ФГБОУ ДОПОЛНИТЕЛЬНОГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

АКАДЕМИЯ МЕДИАИНДУСТРИИ

Л.И. Галкин

Программирование для Excel

Создание макросов и процедур

Учебное пособие

MS Excel 2010 и MS Excel 2013

Москва 2015

Введение

Учебное пособие рассчитано на пользователей приложения[1] Microsoft Excel, ошибочно полагающих, что программирование «не их ума дело». Предполагается, что читатель знаком с основными компонентами этой программы, освоил работу с формулами и диапазонами ячеек, использует в своей практической деятельности встроенные функции, средства визуализации и анализа данных.

Поскольку все инструменты Excel спроектированы так, чтобы ими было максимально удобно пользоваться, многие начинают работать с программой, затратив совсем немного времени на знакомство с ней. **Практически каждый способен поместить данные на рабочий лист и, выполнив несложные расчеты, как-то оформить рабочий лист книги.** При этом создается устойчивое иллюзорное представление о том, что именно в этом и заключается назначение и все функциональные возможности этой офисной программы.

В действительности — это нечто большее, чем просто программа для работы с таблицами данных. С девяностых годов прошлого столетия все версии приложения Excel выпускаются со встроенным макрорекордером, мощным объектно-ориентированным языком программирования Visual Basic for Applications[2] (VBA) и редактором VB.

Принцип действия макрорекордера схож с принципом действия магнитофона. Включив запись, вы с помощью мыши и клавиатуры начинаете задавать последовательность команд, а макрорекордер протоколирует, преобразует и записывает их в виде инструкций на полноценном языке программирования VBA. Эту именованную запись, называемую макросом, вы в любой момент можете воспроизвести, повторив тем самым ту же последовательность действий.

Что такое макрос? Это макрокоманда, интерпретирующая последовательность действий пользователя, записанных встроенным в приложение устройством с помощью языка VBA. Из-за внешнего сходства макроса и написанной вручную процедуры-подпрограммы некоторые

---

[1] Компьютерные программы, работающие под управлением операционной системы windows. принято называть приложениями (applications)

[2] VBA – язык Visual Basic для конкретного приложения пакета MS Office System.

авторы полагают[1], что это – всего лишь разные названия мини-программы на языке VBA, запускаемой из того приложения, где она были создана.

Пользователи приложений пакета MS Office System, как правило, вспоминают о макросах и процедурах тогда, когда им надоедает вручную выполнять одни и те же последовательности рутинных операций или когда сталкиваются со специфической проблемой, для решения которой недостаточно заложенных в него функциональных возможностей.

Поскольку повторяющихся действия в любом производственном процессе всегда присутствуют, автоматизация рутинных операций заметно повышает эффективность работы пользователя с приложением. Важно, что, хотя для автоматизации не требуется ни навыки программирования, ни знания языка VBA, практически все освоившие работу с макрорекордером на этом не останавливаются. Посмотрев на полученный результат и внеся некоторые коррективы в логически понятные инструкции созданного макроса, большинство приходит к выводу, что не боги горшки обжигают.

Опыт показывает, что, как только появляется интерес и понимание того, какую пользу на его рабочем месте он может извлечь, если освоит упомянутые навыки, слушатель быстро их усваивают.

Пособие разделено на три главы.

Глава1 «Создание и редактирование макроса» начинается с краткого обзора мероприятий по подготовке к записи макроса (активизации вкладки «Разработчик», выбора параметра макросов в центре управления безопасностью и формата файлов с поддержкой макросов). Описывается последовательность действий пользователя при записи и выполнении макроса. Показывается влияние установленного при записи макроса режима абсолютных или относительных ссылок на его работу. В процессе обсуждения анатомии созданного макроса рассмотрена объектная модель Excel. На конкретном примере продемонстрированы действия пользователя по упрощению программного кода макроса и созданию на его основе нового макроса.

В Главе 2 «Синтаксис и программные конструкции VBA» на конкретных примерах рассматриваются арифметические и логические операторы, типы данных, операторы, позволяющие управлять ходом выполнения процедуры (операторы условного и безусловного перехода,

[1] Они не учитывают тот факт, что процедуры более гибки и возможности их значительно шире, чем у макросов

операторы циклов с условием и со счетчиком). Работа с массивами данных проиллюстрирована в упражнении, в котором решается задача преобразования денежных сумм в текст. С подобными задачами сталкиваются пользователи, автоматизирующие процесс заполнения стандартных бланков.

Глава 3 «Создание пользовательских форм и диалоговых окон» знакомит с элементами управления формы и элементами ActiveX. Приведено описание их свойств и методов. В трех упражнениях детально рассмотрены вопросы, связанные с применением этих элементов управления. В первом упражнении приведено решение задачи автоматизации процесса заполнения бланка платежного поручения. Нестандартная форма на рабочем листе создается во втором упражнении с целью облегчения нахождения приемлемой для семейного бюджета стоимости недвижимости. И наконец в последнем упражнении предлагается создать нестандартное диалоговое окно «Выплаты по ссуде». удобное для сравнения сумм выплат производимых заемщиком ежемесячно или ежегодно.

Современные версии программы (MS Excel 2010 и MS Excel 2013), объединяющие в себе мощные аналитические средства электронных таблиц и широкие возможности визуального программирования, помогают среднестатистическому пользователю не только автоматизировать выполнение рутинных операций, но и решать специализированные задачи по обработке и анализу данных.

Глава 1. Создание макросов в MS Excel 2010

Пользователи MS Excel, как правило, вспоминают о макросах и процедурах тогда, когда им надоедает вручную выполнять одни и те же последовательности рутинных операций или когда сталкиваются со специфической проблемой, для решения которой недостаточно заложенных в это приложение функциональных возможностей.

1.1 Подготовка к записи и процесс записи макроса

Если с вами, уважаемый читатель, случилась подобная история, то первым делом разместите на ленте вкладку **Разработчик**, которая по умолчанию там отсутствует. Полагаю, что причина тому не перегруженность ленты, а то, чем довольствуется среднестатистический пользователь, применяющий, как правило, незначительную часть средств и инструментов этой программы, находящихся в открытом доступе.

Настроив ленту (см. рис 1.1), отключите блокировку макросов в Центре управления безопасностью, выбрав подходящее для конкретной ситуации значение параметра безопасности.

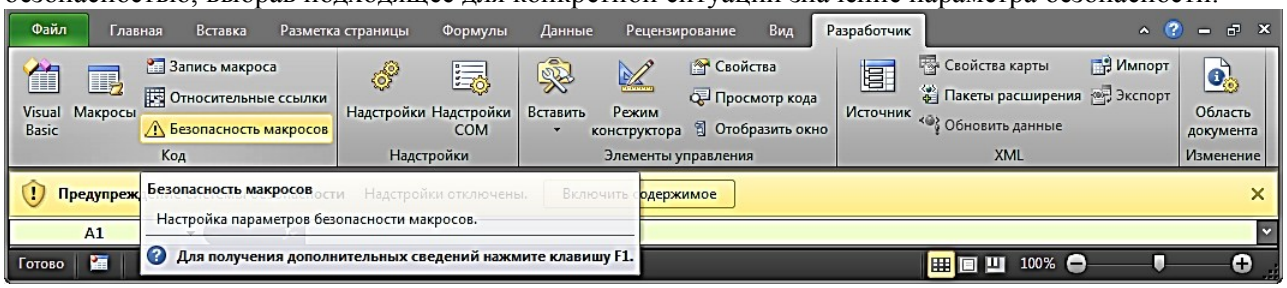


Рис. 1.1

Значения параметра безопасности макросов:

- **Отключить все макросы без уведомления.** Если пользователь не доверяет никаким макросам, он, выбрав это значение параметра, одновременно отключит все макросы и связанные с ними окна оповещения.
- **Отключить все макросы с уведомлением.** Это установленное по умолчанию значение позволит вам включать только те макросы, которым вы полностью доверяете.
- **Отключить все макросы, кроме макросов с цифровой подписью.** Данное значение идентично предыдущему, но если макрос имеет цифровую подпись, и создатель его внесен пользователем в список надежных издателей, то такой макрос доступен для запуска.
- **Включить все макросы (не рекомендуется, возможен запуск опасной программы).** При этом значении выполняются любые макросы, а компьютер становится временно уязвимым для потенциально опасных макросов. Почему временно? Потому что, создавая и используя собственные макросы, вы ничем не рискуете, а когда завершите эту работу, обязательно отключите опасную опцию. Верните установленное по умолчанию значение параметра безопасности и

только после этого приступайте к поиску нужной информации в сети Интернет.

И наконец тщательно в деталях продумайте последовательность своих действий с обязательным уточнением, по каким кнопкам будете щелкать и на какие клавиши нажимать. Помните! Макрорекордер фиксирует и записывает все действия пользователя[1].

УПРАЖНЕНИЕ 1.

1. Откройте новую рабочую книгу и сразу сохраните её в формате «Книга Excel с поддержкой макросов (*.xlsm)».

2. Если на ленте отсутствует вкладка «Разработчик», задайте команду: ***Файл*** ⇒ ***Параметры*** ⇒ ***Настройка ленты.***

3. В списке **Основные вкладки** диалогового окна **Параметры Excel** активизируйте вкладку «Разработчик» (см. рис. 1.2) и щелкните по кн. **ОК**.

4. На ленте появится вкладка «Разработчик» (см. рис. 1.3), где необходимые при работе с макросами команды размещены в группе «Код»**.**

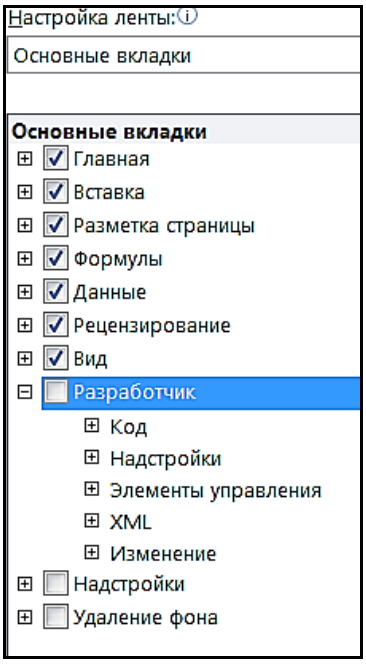


Рис. 1.2

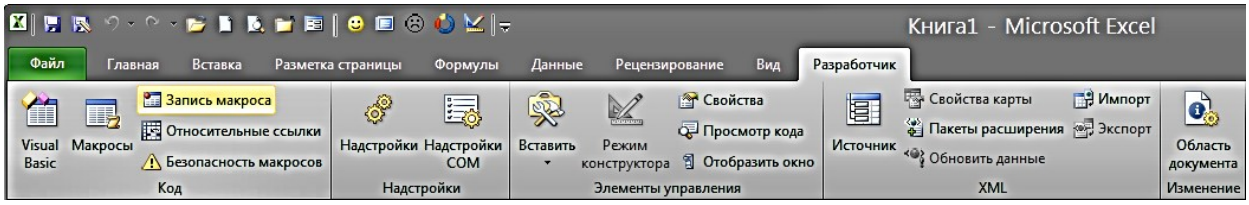


Рис. 1.3

5. Задайте команду: ***Разработчик*** ⇒ ***Код*** ⇒ ***Безопасность макросов***.

6. В диалоговом окне **Центр управления безопасностью**, задав уровень безопасности, временно разрешающий выполнение всех макросов (см. рис. 1.4), щелкните по кнопке **ОК**.

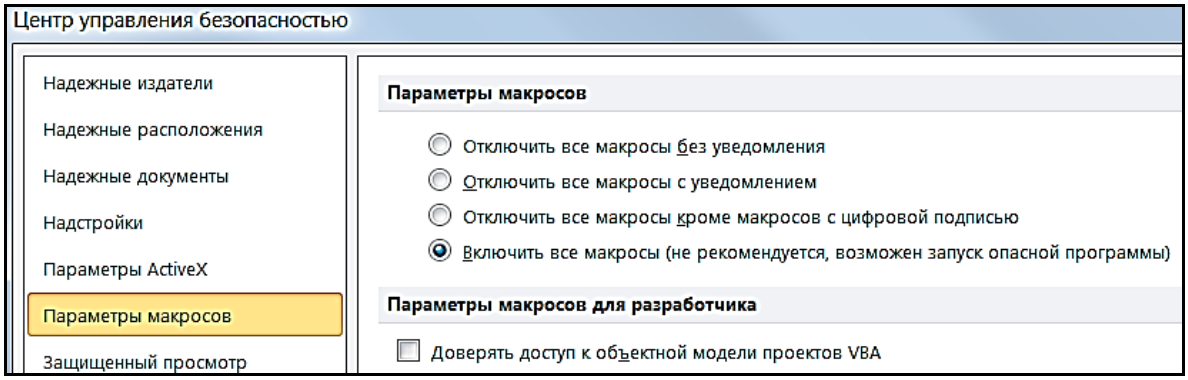


Рис. 1.4

[1] Выделение диапазона, прокрутка рабочего листа, переход на другой лист и т.д.

NB! Для предотвращения запуска потенциально опасных программ по завершению работы с макросами обязательно активизируйте опцию «Отключить все макросы с уведомлением».

7. Продумайте, что пошагово вы собираетесь сделать и, главное, в какой последовательности.

УПРАЖНЕНИЕ 2.

1. Задайте команду: ***Разработчик ⇒ Код⇒ Запись макроса*** или в статусной строке щелкните по кнопке **Запись макроса** (см. рис. 1.5).
2. В поле **Имя макроса** диалогового окна **Запись макроса** (см. рис. 1.6) напечатайте имя «Текст_Пр1».

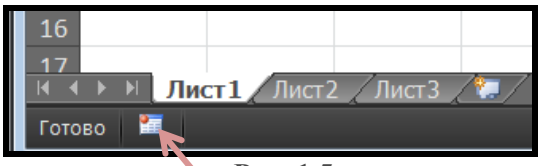


Рис. 1.5

NB! Первым символом имени макроса должна быть буква. Последующие символы могут быть буквами, цифрами или символами подчеркивания.

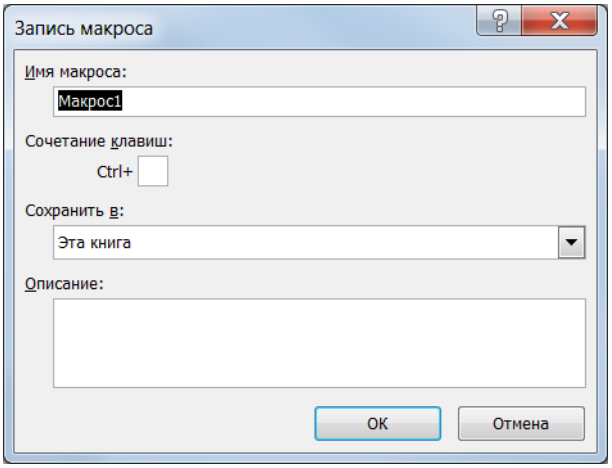


Рис. 1.6

Поскольку в имени макроса не допускаются пробелы, в качестве разделителей слов используйте символы подчеркивания.

3. В поле **Сочетание клавиш,** если предпочитаете работать с «горячими» клавишами, введите любую строчную или прописную букву русского[1] алфавита (например, строчную букву **я**).
4. В списке **Сохранить в** выберите место, например, *Эта книга*, где хотите сохранить макрос.

NB! При выборе варианта «*Личная книга макросов*» создается[2] скрытая книга макросов (Personal.xlsb). Туда и будет помещен ваш макрос.

5. В поле **Описание** напечатайте комментарий к создаваемому макросу (например, текстовая прогрессия в режиме абсолютной адресации).
6. Восстановив в памяти последовательность операций, которые вы задумали автоматизировать, щелкните по кнопке **ОК.**
7. В ячейке A1 напечатайте слово *Январь* и нажмите клавишу Tab..
8. Выделив ячейку A1, попадите курсором мыши в маркер автозаполнения и протяните его до ячейки L1 (включительно).
9. Задав команду: ***Главная ⇒ Выравнивание ⇒ Кнопка запуска диалогового окна,*** в диалоговом окне **Формат ячеек** активизируйте

---

[1] Буквы латинского алфавита лучше не использовать, поскольку заданное сочетание клавиш заменит совпадающие с ним стандартное сочетание клавиш MS Excel на всё то время, пока книга с этим макросом открыта.

[2] Если ваша персональная рабочая книга всё ещё не создана.

вкладку «Выравнивание» и выберите параметру **Выравнивание по горизонтали** значение «По центру».

10. Активизировав вкладку «Шрифт», задайте параметру **Начертание** значение «Полужирный» и нажмите клавишу **ОК**.

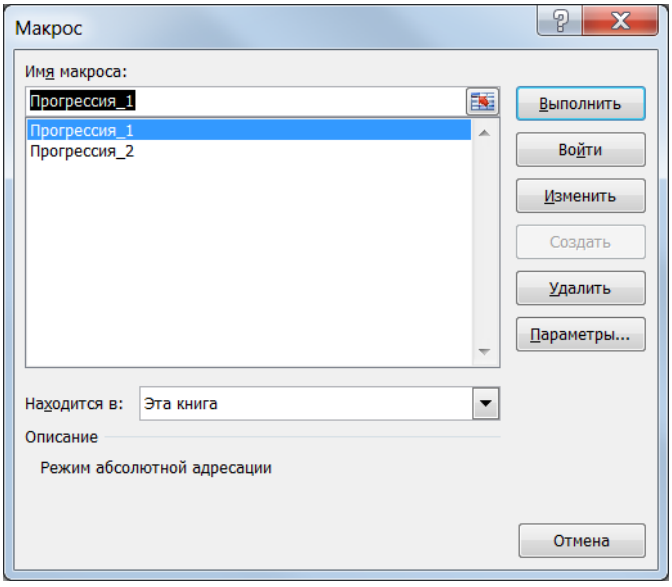
11. Закрыв диалоговое окно, отмените выделение диапазона A1:L1.

12. Задайте команду[1]: ***Разработчик ⇒ Код ⇒ Остановить запись*** или в статусной строке щелкните по кнопке **Остановить запись.**

В момент остановки записи макрорекордер, записывающий последовательность действий пользователя, сгенерирует макрокоманду, оформленную в виде подпрограммы на языке VBA (Visual Basic для Excel) и назначит ей имя «Текст_Пр1».

УПРАЖНЕНИЕ 3.

1. Выделив диапазон A1:L1, отмените результаты выполненных в предыдущем упражнении действий на рабочем листе «Лист1»:
   - для всех ячеек отмените выравнивание текста по центру;
   - отмените полужирный шрифт;
   - удалите содержимое ячеек диапазона.
2. Выделив ячейку A1, задайте команду ***Разработчик ⇒ Код⇒ Макросы.***
3. В диалоговом окне **Макрос** (см. рис. 1.7), выделив имя созданного макроса «Текст_Пр1», щелкните по кнопке **Выполнить** или, если ввели рекомендованное выше сочетание клавиш, Ctrl+я.
4. Повторите пункт 1 этого упражнения.
5. Удалив созданную прогрессию, выделите любую другую ячейку на том же листе (например, C5).
6. Повторите пункт 3 этого упражнения.
7. Сравните результаты двух попыток.



[1] Далее в тексте пособия эти два слова перед названием команды, как правило, опускаются.

Рис. 1.7

Неудача при втором запуске макроса «Текст_Пр1» вполне закономерна. Вспомните проблему с копированием формул, содержащих ссылки на другие ячейки. Преобразовав некоторые относительные ссылки в абсолютные или смешанные, вы уточнили задачу. Макрорекордеру тоже требуется помощь для правильной интерпретации действий пользователя.

