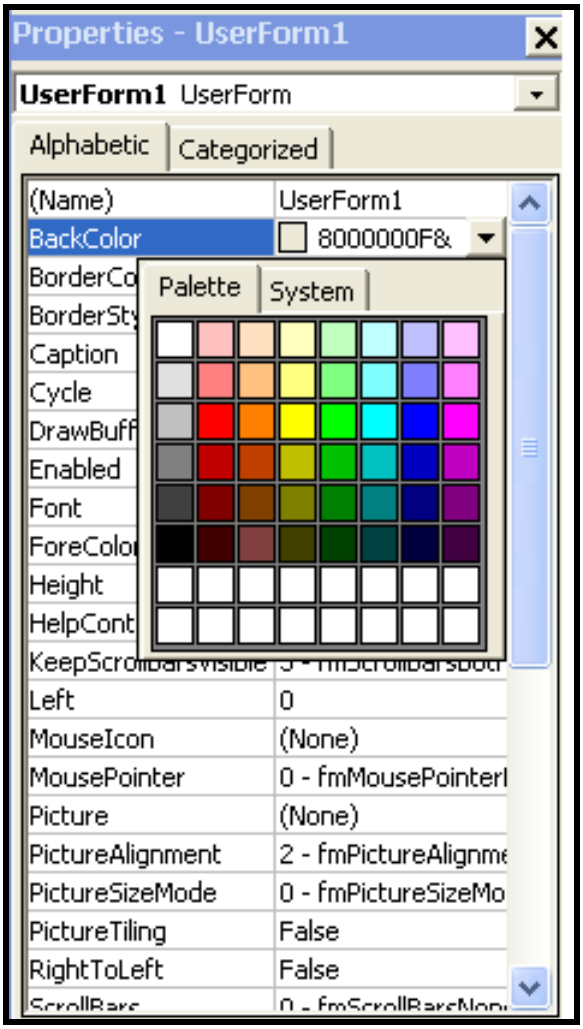


Рис. 1.8. Окно свойств объекта. Выбор цвета экранной формы

Некоторые свойства имеют строго фиксированный набор значений, и тогда конкретное значение нужно выбирать из раскрывающегося списка. Например, свойство «**Видимость**» (**Visible**) может принимать всего два

значения: **True** (истина) или **False** (ложь). В первом случае после запуска программы на выполнение объект будет виден пользователю, во втором – объект останется невидимым для пользователя (при этом в режиме проектирования объект будет все время виден на форме вне зависимости от значения данного свойства).

Если значение свойства нужно выбирать из списка, то после щелчка мышью в правом поле появляется треугольная стрелка, направленная вниз. Щелчок на этой раскрывающей стрелке открывает весь список значений. При выборе цвета объекта (свойства **BackColor** и **ForeColor**) щелчок на раскрывающей кнопке открывает палитру, из которой можно выбрать подходящий цвет (рис. 1.8).

Для некоторых свойств в поле справа от его названия появляется кнопка, на которой имеется изображение многоточия. Такая кнопка (она называется кнопка-построитель) открывает дополнительное диалоговое окно. Такая кнопка-построитель имеется, например, в поле справа от свойства **Font**. Свойство **Font** определяет характеристики шрифта, которым написан текст (если таковой имеется на объекте). Щелчок на кнопке-построителе открывает диалоговое окно «**Шрифт**», которое позволяет изменить размер шрифта, выраженный в пунктах, гарнитуру (вид шрифта), а также выбрать начертание шрифта (обычный, жирный, курсив или жирный курсив).

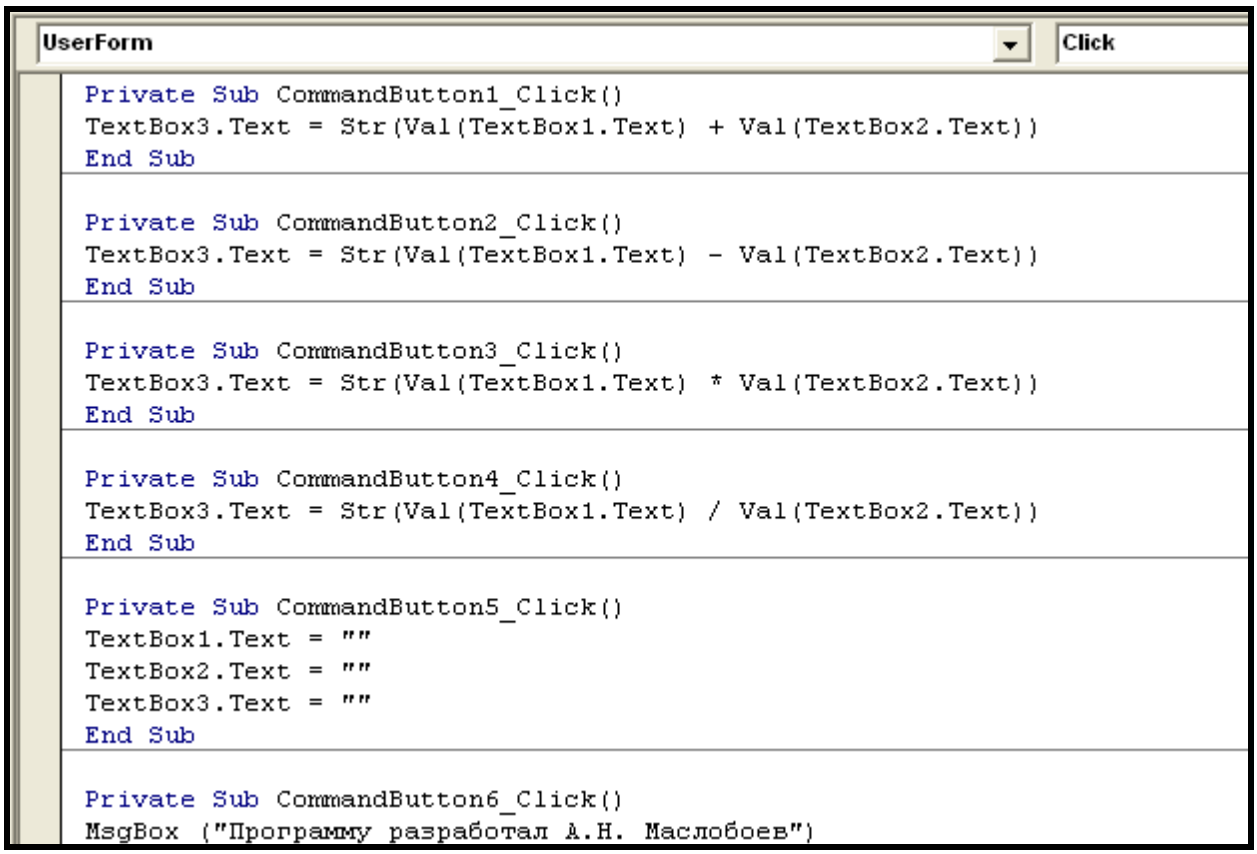


Рис. 1.9. Окно кода системы программирования VBA

Сверху от окна свойств в левом верхнем углу экрана находится окно проекта (**Project**), в котором содержится перечень имеющихся в проекте экранных форм, программных модулей и других элементов проекта.

После того, как пользователь создал все необходимые объекты и произвел с помощью **Окна свойств** их первоначальную настройку, необходимо написать для данных объектов программный код. Для этого нужно вывести на экран окно программного кода, которое показано на рис.1.9. В отличие от других элементов системы VBA, данное окно не появляется сразу при входе в систему.

Существует два способа вызова окна кода. Первый способ заключается в следующем: после создания какого-либо объекта и настройки его свойств следует дважды щелкнуть мышью этот объект, и на экране автоматически появляется окно программного кода с заготовкой для процедуры, соответствующей данному объекту. Далее в окне можно вводить программный код для этого объекта.

Второй способ применяется преимущественно для корректировки уже имеющегося программного кода и заключается в следующем: в меню системы программирования нужно щелчком мыши открыть раздел **View** (Вид) и в этом разделе щелкнуть пункт **Code**. Затем можно работать с окном кода так же, как и в первом варианте.

Окно кода в рабочем состоянии занимает большую часть экрана и закрывает основную экранную форму. После того, как все необходимые строки программного кода введены, пользователь может вернуться к основному экрану системы программирования. Для этого нужно снова открыть меню **View** и щелкнуть мышью в этом меню команду **Object**. Выполнение этой команды приводит к тому, что на экране вновь появляется основная экранная форма, и экран системы программирования приобретает свой обычный вид.

После завершения работы над проектом пользователь может вернуться в основное окно того документа, из которого была запущена система программирования VBA. Это можно сделать двумя способами: либо щелкнуть закрывающую кнопку в строке заголовка системы программирования, либо воспользоваться меню. Для этого в меню системы нужно открыть раздел **File** и в данном разделе использовать команду «**Close and return to Microsoft Word**».

После ознакомления с основными элементами интерфейса системы программирования VBA пользователь может приступить к разработке первого собственного приложения, что и будет рассмотрено в разд. 1.4.

1.4. Разработка первого проекта в VBA

Первый проект, который мы создадим в системе VBA, должен выводить на экран пользовательскую форму с надписью: «Я программирую в системе VBA». Кроме надписи, на основной пользовательской форме должны присутствовать две экранные кнопки. Одиночный щелчок левой кнопкой мыши на одной из этих экранных кнопок должен выводить в нижней части формы вторую надпись со сведениями об авторе программы. При двойном щелчке мышью на этой же кнопке надпись со сведениями об авторе должна исчезнуть. Вторая экранная кнопка нужна для того, чтобы закрыть основную экранную форму и завершить работу данной программы.

Работу над первым проектом начнем с того, что создадим документ Microsoft Word, который так и назовем: **Первая программа.doc**. Затем необходимо открыть этот документ и войти в систему программирования VBA с помощью последовательности команд меню **Сервис** → **Макрос** → **Редактор Visual Basic** (как было указано в разд. 1.3)**.** Выполнение этой последовательности команд открывает систему программирования VBA.

Далее в системе VBA необходимо создать экранную форму с помощью команд меню **Insert** → **UserForm**. Пользовательская форма является таким же объектом программы, как и все элементы интерфейса, которые на ней расположены. Поэтому мы можем менять свойства формы, используя окно свойств **Properties**.

Окно формы имеет строку заголовка, и одним из свойств формы является содержимое текста, находящегося в строке заголовка. По умолчанию в этой строке выводится имя основной формы – **UserForm1**. Для того чтобы изменить содержание текста в строке заголовка, сделав его более информативным, обратимся к окну свойств **Properties**. Так как на форме в данный момент нет еще ни одного объекта, то окно **Properties** показывает свойства самой формы.

Найдем в этом окне свойство **Caption**, которое и отвечает за содержимое текста в строке заголовка формы. Щелкнем мышью в поле, находящемся справа от названия свойства. В этом поле появляется вертикальный курсор, наличие которого говорит о том, что данное поле можно редактировать. Удалим находящееся в этом поле значение **UserForm1** и вместо него введем с клавиатуры следующее значение: **Первая программа в VBA.** Этот же текст теперь отобразится и в строке заголовка экранной формы.

Следующим шагом в настройке свойств основной формы является изменение фонового цвета самой формы. По умолчанию цвет основной формы является светло-серым. Для изменения цвета следует в окне **Properties** найти свойство **BackColor**, которое отвечает за изменение фонового цвета формы. Щелкнув по раскрывающей стрелке, находящейся в

поле справа от названия свойства, можно открыть палитру для выбора другого цвета формы. Следует обратить внимание на то, что открывающаяся палитра имеет две вкладки: **System** (системная) и **Palette** (собственно палитра). По умолчанию открывается вкладка **System**, которая содержит ограниченный набор цветов. Переключившись щелчком мыши на вкладку **Palette**, пользователь открывает палитру с бóльшим количеством цветов. Для нашего проекта выбираем в этой палитре светло-зеленый цвет, который и становится фоновым цветом формы.

После настройки свойств формы создаем на ней объект **Label** (**Надпись**), для чего нужно в **Панели элементов** щелкнуть кнопку с прописной буквой **A**. Появившийся на экранной форме объект получает по умолчанию имя **Label1**. Для данного объекта также необходимо изменить содержание текста надписи. Находим в окне **Properties** объекта **Label1** свойство **Caption** и вместо стоящего там по умолчанию значения «**Label1**» вводим новый текст: «**Я программирую в системе VBA**».

Затем настраиваем шрифт надписи. Для этого находим в окне **Properties** свойство **Font** и щелчком на кнопке-построителе открываем диалоговое окно «**Шрифт**». В диалоговом окне задаем размер шрифта – 18 пунктов и начертание шрифта – жирный.

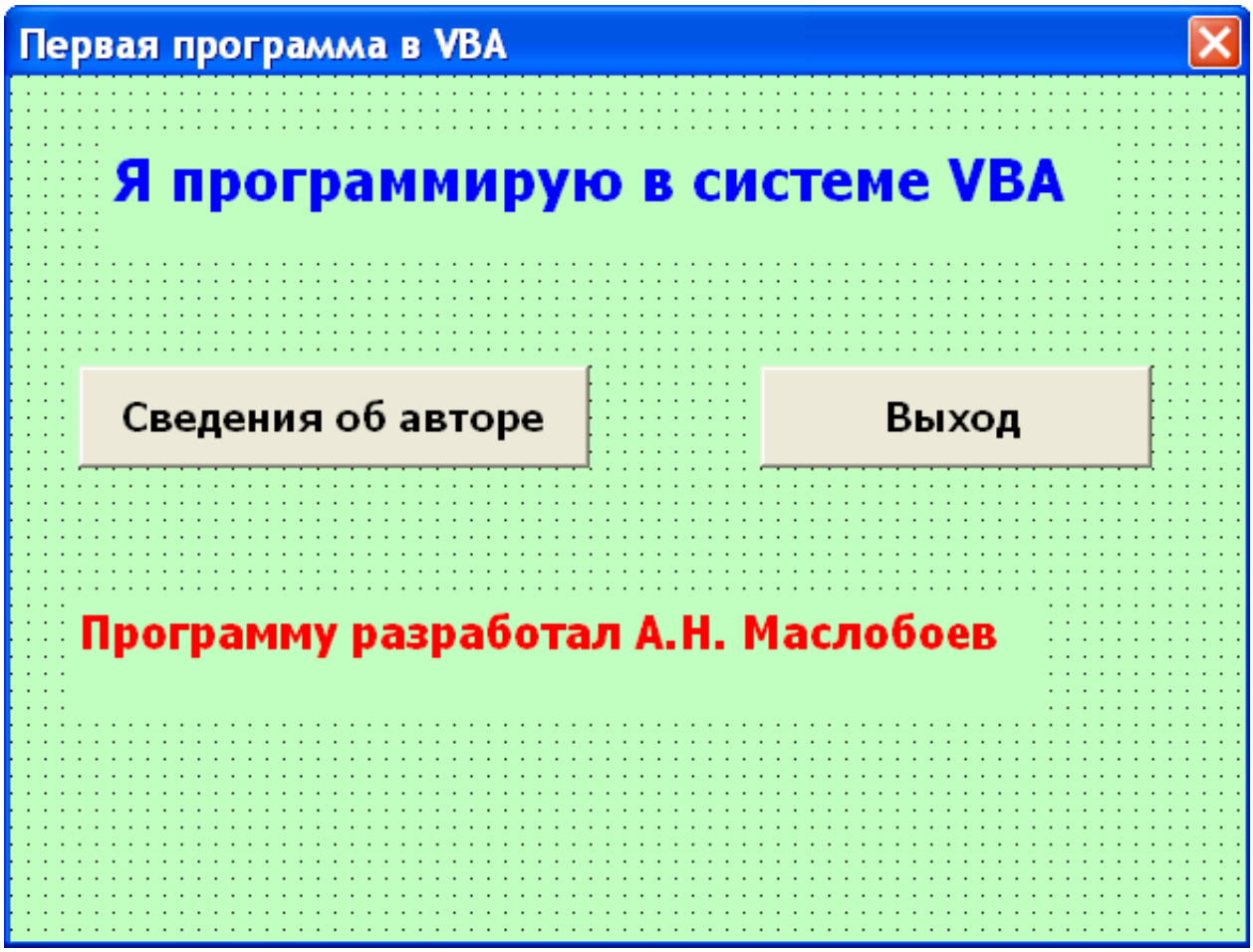


Рис. 1.10. Пользовательский интерфейс первой программы

Далее устанавливаем цвет шрифта надписи. Цвет шрифта определяется свойством **ForeColor**. Для изменения цвета шрифта находим это свойство в окне **Properties**. Затем в поле справа от свойства **ForeColor** щелкаем раскрывающую стрелку. В результате этих действий открывается цветовая палитра, аналогичная такой же палитре для свойства **BackColor**. В открывшейся палитре переходим на вкладку **Palette** и выбираем на ней цвет шрифта (в нашем проекте – синий). На этом настройку свойств первой надписи можно считать завершенной.

Затем создаем на экранной форме с помощью **Панели элементов** две экранные кнопки – **CommandButton1** и **CommandButton2**. Для кнопки **CommandButton1** в окне **Properties** находим свойство **Caption** и вводим следующее содержимое текста: «**Сведения об авторе**». На кнопке **CommandButton2** создаем текст, состоящий из одного слова: «**Выход**». Для обеих кнопок с помощью свойства **Font** устанавливаем размер шрифта, равный 14 пунктам. Цвет шрифта мы в данном случае не меняем, так как он все равно останется черным, даже если мы укажем для свойства **ForeColor** другой цвет.

Ниже экранных кнопок необходимо поместить еще одну надпись, которая по умолчанию получает имя **Label2**. Эта надпись будет содержать сведения о разработчике приложения. Затем для данной надписи производим настройку ее свойств.

Для свойства Caption устанавливаем содержание: «**Программу разработал …**» (вместо многоточия ставим фамилию и инициалы того, кто разработал данную программу). Для свойства **Font** указываем размер шрифта, равный 16 пунктам, и жирное начертание. Используя свойство **ForeColor**, задаем цвет шрифта надписи – красный.

Наконец, задаем еще одно свойство данной надписи. Эта надпись, согласно условиям задания, по умолчанию должна быть невидимой. Для этого в окне **Properties** находим свойство **Visible** и устанавливаем для него значение **False**.

На этом работа по созданию и настройке графического интерфейса пользователя в данном проекте завершена. Получившийся в результате проведенной работы пользовательский интерфейс можно видеть на рис.1.10.

Следующим этапом в работе над проектом является написание программного кода. В данном проекте активными объектами, которые реагируют на программные события, являются две экранные кнопки. Для них и нужно написать процедуры, описывающие их реакции на различные внешние воздействия.

Для того чтобы открыть окно кода для какого-либо объекта, достаточно дважды щелкнуть его мышью. Основным программным событием, на которое должен реагировать любой объект в системе VBA, считается одиночный щелчок левой кнопкой мыши.

Дважды щелкнем мышью на экранной кнопке **CommandButton1**, и в результате откроется окно кода с заготовкой для процедуры, соответствующей этой кнопке. Данная заготовка выглядит следующим образом:

```
Private Sub CommandButton1_Click()

End Sub
```

Первая строка этой заготовки является заголовком процедуры. В заголовке пишутся служебное слово **Sub** (сокращение от **Subroutine**, что по-английски означает подпрограмма) и собственное имя процедуры. Собственное имя процедуры состоит из имени объекта, для которого она пишется, и название события, на которое должен реагировать этот объект. Между этими двумя частями собственного имени процедуры ставится символ подчеркивания. В данном случае имя объекта **CommandButton1**, а название события – **Click** (по-английски так называется одиночный щелчок мышью). В нижней строке заготовки стоит команда **End Sub** – конец процедуры. Между ними пользователь может ввести текст самой процедуры.

В данном случае следует описать действие, которое превращает невидимый ранее объект (нижнюю надпись **Label2**) в видимый. Для этого нужно изменить значение свойства **Visible** с **False** (объект не виден) на **True** (объект виден), или, иначе говоря, присвоить свойству **Visible** новое значение. Оператор (команда языка Visual Basic), который выполняет это действие, называется оператором присваивания.

Синтаксис данного оператора следующий. В начале указывается полное имя изменяемого свойства, которое состоит из имени объекта и собственного имени свойства. Имя объекта и имя свойства отделяются друг от друга с помощью точки. Таким образом, полное имя свойства в данном случае будет: **Label2.Visible**. Затем указывается операция присваивания, которая обозначается в языке Visual Basic знаком равенства. Наконец, после знака равенства указывается новое значение, которое присваивается этому свойству. Таким образом, целиком оператор, делающий невидимую ранее надпись видимой, будет выглядеть следующим образом:

```
Label2.Visible = True
```

Этот оператор мы и записываем в пустой строке, находящейся между заголовком и концом процедуры. В целом же процедура, описывающая реакцию на щелчок левой кнопкой мыши, будет выглядеть таким образом:

```
Private Sub CommandButton1_Click()
Label2.Visible = True
End Sub
```

Но на этом работа с объектом **CommandButton1** еще не завершена. Для данной экранной кнопки необходимо написать еще одну процедуру. Вторая процедура должна при двойном щелчке на кнопке

CommandButton1 делать надпись **Label2** со сведениями об авторе снова невидимой.

