Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ»

Т.Г. Сакова, О.В. Юдина

Разработка бизнес-приложений в среде MS Excel средствами VBA

Учебное пособие

Самара
Издательство
Самарского государственного экономического университета
2022

УДК 004.43
ББК З 973.2я7
С 15

Рецензенты:
кафедра цифровых технологий в образовании
Самарского государственного университета путей сообщения
(зав. кафедрой кандидат педагогических наук *С.В. Горбатов*);
И.В. Тюжина, кандидат педагогических наук, доцент кафедры
цифровых технологий в образовании Самарского государственного
университета путей сообщения

Издается по решению
учебно-методического совета университета

Сакова, Татьяна Германовна.

С 15 Разработка бизнес-приложений в среде MS Excel средствами VBA : учебное пособие / Т.Г. Сакова, О.В. Юдина ; Самарский государственный экономический университет. – Самара : Изд-во СГЭУ, 2022. – 1 CD-R. – Систем. требования: Intel 1,3 GHz и более ; 256 Мб (RAM) ; MS Windows 10 ; Adobe Reader ; дисковод CD-ROM. – Загл. с титул. экрана.

ISBN 978-5-00176-124-2

В учебном пособии, разработанном по дисциплинам «Пакеты офисных программ», «Современные технологии и языки программирования», «Встроенные языки программирования», рассматриваются вопросы, связанные с теорией по теме основ алгоритмизации и разработки бизнес-приложений, теоретическими и практическими аспектами использования встроенного языка программирования VBA в пакете MS Excel для экономических расчетов. Подробно освещаются вопросы разработки бизнес-приложений с использованием встроенных элементов управления. Пособие содержит контрольные вопросы и задания для выполнения лабораторных работ.

Для студентов направления подготовки 09.03.03 «Прикладная информатика».

УДК 004.43
ББК З 973.2я7

ISBN 978-5-00176-124-2

Оглавление

Введение

На сегодняшний день возрастает потребность в применении эффективных и адекватных реальной действительности цифровых технологий. Произошло смещение акцентов в формулировании критериев эффективности систем управления, поскольку понятие экономичности решения вымещается вопросами быстроты его принятия, степенью адекватности аналитических решений и возможности использования прогнозных моделей.

Microsoft Excel – это очень мощный инструмент, который можно использовать для управления данными, их анализа и представления. Он получил широкое распространение в следующих областях:

- бухгалтерский и банковский учет;
- планирование и распределение ресурсов;
- проектно-сметные работы, инженерно-технические расчеты;
- обработка больших массивов информации;
- исследование динамических процессов;
- сфера бизнеса и предпринимательства.

Microsoft Excel содержит мощные математические и инженерные функции, которые позволяют решать множество задач в области естественных и технических наук. Электронные таблицы широко применяются для автоматизации вычислений. Возможность использования математических формул в электронных таблицах позволяет представить взаимосвязи между разными параметрами.

Иногда возникает необходимость решить какую-то задачу или выполнить повторяющееся действие, которые не удается реализовать с помощью интерфейса пользователя Excel, несмотря на достаточно обеспеченный набор его возможностей. Во все приложения Office интегрирован язык программирования VBA (Visual Basic for Applications),

позволяющий расширять возможности этих приложений. Отличительной особенностью VBA является использование обычных переменных и констант с имеющимися объектами приложений Microsoft Office. Например, в Microsoft Office Excel это могут быть рабочие книги, рабочие листы, диапазоны ячеек, диаграммы и т.д. С помощью VBA можно разрабатывать бизнес-приложения различной сложности.

VBA – простой в освоении язык программирования, который позволяет быстро получать результаты. В пособии рассмотрены основные режимы разработки программы (создание макросов и основные приемы работы в редакторе Visual Basic Editor), операторы языка, применение пользовательских форм в MS Excel и описан синтаксис языка программирования VBA.

1 Объектно-ориентированный язык программирования VBA

1.1. Расширение возможностей приложений MS Office средствами VBA

Часто возникают проблемы, требующие больших затрат времени и внимания. Например, перенести список контактов из Microsoft Outlook в таблицу Microsoft Excel. Для решения подобных задач удобнее всего использовать Visual Basic для приложений (VBA) для Office. Это простой и в то же время мощный язык программирования, применение которого позволит расширить возможности приложений Office.

Набор приложений Office обладает большим количеством возможностей. Существует множество разных способов создания, форматирования и управления документами, электронной почтой, базами данных, формами, электронными таблицами и презентациями. Значительное преимущество программирования на VBA в Office заключается в том, что почти каждое действие, осуществляемое с мышью, клавиатурой или диалоговым окном, можно выполнить с помощью VBA, а значит, в дальнейшем его легко осуществить сотни раз (автоматизация повторяющихся задач – одно из наиболее частых применений VBA в Office).

Помимо возможности написания сценария VBA для ускорения повседневных задач, VBA можно использовать для добавления новых функций в приложения Office или создания запросов и взаимодействия с пользователем ваших документов в соответствии с потребностями организации. Например, можно написать код VBA, выводящий всплывающее сообщение, которое напоминает пользователям о необходимости сохранения документа на определенном сетевом диске при первой попытке его сохранения.

Есть несколько основных причин использовать программирование на VBA в Office.

Автоматизация и повторяемость. Программирование на VBA эффективно при создании решений для устранения повторяющихся проблем форматирования и исправлений.

Расширение возможностей взаимодействия с пользователем. Иногда некоторые действия пользователей с документом или приложением Office обязательны или желательны, но недоступны в стандартном приложении. Например, нужно уведомлять пользователей о необходимости что-то сделать при открытии, сохранении или печати документа.

Взаимодействие приложений Office. Нужно скопировать все контакты из Outlook в Word и отформатировать их определенным способом? Или же нужно переместить данные из Excel в набор слайдов PowerPoint? Иногда простое копирование и вставка не работают так, как надо, или же это происходит слишком медленно.

1.2. Возможности языка

Язык Visual Basic for Application (VBA) – объектно-ориентированный язык программирования, базируется на командах и синтаксисе языка Visual Basic. VBA встроен в офисную среду и позволяет манипулировать объектами всех офисных приложений. Язык VBA дает возможность сочетать простые методы по созданию документов (использование команд меню или технологии перетаскивания мышью) и программные методы для разработки эффективного пользовательского приложения. Чаще всего основным документом, на базе которого пользователь разрабатывает свое приложение, является MS Excel.

Язык VBA поддерживает объектную модель, т.е. разработчики организуют объекты программирования в виде иерархии. В Excel, например, есть объект верхнего уровня Application, который содержит объект Worksheet. Объект Worksheet содержит объекты Cells и т.д. В объектных моделях приблизительно отражено то, что вы видите в пользовательском интерфейсе. Они являются концептуальной картой приложения и его возможностей.

Определение объекта называется классом. С технической точки зрения класс – это описание или шаблон, используемый для формирования или создания экземпляра объекта.

Уже существующим объектом можно управлять, задавая его свойства и вызывая его методы. Если представить объект в виде имени существительного, свойства станут прилагательными, описывающими существительное, а методы – глаголами, которые приводят его в действие. Изменение свойства приводит к модификации определенной характеристики внешнего вида или поведения объекта. Вызов одного из методов объекта заставляет его выполнить какое-либо действие.

Редактор Visual Basic. Код программы пишется в редакторе Visual Basic. Он установлен по умолчанию в любом приложении Office и находится на вкладке **Разработчик**. Так как в Office вкладка **Разработчик** не показана по умолчанию, необходимо вывести ее на экран, выполнив указанные ниже действия.

На вкладке **Файл** нажмите кнопку **Параметры**, в открывшемся диалоговом окне **Параметры** выберите категорию **Настройка ленты** в левой части диалогового окна. В разделе **Выбрать команды**, расположенном слева в окне, выберите **Часто используемые команды**. Справа в группе **Настройка ленты** в раскрывающемся списке выберите **Основные вкладки**, а затем установите флажок **Разработчик**. Нажмите кнопку **OK**.

После включения вкладки **Разработчик** можно легко найти кнопки **Visual Basic** и **Макрос** (рис. 1.1).

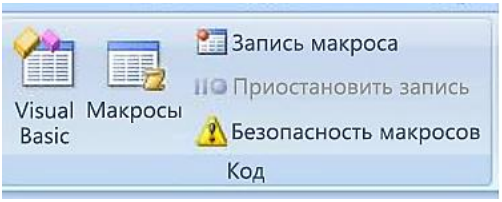


Рис. 1.1. Кнопки на вкладке **Разработчик**

Редактор Visual Basic Editor (VBE) – это среда разработки новых и редактирования существующих программ (макросов) и процедур языка Visual Basic для приложений. VBE включает в себя полный набор средств отладки, обеспечивающих обнаружение ошибок синтаксиса, ошибок выполнения и логических ошибок в программах, представляет собой отдельное приложение, запускающееся только в программах

MS Office. Модули и формы VBA, т.е. место, где хранится код на языке VBA, сохраняются в файлах MS Office.

Ошибки и отладка. Существует два основных вида ошибок программирования: синтаксические ошибки, которые нарушают грамматические правила языка программирования, и ошибки времени выполнения, которые синтаксически правильны, но вызывают сбой, когда VBA пытается выполнить код.

Синтаксические ошибки легко обнаружить. Редактор Visual Basic выдает сопровождающееся звуковым сигналом сообщение об ошибке и меняет цвет текста, если при вводе кода найдена синтаксическая ошибка.

Ошибки времени выполнения обнаружить сложнее, так как синтаксически все выглядит правильно, но при попытке выполнить код возникает сбой.

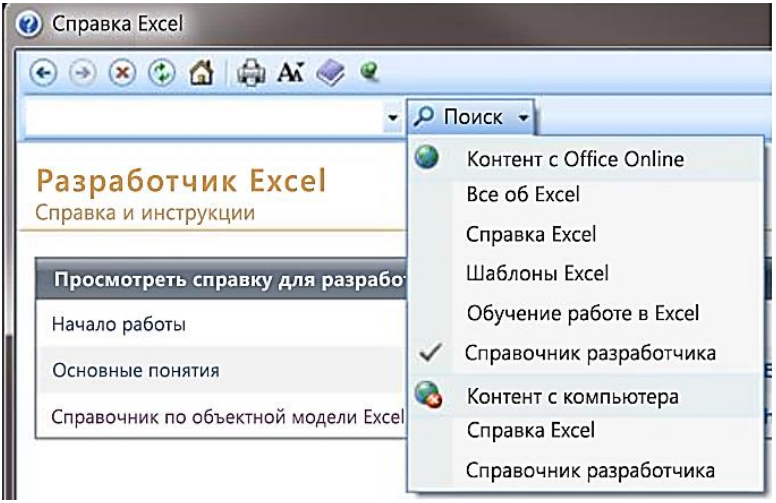


Рис. 1.2. Фильтрация справки разработчика (работает во всех приложениях Office)

Правильное использование справочных материалов. Чтобы открыть справочник разработчика (см. рис. 1.2), встроенный в справку Office, откройте справку в любом приложении Office, выбрав вопросительный знак на ленте или нажав клавишу **F1**. Затем справа от кнопки **Поиск** выберите стрелку раскрывающегося меню для фильтрации

содержимого. Выберите **Справочник разработчика**. Если на левой панели не отображается содержание, щелкните маленький значок книги, чтобы открыть его, и разверните справочник по объектной модели.

Контрольные вопросы

1. Что представляет собой редактор Visual Basic Editor?
2. Где сохраняются программы, написанные на VBA?
3. В чем суть объектно-ориентированного программирования?
4. Что такое методы объекта?
5. Что такое свойства объекта?

2 Макросы

2.1 Понятие макроса, назначение макросов

Работая в Excel, приходится повторять одни и те же действия и процедуры по обработке данных. Использование макросов позволяет автоматизировать эти операции.

Макрос – это запрограммированная последовательность действий (программа, процедура), записанная на языке программирования Visual Basic for Applications (VBA). Можно запускать макрос необходимое количество раз, автоматически выполняя последовательность любых действий.

Что можно сделать с помощью VBA?

Вставить строку текста или формулу. Например, если нужно часто вводить в рабочие таблицы название своей фирмы.

Автоматизировать часто выполняемую процедуру. Если нужно подготовить месячный отчет и его структура не слишком сложна, то вы можете написать макрос.

Автоматизировать повторяющиеся операции. Если нужно выполнить некоторое действие в 12 различных рабочих книгах, можно записать макрос при первом выполнении этой операции, а затем позволить ему повторить указанное действие в остальных рабочих книгах.

Создать новую команду. Например, можно объединить несколько команд из меню Excel, чтобы можно было выполнить их с помощью только одной комбинации клавиш или одного щелчка мыши.

Создать новую кнопку на панели инструментов для запуска написанных макросов.

2.2. Создание макросов макрорекордером

Макрорекордер – это небольшая программа, встроенная в Excel, которая переводит любое действие пользователя на язык программирования VBA и записывает получившуюся команду в программный модуль.

Такой способ создания макросов не требует от пользователя знаний программирования в VBA и позволяет применять макросы. Вместе с тем такой способ имеет свои достоинства и недостатки:

- макрорекордер записывает действия, которые выполняются во время работы Microsoft Excel;
- макрорекордер может записать только те действия, которые можно выполнить с помощью команд Excel;
- макрорекордером записываются все действия пользователя, включая ошибочные. Режим редактирования макроса позволяет откорректировать или удалить неверные команды.

Чтобы записать макрос с помощью макрорекордера, необходимо:

1. Выбрать вкладку **Вид**.
2. Выполнить команду **Макросы** – **Запись макроса** (рис. 2.1).

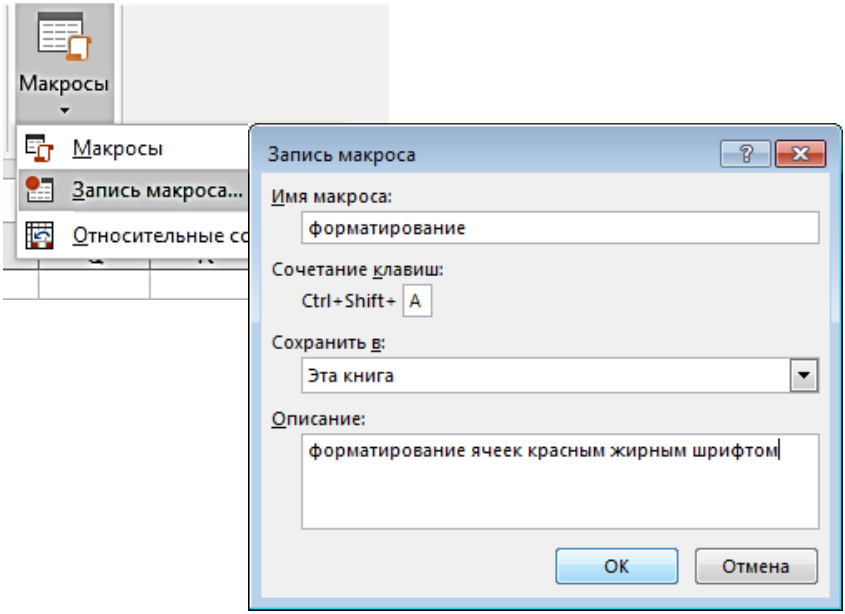


Рис. 2.1. Окно создания макроса

В появившемся диалоговом окне следует задать:

1. Имя макроса – смысловое имя на русском или английском языке. Имя должно начинаться с буквы и не должно содержать пробелы и знаки препинания.

2. Сочетание клавиш – для быстрого запуска макроса.

3. Сохранить в – здесь задается место, куда будет сохранен текст макроса, т.е. набор команд на VBA, из которых и состоит макрос:

- *Эта книга* – макрос сохраняется в модуль текущей книги и, как следствие, будет выполняться, только пока эта книга открыта в Excel;
- *Новая книга* – макрос сохраняется в шаблон, на основе которого создается любая новая пустая книга в Excel, т.е. макрос будет содержаться во всех новых книгах, создаваемых на данном компьютере, начиная с текущего момента;
- *Личная книга макросов* – это специальная книга Excel с именем Personal.xls, которая используется как хранилище макросов. Все макросы из Personal.xls загружаются в память при старте Excel и могут быть запущены в любой момент и в любой книге.

4. Описание – краткое понятное описание действий макроса.

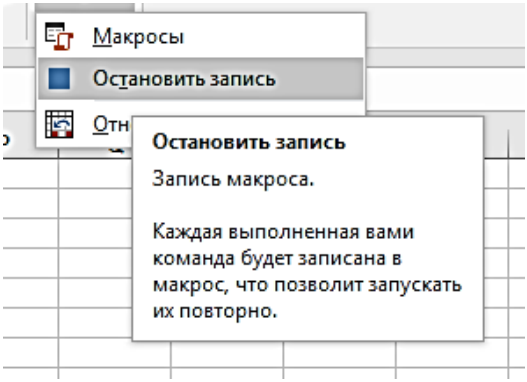


Рис. 2.2. Команда **Остановить запись**

После включения записи и выполнения действий, которые необходимо записать, запись следует остановить командой **Остановить запись** (рис. 2.2).

2.3. Виды записи: абсолютная и относительная

Обычно при записи макроса Excel сохраняет точные адреса ячеек, которые вы выбираете (т.е. выполняет **абсолютную запись**). Например, если при записи макроса вы выбираете диапазон B1: B10, то Excel запишет это следующим образом:

```
Range(«B1:B10»).Select
```

При вызове данного макроса всегда будут выделяться именно указанные ячейки, **независимо от расположения текущей ячейки**.

Когда Excel работает в режиме относительной записи, кнопка **Относительная ссылка** отображена **нажатой**. Для возврата в режим абсолютной записи достаточно снова щелкнуть на эту кнопку, и она примет вид своего нормального (отжатого) состояния.

Например, если при записи макроса в относительном режиме активной является ячейка A1, то операция выбора диапазона ячеек B10 приведет к записи следующего оператора:

```
ActiveCell.offset(0,1) . Range («A1 :A10») .Select
```

Этот оператор можно расшифровать так: от активной ячейки нужно переместиться на 0 строк вниз и на 1 столбец вправо и считать, что это ячейка A1. Относительно нового положения выбрать диапазон A1:A10. Другими словами, макрос, записанный в относительном режиме, в качестве отправной точки использует активную ячейку, а затем выбирает диапазон относительно этой ячейки. Таким образом, в зависимости от расположения активной ячейки вы будете получать различные результаты.