При записи макроса по умолчанию реализуется режим абсолютных ссылок. Воспроизведя такой макрос, приложение Excel выполнит команды макроса, начиная с ячейки, которая была в активном состоянии на момент начала записи. Тем самым действия пользователя стандартно повторятся в одних и тех же ячейках рабочего листа. Если перед началом записи задать команду ***Разработчик ⇒ Код ⇒ Относительные ссылки,*** Excel будет выполнять макрос в соответствии с тем, где расположена активная ячейка в момент запуска макроса. Это смещение сохраняется для всех остальных записанных ячеек.

УПРАЖНЕНИЕ 4.

1. Задайте команду ***Разработчик ⇒ Код ⇒ Относительные ссылки.***
2. Выделив ячейку A1, задайте команду ***Разработчик ⇒ Код⇒ Макросы.***
3. В поле «Имя макроса» диалогового окна **Запись макроса** (см. рис.1.6) напечатайте имя *Прогрессия_2*.
4. В поле «Сочетание клавиш» введите прописную букву Я.
5. В списке «Сохранить в» выберите вариант – *Эта книга*.
6. В поле «Описание» напечатайте комментарий к создаваемому макросу – *Текстовая прогрессия в режиме относительной адресации*.
7. В ячейке A1 напечатайте слово *Январь* и нажмите клавишу Tab**.**
8. Выделив ячейку A1, попадите курсором мыши в маркер автозаполнения и протяните его до ячейки L1 (включительно).
9. ***Главная ⇒ Выравнивание ⇒ Кнопка запуска диалогового окна.***
10. В диалоговом окне **Формат ячеек**, активизировав вкладку «Выравнивание», задайте параметру **Выравнивание по горизонтали** значение «По центру» и нажмите кн. **ОК**
11. ***Главная ⇒ Шрифт ⇒ Ж (Полужирный).***
12. Закрыв диалоговое окно, отмените выделение диапазона A1:L1.
13. ***Разработчик ⇒ Код⇒ Остановить запись*** или в статусной строке щелкните по кнопке **Остановить запись.**
14. Выделив диапазон A1:L1**,** отмените результаты ваших действий на рабочем листе «Лист1»:
    - для всех ячеек отмените выравнивание текста по центру;
    - отмените полужирный шрифт;
    - удалите содержимое ячеек диапазона.

15. Выделив любую ячейку на первом листе (например, С1), задайте команду ***Разработчик ⇒ Код⇒ Макросы.***
16. В диалоговом окне **Макрос** (см. рис. 1.7), выделив имя созданного макроса «Тест_Пр2», щелкните по кнопке **Выполнить** или, если ввели рекомендованное выше сочетание клавиш, Ctrl+Shift+я**.**

В результате выполнения этого упражнения получена более универсальная, чем «Тест_Пр1» подпрограмма. Обе подпрограммы представлены на рисунке 1.8. Если на их примере мы разберёмся в том, что же вы получили, записывая выполняемые на рабочем листе действия, то легко сможем их отредактировать, а затем оптимизированную подпрограмму использовать для создания на её основе других макросов.

1.2 Альтернативные способы запуска макросов

Самый простой, но и самый неудобный способ запуска макросов – щелчок по кнопке **Выполнить** после выделения нужного макроса в диалоговом окне **Макрос**, которое открывается на экране по команде ***Разработчик ⇒ Код ⇒ Макросы.***

Самый быстрый способ запуска нужного макроса – нажать комбинацию «горячих» клавиш, назначенную при его записи. Вряд ли найдется много пользователей, пожелавших и способных удержать в голове большое количество подобных комбинаций.

Компромиссное решение – установить кнопки на панель быстрого доступа (ПБД) или на ленту в настраиваемую группу вкладки «Разработчик», а затем назначить им макросы.

УПРАЖНЕНИЕ 5.

1. Задайте команду: ***Файл ⇒ Параметры ⇒ Настройка ленты.***
2. В группе **Основные вкладки** диалогового окна **Параметры Excel** щелчком выделите название вкладки «Разработчик»[1].
3. Щелкните по кнопке **Создать группу** (см. рис. 1.9).
4. Если хотите присвоить созданной группе «Новая группа (настраиваемая)» содержательное название, выделив её, щелкните по кнопке **Переименовать**….
5. В поле «Отображаемое имя» диалогового окна **Переименовать** (см. рис. 1.10) напечатайте ваше название (к примеру, *Макросы* или *Мои макросы*).

[1] Именно название, а не опцию «флажок»

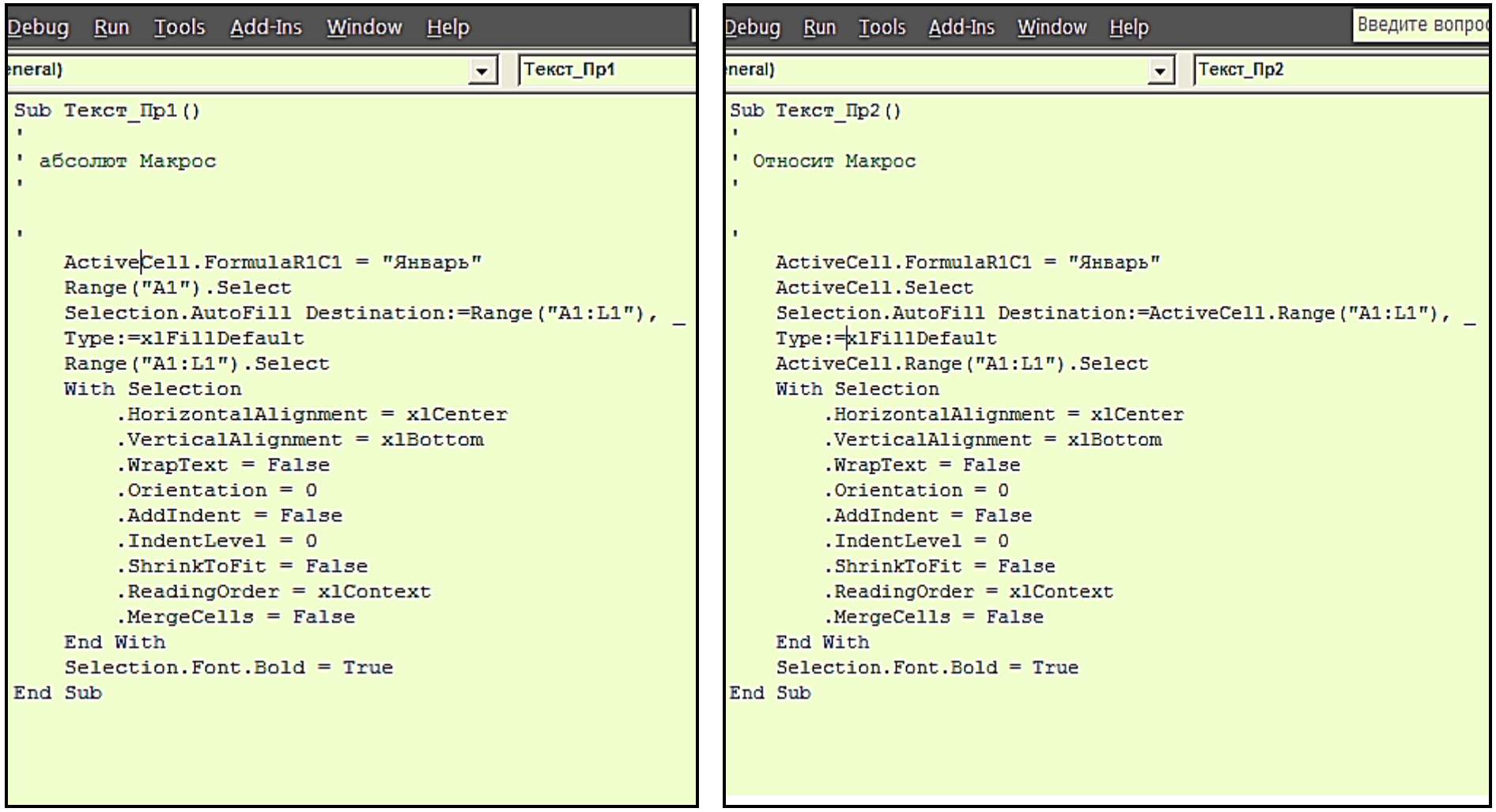

Debug Run Tools Add-Ins Window Help
neral)
Текст_Пр1
Sub Текст_Пр1()
'
' абсолют Макрос
'
'
ActiveCell.FormulaR1C1 = "Январь"
Range("A1").Select
Selection.AutoFill Destination:=Range("A1:L1"), _
Type:=xlFillDefault
Range("A1:L1").Select
With Selection
.HorizontalAlignment = xlCenter
.VerticalAlignment = xlBottom
.WrapText = False
.Orientation = 0
.AddIndent = False
.IndentLevel = 0
.ShrinkToFit = False
.ReadingOrder = xlContext
.MergeCells = False
End With
Selection.Font.Bold = True
End Sub
Debug Run Tools Add-Ins Window Help
Введите вопрос
neral)
Текст_Пр2
Sub Текст_Пр2()
'
' Относит Макрос
'
'
ActiveCell.FormulaR1C1 = "Январь"
ActiveCell.Select
Selection.AutoFill Destination:=ActiveCell.Range("A1:L1"), _
Type:=xlFillDefault
ActiveCell.Range("A1:L1").Select
With Selection
.HorizontalAlignment = xlCenter
.VerticalAlignment = xlBottom
.WrapText = False
.Orientation = 0
.AddIndent = False
.IndentLevel = 0
.ShrinkToFit = False
.ReadingOrder = xlContext
.MergeCells = False
End With
Selection.Font.Bold = True
End Sub

Рис. 1.8

6. В списке «Выбрать команды из» выделите пункт **Макросы**.
7. Найдя имя созданного макроса, щелкните по кнопке **Добавить >>**.
8. Выделив имя добавленного макроса в группе «Макросы», щелкните по кнопке **Переименовать**.
9. Выделив пиктограмму в окне **Переименовать**, щелкните по кн. **ОК**.
10. Щелкните по кн. **ОК** в окне **Параметры Excel**.
11. Кнопка, запускающая созданный макрос, появится в настраиваемой группе «Макросы» на вкладке «Разработчик».

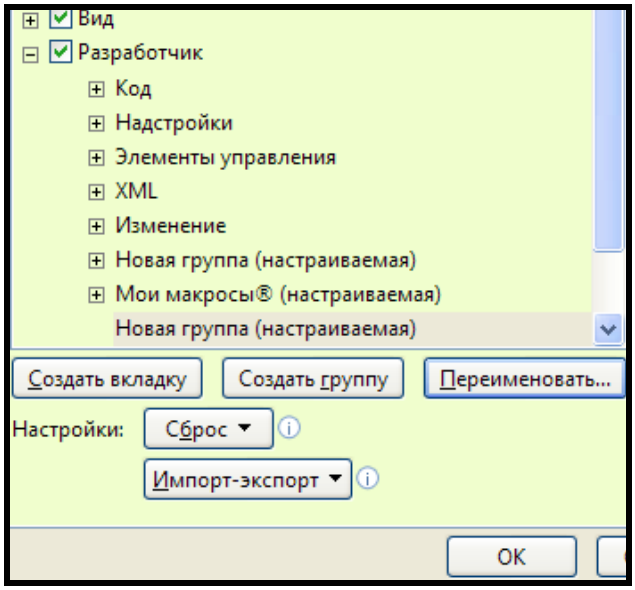


Рис. 1.9

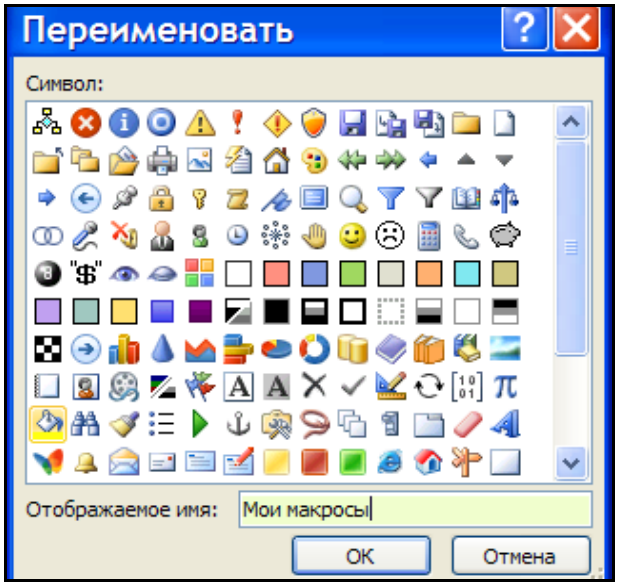


Рис. 1.10

УПРАЖНЕНИЕ 6.

1. Щелкните по кнопке **Настройка панели быстрого доступа** на **ПБД**.
2. В открывшемся списке выберите пункт «Другие команды…».
3. В списке «Выбрать команды из» диалогового окна **Параметры Excel** выделите пункт **Макросы**.
4. Найдя имя макроса «Текст_Пр2», щелкните по кнопке **Добавить >>**.
5. Щелкнув по кнопке **Изменить,** в диалоговом окне **Изменение кнопки**[9] выберите значок, а в поле **Отображаемое имя** задайте макросу более короткое имя и щелкните по кнопке **ОК.**
6. Щелчком по кнопке **ОК** закройте диалоговое окно **Параметры Excel.**
7. Кнопка запуска созданного макроса появится на **ПБД**.

УПРАЖНЕНИЕ 7.

1. Задайте команду: ***Файл*** ⇒ ***Параметры*** ⇒ ***Настройка ленты.***

[9] Аналог окна **Переименовать** на рисунке 1.10

2. Щелчком по значку + откройте содержимое группы «Мои макросы (настраиваемые)» вкладки «Разработчик».
3. Выделив кнопку, запускающую ненужный вам макрос, щелкните по кнопке **Удалить**.
4. На **ПБД** из контекстного меню кнопки, запускающей ненужный вам макрос, задайте команду ***Удалить с панели быстрого доступа*.**

1.3 «Анатомия» макроса

В предыдущих упражнениях для создания двух работоспособных подпрограмм «Текст_Пр1» и «Текст_Пр2», представленных на рисунке 1.8, вам не понадобились ни навыки программирования, ни знание языка программирования VBA (Visual Basic for Application), на котором они написаны. Вы с помощью макрорекордера получили программный код макроса на языке VBA, не написав при этом ни одной инструкции на этом языке. По сути, вы занимались программированием без программирования[10].

Теперь попытаемся, взглянув на коды этих подпрограмм, ответить на ряд вопросов, почему код одной из них более универсален, можно ли его оптимизировать и что нужно сделать, чтобы на основе созданного макроса получить новый макрос?

Благодаря тому, что структура программного кода инструкций на языке VBA максимально приближена к человеческому языку, понять и разобраться в том, что содержится в записанном макросе не сложно. Для ответа на все эти вопросы достаточно познакомиться с объектной моделью[11] Excel. Знание свойств и методов основных объектов этого приложения помогает не только понять, что генерируется в процессе записи выполненных действий, но и выполнить оптимизацию кода подпрограммы.

Объектно-ориентированный язык программирования VBA разрабатывался специально для доступа извне к объектам приложений пакета MS Office System. Краткое описание версии VBA для Excel и подборка упражнений в следующей главе призваны показать пользователям, как управлять приложением MS Excel традиционным программным образом. Выполнив их, вы сможете вручную писать более гибкие, чем макросы процедуры-подпрограммы и процедуры-функции.

1.3.1 Объектная модель приложения MS Excel

Приложения офисного пакета образованы из множества совместно действующих обособленных объектов, каждый из которых обладает определенным набором свойств (характеристик) и методами (способами) их изменения. Совокупность этих объектов обеспечивает функциональные возможности каждого конкретного приложения.

С объектной точки зрения все приложения MS Office построены одинаково. Всегда существует объект контейнерного типа Application, определяющий основную структуру данного приложения. В этот корневой объект встроены все остальные объекты данного приложения.

Общая для всех приложений схема иерархии такова:

- когда в объект Application вложен объект x1, говорят, что у объекта Application имеется свойство x1, а запись Application.x1 означает обращение к объекту x1;
- вложенный объект x1 может быть не менее сложен: в него могут быть встроены другие объекты, например, объект x2, в который в свою очередь вложен объект x3 и так далее;

---

[10] Провести грань между традиционным программированием и программированием с помощью макрорекордера довольно сложно.

[11] У каждого офисного приложения (MS Excel, MS Word, MS Access, MS PowerPoint и т.п.) своя объектная модель.

- в некоторых случаях для того, чтобы добраться до объекта x3, свойство или метод которого требуется использовать, приходится воспроизводить всю эту иерархическую цепочку.

Схема логична и хорошо структурирована. Проиллюстрируем её на примере программы MS Excel 2010.

На вершине объектной модели приложения Excel находится объект **Application**[12] – сама программа Excel.

Далее:

- x1 это **Workbooks** – коллекция[13] всех открытых рабочих книг (объектов Workbook);
- x2 это **Worksheets** – коллекция рабочих листов (объектов Worksheet) в открытой рабочей книге;
- x3 это **Range** — диапазон ячеек активного рабочего листа.

При обращении к объекту приложения из подпрограммы, создаваемой традиционным способом[14], в ссылке на него вводят разделенные точками названия всех расположенных выше в иерархической структуре объектов.

В большинстве случаев объект Application в ссылках можно опускать, поскольку, кроме приложения, с которым работает пользователь, больше обращаться не к чему.

Допустим, в Excel открыты две рабочие книги, в обеих имеется рабочий лист с названием Лист1, и нужно обратиться к ячейке A2 первого листа рабочей книги «Мои макросы». В этом случае ссылка, указывающая месторасположение ячейки A2, выглядит так:

Workbooks("Мои макросы").Worksheets ("Лист1").Range ("A2").

Без указания названия рабочей книги будет найден Лист1 в активной (открытой) рабочей книге.

Если рабочая книга «Мои макросы» — активный объект, то ссылку на неё также можно опустить и выражение станет ещё короче:

Worksheets ("Лист1").Range ("A2").

Когда Лист1 — активный рабочий лист, можно еще более упростить выражение: **Range("A2").**

Отметим, что в Excel отсутствует объект отдельная ячейка. Объект Range("A2"), состоящий из одного элемента, представляет собой ячейку A2. Альтернативой этому указанию является Cells(2,1), где 2 – порядковый номер строки, а 1 – порядковый номер столбца.

С помощью порядковых номеров можно определять единственный объект в коллекциях Workbooks (учитывается последовательность открывания рабочих книг) и Worksheets (учитывается расположение ярлычков).

Если в примере с двумя открытыми рабочими книгами первой была открыта другая книга, то обращение к ячейке A2 на первом рабочем листе в книге «Мои макросы» выглядит так:

Workbooks(2).Worksheets (1). Cells(2,1).

Некоторые свойства и методы вышеупомянутых объектов приведены в таблице 1.1

Объект	Свойства	Методы
Application	**ActiveCell (выделенная ячейка[15])**	

[12] Но если вы программируете в VBA, запуская редактор VB в MS Word, то объектом **Application** будет выступать Word.

[13] Объект коллекция — это множество однотипных объектов (все они вышли из одного класса).

[14] Программные коды печатаются пользователем в окне **Code** редактора VB.

[15] Это свойство представляет собой объект типа **Range.**

Workbook	**Name (имя)**	**Close (закрыть), Save (сохранить)**
Workbooks	**ActivateWorkbook**	**Activate (активизировать), Add**
Worksheet	**Name (Имя), Cells (все ячейки[16])**	**Activate, Delete (удалить)**
Worksheets	**Count (счет)**	**Add (добавить[17])**
Range	**Address, Formula[18], Offset (смещение[19])**	**Clear, Select (выделить)**
Selection	**Columns (столбцы), Rows (строки), Cells**	

1.3.2. Редактор VB

Для просмотра программного кода созданного макроса вновь воспользуемся диалоговым окном **Макрос**, из которого, выделив нужный макрос, можно запустить редактор VB (Visual Basic Editor) в режиме отладки.