Для того чтобы создать вторую процедуру, поступаем следующим образом. В окне кода в правой верхней его части находится вспомогательное текстовое поле (см. рис.1.9). В нем содержится список программных событий, на которые может реагировать данный объект. По умолчанию в списке указано событие **Click** (одиночный щелчок левой кнопкой мыши). В текстовом поле справа от названия события имеется направленная вниз треугольная стрелка. Щелкнув по этой стрелке, мы увидим весь список программных событий, на которые может реагировать данный объект. В этом списке щелчком мыши выбираем событие **DblClick** (двойной щелчок левой кнопкой мыши), а окне кода появляется шаблон для второй процедуры. Этот шаблон имеет следующий вид:

```
Private Sub CommandButton1_DblClick (ByVal Cancel
As MSForms.ReturnBoolean)

End Sub
```

В пустую строку, находящуюся между заголовком и концом процедуры, записываем программный код, который выполняет действие, обратное действию, описанному в предыдущей процедуре, – скрывает надпись со сведениями об авторе. Этот код выглядит так:

```
Label2.Visible=False
```

В итоге для описания реакции экранной кнопки **CommandButton1** на двойной щелчок левой кнопкой мыши получаем следующую процедуру:

```
Private Sub CommandButton1_DblClick (ByVal Cancel
As MSForms.ReturnBoolean)
Label2.Visible=False
End Sub
```

На этом программирование кнопки **CommandButton1** завершено. Возвращаемся к основной экранной форме, для чего можно использовать команду меню **View → Object**.

Следующим объектом, для которого нужно написать программный код, является экранная кнопка **CommandButton2**. Дважды щелкаем мышью по этой кнопке, и снова на экране появляется окно кода с шаблоном уже для третьей процедуры, имеющим следующий вид:

```
Private Sub CommandButton2_Click()

End Sub
```

Этот шаблон аналогичен шаблону первой процедуры, за исключением названия объекта. В пустую строку между заголовком и концом процедуры нужно вставить команду, которая будет закрывать приложение при

одиночном щелчке мышью по экранной кнопке **CommandButton2.** Эта команда состоит из одного служебного слова и выглядит так:

```
End
```

А вся процедура для описания реакции второй экранной кнопки на щелчок левой кнопки мыши будет выглядеть так:

```
Private Sub CommandButton2_Click()
End
End Sub
```

Таким образом, мы завершили написание программного кода для нашего первого приложения. Ниже приводим весь листинг целиком:

```
Private Sub CommandButton1_Click()
Label2.Visible = True
End Sub

Private Sub CommandButton1_DblClick(ByVal Cancel As
MSForms.ReturnBoolean)
Label2.Visible = False
End Sub

Private Sub CommandButton2_Click()
End
End Sub
```

Теперь, когда работа над программой завершена, можно запустить ее на выполнение. Для этого нужно использовать команду меню **Run → Run** или просто щелкнуть треугольную стрелку в панели инструментов. На экране появится рабочее окно приложения, внешний вид которого приведен на рис. 1.11.

Обратите внимание на то, что при переходе программы в режим выполнения исчезла покрывавшая основное окно координатная сетка, а также стала невидимой надпись со сведениями об авторе. Теперь осталось протестировать созданное приложение. Для этого один раз щелкаем мышью на экранной кнопке «**Сведения об авторе**», – надпись со сведениями об авторе должна появиться ниже этой кнопки. Затем дважды щелкаем ту же экранную кнопку, – надпись должна исчезнуть. Наконец, завершаем работу приложения одиночным щелчком на экранной кнопке «**Выход**».

Конечно, не исключено, что где-то в программном коде разработчик допустил ошибку. В этом случае при запуске программы на выполнение появится сообщение об ошибке. Такое сообщение выводится в виде диалогового окна, содержащего текст: «**Compile error**» (Ошибка компиляции). Ниже, как правило, выводится дополнительная информация о конкретном типе ошибки. Кроме того, как правило, та строка, в которой содержится ошибка, выделяется в окне кода синим цветом, что существенно

облегчает поиск этой ошибки. После нахождения и исправления ошибки можно повторно запустить эту программу на выполнение.

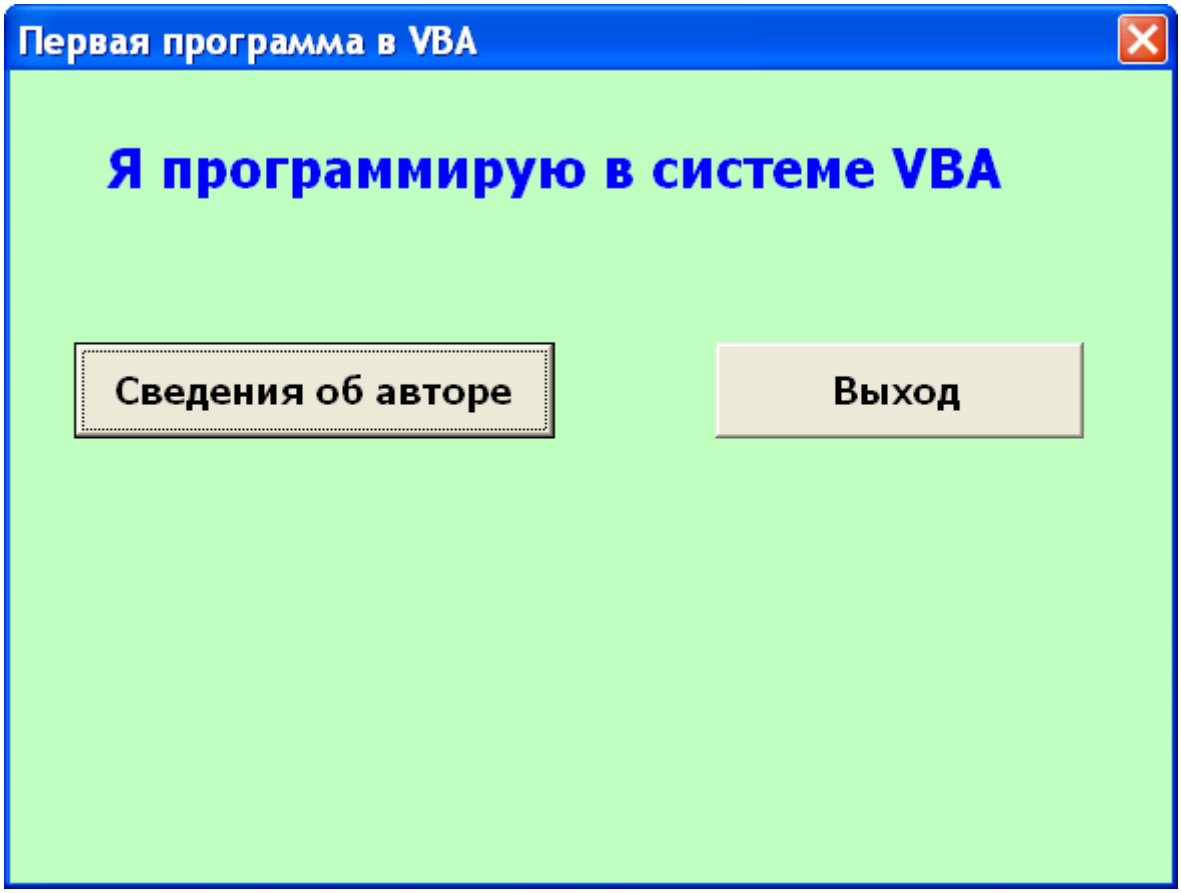


Рис. 1.11. Первая программа в рабочем режиме

Когда пользователь устранил все ошибки, и программа при тестировании показала свою работоспособность, эта программа должна быть сохранена. Поскольку программа на Visual Basic сохраняется вместе с исходным файлом, то нужно закрыть систему программирования одним из указанных выше способов (через меню **File** или через закрывающую кнопку в строке заголовка), а затем закрыть файл **Первая программа.doc** с сохранением сделанных в нем изменений.

1.5. Изменение цвета экранной формы. Работа с переключателями

Как уже было отмечено, к числу объектов системы VBA относятся не только элементы графического интерфейса пользователя, создаваемые на экранной форме, но и сама эта форма. Возможности изменения свойств основной экранной формы (**UserForm**) продемонстрирует проект, разработка которого описывается в данном разделе пособия.

Новый проект будет изменять цвет экранной формы с помощью группы из 8 переключателей. Как известно, в Windows переключатели (другое название этих элементов – радиокнопки) – это такая группа элементов управления, в которой активным (включенным) всегда может быть один и только один элемент, а остальные выключены. В данном случае каждый из 8 переключателей отвечает за один из цветов, в который будет окрашена экранная форма. Форма в данном проекте должна окрашиваться в один из следующих цветов: белый, черный, красный, желтый, синий, зеленый, фиолетовый, бирюзовый.

Как и в предыдущем проекте, вначале нужно создать новый файл Microsoft Word (например, **Цвета.doc**). Затем в Word с помощью команды **Сервис → Макрос → Редактор Visual Basic** открываем экран системы VBA. На этом экране с помощью команды **Insert → UserForm** создаем основную пользовательскую форму. Далее на форме необходимо создать пользовательский интерфейс проекта. Пользовательский интерфейс должен включать следующие элементы: рамка, содержащая группу переключателей, сами переключатели, экранная кнопка «**Выход**» и экранная кнопка «**Сведения об авторе**».

В начале работы над проектом задаем текст в строке заголовка пользовательской формы. Для этого нужно щелкнуть мышью на любом свободном месте внутри формы и затем в окне **Properties** (Свойства) найти свойство **Caption**. В поле, расположенном справа от названия этого свойства, вводим следующий текст: «**Изменение цвета формы**». Этот текст будет отображаться в строке заголовка основной формы проекта.

Для создания рамки, внутри которой будут размещены переключатели, необходимо в панели элементов **Toolbox** найти элемент **Frame** (в русифицированных версиях VBA он называется – «**Рамка**»). Этот элемент выглядит следующим образом:  . Щелкнув мышью по данному элементу, а затем по экранной форме, помещаем этот элемент на форму, создавая новый объект **Frame1**.

Объект Frame1 имеет на границах маркеры (маленькие белые квадраты), с помощью которых можно увеличивать или уменьшать размеры

этого объекта. Размеры рамки следует изменить таким образом, чтобы в ней разместились 8 переключателей с надписями.

Подобно другим элементам пользовательского интерфейса в VBA, объект **Frame1** имеет свойство **Caption**, отвечающее за содержание текста, который виден в верхней части рамки. В окне **Properties** для рамки находим свойство **Caption** и в поле справа вводим его значение: «**Выберите цвет формы**».

Затем внутрь рамки помещаем переключатели. Элемент **OptionButton** (переключатель) также находится на панели элементов **Toolbox** и выглядит так:  . Последовательно размещаем переключатели один под другим. Эти элементы управления получат в нашем проекте такие имена: **OptionButton1, OptionButton2, OptionButton3, OptionButton4, OptionButton5, OptionButton6, OptionButton7, OptionButton8**. Каждый из этих переключателей также имеет свойство **Caption**. В данном случае свойство **Caption** отвечает за надпись, которая расположена справа от самого переключателя. Для переключателя **OptionButton1** в свойстве **Caption** пишем слово «**Белый**», для **OptionButton2** в этом свойстве пишем «**Черный**», для **OptionButton3 – «Красный», OptionButton4 – «Желтый», OptionButton5 – «Синий», OptionButton6 – «Зеленый», OptionButton7 – «Фиолетовый», OptionButton8 – «Бирюзовый».**

Далее на экранной форме необходимо создать экранные кнопки **CommandButton1** и **CommandButton2**. Для первой кнопки в **Caption** пишем фразу «**Сведения об авторе**», для второй — «**Выход**». Для всех надписей на рамке, переключателях и экранных кнопках настраиваем необходимым образом шрифт с помощь свойства **Font**. На этом создание графического интерфейса пользователя для данной программы завершено. Внешний вид интерфейса показан на рис. 1.12.

Затем следует написать программный код для переключателей и кнопок. Процедура, описывающая работу каждого переключателя, должна окрашивать основную экранную форму **UserForm1** в один из перечисленных цветов. Экранная форма, как мы знаем из разд. 1.4, имеет свойство **BackColor** (цвет фона), которому и присваивается цветовое значение.

Цвет в Visual Basic может задаваться двумя способами: в виде шестнадцатеричного числа или в виде словесной константы этого языка. В нашей программе мы будем использовать второй способ обозначения цвета, как более наглядный. Константа, соответствующая определенному цвету, состоит из приставки **vb** (сокращение от Visual Basic) и названия конкретного цвета. Белому цвету соответствует константа **vbWhite**, черному – **vbBlack**, красному – **vbRed**, желтому – **vbYellow**, синему – **vbBlue**, зеленому – **vbGreen**, фиолетовому – **vbMagenta**, бирюзовому –

vbCyan. Тогда окрашивание основной формы в белый цвет будет производиться оператором:

```
UserForm1.BackColor = vbWhite
```

Аналогичные операторы используются и для окрашивания формы в другие цвета. Следует обратить внимание на то, что пространство, находящееся внутри рамки, данными командами не закрашивается.

Для того чтобы ввести необходимый программный код, на форме дважды щелкаем мышью переключатель **OptionButton1**, появляется окно кода с шаблоном следующего вида:

```
Private Sub OptionButton1_Click()

End Sub
```

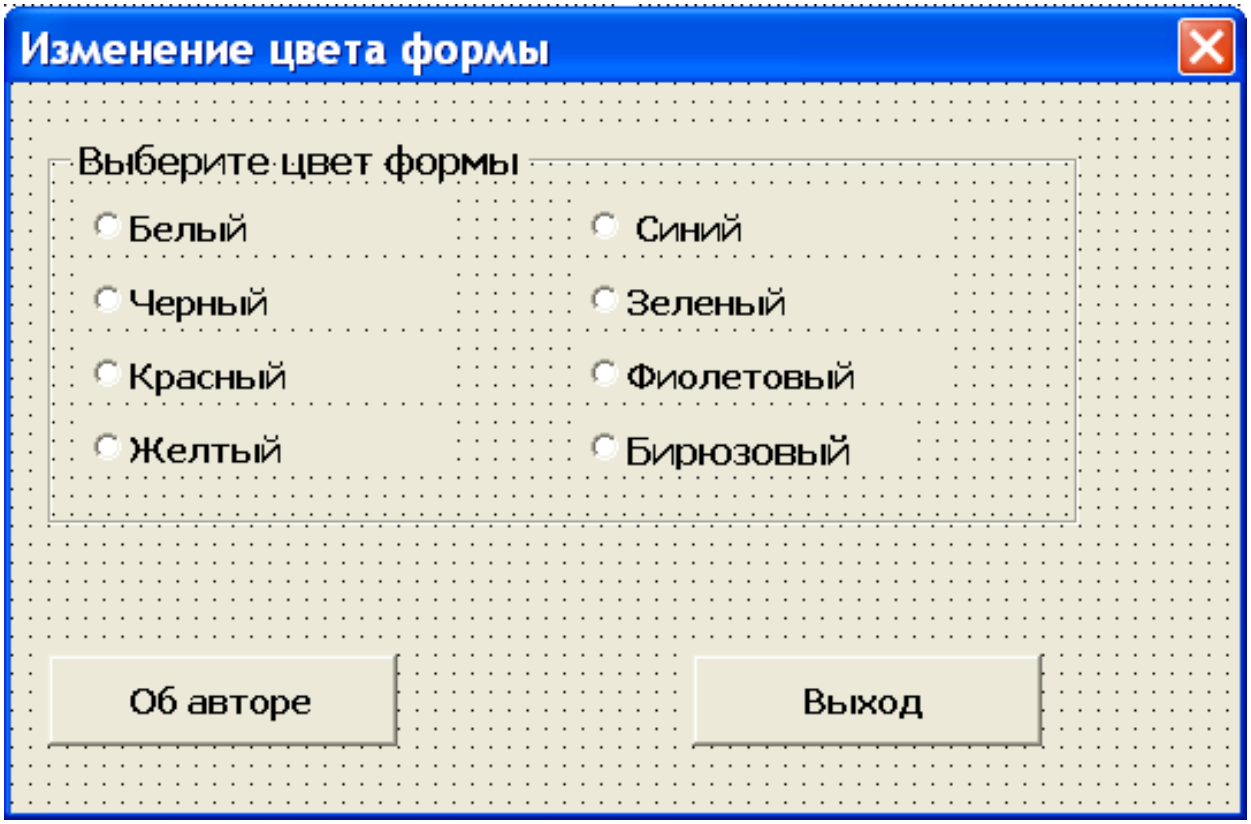


Рис. 1.12. Пользовательский интерфейс программы «**Изменение цвета формы**»

В пустую строку, находящуюся между заголовком и концом процедуры, записываем указанный выше оператор. Тогда в целом процедура для первой радиокнопки будет выглядеть так:

```
Private Sub OptionButton1_Click()
UserForm1.BackColor = vbWhite
End Sub
```

Затем возвращаемся к основной форме, дважды щелкаем мышью переключатель **OptionButton2,** появляется окно кода с шаблоном для второй процедуры. В пустой строке этой процедуры записываем оператор:

```
UserForm1.BackColor = vbBlack
```

Процедура для второго переключателя **OptionButton2** в целом будет выглядеть так:

```
Private Sub OptionButton2_Click()
UserForm1.BackColor = vbBlack
End Sub
```

Выполнение этой процедуры при щелчке по второму переключателю будет окрашивать экранную форму в черный цвет. Аналогичным образом программируем все остальные 6 переключателей, находящихся внутри рамки.

Затем необходимо написать программный код для экранной кнопки **CommandButton1** (кнопка «**Об авторе**»). Эта кнопка будет работать иначе, чем в предыдущем проекте. Пользовательский интерфейс данной программы и так содержит достаточно большое количество элементов, поэтому не будет перегружать его, создавая на основной форме еще одну надпись. В данном проекте щелчок мышью по кнопке «**Об авторе**» должен выводить на экран дополнительное диалоговое окно, содержащее информацию о разработчике программы, и экранную кнопку «**Ok**», закрывающую диалоговое окно.

Для создания такого окна используется команда **Msgbox**. В данной команде после служебного слова **Msgbox** в круглых скобках и в кавычках указывается текст, выводимый в появляющемся диалоговом окне. Пример реализации такой команды приведен ниже:

```
Msgbox("Программу разработал А.Н. Маслобоев")
```

Записываем указанную команду в соответствующую строку процедуры **CommandButton1_Click,** расположенную между заголовком процедуры и концом процедуры**.** Процедура в целом будет выглядеть так:

```
Private Sub CommandButton1_Click()
Msgbox("Программу разработал А.Н. Маслобоев")
End Sub
```

Выполнение этой команды при щелчке на экранной кнопке и выведет на экран диалоговое окно, которое появится поверх основной экранной формы. Внешний вид данного окна показан на рис.1.13. После ознакомления с данным окном пользователь может закрыть его щелчком на экранной кнопке «**OK**» и вернуться к работе с основным окном приложения.

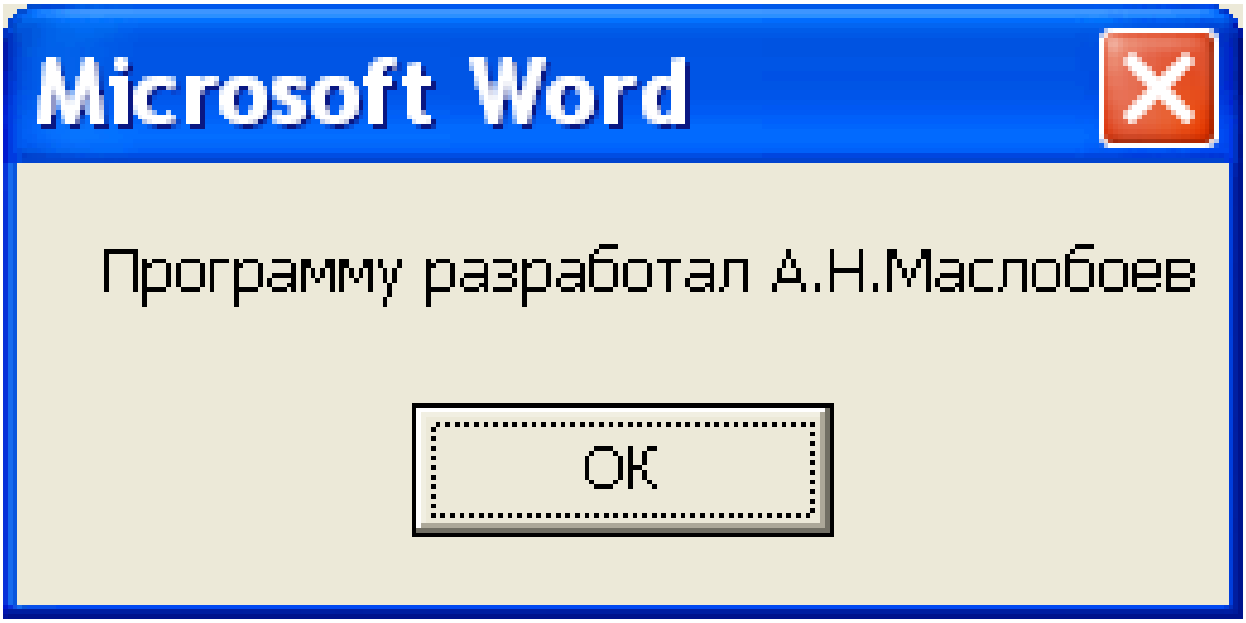


Рис. 1.13. Диалоговое окно со сведениями об авторе

Экранная кнопка **CommandButton2** («**Выход**»), закрывающая данное приложение по окончании его работы, в этом проекте программируется так же, как и в предыдущем.