2.4. Редактирование и удаление макросов

Пример. Запишите макрос, который форматирует выделенные ячейки жирным красным цветом 13 размером шрифтом Arial.

Макрос был записан в новом модуле. Чтобы просмотреть текст макроса в этом модуле, необходимо активизировать средство Редактор Visual Basic одним из двух способов:

1. Нажать комбинацию клавиш **ALT+F11**.

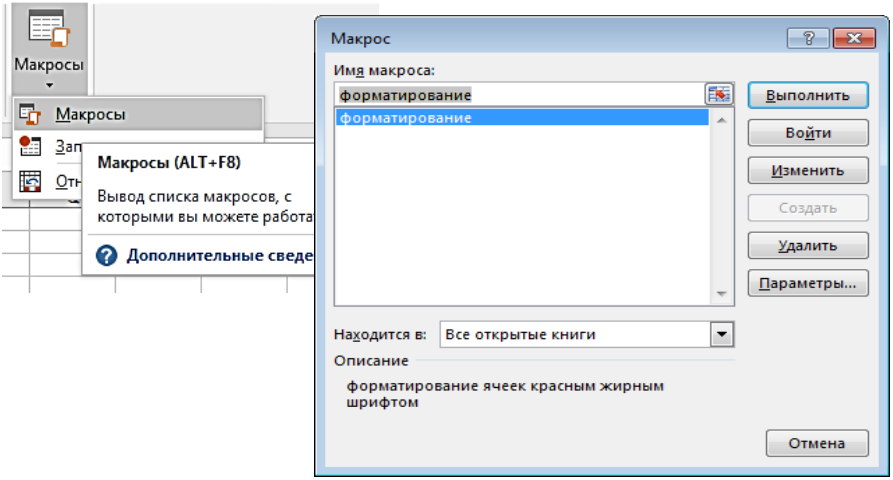


Рис. 2.3. Окно редактирования макросов

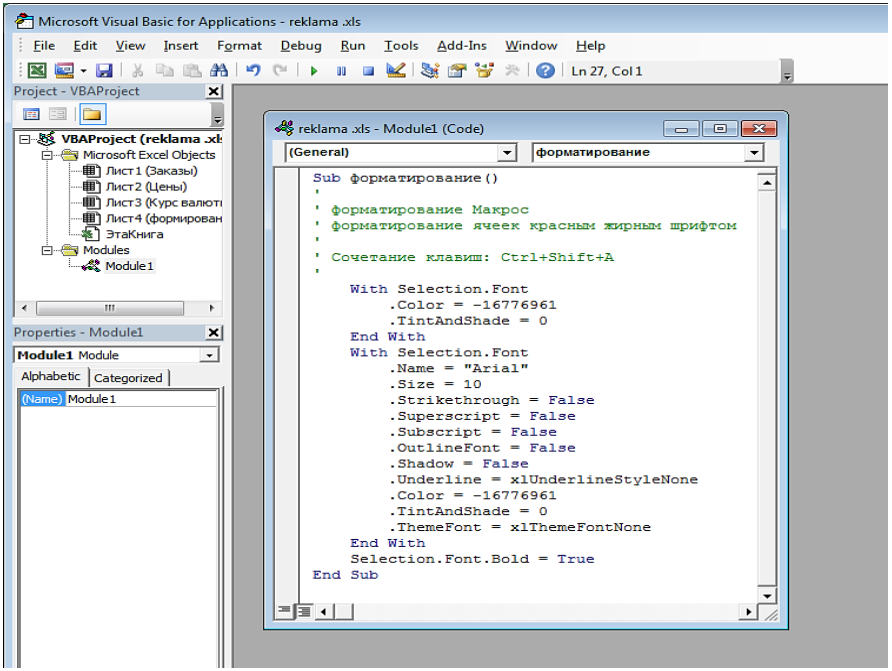


Рис. 2.4. Окно листинга исходного кода программы

2. Выполнить команду вкладки **Вид – Макросы – Макросы**, в открывшемся диалоговом окне выбрать кнопку **Изменить** или **Удалить** в соответствии с поставленной задачей (см. рис. 2.3).

В окне проекта отображен список всех открытых рабочих книг и надстроек. Этот список имеет вид древовидной диаграммы. Текст макроса хранится в отдельном модуле в текущей рабочей книге (см. рис. 2.4).

Не зная команд VBA, пользователь может изменить содержимое макроса, например, задать новый размер шрифта или его название.

2.5. Запуск макросов

Необходимо вызвать диалоговое окно, выполнив команду вкладки **Вид – Макросы – Макросы**, и выбрать кнопку **Выполнить** (см. рис. 2.3).

Нажать сочетание клавиш, закрепленное за макросом.

Добавить кнопку на панель быстрого доступа. Для этого выполните команду **Файл – Параметры**, в появившемся диалоговом окне (рис. 2.5) выделите последовательно строку **Панель быстрого доступа**, затем категорию **Макросы** и при помощи кнопки **Добавить** перенесите выбранную кнопку на панель быстрого доступа.

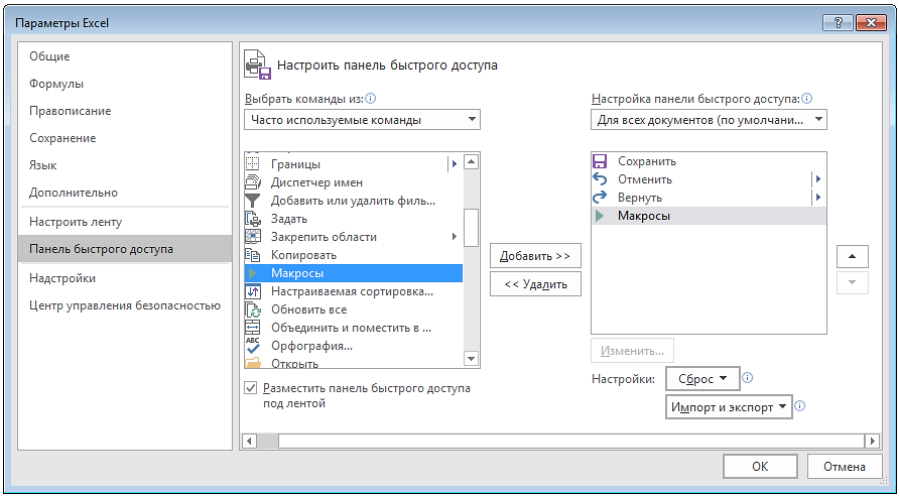


Рис. 2.5. Окна настройки панели быстрого доступа

Добавить кнопку на рабочем листе. Этот способ подходит для любой версии Excel. Мы добавим кнопку запуска макроса прямо на рабочий лист как графический объект. Для этого:

1. Выполните команду **Файл – Параметры.**

2. Выделите строку **Панель быстрого доступа.**

3. В списке **Выбрать команды из** установите **Вкладка «Разработчик»**.

4. Добавьте на **Панель быстрого доступа** панель **Элементы управления формы** (рис. 2.6).

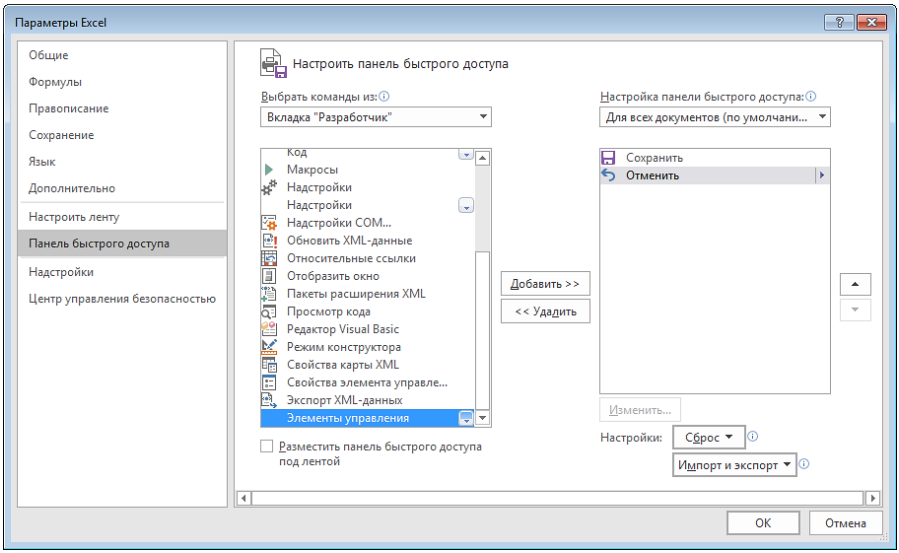


Рис. 2.6. Подключение элементов управления

Затем нарисуйте кнопку на листе, удерживая левую кнопку мыши. Автоматически появится окно, где нужно выбрать макрос, который должен запускаться при щелчке по нарисованной кнопке (рис. 2.7).

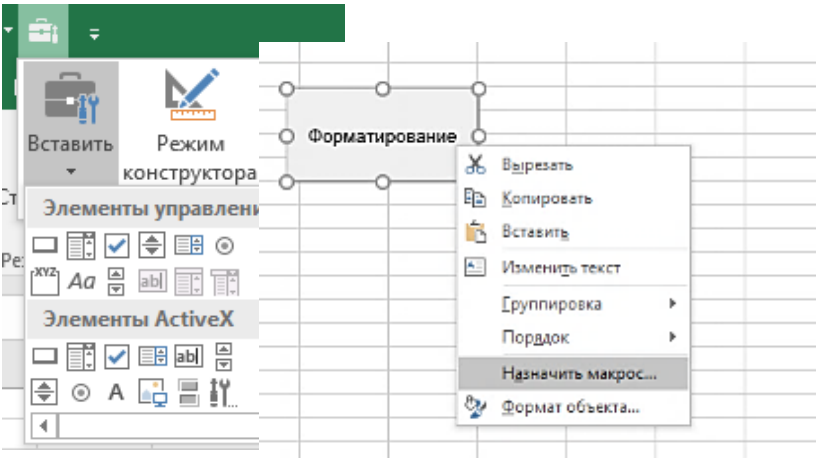


Рис. 2.7. Назначение макроса для кнопки **Форматирование**

Контрольные вопросы

1. Каково основное назначение макросов?
2. Можно ли редактировать созданный в макрорекордере макрос?
3. Где хранится созданный макрос?
4. В каких случаях необходимо включать режим относительной записи?
5. Сколько способов запуска макросов вы знаете?

3 Алгоритмизация и технологии программирования

3.1. Понятие и свойства алгоритма

Понятие алгоритма так же фундаментально для информатики, как и понятие информации. В настоящее время понятие алгоритма не ограничивается только арифметическими вычислениями, а понимается гораздо шире.

Алгоритм – точное предписание, определяющее процесс перехода от исходных данных к результату, требуемому пользователю.

Основные требования к алгоритму:

1. **Точность** – каждая команда алгоритма должна определять однозначное действие исполнителя.
2. **Понятность** – алгоритм для исполнителя должен включать в себя только те команды, которые входят в его систему команд.
3. **Конечность (результативность)** – исполнение алгоритма должно завершиться за конечное число шагов.

При выполнении этих требований к алгоритму исполнителем он будет выполняться *формально*, т.е. без элементов творчества.

Предписание считается алгоритмом, если оно обладает тремя следующими свойствами:

- определенностью, т.е. точностью, не оставляющей места для произвола;
- универсальностью, т.е. возможностью исходить из меняющихся в известных пределах значений исходных данных;
- результативностью, т.е. направленностью на получение результата.

Программа – это алгоритм, записанный на языке исполнителя.

Задача программиста состоит в определении последовательности действий, которые необходимо выполнить, чтобы достигнуть нужного результата.

3.2. Способы записи алгоритмов

Можно выделить три основных способа записи алгоритмов: словесный, формульно-словесный и графический.

Словесный алгоритм – это алгоритм, записанный на естественном языке. Недостаток такого описания – слабая наглядность и формализуемость.

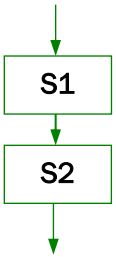
Формульно-словесный алгоритм – это алгоритм, в котором вычисления записаны с помощью математических формул, а логические переходы – на естественном языке. Достоинство такого описания – любая степень детализации, недостаток – слабая наглядность.

Графический способ описания алгоритма предполагает, что каждому действию соответствует определенного вида геометрическая фигура. Этот способ является наиболее наглядным и формализуемым, называется записью алгоритма с помощью блок-схем.

3.3. Типы алгоритмических конструкций

При записи любого алгоритма используются следующие стандартные структуры.

- **Линейный алгоритм:**

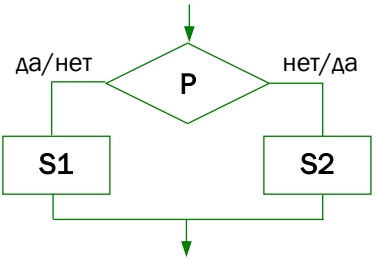


Следование. Указывает, что управление передается последовательно от одного процесса к другому.

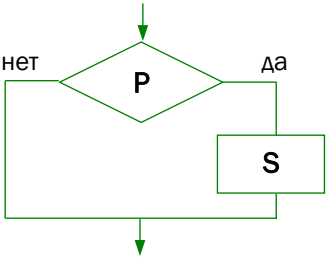
- **Разветвляющиеся алгоритмы:**

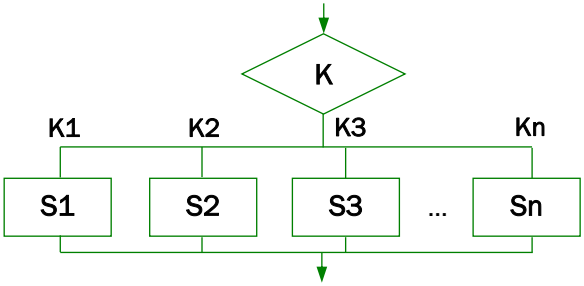
Развилка полная. Используется в случае, когда выполнение программы может пойти двумя различными (альтернативными) путями. P – логическое условие, по которому осуществляется выбор требуемого направления выполнения алгоритма.

В зависимости от значения логического условия (да/нет) дальнейшее выполнение алгоритма идет либо по левой (S1), либо по правой (S2) ветви. S1, S2 обозначают унифицированные структуры, процедуры, функции и алгоритмы любой сложности.



Развилка неполная. Используется так же, как и развилка полная, с тем отличием, что при выполнении одной из ветвей никаких изменений данных, поступивших на вход этой унифицированной структуры, не происходит.

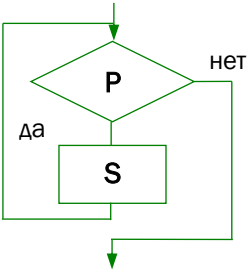




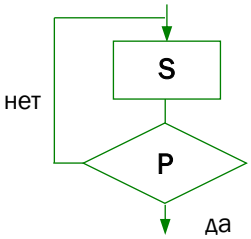
Выбор. Предназначен для выбора из многих вариантов. Данную унифицированную структуру можно заменить несколькими вложенными друг в друга структурами *Развилка полная*.

Цикличные алгоритмы:

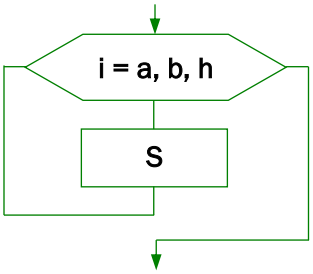
Цикл – пока. Служит для организации итерационных циклов, но в отличие от цикла – до может не выполниться ни одного раза.



Цикл – до. Служит для организации циклов с заранее неизвестным числом повторений, т.е. итерационных циклов. Цикл данного типа всегда выполняется хотя бы один раз, так как проверка условия завершения цикла проводится после выполнения тела цикла.



Цикл с параметром. Предназначен для организации повторения некоторого участка программы – тела цикла. Используется, когда число повторений цикла известно. Параметры: i – переменная цикла, a – начальное значение переменной цикла, b – конечное значение переменной цикла, h – шаг изменения переменной цикла.



i=a
i=a+h
i=a+h+h
...
i=b

Как следует из приведенных схем, любая унифицированная структура имеет один вход и один выход.

В зависимости от использованных унифицированных структур алгоритмы программных модулей, составляющих программный комплекс, могут быть линейными, разветвляющимися, циклическими и сложными.

Исполнитель алгоритма – это устройство управления, соединенное с набором инструментов. Устройство управления понимает алгоритмы и организует их выполнение, командуя соответствующими инструментами.

Представление алгоритма в виде блок-схемы позволяет наглядно отразить последовательность действий, необходимых для решения поставленной задачи.

После разработки алгоритма решения задачи и представления его в виде блок-схемы можно перейти к написанию программы –

последовательности команд на выбранном языке программирования, соответствующей разработанному алгоритму.

Программа, написанная на языке программирования, является исходной программой. Она состоит из команд, понятных человеку, но непонятных процессору компьютера. Чтобы процессор смог выполнить работу в соответствии с командами исходной программы, исходная программа должна быть переведена на машинный язык – язык команд процессора. Эту задачу выполняет специальная программа – **компилятор**. Компилятор выполняет последовательно две задачи:

- проверяет текст программы на отсутствие синтаксических ошибок;
- создает (генерирует) выполняемую программу – машинный код.

Следует отметить, что генерация выполняемой программы происходит только в том случае, если в тексте программы нет синтаксических ошибок (рис. 3.1).

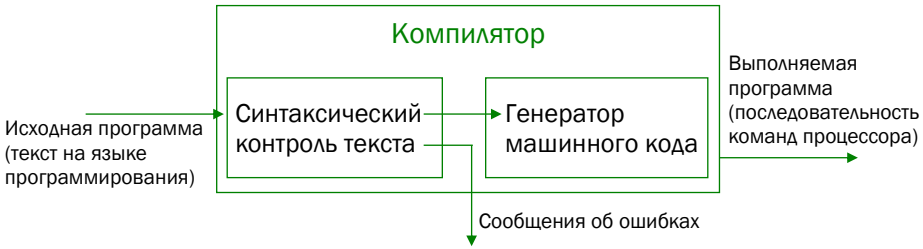


Рис. 3.1. Схема работы компилятора языка

Генерация машинного кода компилятором свидетельствует только об отсутствии в тексте программы синтаксических ошибок. Убедиться в правильности работы программы можно только во время ее тестирования – пробных запусков программы и анализа полученных результатов.