Редактор VB допускает одновременную работу с ним и с тем приложением, откуда этот редактор был вызван.

В редакторе VB все создаваемые программные компоненты управления приложением объединяются в одно целое, называемое «Project»[20] (проект). Проект является неотъемлемой частью приложения, хранится вместе с ним и вне приложения не существует.

УПРАЖНЕНИЕ 8.

1. В этот раз, выделив любую ячейку на втором листе (к примеру, **B3)** книги **«**Мои макросы**»**, задайте команду ***Разработчик ⇒ Код⇒ Макросы.***
2. В диалоговом окне **Макрос** (см. рис. 1.7), выделив имя созданного макроса «Текст_Пр2», щелкните по кнопке **Войти.**
3. Для удобного просмотра программного кода и результатов пошагового выполнения инструкций разместите на экране окно приложения Excel и окно редактора VB так, как на рисунке 1.11, и задайте команду меню ***Debug ⇒ Step Into*** (пошаговая отладка).

[16] Или конкретная ячейка, если есть на нее ссылка.
[17] Новый рабочий лист или новую книгу.
[18] При стиле адресации A1.
[19] Смещение относительно активной ячейки.
[20] На языке VBA невозможно создать независимый от приложения проект.

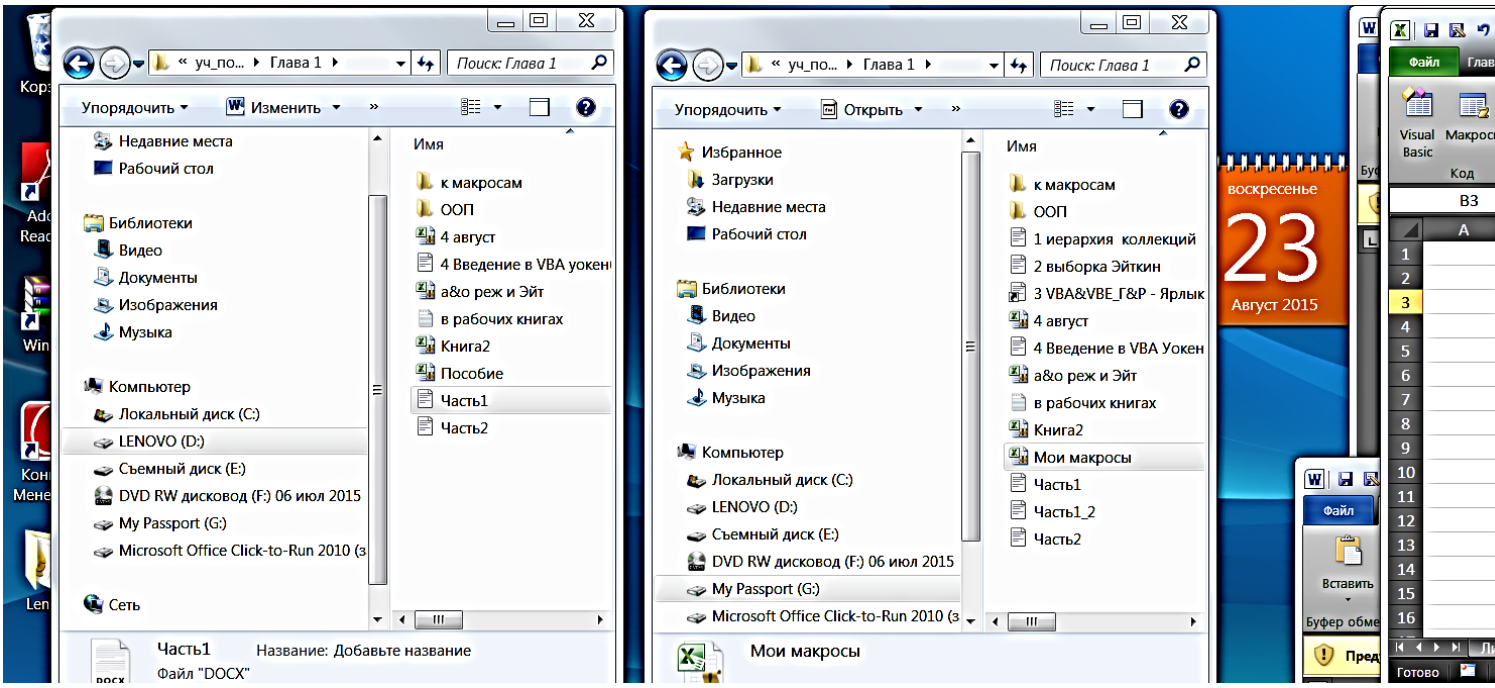


Рис. 1.11

4. Редактор VBE открывается с тремя окнами[21].

В окне **Project Explorer** (**Проводник проектов**) отображается древовидная структура всех открытых в данный момент рабочих книг[22], каждая из которых именуется здесь как *проект.* Проект «Project (Мои макросы)» включает в себя две папки:

- папка «MS Excel Objects» с тремя открытыми листами рабочей книги и самой книгой (эта папка отображается всегда). Если щелкнуть по объекту «Лист1», справа откроется пустое окно **Code;**
- папка «Modules» предназначена для хранения модулей с записанными в них макросами и процедурами (два сгенерированных вами макроса находятся в модуле «Module1»).

Если, открыв папку «Modules», щелкнуть по модулю «Module1», в окне **Code** отобразится программный код макроса «Текст_Пр2», оформленный в виде процедуры-подпрограммы.

5. Взгляните на строки подпрограммы. Начинается она с подсвеченного желтым цветом оператора Sub Текст_Пр2() и заканчивается оператором End Sub. Стрелка, указывающая на подсвеченную строку, свидетельствует о готовности строки к выполнению инструкции. Четыре строки с комментариями обозначены символом апостроф[23] (одиночная кавычка). На комментарии и пустые строки Excel не реагирует.

---

[21] Всего в редакторе предусмотрено девять окон

[22] Включая скрытые окна

[23] Любая исполняемая инструкция превращается в неисполняемую, если перед ней установить этот символ.

Текст каждой инструкции размещается в отдельной строке, но при этом допускается перенос на следующую строку. Знак переноса обозначен парой символов – пробел и нижний дефис. Если мысленно пронумеровать строки с исполняемыми инструкциями, нетрудно заметить, что инструкция в третьей строке не уместилась и была перенесена.

Для выполнения инструкции в подсвеченной строке нажмите на клавишу F8 и взгляните на рабочий лист. Если стрелка переместилась на следующую строку, а на рабочем листе ничего не изменилось, продолжайте нажимать на клавишу F8 до тех пор, пока, пройдя по всем строкам с комментариями, не подсветится строка с первой исполняемой инструкцией.

6. Нажмите на клавишу F8 и взгляните на рабочий лист. В выделенной (активной) ячейке отобразилось слово Январь. Стрелка же в окне **Code** сместилась и указывает на следующую подсвеченную строку.

7. Продолжайте нажимать на клавишу F8**.** Как только Excel выполнит строку с очередной инструкцией, стрелка переходит к следующей строке и подсвечивает ее желтым цветом.

Далее построчно приведено толкование каждой инструкции[24] с ссылками на действия, выполненные при этом пользователем.

1-я строка. В объект **ActiveCell** (активная ячейка) методом **FormulaR1C1** вводится текст (*Напечатав слово Январь в ячейке A1, вы, чтобы завершить ввод данных в активную ячейку, нажали на клавишу Tab*).

2-я строка. Инструкция **ActiveCell.Offset(0,1).Range("A1").Select** формируется одновременно с инструкцией в первой строке и означает она следующее: методом Offset(0,1) сместить вправо на один столбец активную ячейку и выделить её рамкой *(Так макрорекордер интерпретировал ваше второе действие – нажатие на клавишу Tab. Затем, нажав на клавишу «стрелка влево», вы отменили выделение соседней ячейки и тем самым изменили эту инструкцию. В результате она стала такой, какой вы её видите на листинге рисунка 1.8).*

Команда «сместить рамку вправо и тут же вернуть ее обратно» лишняя. Чтобы описываемая трансформация инструкции стала понятнее, прочитайте рекомендации, приведенные в конце этого подраздела, и выполните их.

Строки 3 и 4. Selection. AutoFill[25]. К объекту «Выделение» применен метод «Автоматическое заполнение» с двумя аргументами:

- **Destination** (конечная цель) со значением **ActiveCell.Range("A1:L1").**
- **Type** (тип заполнения) со значением **xlFillDefault** (по умолчанию).

(*Щелкнув курсором мыши по маркеру автозаполнения активной ячейки, вы протянули его вправо на двенадцать ячеек*).

5-я строка. К объекту **ActiveCell** со свойством **Range("A1:L1")** применить метод **Select** (выделить). (*При отпускании левой кнопки мыши создан объект **Selection.***)

Строки 6-16. Пример того, что некоторые простые действия пользователя записываются на нескольких строках. В конструкции **With … End With** (от и до) для объекта **Selection** перечислены значения девяти его параметров, из которых вы задали только один **HorizontalAlignment = xlCenter** (выравнивание по горизонтали со значением по центру), а для остальных оставлены значения по умолчанию. (*На вкладке «Выравнивание» диалогового окна «Формат ячеек» из списка «Выравнивание по горизонтали» выбрано значение «По центру»*).

17-я строка. **Selection. Font. Bold =True**. Свойство **Bold** (полужирный) применено к вложенному в объект **Selection** (выделение) объекту **Font** (шрифт).

---

[24] Здесь и далее инструкции и отдельные их фрагменты напечатаны гарнитурой Calibri.

[25] Макрорекордер определяет выделенный объект на рабочем листе, а затем в генерируемых инструкциях использует объект **Selection**

(*Вы задали команду* ***Главная ⇒ Шрифт ⇒ Полужирный***).

УПРАЖНЕНИЕ 9.

Пояснения к инструкциям макроса Текст_Пр2() станут понятнее, если в двухоконном режиме вы, повторяя все пункты упражнения 2, будете отслеживать появление инструкций создаваемого макроса в окне **Code**.

1.4 Редактирование и оптимизация программного кода макроса

Каждый раз обязательно сделайте копию того макроса, который вы хотите оптимизировать. Изменив название копии, приступайте к оптимизации. Закончив эту процедуру, запустите копию на выполнение и, если она сработает аналогично оригиналу, его можете стереть из модуля.

УПРАЖНЕНИЕ 10.

1. Выделив в окне **Code** всю подпрограмму «Текст_Пр2», включая операторы Sub Текст_Пр2() и End Sub, скопируйте её и копию вставьте на свободное место в модуле1, задав поочередно команды ***Edit ⇒ Copy*** и ***Edit ⇒ Paste*** из меню редактора VBA.
2. Измените название копии. Например, на такое: Sub Текст_Пр3().
3. Выделив все строки с непрописанными комментариями, удалите их.
4. Выделив строку с инструкцией **ActiveCell.Select**, удалите её.
5. Выделите восемь строк конструкции With … And With, содержащих параметры с оставленными по умолчанию значениями, и удалите их.
6. В семнадцатой строке, выделив правую часть инструкции вместе с разделяющей точкой (**.Font.Bold = True**), скопируйте её и копию вставьте в конструкцию **With … End With**.

После этой операции конструкция должна выглядеть так:

```
With Selection
        .HorizontalAlignment = xlCenter
        .Font.Bold = True
End With
```

7. Выделив семнадцатую строку (**Selection.Font.Bold = True**), удалите её.
8. Сравните оптимизированный макрос, представленный на рисунке 1.12, с исходным программным кодом этого же макроса (см. рис. 8).

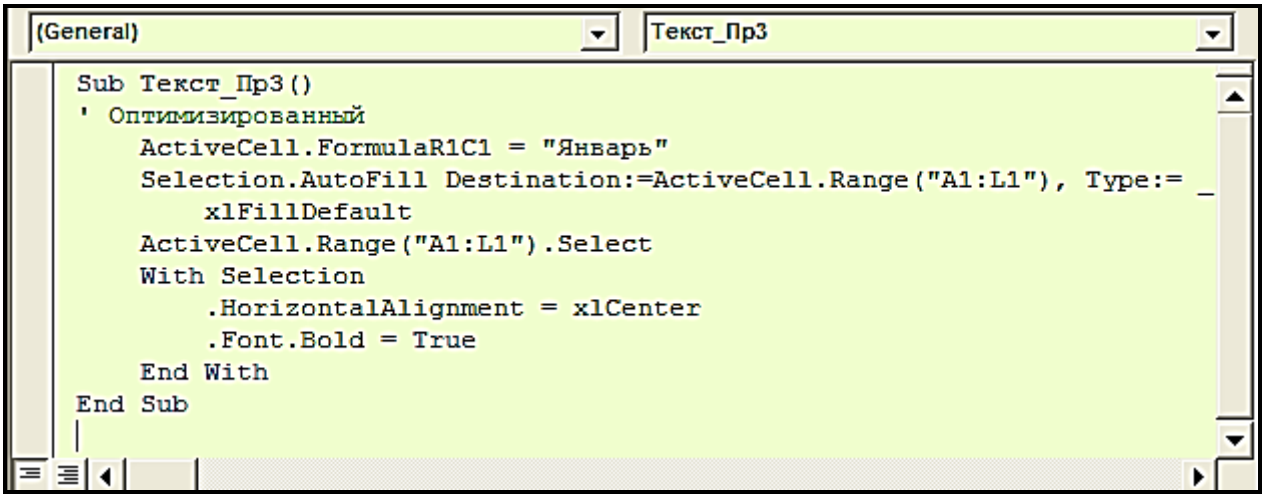


Рис. 1.12

9. Для сохранения оптимизированного макроса Sub Текст_Пр3() закройте окно **Code** редактора VB.

10. Задав команду ***Разработчик ⇒ Код ⇒ Макросы***, выделите оптимизированный макрос и выполните его.

11. Если все сработало правильно, удалите макрос Текст_Пр2().

1.5 Создание макроса на основе ранее созданного

УПРАЖНЕНИЕ 11.

1. Выделив в окне **Code** макрос Sub Текст_Пр3(), включая операторы Sub Текст_Пр3() и End Sub, скопируйте его и копию вставьте на свободное место в модуле1, задав поочередно команды ***Edit ⇒ Copy*** и ***Edit ⇒ Paste*** из меню редактора VBA или контекстных меню.

2. Измените текст комментария на слово «*Преобразованный*» и название созданной копии. Sub Текст_Пр3() замените на Sub Текст_Пр4().

3. В двух строках копии макроса измените значение свойства Range с Range("A1:L1") на Range ("A1:A12").

4. Закройте редактор VB и, щелкнув по любой ячейке, запустите преобразованный макрос (см. рис. 1.13) на выполнение.

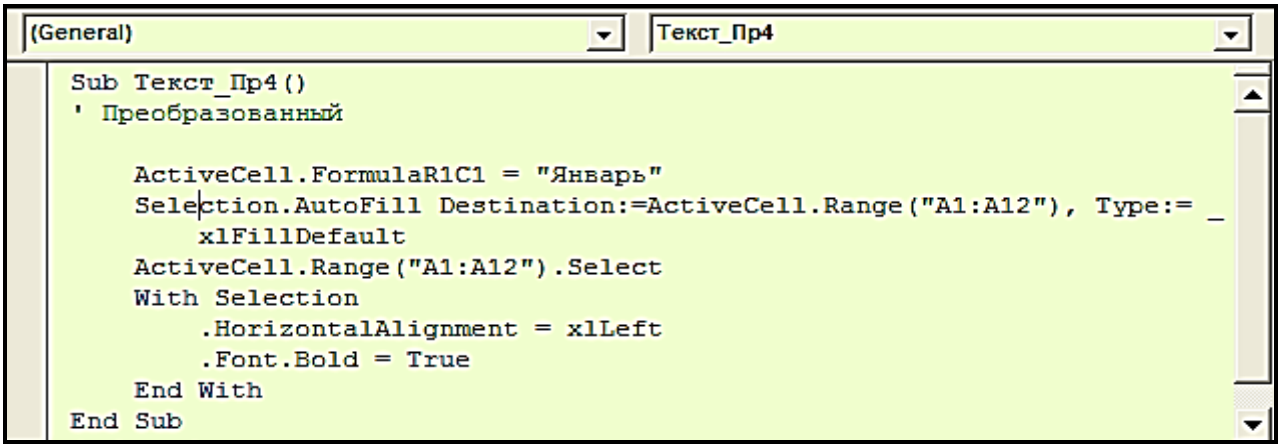


Рис. 1.13

Созданный макрос Sub Текст_Пр4() самостоятельно преобразуйте в макрос, формирующий числовую прогрессию. Кроме небольшой коррекции значений, вводимых в активные ячейки, вам придется добавить инструкцию, схожую с той, что упоминалась в восьмом упражнении, – методом **Offset(0,1)** сместить вправо на один столбец активную ячейку:

ActiveCell.Offset(0,1).Range("A1").Select .

Окончательный вид программного кода макроса Sub Число_Пр5() будет выглядеть примерно так:

```
Sub Число_Пр5()
' Арифметическая прогрессия
 ActiveCell.FormulaR1C1 = 1
 ActiveCell.Offset(1, 0).Range("A1").Select
 ActiveCell.FormulaR1C1 = 2
 ActiveCell.Offset(-1, 0).Range("A1:A2").Select
 Selection.AutoFill Destination:=ActiveCell.Range("A1:A12"), Type:= _
    xlFillDefault
End Sub
```

Поскольку панель быстрого доступа имеет ограниченные размеры, продолжим рассмотрение альтернативных способов запуска макросов. Макрос, регулярно используемый в конкретной рабочей книге, удобно запускать прямо с ее рабочего листа.

Поместите элемент управления **Кнопка** на рабочий лист книги, где создавали макросы Sub Текст_Пр1() –Пр4(), и свяжите ее с одним из них.

УПРАЖНЕНИЕ 12.

1. Задайте команду: ***Разработчик*** ⇒***Элементы управления***⇒ ***Вставить*** ⇒ ***Кнопка***[26] (см. рис. 1.14).

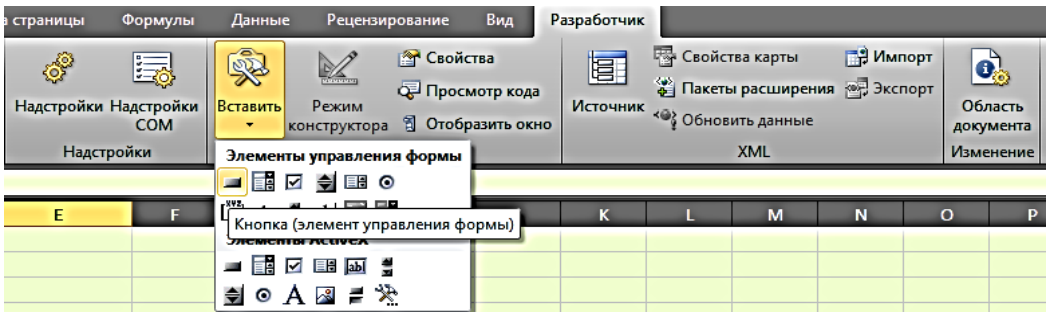


Рис. 1.14

[26] Кнопка из группы «Элементы управления формы».