В итоге после написания программного кода для всех объектов приложения мы получаем листинг, текст которого приводится ниже:

```
Private Sub CommandButton1_Click()
MsgBox ("Программу разработал А.Н.Маслобоев")
End Sub

Private Sub CommandButton2_Click()
End
End Sub

Private Sub OptionButton1_Click()
UserForm1.BackColor = vbWhite
End Sub

Private Sub OptionButton2_Click()
UserForm1.BackColor = vbBlack
End Sub

Private Sub OptionButton3_Click()
UserForm1.BackColor = vbRed
End Sub

Private Sub OptionButton4_Click()
UserForm1.BackColor = vbYellow
End Sub

Private Sub OptionButton5_Click()
```

```
UserForm1.BackColor = vbBlue
End Sub

Private Sub OptionButton6_Click()
UserForm1.BackColor = vbGreen
End Sub

Private Sub OptionButton7_Click()
UserForm1.BackColor = vbMagenta
End Sub

Private Sub OptionButton8_Click()
UserForm1.BackColor = vbCyan
End Sub
```

После завершения работы над программным кодом запускаем программу на выполнение. На рис. 1.14 показана программа, находящаяся в рабочем режиме. Активным является переключатель, окрашивающий экранную форму в бирюзовый цвет.

После тестирования всех 8 переключателей, имеющихся на форме, и кнопки, открывающей окно со сведениями об авторе, можно завершить работу приложения щелчком на экранной кнопке «Выход».

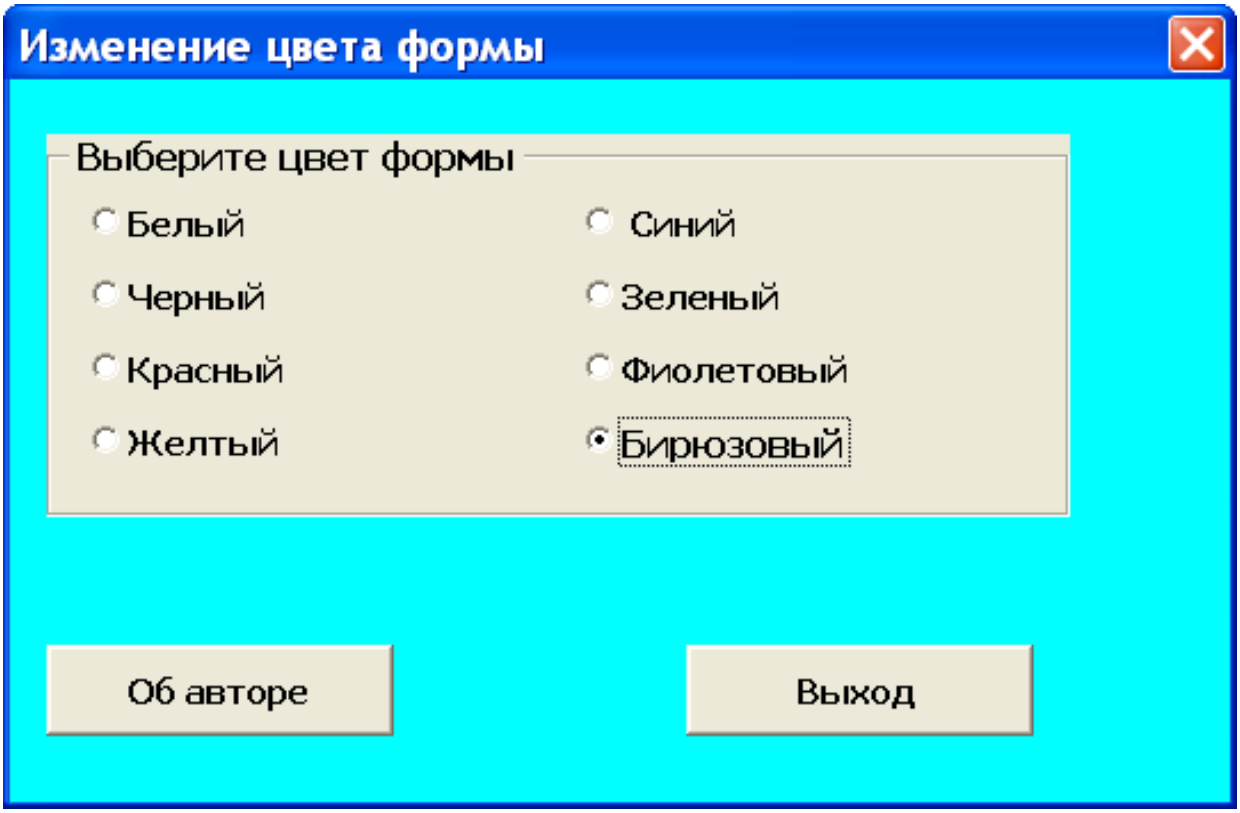


Рис. 1.14. Программа «Изменение цвета формы» в процессе работы

Вопросы для самоконтроля к главе 1

1. Что такое макрокоманда (макрос)?
2. Какая последовательность команд в программе Word позволяет начать запись макроса?
3. По каким признакам можно определить, что в программе Word включен режим записи макроса?
4. Какую экранную кнопку следует щелкнуть мышью для того, чтобы остановить запись макроса?
5. Какую последовательность команд нужно использовать в программе Word для того, чтобы войти в среду программирования VBA?
6. Какую последовательность команд нужно использовать для того, чтобы в среде VBA создать пользовательскую форму?
7. Укажите основные объекты, которые можно создавать в VBA с помощью панели элементов.
8. Какие вкладки содержит окно свойств объекта?
9. Как вывести в среде VBA на экран окно программного кода?
10. Какое свойство определяет фоновый цвет объекта?
11. Какие свойства определяют основные характеристики шрифта, которым сделана надпись на объекте?
12. Какое свойство определяет видимость или невидимость объекта?
13. Какие цвета входят в стандартную цветовую палитру Visual Basic?
14. Какая команда создает в VBA дополнительное диалоговое окно?
15. Какую последовательность действий нужно выполнить в VBA для создания группы переключателей?

Глава 2. Использование переменных в VBA

2.1. Вычисление площади прямоугольника

Одной из основных задач, для которых используется любая система программирования, является выполнение вычислений на основе имеющихся исходных данных и получение некоторого конечного результата, который должен быть представлен в понятной и удобной для пользователя форме.

Для того чтобы можно было производить необходимые вычисления, в программировании используются как постоянные числовые величины (константы), так и переменные. Переменной называется величина, значение которой может изменяться в процессе работы программы.

Каждая переменная, используемая в программе, обладает определенным именем (идентификатором). Имя переменной может включать латинские буквы и цифры, причем первым символом в имени должна быть буква. Использование букв кириллицы в именах переменных не допускается. Имена переменных не должны совпадать по написанию со служебными словами языка Visual Basic.

Также в имени переменной не должно быть пробелов. В принципе возможно, чтобы имя переменной состояло из двух и более слов, но в этом случае пробелы должны заменяться символом подчеркивания. Кроме того, не допускается, чтобы в имени переменной присутствовали символы, которые обозначают математические операции и операции сравнения **(+ - * / < > = <= >= <>**).

Примеры правильных имен переменных:

x

Summa

faktorial

y1

z234

Summa_vklada

Неправильные имена переменных:

1y (первым символом в имени не должна быть цифра)

площадь (в имени присутствуют символы кириллицы)

Summa vklada (в имени переменной присутствует пробел)

x/y (в имени переменной использован символ «слэш», который применяется в Visual Basic для обозначения операции деления).

При выборе имен переменных желательно, чтобы имя по возможности отображало характер той величины, которая в этой переменной хранится. Например, переменную, содержащую сумму некоторых чисел, логично назвать **Sum**. Переменной, содержащей длину окружности, подойдет имя **l** (традиционное обозначение длины в математике), для переменной, хранящей значение площади геометрической фигуры, применим обозначение – **s**, для объема фигуры – **v** и т. д.

В отличие от других языков программирования (например, языков Turbo Pascal и Object Pascal), в языке Visual Basic не требуется обязательного описания переменных с указанием их имен и типов, к которым принадлежат переменные, перед их использованием в программе. В Visual Basic считается, что все переменные, для которых это специально не оговорено, относятся к универсальному типу **Variant**. Можно сказать, что объявление переменной производится при ее использовании в программе. Это называется неявное объявление переменной.

Однако в некоторых случаях и в Visual Basic желательно выполнять явное объявление переменных, т.е. предварительно их описывать. В этом случае для явного объявления переменных используется конструкция следующего вида:

```
Dim Name
```

где **Name** – это имя переменной.

Часто встречается ситуация, когда при объявлении переменной желательно указать ее тип (такая переменная называется в Visual Basic типизированной). Тип переменной указывается с помощью команды:

```
Dim Name as Type
```

где **Type** – тип описываемой переменной.

В языке Visual Basic используются следующие основные типы переменных:

Byte – используется для хранения целого положительного числа в диапазоне от 0 до 255;

Boolean – применяется для логических переменных, т.е. таких, которые могут принимать только два значения: **True** (истина) и **False** (ложь);

Integer – для представления целых чисел в диапазоне от –32 768 до 32 767;

Long – для представления больших целых чисел в диапазоне от –2 147 483 648 до 2 147 483 647;

Single – основной тип для представления вещественных чисел. Охватывает отрицательные числа от $-3.\,402\,823 * 10^{38}$ до $-1.\,401\,298 * 10^{-45}$ и положительные числа от $1.\,401\,298 * 10^{-45}$ до $3.\,402\,823 * 10^{38}$ (в данной записи для отделения целой части от дробной используется точка, как это принято в информационных технологиях);

Double – используется для представления вещественных чисел с двойной точностью. Охватывает диапазон отрицательных чисел от -1. 797 693 134 862 32 * 10^{308} до -4. 940 656 458 412 47 * 10^{-324} для отрицательных значений и от 4. 940 656 458 412 47 * 10^{-324} до 1. 797 693 134 862 32 * 10^{308} для положительных значений;

Currency – используется для записи чисел с фиксированной точкой. В этом случае справа от десятичной точки, отделяющей целую часть от дробной, всегда находится 4 цифры. Этот тип охватывает числа в диапазоне от -922337203685477.5808 до 922337203685477.5807. Этот тип применяется в том случае, когда важна большая точность вычислений, например, при расчетах финансового характера;

String – используется для представления строковых переменных. Строковыми величинами в VBA являются любые данные текстового типа, в частности, информация, вводимая в текстовые окна.

Изменение значения переменной в программе, как правило, осуществляется с помощью оператора присваивания. Общий вид данного оператора следующий:

```
Name = Value
```

Здесь **Name** – имя переменной, значение которой изменяет данный оператор; **Value** – присваиваемое значение; = – обозначение операции присваивания. Присваиваемое значение в простейшем случае может быть одной константой (числовой или текстовой). Если присваиваемая константа является текстовой, то она должна быть заключена в двойные кавычки. Примеры операций присваивания:

```
x = 10
```

```
s = «Персональный компьютер»
```

В первом случае численной переменной **x** присваивается значение, равное 10, а во втором случае строковой переменной **s** присваивается текстовое значение «Персональный компьютер».

В общем случае в операторе присваивания справа от знака равенства может находиться выражение, включающее числа, имена переменных и знаки математических операций. При выполнении программы значение такого выражения вначале вычисляется, и уже затем вычисленное значение присваивается переменной, находящейся слева от знака равенства. Например, если мы видим в программе такой оператор присваивания:

```
y = 5*8+30/6
```

то вначале вычисляется значение расположенного справа выражения, а затем вычисленное значение (число 45) присваивается переменной **y**. Обратите внимание на то, что в данном выражении операция умножения обозначается звездочкой, а деление – косой чертой (знаком «слэш»). Никакие

другие символы для обозначения операций умножения и деления в языке Visual Basic использовать нельзя. Сложение и вычитание в Visual Basic обозначаются плюсом и минусом, как и в математике.

Говоря о работе с переменными в VBA, необходимо остановиться еще на одном обстоятельстве. Дело в том, что для ввода исходных данных в приложениях обычно используется такой элемент пользовательского интерфейса, как текстовые окна. Но надо иметь в виду, что информация, вводимая в текстовое окно, по умолчанию воспринимается компьютером как текстовая, а над текстовыми величинами нельзя выполнять математические вычисления.

Поэтому если с данными, введенными в текстовое окно, в дальнейшем нужно произвести вычисления, то поступают следующим образом. Вначале текстовые данные следует преобразовать в числовую форму, а затем уже над числовыми величинами можно производить различные математические операции. Это преобразование можно осуществить с помощью функции **Val**. Аргументом данной функции является строка, содержащая числовую величину, а значением – число соответствующего типа.

Когда же все необходимые для решения поставленной задачи вычисления выполнены, и получены конечные результаты, то нужно их вывести на экранную форму. Вывод результатов также обычно производится в текстовые окна. Следовательно, возникает необходимость выполнить обратное преобразование: численные результаты должны быть преобразованы в строковую форму. Для такого преобразования применяется функция **Str**, аргументом которой является число, а значением – соответствующая ему строковая величина.

Рассмотрим основные приемы работы с переменными в системе программирования VBA на следующем примере. Требуется создать приложение, которое по заданной длине и ширине вычисляет площадь прямоугольника. Для решения поставленной задачи создадим новый документ **Прямоугольник.doc,** войдем в систему программирования VBA и создадим экранную форму. На экранной форме должны быть представлены следующие элементы интерфейса:

1. Три текстовых окна. Два окна предназначены для ввода исходных данных – длины и ширины прямоугольника, а в третьем окне должен быть выведен результат – площадь этого прямоугольника.

2. Три надписи. Каждая из надписей должна находиться слева от одного из тестовых окон и пояснять пользователю его назначение.

3. Четыре экранных кнопки. Первая кнопка «**Подсчитать**» нужна для выполнения вычислений. Щелчок мышью на этой кнопке после ввода исходных данных в двух верхних окнах должен производить необходимые вычисления и обеспечивать вывод результата в третьем текстовом окне. Вторая кнопка «**Сброс**» должна обеспечивать очистку всех трех текстовых окон. Она нужна для того, чтобы при необходимости выполнить повторные вычисления, пользователю не надо было очищать текстовые окна вручную.

Третья кнопка «**Об авторе**» выводит сведения об авторе в дополнительном диалоговом окне, как в предыдущем проекте. Четвертая кнопка «**Выход**» закрывает приложение.

В строке заголовка основной экранной формы вводим текст: «**Вычисление площади прямоугольника**». Затем приступаем к созданию и настройке текстовых окон. Для того, чтобы создать на форме новое текстовое окно, нужно на **Панели элементов** найти необходимую заготовку. Соответствующая кнопка имеет пиктограмму с изображением букв **ab** и вертикального текстового курсора. Щелкнув по кнопке и затем по экранной форме, создаем первое текстовое окно – объект **Textbox1**. Настраиваем свойство **Font** данного объекта таким образом, чтобы цифры, отображаемые в окне, были хорошо читаемы. В это окно будет вводиться длина прямоугольника. Далее создаем и настраиваем текстовые окна **Textbox2** для ввода ширины прямоугольника. Ниже располагаем текстовое окно **Textbox3** для вывода искомого результата – площади прямоугольника.

Затем создаем три надписи – **Label1, Label2 и Label3**. Надпись **Label1** должна содержать текст: «**Введите длину**», а надпись **Label2**: «**Введите ширину**». Для надписи **Label3** вводим следующий текст: «**Площадь прямоугольника равна**».

Далее создаем на форме четыре экранные кнопки: **CommandButton1** с текстом «**Подсчитать**», **CommandButton2** – «**Сброс**», **CommandButton3** – «**Об авторе**» и **CommandButton4** – «**Выход**». Создание и начальная настройка свойств надписей и экранных кнопок производится аналогично тому, как это делалось в предыдущих проектах. Созданный для данного проекта интерфейс приведен на рис. 2.1.

Написание программного кода начинаем с экранной кнопки «**Подсчитать**». Вначале описываем используемые в программе переменные **a,b** и **s** как переменные целого типа (**a** – длина прямоугольника, **b** – ширина прямоугольника, **s** – площадь прямоугольника) с помощью команды:

```
Dim a,b,s As Integer
```

Затем нужно получить численные значения исходных величин **a** и **b**. Эти значения можно определить следующим образом. Длину прямоугольника пользователь должен ввести в верхнее текстовое окно **Textbox1**. Введенное значение представляет собой строковую величину и присваивается свойству **Text** объекта **Textbox1**. Это значение можно преобразовать в числовую форму с помощью функции **Val** и присвоить его переменной **a**. Эти действия мы запишем с помощью следующего оператора присваивания:

```
a = Val(Textbox1.Text)
```

Значение ширины прямоугольника можно взять из текстового окна **Textbox2** и присвоить его переменной **b**, используя следующий оператор:

```
b = Val(Textbox2.Text)
```

С помощью следующего оператора вычисляем площадь прямоугольника, перемножив его длину и ширину, и присваиваем полученное значение переменной **s**:

```
s = a*b
```

Теперь осталось преобразовать полученный результат из числовой формы в строковую с помощью функции **Str** и вывести его в третьем текстовом окне с помощью следующего оператора:

```
TextBox3.Text = Str(s)
```

Весь же текст процедуры для кнопки **CommandButton1** будет выглядеть таким образом:

```
Private Sub CommandButton1_Click()
Dim a, b, s As Integer
a = Val(TextBox1.Text)
b = Val(TextBox2.Text)
s = a * b
TextBox3.Text = Str(s)
End Sub
```

Далее программируем кнопку **CommandButton2** (кнопка «**Сброс**»), которая должна очищать все текстовые окна перед новым вводом исходных данных. Для очистки текстового окна **Textbox1** нужно присвоить свойству **Text** этого окна «нулевое» значение, состоящее из двух двойных кавычек и пробела между ними, посредством следующего оператора:

```
Textbox1.Text = " "
```

Такими же операторами очищаются и два других текстовых окна:

```
TextBox2.Text = " "
TextBox3.Text = " "
```

Полный текст процедуры для кнопки «**Сброс**» приведен ниже:

```
Private Sub CommandButton2_Click()
TextBox1.Text = " "
TextBox2.Text = " "
TextBox3.Text = " "
End Sub
```