Контрольные вопросы

1. Что такое алгоритм?
2. Какими свойствами обладает алгоритм?
3. Какие способы записи алгоритма существуют?
4. Какие виды алгоритмических конструкций существуют?
5. Что такое компилятор языка?

4 Основы языка VBA

Программа, написанная на языке Visual Basic for Applications (VBA), представляет собой последовательность инструкций, которые принято называть операторами. Каждый оператор пишется с новой строки.

Если строка начинается с «`» (апострофа), то это комментарий. Комментарий – это фрагмент программы, который не влияет на ее выполнение, служит для пояснений написанных команд.

4.1. Типы данных

В VBA все данные, используемые программой, должны принадлежать к какому-либо заранее известному типу данных – стандартному (тип данных, предусмотренный VBA) или пользовательскому (тип данных, разработанный программистом).

Тип данных определяет:

- формат представления данных в памяти компьютера;
- множество допустимых значений, которые может принимать принадлежащая к выбранному типу переменная или константа;
- множество допустимых операций, применимых к этому типу.

Тип переменной или константы определяется при ее декларации (описании). Тем самым определяют формат, в котором эта переменная хранится в памяти, и диапазон ее возможных значений. VBA строго следит за соответствием типов переменных в выражениях. В общем случае с обеих сторон операции присвоения должны стоять объекты одного типа (переменная – с одной стороны и выражение – с другой). Исключением является ситуация, когда слева стоит переменная действительного типа, а справа – целочисленное выражение.

Типы данных в языке программирования VBA делятся на пять основных классов: простые типы, структурированные типы, ссылочные типы, процедурные типы, объектные типы.

К простым типам данных в VBA относятся:

- целочисленный – Integer (положительное или отрицательное), Long (двойное целое с любым знаком), Single (десятичные целые числа);
- вещественный – Double (длинные числа с плавающей точкой);
- логический – Boolean. Переменные этого типа могут принимать только два значения: истина (True) или ложь (False);
- символьный – Char – тип данных, предназначенный для хранения одного символа (управляющего или печатного) в определенной кодировке. Может являться как однобайтовым (для стандартной таблицы символов), так и многобайтовым (к примеру, для Юникода). Основным применением служит обращение к отдельным знакам строки;
- строковый – String – представляет собой статически размещаемые в памяти компьютера строки указанной при описании длины.

Таблица 4.1. **Типы данных VBA**

Тип данных	Число байт	Приблизительный диапазон значений
Boolean (булево)	2	True (истина) или False (ложь)
Integer	2	от –32768 до 32767
Long (длинное целое)	4	от –2 147 483 648 до 2 147 483 678
Single (значение с плавающей точкой одинарной точности)	4	от –3,4 x 1038 до 3,4 x 1038
Double (значение с плавающей точкой двойной точности)	8	от –1,797 x 10308 до 1,797 x 10308
Currency (денежный)	8	Данные в денежном поле не округляются во время вычислений. Значение в денежном поле содержит до 15 цифр слева от десятичной запятой и 4 цифры справа
Date	8	от 1 января (100) до 31 декабря (9999)
Object	4	Любая ссылка на объект
String	10 + длина строки	от 1 до 65 400
Variant (вариантный с числами)	16	Любое числовое значение в диапазоне Double
Variant (вариантный с символами)	22 + длина строки	от 0 до 2 миллиардов

Тип данных Date используется для работы с информацией, хранящей даты. VBA может воспринимать символ даты, не отличая их от других стандартных форматов. Для указания того, что значение является именно датой, с обеих сторон ставится символ # (#10/25/01#).

Денежный тип данных Currency используется для хранения информации в денежных единицах.

Variant – тип данных общего назначения (ссылка на любой объект).

Сведения о типах данных представлены в табл. 4.1.

Кроме перечисленных типов данных могут использоваться структурированные типы данных, например массивы.

Массивы представляют собой последовательную упорядоченную совокупность элементов некоторого типа, которые адресуются с помощью некоторого индекса. Индексная переменная, служащая для указания отдельного элемента массива, должна быть простого целочисленного типа. Элементы массива в памяти хранятся по соседству, в то время как одиночные элементы простого типа не гарантируют такого расположения данных в памяти. Другими словами, **массив** – это структура данных, которую можно рассматривать как набор переменных одинакового типа, имеющих общее имя. Массивы содержат фиксированное число элементов одного типа.

Различают одномерные и многомерные массивы данных. Размер массива в VBA ограничивается только объемом рабочей памяти компьютера.

Ключевое слово – **Array**, обозначающее, что переменная является массивом.

4.2. Понятия переменной и константы

Переменной называют элемент программы, который предназначен для хранения, коррекции и передачи данных внутри нее. Раздел объявления переменных начинается зарезервированным словом **DIM**, вслед за которым располагаются конкретные переменные.

Формат описания переменной:

Dim < имя переменной > as < тип переменной >

т.е. для объявления переменной необходимо указать ее имя и тип.

Например,

Dim tax AS Integer

Dim d as Date

Переменная tax будет целочисленного типа, а переменная d будет хранить дату.

Константа – это идентификатор, обозначающий некоторую неизменную величину (значение данных) определенного типа. Идентификатор константы не может быть включен в свое собственное описание. Константы описываются после слова CONST.

Формат описания константы:

CONST < идентификатор > = < значение >

Например:

CONST IT=0,45

Для имен констант по традиции используются только прописные буквы.

Так как встроенные константы VBA начинаются с xl и vb, эти символы не рекомендуется использовать в начале имен собственных констант.

4.3. Основные операторы языка

К основным операторам относятся:

- оператор присвоения;
- операторы проверки условия;
- операторы цикла;
- операторы ввода и вывода данных.

4.3.1. Оператор присвоения

Оператор присвоения имеет формат:

<имя переменной>=<выражение>

При записи выражения могут использоваться функции, знаки математических операций и скобки для определения приоритета операций.

При использовании данного оператора необходимо учитывать, что справа и слева должен быть одинаковый тип данных. Исключение составляет случай, когда слева – действительный тип данных, а справа – целочисленный.

4.3.2. Операторы реализации условий

Условный оператор позволяет проверить некоторое условие и в зависимости от результатов проверки выполнить то или иное действие. Таким образом, **условный оператор** – это средство ветвления вычислительного процесса.

Структура условного оператора имеет следующий вид:

```
if <условие> then <оператор s1> else <оператор s2> end if
```

где if, then, else – зарезервированные слова (если, то, иначе);

<условие> – произвольное выражение логического типа;

<оператор s1>, <оператор s2> – любые операторы языка VBA.

Этот оператор соответствует следующей алгоритмической конструкции (рис. 4.1).

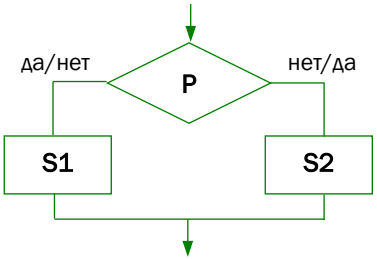


Рис. 4.1. **Алгоритмическая конструкция**
Ветвление полное

Например: даны два целых числа, если они имеют одинаковый знак, то найти их сумму, если разный или числа равны нулю, то разность.

```
if a*b>0
   then
     y=a+b
```

```
    else
      y=a-b
end if
```

Часть else <оператор s2> условного оператора может быть опущена. Тогда при значении true условного выражения выполняется <оператор s1>. В противном случае этот оператор пропускается (рис. 4.2):

```
if <условие> then <оператор s1> end if
```

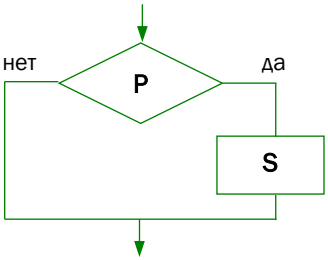


Рис. 4.2. Алгоритмическая конструкция
Ветвление неполное

Поскольку любой из операторов <оператор s1> и <оператор s2> может быть любого типа, в том числе и условным, а в то же время не каждый из «вложенных» условных операторов может иметь часть else <оператор s2>, то возникает неоднозначность трактовки условий. Эта неоднозначность в VBA решается следующим образом: любая встретившаяся часть else соответствует ближайшей к ней «сверху» части then условного оператора.

Условие может быть сложным, т.е. состоять из нескольких простых. В этом случае каждое простое условие записывается в круглых скобках и соединяется логическими операциями (and – и, or – или). Если условий много, могут стоять дополнительные круглые скобки, чтобы определить приоритет операций.

Например:

```
if (a>0) and (a<=10)
    then
        y=a+b
```

```
    else
        y=a-b
end if
```

Оператор выбора позволяет выбрать одно из нескольких возможных продолжений программы (сравнение выполняется только на знак равенства). Параметром, по которому осуществляется выбор, служит ключ выбора – выражение любого порядкового типа (любого из рассмотренных, кроме типов real и string).

Структура оператора выбора следующая:

```
SELECT Case  имя_переменной
 Case условие_выбора-1
 Операторы -1
Case условие_выбора-2
 Операторы -2
...
Case условие_выбора-n
 Операторы -n
 [Case Else
[операторы умолчания]]
end Select
```

Здесь: select, case, else, end – зарезервированные слова;

<условие_выбора> – ключ выбора, имя любой переменной;

<операторы> – произвольные операторы VBA.

Оператор выбора работает следующим образом. Переменная сравнивается со всеми значениями выражения <условие_выбора>, отыскивается такой оператор, которому она соответствует. Найденный оператор выполняется, после чего оператор выбора завершает свою работу. Если в списке выбора не будет найдено необходимое значение, соответствующее вычисленному значению переменной, управление передается операторам, стоящим за словом else. Часть else <оператор> можно опускать. Тогда при отсутствии в списке выбора нужной константы ничего не произойдет, а оператор выбора просто завершит свою работу.

Данному оператору соответствует схема, представленная на рис. 4.3.

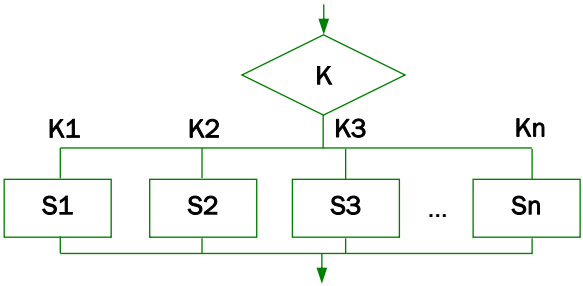


Рис. 4.3. Схема оператора выбора

Ниже приведен пример программы, которая в зависимости от значения переменной a выполняет различные вычисления. Если a=3, вычисляем сумму элементов s и d, если a>5, то их произведение, в противном случае – разность.

```
Select case a
   Case a=3
     otvet=s+d
   Case a >5
     otvet=s*d
   Case Else
     otvet=s-d
End select
```

4.3.3. Операторы реализации циклов

В языке VBA имеются два оператора цикла: цикл с параметром, цикл – пока.

Цикл с параметром имеет следующий формат:

```
For Счетчик = Начальное значение To Конечное значение [Step Шаг]
[Операторы]
[exit for]
Операторы
Next [Счетчик]
```

где операторы – произвольные операторы VBA.

В квадратных скобках указываются необязательные параметры. Конструкция [exit for] будет использоваться, если необходимо выйди из цикла до полного его завершения. В этом случае данная конструкция будет использоваться только внутри оператора проверки условия if.

Данному оператору соответствует следующая схема (рис. 4.4).

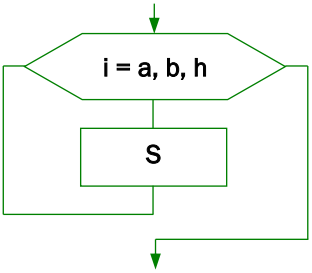


Рис. 4.4. Схема цикла с параметром

Отметим два обстоятельства. Во-первых, условие, управляющее работой оператора for, проверяется перед выполнением оператора цикла: если условие не выполняется в самом начале работы оператора for, исполняемый оператор не будет выполнен ни разу. Другое обстоятельство – если шаг не указан, то равен +1.

Например, вычислим сумму чисел от 5 до 25:

```
s =0
for i =5 to 25
        s=s+i
  next i
```

Оператор цикла – пока. Выполняется, пока условие истинно, имеет вид:

```
while  <условие окончания цикла>
операторы
wend
```

где операторы – произвольные операторы VBA.

Данному оператору соответствует схема, представленная на рис. 4.5.

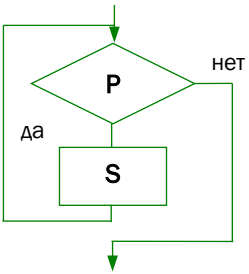


Рис. 4.5. **Схема цикла – пока**

Если <условие> имеет значение true, то выполняются <операторы>. Если <условие> имеет значение false, то оператор while прекращает свою работу и управление передается оператору, первому за циклом. Если на первом шаге условие было ложным, то оператор не выполнится ни разу, требует искусственного изменения переменной цикла. Шаг изменения переменной цикла может быть любым, сама переменная может быть как целого, так и вещественного типа.

Рассмотрим предыдущий пример:

```
s=0
i=5
while i<=25
    s=s+i
    i=i+1
wend
```

4.3.4. Операторы ввода и вывода данных

Рассмотрим форматы основных команд для обмена данными с приложением.

Ввод данных из окна ввода.

Имя_переменной = Inputbox (текст, [заголовок], [значение по умолчанию])

где текст, который отображается в окне, – обязательный параметр. Это комментарий, который объясняет, какое значение будет вводиться;

заголовок – текст заголовка окна ввода;

значение по умолчанию – величина, заданная по умолчанию.

Например, в окно ввести длину в метрах для ее перевода в сантиметры (рис. 4.6).

A=inputbox(«введите длину в метрах», «Метры – сантиметры»,» «)

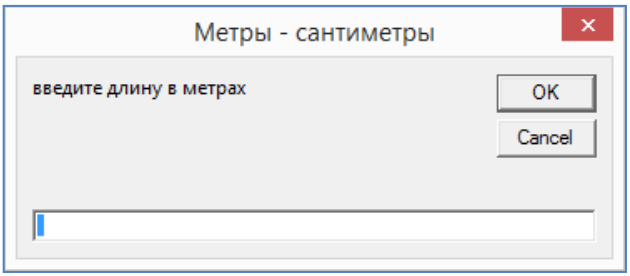


Рис. 4.6. **Окно для ввода длины в метрах**

В VBA команда **Inputbox** всегда возвращает строку, поэтому может возникнуть необходимость в преобразовании результата в число (рис. 4.7).

R= VAL(InputBox («введите строку комиссионных», «Расчет комиссионных»))

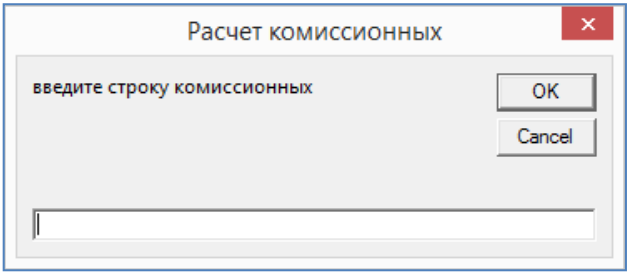


Рис. 4.7. **Окно для ввода комиссионных**

Вывод данных может выполняться командой, которая выводит сообщение и позволяет обрабатывать результат вывода:

showmessage (сообщение)

Эта процедура позволяет вывести на экран простое диалоговое окно с текстом и одной командной кнопкой, где сообщение – выражение строкового типа.

MsgBox (prompt [, buttons [, title]])

где prompt – текст, который отображается в окне сообщений;

buttons – коды кнопок, которые будут отображаться в окне сообщений;

title – текст, который появляется в окне заголовка.

VBA-команда **MsgBox** – это удобное средство, для того чтобы отобразить на экране информацию и попросить пользователей сделать выбор, щелкнув на одной из предложенных кнопок.

Функцию MsgBox можно вызывать как в виде отдельного оператора, так и в виде присваивания ее результата какой-то переменной. Если функция вызывается самостоятельно, не заключайте аргументы в круглые скобки. В следующем примере функция MsgBox выводит на экран сообщение и не возвращает результат (рис. 4.8):

```
MsgBox «Для продолжения щелкните на кнопке OK»
```

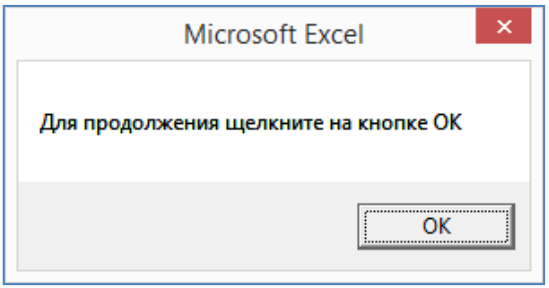


Рис. 4.8. **Вывод на экран сообщения без возврата к результату**

Чтобы запросить информацию, используя окно сообщения, можно присвоить результат функции MsgBox какой-то переменной. В приведенной ниже программе использованы некоторые встроенные константы (они будут описаны позже), чтобы упростить работу со значениями, возвращаемыми функцией MsgBox (рис. 4.9).

```
Ans = MsgBox («Продолжить ? «, vbYesNo)
Select Case Ans
   Case vbYes
     [операторы, выполняющиеся, на кнопке Да ]...
   Case vbNo
     [операторы, выполняющиеся, на кнопке Нет ]...
End Select
```

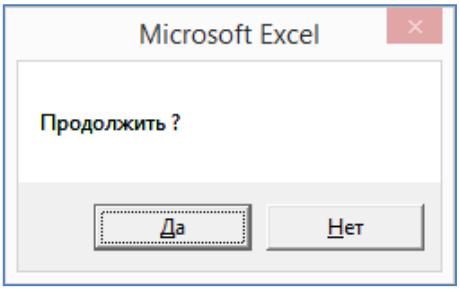


Рис. 4.9. **Вывод на экран сообщения с возвратом к результату**

Таблица 4.2. **Константы, определяющие вид сообщения**

Константа	Значение	Описание
vbOKOnly	0	Отображается только кнопка OK
vbOKCancel	1	Отображаются кнопки: OK и Отмена
vbAbortRetryIgnore	2	Отображаются кнопки: Прервать, Повторить и Пропустить
vbYesNoCancel	3	Отображаются кнопки: Да, Нет и Отмена
vbYesNo	4	Отображаются кнопки: Да и Нет
vbRetryCancel	5	Отображаются кнопки: Повторить и Отмена
vbCritical	16	Отображается значок важного сообщения
vbQuestion	32	Отображается значок Warning Query (запрос с предупреждением)
vbExclamation	48	Отображается значок Warning Message (сообщение с предупреждением)
vbInformation	64	Отображается значок информационного сообщения
vbDefaultButton1	0	Активна первая кнопка
vbDefaultButton2	256	По умолчанию активна вторая кнопка
vbDefaultButton3	512	По умолчанию активна третья кнопка
vbDefaultButton4	768	По умолчанию активна четвертая кнопка
vbApplicationModal	0	Модальность на уровне приложения. Пользователь должен ответить на сообщение, чтобы продолжить работу в текущем приложении

Окончание табл. 4.2

Константа	Значение	Описание
vbMsgBoxHelpButton	16384	Добавляет кнопку Справка в окно сообщения
vbMsgBoxSetForeground	65536	Расположение окна сообщения на переднем плане
vbMsgBoxRight	524288	Текст выравнивается по правому краю

Можно без труда настраивать окна сообщений с помощью соответствующих кодов кнопок. В табл. 4.2 перечислены встроенные константы VBA, которые используются в качестве аргументов функции MsgBox для указания кодов отражаемых кнопок. Можно указать, какие кнопки отобразить, должна ли появляться пиктограмма и какая кнопка принимается по умолчанию.