2. Щелкнув в любом месте рабочего листа и протащив по диагонали курсор мыши, начертите объект **Кнопка**.
3. В момент отпускания кнопки мыши появится диалоговое окно **Назначить макрос объекту** со списком доступных макросов (см. рис. 1.15)
4. Выберите нужный макрос и нажмите кн. «ОК».
5. Из контекстного меню кнопки выберите команду **Изменить текст** и замените стандартную надпись «*Кнопка 1*» на содержательную, к примеру, «*Вставить прогрессию*».

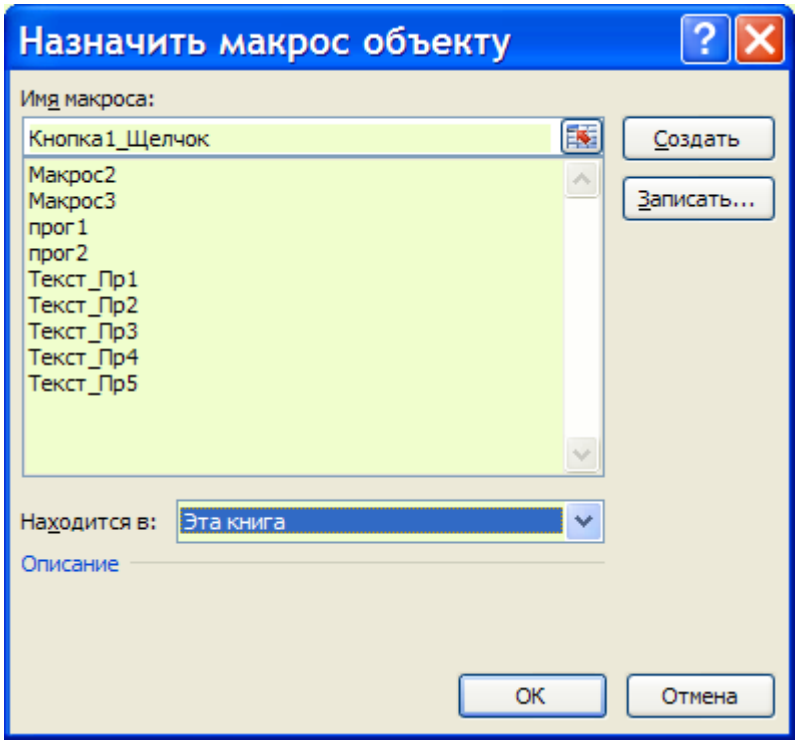


Рис. 1.15

6. Для завершения первой части этого упражнения щелкните по любой ячейке листа.

Итак, создав пять макросов и обсудив программный код одного из них, вы узнали:

1. Excel содержит более сотни объектов, большинство из которых объединены в коллекции. Коллекция – группа однотипных объектов.
2. Объекты образуют иерархическую структуру. Верхнюю ступень иерархии занимает объект Application (программа Excel), затем рабочая книга (Workbook) и далее рабочий лист (Worksheet), который исполняет роль контейнера для объекта диапазон (Range).
3. Объект Range используется для хранения и отображения данных.
4. При обращении к объекту из коллекции указывают его порядковый номер или имя. Например, Worksheets (1) или Worksheets (“Лист 1”).
5. Каждый объект обладает набором свойств и методов.

6. Свойство – это характеризующий объект параметр, а метод определяет действие, которое может быть совершено объектом или над объектом.

Глава 2. Синтаксис и программные конструкции VBA

Макрорекордер, задействованный во всех упражнениях предыдущей главы, интерпретируя физические действия пользователя и используя язык VBA**,** создает программный код макросов. Все макросы реализуют линейные алгоритмы[27]. Запуская их в процессе обработки данных в Excel, пользователь автоматизирует рутинные операции, что, естественно, повышает эффективность и комфортность работы за компьютером.

Если приложить не физические, а умственные усилия, эффект может стать еще более впечатляющим. Помимо автоматизации продвинутый пользователь способен решать специфические для его профессии задачи, расширяя тем самым стандартные функциональные возможности программы Excel.

Что для этого потребуется? Узнать чуть больше об объектно-ориентированном языке программирования VBA, а для этого вполне достаточно разобраться и выполнить предлагаемые в этой и следующей главе упражнения.

2.1 Синтаксические правила VBA

1. Ввод каждой инструкции начинается с новой строки и заканчивается нажатием на клавишу Enter.
2. Используя пару символов – пробел и подчеркивание, можно часть инструкции перенести на нижнюю строку, объединив тем самым несколько физических строк в одну логическую.
3. Одинарная кавычка, поставленная перед исполняемой инструкцией, преобразует содержимое строки в комментарий.
4. Имена переменных, констант и процедур пишутся без пробелов длиной не более 255 символов.
5. Символьные значения заключаются в двойные кавычки (например, Range(“A1”)).
6. При написании программных кодов процедур вместо употребляемой при вводе дробных чисел в Excel плавающей запятой (ПЗ) используется плавающая точка (ПТ).
7. Значения дат вводят в формате #дд\мм\гггг#. Например, #10\8\2015#.
8. Для объединения строковых значений применяется оператор конкатенации &. Например, MsgBox "Ответ=" & Str(z)[28].
9. Язык VBA не чувствителен к регистру.

2.2 Операторы

Для выполнения различных операций в программе на языке используются операторы. *Оператор* — наименьшая выполняемая единица программного кода VBA.

---

[27] Увы, записывать логические условия разветвления и выхода из циклов невозможно.

[28] Для целых чисел функцию **Str(),** преобразующую числовой формат данных в текстовой формат, можно не использовать.

В качестве оператора присваивания используется символ **=**. Выражение a=5 означает "переменной a присвоить значение 5".

Арифметические операторы представлены в таблице 2.1, шесть операторов сравнения – в таблице 2.2 и три логических оператора в таблице 2.3.

Таблица2.1

Оператор	Название	Пример	Результат при a=5 и b =2
+	Сложение	a+b	7
-	Вычитание	a-b	3
*	Умножение	a*b	10
/	Деление	a/b	2,5
^	Возведение в степень	a^b	25
\	Целочисленное деление	a\b	2
Mod	Деление по модулю	a Mod b	1

Таблица 2.2

Оператор	Название	Пример	Результат при a=5 и b =2
=	Равно	If (a = b)	False
<>	Не равно	If (a <> b)	True
>	Больше чем	If (a > b)	True
<	Меньше чем	If (a < b)	False
>=	Больше или равно	If (a >= b)	True
<=	Меньше или равно	If (a <= b)	False

Значения **True** (Истина) и **False** (Ложь) называются булевскими значениями.

К логическим операторам обращаются при проверке нескольких условий. Три наиболее часто используемых (**And**, **Or** и **Not**) представлены в таблице 2.3, отображающей результаты этих логических операций над двумя переменными X1 и X2 типа ***Boolean***.

Таблица 2.3

X_1	X_2	X_1 And X_2	X_1 Or X_2	Not X_1
False	False	False	False	True
False	True	False	True	True
True	False	False	True	False
True	True	True	True	False

2.3 Типы данных

Основное назначение приложения Excel – обработка числовых данных, для описания которых в языке применяется набор типов с указанием диапазона значений и объема[29] резервируемой для их хранения памяти компьютера.

В начале программы следует объявить обо всех используемых в ней переменных. Переменная – это идентификатор данных, размещаемых в памяти компьютера. Объявляя переменную в разделе объявлений модуля или внутри процедуры, пользователь сообщает компилятору о своем намерении использовать указанные данные и описывает их тип.

[29] В байтах.

Как правило, в объявлении переменных используется ключевое слово **Dim** (сокращение слова dimension - размер), но пользователи, пожелавшие расширись область видимости переменной, заменяют его на **Public**. Например, **Dim** X **As Byte** или **Public** Y **As Single**, где X и Y – имена переменных, а Byte и Single – типы числовых данных, которым будет присвоена эта переменная.

Типы данных, которыми могут оперировать программы на языке VBA для Excel:

- **Byte, Integer, Long** для целых чисел;
- **Byte, Integer, Long** для дробных чисел;
- **String** для символьных строк;
- **Boolean** для логических величин;
- **Date** для дат.

2.3.1 Целые и дробные числа

Разделение переменных по типу позволяет под каждую переменную отводить нужный объем памяти. Под шесть числовых типов данных в VBA отводится от одного до двадцати двух байт.

Для целых чисел (см. таблицу 2.4) отведены объемы памяти в один, два и четыре байта. Это меньше, чем для остальных численных типов данных. Математические операции над числами типа **Byte, Integer** и **Long,** а также операции сравнения их выполняются быстрее, чем эти же операции с дробными числами. Кроме того, при выводе на экран с помощью функции **MsgBox** числовой формат результата вычислений целых чисел автоматически преобразуется[30] в текстовой формат.

Таблица 2.4

Типы данных	Диапазон целых чисел
Byte	**от 0 до 255**
Integer	**от -32 768 до 32 767**
Long	**от -2 147 483 648 до 2 147 483 647**

УПРАЖНЕНИЕ 13

1. Открыв новую рабочую книгу, задайте команду ***Разработчик ⇒ Код⇒ Visual Basic*** или нажмите на клавиши **Alt+F11.**
2. Если окно **Code** не отобразилось, из меню редактора VB задайте команду ***View ⇒ Code.***
3. Перепечатав программный код процедуры Sub Пример1( ), представленный на рисунке 2.1, задайте команду ***Run ⇒ Sub/User Form.***
4. Напечатанную процедуру можно запустить и из диалогового окна **Макрос**. Для этого закройте редактор VB[31], задайте команду ***Разработчик ⇒ Код⇒ Макросы*** и, выделив эту процедуру, щелкните по кнопке «Выделить».
5. Диалоговое окно с результатом вычисления должно появиться на рабочем листе (см. рис. 2.2). Поскольку это окно модально по отношению к приложению, его следует закрыть, чтобы продолжить работу с Excel.

[30] Для преобразования чисел других форматов используется встроенная функция **Str ( ).**

[31] Повторно нажмите на клавиши **Alt+F11.**

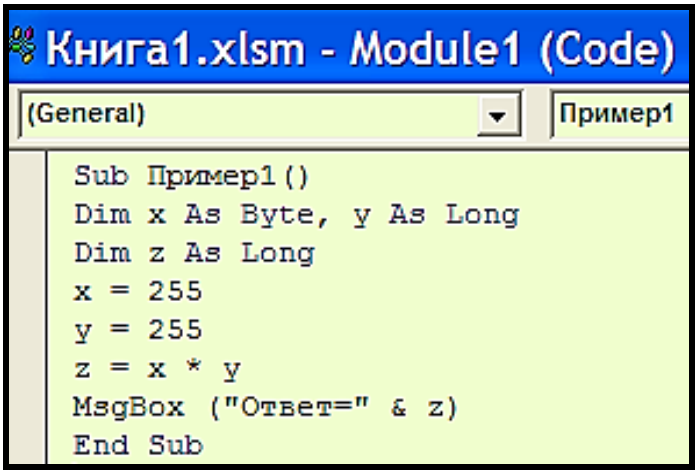


Рис.2.1

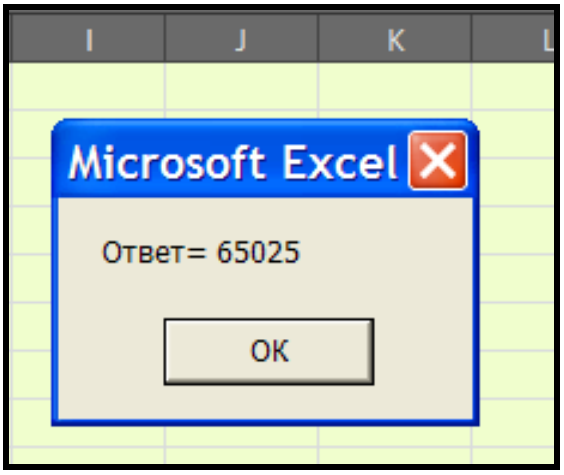


Рис. 2.2

Для дробных чисел в формате с плавающей запятой (ПЗ) язык VBA резервируют объемы памяти размером в четыре и восемь байт соответственно. Данные, хранимые как тип **Single,** называют числами одинарной точности, а хранимые как **Double** – числами двойной точности. И те и другие подвергаются округлению.

Очень большие или очень малые числа отображаются на экране в экспоненциальной форме (без начальных и конечных нулей и с одной цифрой перед десятичным знаком).

Например, 5.4E6=5400000 и 5.01E-2=0.0501

Чтобы указать на действительное расположение десятичного знака, представленное число умножается на 10 в некоторой степени. На экране вместо 10 используется буква E.

Тип данных **Currency** используется при денежных вычислениях, когда требуется высокая точность (резервируют объемы памяти размером в 8 байт). Диапазоны хранимых дробных чисел в обоих форматах представлены в таблице 2.5.

Таблица 2.5

Дробные	**Диапазон**	
В формате с ПТ	**Отрицательные числа**	**Положительные числа**
Single	**от -3.4 *10^{38} до -1.4 *10^{-45}**	**от 1.4 *10^{-45} до 3.4 *10^{38}**
Double	**от -1.7*10^{308} до -4.9 *10^{-324}**	**от 4.9*10^{-324} до 1.7*10^{308}**
В формате с ФТ		
Currency	**от -922337203685477.5808 до 922337203685477.5807**	

Деньги отображаются как дробные числа в формате с фиксированной точкой (ФТ), т.е., десятичный знак всегда находится в одном и том же месте, а справа от него расположены четыре цифры.

2.3.2 Строки

Тексты, сохраняемые в VBA, называют строками. Строка может содержать текстовые символы любых типов: буквы, цифры, знаки пунктуации, разделительные символы и пр.

Строки всегда заключаются в двойные кавычки. Для их хранения используют тип данных **String**. Существует две категории строк: строки переменной длины (*по умолчанию*) и строки фиксированной длины (от 1 до 65400 символов). Под каждый символ отводится в памяти один байт.

Числовые данные, вводимые с экрана монитора с помощью функции **InputBox,** следует преобразовать в числовой формат с помощью встроенной функции **Val()**. Любые данные, кроме типа **String**, перед выводом на экран должны быть преобразованы в строковый формат.

УПРАЖНЕНИЕ 14

Перепечатав программный код процедуры Sub ВводВывод (), представленной на рисунке 2.3, задайте команду ***Run ⇒ Sub/User Form.***

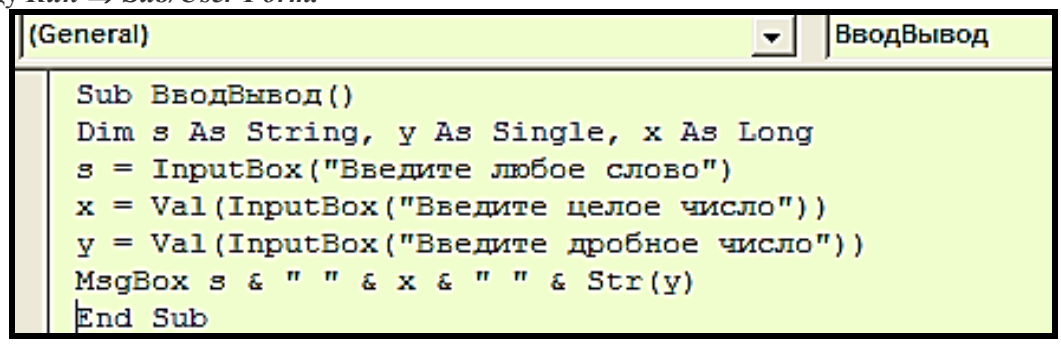


Рис. 2.3

2.3.3 Дата и время

Для хранения дат и времени VBA, как и приложение Excel, использует тип **Date**. Это тип последовательных дат, отсчитываемых от базовой даты[32] (число нуль). Базовой датой для VBA является 30 декабря 1899 года, а для MS Excel - 1 января 1900 года.

Даты можно складывать или вычитать одну из другой. Для изменения значения даты можно прибавить к ней или отнять от нее число.

УПРАЖНЕНИЕ 15

Перепечатав программный код процедуры Sub Гадалка (), представленной на рисунке 2.4, задайте команду ***Run ⇒ Sub/User Form.***

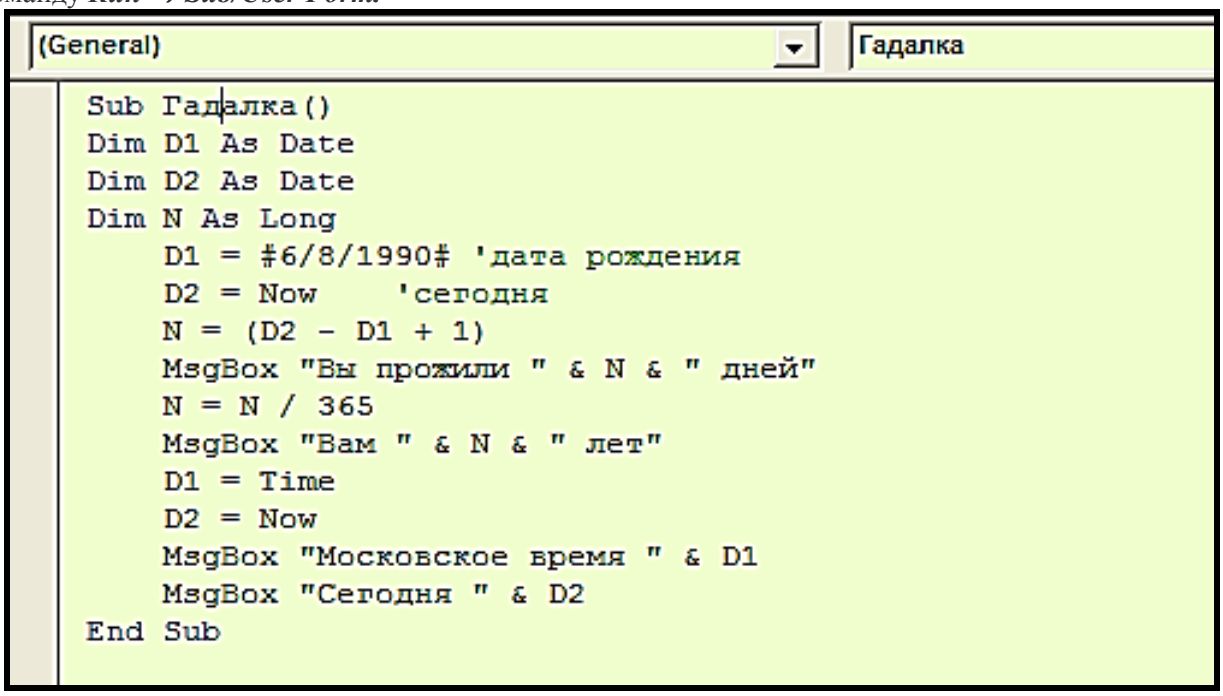


Рис. 2.4

В листинге на рисунке 2.4 применены две стандартные функци VBA:

- **Time** возвращает текущее системное время;

[32] Для представления дат, предшествующих базовой дате, VBA использует отрицательные числа.

- **Now** возвращает текущие системные время и дату.

Обращаться к стандартным функциям VBA можно напрямую из программного кода, а к пользовательским функциям – из мастера функций Excel[33].

2.3.4 Логические величины

Результатом выполнения любой операции сравнения является одно из двух возможных значений[34] **True** (Истина) или **False** (Ложь). Для объединения нескольких условий применяют логические функции (см. таблицу 2.3). Программы, основываясь на этих условиях, способны принимать решения на выполнение тех или иных действий. Напомним, что значения **True** и **False** называются булевскими значениями, а тип данных - ***Boolean***.

УПРАЖНЕНИЕ 16

1. Создав или открыв рабочую книгу с поддержкой макросов (*.xlsm), задайте команду ***Разработчик ⇒ Код⇒ Visual Basic***[35] или нажмите на клавиши **Alt+F11.**
2. Задав команду ***Insert ⇒Module,*** напечатайте программный код процедуры Sub Четыре_Ф (), представленный на рисунке 2.5, и закройте окно редактора VBE.