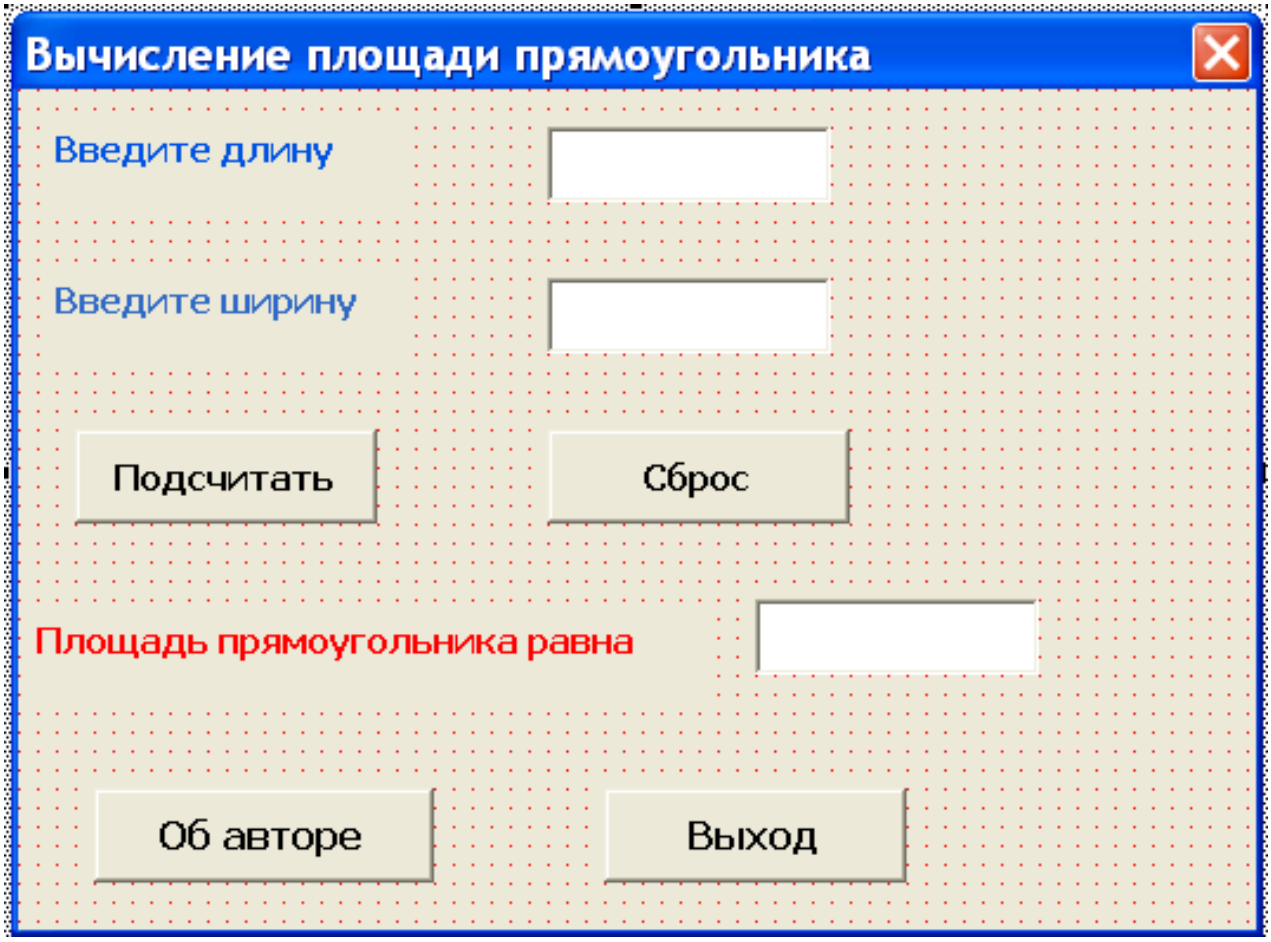


Рис. 2.1. Пользовательский интерфейс программы «Вычисление площади прямоугольника»

Программирование кнопок «**Об авторе**» и «**Выход**» не должно вызвать у пользователя никаких затруднений, так как производится аналогично тому, как это было сделано в предыдущем проекте. Полный текст программного модуля для данного проекта выглядит таким образом:

```
Private Sub CommandButton1_Click()
Dim a, b, s As Integer
a = Val(TextBox1.Text)
b = Val(TextBox2.Text)
s = a * b
TextBox3.Text = Str(s)
End Sub

Private Sub CommandButton2_Click()
TextBox1.Text = " "
TextBox2.Text = " "
TextBox3.Text = " "
End Sub

Private Sub CommandButton3_Click()
```

```
MsgBox ("Программу разработал А.Н. Маслобоев")
End Sub

Private Sub CommandButton4_Click()
End
End Sub
```

После окончания записи программного кода приложение можно запускать на выполнение. Внешний вид приложения в рабочем режиме с исходными данными и результатом вычисления приведен на рис. 2.2. После ввода исходных данных и получения искомой величины площади прямоугольника щелчком на кнопке «**Подсчитать**» можно очистить все текстовые окна кнопкой «**Сброс**» и ввести новые исходные данные, а затем вновь очистить окна. Далее можно проверить работу кнопки «**Об авторе**», выводящей дополнительное диалоговое окно, закрыть это окно щелчком на кнопке «**Ok**» и закрыть приложение щелчком на кнопке «**Выход**».

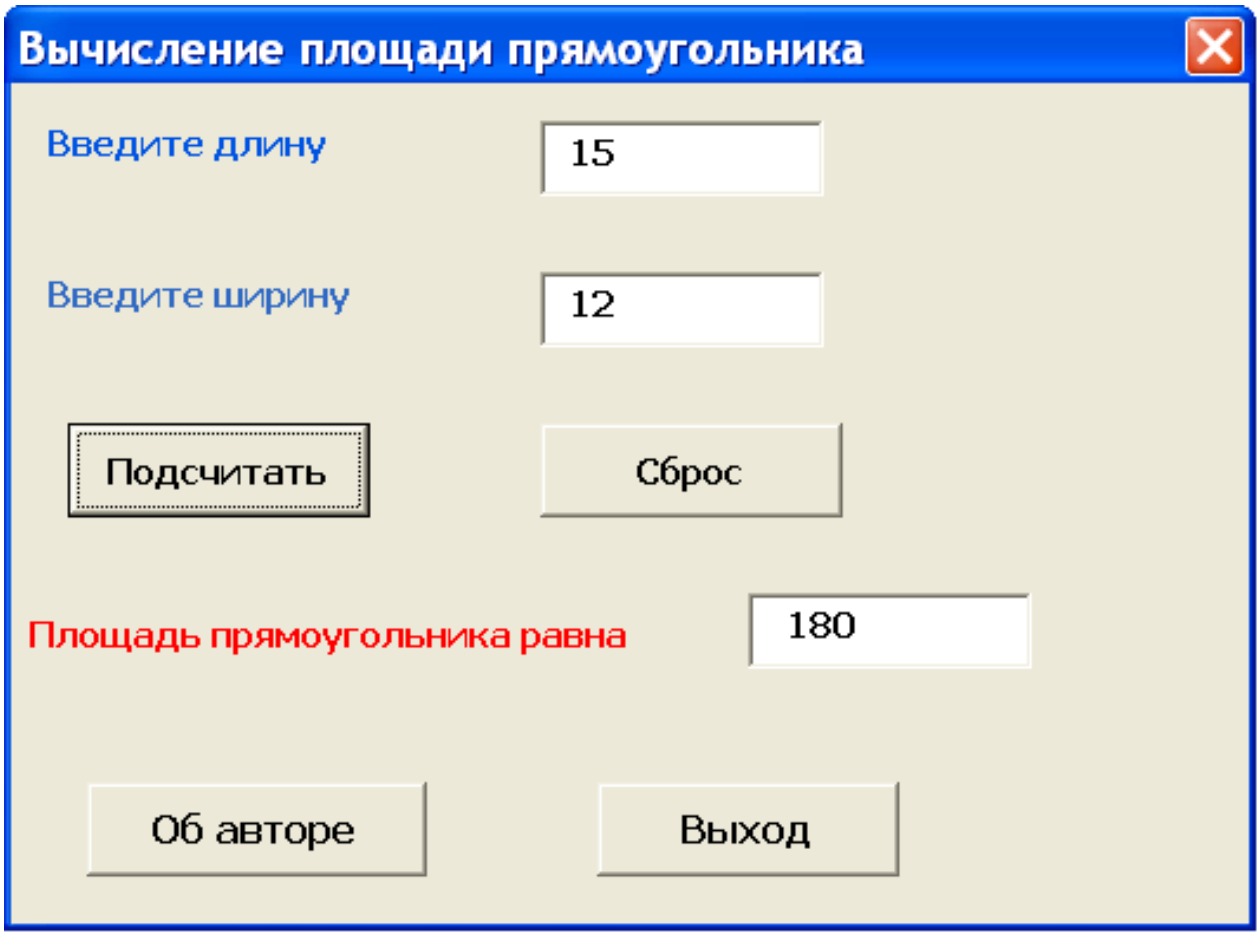


Рис. 2.2. Программа «Вычисление площади прямоугольника» в рабочем режиме

2.2. Вычисление суммы цифр числа

В предыдущем разделе пособия мы познакомились с четырьмя основными математическими операциями, которые используются в языке Visual Basic. Но наряду с ними, в Visual Basic существуют еще две специальные операции, которые применяются только по отношению к целым величинам (переменным и константам).

Первая из этих операций называется операцией целочисленного деления и обозначается обратной косой чертой \ (для этого символа также нередко употребляется заимствованное из английского языка наименование «бэкслэш»). Выполнение этой операции производится следующим образом: вначале выполняется обычное деление, затем берется получившееся при делении частное, от него отбрасывается дробная часть и сохраняется только целая. Эта целая часть частного и будет результатом целочисленного деления.

Таким образом, при выполнении операции целочисленного деления 17 \ 5 результат будет равен 3 (частное от обычного деления равно 3,4, а дробную часть 0,4 мы отбрасываем), а при выполнении операции 42 \ 8 получим число 5 (частное при обычном делении равно 5,25, и дробную часть этого числа также отбрасываем).

Вторая операция – это нахождение остатка при целочисленном делении. Эта операция в языке Visual Basic обозначается служебным словом **Mod**, которое должно быть отделено пробелами от величин, над которыми выполняется операция. Результатом выполнения операции 17 **Mod** 5 будет число 2.

Рассмотрим использование этих операций на следующем примере. Требуется найти сумму цифр трехзначного числа, введенного пользователем. В общем виде алгоритм решения этой задачи будет выглядеть таким образом.

Вначале нужно найти старший разряд числа, для чего необходимо разделить исходное число на 100 посредством операции целочисленного деления. Если, например, в качестве исходного числа взять число 365 то результат операции 365 **** 100 будет равен 3. Следовательно, старшей цифрой числа будет цифра 3.

Затем следует найти остаток от деления исходного числа на 100. Этот остаток содержит средний и младший разряды числа (десятки и единицы). В нашем примере при выполнении операции 365 **Mod** 100 результат будет равен числу 65.

Для нахождения среднего разряда нужно выполнить операцию целочисленного деления найденного остатка на 10. В рассматриваемом примере при выполнении операции 65 \ 10 получим результат 6, который и будет средней цифрой числа.

Затем, выполнив операцию нахождения остатка от деления этой же величины на 10, найдем младший разряд трехзначного числа. Для нашего примера результат операции 65 **Mod** 10 будет равен 5, т.е. младшей цифре

числа. Сложив все найденные величины, получим искомый результат. Для данного примера результат сложения чисел 3, 6 и 5 будет равным числу 14.

Данный алгоритм реализован в приведенном ниже проекте «Вычисление суммы цифр трехзначного числа». Работу над проектом начинаем с создания нового файла **Трехзначное число.doc**, затем запускаем систему программирования VBA и создаем для проекта новую экранную форму.

Вначале на экранной форме создаем пользовательский интерфейс приложения. Данный интерфейс включает следующие элементы: текстовое окно для ввода исходных данных **TextBox1** с поясняющей надписью **Label1**, экранная кнопка **CommandButton1** (кнопка «**Ввод**»), текстовое окно **TextBox2** для вывода результата с поясняющей надписью **Label2**, кнопка **CommandButton2** (кнопка «**Сброс**»), очищающая оба текстовых окна, и кнопка **CommandButton3** (кнопка «**Выход**»), закрывающая экранную форму.

Все эти элементы интерфейса нам уже знакомы по предыдущим проектам, поэтому нет необходимости в подробном описании процесса создания пользовательского интерфейса и настройки его элементов. Все эти действия выполняются по аналогии с предыдущими проектами. Внешний вид пользовательской формы показан на рис. 2.3.

Программирование в данном проекте начнем с кнопки «**Ввод**». В начале процедуры описываем используемые переменные как переменные целого типа. В процедуре используются следующие переменные: **x** – исходное трехзначное число; **r1** – старший разряд числа; **r2** – средний разряд числа; **r3** – младший разряд числа; **v** – остаток от деления исходного числа на 100 (вспомогательная переменная); **Sum** – искомая сумма цифр трехзначного числа. Описание этих переменных выполняется следующей командой:

```
Dim x,r1,r2,r3,v,Sum As Integer
```

Затем необходимо получить исходное значение трехзначного числа, которое вводится в верхнее текстовое окно. Это значение нужно преобразовать из текстовой формы в числовую с помощью функции **Val** и присвоить переменной **x**. Данные действия выполняются с помощью следующего оператора:

```
x = Val(TextBox1.Text)
```

Далее можно определить старшую цифру трехзначного числа и присвоить его переменной **r1**, разделив это число с помощью операции целочисленного деления на 100:

```
r1 = x \ 100
```

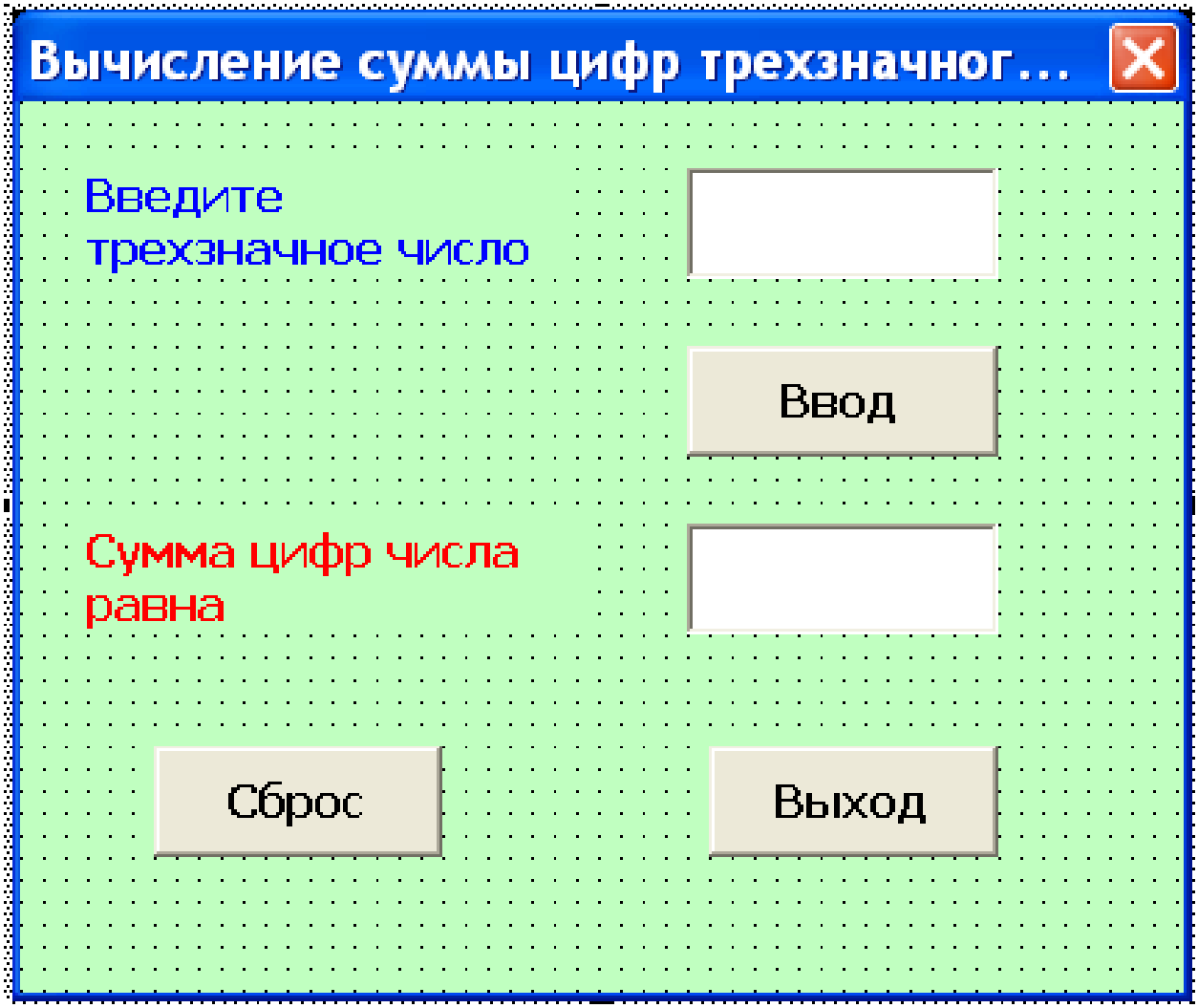


Рис. 2.3. Пользовательский интерфейс программы «Вычисление суммы цифр трехзначного числа»

На следующем этапе нужно найти остаток от деления исходного числа на 100 с помощью операции **Mod** и присвоить полученное значение вспомогательной переменной **v**:

```
v = x Mod 100
```

Затем находим среднюю цифру числа **r2** путем целочисленного деления вспомогательной переменной **v** на 10 и младшую цифру числа **r3** посредством нахождения остатка при делении переменной **v** на 10:

```
r2 = v \ 10
r3 = v Mod 10
```

Наконец, сложив все найденные значения цифр трехзначного числа, находим искомый результат и присваиваем его переменной **Sum**:

```
Sum = r1 + r2 + r3
```

Последнее, что осталось сделать в данной процедуре, – это преобразовать полученный результат в текстовую форму для вывода его во втором текстовом окне:

```
TextBox2.Text = Str(Sum)
```

Таким образом, полный текст процедуры для экранной кнопки **CommandButton1** (кнопка «**Ввод**») будет следующим:

```
Private Sub CommandButton1_Click()
Dim x, r1, r2, r3, v, Sum As Integer
x = Val(TextBox1.Text)
r1 = x \ 100
v = x Mod 100
r2 = v \ 10
r3 = v Mod 10
Sum = r1 + r2 + r3
TextBox2.Text = Str(Sum)
End Sub
```

Программирование кнопки **CommanButton2** (кнопка **«Сброс»)** выполняется аналогично тому, как это было сделано в предыдущем проекте, т.е. выполняется очистка текстовых окон **Textbox1** и **Textbox2** путем присваивания им «нулевого значения». Также по аналогии с предыдущими проектами программируется и кнопка **CommandButton3** (кнопка «**Выход**»). Ниже приводится полный текст программного модуля для данного проекта:

```
Private Sub CommandButton1_Click()
Dim x, r1, r2, r3, v, Sum As Integer
x = Val(TextBox1.Text)
r1 = x \ 100
v = x Mod 100
r2 = v \ 10
r3 = v Mod 10
Sum = r1 + r2 + r3
TextBox2.Text = Str(Sum)
End Sub

Private Sub CommandButton2_Click()
TextBox1.Text = ""
TextBox2.Text = ""
End Sub

Private Sub CommandButton3_Click()
End
End Sub
```

После завершения работы над программным кодом можно запустить приложение на выполнение. Внешний вид приложения, находящегося в рабочем режиме, показан на рис. 2.4.

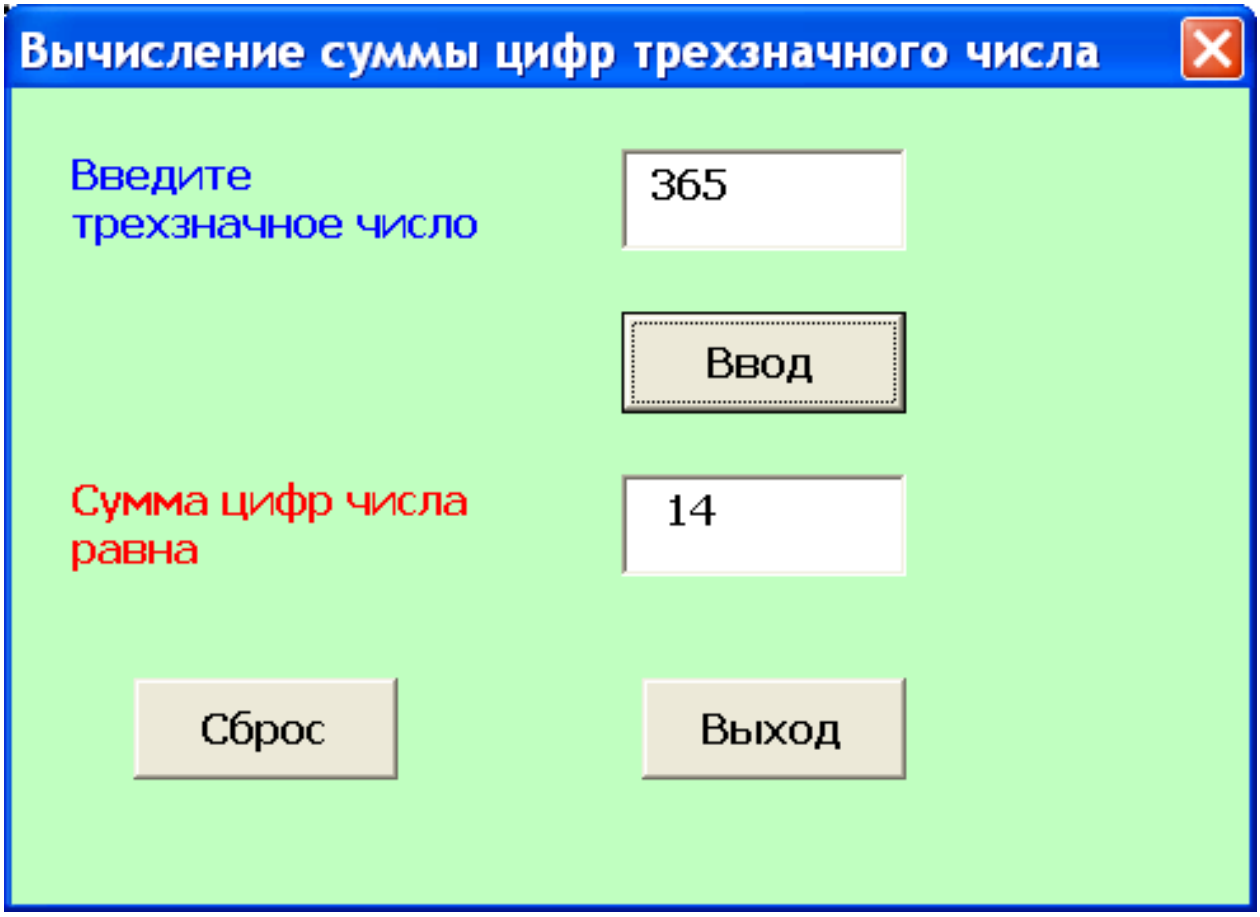


Рис. 2.4. Программа «Вычисление суммы трехзначного числа» в процессе выполнения

Далее следует проверить работу экранных кнопок «**Ввод**» и «**Сброс**», введя в верхнее окно несколько трехзначных чисел а затем очистив окна от результатов вычислений. После выполнения проверки можно завершить тестирование проекта щелчком на экранной кнопке «**Выход**».