Контрольные вопросы

1. Что такое переменные и константы? Как они описываются в программе?
2. Какие типы данных можно использовать в программе? Как они классифицируются?
3. Каково назначение операторов присвоения, ввода и вывода данных?
4. Каково назначение операторов условия, выбора? Как они используются в программе?
5. Каково назначение операторов цикла – пока, цикла с параметром? Как они используются в программе?

5 Основные принципы проектирования бизнес-приложения в VBA

Разработку бизнес-приложения будем рассматривать на примере. Создадим форму, которая позволяла бы вводить исходные данные для расчета заработной платы сотрудников. Необходимо учитывать месяц начисления заработной платы, табельный номер сотрудника, количество отработанных дней, где происходят выплаты (карта, касса). Данный пример решим двумя способами: через стандартные функции Excel и форму, которая обеспечит удобный ввод данных через программирование.

5.1 Элементы управления панели инструментов – Форма

Чтобы начать работу с элементами панели инструментов формы, следует выполнить следующие действия.

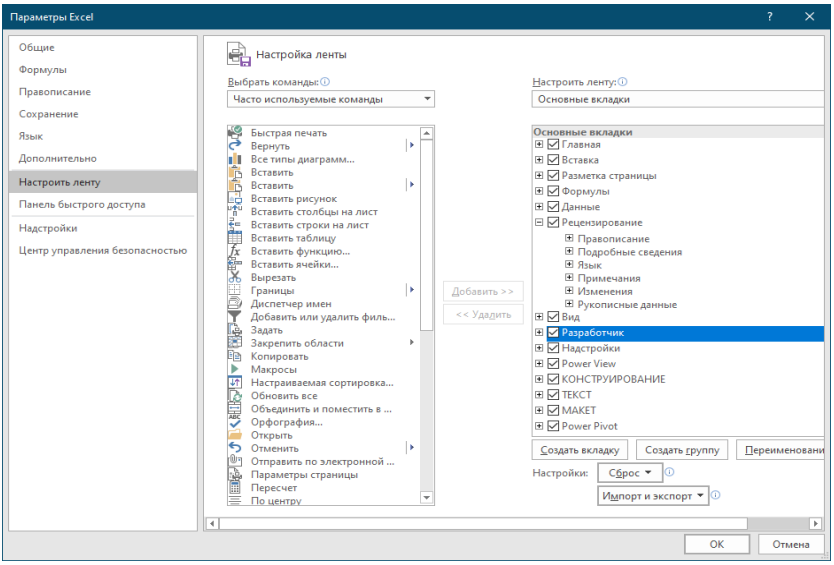


Рис. 5.1. Подключение панели **Разработчик**

Необходимо выполнить команду **Файл – Настройки** и в появившемся диалоговом окне (рис. 5.1) выбрать команду **Настроить ленту**, далее следует включить флажок в строке **Разработчик**.

Теперь в главном меню появились вкладка **Разработчик** и возможность работы с элементами управления: список, счетчик, флажок, переключатель, кнопка и др. (рис. 5.2).

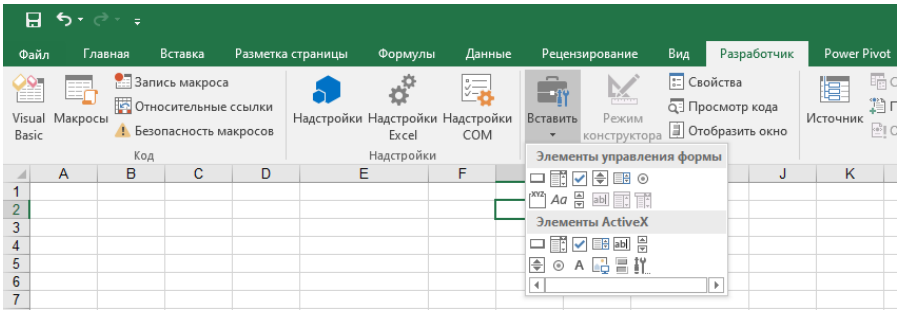
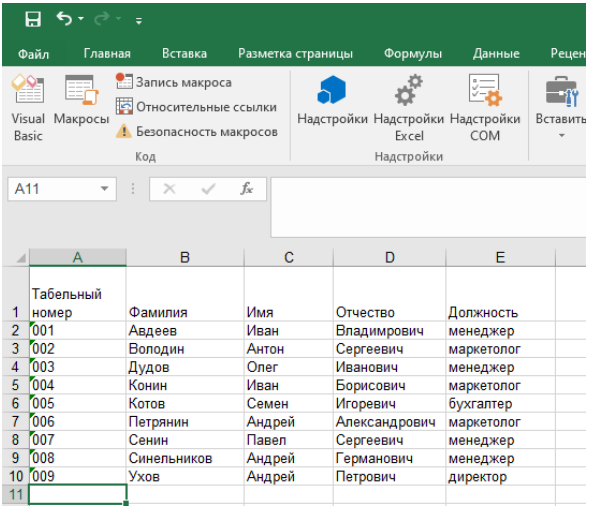


Рис. 5.2. Панель инструментов вкладки **Разработчик**

Рассмотрим работу с элементами управления на следующем примере: создадим на отдельном рабочем листе ведомость расчета заработной платы сотрудников организации. Этот лист так и называется **Ведомость**.



	A	B	C	D	E
1	Табельный номер	Фамилия	Имя	Отчество	Должность
2	001	Авдеев	Иван	Владимирович	менеджер
3	002	Володин	Антон	Сергеевич	маркетолог
4	003	Дудов	Олег	Иванович	менеджер
5	004	Конин	Иван	Борисович	маркетолог
6	005	Котов	Семен	Игоревич	бухгалтер
7	006	Петрянин	Андрей	Александрович	маркетолог
8	007	Сенин	Павел	Сергеевич	менеджер
9	008	Синельников	Андрей	Германович	менеджер
10	009	Ухов	Андрей	Петрович	директор
11					

Рис. 5.3. Лист **База данных**

На других листах содержатся:

1) информация о сотрудниках: фамилия, имя, отчество, должность. Этот лист имеет название **База данных** (рис. 5.3);

2) справочная информация об окладах сотрудников. Этот лист имеет название **Оклады** (рис. 5.4).

На отдельном листе, который называется **Форма**, создадим с помощью **Элементов управления**

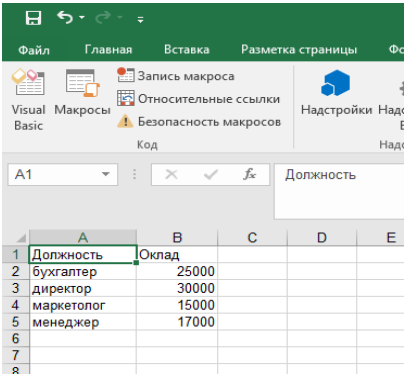


Рис. 5.4. **Лист Оклады**

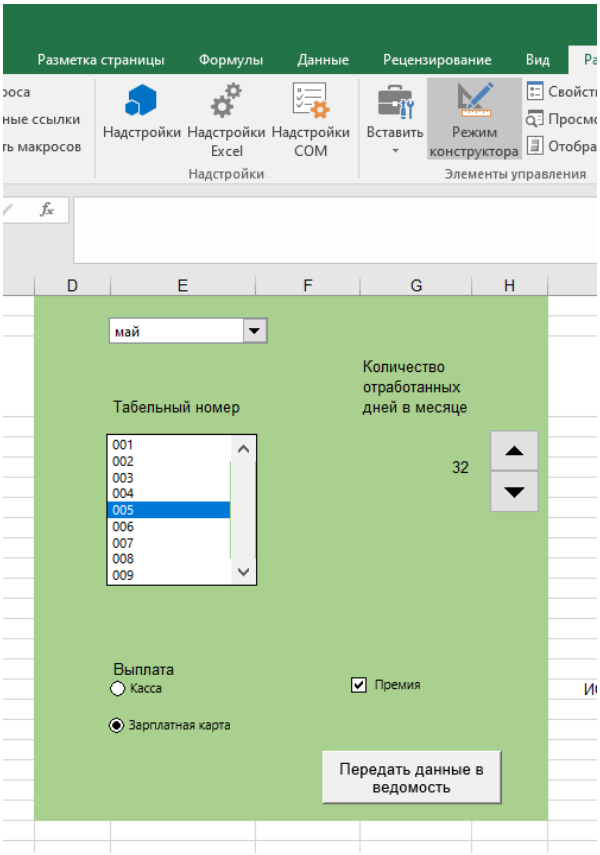


Рис. 5.5. **Лист Форма**

панели инструментов **Форма** удобный интерфейс для ввода исходной информации, которая требуется для расчета заработной платы сотрудников. Она содержит: список с возможностью выбора табельного номера сотрудника, переключатель для указания выплаты заработной платы, счетчик (или полосу прокрутки) для задания количества отработанных дней, назначение премии (флажок) (рис. 5.5).

Создание списков.

1. Прорисуйте объект **Поле со списком**.
2. Вызовите из контекстного меню команду **Формат объекта**.
3. Укажите блок ячеек, на основании данных которого формируется список, и обязательно ссылку на ячейку, с которой связан данный элемент управления (рис. 5.6).

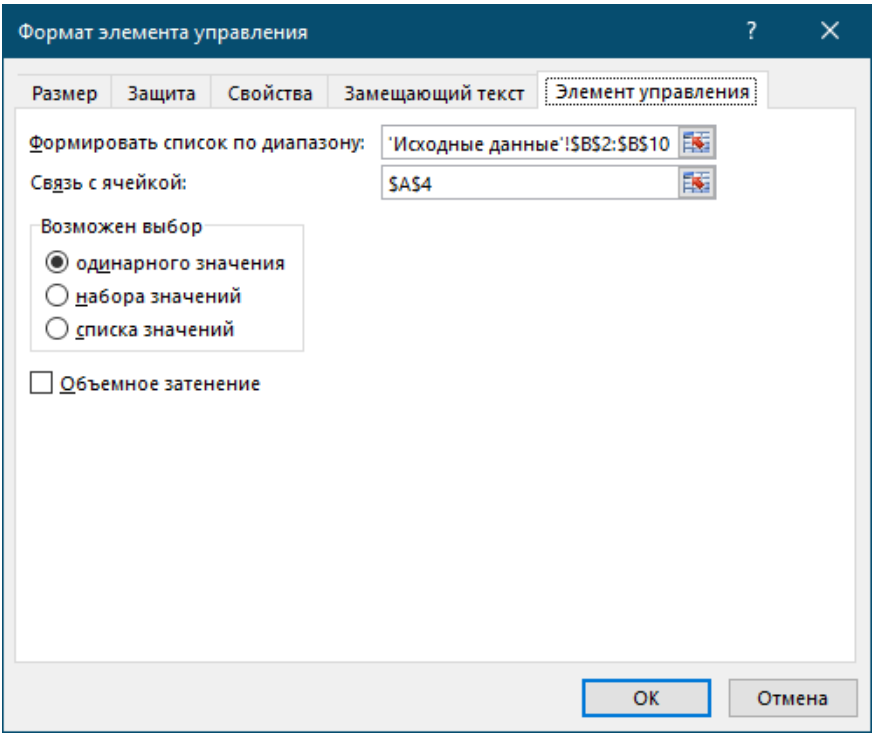


Рис. 5.6. Создание списков

При работе со списком Excel в связанной ячейке содержится порядковый номер элемента указанного диапазона.

Создание счетчиков.

1. Прорисуйте объект **Счетчик**.
2. Вызовите из контекстного меню команду **Формат объекта**.
3. Укажите ячейку, с которой связан данный элемент управления, а также дополнительные параметры работы объекта **Счетчик** (рис. 5.7).

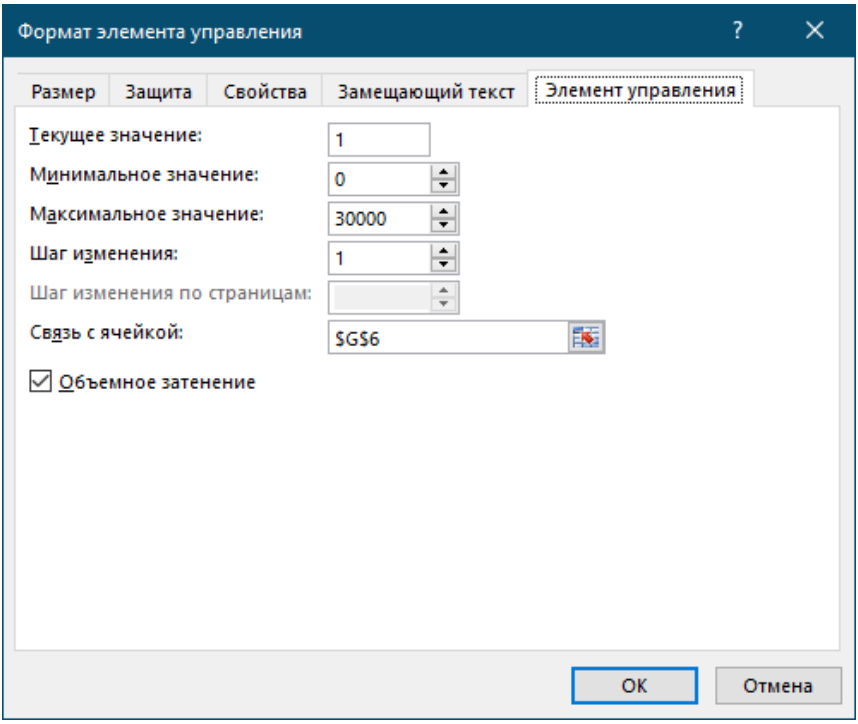


Рис. 5.7. Формат элемента управления **Счетчик**

Создание полосы прокрутки.

1. Прорисуйте объект **Полоса прокрутки**.
2. Вызовите из контекстного меню команду **Формат объекта**.
3. Укажите ячейку, с которой связан данный элемент управления, а также дополнительные параметры работы объекта **Полоса прокрутки** (рис. 5.8).

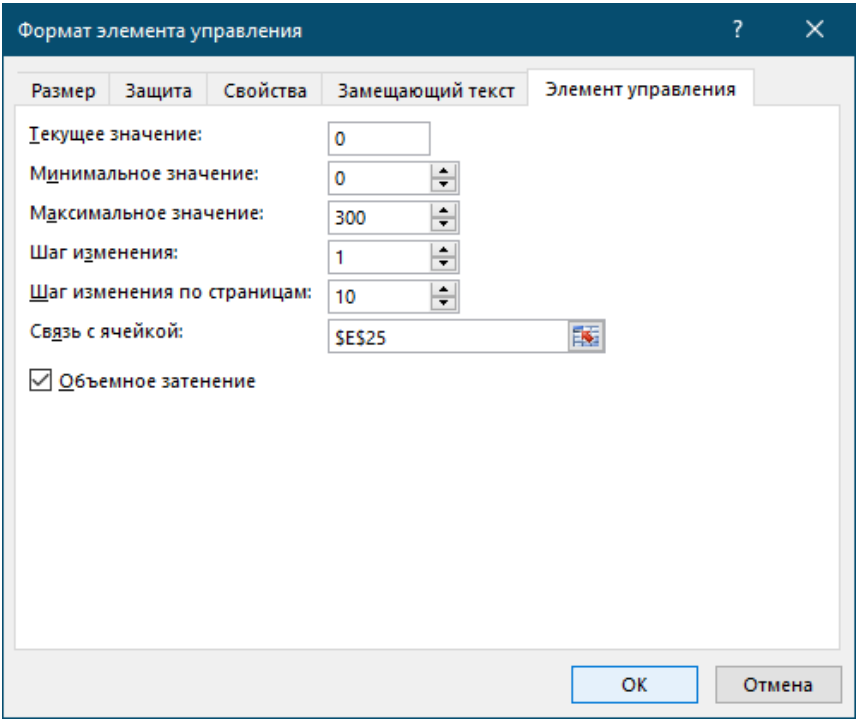


Рис. 5.8. Формат элемента управления **Полоса прокрутки**

Создание переключателей.

1. Прорисуйте объект **Переключатель**.
2. Вызовите из контекстного меню команду **Формат объекта**.
3. Укажите ячейку, с которой связан данный элемент управления, а также дополнительные параметры работы объекта **Переключатель**.

В связанной с элементом ячейке отражается порядковый номер переключателя. Далее целесообразно использовать логическую функцию ЕСЛИ для хода решения задачи при выборе того или иного **Переключателя** (рис. 5.9).