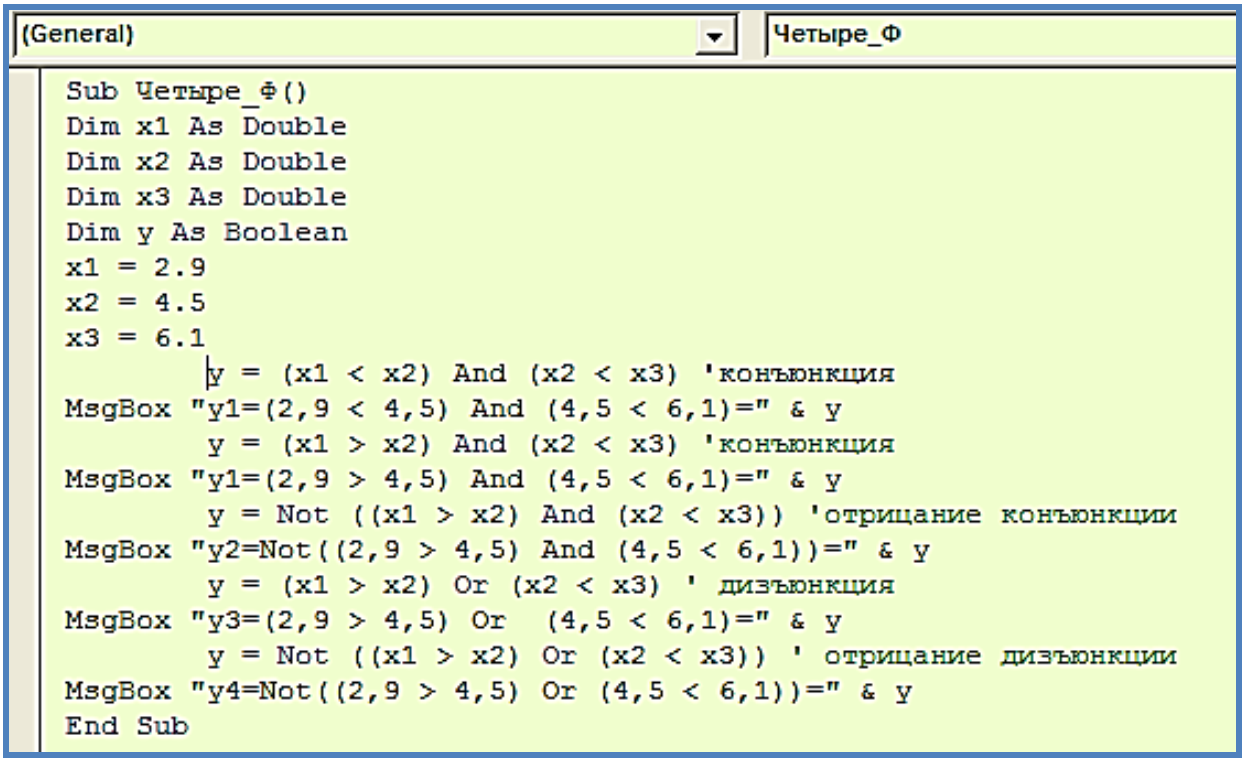


Рис. 2.5

[33] Команда вызова **Call** не требуется.
[34] Любое условие может принимать два значения.
[35] Если окно **Code** не отобразилось, из меню редактора VB задайте команду ***View ⇒ Code.***

3. Задав команду ***Разработчик ⇒ Код ⇒ Макросы,*** выделите созданную процедуру и щелкните по кнопке **Выполнить.**
4. Пять диалоговых окон с результатами вычислений поочередно появятся на открытом рабочем листе (см. рис 2.6,)

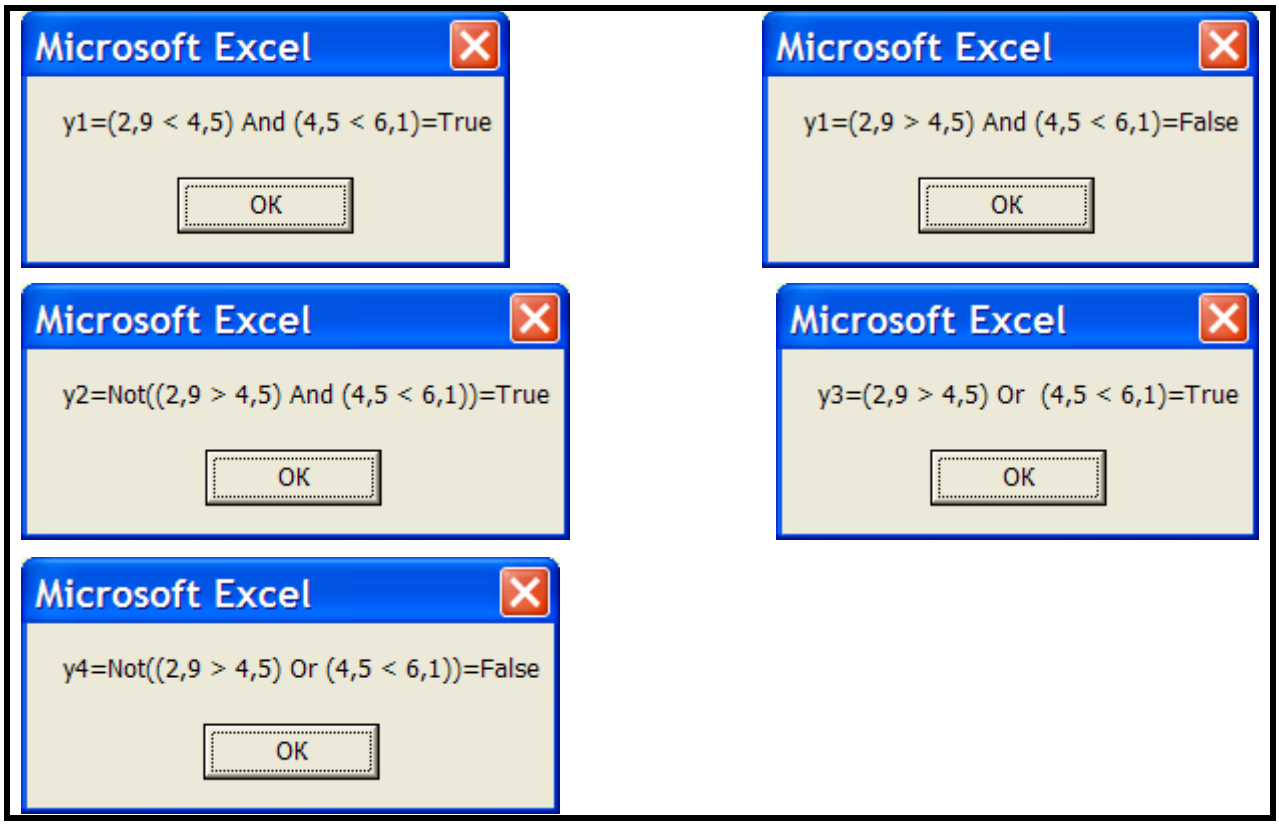


Рис. 2.6

2.3.5 Любые данные

По умолчанию в программах на языке VBA не запрещено использование переменных без предварительного их объявления. При написании программ на языке VBA можно не объявлять переменные! Просто включайте их в текст программы по мере надобности, и VBA автоматически присвоит им универсальный тип данных - **Variant**[36].

В переменной, имеющей этот универсальный тип, может содержаться что угодно (и число, и текст). Несмотря на то, что этот тип данных может сохранять любые числовые и строковые значения, его следует избегать по ряду причин. Во-первых, для хранения значений типа **Variant** выделяется 22 байта памяти, а, к примеру, для **Double** и **Currency** – 8 байт. Во-вторых, математические операции и операции сравнения над данными типа **Variant** выполняются медленнее, чем подобные операции над данными любого другого типа.

2.4 Операторы условного и безусловного перехода

В языке VBA предусмотрен ряд операторов, позволяющих управлять ходом выполнения программы[37]:

---

[36] Тип данных **Variant** аналогичен формату **Общий** в Excel.

[37] Реализуя тем самым более гибкие, чем линейные, нелинейные алгоритмы.

- операторы условного перехода **If ... Then** и **Select Case,** предназначенные для выполнения той или иной инструкции в зависимости от результатов проверки указанных в них условий;
- операторы цикла с условием **Do Until … Loop** и **Do While … Loop** многократно выполняют блок инструкций до тех пор, пока результат проверки заданного условия соответствует определенному результату (True или False)**.**
- оператор цикла со счетчиком **For … Next** и его разновидность оператор **For Each … Next.**

2.4.1 Оператор If…Then

Синтаксис однострочной конструкции самого популярного у программистов оператора **If ... Then:** **If *Условие* Then *Инструкция_1* [Else *Инструкция_2*]**.

Здесь и далее необязательная часть оператора заключена в квадратные скобки.

Если результат проверки условия **True** (истина)**,** то выполняется первая инструкция, в противном случае – вторая.

Справа представлен синтаксис многострочной конструкции оператора условного перехода.

В этой конструкции каждая инструкция может быть заменена блоком инструкций.

Многострочная конструкция заканчивается строкой с ключевым словом End If.

```
If Условие Then
    Инструкция_1
[Else
    Инструкция_2]
End If
```

УПРАЖНЕНИЕ 17

1. Открыв рабочую книгу, сохраненную в формате с поддержкой макросов (*.xlsm), поместите любые три целых положительных числа в ячейки

A1, A2 и A3.

2. Используя алгоритм, представленный на рисунке 2.7, напечатайте в окне **Code** программный код процедуры «Нахождение корней квадратного уравнения $ax^2+bx+c=0$».
3. Для вычисления значения переменной ds используйте встроенную функцию **Sqr().**

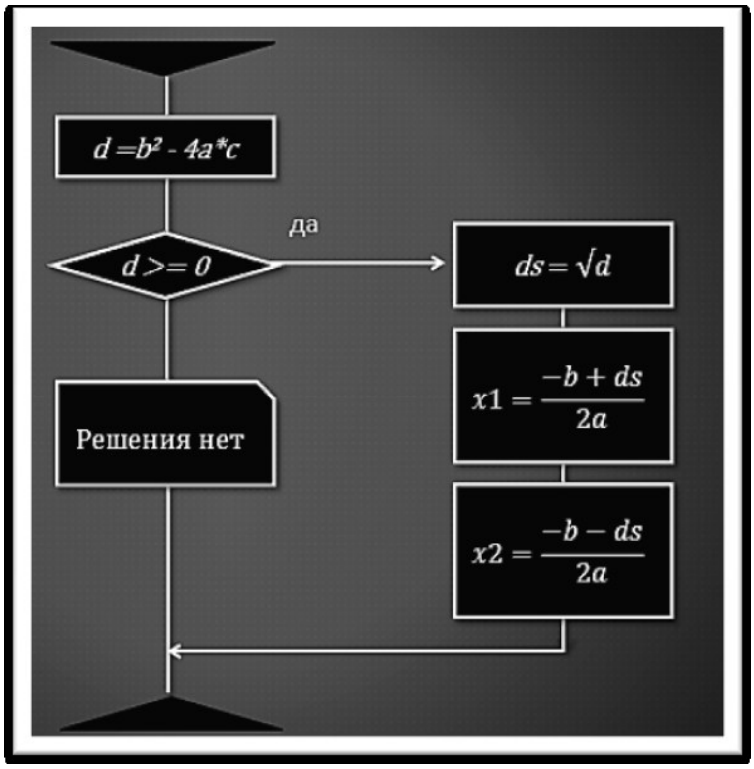


Рис.2.7

4. Результат решения выведите в диалоговом окне с помощью функции **MsgBox()** и функции преобразования чисел в текстовой формат **Str().**
5. Сравните программный код созданной процедуры с рисунком 2.8.

```
Sub Уравнение()
Dim a, b, c As Integer
Dim d As Long
Dim ds, x1, x2 As Single
a = Cells(1, 1)
b = Cells(2, 1)
c = Cells(3, 1)
d = b ^ 2 - 4 * a * c
    If d > 0 Then
        ds = Sqr(d)
        x1 = (-b + ds) / 2 * a
        x2 = (-b - ds) / 2 * a
        MsgBox "x1= " & Str(x1) & " x2= " & Str(x2)
    Else
        MsgBox "Решения нет"
    End If
End Sub
```

Рис. 2.8

В тех случаях, когда какие-либо действия должны быть предприняты только при одновременном выполнении нескольких условий, можно вложить операторы **If...Then** один в другой (см. рис 2.9).

Если в многострочную конструкцию условного оператора вы добавите ключевое слово **Elself,** то сможете**,** поочередно проверив несколько условий, выбрать один из предлагаемых на рисунке 2.9 вариантов.

Отметим, что в любой из предложенных конструкций ключевое слов**о Then** должно находиться в одной строке с **If** и условием.

Многострочные конструкции должны заканчиваться строкой с ключевым словом **End If.** Для удобства их чтения используйте отступы.

```
If Условие_1 Then
  Инструкция_1
          If Условие_2 Then
              Инструкция_2
          Else
              Инструкция_3
          End If
Else
  Инструкция 4
End If
```

```
If Условие_1 Then
    Инструкция_1
        ElseIf Условие_2 Then
                Инструкция_2
            ……….
        ElseIf Условие_n Then
                Инструкция_n
Else
    Инструкция_n+1
End If
```

Рис.2.9

УПРАЖНЕНИЕ 18

Создайте процедуру-функцию для вычисления стоимости партии закупаемых у издательства книг. В качестве обязательных аргументов создаваемой функции используйте количество книг в партии и цену одной книги.

Предлагаемую издательством скидку и условия предоставления смотри в таблице 2.5.

Таблица 2.5

Количество	Скидка (%)
0 - 99	0
100 – 200	7
201 – 300	10
301 и более	15

1. Создав или открыв рабочую книгу с поддержкой макросов (*.xlsm), задайте команду ***Разработчик ⇒ Код⇒ Visual Basic***[38] или нажмите на клавиши **Alt+F11.**
2. Задав команду ***Insert ⇒ Module,*** напечатайте программный код процедуры-функции Function СП (Цена, Количество), представленный на рисунке 2.10, и закройте окно редактора VB.

[38] Если окно **Code** не отобразилось, задайте команду ***View ⇒ Code.***

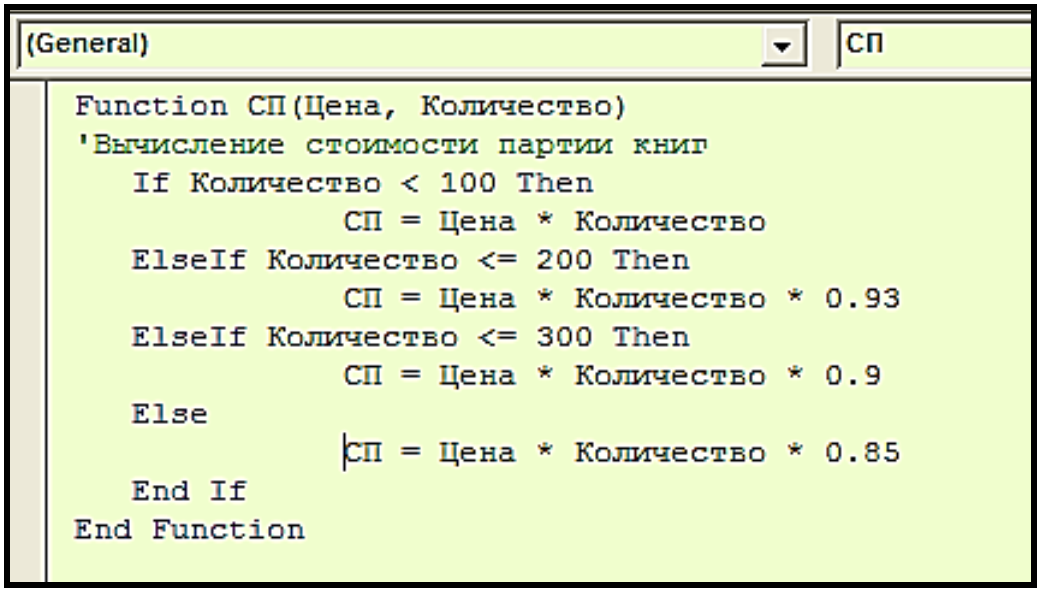


Рис. 2.10

3. Выделив на рабочем листе любую из ячеек, откройте диалоговое окно **Мастер функций** (см. рис 2.11).

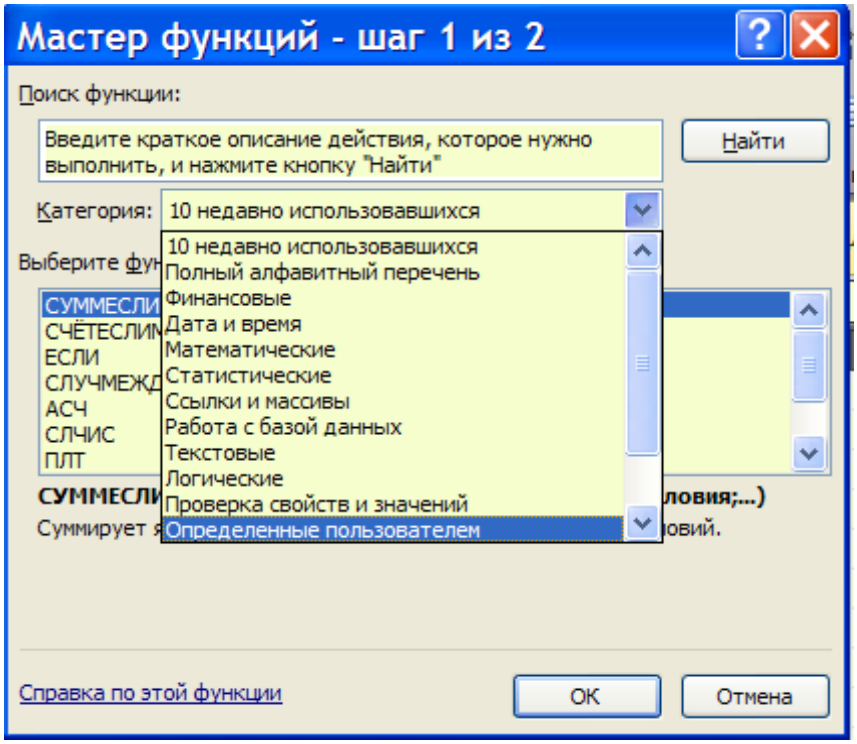


Рис. 2.11

4. Выбрав категорию «Определенные пользователем», выделите созданную функцию СП и щелкните по кнопке **ОК**.
5. Введите аргументы функции СП.