Вопросы для самоконтроля к главе 2

1. Каковы основные правила наименования переменных в языке Visual Basic?
2. Какие символы применяются в Visual Basic для обозначения операций сравнения?
3. Какая команда используется для явного объявления переменных в Visual Basic?
4. Укажите основные типы переменных, имеющиеся в Visual Basic.
5. Как выглядит в Visual Basic оператор присваивания?
6. Какая стандартная функция применяется для преобразования строковой величины в числовую?
7. Какая стандартная функция применяется для преобразования числовой величины в строковую?
8. Каким образом в VBA следует производить очистку текстовых окон перед повторным вводом исходных данных?
9. Какой символ применяется в Visual Basic для обозначения операции целочисленного деления?
10. Как обозначается в Visual Basic операция нахождения остатка при целочисленном делении?

Глава 3. Операторы выбора в VBA

3.1. Использование условного оператора. Определение суммы скидки на продукцию фирмы

При решении многих задач из области программирования возникает необходимость выбора одного варианта действий из двух или более возможных. Для того, чтобы правильно осуществить этот выбор, в языках программирования используются специальные операторы. Такие операторы есть и в языке VBA. Если необходимо выбрать один вариант из двух возможных, то применяется оператор условного перехода. Оператор условного перехода выбирает тот или иной вариант действия в зависимости от значения некоторого условия. Общий вид этого оператора в языке VBA следующий:

```
If условие Then вариант 1 Else вариант 2
```

Механизм работы этого оператора следующий: вначале проверяется условие, находящееся после служебного слова **If**. В качестве такого условия обычно выступает какая-либо операция сравнения. Результатом выполнения такой операции является логическая величина, которая может принимать только два значения: **True** (истина) или **False** (ложь). Если условие истинно, то программа выполняет **вариант 1**, находящийся после служебного слова **Then**. Если же условие ложно, то выполняется **вариант 2**, который находится после служебного слова **Else**. Каждый из этих вариантов в свою очередь может представлять собой один оператор или группу операторов языка VBA.

Приведенная выше форма записи условного оператора **If ... Then ... Else** называется однострочной, так как все служебные слова и соответствующие им варианты записаны в одну строку. Эта форма записи не всегда бывает удобна для программиста. Если какой-либо из вариантов (или оба) представляют собой достаточно длинную последовательность операторов, то строка выйдет за пределы экрана, и для того чтобы отредактировать такую строку (или даже просто просмотреть ее содержимое) придется постоянно «прокручивать» ее в ту или иную сторону.

Поэтому наряду с однострочной формой записи применяется и другая форма записи условного оператора, называемая блочной. Для блочной формы записи общий вид данного оператора будет следующим:

```
If условие Then
вариант 1
Else
вариант 2
End If
```

Синтаксис оператора при такой форме записи отличается наличием служебных слов **End If**, которые необходимы для того, чтобы можно было определить, где заканчивается данный оператор. Механизм работы условного оператора при блочной форме записи не отличается от механизма при однострочной, но блочная форма записи часто бывает удобней для восприятия и понимания текста программы.

Тот вид условного оператора, который описан выше, называется полным условным оператором (это относится и к однострочной, и к блочной форме записи). Наряду с ним в программировании используется и сокращенный условный оператор. Общий вид сокращенного условного оператора (для однострочной формы записи) следующий:

```
If условие Then вариант
```

В этом случае механизм работы условного оператора изменится следующим образом. Если условие, находящееся после служебного слова **If**, является истинным, то выполняется тот вариант, который указан после слова **Then** (в данном случае описываемый вариант так же может представлять собой один оператор или группу операторов). Но если условие, находящееся после **If**, оказывается ложным, то данный оператор не выполняет никаких действий и программа переходит к выполнению следующего оператора. При блочной форме записи сокращенный условный оператор в общем виде будет выглядеть следующим образом:

```
If условие Then
вариант
End If
```

Можно применять условный оператор и для выбора одного варианта из трех имеющихся. В этом случае в общем виде условный оператор будет выглядеть так:

```
If условие 1 Then
вариант 1
ElseIf  условие 2 Then
вариант 2
Else
вариант 3
End If
```

Для такого условного оператора применяется следующий механизм работы. В начале проверяется **условие 1**, находящееся в верхней строке оператора после служебного слова **If**. Если это условие истинно (его значение равно **True**), то выполняется **вариант 1**, расположенный между служебными словами **Then** и **ElseIf**.

Если же исходное **условие 1** ложно (его значение равно **False**), то производится новая проверка. На этот раз проверяется **условие 2**. Если значение этого условия равно **True**, то выполняется **вариант 2**, находящийся между вторым служебным словом **Then** и **Else**. Если же **условие 2** тоже оказывается ложным, то выполняется **вариант 3**, расположенный между служебными словами **Else** и **End If**.

При необходимости же выбора одного из еще большего количества имеющихся вариантов рекомендуется использовать оператор множественного выбора, который будет подробно рассмотрен в разд. 3.2.

Рассмотрим применение условного оператора на следующем примере. Некоторая компьютерная фирма, для того чтобы стимулировать клиентов к приобретению ее продукции, предложила следующую схему. Любой клиент, который приобретает у фирмы ее продукцию на сумму свыше 10000 руб., получает скидку в размере 5 % от стоимости приобретаемой продукции.

Требуется разработать программу, которая в зависимости от введенной пользователем суммы определяет размер скидки на продукцию и сообщает пользователю стоимость приобретаемой им продукции с учетом скидки. Если же приобретаемая пользователем продукция стоит менее 10000 руб., то программа должна выдать сообщение о том, что данный пользователь не имеет права на скидку.

Интерфейс данной программы должен включать следующие элементы:

1. Три текстовых окна. В первое окно **Textbox1** будет вводиться исходная сумма. Во втором окне **Textbox2** будет выводится сумма скидки. Это окно по умолчанию сделаем невидимым, так как в случае отсутствия у пользователя права на скидку в наличии окна **Textbox2** на экранной форме нет смысла. Это окно будет становиться видимым только в том случае, если сумма заказа превышает 10000 руб.

Для того чтобы сделать окно **Textbox2** невидимым, в окне свойств для данного объекта найдем свойство **Visible** и установим для него значение **False**. При работе в режиме проектирования программы это окно все равно будет оставаться видимым, но как только мы запустим программу на выполнение, это окно «исчезнет» с экранной формы и «проявится» только при указанном выше условии.

В третьем текстовом окне **Textbox3** будет отображаться стоимость покупки с учетом положенной клиенту скидки, которая равна исходной сумме покупки за вычетом этой скидки. Если скидка клиенту не положена, то величина, отображаемая в текстовом окне **Textbox3,** будет равна исходной величине, вводимой в окно **Textbox1**.

2. Три надписи **Label1, Label2** и **Label3**, которые поясняют пользователю назначение каждого из текстовых окон. Для текста первой надписи **Label1** («Введите сумму заказа») зададим синий цвет, для текстов второй и третьей надписи (вывод результатов расчетов) зададим красный цвет. Вторую надпись **Label2** («Сумма скидки составляет») подобно окну

Textbox2, также сделаем по умолчанию невидимой, так как ее наличие имеет смысл только при сумме заказа, превышающей 10000 руб. Третья надпись **Label3** сообщает об итоговой сумме покупки.

3. Четыре экранные кнопки **CommandButton1, CommandButton2, CommandButton3** и **CommandButton4**. Щелчок на кнопке **CommandButton1** («**Расчет**») позволяет подсчитать сумму скидки и итоговую стоимость покупки с учетом скидки. Кнопка **CommandButton2** («**Сброс**») очищает все три текстовых окна и делает невидимым второе текстовое окно и вторую надпись, для того чтобы подготовить программу к новому вводу данных и дальнейшим расчетам. Кнопка **CommandButton3** («**Об авторе**») выводит на экран дополнительное диалоговое окно со сведениями о разработчике программы. Кнопка **CommandButton4** («**Выход**») закрывает основную экранную форму.

Пользовательский интерфейс программы приведен на рис.3.1.

Написание программного кода начнем с кнопки **CommandButton1**. В процедуре **CommandButton1_Click**, описывающей реакцию этой экранной кнопки на щелчок левой кнопкой мыши введем следующие локальные переменные: **Sum** – исходная стоимость приобретаемой продукции; **Skidka** – размер скидки, положенной клиенту фирмы; **Itog** – итоговая стоимость приобретаемой продукции, вычисляемая с учетом скидки. Описываем **Sum** как переменную целого типа, а переменные **Skidka** и **Itog** – как вещественные переменные двойной точности.

Затем выполняем с помощью функции **Val** преобразование значения исходной суммы, введенного в окно **Textbox1,** из текстовой формы в числовую и присваиваем получившееся числовое значение переменной **Sum**.

Создаем в процедуре условный оператор и в заголовке этого оператора после служебного слова **If** записываем следующее условие: **Sum>10000**.

Дальнейший ход действий в программе зависит от значения переменной **Sum**. Если значение этой переменной более 10000, то необходимо выполнить следующие действия:

1. Вычислить величину скидки, взяв 5 % от исходной суммы, и присвоить полученное значение переменной **Skidka**.

2. Подсчитать итоговую стоимость приобретаемой продукции, которая будет равна исходной за вычетом скидки. Полученное значение присвоить переменной **Itog**.

3. Сделать видимыми текстовое окно **Textbox2** и соответствующую ему надпись **Label2**.

4. Надписи **Label2** присвоить значение: «Сумма скидки составляет».

5. Преобразовать вычисленную сумму скидки и итоговую стоимость из числовой формы в текстовую с помощью функции **Str** и вывести эти текстовые величины соответственно в окнах **Textbox2** и **Texbox3**.

Все эти действия записываются в условном операторе после служебного слова **Then**.

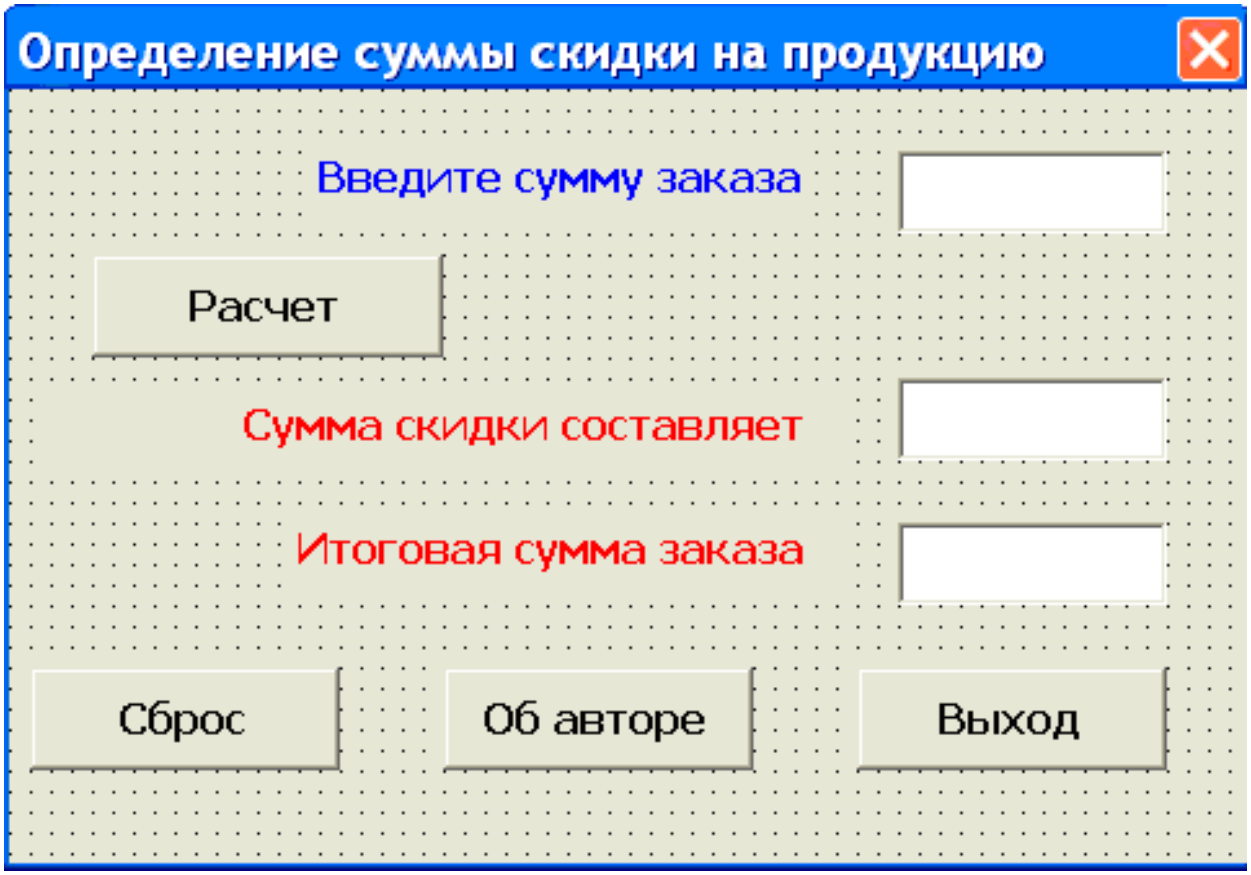


Рис. 3.1. Пользовательский интерфейс программы «Вычисление суммы скидки на приобретаемую продукцию»

Если сумма приобретаемой продукции меньше 10000, то нужно сделать следующее:

1. Присвоить переменной **Itog** значение, равное переменной **Summa**, так как итоговая сумма будет равна исходной ввиду отсутствия скидки, преобразовать это значение переменной **Itog** в текстовую форму и вывести это значение в окне **Textbox3**.

2. Сделать видимой надпись **Label2** и присвоить этой надписи следующее значение: «Извините, скидка Вам не положена».

Эти действия записываются в условном операторе после служебного слова **Else**.

В целом, исходя из вышеуказанного, условный оператор в данной процедуре будет выглядеть следующим образом:

```
If Sum > 10000 Then
  Skidka = Sum * 0.05
  TextBox2.Text = Str(Skidka)
  TextBox2.Visible = True
  Itog = Sum - Skidka
  TextBox3.Text = Str(Itog)
```

```
  Label2.Caption  =  "                    Сумма  скидки
составляет"
  Label2.Visible = True
Else
  Itog = Sum
  TextBox3.Text = Str(Itog)
  Label2.Visible = True
  Label2.Caption  =  "Извините,  скидка  Вам  не
положена"
End If
```

Далее, записываем код процедуры для экранной кнопки **CommandButton2**. Эта процедура должна делать следующее:

1. Очистить все имеющиеся в программе текстовые окна – **TextBox1**, **TextBox2** и **Textbox3**.

2. Сделать невидимой надпись **Label2** и текстовое окно **Textbox2** до следующего ввода исходных данных.

В случае с первой операцией свойству **Text** всех трех текстовых окон присваивается «нулевое» значение. Для выполнения второй операции свойству **Visible** обоих рассматриваемых объектов присваивается значение **False**. Затем пишем процедуры для экранных кнопок **CommandButton3** («**Об авторе**») и **CommandButton4** («**Выход**»). Данные кнопки программируются аналогично тому, как это делалось в предыдущих проектах.

После написания программного кода для всех четырех экранных кнопок создание программного модуля для данного проекта можно считать завершенным. Полный текст программного модуля приводится ниже:

```
Private Sub CommandButton1_Click()
Dim Sum As Integer
Dim Skidka, Itog As Double
Sum = Val(TextBox1.Text)
If Sum > 10000 Then
Skidka = Sum * 0.05
TextBox2.Text = Str(Skidka)
TextBox2.Visible = True
Itog = Sum - Skidka
TextBox3.Text = Str(Itog)
Label2.Caption = "          Сумма скидки составляет"
Label2.Visible = True
Else
Itog = Sum
TextBox3.Text = Str(Itog)
Label2.Visible = True
Label2.Caption = "Извините, скидка Вам не положена"
```

```
End If
End Sub

Private Sub CommandButton2_Click()
TextBox1.Text = ""
TextBox2.Text = ""
TextBox3.Text = ""
Label2.Visible = False
TextBox2.Visible = False
End Sub

Private Sub CommandButton3_Click()
MsgBox ("Программу разработал А.Н. Маслобоев")
End Sub

Private Sub CommandButton4_Click()
End
End Sub
```

После завершения работы над программой следует запустить ее на выполнение и протестировать программу при различных исходных данных, вводимых в первое текстовое окно.

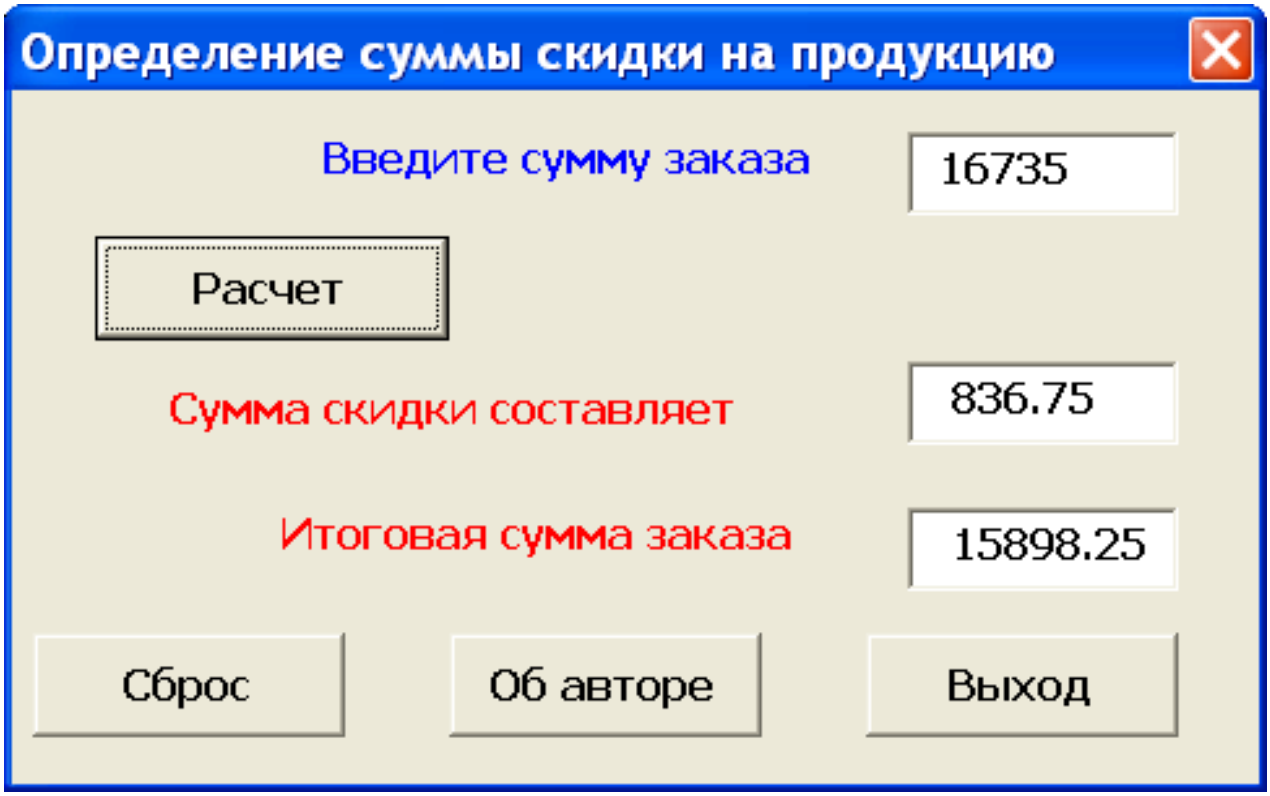


Рис.3.2. Программа «Вычисление суммы скидки на продукцию» в процессе работы при наличии скидки

На рис 3.2 показан вариант работы приложения, в котором клиенту компании полагается скидка, так как он приобретает продукцию на сумму более 10000 руб.