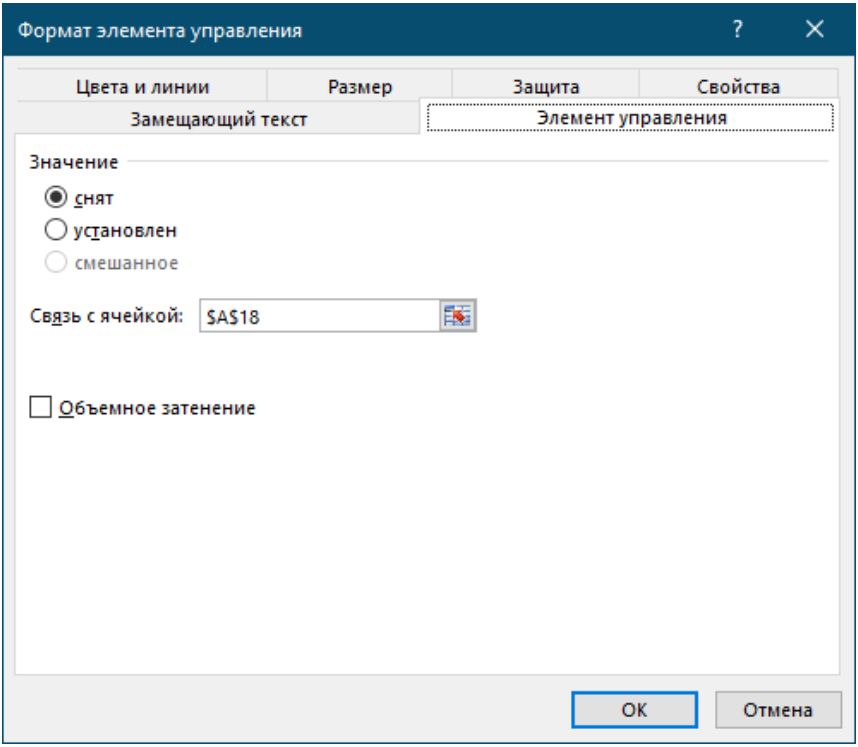


Рис. 5.9. Формат элемента управления **Переключатель**

Создание флажка.

1. Прорисуйте объект **Флажок**.
2. Вызовите из контекстного меню команду **Формат объекта**.
3. Укажите ячейку, с которой связан данный элемент управления, а также дополнительные параметры работы объекта **Флажок** (рис. 5.10).

В связанной с элементом ячейке отражается ЛОЖЬ, если переключатель отключен, и ИСТИНА, если он включен. Далее целесообразно использовать логическую функцию ЕСЛИ для хода решения задачи при выборе того или иного решения.

Мы создали форму, далее формируем на отдельном листе ведомость, которая будет выглядеть следующим образом (рис. 5.11).

Рассмотрим, каким образом информация попадает во вторую строку.

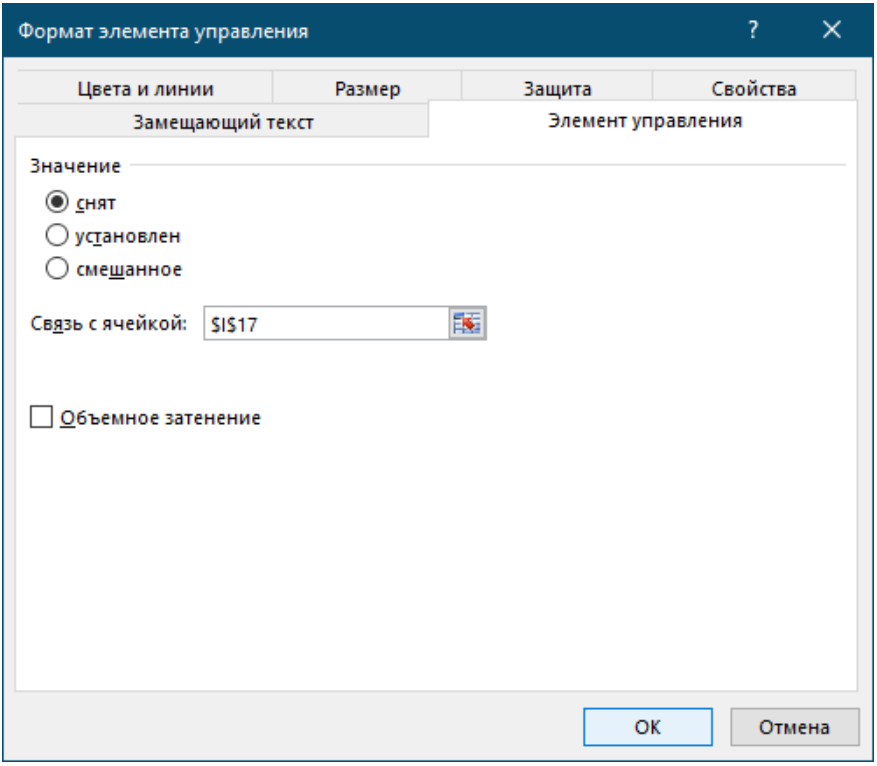
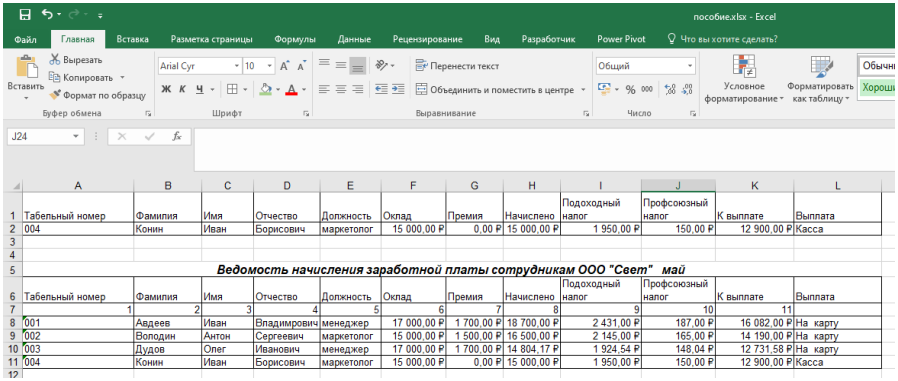


Рис. 5.10. Формат элемента управления **Флажок**



	A	B	C	D	E	F	G	H	I	J	K	L
1	Табельный номер	Фамилия	Имя	Отчество	Должность	Оклад	Премия	Начислено	Подоходный налог	Профсоюзный налог	К выплате	Выплата
2	004	Конин	Иван	Борисович	маркетолог	15 000,00 ₽	0,00 ₽	15 000,00 ₽	1 950,00 ₽	150,00 ₽	12 900,00 ₽	Касса
3												
4												
5	***Ведомость начисления заработной платы сотрудникам ООО "Свет" май***											
6	Табельный номер	Фамилия	Имя	Отчество	Должность	Оклад	Премия	Начислено	Подоходный налог	Профсоюзный налог	К выплате	Выплата
7	1	2	3	4	5	6	7	8	9	10	11	
8	001	Авдеев	Иван	Владимирович	менеджер	17 000,00 ₽	1 700,00 ₽	18 700,00 ₽	2 431,00 ₽	187,00 ₽	16 082,00 ₽	На карту
9	002	Володин	Антон	Сергеевич	маркетолог	15 000,00 ₽	1 500,00 ₽	16 500,00 ₽	2 145,00 ₽	165,00 ₽	14 190,00 ₽	На карту
10	003	Дудов	Олег	Иванович	менеджер	17 000,00 ₽	1 700,00 ₽	14 804,17 ₽	1 924,54 ₽	148,04 ₽	12 731,58 ₽	На карту
11	004	Конин	Иван	Борисович	маркетолог	15 000,00 ₽	0,00 ₽	15 000,00 ₽	1 950,00 ₽	150,00 ₽	12 900,00 ₽	Касса
12												

Рис. 5.11. Ведомость начисления заработной платы

Выбранный конкретный табельный номер в созданной форме имеет в ячейке, связанной со списком табельных номеров, порядковый номер элемента указанного диапазона.

Используя функцию **ИНДЕКС**, которая находится в категории **Ссылки и массивы**, мы разместим в ячейке A2 выбранный табельный номер сотрудника.

Эта функция имеет 3 аргумента:

1. Массив значений.
2. Номер строки, из которой извлекается информация.
3. Номер столбца, из которого извлекается информация.

В нашем случае:

1-й аргумент – это весь перечень табельных номеров;

2-й аргумент – ссылка на ячейку, которая связана с элементом управления **Список**, увеличенным на единицу, так как информация о табельных номерах начинается со второй строки листа **База данных**;

3-й аргумент – равен 1, так как наш массив состоит из одного столбца (рис. 5.12).

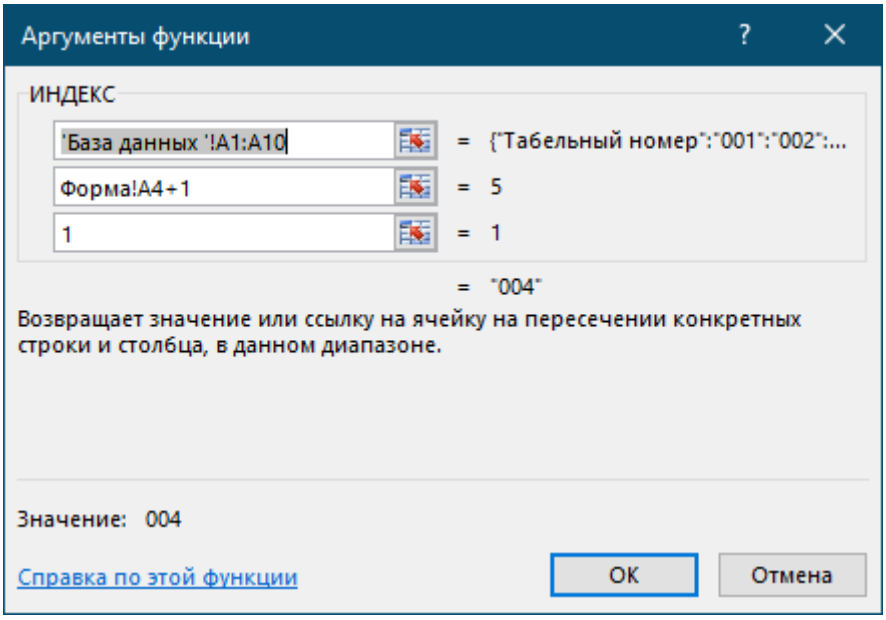


Рис. 5.12. Аргументы функции ИНДЕКС

Итак, в ячейке А2 листа **Ведомость** содержится выбранный табельный номер сотрудника.

Используя знакомую функцию **ВПР**, получаем в последующих графах, соответственно:

Столбец B – фамилию сотрудника

=ВПР(Ведомость!E2;Оклады!$A$1:$B$5;2;0)

Столбец C – имя сотрудника

=ВПР(Ведомость!E2;Оклады!$A$1:$B$5;3;0)

Столбец D – отчество сотрудника

=ВПР(Ведомость!E2;Оклады!$A$1:$B$5;4;0)

Столбец E – должность сотрудника

=ВПР(Ведомость!E2;Оклады!$A$1:$B$5;5;0)

Столбец F – оклад сотрудника

=ВПР(Ведомость!E2;Оклады!$A$1:$B$5;2;0)

Столбец G – расчет 10% премии

=ЕСЛИ(Форма!I17=ИСТИНА;10%*F2;0)

Форма!I17 – ссылка на ячейку, которая содержит слово ИСТИНА, если флажок включен, и ЛОЖЬ – в противном случае

Столбец H – расчет графы **Начислено**

=(F2+G2)*Форма!G6/Форма!G13

Форма!G6 – ссылка на ячейку, которая содержит количество отработанных дней сотрудником

Форма!G13 – ссылка на ячейку, которая содержит количество рабочих дней в отчетном месяце

Столбец I – расчет подоходного налога от начисленной суммы (13%)

Столбец J – расчет профсоюзного налога от начисленной суммы (1%)

Столбец L – способ выплаты заработной платы

=ЕСЛИ(Форма!A18=1;»Касса»;»На карту»)

Форма!A18=1 – ссылка на ячейку, которая содержит 1, если выбран первый переключатель (выплата в кассе), 2 – если выбран второй переключатель (выплата на карту).

Для того, чтобы перенести данные, которые связаны с Формой, в создаваемую ведомость, следует вспомнить режимы записи –

абсолютный либо относительный, которые имеют большое значение, так как от них зависит работа макроса (см. раздел 2.3).

На панели инструментов **Остановка записи** есть кнопка **Относительная ссылка**. Если щелкнуть по этой кнопке во время записи макроса, Excel перейдет из абсолютного **(по умолчанию)** в относительный режим записи. Текст макроса будет выглядеть следующим образом:

```
Sub абсолютно()
 Range(«c1:c5»).Select
End Sub

Sub относительно()
'относительно ячейки A1
ActiveCell.Offset(0, 2).Range(«A1:A5»).Select
End Sub
```

Для того, чтобы, не прибегая к программированию, создать ведомость, которая указана на рис. 5.11, создадим в диапазоне ячеек A6:L7 **Базу данных**.

Далее запишем макрос (**Макрос – Запись макроса**), который назовем **Ведомость**, содержащий следующую последовательность действий:

1. Выделим те данные, которые получили из созданной Формы. Они содержатся в диапазоне A2:L2.

2. Скопируем их в буфер обмена.

3. Перейдем в начало **Базы данных** – ячейка A6.

4. Определим конец **Базы данных** с помощью сочетания клавиш:
CTRL+SHIFT+(стрелка вперед),
CTRL+(стрелка вниз).

5. Включим режим относительной адресации, так как каждый раз место вставки информации, взятой в буфер, будет изменяться.

6. Перейдем на одну ячейку вниз, именно она будет определять начало вставки информации из буфера обмена.

7. Выполним команду **Вставка – Специальная вставка –Значения.**

8. Нажмем клавишу ESC для снятия выделения.

Далее следует остановить запись макроса и закрепить его за созданной кнопкой.

Текст записанного макроса выглядит так:

```
Sub Ведомость()
" Ведомость Макрос
'    Range(«A2:L2»).Select
    Selection.Copy
    Range(«A6»).Select
    Range(Selection, Selection.End(xlToRight)).Select
    Selection.End(xlDown).Select
    ActiveCell.Offset(1, 0).Range(«A1»).Select
    Selection.PasteSpecial Paste:=xlPasteValues, Opera-
tion:=xlNone, SkipBlanks _
        :=False, Transpose:=False
    Application.CutCopyMode = False
End Sub
```

Задания для выполнения лабораторной работы

При выполнении лабораторной работы всем листам давайте смысловые названия.

1. По выданному преподавателем индивидуальному заданию подготовьте:

- на первом листе рабочей книги базу данных, состоящую не менее чем из 15 записей;
- на втором – справочные данные;
- на третьем листе, используя панель инструментов **Форма**, организуйте ввод данных. Создайте и обработайте необходимые элементы управления: счетчики, полосы прокрутки, списки, переключатели, кнопки (Печать ведомости, Просмотр ведомости, Очистка, Сохранение и др.). Выполните интерфейс в соответствии с вашим эстетическим вкусом;
- на четвертом – Бланк-ведомость с выходными данными для печати. Выполните необходимую обработку данных, используя функции ВПР, ИНДЕКС, БИЗВЛЕЧЬ и пр.

2. В строке **Меню** добавьте две собственные пиктограммы, которые скрывали бы рабочие элементы листа и возвращали первоначальное его состояние.

3. Подготовьте отчет о проделанной работе.

Задания для самостоятельной работы

Вариант 1

1. Создайте таблицу, содержащую сведения о работниках, перечисляющих заработную плату в банк, со следующими полями: табельный номер, название банка (3-4 наименования), номер лицевого счета.

2. Создайте таблицу (справочник), каждая запись которой должна состоять из следующих полей: табельный номер работника, его фамилия, должность.

3. Входные данные: табельный номер работника (список), месяц (список) и процент перечисления (счетчик), сумма начисленной заработной платы.

4. Создайте ведомость для каждого работника и отразите в ней необходимые данные: фамилия работника, его должность, начисленная заработная плата, перечисления в банк (его название) по номеру лицевого счета, итоговая выплата за определенный месяц с учетом вычетов.

5. Напишите отчет о ходе выполнения работы.

Вариант 2

1. Создайте таблицу (базу данных), содержащую сведения о работниках. Каждая запись должна состоять из следующих полей: наименование подразделения (3-4 названия), табельный номер, фамилия работника, должность, оклад, количество иждивенцев, группа инвалидности.

2. Создайте таблицу (справочник), в которой содержатся сведения о надбавках к основному окладу со следующими полями: код надбавки, сумма надбавки.

3. Вводимые данные: месяц (список), табельный номер работника (список), код надбавки (в виде списка), процент премии (счетчик).

4. Ведомость включает в себя: месяц, табельный номер работника, его фамилию, подразделение, должность, сумму выплат с учетом надбавок и вычетов (подоходный налог и вычеты в пенсионный фонд). Подоходный налог рассчитывается по следующей формуле:

13% * (Начислено – (кол-во иждивенцев + 1) *
* Минимальная_ЗПЛ * кол-во льгот)

Количество льгот указано в следующей таблице:

Группа инвалидности	Количество льгот
1	5
2	3
3	1

Выплаты в пенсионный фонд – 1% от всех начислений.

5. Напишите отчет о ходе выполнения работы.

Вариант 3

1. Создайте таблицу, содержащую сведения о сотрудниках фирмы, со следующими полями: наименование подразделения (3-4 номера), табельный номер, фамилия работника, должность, дата принятия на работу, группа инвалидности, код оплаты.

2. Создайте таблицу (справочник), содержащую сведения об оплате по тарифной ставке: код оплаты, тарифная ставка.

3. Вводимые данные: месяц (список), табельный номер (список), количество отработанных часов (счетчик).

4. Ведомость включает в себя: месяц, наименование подразделения, фамилию, должность, стаж, надбавку за стаж (если стаж более 20 лет – надбавка 25%, в остальных случаях –10%), заработную плату и итоговую выплату за конкретный месяц с учетом вычета подоходного налога.

Подоходный налог рассчитывается по следующей формуле:

13% * (Начислено – (кол-во иждивенцев + 1) *
* Минимальная_ЗПЛ * кол-во льгот)

Количество льгот указано в следующей таблице:

Группа инвалидности	Количество льгот
1	5
2	3
3	1

5. Напишите отчет о ходе выполнения работы.

Вариант 4

1. Создайте таблицу, содержащую сведения о студентах, сдавших сессию: номер зачетной книжки, фамилия, номер курса, дата рождения, специальность.

2. Создайте таблицу, содержащую сведения о надбавках к стипендии: код надбавки, наименование надбавки (без надбавки, на транспорт иногородним, министерская, президентская), размер надбавки.

3. Вводимые данные: номер зачетной книжки (список), результат сдачи сессии в баллах (счетчик), базовый размер стипендии (счетчик), код надбавки (список).