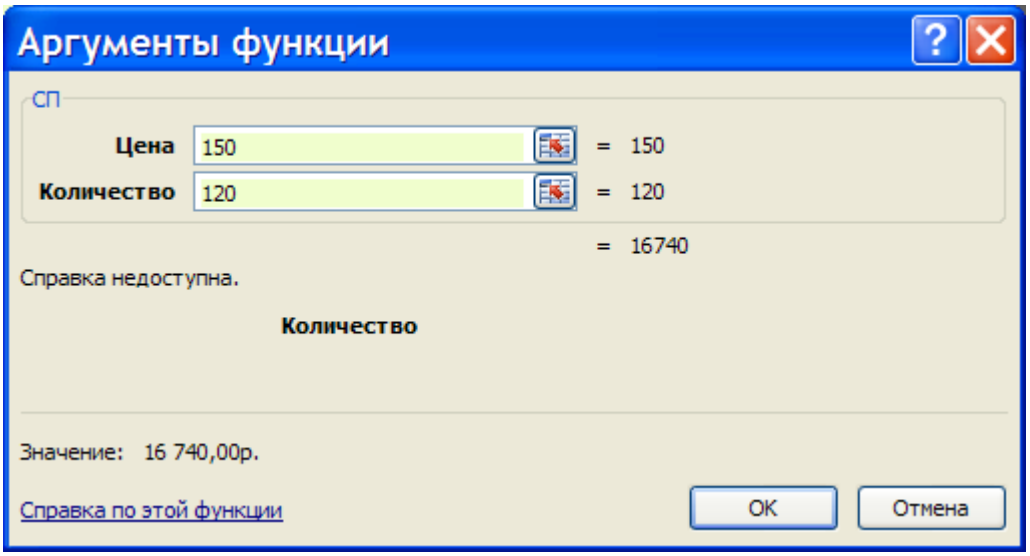


Рис.2.12

2.4.2 Оператор Select Case

Оператор Select Case позволяет осуществлять выбор одного из перечисленных в Case-блоках вариантов. Основой выбора служит значение тестируемого выражения, которое сравнивается с условиями, каждое из которых начинается с ключевого слова Case (см. рис.2.13).

Синтаксис оператора **Select Case**:

```
Select Case Выражение
        Case Условие_1
                Инструкция_1
        Case Условие_2
                Инструкция_2
        ……
        Case Условие_n
                Инструкция_n
        Case Else
                Инструкция_Else
End Select
```

Вначале оценивается тестовое выражение. Затем программа переходит к списку условий (шаблонов), начинающихся ключевым словом **Case**, и сравнивает полученное значение выражения с каждым шаблоном.

Если найдено соответствие условию, выполняется соответствующая инструкция.

Если выражение не соответствует ни одному условию, выполняется инструкция, обозначенная ключевым словом **Case Else**.

Рис.2.13

Можно создать любое количество Case-блоков. Применять же последний блок **Case Else** не обязательно.

В том случае, когда несколько условий соответствует заданному выражению, выполняется только инструкция первого выявленного блока.

Условие в Case-блоках может быть единственным, когда требуется точное соответствие, или составным из нескольких разделенных запятыми условий. К примеру, **Case 12, Is >= 20, 1 To 4** соответствует выражению, если полученное при тестировании значение этого выражения равно 12, либо оно больше или равно 20, либо попадает в диапазон от 1 до 4 включительно.

УПРАЖНЕНИЕ 19

Создадим процедуру-функцию для расчёта комиссионных выплачиваемых распространителям продукции сетевой компании. На роль обязательного аргумента создаваемой функции назначим ежемесячный объем продаж в рублях, поскольку от

Таблица 2.6

Ежемесячный объем продаж (руб)	Комиссионные (%)

него прямо зависит процент выплачиваемых комиссионных (см. таблицу 2.6).

0 - 19999	10
20000 – 39999	12,5
40000 – 49999	13,5
50000 и более	15

1. Открыв рабочую книгу с поддержкой макросов (*.xlsm), задайте команду ***Разработчик ⇒ Код⇒ Visual Basic***[39] или нажмите **Alt+F11.**
2. Задав команду ***Insert ⇒Module,*** напечатайте программный код процедуры-функции Function *Комиссионные* (*Продажи*), представленный на рисунке 2.14, и закройте окно редактора VB.
3. Выделив на рабочем листе любую из ячеек, откройте диалоговое окно **Мастер функций** (см. рис 2.11).
4. В категории «Определенные пользователем», выберите созданную функцию.

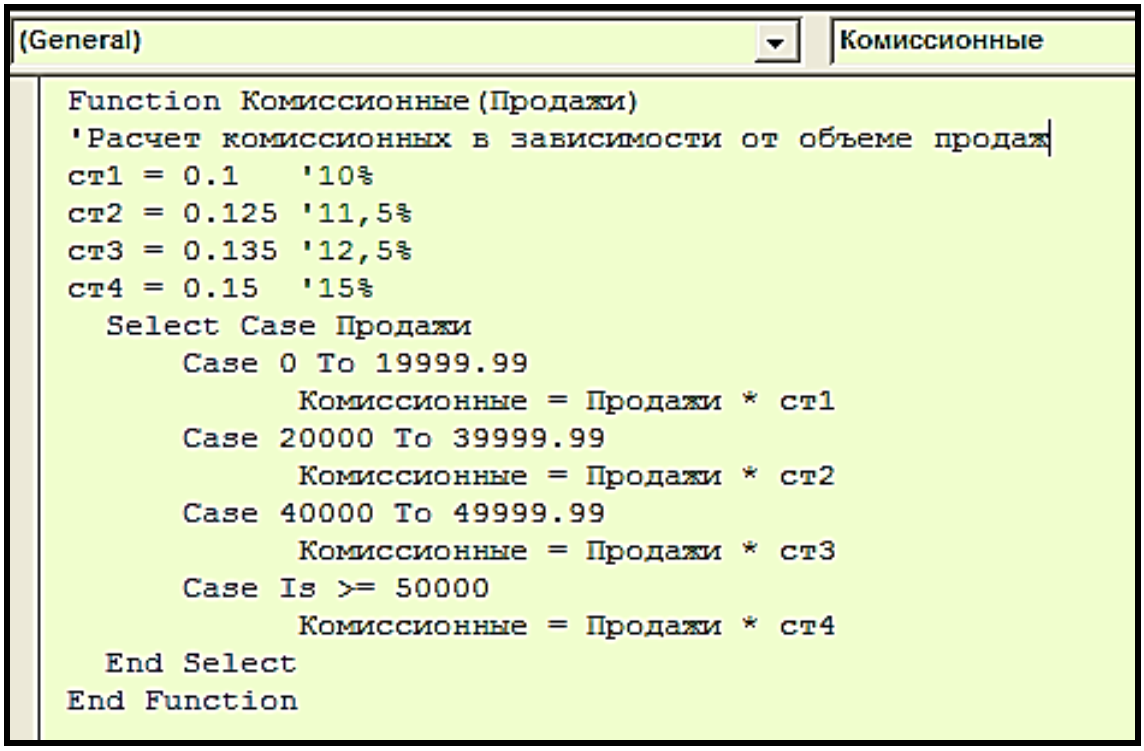


Рис. 2.14

5. Щелкнув по кнопке ОК, закройте диалоговое окно **Мастер функций**.
6. В диалоговом окне **Аргументы функции** (см. рис. 2.15) введите значение аргумента (объем продаж в текущем месяце) и, щелкнув по кнопке **ОК**, закройте его.

[39] Если окно **Code** не отобразилось, из меню редактора VB задайте команду ***View ⇒ Code***

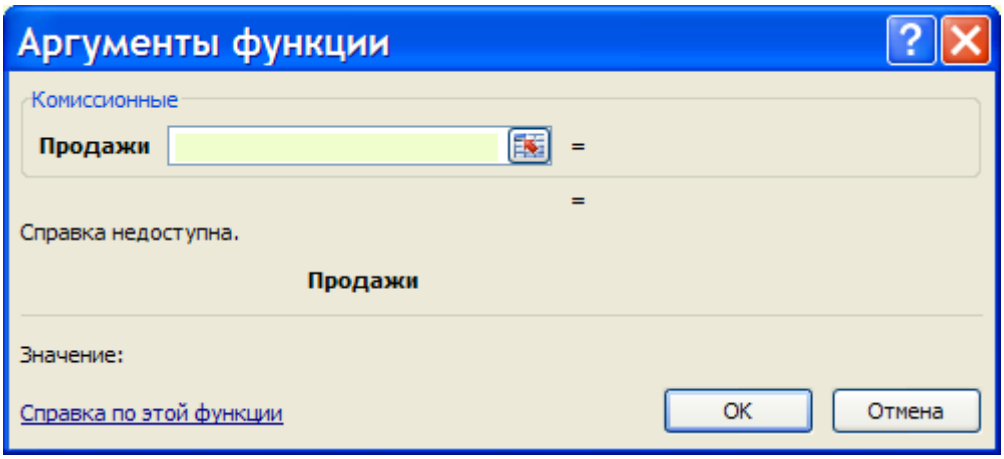


Рис. 2.15

2.4.3 Операторы цикла с условием

Операторы цикла с условием **Do Until … Loop** и **Do While … Loop** используются тогда, когда нужно многократно выполнить какое-то действие, но количество повторов заранее неизвестно.

Таблица 2.7

Синтаксис оператора	Как действует
Do ***Блок инструкций*** **Loop Until *Условие***	Выполнив один раз блок инструкций и одновременно изменив значение переменной в операции сравнения, повторяет эти действия до тех пор, пока условие не станет истинным (True).
Do Until *Условие* ***Блок инструкций*** **Loop**	Пока условие ложно (False) блок инструкций многократно повторяется. Если сразу (True), цикл не выполняется
Do ***Блок инструкций*** **Loop While *Условие***	Выполнив один раз блок инструкций и одновременно изменив значение переменной в операции сравнения, повторяет эти действия до тех пор, пока условие остаётся истинным (True).
Do While *Условие* ***Блок инструкций*** **Loop**	Пока условие истинно (True) блок инструкций многократно повторяется. Если сразу (False), цикл не выполняется

УПРАЖНЕНИЕ 20

1. Открыв любой файл с поддержкой макросов (*.xlsm), нажмите **Alt+F11.**
2. Задав команду ***Insert ⇒ Module,*** напечатайте программный код процедуры (см. рис. 2.16) и задайте команду ***Run ⇒ Sub/UserForm***.

```
(General)                                   Два_цикла_Do
Sub Два_цикла_Do()
Dim x As Integer
Do
x = Val(InputBox("Сколько столбцов на листе Excel 2010?"))
Loop Until x = 16384
MsgBox "Ответ правильный!"
Do Until x = 16390
x = x + 1
MsgBox "Выход из второго цикла Do при x=16390. " _
& "В данный момент x=" & x
Loop
End Sub
```

Рис. 2.16

3. Отвечая на вопрос в диалоговом окне, которое отобразится на экране сразу после запуска процедуры (см. рис. 2.17), несколько раз введите неправильный ответ.

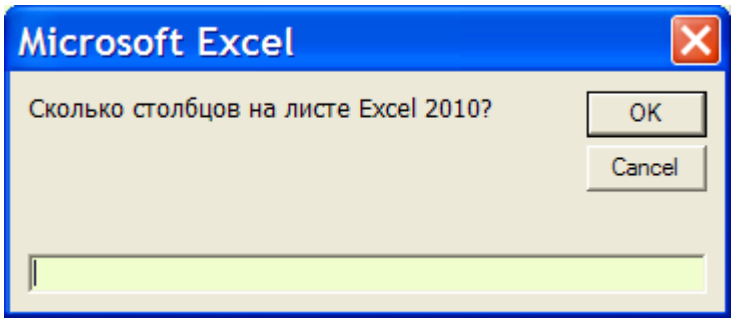


Рис. 2.17

4. В редакторе VB откройте модуль с процедурой Sub Два_цикла_Do и добавьте в него программный код процедуры, представленной на рисунке 2.18.
5. Выполнив добавленную процедуру, сделайте вывод о том, как действуют четыре представленные в таблице конструкции оператора цикла с условием.
6. Программируя циклы, следите за тем, чтобы они не оказались замкнутыми (повторялись до бесконечности). Это может происходить в том случае, когда неверно сформулировано условие цикла, или когда в процессе выполнения цикла сравниваемые в условии значения не изменились.

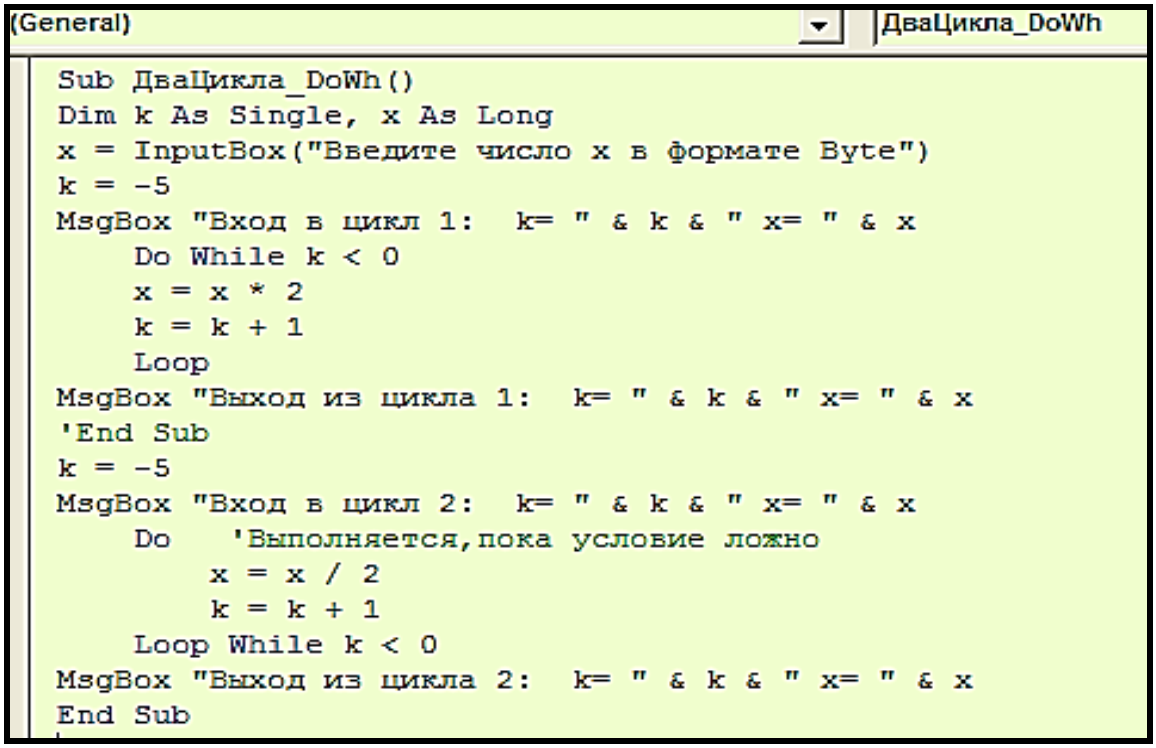


Рис. 2.18

Для прерывания бесконечного цикла:
1. Нажмите клавиши **Ctrl+Alt+Del**, чтобы открыть диалоговое окно **Диспетчер задач Windows**.
2. На вкладке **Приложения** выделите запись ***Microsoft Visual Basic***.
3. Щелкните на кнопке **Снять задачу**.

Эти действия завершают работу редактора VB. Несохраненные изменения в вашей программе при этом будут потеряны.

2.4.4 Операторы цикла со счетчиком

Оператор **For ... Next** выполняет блок инструкций заданное количество раз. Его синтаксис:

For ***счет*** **=** ***старт*** **To** ***финиш***
Блок инструкций
Next ***счет***

Здесь:

- ***счет*** — переменная типа **Integer**, используемая для отслеживания количества выполненных итераций;

• ***старт*** — начальное значение переменной *счет*;
• ***финиш*** — конечное значение переменной *счет*.

Когда в программе встречается оператор **For ... Next**, переменной присваивается значение *старт*. Затем выполняется блок инструкций, и одновременно значение переменной увеличивается[40] на единицу. До тех пор, пока значение переменной *счет* не превысит значение *финиш*, продолжается выполнение новых итераций (цикл повторяется).

[40] индекс инкрементируется

УПРАЖНЕНИЕ 21

Заполнив диапазон A1:J10 случайными числами, закрасить сто его ячеек, используя цветовую палитру из 56 индексированных цветов.

1. Откройте любой файл с поддержкой макросов (*.xlsm) и нажмите на клавиши **Alt+F11.**
2. Задав команду ***Insert ⇒Module,*** напечатайте программный код процедуры, представленный на рисунке 2.19. В этом листинге:
   - Для отображения на экране монитора числа, присвоенного переменной «a», применена функция Str(a), преобразующая числовой тип переменной в текстовой;
   - Команда ***Randomize*** запускает датчик случайных чисел;
   - Для получения целой части каждого формируемого датчиком числа применена функция Int();
   - Для перемещения по строкам диапазона ячеек на рабочем листе применена конструкция **For i=1 To N … Next i**, а по столбцам этого диапазона – вложенная в неё конструкция **For j=1 To N … Next j**;
   - Обе эти конструкции используются повторно для заливки цветом фона в каждой ячейке диапазона;
   - Цвет соответствует результату операции деления по модулю (с Mod 56), где с – случайное число в проверяемой ячейке диапазона, а 56 – максимальное значение индекса цветовой палитры.
3. Задайте команду: ***Run ⇒ Sub/UserForm***.

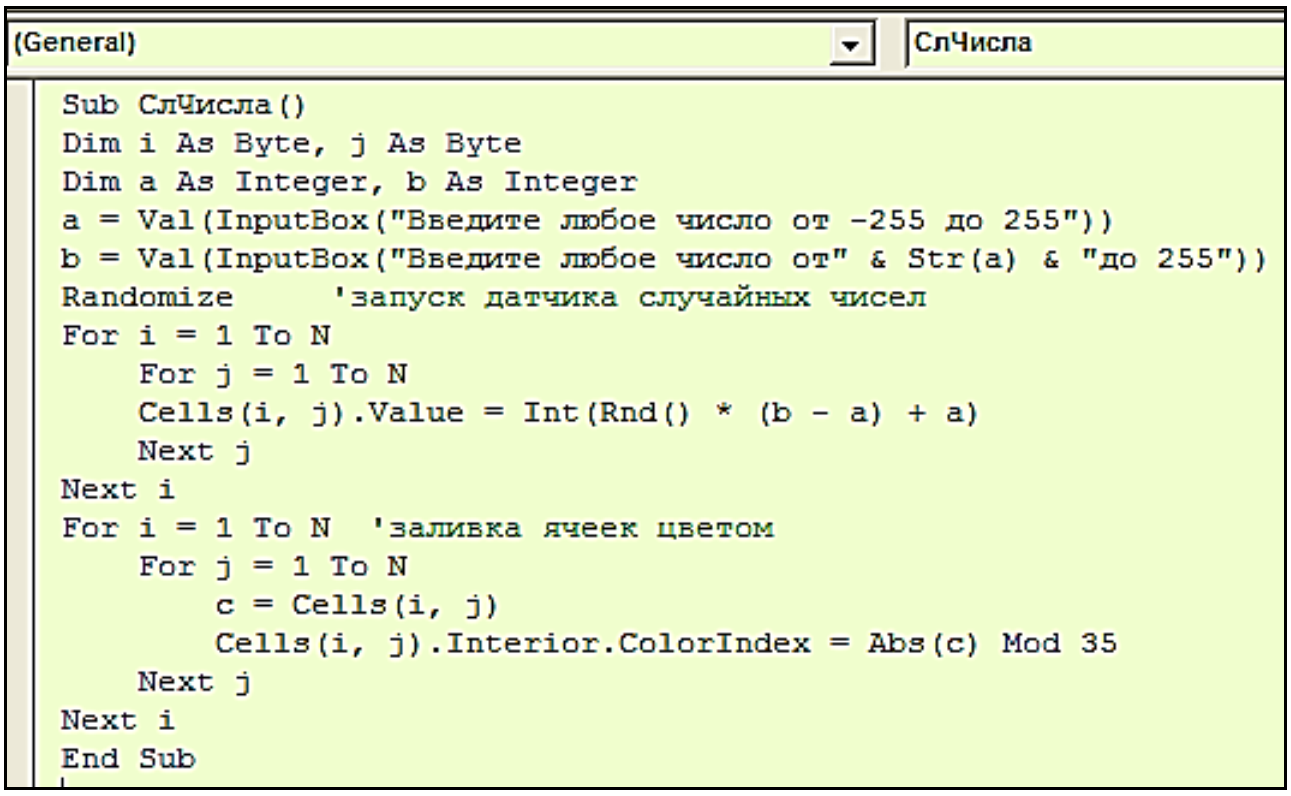


Рис. 2.19

Конструкция, позволяющая изменять значение счетчика с шагом отличным от единицы:

For ***счет*** = ***старт*** **To** ***финиш*** **Step** ***шаг***
Блок инструкций
Next ***счет***

При использовании этой конструкции следует помнить:

- Когда значение переменной *шаг* отрицательное число, значение переменной *начало* должно быть больше, чем значение переменной *финиш*;
- Если значение переменной *шаг* дробное число, то тип переменной *счет* должен быть объявлен как **Single** или **Double;**
- Если после ключевого слова **Next** не поставить переменную *счет*, VBA автоматически соотнесет его с предыдущим ключевым словом **For**.