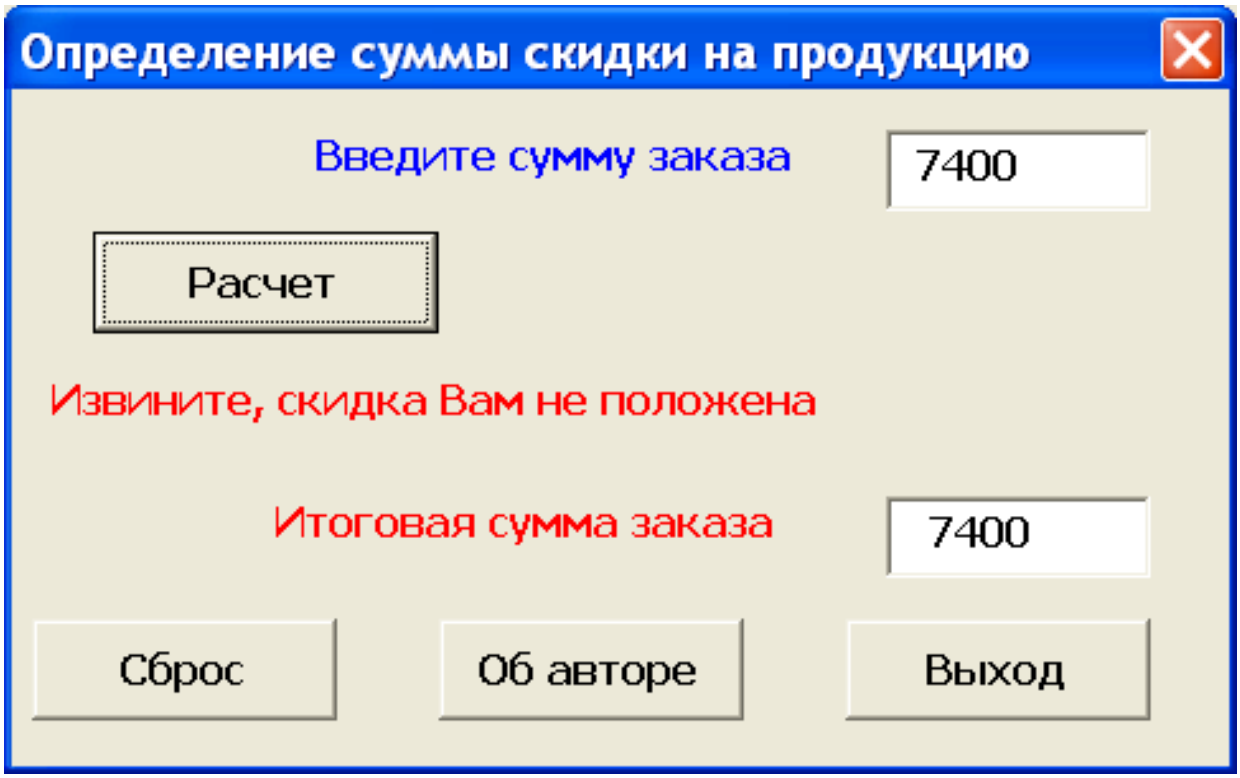


Рис 3.3. Программа «Вычисление суммы скидки на продукцию» в процессе работы при отсутствии скидки

На рис. 3.3 показано, как выглядит экран приложения при отсутствии у клиента права на скидку. После ввода нескольких различных значений исходной суммы для проверки работоспособности приложения, а также проверки работы кнопки «**Об авторе**», можно завершить выполнение программы щелчком на кнопке «**Выход**».

3.2. Оператор множественного выбора. Модификация программы определения суммы скидки

В предыдущем разделе пособия был рассмотрен вопрос о выборе одного варианта действия из двух или трех возможных, но часто возникает необходимость выбрать подходящий вариант из большего количества имеющихся.

Для этой цели можно использовать вложенные условные операторы **If...Then...Else** (вложенным называется оператор, находящийся внутри другого оператора). В каждую из ветвей условного оператора, находящуюся после служебного слова **Then** или после слова **Else** (а также после обоих этих слов), можно вставить еще один условный оператор, что увеличивает количество выбираемых вариантов. Можно вставлять вложенные операторы и внутри других вложенных. Но практика программирования показывает, что при большом количестве вложений программа становится плохо читаемой и понимаемой и с трудом поддается корректировке в случае наличия в ней ошибок.

Поэтому при выборе одного варианта из нескольких возможных для решения задачи целесообразно использовать оператор множественного выбора **Select Case**. Данный оператор позволяет выбрать один из нескольких вариантов действия в зависимости от значения переменной, называемой переменной-селектором (от английского глагола ***to select*** – выбирать). Общий вид данного оператора следующий:

```
Select Case N
Case N1
вариант 1
Case N2
вариант 2
……….……
Case Nk
вариант k
Case Else
вариант k+1
End Select
```

В данной записи **Select Case** – служебные слова, которые открывают оператор множественного выбора; **N** – имя переменной-селектора; **N1,N2...Nk** – возможные значения переменной-селектора**.** В простейшем случае значение переменной-селектора представляет собой целое число, но в общем случае после любого из слов **Case** может быть записано и выражение, значение которого вычисляется и присваивается переменной-селектору.

Каждому из указанных в операторе значений переменной селектора соответствует свой вариант действия. В приведенной записи эти варианты

обозначены как **вариант 1, вариант 2,… вариант k**. Каждый из этих вариантов может представлять собой либо один из операторов языка Visual Basic, либо группу операторов. Если же переменная-селектор **N** не принимает ни одного из перечисленных значений, то выполняется вариант **k+1**, который записан в операторе после служебных слов **Case Else**. Оператор множественного выбора должен завершаться служебными словами **End Select**.

Приведенная выше форма записи оператора множественного выбора называется полной формой записи. Наряду с ней используется сокращенная форма записи оператора. Отличие сокращенной формы заключается в том, что в ней отсутствуют служебные слова **Case Else** и соответствующий им вариант **k+1**, что приводит к изменению механизма работы оператора. Изменение заключается в следующем: если переменная-селектор **N** не принимает ни одного из значений, перечисленных в операторе множественного выбора, то оператор просто не выполняет никаких действий, и программа переходит к выполнению следующего за ним оператора.

Следует отметить еще одну особенность оператора множественного выбора. Каждому из указанных в операторе вариантов может соответствовать не одно значение переменной-селектора, а целая группа значений или даже определенный диапазон значений переменной.

Если одному варианту соответствуют несколько значений переменной, то они должны быть перечислены через запятую. В общем виде подобная запись выглядит так:

```
Case N1, N2,....,Nt
вариант X
```

где **N_1, N2,....,Nt** – возможные значения переменной-селектора; **вариант X** – вариант действий, соответствующий этой группе переменных.

Если же значения переменной-селектора образуют непрерывный диапазон, то запись в общем виде представляется таким образом:

```
Case Nn to Nk
вариант Y
```

где **Nn –** начальная величина диапазона значений переменной-селектора; **Nk –** конечная величина этого диапазона; **вариант Y –** вариант действий, соответствующий этому диапазону.

Рассмотрим использование оператора множественного выбора **Case** на следующем примере. За основу возьмем предыдущий проект, в котором вычисляется сумма скидки на продукцию компьютерной фирмы. Но теперь мы усложним поставленную задачу, так как скидка будет дифференцироваться в зависимости от стоимости заказа.

При сумме заказа менее 1000 руб. клиент не имеет права на скидку. Если сумма заказа составляет от 1000 до 4999 руб. включительно, то размер

скидки составит 3 %. Если стоимость заказа составит от 5000 до 19999 руб. включительно, то скидка увеличивается до 5 %. Если же клиент приобретает у фирмы продукцию на сумму от 20000 руб. и выше, то он получает право на скидку в размере 10 %. Таким образом, в данной задаче необходимо предусмотреть 4 различных варианта работы программы в зависимости от суммы заказа, которая и будет в данном случае выступать в качестве переменной-селектора.

Описание пользовательского интерфейса данного проекта мы приводить не будем, так как он полностью идентичен интерфейсу предыдущего проекта, который подробно описан в разделе 3.1 данного пособия и приведен на рис. 3.1. Отличие же данного проекта от предыдущего будет заключаться в том, что мы изменим программный код процедуры для экранной кнопки «**Расчет**» в соответствии с заданными условиями.

Используемые переменные останутся теми же, что и ранее: **Sum** – исходная сумма заказа, **Skidka** – величина полагающейся клиенту скидки, **Itog** – итоговая стоимость заказа с учетом скидки. Первую переменную опишем как переменную целого типа, а вторую и третью – как вещественные переменные с двойной точностью.

В предыдущем проекте для выбора варианта действия использовался условный оператор **If ... Then ... Else**, так как предлагаемых вариантов выбора было всего два. Поскольку количество выбираемых вариантов увеличилось, то в новом проекте вместо условного оператора мы используем оператор множественного выбора. В качестве переменной-селектора в этом операторе используется переменная **Sum**.

В первом случае (когда сумма заказа составляет менее 1000 руб., и, следовательно, право на скидку у клиента отсутствует) значение переменной-селектора **Sum** будет находиться в диапазоне от 1 до 999 руб., что записывается следующим образом: **Case 1 To 999**. Этому диапазону значений соответствует следующий вариант действий:

1. Необходимо сообщить пользователю о том, что у него отсутствует право на скидку, для чего изменить с помощью оператора присваивания значение свойства **Caption** для объекта **Label2**.

2. Сделать невидимым текстовое окно **Textbox2**, в котором указывается сумма скидки, так как скидка отсутствует.

3. В окне **Textbox3**, где отображается итоговая сумма заказа, вывести то же значение, что и в окне **Textbox1**, в которое вводится исходная сумма.

Группа операторов, описывающая вышеуказанные действия, будет выглядеть следующим образом:

```
Case 1 To 999
Label2.Caption = "Извините, скидка Вам не положена"
TextBox2.Visible = False
TextBox3.Text = Sum
```

Если сумма заказа составляет от 1000 до 4999 руб. (причем клиент фирмы получает право на 3%-ю скидку), то значение переменной-селектора **Sum** будет записываться таким образом: **Case 1000 To 4999**. Данному диапазону значений переменой-селектора должен соответствовать следующий вариант действий:

1. Необходимо сообщить пользователю о том, что он обладает правом на получение скидки, для чего изменить свойство **Caption** объекта **Label2**.

2. Вычислить сумму скидки, равную в данном случае 3 %, для чего умножить сумму заказа на коэффициент 0.03 и присвоить полученное значение переменной **Skidka**.

3. Сделать видимым второе текстовое окно (объект **TextBox2**), так как информация о размере скидки становится актуальной для пользователя.

4. В текстовом окне **Texbox2** ввести вычисленное значение скидки (для преобразования численной величины в строковую как обычно используем функцию **Str**).

5. Подсчитать итоговую стоимость заказа с учетом скидки и присвоить полученное значение переменной **Itog**.

6. Вывести полученное значение в текстовом окне **Textbox3**, преобразовав его в строковую форму с помощью функции **Str**.

Ниже приводится группа операторов, соответствующая этому варианту:

```
Case 1000 To 4999
Label2.Caption = "Сумма скидки составляет "
Skidka = Sum * 0.03
TextBox2.Visible = True
TextBox2.Text = Str(Skidka)
Itog = Sum - Skidka
TextBox3.Text = Str(Itog)
```

Если сумма заказа составляет от 5000 до 19999 руб., то этот диапазон значений переменной-селектора **Sum** записывается таким образом: **Case 5000 To 19999**. Вся последовательность действий аналогична предыдущему варианту за исключением того, что при вычислении суммы скидки будет использоваться коэффициент 0.05 (а не 0.03 как в варианте, приведенном ранее). Соответствующий фрагмент программы будет выглядеть так, как показано ниже:

```
Case 5000 To 19999
Label2.Caption = "Сумма скидки составляет "
Skidka = Sum * 0.05
TextBox2.Visible = True
TextBox2.Text = Str(Skidka)
Itog = Sum - Skidka
TextBox3.Text = Str(Itog)
```

Осталось рассмотреть последний вариант, когда сумма заказа превышает 20000 руб. Для этого варианта используем еще одну форму записи значений переменной-селектора:

```
Case Is Оператор сравнения
```

Оператор сравнения включает одну из операций сравнения (больше, меньше, равно, больше или равно, меньше или равно, не равно) и численное значение, с которым сравнивается значение переменной-селектора. В нашем примере варианту с суммой заказа более 20000 руб. будет соответствовать следующая запись:

```
Case is > 20000
```

Действия, производимые в этой ветви оператора множественного выбора, аналогичны двум предыдущим вариантам, но значение коэффициента для вычисления суммы скидки будет равно 0.1. Соответствующий фрагмент программы приводим ниже:

```
Case Is > 20000
Label2.Caption = "Сумма скидки составляет "
Skidka = Sum * 0.1
TextBox2.Visible = True
TextBox2.Text = Str(Skidka)
Itog = Sum - Skidka
TextBox3.Text = Str(Itog)
```

Объединив все вышеописанные варианты, мы в итоге получим в нашем проекте для кнопки «**Расчет**» следующий программный код:

```
Private Sub CommandButton1_Click()
Dim Sum As Integer
Dim Skidka, Itog As Double
Sum = Val(TextBox1.Text)
Select Case Sum
Case 1 To 999
Label2.Caption = "Извините, скидка Вам не положена"
TextBox2.Visible = False
TextBox3.Text = Sum
Case 1000 To 4999
Label2.Caption = "Сумма скидки составляет "
Skidka = Sum * 0.03
TextBox2.Visible = True
TextBox2.Text = Str(Skidka)
Itog = Sum - Skidka
TextBox3.Text = Str(Itog)
Case 5000 To 19999
```

```
Label2.Caption = "Сумма скидки составляет "
Skidka = Sum * 0.05
TextBox2.Visible = True
TextBox2.Text = Str(Skidka)
Itog = Sum - Skidka
TextBox3.Text = Str(Itog)
Case Is > 20000
Label2.Caption = "Сумма скидки составляет "
Skidka = Sum * 0.1
TextBox2.Visible = True
TextBox2.Text = Str(Skidka)
Itog = Sum - Skidka
TextBox3.Text = Str(Itog)
End Select
End Sub
```

Таким образом, изменив программный код для кнопки «**Расчет**» (без изменения интерфейса приложения «Определение суммы скидки» и изменения программного кода для других экранных кнопок) мы получаем возможность производить выбор не из 2, а из 4 возможных вариантов, причем в случае необходимости количество рассматриваемых в приложении вариантов всегда можно увеличить.

Вопросы для самоконтроля к главе 3

1. Опишите общий вид условного оператора в языке Visual Basic.
2. Чем отличаются однострочная форма записи условного оператора и блочная форма записи в Visual Basic?
3. Чем отличаются полная и сокращенная форма записи условного оператора в Visual Basic?
4. Как записывается в общем виде оператор, который позволяет выбрать один вариант действий из трех имеющихся?
5. Для каких целей используется оператор множественного выбора?
6. Как записывается в общем виде оператор множественного выбора?
7. Чем отличаются друг от друга полная и сокращенная формы записи оператора множественного выбора?
8. Как в операторе множественного выбора указать, что одному варианту действий соответствуют несколько значений переменной-селектора?
9. Как записываются значения переменной-селектора, если они образуют непрерывный диапазон?
10. Как используются в операторе множественного выбора операции сравнения?

Глава 4. Операторы цикла в VBA

4.1. Программа «Вычисление факториала»

При решении многих задач из области информационных технологий часто возникает необходимость многократного повторения одного и того же оператора или группы операторов. Для этой цели в языке Visual Basic (как и в других языках программирования) предусмотрена такая структура, как цикл. В Visual Basic существует две основные циклические структуры: цикл с заранее известным числом повторений и цикл с заранее неопределенным числом повторений.

В данном разделе мы рассмотрим проект, в котором используется цикл с заранее известным числом повторений. Общий вид данного цикла в языке Visual Basic следующий:

```
For i:=a To b
тело цикла
Next i
```

Первая строка в данном случае является строкой заголовка, который определяет количество повторений цикла. Переменная **i** называется переменной цикла. Данная переменная должна относиться к целому типу. Эта переменная имеет начальное значение **a** и конечное значение **b,** причем величина **a** должна быть меньше, чем величина **b**. Тело цикла представляет собой собственно тот оператор или группу операторов, которые многократно повторяются в ходе выполнения цикла. Команда **Next i** необходима для того, чтобы однозначно определить, где заканчивается цикл.

Механизм работы данного оператора следующий: вначале переменной цикла присваивается начальное значение **a**, после чего выполняется тело цикла. Затем значение переменной цикла **i** увеличивается на единицу. Далее вновь происходит выполнение тела цикла, и после этого значение переменной цикла увеличивается еще на единицу. Таким образом, увеличение значение данной переменной на единицу производится после каждого очередного выполнения тела цикла до тех пор, пока значение переменной **i** не станет равным конечной величине **b**. Когда это происходит, тело цикла выполняется в последний раз. На этом цикл завершает свою работу, и программа переходит к выполнению оператора, следующего за оператором цикла. Следовательно, количество повторений цикла можно заранее определить и оно будет равно **b-a+1**.

Говоря о цикле с заранее заданным количеством повторений, следует отметить еще два обстоятельства. Во-первых, начальное или конечное значения переменной цикла **a** и **b** могут быть как числовыми константами или переменными, так и выражениями, значения которых вычисляются и

используются в данном операторе. Во-вторых, увеличение значения переменной цикла может происходить не только на единицу, но и на другую величину. Для того, чтобы задать эту величину, которая называется шагом изменения значения переменной цикла, следует в заголовке цикла добавить служебное слово **Step** и после него указать величину шага изменения переменной цикла. В этом случае в общем виде заголовок цикла будет выглядеть так:

```
For i:=a to b Step k,
```

где **k** – шаг изменения переменной цикла.

Наряду с вышеуказанной формой цикла с заранее известным количеством повторений возможна и альтернативная форма цикла, в которой после каждого очередного выполнения тела цикла происходит не увеличение, а уменьшение значения переменной цикла.

В этом случае в заголовке цикла в обязательном порядке должно присутствовать служебное слово **Step,** и после него указывается шаг **k**, который должен быть отрицательной величиной. Тогда после каждого очередного выполнения тела цикла значение переменной цикла **i** будет уменьшаться на величину **k**. Понятно, что в таком случае для правильной работы цикла начальное значение переменной цикла **a** должно быть больше, чем конечное значение **b**.

Рассмотрим применение оператора цикла с заранее известным количеством повторений на следующем примере: необходимо вычислить факториал натурального числа **n**, введенного пользователем с клавиатуры. Понятие факториала применимо только к натуральным числам. Факториалом числа **n** называется произведение всех натуральных чисел от 1 до **n**. В математике факториал обозначается восклицательным знаком: **n!**. Например, факториал числа 5 обозначается как 5! и вычисляется следующим образом:

```
5! = 1*2*3*4*5 = 120.
```

В программе также следует предусмотреть защиту от ввода слишком больших значений числа **n** (условимся, что значение **n** не должно превышать 12). В том случае, если пользователь случайно введет большее число, программа должна выдавать сообщение об ошибке.

Пользовательский интерфейс данного приложения должен включать следующие элементы:

1. Два текстовых окна. Текстовое окно **Textbox1** для ввода исходной величины **n** и окно **Textbox2** для вывода искомого результата – факториала числа **n**.

2. Две надписи **Label1** и **Label2**, поясняющие назначение текстовых окон, указанных в п. 1.

3. Четыре экранные кнопки (объекты **CommandButton**). Экранная кнопка «**Подсчитать**» (**CommandButton1**) либо вычисляет по введенному исходному значению величину факториала, либо выводит дополнительное диалоговое окно с сообщением об ошибке в случае неправильного ввода исходных данных. Кнопка «**Сброс**» (**CommandButton2**) очищает текстовые окна для того, чтобы подготовить их к новому вводу данных и расчету. Кнопка «**Об авторе**» (**CommandButton3)** выводит на экран диалоговое окно со сведениями об авторе данной программы. Кнопка «**Выход**» (**CommandButton4**) закрывает приложение.

Пользовательский интерфейс приложения «Вычисление факториала» приведен на рис. 4.1.