4. Ведомость включает в себя: фамилию, возраст, номер зачетной книжки, номер курса, специальность, наименование и размер надбавки, назначенную стипендию с учетом надбавок. (Если результат сдачи сессии ниже установленного – стипендию не назначать.)

5. Напишите отчет о ходе выполнения работы.

Вариант 5

1. Создайте файл, содержащий сведения об альбоме. Каждая запись должна содержать следующие поля: код альбома, название альбома, цена альбома, дата записи, студия записи.

2. Создайте таблицу, содержащую сведения об исполнителях музыкальных альбомов, со следующими полями: код альбома, исполнитель (3-4 исполнителя).

3. Входные данные: дата продажи, код альбома (список), количество проданных альбомов (счетчик).

4. Ведомость включает в себя: дату продажи, название альбома, студию записи, количество проданных альбомов, скидку (если продажа в течение месяца от даты записи, то скидка 20%) и сумму, полученную от продажи данного альбома.

5. Напишите отчет о ходе выполнения работы.

Вариант 6

1. Создайте таблицу, содержащую сведения о сотрудниках, со следующими полями: фамилия, номер отдела (3-4 отдела), код должности, оклад, дата приема на работу.

2. Создайте таблицу (справочник), содержащую сведения о надбавках: код должности (1-5), наименование должности, надбавка за должность.

3. Вводимые данные: месяц (список), фамилия сотрудника (список), код должности, процент надбавки (счетчик).

4. Ведомость включает в себя: месяц, фамилию сотрудника, номер отдела, должность и сумму выплат с учетом всех надбавок: надбавка за должность и надбавка за стаж (0–5 лет – 10%, 6–20 – 30%; свыше 20 – 50%).

5. Напишите отчет о ходе выполнения работы.

Вариант 7

1. Создайте таблицу, содержащую сведения о наличии лекарств в городе, со следующими полями: код лекарства, номер аптеки (3-4 аптеки), дата поступления, единица измерения, поступившее количество.

2. Создайте таблицу, содержащую сведения о лекарствах: код лекарства, наименование лекарства, срок хранения до, страна изготовитель.

3. Вводимые данные: текущая дата, наименование лекарства (список), количество, проданное за день (счетчик), тип лекарства – детское/взрослое (флажок или переключатель).

4. Ведомость включает в себя: текущую дату, номер, наименование лекарства, страну-изготовителя, срок хранения, итоговую сумму продаж. Если срок хранения истек, выдать соответствующее сообщение – не выдавать. Если тип лекарства «детское», то скидка – 40%. В случае, когда заказываемое количество превышает имеющееся на складе, – выдать соответствующее сообщение.

5. Напишите отчет о ходе выполнения работы.

Вариант 8

1. Создайте таблицу, содержащую сведения о наличии товаров на складе: код товара, номер склада (1-3 склада), остаток.

2. Создайте таблицу, содержащую сведения о товарах. Каждая запись должна содержать следующие поля: код товара, наименование товара, фирма-изготовитель, цена товара.

3. Вводимые данные: текущая дата, фамилия заказчика, код товара (список), заказанное количество (счетчик или полоса прокрутки).

4. Ведомость включает в себя: текущую дату, фамилию покупателя, наименование товара, фирму изготовителя, номер склада, общую выручку с учетом скидки. Если количество заказанных товаров больше 100, то скидка – 10%. В случае, когда количество заказываемых товаров превышает остаток на складе, – выдать соответствующее сообщение.

5. Напишите отчет о ходе выполнения работы.

Вариант 9

1. Создайте таблицу, содержащую сведения о наличии материала на складе, со следующими полями: код материала, дата поступления, единица измерения.

2. Создайте таблицу, содержащую данные о цене материалов. Каждая запись должна состоять из следующих полей: код материала, наименование материала, цена за единицу измерения.

3. Вводимые данные: дата, фамилия покупателя, дата рождения покупателя, код материала (список), требуемое количество материала (счетчик).

4. Ведомость включает в себя: дату, фамилию покупателя, наименование материала, количество, единицу измерения, общую стоимость купленного материала с учетом скидки. Если материал куплен в день рождения покупателя, скидка – 15%.

5. Напишите отчет о ходе выполнения работы.

Вариант 10

1. Создайте таблицу, содержащую сведения о наличии путевок: код маршрута, количество.

2. Создайте таблицу, содержащую данные о цене путевки. Каждая запись должна состоять из следующих полей: код маршрута, наименование маршрута, цена.

3. Вводимые данные: текущая дата, фамилия покупателя, дата рождения покупателя, код маршрута (список), заказываемое количество путевок (переключатели): взрослых и детских.

4. Ведомость включает в себя: дату, фамилию покупателя, код маршрута, наименование маршрута, количество путевок (детских и взрослых), общую стоимость заказанных путевок с учетом скидки. Если путевка куплена в день рождения покупателя, скидка – 15%, льготы на детей – 30%. Если количество заказываемых путевок превышает имеющееся, выдать соответствующее сообщение.

5. Напишите отчет о ходе выполнения работы.

Контрольные вопросы

1. Что отражается в ячейке, связанной с элементом управления **Счетчик**?
2. Что отражается в ячейке, связанной с элементом управления **Список**?
3. Что отражается в ячейке, связанной с элементом управления **Переключатель**?
4. Что отражается в ячейке, связанной с элементом управления **Флажок**?
5. Зачем нужен режим абсолютной и относительной записи макросов?

5.2. Создание пользовательских форм

Как правило, функций Input:Box и MsgBox вполне достаточно для ведения диалога с пользователем.

Рис. 5.13. Панель **Элементы управления ToolBox**

Однако, если нужно запросить дополнительную информацию, необходимо построить новое диалоговое окно, которое создается на пользовательской форме с помощью средства **Редактор VBA** с использованием элементов управления ActiveX.

Панель элементов Элементы управления. В табл. 5.1 приведены основные элементы управления, содержащиеся на панели **Элементы управления ToolBox** (рис. 5.13).

Таблица 5.1. Состав панели инструментов Элементы управления (ToolBox)

Элемент управления	Описание
Надпись (Label)	Добавляет надпись
Поле (TextBox)	Добавляет поле ввода текста
Поле со списком (ComboBox)	Добавляет поле с раскрывающимся списком
Список (ListBox)	Добавляет список
Флажок (CheckBox)	Добавляет флажок
Переключатель (OptionButton)	Добавляет переключатель
Выключатель (ToggleButton)	Добавляет выключатель
Рамка (Frame)	Контейнер для других объектов
Кнопка (CommandButton)	Командная кнопка
Полоса прокрутки (ScrollBar)	Добавляет полосу прокрутки
Счетчик (SpinButton)	Добавляет счетчик
Рисунок (Image)	Добавляет элемент управления, который может содержать изображение

Свойства элементов управления. Сама форма и каждый элемент управления, добавляемый на пользовательскую форму, имеют определенное количество свойств, которые определяют внешний вид элемента и его поведение. Такие свойства, как высота (Height) и ширина (Width), можно изменить, щелкнув и перетащив границу рамки.

Каждый элемент управления обладает своим собственным (уникальным) набором свойств. Но многие объекты имеют общие свойства.

- Caption (Заголовок) – текст, отображаемый на элементе управления.
- Value (Значение) – значение элемента управления.
- Visible (Видимый) – при равенстве значению Ложь (False) элемент скрыт.
- Name (Имя) – имя элемента управления. По умолчанию в основу имени положен тип элемента. Это имя можно заменить любым допустимым именем.
- BackColor (Цвет фона) – цвет фона элемента управления.
- BackStyle (Стиль фона) – стиль фона (прозрачный или нет).

Для задания других свойств диалогового окна или любого элемента следует вызвать контекстное меню на объекте и выбрать команду **Properties**.

В верхней части окна **Properties** содержится раскрывающийся список, позволяющий выбирать элемент управления для работы с его свойствами. Кроме того, можно выбрать элемент, щелкнув на него, и при этом также отобразятся его свойства.

Чтобы получить сведения о конкретном свойстве, выберите его в окне **Properties** и нажмите **<F1>**. Тщательно подготовленная интерактивная справочная система по элементам управления пользовательскими формами предоставит вам исчерпывающую информацию.

Обработка событий. Когда вы вставляете пользовательскую форму, эта форма может также содержать VBA-подпрограммы для обработки событий, которые генерируются формой.

Событие – это то, что происходит, когда пользователь воздействует на элемент управления. К событиям, например, можно отнести щелчок на кнопке или выбор элемента в списке. Чтобы сделать диалоговое окно полезным, необходимо написать VBA-программу, которая при возникновении определенных событий выполняла бы нужные нам действия (табл. 5.2).

Таблица 5.2. **События пользовательской формы**

Событие	Когда происходит
Initialize	При отображении формы на экране
Terminate	При закрытии формы

Подпрограммы обработки событий носят имена, в которых название элемента управления объединено с событием с помощью символа подчеркивания:

- Click – при щелчке мыши;
- DblClick – при двойном щелчке;
- KeyPress – при нажатии клавиши;
- Change – при изменении значений элементов управления;
- GotFocus – когда объект получает фокус (становится текущим);
- LotFocus – когда объект теряет фокус;
- Error – при возникновении ошибки.

Программный код по событию Initialize объекта **UserForm** может быть использован для начальных установок и задания начальных значений:

```
Option Explicit
Dim Chiclo as Integer, Stroka as String
Private Sub UserForm3_Initialaze()
Chislo=100
Stroka= «Компания Самараэнерго»
```

Создание диалогового окна. Для создания новой пользовательской формы (рис. 5.14):

1. Войдите в редактор VBA.
2. Убедитесь, что выбрана текущая рабочая книга.
3. Выполните команду **Insert – UserForm**. В окне VBA отобразится пустая форма.

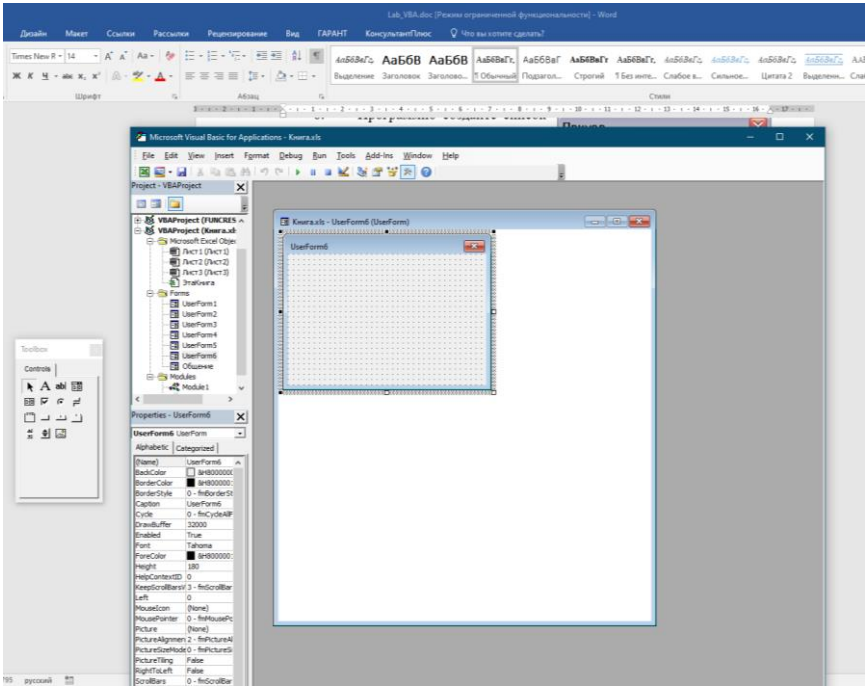


Рис. 5.14. Создание диалогового окна

4. Когда вы активизируете форму, появится панель элементов **Элементы управления ToolBox**, которая используется для добавления элементов управления к диалоговому окну.

5. Добавьте к диалоговому окну нужные элементы управления.

6. Создайте VBA-подпрограммы **Обработчик событий**, которые выполняются в случае, если пользователь *манипулирует* элементами управления (например, щелкает на кнопке **ОК**).

7. Запустите форму на выполнение.

Создание подпрограммы Обработчик событий. Рассмотрим подпрограмму для обработки события Click, которое генерируется, когда пользователь щелкает на кнопке **ОК**.

1. Войдите в редактор VBA.

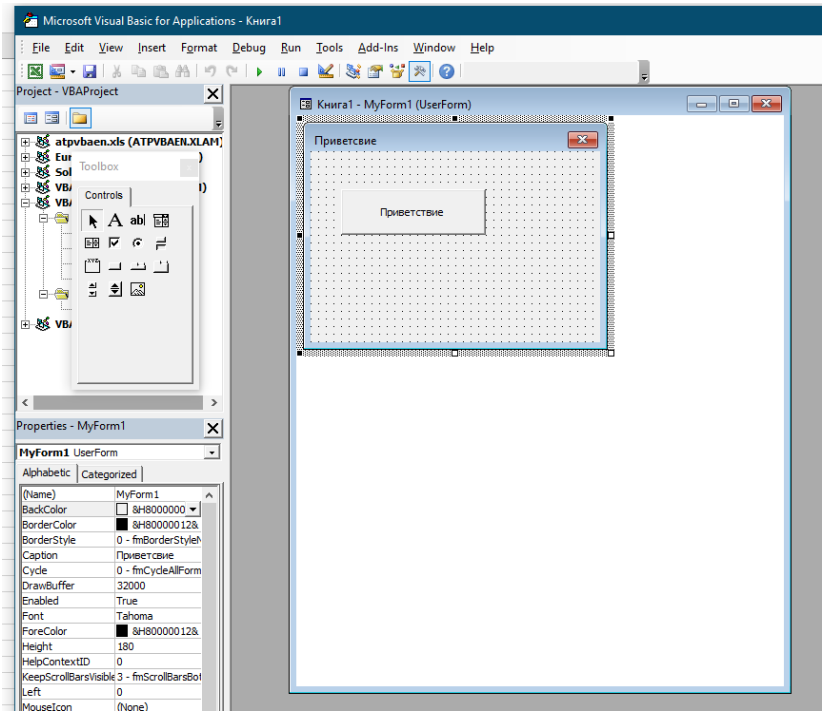


Рис. 5.15. Вид формы с кнопкой

2. В окне VBA выберите команду **Insert – UserForm** и создайте собственную форму с именем **MyForm1**. Новое имя формы задается с

помощью свойства Name, другое название формы – с помощью свойства Caption окна **Property**. Внесите другие изменения, чтобы форма нравилась. Можно изменить размер формы, передвинуть, увеличить или уменьшить элементы управления.

3. Нарисуйте элемент управления CommandButton на форме. Для изменения текста на кнопке воспользуйтесь свойством Caption.

4. Дважды щелкните на элемент CommandButton1.

5. VBA активизирует модуль для пользовательской формы и вставит в него текст программы «Приветствую ВАС!!!» (см. рис. 5.15).

Форму на выполнение можно запустить с помощью пиктограммы **Run Sub/UserForm**.

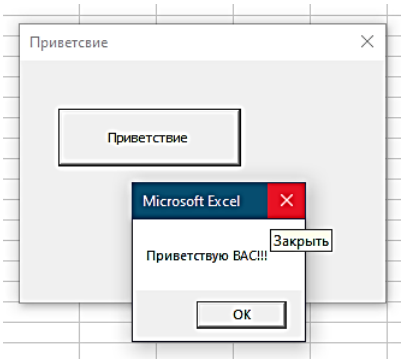


Рис. 5.16. **Диалоговое окно «Приветствую ВАС!!!»**

Отображение пользовательской формы из VBA-процедуры. С этой целью воспользуйтесь методом Show объекта **UserForm1** и напишите VBA-процедуру для ее запуска.

```
sub showDialog()
myForm1.Show
End Sub
```

Процедура отображает диалоговое окно, расположенное на форме **myForm1** (рис. 5.16).

В табл. 5.3 приведены другие основные методы пользовательской формы.

Таблица 5.3. **Основные методы пользовательской формы**

Метод	Действия
Show	Отображает форму на экране
Hide	Закрывает форму
Move	Перемещает форму

Пример создания пользовательской формы. Создадим форму, которая позволяла бы вводить исходные данные для расчета заработной платы сотрудников, рассмотренную в предыдущей главе (рис. 5.17).

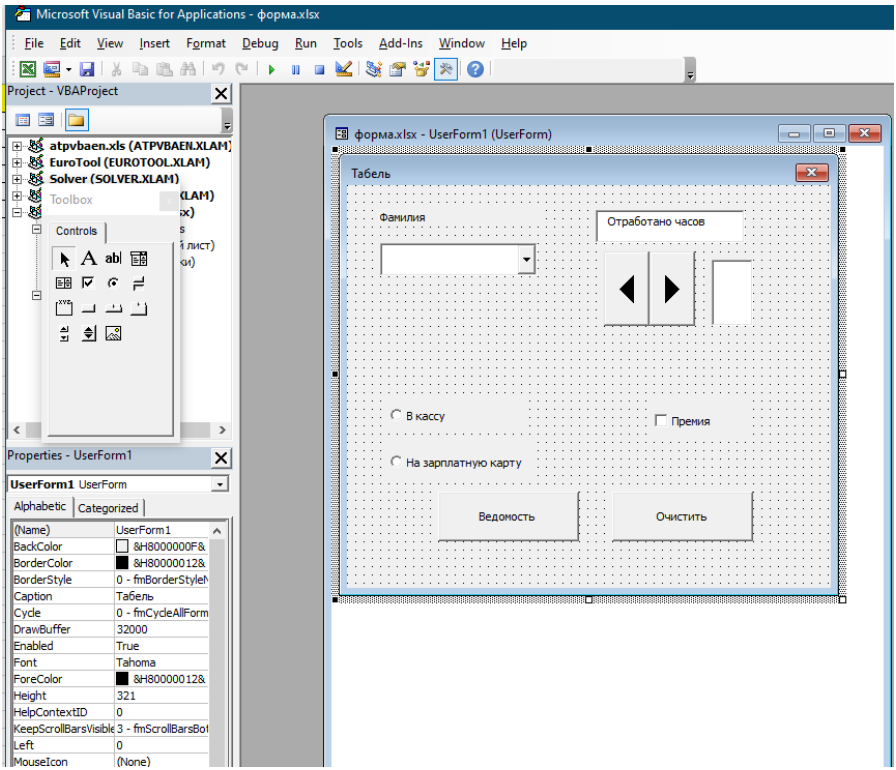


Рис. 5.17. Вид диалогового окна ввода данных для начисления заработной платы

Данная форма содержит такие элементы управления ActiveX, как:

- Label (надпись);
- CheckBox (флажок);
- ComboBox (раскрывающийся список);
- TextBox (поле);
- OptionButton (переключатель);
- CommandButton (кнопка).