2.5 Массивы VBA

Массив – это упорядоченная последовательность однотипных данных, имеющих общее имя. С такой коллекцией переменных можно работать как с единым целым, а для обращения к конкретному элементу коллекции достаточно указать имя массива и индекс элемента.

При назначении имен массивам рекомендуется придерживаться следующих правил:

- имя должно быть коротким, но содержательным;
- имя должно начинаться с буквы и не содержать пробелов;

- имя не должно быть похожим на адрес ячейки (например, F5);
- в имени спецсимволы и знаки пунктуации, кроме точки и подчеркивания, запрещены.

Индекс всегда заключается в круглые скобки. Количество элементов в массиве всегда конечно.

Хотя язык VBA позволяет создавать массивы, имеющие до 60 измерений, на практике используются одно и двухмерные массивы. Вектор в математике представляется в виде одномерного массива, таблица – в виде двухмерного.

Одномерный массив - это самый простой вариант массива, использующий обыкновенный список числовых или строковых данных.

Для представления таблиц на рабочих листах используются двумерные массивы данных.

По умолчанию нумерация элементов в массиве начинается с 0. Такая система нумерации довольно распространена в программировании и называется нумерацией с *нулевой базой*. Поскольку нумерация с нулевой базой не всегда удобна (мы привыкли считать раз, два, три и т.д.), в VBA имеется инструкция **Option Base**, позволяющая исправить это "неудобство":

Option Base 0 - индексы массивов начинаются с 0;

Option Base 1 - индексы массивов начинаются с 1.

Эта инструкция, размещенная в области объявлений модуля, перед объявлениями переменных и констант влияет на все массивы, объявляемые в модуле[41].

Прежде чем использовать массив, его следует объявить[42], чтобы зарезервировать данным массива место в памяти компьютера. Объявляются массивы так же, как и переменные.

Локальный массив объявляется в процедуре, а глобальный – в модуле. Рассмотрим два примера:

- одномерный строковый массив из десяти элементов объявляется так: **Dim Array (1 To 10) As String** или **Dim Array (9) As String.** Если в скобках указано единственное число, оно определяет верхнюю границу и то, что нумерация начинается с нуля. Обращение к седьмому элементу этого массива **– Array (7)** или **Array (6)**. Указывается имя массива, а в скобках – индекс этого элемента;
- **Dim Таблица (1 To 5, 1 To 3) As Single** или **Dim Таблица (4, 2) As Single –** пример объявления двухмерного числового массива из пятнадцати целых чисел $\mathbf{a_{i,j}}$, расположенных в пяти строках по три элемента в каждой. При втором варианте индекс строки i будет изменяться от 0 до 4, а индекс столбца j будет изменяться от 0 до 2. Если такое упрощенное объявление вас устраивает, а обращение к конкретному элементу затруднено, вставьте оператор **Option Base 1**.

Макрос станет более привычным и удобным для восприятия.

После объявления массива, его надо заполнить элементами. Это можно сделать, используя клавиатуру, функцию **InputBox**, датчик случайных чисел, выбрать данные из файла или напечатать одновременно с их объявлением в процедуре.

---

[41] Нельзя помещать инструкцию *Option Base* внутри процедуры.

[42] Если массив объявлен как **Variant**, он может хранить разнотипные данные

УПРАЖНЕНИЕ 22

Задача преобразования чисел в текст, возникающая при автоматизации процесса заполнения платежных поручений, решается по-разному. Известно, что в платежных поручениях денежные суммы записываются прописью. Работу с массивами рассмотрим на простом примере преобразования любого числа от 1 до 999 в текст.

Алгоритм наших действий представлен на рисунке 20. В нем использованы операторы сравнения, обычного деления и деления по модулю, а также встроенная функция **Int**(). Для пояснения этого алгоритма рассмотрим конкретные числовые примеры.

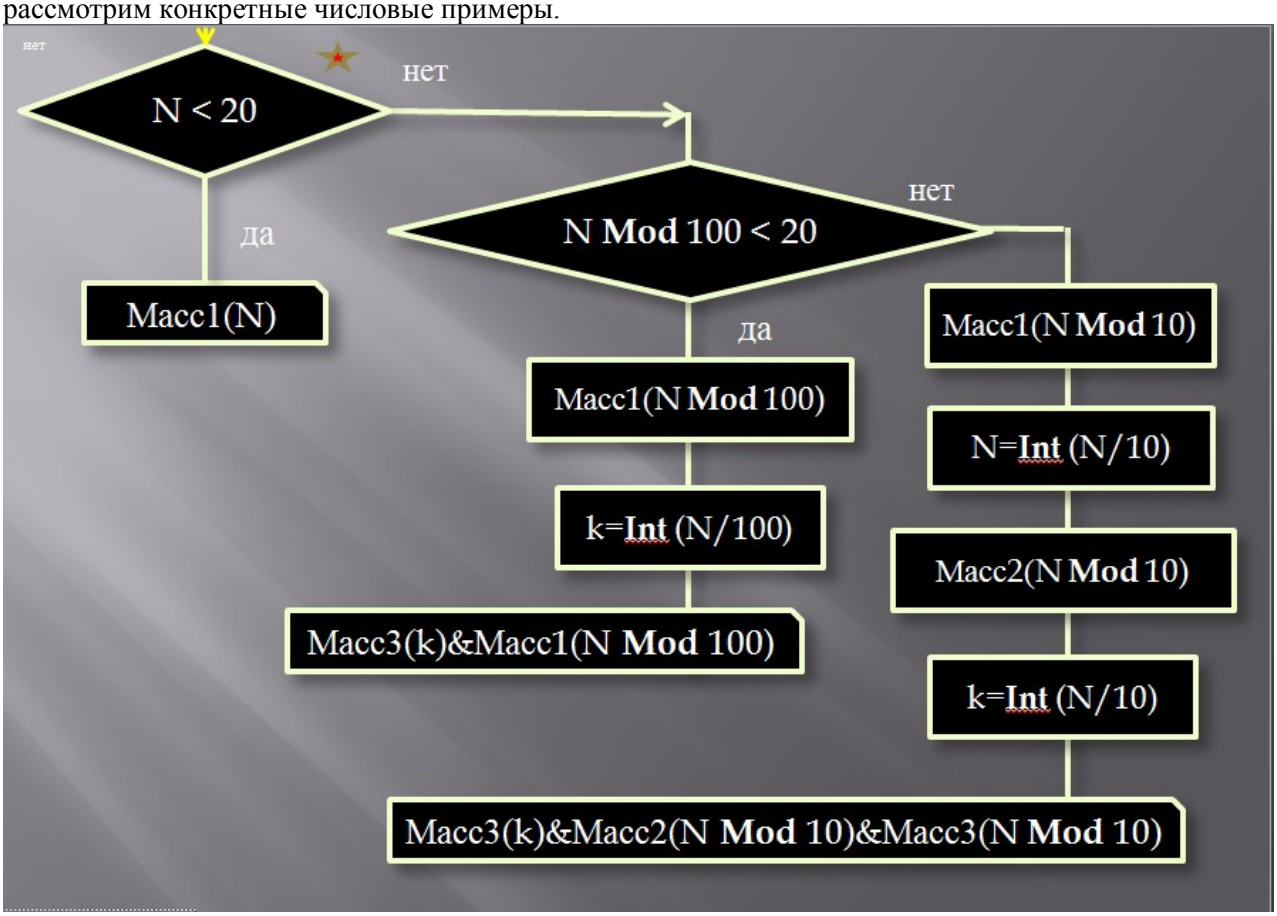


Рис. 2.20

Любое число меньше двадцати (к примеру, N=15) используется в качестве аргумента текстового массива **Macc1**, элементы которого перечислены в скобках (“ “, “один”, “два”, … , “девятнадцать”), в противном случае (например, N=915) выполняется его деление по модулю 100.

Остаток от деления равный 15 вновь сравнивается с числом 20 и, поскольку он меньше, из массива **Macc1 (0 To 19)** выбирается текст “пятнадцать”. Далее, разделив 915 на 100, получим целое число 9.

Текстовой массив **Macc3(0 To 9)** образован из следующих элементов: “ “, “сто”, ”двести”,…”девятьсот”. В результате конкатенации текстов из массивов **Macc1 (0 To 19)** и **Macc3(0 To 9)** на выходе алгоритма получим “девятьсот пятнадцать”.

Если же на вход алгоритма поступит число 925, то, выполнив операцию 925 Mod 10= 5, вы сразу получаете младший разряд прописью. Поскольку целая часть от 92,5 равна 92, то деление по модулю 10 позволяет тут же получить их массива **Macc2(0 To 9)** число 20 прописью. Далее, разделив 92 на 10, получим целое число 9 и вновь обратимся к массиву **Macc3**.

Когда алгоритм понятен, можно приступать к его реализации.

1. Создайте новую рабочую книгу с поддержкой макросов (*.xlsm).

2. Задайте команду: ***Разработчик ⇒ Код⇒ Visual Basic***[43] или нажмите на клавиши **Alt+F11.**
3. Задав команду ***Insert ⇒Module,*** напечатайте представленный ниже программный код процедуры-подпрограммы **Sub Число_прописью ()** и закройте окно редактора VB.

```
Sub Число_прописью()
Dim nn As String
Dim n As Integer, k As Integer
Масс1 = Array(" ", "один", "два", "три", "четыре", "пять", "шесть", "семь", _
"восемь", "девять", "десять", "одиннадцать", "двенадцать", "тринадцать", _ "четырнадцать", "пятнадцать",
"шестнадцать", "семнадцать", _
 "восемнадцать", "девятнадцать")
Масс2 = Array(" ", " ", "двадцать ", "тридцать", "сорок", "пятьдесят", _
"шестьдесят ", "семьдесят", "восемьдесят", "девяносто")
Масс3 = Array(" ", "сто", "двести", "триста", "четыреста", "пятьсот ", _
"шестьсот", "семьсот", "восемьсот", "девятьсот")
n = Val(InputBox("Введите любое целое число от 1 до 999"))
    If n < 20 Then
          Cells(1, 1) = Масс1(n) 'двухзначные числа от 1 до 19
    Else
        k = n Mod 100    ' остаток от деления трехзначного числа
            If k < 20 Then
               nn = Масс1(k)   '
               n = Int(n / 100)
            Else
               nn = Масс1(n Mod 10) 'младший разряд трёхзначного числа
               n = Int(n / 10)
               nn = Масс2(n Mod 10) + nn  'два разряда прописью
               n = Int(n / 10)
            End If
        Cells(1, 1) = Масс3(n Mod 10) + nn  ' число прописью
    End If
    End Sub
```

4. Задайте команду: ***Разработчик ⇒ Код ⇒Макросы.***
5. Найдите созданную процедуру и проверьте ее работу, задавая любые трехзначные целые числа.

---

[43] Если окно «Code» не отобразилось, из меню редактора VB задайте команду ***View ⇒ Code***

УПРАЖНЕНИЕ 23

1. Нажмите на клавиши **Alt+F11.**
2. Задав команду ***Insert ⇒ Module,*** напечатайте представленный ниже программный код процедуры-функции **Function Число_прописью (n)**, и закройте окно редактора VB.

```
Function Число_прописью(n) As String
Масс1 = Array(" ", "один", "два", "три", "четыре", "пять", "шесть", "семь", _
"восемь", "девять", "десять", "одиннадцать", "двенадцать", "тринадцать", _
 "четырнадцать", "пятнадцать", "шестнадцать", "семнадцать", _
 "восемнадцать",  "девятнадцать")
Масс2 = Array(" ", " ", "двадцать", "тридцать", "сорок", "пятьдесят ", _
"шестьдесят ", "семьдесят", "восемьдесят", "девяносто")
Масс3 = Array(" ", "сто", "двести", "триста", "четыреста", "пятьсот ", _
"шестьсот ", "семьсот ", "восемьсот ", "девятьсот ")
    If n < 20 Then
    Число_прописью = Масс1(n)
    Else
        If n Mod 100 < 20 Then
            nn = Масс1(n Mod 100)
            n = Int(n / 100)
        Else
            nn = Масс1(n Mod 10)
            n = Int(n / 10)
            nn = Масс2(n Mod 10) + nn
            n = Int(n / 10)
        End If
    nn = Масс3(n Mod 10) + nn
    Число_прописью = nn
    End If
    End Function
```

3. Выделив ячейку на рабочем листе, откройте диалоговое окно **Мастер функций**, найдите в категории **Определенные пользователем** созданную функцию и запустите ее.
4. Введите в диалоговое окно **Аргументы функции** изображенное на рисунке 2.21 любое трехзначное число.

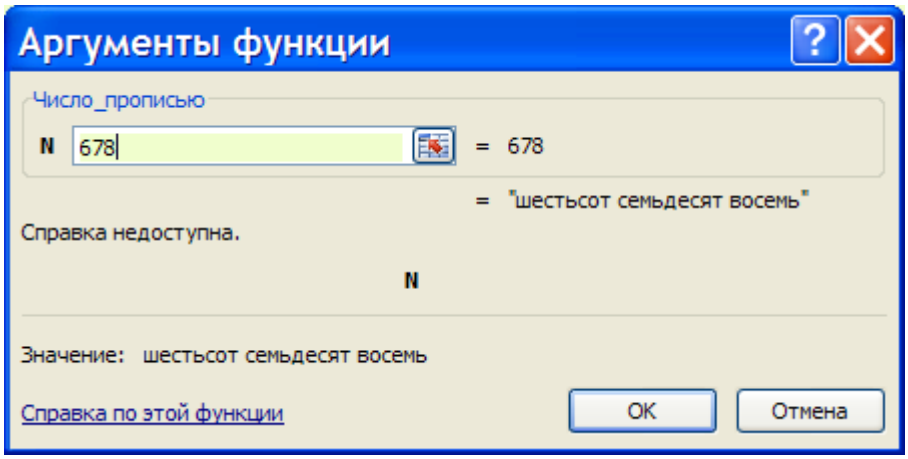


Рис. 2.21

Итак, познакомившись с основными компонентами языка программирования VBA и выполнив очередную порцию упражнений, вы узнали:

- Язык VBA поддерживает переменные следующих типов – Byte, Integer, Long, Single, Double, Currency, Boolean, Date, String и Variant[44].
- Данные, изменяющиеся в процессе выполнения процедур, хранятся в переменных, которые следует объявлять.
- Процедуры бывают двух типов – подпрограммы, которые выполняются самостоятельно, и функции, возвращающие вычисленные значения.
- Каждая переменная и константа характеризуется областью видимости (процедура, модуль или проект)
- Порядок выполнения инструкций можно изменять с помощью управляющих операторов – **If … Then, Select Case, Do … Loop, For … Next** и **For Each … Next.**
- Массивы VBA незаменимы при работе с большими объемами однотипных данных.

[44] Переменные типа Object в данном пособии не упоминались

Глава 3. Создание нестандартных форм и диалоговых окон

Помимо команд на вкладках ленты и стандартных диалоговых окон, пользователь для взаимодействия с приложением Excel может применять созданные им на рабочих листах формы[45] и диалоговые окна UserForm.

Формы на рабочих листах, как правило, проектируют и создают вручную. Для превращения рабочего листа в пользовательскую форму надо диапазоны ячеек отформатировать в диалоговом окне **Формат ячеек**, которое вызывается на экран по команде ***Главная ⇒ Ячейки ⇒ Формат ⇒ Формат ячеек.***

На созданной таким образом форме можно разместить элементы управления (кнопки, переключатели, списки и др.). Доступ к двум группам элементов управления (см. рис. 3.1) открывает команда ***Разработчик ⇒ Элементы управления (ЭУ) ⇒ Вставить.***

В группу «Элементы управления формы» входят ЭУ, которые использовались в старых версиях (выпуск до 1997 года) приложения Excel.

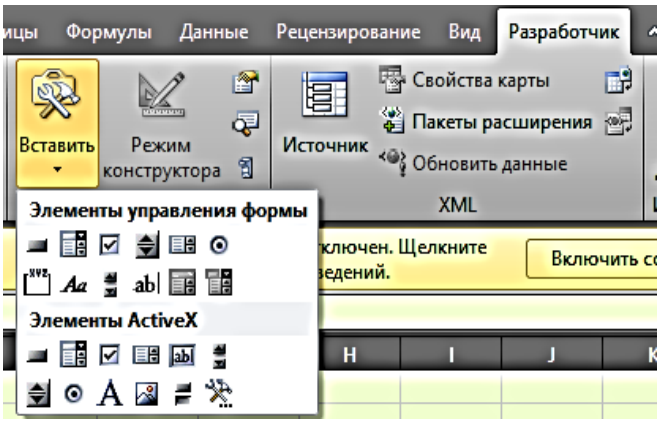


Рис 3.1

Так называемые старые элементы управления формы в настоящее время применяются только на рабочих листах и на диаграммах. Новые[46] элементы ActiveX можно использовать как для создания диалоговых окон UserForm, так и для создания форм на рабочих листах.

Изображенная на рисунке 3.2 готовая форма платежного поручения – всего лишь отформатированный рабочий лист. Если на этот же лист поместить таблицы с реквизитами плательщика и получателей, а в окне **Code** написать

[45] Пользовательская форма – это всего лишь отформатированный рабочий лист с размещенными на нем элементами управления.
[46] Элементы ActiveX поддерживаются Excel с 1997 года.

программный код процедуры события **Click**, то работа по заполнению подобного рода бланков сведется к щелчку по кнопке.