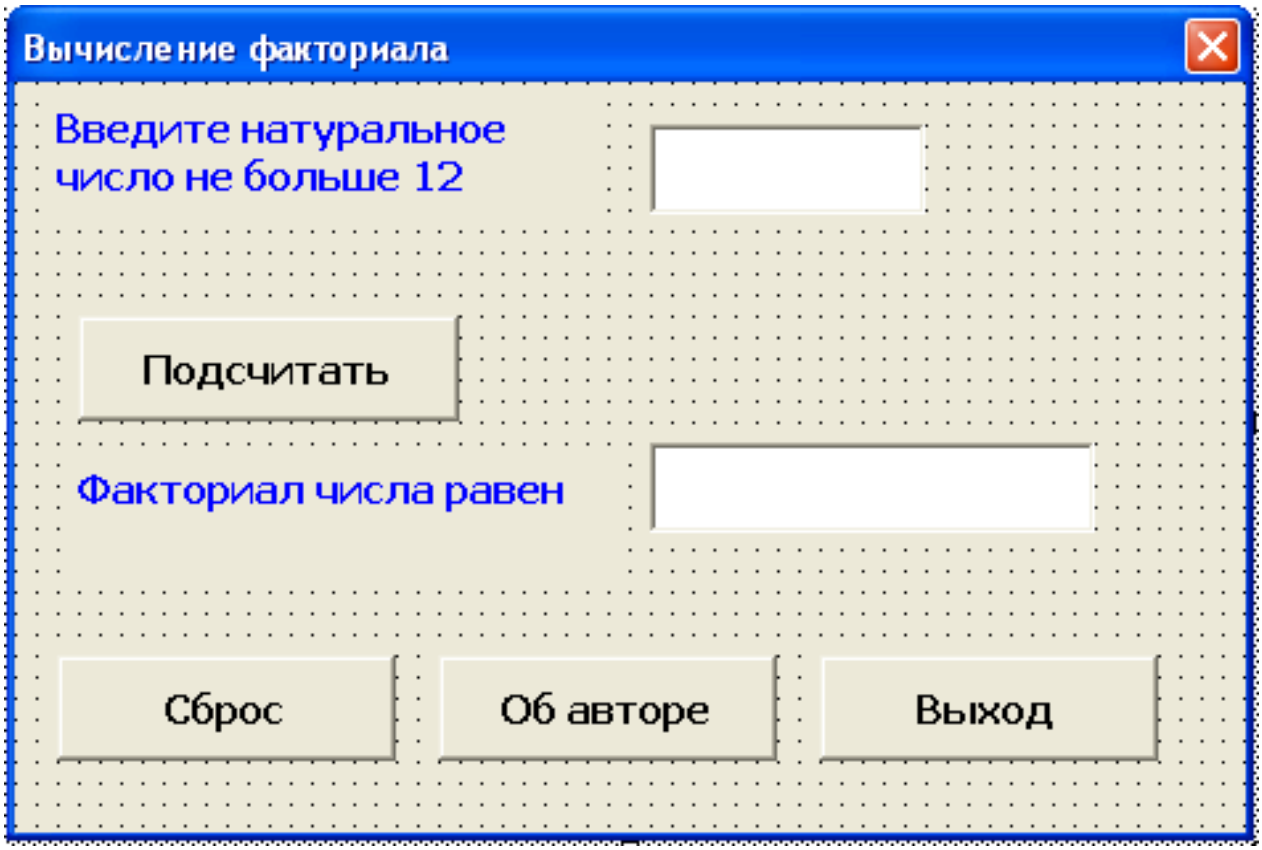


Рис. 4.1. Пользовательский интерфейс программы «Вычисление факториала»

Написание программного кода начнем с кнопки «**Подсчитать**». Вначале описываем используемые переменные. Это исходная величина **n** и искомый факториал **f**. Обе эти величины описываем как переменные «большого» целого типа – типа **Long**, который может оперировать целыми числами до 2 млрд. Затем определяем исходную величину **n** путем преобразования из строковой форму в числовую данных, введенных в окно **Textbox1**.

Дальнейший ход действий будет зависеть от имеющегося значения **n**. Выбор одного из двух возможных вариантов будет производиться с помощью условного оператора **If**. В этом операторе после служебного слова

Then рассматривается случай некорректного ввода исходных данных, когда введеное значение **n** меньше 1 или больше 12 (объединяем оба этих случая с помощью операции **Or** – «логическое или»). В первом случае ввод данных будет неправильным, так как у отрицательных чисел или ноля факториал вообще отсутствует. Во втором случае значение вычисленного факториала превысит 2 миллиарда и выйдет за рамки типа **Long**, что приведет к искажению результата. Поэтому для любого неправильно введенного значения с помощью команды **Msgbox** выводится диалоговое окно, содержащее следующее сообщение: «Ошибка ввода. Число должно быть в диапазоне от 1 до 12».

В том случае, если ввод данных был произведен корректно, программа должна приступить к вычислению факториала **f**. Для этого вначале переменной **f** должно быть присвоено значение, равное 1. Это наименьшее возможное значение факториала. Дальнейшее вычисление значения факториала производится в цикле с заранее известным числом повторений. Вычисление факториала производится путем последовательного умножения его исходного значения на каждое целое число в диапазоне от 1 до **n**. Эти значения последовательно принимает переменная цикла **i**. Приведенный выше алгоритм вычисления факториала на языке Visual Basic может быть записан следующим образом:

```
f = 1
For i = 1 To n
f = f * i
Next i
```

После того, как значение факториала вычислено, его нужно преобразовать из целочисленной формы в строковую и вывести в текстовом окне **Textbox2**. Весь программный код для экранной кнопки «**Расчет**» будет выглядеть так:

```
Private Sub CommandButton1_Click()
Dim n, f As Long
n = Val(TextBox1.Text)
If (n > 12) Or (n < 1) Then
MsgBox ("Ошибка ввода. Число должно быть в
диапазоне от 1 до 12")
Else
f = 1
For i = 1 To n
f = f * i
Next i
TextBox2.Text = Str(f)
End If
End Sub
```

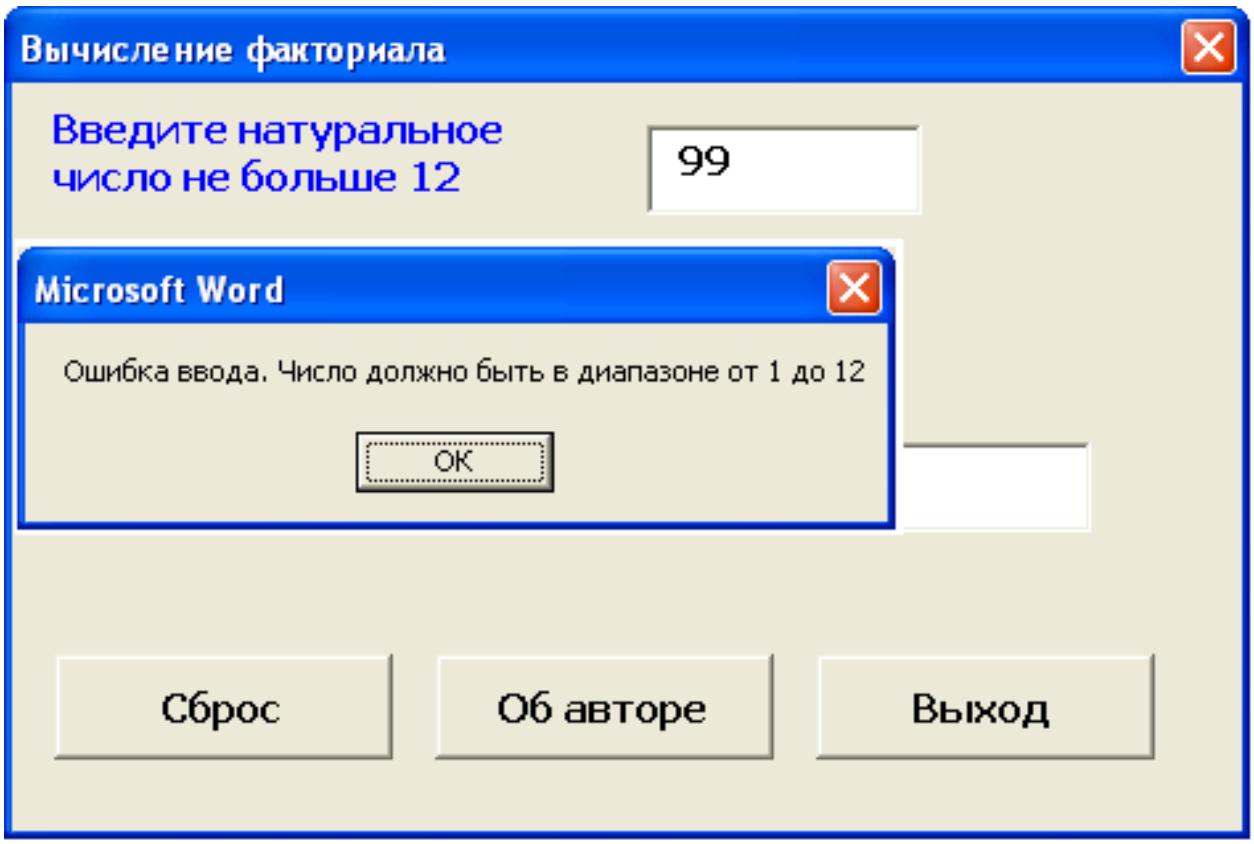


Рис. 4.2. Работа программы «Вычисление факториала при некорректном вводе исходных данных

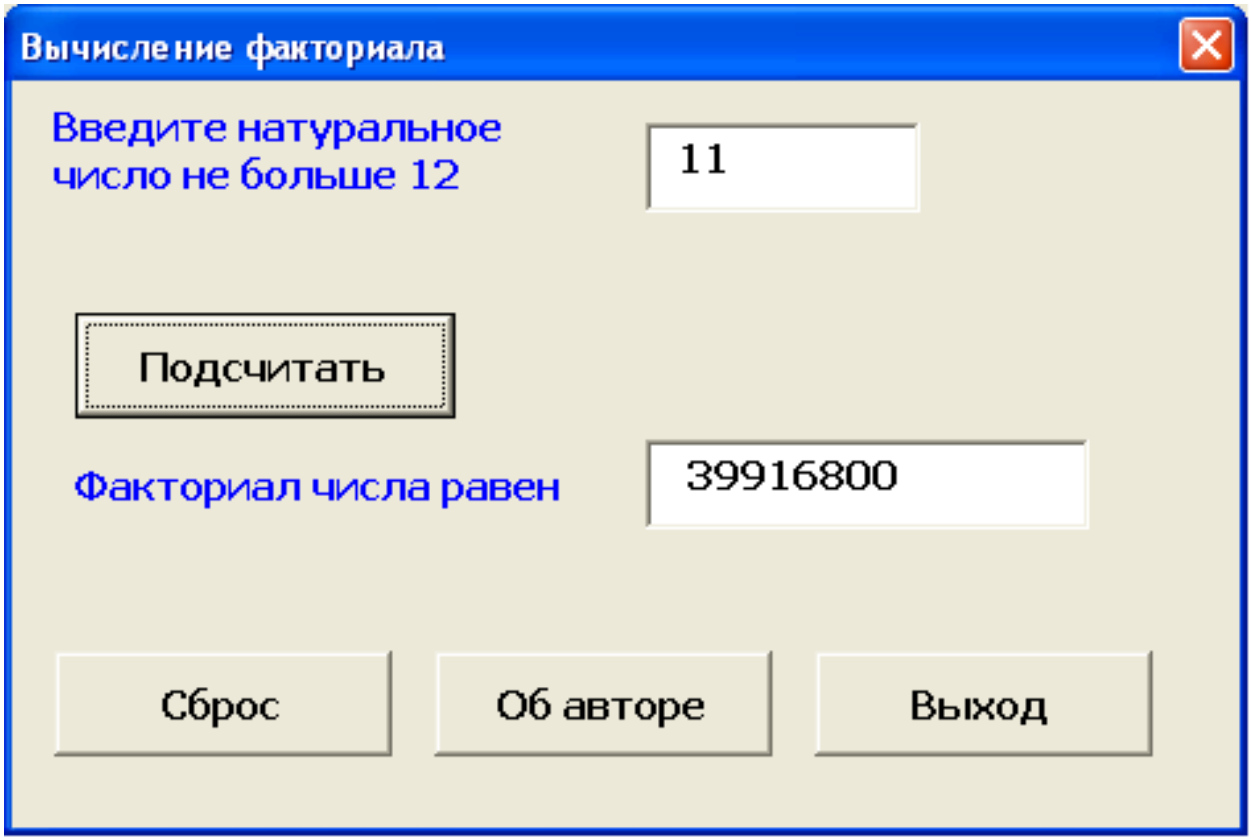


Рис 4.3. Работа программы «Вычисление факториала» при правильном вводе исходных данных

Программирование экранных кнопок «**Сброс**», «**Об авторе**» и «**Выход**» не должно вызвать у пользователя затруднений, так как программирование аналогичных кнопок описывалось в предыдущих главах пособия.

Ниже приводится полный программный код для данного проекта:

```
Private Sub CommandButton1_Click()
Dim n, f As Long
n = Val(TextBox1.Text)
If (n > 12) Or (n < 1) Then
MsgBox ("Ошибка ввода. Число должно быть в
диапазоне от 1 до 12")
Else
f = 1
For i = 1 To n
f = f * i
Next i
TextBox2.Text = Str(f)
End If
End Sub

Private Sub CommandButton2_Click()
TextBox1.Text = ""
TextBox2.Text = ""
End Sub

Private Sub CommandButton3_Click()
MsgBox ("Программу составил А.Н. Маслобоев")
End Sub

Private Sub CommandButton4_Click()
End
End Sub
```

На рис. 4.2 и 4.3 показаны два варианта работы программы при различных исходных данных. В случае, приведенном на рис.4.2, пользователь неверно ввел исходное значение, и поэтому на переднем плане видно окно с сообщением об ошибке. Это окно следует закрыть щелчком на экранной кнопке «**Ok**», затем очистить верхнее текстовое окно щелчком на кнопке «**Сброс**» и заново ввести в него данные (желательно правильные). В примере, приведенном на рис. 4.3, исходные данные введены верно, и в нижнем текстовом окне отображается вычисленное значение факториала.

4.2. Программа «Расчет суммы вклада». Вставка иллюстраций

В данном разделе пособия мы рассмотрим еще один проект, в котором для нахождения искомого значения используется циклическая структура с заранее заданным числом повторений. При разработке пользовательского интерфейса данного проекта мы используем новый, не рассматривавшийся ранее элемент интерфейса – иллюстрацию, а также научимся определять формат выводимых в текстовое окно данных.

Новый проект используется для решения следующей задачи. Клиент положил в банк определенную денежную сумму и заключил с банком договор. В этом договоре должны быть зафиксированы следующие условия: срок, на который заключен данный договор (для упрощения задачи мы условимся, что договор заключается на целое количество лет), и процент по вкладу, который ежегодно начисляется пользователю (в том числе, и на те проценты, которые были начислены ранее). Требуется определить, какая сумма будет находиться на счете у клиента по окончании срока действия договора при условии, что клиент в течении срока действия договора не изменял основную сумму и не снимал с данного вклада проценты.

Пользовательский интерфейс данного проекта должен включать следующие элементы:

1. Четыре текстовых окна. Из этого количества три текстовых окна предназначены для ввода исходных данных и одно – для вывода результата. В окно **Textbox1** вводится исходная сумма вклада, в **Textbox2** – срок, на который клиент заключает договор с банком (целое количество лет), в **Textbox3** – процент по вкладу согласно договору. В окне **Textbox4** должна выводиться сумма вклада по окончании срока действия договора.

2. Четыре надписи **Label1**, **Label2**, **Label3** и **Label4**, которые содержат поясняющие комментарии к содержимому текстовых окон.

3. Три экранные кнопки. Экранная кнопка «**Подсчитать итоговую сумму**» (**CommandButton1**) предназначена для того, чтобы на основе исходных данных, находящихся в трех верхних текстовых окнах, подсчитать искомый результат и вывести его в четвертом окне. Кнопка «**Сброс**» (**CommandButton2**) должна очищать сразу все четыре текстовых окна, для того чтобы в случае необходимости пользователь мог произвести повторный ввод данных и получить новый результат. Кнопка «**Выход**» закрывает окно проекта.

4. Иллюстрация к данному проекту. В качестве иллюстрации возьмем графический файл из коллекции **Clipart**, которая прилагается к программному пакету Microsoft Office. Этот векторный графический файл **Coins.wmf**, который для удобства поместим в ту же папку, где будет находиться наш проект.

Работу над проектом мы, как обычно, начнем с того, что создадим новый документ **Вклад.doc**. Открыв среду программирования VBA, создаем графический интерфейс данного проекта. Так как элементы, подобные тем, что описаны в пп. 1-3, мы создавали уже во многих предыдущих проектах, мы не будет здесь рассматривать их создание и настройку, а более подробно остановимся только на объекте из пункта 4 – иллюстрации к проекту.

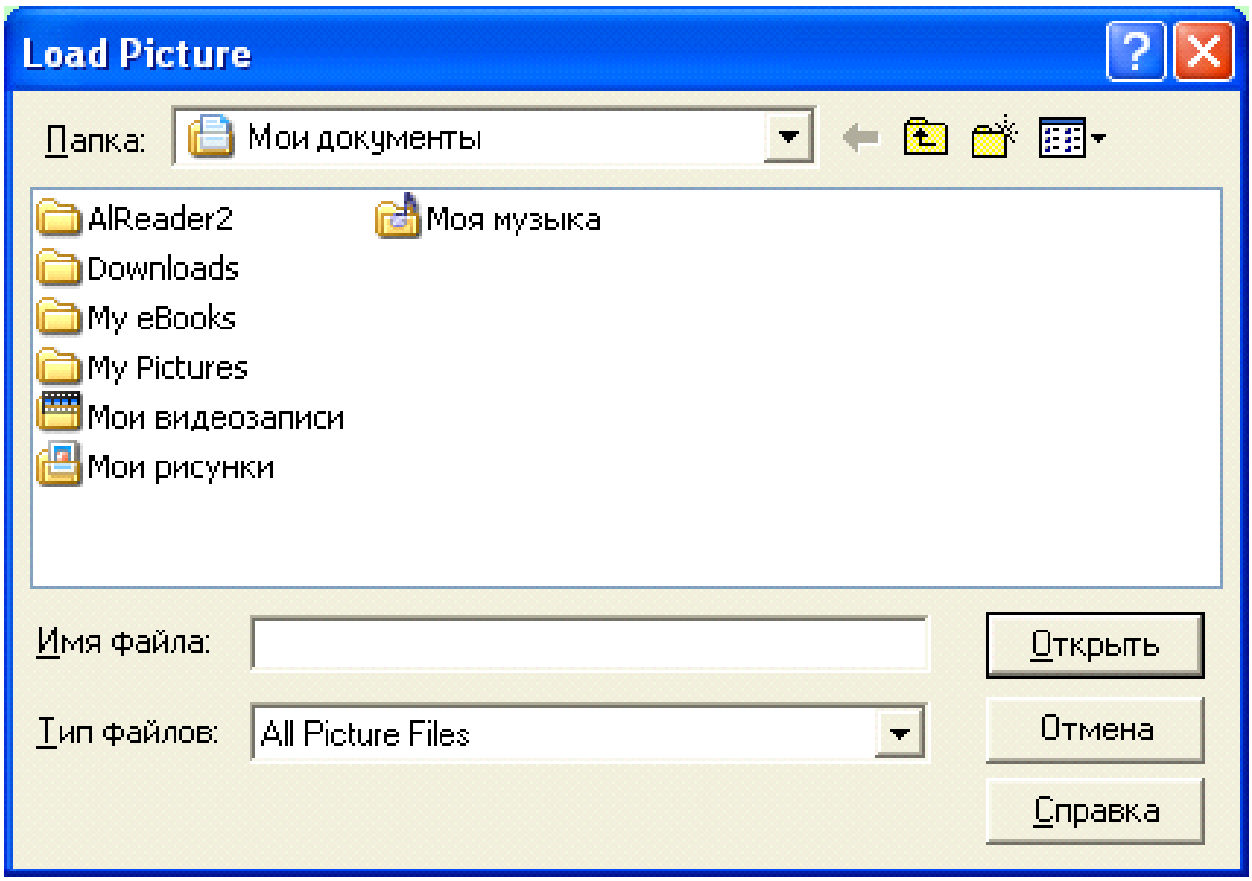


Рисунок 4.4. Диалоговое окно, используемое для загрузки графического файла, вставляемого в объект **Image**

Заготовка для объекта **Иллюстрация** (как и для других объектов) представляет собой кнопку, находящуюся в **Панели элементов** (**Toolbox**). Эта кнопка выглядит следующим образом:  . Если навести на нее указатель мыши, то появляется всплывающая подсказка – **Image**. Щелкнув мышью на основной экранной форме создаем объект **Image1** и методом протягивания увеличиваем его размеры.