Для создания **Раскрывающегося списка** в данном примере используется свойство элемента управления ComboBox, которое называется RowSourse, в котором вручную внесен диапазон ячеек, на базе которых и создан список: Сотрудники!A1:A10 (рис. 5.18).

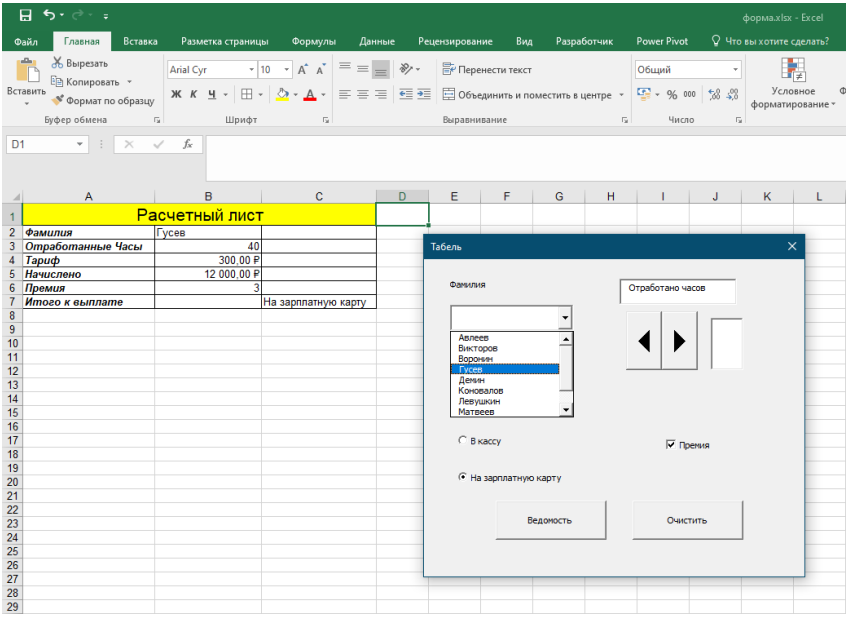


Рис. 5.18. **Форма в режиме ввода данных**

Ниже приведены VBA-программы обработки событий:

```
Private Sub CheckBox1_Click()
tarif = Sheets(«Расчетный лист»).Range(«B4»)
If CheckBox1.Value = True Then
Sheets(«Расчетный лист»).Range(«B6») = tarif * SpinButton1.Value * 0.1
Else
Sheets(«Расчетный лист»).Range(«B6») = 0
End If
End Sub

Private Sub ComboBox1_Change()
Sheets(«Расчетный лист»).Range(«B2») = ComboBox1.Value
End Sub

Private Sub CommandButton1_Click()
 CheckBox1.Value = False
Range(«B3»).Clear
```

```
TextBox1.Text = Clear
ComboBox1.Text = Clear
OptionButton1.Value = False
OptionButton2.Value = False
End Sub

Private Sub OptionButton1_Click()
If OptionButton1 = True Then
Sheets(«Расчетный лист»).Range(«C7») = «В кассу»
End If
End Sub

Private Sub OptionButton2_Click()
Sheets(«Расчетный лист»).Range(«C7») = «На зарплатную карту»
End Sub

Private Sub SpinButton1_Change()
TextBox1.Value = SpinButton1.Value
Sheets(«Расчетный лист»).Range(«B3») = TextBox1.Value
End Sub
```

Задания для выполнения лабораторной работы

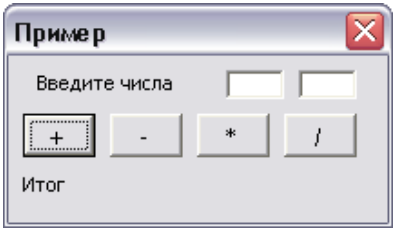
Для каждого задания разработайте отдельную форму кнопкой с рабочего листа.

1. Создайте форму – «хамелеон», реагирующую на 3-5 различных событий изменением цвета и текста заголовка.

2. Внесите в форму два поля ввода (фамилия и имя) и две кнопки, при нажатии на которые форма должна приветствовать или прощаться лично с тем, чье имя было введено.

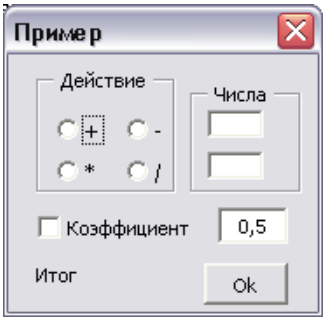
3. Создайте новую форму с двумя полями ввода чисел и четырьмя кнопками выполнения арифметических операций. Результат запишите в форму и ячейку Z1 текущего листа.

Учтите, что на ноль делить нельзя.



4. Выполните предыдущее задание, используя для выбора арифметических действий переключатели.

Учтите, что итоговое значение может быть скорректировано за счет коэффициента.

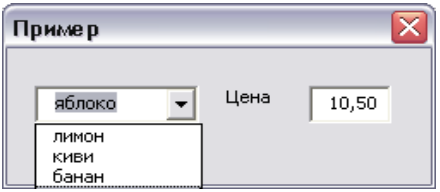


5. Создайте список из 10 наименований. При выборе в списке элемента выводите на форму соответствующую ему картинку или надпись, свидетельствующую об отсутствии картинки на эту тему.

6. На текущем листе Excel создайте таблицу вида (число строк – произвольное):

	A	B	C
1	Товар	Дата	Количество

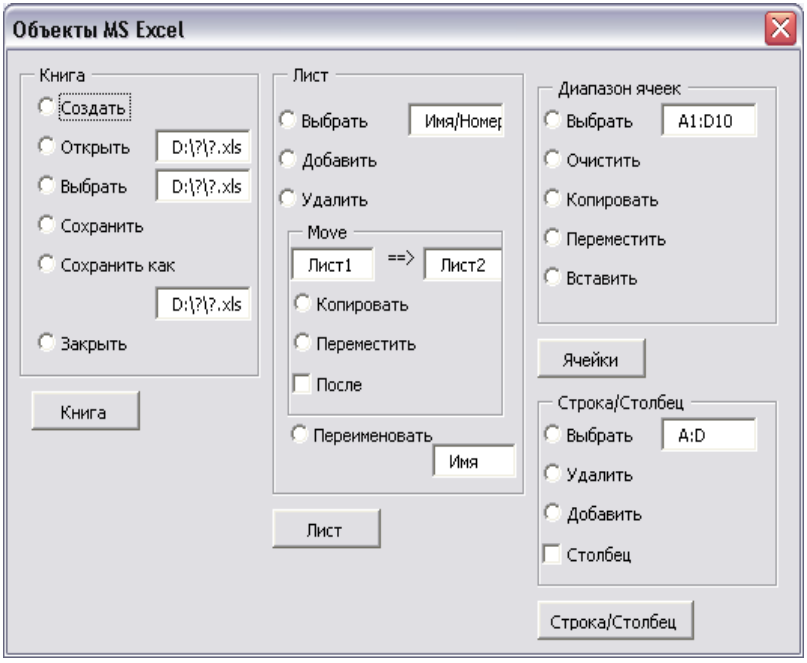
7. Создайте список неповторяющихся товаров через программный модуль. При выборе очередного товара в списке вводите в форме его цену и рассчитывайте стоимости продаж по таблице.



8. Создайте форму, позволяющую пользователю-неспециалисту в программировании использовать основные методы обработки объектов MS Excel в соответствии с данной формой-образцом. Оформление диалогового окна – на ваш вкус. Полнота охвата методов и объектов может меняться по заданию преподавателя.

9. Создайте форму ввода торговых заказов, которая содержит в себе следующие элементы:

- компания (TextBox);
- продавец (TextBox);
- количество (TextBox);
- ставка налога (TextBox);
- цена за единицу (TextBox);
- товар (ComboBox);
- облагается налогом (CheckBox).



На форме организуйте кнопки ВВОД и ВЫХОД.

На отдельном рабочем листе организуйте базу данных из вводимых значений, содержащую следующие поля: наименование товара,

фамилия продавца, количество заказываемых товаров, ставка налога, цена за единицу товара, наименование товара, общая сумма.

Если товар облагается налогом, то его величина рассчитывается так:

*Налог = Количество * Цена за единицу * Ставка налога / 100,*

в противном случае налог = 0.

Для вычисления общей суммы воспользуйтесь следующей формулой:

ВСЕГО = КОЛИЧЕСТВО*ЦЕНА + НАЛОГ.

10. Отчет о выполнении работы должен содержать листинги отлаженных программ.

Контрольные вопросы

1. Какие основные события пользовательской формы вы знаете?
2. Перечислите основные элементы управления.
3. Какой метод используется для вызова пользовательской формы?
4. Что понимается под обработкой событий элементов управления ActiveX?

5.3. Создание собственных функций рабочего листа

Возможность создания собственных новых функций значительно упрощает работу пользователя с формулами. Более компактные формулы легче воспринимаются, и с ними удобнее работать. Например, можно заменить сложную формулу одной функцией. Но главная причина создания собственных пользовательских функций – выполнение тех операций, которые другим способом выполнить невозможно.

Функция-процедура – это особый вид процедуры VBA, возвращающей результат. VBA-функции универсальны и могут вызываться в выражениях другой VBA-процедуры или в конкретных формулах рабочего листа. В первом случае аргументом может быть переменная, конкретное значение, значение, введенное с клавиатуры, или ссылка на ячейку. Если функция используется в рабочем листе, аргументом

может быть адрес ячейки (например, C1) или конкретное значение (например, 345).

Собственную функцию можно использовать везде, где используются другие функции рабочих листов Excel или встроенные VBA-функции. Созданные пользователем функции отображаются в диалоговом окне **Мастер функций**, в категории **Определенные пользователем** (рис. 5.19).

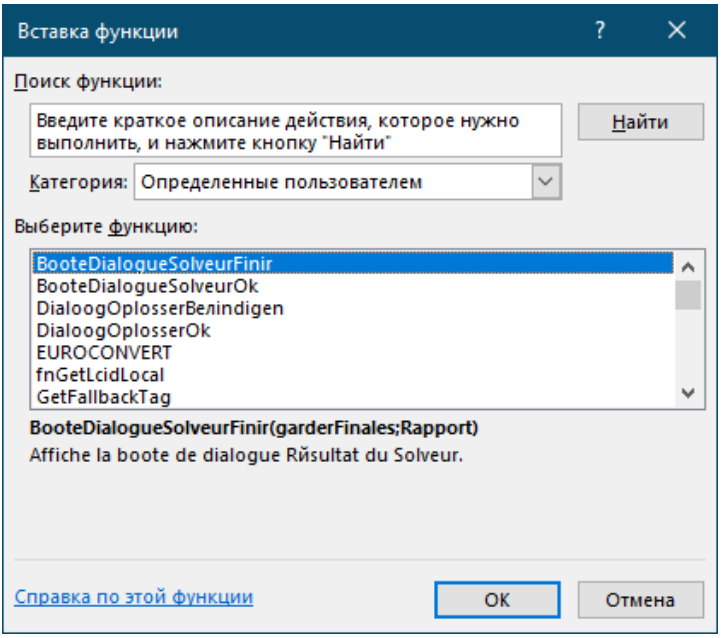


Рис. 5.19. Окно мастера функций

5.3.1. Синтаксис описания новой функции

Созданные пользователем процедуры-функции имеют много общего с процедурами-подпрограммами, которые рассмотрены ранее. Но при этом есть и важные особенности.

Процедуры-функции, создаваемые на языке VBA, начинаются с ключевого слова **Function**, за которым следуют имя самой функции и

перечень имен аргументов, заключенных в круглые скобки и разделенных символом «,».

Внимание! После окончания выполнения функции полученное значение присваивается имени функции.

Формат VBA-функции:

```
[Public| Private] [Static] Function Имя [(список аргументов)] [As Тип]
[операторы]
[Имя = выражение]
[Exit Function]
[операторы]
[Имя = выражение]
End Function
```

Указанные элементы определяются следующим образом (необязательные операторы заключены в квадратные скобки):

- параметр **Public** показывает, что функция доступна для любых других процедур, находящихся в любых других модулях рабочей книги;
- параметр **Private** указывает на то, что функция доступна только для процедур текущего модуля. Функции с описателем **Private** нельзя использовать в формулах рабочих листов, они не отображаются в диалоговом окне **Мастер функций**;
- параметр **Static** указывает на то, что значения переменных, которые объявлены в функции, сохраняются между последовательными вызовами данной функции;
- обязательное ключевое слово **Function** – начало процедуры-функции;
- параметр **Имя** требует задания любого допустимого имени функции. Имена функций должны удовлетворять тем же условиям, которые установлены для имен переменных. Нельзя использовать имя функции, напоминающее адрес ячейки, или зарезервированные VBA имена;
- параметр **список_аргументов** – это список переменных, которые представляют собой аргументы, передаваемые функции. Аргументы могут быть переменными, константами или выражениями.

VBA-функция может иметь фиксированное число обязательных аргументов (от 1 до 255);

- параметр Тип – это описание типа данных, возвращаемых функцией;
- операторы представляют собой любые допустимые операторы VBA: оператор Exit Function прекращает выполнение функции и передает управление вызываемой процедуре; оператор End Function указывает на окончание функции.

Пример создания новой функции. Создадим функцию с одним аргументом под именем OtrPol, которая возвращает текстовую строку Plus (Положительный), если ее аргумент больше 0, Minus (Отрицательный), если он меньше 0, и Zero (Ноль), если он равен 0.

Для создания новой функции следует выполнить следующие действия:

1. Войти в редактор VBA.

2. Выбрать необходимую рабочую книгу.

3. Выполнить команду **Insert – Module** для добавления нового модуля или использовать имеющийся модуль.

4. Задать ключевое слово Function, имя функции и список аргументов (если они есть), заключив в круглые скобки.

5. Ввести программу на языке VBA, выполняющую заданный алгоритм действий:

```
Function OtrPol(newVal)
Select Case newVal
Case Is < 0: OtrPol g = «Отрицательный»
Case Is = 0: OtrPol = «Ноль»
Case Is > 0: OtrPol = «Положительный»
End Select
End Function
```

6. Закончить функцию оператором End Function.

Данная VBA-функция начинается с ключевого слова Function, за которым следует имя самой функции OtrPol. Эта функция имеет только один аргумент (newVal), имя аргумента заключено в круглые скобки. Вместо newVal при вызове функции подставляется либо ссылка на ячейку, либо конкретное значение, либо переменная. Если функция

используется в рабочем листе, аргументом может быть адрес ячейки (например, C8) или конкретное значение (например, -78). Если функция используется в другой процедуре, то аргументом может быть переменная, фиксированное значение или значение, полученное из ячейки.

В приведенной функции используется конструкция Select Case, с помощью которой осуществляются анализ входного значения newVal и выбор возвращаемого значения. Если newVal меньше 0, то функции OtrPol присваивается текстовая строка «Отрицательный», если newVal равно 0, то функции OtrPol присваивается текстовая строка «Нуль», а если newVal больше 0, то функции OtrPol присваивается текстовая строка «Положительный». Значение, возвращаемое функцией, всегда присваивается имени функции. Процедура заканчивается оператором End Function.

5.3.2. Добавление новых функций в категорию Определенные пользователем

С помощью диалогового окна **Мастер функций** Excel можно выбрать функции рабочих листов, которые созданы пользователем, кроме этого, можно отобразить описание созданной VBA-функции.

Для этого нужно выполнить следующие действия:

1. Активизировать Excel.

2. Выбрать команду **Вид – Макросы – Макросы**. Появится диалоговое окно **Макрос**.

3. В поле **Имя макроса** диалогового окна **Макрос** ввести имя новой функции (обращаем внимание на то, что обычно в этом окне функции не отображаются, поэтому ввод имени функции осуществляется самостоятельно).

4. Щелкнуть на кнопке **Параметры**. Появится диалоговое окно **Параметры макроса** (рис. 5.20).

5. Ввести описание функции и щелкнуть на кнопке **ОК**. Поле **Сочетание клавиш** не заполняется.

Описание функции отобразится в диалоговом окне **Мастер функций**. Созданные функции попадут в категорию **Определенные пользователем**.

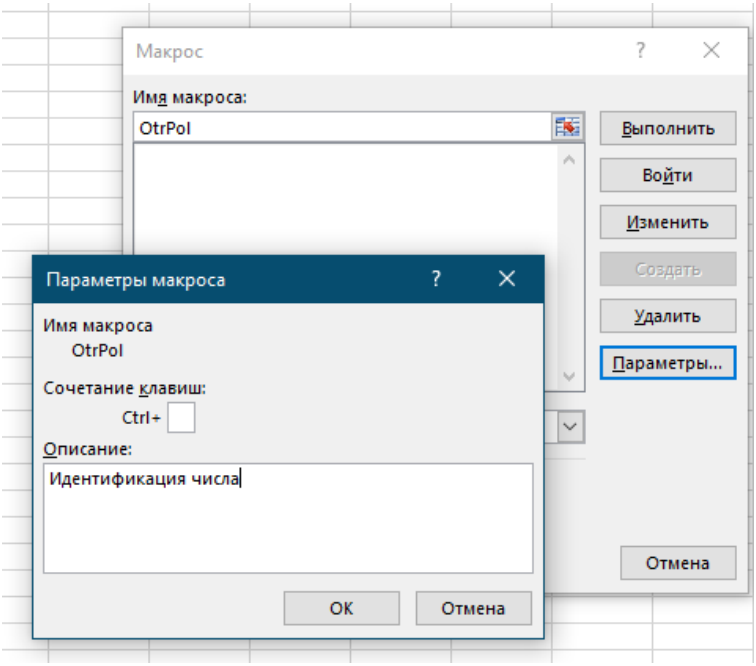


Рис. 5.20. Окно **Параметры макроса**

Вызов пользовательской функции в рабочем листе. Функция работает аналогично любым другим встроенным функциям рабочих листов. Чтобы вставить ее в формулу, нужно воспользоваться командой **Вставка – Функция**, открыв диалоговое окно **Мастер функций**. Новые функции находятся в категории **Определенные пользователем**. Можно также выполнять вложение новых функций и комбинировать их с другими элементами в формулах.