Созданное на экранной форме окно объекта пока что является пустым. Для того, чтобы поместить в это окно какой-либо готовый рисунок, нужно в **окне свойств (Properties)** для объекта **Image1** найти свойство **Picture**. По умолчанию это свойство имеет значение **None** (ничего). Щелкнув мышью в поле справа от названия свойства **Picture**, видим, что

появляется кнопка-построитель (на этой кнопке изображено многоточие). Щелкнув по кнопке-построителю, открываем диалоговое окно **Load Picture**, используемое для загрузки рисунка (рис. 4.4).

По умолчанию данное окно для выбора рисунка открывает папку «**Мои документы**». Для того чтобы переместиться в папку с проектами, где находится заготовленный файл, нужно использовать поле **Папка,** находящемся в верхней части диалогового окна **Load Picture**. Щелкнув на раскрывающей стрелке, расположенной справа от этого поля, открываем структуру компьютерной системы (подобно тому, как это делается в программе **Проводник**) и находим необходимую нам папку с проектами. Выделив в папке необходимый графический файл (в данной случае – это файл **Coins.wmf**), щелкаем по экранной кнопке **Открыть**, расположенной в нижней части диалогового окна. В результате наших действий окно объекта **Image1** оказывается заполненным картинкой из файла.

При вставке картинки в окно **Image1** возможна такая ситуация, когда в окне мы видим не всю картинку, а только ее фрагмент. Причина заключается в том, что картинка больше по размеру, чем отведенное для нее место на форме в окне **Image1**. Из этой ситуации возможны два выхода: либо нужно увеличить размеры окна, чтобы в него помещалась вся картинка, либо подогнать размеры картинки к размерам уже имеющегося окна.

Второй вариант является более предпочтительным, так как при этом не нужно перемещать другие, ранее созданные элементы интерфейса или уменьшать их размеры. Для реализации этого варианта в окне свойств для объекта **Image1** находим свойство **PictureSizeMode** (режим, определяющий размеры картинки). По умолчанию в поле справа от названия свойства стоит значение «**0**», которое соответствует оригинальным размерам картинки. Если из раскрывающегося списка выбрать значение «**1**», то автоматически произойдет подгонка размеров картинки к размерам имеющегося окна, и картинка будет полностью видна в окне **Image1**.

На этом создание пользовательского интерфейса для проекта «**Вклад**» завершено. Внешний вид пользовательского интерфейса для данного проекта со всеми его элементами, включая иллюстрацию, показан на рис. 4.5.

Написание программного кода для данного проекта начнем с экранной кнопки **CommandButton1** (**Подсчитать итоговую сумму**). В начале описываем используемые в процедуре переменные: переменные **n** (срок действия договора) и **p** (процент по вкладу) описываем как переменные целого типа, переменные **Summa** (исходная сумма) и **Sum** (итоговая сумма по окончании срока действия договора) описываем как переменные вещественного типа двойной точности.

Значение переменной **Summa** берем из текстового окна **Textbox1** (с помощью функции **Val** преобразуем это значение из строковой формы в числовую), переменной **n** – из текстового окна **Textbox2**, а переменной **p** – из окна **Textbox3**.

Затем с помощью цикла с заранее заданным количеством повторений определяем значение итоговой суммы. По истечении каждого года действия договора к сумме вклада за предыдущий год добавляются проценты, начисленные на эту сумму, что находит отражение в формуле, записанной в теле цикла. Переменная цикла изменяется от 1 до **n**. В целом же цикл будет выглядеть следующим образом:

```
For i = 1 To n
Summa = Summa * (1 + p / 100)
Next i
```

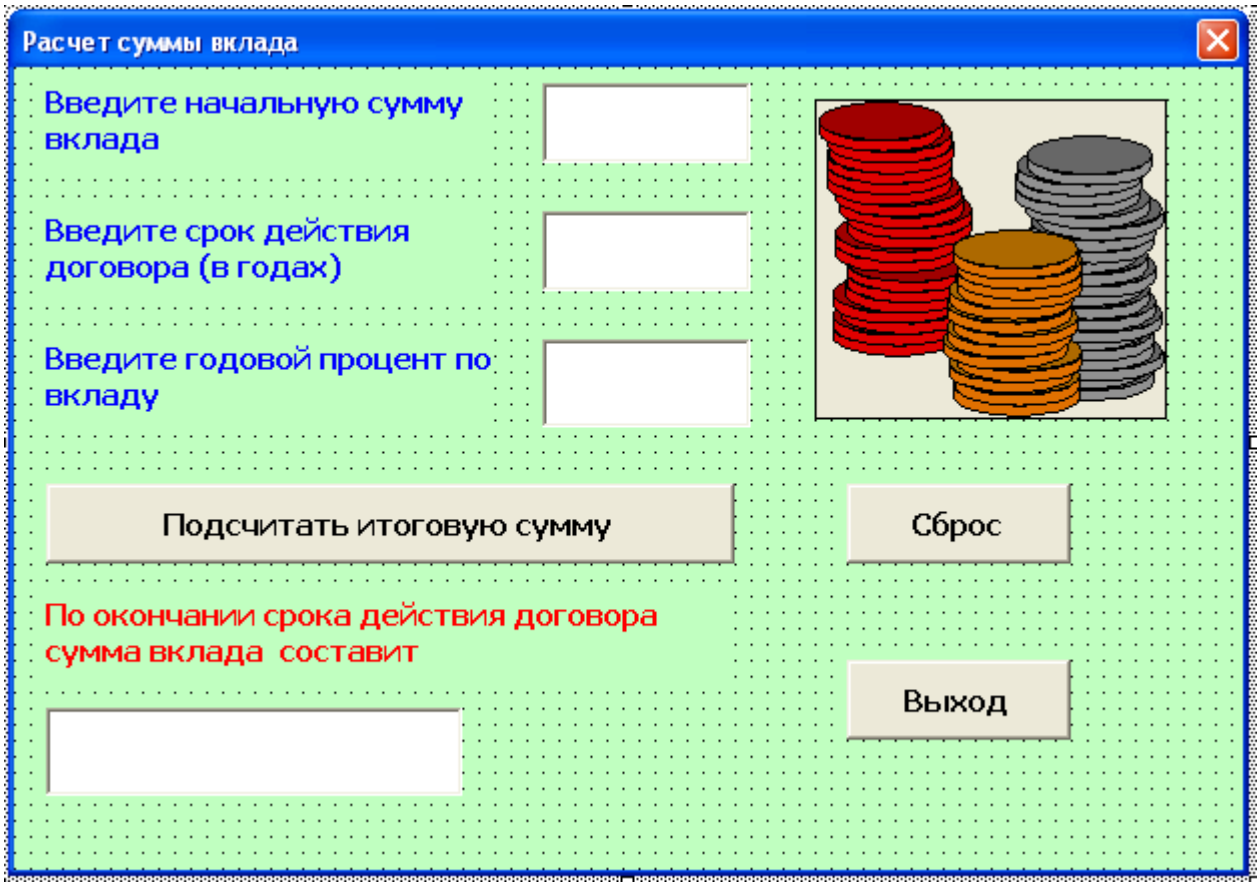


Рис. 4.5. Пользовательский интерфейс программы «**Вклад**»

После того, как итоговая сумма вычислена, ее нужно вывести в текстовом окне **Textbox4**. Но при выводе данных возникает следующая проблема: при расчете может получиться значение с большим количеством знаков после точки, а реально пользователю нужна информация о сумме, выраженной в рублях и копейках, т.е. после точки должно быть всего два разряда, соответствующие копейкам. Поэтому полученную величину **Summa** нужно преобразовать соответствующим образом.

Для подобного преобразования числовых данных в языке Visual Basic существуют специальные функции. К ним относится функция **FormatNumber**, которую мы используем в данном проекте. Общий вид данной функции следующий:

FormatNumber(Expression, NumDigitsAfterDecimal)

Здесь **Expression** – это выражение, которое подвергается преобразованию. В простейшем случае выражение может быть одной константой или одной переменной (как это имеет место в нашем проекте). **NumDigitsAfterDecimal** – количество знаков после десятичной точки (являющейся в информационных технологиях эквивалентом запятой), которое сохраняется в числе. Преобразованное таким образом числовое значение должно быть присвоено какой-либо переменной. В данном случае мы присваиваем это значение переменной **Sum**:

```
Sum = FormatNumber(Summa, 2)
```

Далее значение переменной **Sum** (итоговая сумма на счете вкладчика по окончании срока действия договора) преобразуется в строковую форму с помощью функции **Str** и выводится в текстовом окне **Textbox4**.

Программирование экранных кнопок **CommandButton2** (**Сброс**) и **CommandButton3 (Выход)** производится аналогично тому, как это делалось в предыдущих проектах. Ниже приводится полный программный код данного проекта:

```
Private Sub CommandButton1_Click()
Dim n, p As Integer
Dim Summa, Sum As Double
Summa = Val(TextBox1.Text)
n = Val(TextBox2.Text)
p = Val(TextBox3.Text)
For i = 1 To n
Summa = Summa * (1 + p / 100)
Next i
Sum = FormatNumber(Summa, 2)
TextBox4.Text = Str(Sum)
End Sub

Private Sub CommandButton2_Click()
TextBox1.Text = ""
TextBox2.Text = ""
TextBox3.Text = ""
TextBox4.Text = ""
End Sub

Private Sub CommandButton3_Click()
End
End Sub
```

На рис. 4.6 показана программа «**Вклад**», находящаяся в рабочем режиме.

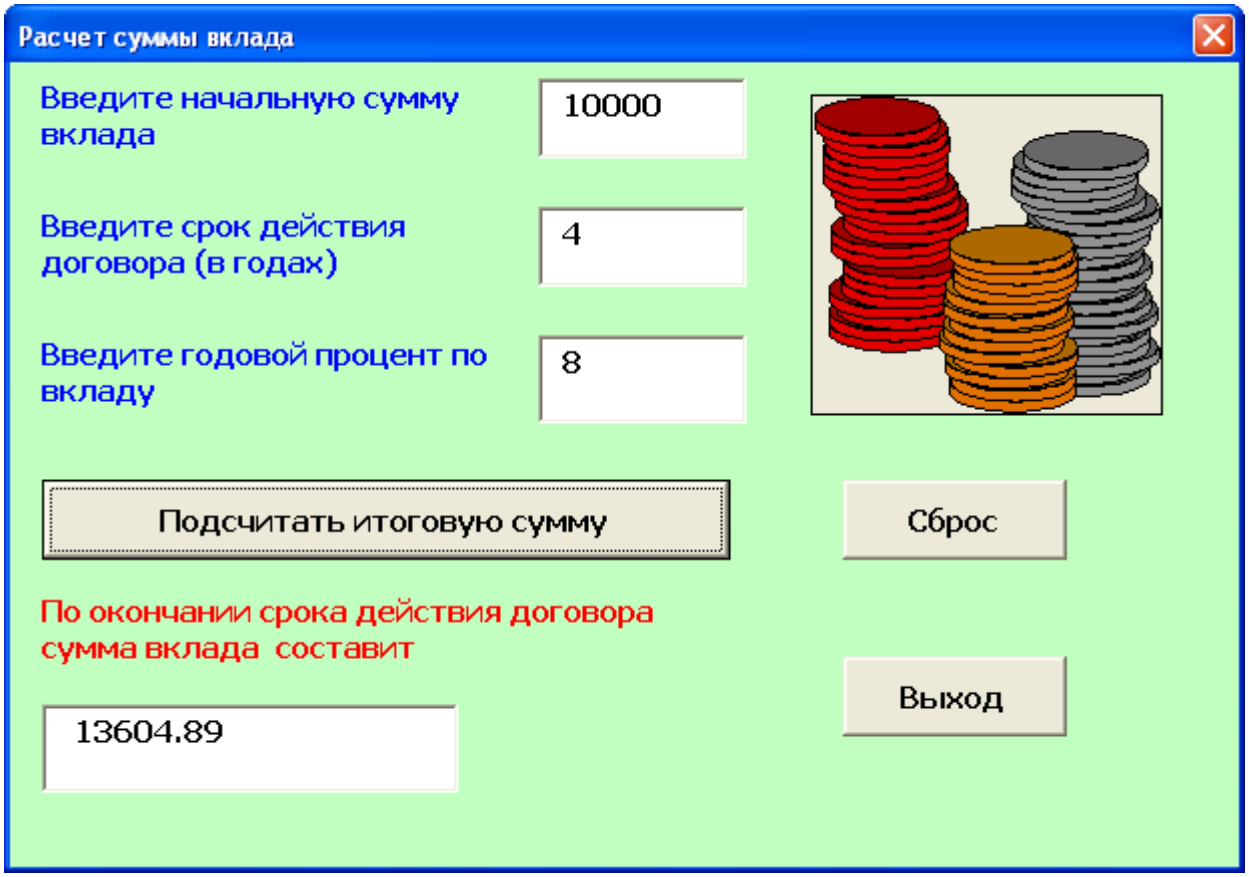


Рис. 4.6. Программа «Вклад» в процессе выполнения

После тестирования приложения при различных исходных данных работу над данным проектом можно завершить.

Вопросы для самоконтроля к главе 4

1. В каких случаях возникает необходимость использования в языках программирования циклических структур?
2. Как в общем виде записывается в Visual Basic цикл с заранее известным количеством повторений?
3. Что такое тело цикла?
4. Опишите механизм работы цикла с заранее известным количеством повторений.
5. Как определить количество повторений цикла?
6. Каким образом в Visual Basic можно задать шаг изменения переменной цикла?
7. Как изменить запись цикла таким образом, чтобы в процессе выполнения программы происходило уменьшение значения переменной цикла?
8. Каким образом можно добавить иллюстрацию в пользовательский интерфейс приложения, созданного в системе VBA?
9. Как можно произвести автоматическую подгонку размеров изображения к размерам уже имеющегося на пользовательской форме окна?
10. Как можно в системе VBA определить формат выводимого числового значения?

Итоговый тест

Для повторения и закрепления материала, изложенного в данном учебном пособии, читателю предлагается пройти следующий тест, содержащий вопросы по всем разделам пособия.

В каждом вопросе тестируемому предлагается 3 или 4 варианта ответа, причем правильным всегда является только один из предложенных на выбор вариантов.

В том случае, если тестируемый затрудняется с ответом на какой-либо из поставленных вопросов, необходимо вернуться к основному тексту пособия и вновь внимательно проработать соответствующий раздел пособия, для того чтобы найти правильный ответ.

1. В режиме записи макроса рядом с указателем мыши появляется следующее изображение:
а) песочные часы;
б) вопросительный знак;
в) магнитофонная кассета;
г) восклицательный знак.

2. В режиме записи макроса в строке состояния активизируется индикатор
а) ЗАП;
б) ИСПР;
в) ВДЛ;
г) ЗАМ.

3. Включить режим выделения при записи макроса можно нажатием функциональной клавиши
а) F6;
б) F7;
в) F8;
г) F9.

4. Для того чтобы пользователь самостоятельно мог решать вопрос о запуске макросов при открытии документа, в меню **Сервис** нужно установить следующий уровень безопасности:
а) высокий;
б) средний;
в) низкий.

5. Для того чтобы в системе VBA вставить в проект новую пользовательскую форму, нужно использовать команды
а) Insert → File;
б) Insert → Module;
в) Insert → Procedure;
г) Insert → UserForm.

6. Фоновый цвет объекта в VBA определяется свойством, которое называется
а) BackColor;
б) ForeColor;
в) TextColor;
г) TextBackground.

7. Цвет текста для объекта в VBA определяется свойством, которое называется
а) BackColor;
б) ForeColor;
в) TextColor;
г) TextBackground.

8. Основные характеристики шрифта объекта в VBA (размер шрифта, гарнитура, начертание) определяются свойством, которое называется
а) Caption;
б) Font;
в) Text.

9. Если после названия команды в меню системы VBA стоит многоточие, то данная команда
а) открывает вложенное подменю;
б) открывает диалоговое окно;
в) является активной командой.

10. В каких единицах измеряются геометрические размеры объекта (длина и ширина) в окне свойств VBA
а) в дюймах;
б) в пикселях;
в) в сантиметрах;
г) в пунктах.

11. Ряд объектов в системе VBA обладает свойством Visible, которое может принимать следующее количество различных значений:
а) 2
б) 3
в) 4
г) 5

12. Имя объекта и имя его свойства в языке Visual Basic отделяются друг от друга
а) двоеточием;
б) запятой;
в) точкой;
г) точкой с запятой.

13. В системе VBA можно создать диалоговое окно, содержащее какой-либо текст и закрывающую кнопку «Ok», с помощью оператора
а) Msgbox;
б) Textbox;
в) Toolbox.

14. Какое из указанных имен переменных является правильным с точки зрения языка Visual Basic
а) 1y;
б) сумма;
в) Sum2.

15. Какое служебное слово открывает раздел описания переменных в Visual Basic

а) Dim;
б) Var;
в) Variable.

16. Какой тип используется в Visual Basic для описания больших целых чисел
а) Double;
б) Long;
в) Single.

17. Какая функция применяется в Visual Basic для преобразования текстовой величины в числовую форму
а) Str;
б) StrtoInt;
в) InttoStr;
г) Val.

18. Какая функция применяется в Visual Basic для преобразования числовой величины в текстовую форму:
а) Str;
б) StrtoInt;
в) InttoStr;
г) Val.

19. Операция целочисленного деления обозначается в языке Visual Basic следующим символом:
а) /
б) \
в) |
г) :

20. Блочная форма записи условного оператора в языке Visual Basic завершается следующей командой:
а) End;
б) End If;
в) Exit;
г) Else If.

21. В сокращенной форме записи условного оператора отсутствует служебное слово
а) If;
б) Then;
в) Else.

22. Запись оператора множественного выбора в Visual Basic начинается со служебных слов

а) Case of;
б) Select Case;
в) Case Else.

23. Переменная цикла должна относиться к следующему типу:
а) вещественному
б) целому;
в) строковому;
г) логическому.

24. Оператор цикла с заранее известным числом повторений в Visual Basic завершается командой
а) Goto I;
б) Next I;
в) Step I.

25. Для того чтобы определить количество цифр после точки в выводимом числе, следует использовать функцию
а) Val;
б) FormatNumber;
в) Str.

Библиографический список

Андерсон Т. Visual Basic. — М.: ЗАО «Издательство Бином», 1998.

Биллиг В.А. VBA в Office 2000. Офисное программирование — М.: Русская редакция, 1999.

Гарнаев А.Ю. Самоучитель VBA. — СПб.: БХВ–Петербург, 1999.

Гольденберг В.А. Введение в программирование. — Минск: ООО «Харвест», 1997.

Джекобсон Р. Microsoft Office 2000. Автоматизация и Интернет-возможности. — М.: Русская редакция, 2000.

Информатика. Базовый курс /Симонович С.В. и др. — СПб.: Питер, 2001.

Комягин В.Б. Программирование в Excel 5 и Excel 7 на языке Visual Basic. — М.: Радио и связь, 1996.

Король В.И. Visual Basic 6.0, Visual Basic For Applications 6.0. Язык программирования. — М.: Кудиц, 2000.

Кузьменко В.Г. Программирование на VBA 2002. — М.: ООО «Бином-Пресс», 2003.

Культин. Н.Б. Макрокоманды MS Word. — СПб.: БХВ-Петербург, 1999.

Пестриков В.М., Маслобоев А.Н. Основы программирования в системе Borland Delphi: учебное пособие/ СПбГТУ РП. — СПб., 2004.

Пестриков В.М., Маслобоев А.Н. Delphi на примерах. — СПб.: БХВ-Петербург, 2005.

Санна. П. Visual Basic 5 для приложений (VBA) в подлиннике. — СПб.: БХВ-Петербург, 1997.

Сдвижков О.А. Excel-VBA. Словарь-справочник пользователя. — М.: Эксмо, 2008.

Слепцова Л.Д. Программирование на VBA в Microsoft Office 2007. — М.: Диалектика, 2007.

Соломон К. Microsoft Office 97: разработка приложений. — СПб.: БХВ-Петербург, 1998.

Учебное издание

Виктор Михайлович Пестриков
Артур Николаевич Маслобоев

Основы программирования в Microsoft Word на Visual Basic for Applications

Учебное пособие

Редактор и корректор Н.П. Новикова
Техн. редактор Л.Я. Титова Темплан 2010 г., поз. 104

Подп. к печати 25.10.2010. Формат 60x84/16. Бумага тип. № 1.
Печать офсетная. 5,5 уч.-изд. л. ; 5,5 усл. печ. л. Тираж 100 экз.
Изд. № 104. Цена «С». Заказ

Ризограф ГОУВПО Санкт-Петербургского государственного технологического университета растительных полимеров
198095, СПб., ул. Ивана Черных, 4.