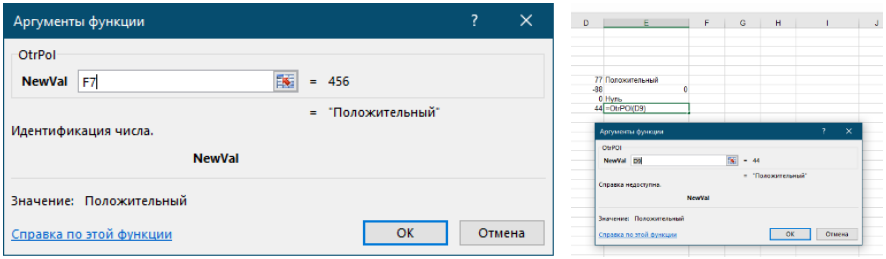


Рис. 5.21. Созданная пользовательская функция в режиме работы

Использование созданной функции в рабочем листе аналогично использованию встроенных функций (см. рис. 5.21). Но следует быть уверенным в том, что Excel сможет найти процедуру-функцию. Если она находится в той же рабочей книге, не нужно предпринимать никаких специальных действий. Если же функция определена в другой рабочей книге, следует сообщить Excel, где ее искать.

Чтобы воспользоваться функцией OtrPol, которая определена в текущей рабочей книге, пишется:

```
=OtrPol(B4)
```

Если функция была помещена в рабочий лист с помощью диалогового окна **Мастер функций**, то ссылка на рабочую книгу будет вставлена автоматически.

Если работа ведется с функцией другой рабочей книги, необходимо указать ссылку на рабочую книгу

```
=MyBook!OtrPol(B4)
```

5.3.3. Использование VBA-функции в VBA-подпрограмме

В приведенной ниже процедуре-подпрограмме, написанной на языке VBA, которая определена в том же модуле, что и новая функция OtrPol, используется встроенная функция MsgBox. С ее помощью отображается результат вычисления функции OtrPol:

```
Sub ViewOtrPol()
Ce11Va1ue = Sheets(«Значения»).Range(«B4») .Value
MsgBox OtrPol (CellValue)
End Sub
```

В этом примере переменной CellValue присваивается значение ячейки B4 листа **Значения**. Затем CellValue используется в качестве аргумента функции. На рис. 5.22 показан текст программы в редакторе кода ViewOtrPol.

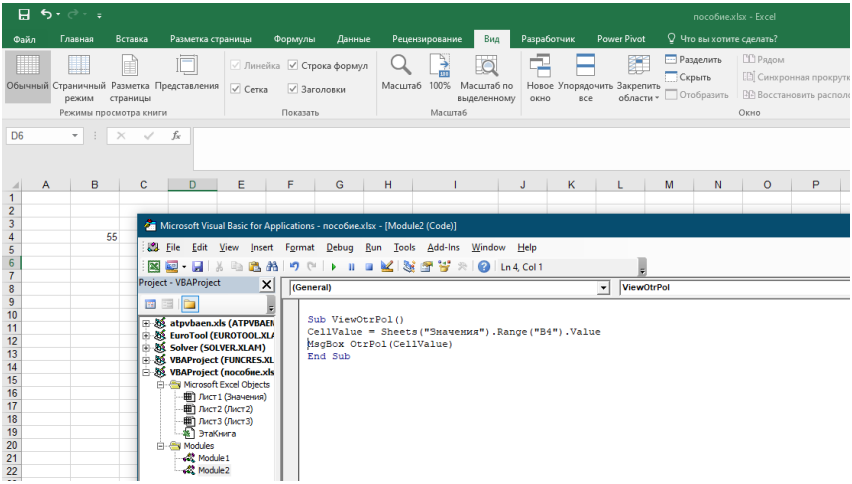


Рис. 5.22. Текст программы в редакторе кода

VBA-функция без аргументов. Как и подпрограммы, функции необязательно должны использовать аргументы. В Excel, например, есть несколько встроенных функций рабочих листов, не имеющих аргументов. К этим функциям относится хорошо знакомая вам функция СЕГОДНЯ().

Приведем пример функции, не имеющей аргументов. Она возвращает свойство UserName (Имя пользователя) объекта **Application**. Это имя, которое появляется во вкладке **Общие** диалогового окна **Параметры**. Этот пример создания пользовательской функции очень актуален, так как не существует другого способа вставить имя пользователя в формулу рабочего листа:

```
Function MyUser()
MyUser = Application.UserName
End Function
```

Если ввести в ячейку рабочего листа приведенную ниже формулу, в этой ячейке отобразится имя текущего пользователя:

```
=User()
```

Как и в случае встроенных функций Excel, при использовании функций без аргументов нужно добавлять круглые скобки.

Ниже приведен пример простой подпрограммы, в которой созданная пользователем функция User используется в качестве аргумента

функции MsgBox. Оператор амперсанд & объединяет строку текста с результатом функции User.

```
Sub ViewUser()
MsgBox («Пользователь» & MyUser())
End Sub
```

VBA-функция с одним аргументом. Ниже приведен пример более сложной функции, необходимой менеджеру по продажам, который рассчитывает свои комиссионные. Процент комиссионных зависит от объема проданного товара. Приведенная ниже функция возвращает величину размера комиссионных, который зависит от объема продажи. Объем продаж – это единственный и обязательный аргумент приведенной функции.

Вычисления, выполняемые в этой функции, основаны на данных, указанных в табл. 5.4.

Таблица 5.4. Данные для решения задачи

Объем продаж	Комиссионные, $
0–9999	6
10000–19000	8
20000–39000	10
Более 40000	12

```
Function Procent(kol)
rate1 = 0.06
rate2 = 0.08
rate3 = 0.1
rate4 = 0.12
Select Case kol
Case 0 To 9999.99: Procent = kol * rate1
Case 10000 To 19999.99: Procent = kol * rate2
Case 20000 To 39999.99: Procent = kol * rate3
Case Is >= 40000: Procent = kol * rate4
End Select
End Function

= Procent (45000)
```

Даже если у вас нет необходимости создавать новые функции рабочего листа, все равно процедуры-функции намного проще VBA-макросов. Например, если в VBA-процедуре вычисляется размер комиссионных, можно воспользоваться той же самой функцией и вызвать ее из VBA-подпрограммы. Вот пример маленькой подпрограммы, которая запрашивает у пользователя объем продаж, а затем использует функцию Procent для вычисления размера комиссионных и отображает полученное значение на экране:

```
Sub CalProcent()
Kol = InputBox(«Введите объем продаж»)
MsgBox «Размер комиссионных составляет» & Procent (Kol)
End Sub
```

Программа начинается с того, что на экране отображается окно ввода, в котором запрашивается объем продаж. Процедура отображает окно сообщения с вычисленным размером комиссионных, который соответствует заданному объему продаж. Функция Procent должна находиться в активной рабочей книге, в противном случае Excel выдаст сообщение о том, что функция не определена.

VBA-функция с двумя аргументами. Новая функция создается на основе предыдущей. Известно: менеджер по продажам ввел новые условия расчета комиссионных, которые увеличиваются на 1% за каждый год работы продавца в данной компании. Изменим функцию Procent, приведенную ранее, таким образом, чтобы у нее было два аргумента.

Эта функция будет иметь имя Procentnew:

```
Function ProcentNew(Kol, Period)
rate1 = 0.06
rate2 = 0.08
rate3 = 0.1
rate4 = 0.12
Select Case Kol
Case 0 To 9999.99: ProcentNew = Kol * rate1
Case 10000 To 19999.99: ProcentNew = Kol * rate2
Case 20000 To 39999.99: ProcentNew = Kol * rate3
Case Is >= 40000: ProcentNew = Kol * rate4
```

```
End Select
ProcentNew = ProcentNew + (ProcentNew * Period / 100)
End Function
```

Добавим второй аргумент Period к функции Procentnew (в операторе **Function**) и перед выходом из функции включим дополнительного оператора для расчета окончательного значения размера комиссионных.

Ниже пример того, как написать формулу с использованием данной функции. Величина продаж находится в ячейке C1, а количество проработанных продавцом лет – в ячейке C2.

```
= Procentnew (C1;C2).
```

VBA-функция с аргументами-диапазонами. Этот пример очень важен, так как демонстрирует использование диапазона ячеек рабочего листа в качестве аргумента функции.

Предположим, нужно вычислить среднее из десяти наибольших величин из диапазона Diapoz. В Excel нет функции для выполнения такого расчета, поэтому нужно написать следующую формулу:

```
=(НАИБОЛЬШИЙ (Diapoz;1) +НАИБОЛЬШИЙ (Diapoz; 2) +НАИБОЛЬШИЙ (Diapoz; 3)... +НАИБОЛЬШИЙ (DATA; 10) +НАИБОЛЬШИЙ (Diapoz;10) ) /10
```

В этой формуле используется встроенная функция Excel НАИБОЛЬШИЙ, которая возвращает *n*-е (*n* = 1, ..., 10) наибольшее число диапазона. Далее десять наибольших чисел диапазона Data складываются и результат делится на 10. Формула прекрасно работает, но она все-таки довольно громоздкая и не универсальна.

Все значительно упрощается, если создается пользовательская функция TopAvg. Тогда для вычисления можно ввести формулу:

```
=TopAvg(Diapoz;10)
```

Код новой VBA-функции, которая возвращает среднее для *n* наибольших величин диапазона, выглядит следующим образом:

```
Function TopSred(NewRange, N)
Summ = 0
For j = 1 To N
Summ = WorksheetFunction.Large(NewRange, j)
Next j
```

```
TopSred = Summ / N
End Function
```

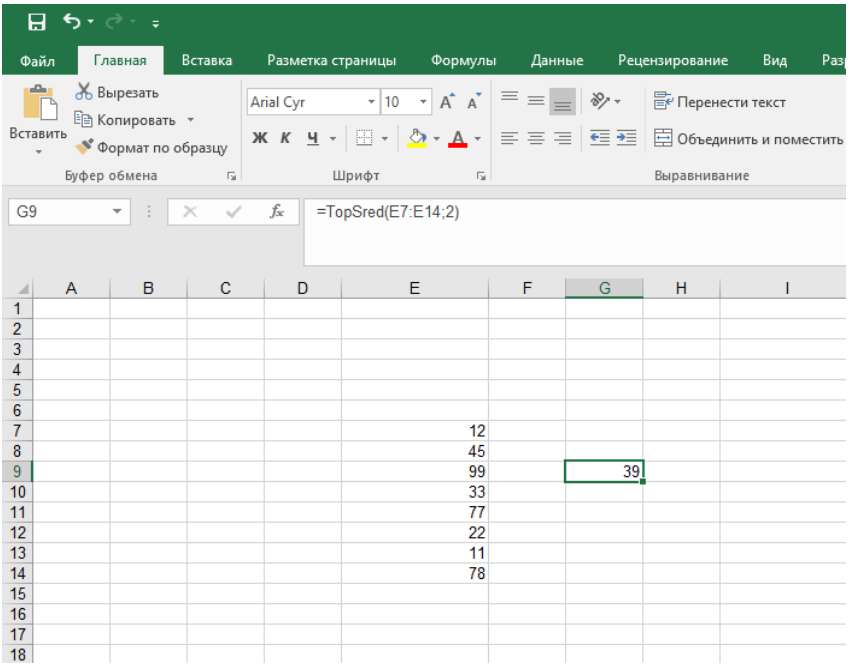


Рис. 5.23. **Результат работы функции**

У этой функции есть два аргумента: InRange (диапазон рабочего листа) и N (количество величин, для которых вычисляется среднее).

В теле цикла использована функция Large (Наибольший) программы Excel. Ее можно использовать, если вставить перед ее именем объект WorksheetFunction и точку.

Если вы не знаете, как записывается функция на английском языке, выполните макрос с содержанием этой функции (см. рис. 5.23).

5.3.4. Отладка функций, созданных пользователем

Отладить процедуру-функцию немного сложнее, чем процедуру-подпрограмму. Если вы создали функцию для использования в формулах рабочих листов, то скоро обнаружите, что ошибка в этой функции

приведет к появлению сообщения об ошибке в ячейке формулы. Другими словами, вы не получите сообщения об ошибке, возникшей во время выполнения функции, с помощью которого можно обнаружить оператор с ошибкой. Ниже приведены методы, которыми вы можете воспользоваться для поиска ошибок.

Поместите в наиболее важных местах программы функцию MsgBox, чтобы получить возможность проверить значения некоторых переменных. Помните, что окна сообщений в процедурах-функциях появляются во время выполнения процедуры. Целесообразно сделать так, чтобы в рабочем листе была только одна формула, в которой используется функция, иначе окна сообщений будут появляться для каждой вычисляемой формулы.

Протестируйте созданную функцию, вызвав ее из процедуры-подпрограммы. Для этого необходимо вызвать функцию в режиме работы. Если при ее выполнении произойдет ошибка, то редактор VBA выдаст соответствующее ошибке сообщение. После ее исправления функцию необходимо запустить повторно, чтобы убедиться, что она работает правильно.

Задание для выполнения лабораторной работы

Напишите пользовательскую функцию, которая:

- создает инициалы из имеющихся данных в отдельных графах: фамилии, имени и отчестве;
- переводит число в текстовое значение;
- рассчитывает стаж сотрудника. Исходные данные – дата приема на работу и текущая дата.

Напишите отчет о ходе выполнения работы.

Контрольные вопросы

1. В какой категории размещается функция, созданная пользователем?
2. Сколько способов вызова VBA-функции вы знаете? Назовите их.
3. Каким символом разделены аргументы VBA-функции?
4. Чему присваивается результат выполнения VBA-функции?
5. С какой целью создаются пользовательские функции?

Список литературы

1. Андронов, А. Самоучитель по Excel VBA / А. Андронов. – Текст : электронный // ОфисГуру : журнал новых технологий : [сайт]. – URL: https://office-guru.ru/excel/samouchitel-po-excel-vba-453.html (опубликовано: 11.05.2020).

2. Гаврилов, М.В. Информатика и информационные технологии : учебник для вузов / М.В. Гаврилов, В.А. Климов. – 4-е изд., перераб. и доп. – Москва : Юрайт, 2021. – 383 с. – (Высшее образование). – ISBN 978-5-534-00814-2. – Текст : электронный // Юрайт : образовательная платформа для университетов и колледжей : [сайт]. – URL: https://urait.ru/bcode/468473 (дата обращения: 11.04.2022).

3. Гниденко, И.Г. Технологии и методы программирования : учебное пособие для вузов / И.Г. Гниденко, Ф.Ф. Павлов, Д.Ю. Федоров. – Москва : Юрайт, 2021. – 235 с. – (Высшее образование). – ISBN 978-5-534-02816-4. – Текст : электронный // Юрайт : образовательная платформа для университетов и колледжей : [сайт]. – URL: https://urait.ru/bcode/469759 (дата обращения: 23.04.2022).

4. Зыков, С.В. Программирование : учебник и практикум для вузов / С.В. Зыков. – Москва : Юрайт, 2021. – 320 с. – (Высшее образование). – ISBN 978-5-534-02444-9. – Текст : электронный // Юрайт : образовательная платформа для университетов и колледжей : [сайт]. – URL: https://urait.ru/bcode/469579 (дата обращения: 23.04.2022).

5. Зыков, С.В. Программирование. Объектно-ориентированный подход : учебник и практикум для вузов / С.В. Зыков. – Москва : Юрайт, 2021. – 155 с. – (Высшее образование). – ISBN 978-5-534-00850-0. – Текст : электронный // Юрайт : образовательная платформа для университетов и колледжей : [сайт]. – URL: https://urait.ru/bcode/470281 (дата обращения: 15.03.2022).

6. Зыков, С.В. Программирование. Функциональный подход : учебник и практикум для вузов / С.В. Зыков. – Москва : Юрайт, 2021. – 164 с. – (Высшее образование). – ISBN 978-5-534-00844-9. – Текст : электронный // Юрайт : образовательная платформа для университетов и

колледжей : [сайт]. – URL: https://urait.ru/bcode/470387 (дата обращения: 02.03.2022).

7. Информатика для экономистов : учебник для вузов / В.П. Поляков [и др.] ; под редакцией В.П. Полякова. – Москва : Юрайт, 2021. – 524 с. – (Высшее образование). – ISBN 978-5-534-11211-5. – Текст : электронный // Юрайт : образовательная платформа для университетов и колледжей : [сайт]. – URL: https://urait.ru/bcode/468654 (дата обращения: 28.03.2022).

8. Казанский, А.А. Прикладное программирование на Excel 2019 : учебное пособие для вузов / А.А. Казанский. – 2-е изд., перераб. и доп. – Москва : Юрайт, 2021. – 171 с. – (Высшее образование). – ISBN 978-5-534-12022-6. – Текст : электронный // Юрайт : образовательная платформа для университетов и колледжей : [сайт]. – URL: https://urait.ru/bcode/470200 (дата обращения: 01.04.2022).

9. Лебедев, В.М. Программирование на VBA в MS Excel : учебное пособие для вузов / В.М. Лебедев. – 2-е изд., испр. и доп. – Москва : Юрайт, 2020. – 306 с. – (Высшее образование). – ISBN 978-5-534-12231-2. – Текст : электронный // Юрайт : образовательная платформа для университетов и колледжей : [сайт]. – URL: https://urait.ru/bcode/447096 (дата обращения: 11.05.2022).

10. Справочник по языку Visual Basic (VBA) для приложений // Майкрософт – облачные технологии, приложения и игры : [офиц. сайт]. – URL: https://docs.microsoft.com/ru-ru/office/vba/api/overview/language-reference (дата обращения: 03.04.2022). – Текст : электронный.

11. Трофимов, В.В. Алгоритмизация и программирование : учебник для вузов / В.В. Трофимов, Т.А. Павловская ; под редакцией В.В. Трофимова. – Москва : Юрайт, 2021. – 137 с. – (Высшее образование). – ISBN 978-5-534-07834-3. – Текст : электронный // Юрайт : образовательная платформа для университетов и колледжей : [сайт]. – URL: https://urait.ru/bcode/471125 (дата обращения: 08.04.2022).

Учебное издание

Сакова Татьяна Германовна
Юдина Ольга Владимировна

Разработка бизнес-приложений в среде MS Excel средствами VBA

Учебное пособие

ISBN 978-5-00176-124-2

Издательская группа: О.В. Егорова, И.А. Барханская

Подписано к изданию 06.06.2022. Печ. л. 5,25.
ФГАОУ ВО «Самарский государственный экономический университет».
443090, Самара, ул. Советской Армии, 141.