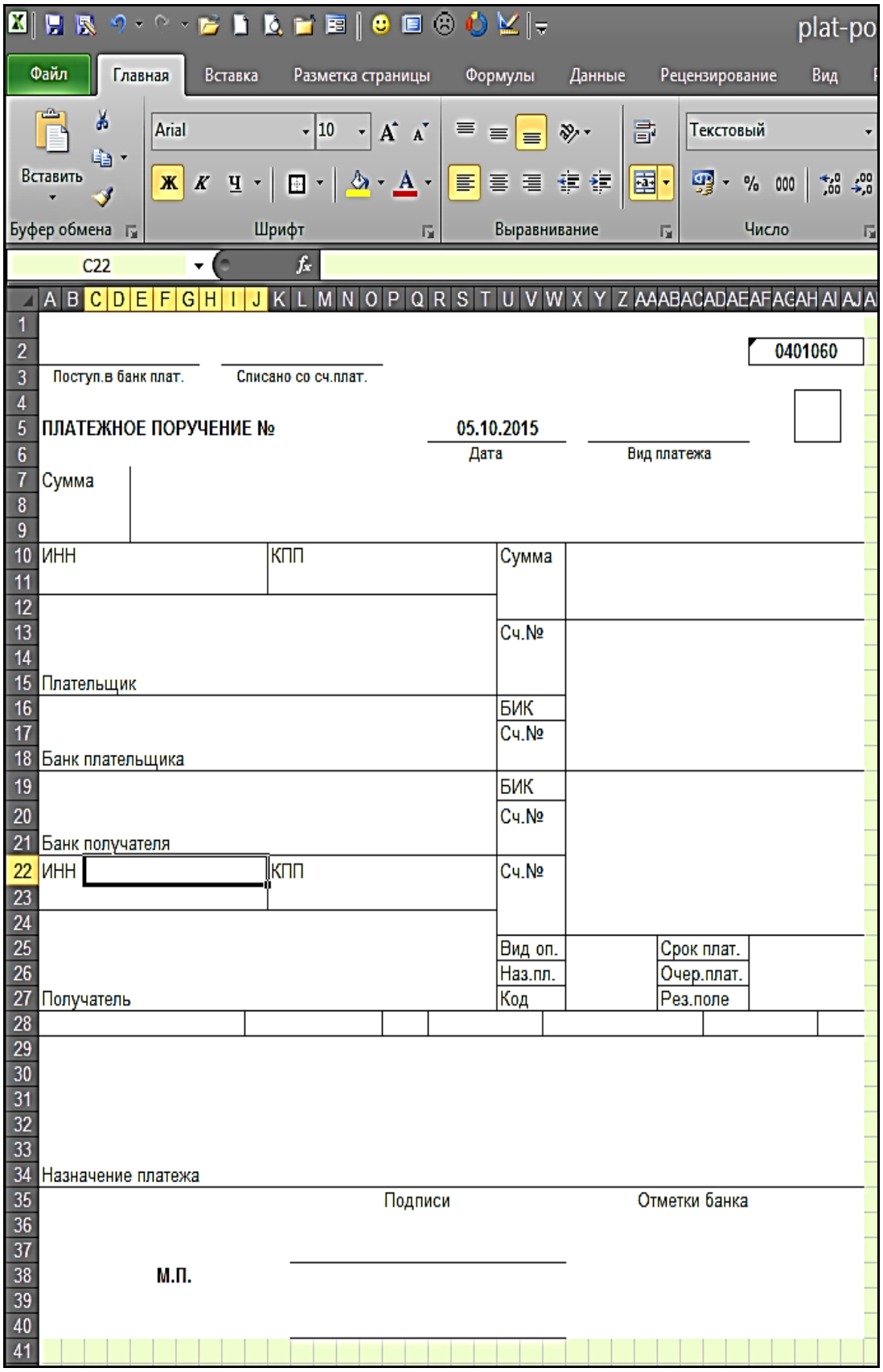


Рис. 3.2

Чтобы почувствовать разницу между элементами управления формы и элементами ActiveX, представленными на рисунке 3.1, поместите на рабочий лист две кнопки: «старую» (**Кнопка**) и новую (**CommandButton).**

Для размещения на рабочем листе кнопки из верхней группы щелкните по ней и очертите курсором мыши ее контур. Появится диалоговое окно «Назначить макрос объекту» (см. рис. 3.3), в котором надо указать имя процедуры, выполняемой при щелчке по этой кнопке. Закрыв окно, вы увидите изображение кнопки с надписью “Кнопка1” (см. рис. 3.4).

Для размещения на том же листе элемента ActiveX **CommandButton** щелкните по изображению кнопки в нижний группе и курсором мыши очертите прямоугольный контур. Появится изображение кнопки с надписью ”CommandButton1” (см. рис. 3.4).

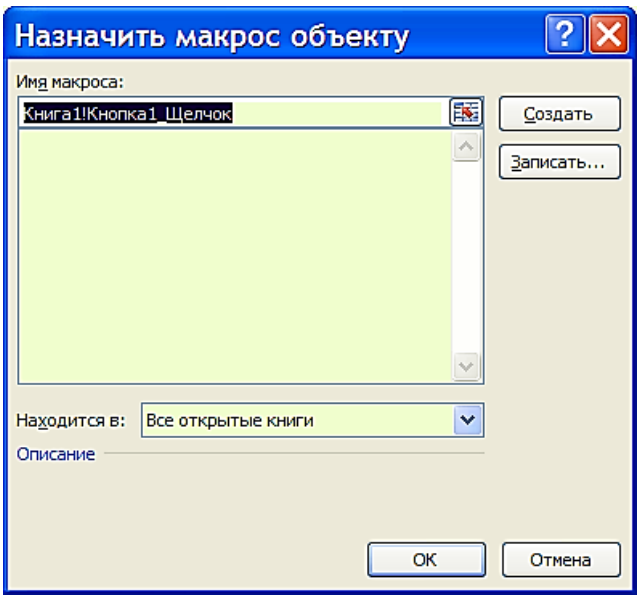


Рис. 3.3

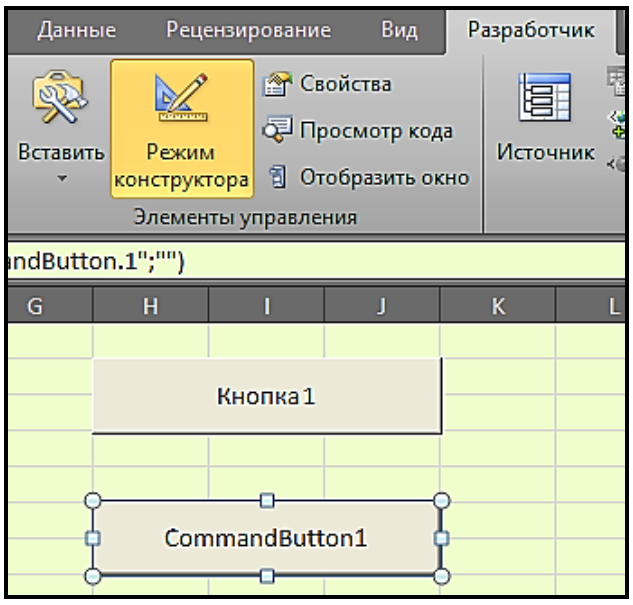


Рис. 3.4

Для взаимодействия с элементами управления ActiveX с целью изменения их свойств используется режим конструктора. В этом режиме подсвечена кнопка **Режим конструктора**[47] и отключен запуск событий, связанных с этими элементами.

[47] При щелчке по этой подсвеченной кнопке режим конструктора отключается.

С элементом **CommandButton** связано единственное событие **Click**, для обработки которого необходимо написать в окне **Code** программный код процедуры. Окно, показанное на рисунке 3.5, появляется на экране по команде ***Разработчик ⇒ Элементы управления⇒ Просмотр кода.***

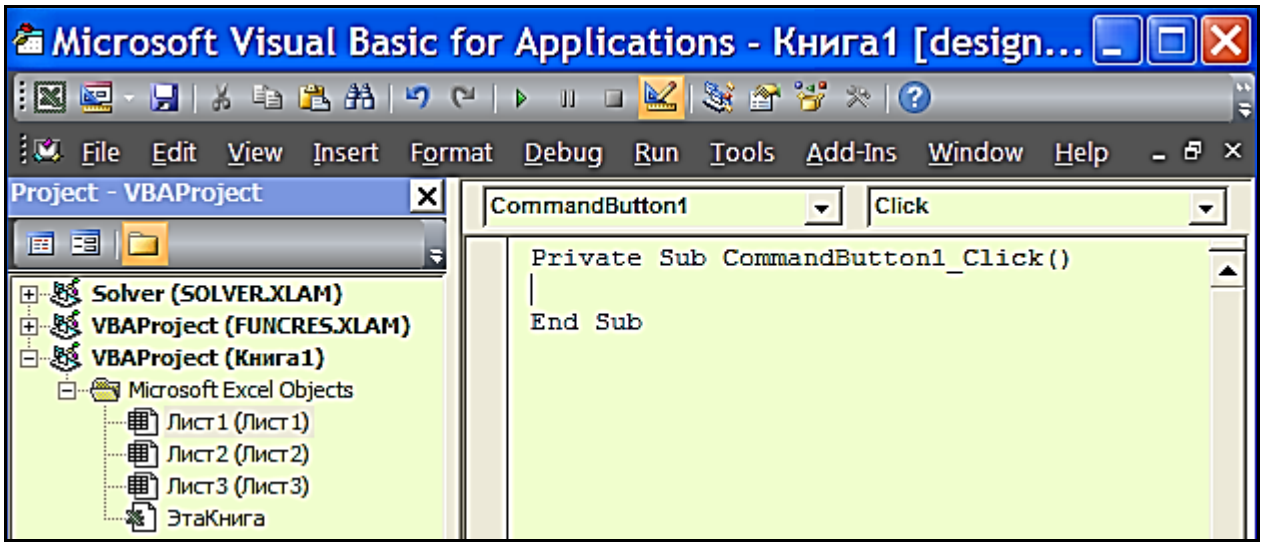


Рис. 3.5

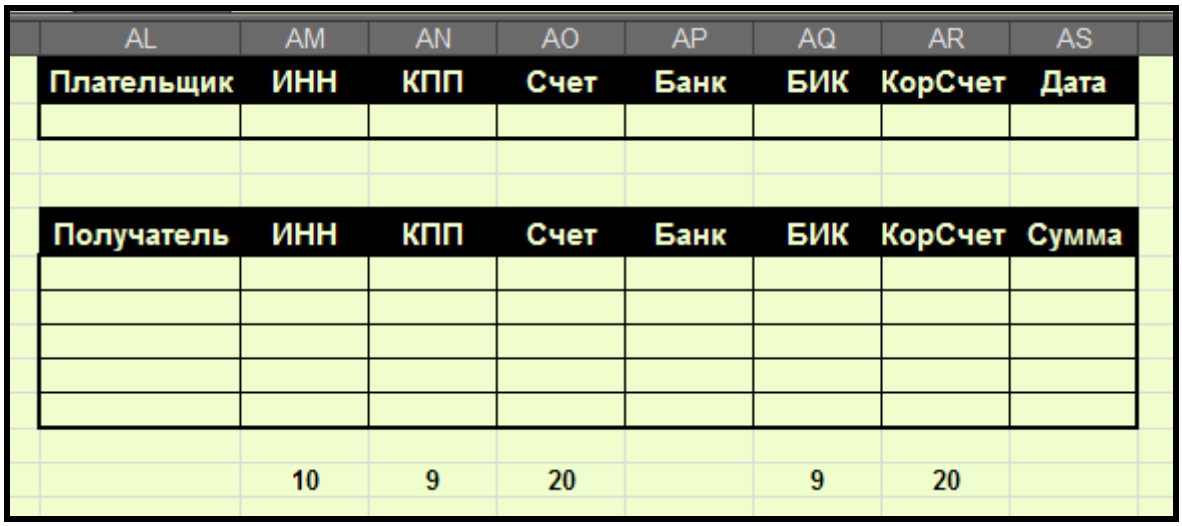
3.1 Автоматизация процесса заполнения стандартной формы

УПРАЖНЕНИЕ 24

1. Скачайте из Интернет[48] бланк платежного поручения в формате Excel.
2. Скопируйте бланк на один из листов открытой рабочей книги.
3. Постройте на листе с готовой формой[49] две таблицы с исходными данными (см. рис. 3.6). Таблицу «Плательщик» разместите в диапазоне AL1:AS2, а таблицу «Получатели» – в диапазоне AL5:AS10.

[48] Например, http://lugasoft.ru/blank/platezhnoe-poruchenie

[49] Можно создать таблицы на любом другом листе, но так нагляднее



AL	AM	AN	AO	AP	AQ	AR	AS
Плательщик	**ИНН**	**КПП**	**Счет**	**Банк**	**БИК**	**КорСчет**	**Дата**
Получатель	**ИНН**	**КПП**	**Счет**	**Банк**	**БИК**	**КорСчет**	**Сумма**
	10	9	20		9	20	

Рис.3.6

4. Заполните обе таблицы (разрядность чисел указана под столбцами).
5. Щелчками по форме определите и запишите адреса ячеек, которые должны быть заполнены данными из этих таблиц (на рис. 3.2 видно, что выделена ячейка C22, куда следует поместить ИНН получателя).
6. Щелкнув по любой ячейке рабочего листа, нажмите Alt+F11 и задайте команду ***View⇒ Immediate Window***
7. В диалоговом окне «Immediate» (см. рис. 3.7) напечатайте вопрос в таком виде: **? ActiveCell.Row**. Редактор VB воспринимает его так : «Какой индекс у строки, где находится объект активная ячейка»?

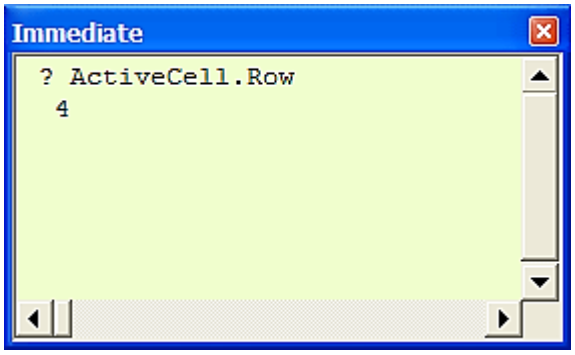


Рис. 3.7

8. Нажмите на клавишу Enter.
9. Щелкните по изображению кнопки в группе «Элементы ActiveX», курсором мыши очертите под таблицами прямоугольный контур. Появится изображение кнопки с надписью ”CommandButton1”.
10. Двойным щелчком по созданной кнопке откройте окно **Code** и напечатайте в нем текст программного кода (размещение реквизитов конкретного Получателя в ячейки бланка платежного поручения) отклика на событие **Click** (щелчок) по кнопке **CommandButton1:**

```
    Sub CommandButton1_Click()
Range("A24").Value = Cells(ActiveCell.Row, 38).Value ‘ Получатель
Range("C22").Value = Cells(ActiveCell.Row, 39).Value   'ИНН
Range("M22").Value = Cells(ActiveCell.Row, 40).Value   'КПП
Range("Y22").Value = Cells(ActiveCell.Row, 41).Value   'Счет
Range("A19").Value = Cells(ActiveCell.Row, 42).Value   'Банк
Range("Y19").Value = Cells(ActiveCell.Row, 43).Value   'БИК
Range("Y20").Value = Cells(ActiveCell.Row, 44).Value   'КорСчет
Range("E7").Value = Cells(ActiveCell.Row, 45).Value    ‘Сумма
    End Sub
```

11. Аналогичным образом (см. пункты 9 и 10 данного упражнения), создав кнопку **CommandButton2,** напечатайте в окне **Code** текст программного кода (размещение реквизитов Плательщика в ячейки бланка платежного поручения) отклика на событие **Click** (щелчок) по второй кнопке:

```
    Sub CommandButton2_Click()
Range("A12").Value = Cells(2, 38).Value 'Плательщик
Range("C10").Value = Cells(2, 39).Value 'ИНН
Range("M10").Value = Cells(2, 40).Value 'КПП
Range("Y13").Value = Cells(2, 41).Value 'Счет
Range("A16").Value = Cells(2, 42).Value 'Банк
Range("Y16").Value = Cells(2, 43).Value 'БИК
Range("Y17").Value = Cells(2, 44).Value 'КорСчет
Range("R5").Value = Cells(2, 45).Value  'Дата
    End Sub
```

12. В режиме конструктора, выделив поочередно управляющие кнопки, задайте команду ***Разработчик ⇒ Элементы управления ⇒ Свойства*** и в окне «Properties» (см. рис. 3.10) измените значение свойства **Caption** (у **CommandButton1** на Плательщик, а у **CommandButton2** на Получатель).

13. Щелчком по подсвеченной кнопке **Режим конструктора** отключите этот режим и, выделив любого Получателя в нижней таблице на рабочем листе, вставьте его реквизиты в бланк.

14. Вставьте в бланк реквизиты Плательщика.

3.2 Конструирование пользовательских форм

Нестандартные формы и диалоговые окна, создаваемые пользователем, упрощают ввод и обработку исходных данных в процессе решения конкретной задачи (например, расчет платежей по кредитам и ипотеке).

Прежде всего для реализации функциональных возможностей проектируемой формы или диалогового окна нестандартного вида необходимо, выбрав нужные элементы управления ActiveX (кнопки, переключатели, списки и др.), разместить их в первом случае на рабочем листе, а во втором – на изображенном на рисунке 3.8 объекте UserForm1[50].

Поскольку любой элемент управления можно связать с ячейкой рабочего листа, на создание формы затрачивается меньше усилий, чем на конструирование диалогового окна. Например, если сгруппировать несколько переключателей ObtionButton, то в ячейке, связанной с активизированной опцией, отобразится булевское значение True, а в остальных Folse.

Кроме вкладки **Разработчик**, элементы ActiveX размещены на панели **Toolbox** (см. рис 3.8), которая появляется в редакторе VB после задания команды ***Insert⇒ UserForm.*** С элементами ActiveX, как и с другими объектами Excel, связаны наборы свойств, методов и событий.

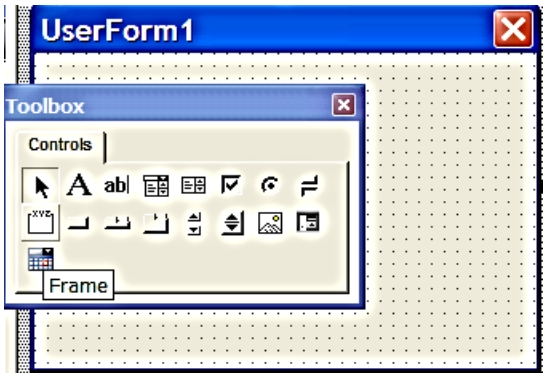


Рис. 3.8

Назначение этих элементов управления и краткое описание свойств, методов и событий тех из них, которые чаще всего применяются пользователями при конструировании нестандартных форм и диалоговых окон UserForm, **приведено ниже.** Имена элементов ActiveX представлены на рисунке 3.9.

SelectObjects (Выбор объектов)	**ListBox** (Список)

[50] Подложку нестандартного диалогового окна в других публикациях называют по-разному: формой, экранной формой или пользовательской формой

. **Label** (Надпись)

. **TextBox** (Текстовое поле)

. **ComboBox** (Поле со списком)

. **CheckBox** (Флажок)

. **ObtionButton** (Переключатель)

. **ToggleButton** (Выключатель)

0 1 2 3 4 5 6

Frame (Рамка**)**

. **CommandButton** (Копка)

0. **TabStrip** (Вкладки**)**

1. **MultiPage** (набор страниц)

2. **ScrollBar** (Полоса прокрутки)

3. **SpinButton** (Счетчик)

4. **Image** (Изображение)

5. **RefEdit** (ссылка на лист)

6.

Рис. 3.9

Поскольку элементы управления ActiveX более гибкие, чем изображенные на рисунке 3.1 старые элементы управления формы, по возможности используйте их.

Если, выделив кнопку **CommandButton1**, задать команду ***Разработчик*** ⇒ ***Элементы управления*** ⇒ ***Свойства,*** на экране отобразится диалоговое окно «Properties», с помощью которого удобно настраивать параметры элементов управления ActiveX (см. рис. 3.10).

Настройку параметров можно выполнять на вкладке **Alphabetic**, там они перечислены в алфавитном порядке, или на вкладке **Categorized**, где они сгруппированы по категориям.

3.2.1 Свойства и события, присущие всем элементам ActiveX

Name — имя элемента управления ActiveX, под которым он значится в программном коде процедур.

AutoSize — если значение равно True, элемент управления ActiveX (ЭУ) автоматически уменьшается до размеров, минимально необходимых для отображения текста, присвоенного свойству Caption.

Enabled — если значение равно True, ЭУ доступен для взаимодействия с пользователем. В противном случае он не находится в фокусе ввода и не может принимать сигналы ни от мыши, ни от клавиатуры[51].

Font — параметр, задающий гарнитуру шрифта, используемого для отображения текста в элементе управления ActiveX.

Left, Top, Width и **Height** — координаты левого верхнего угла элемента управления ActiveX и его геометрические размеры.

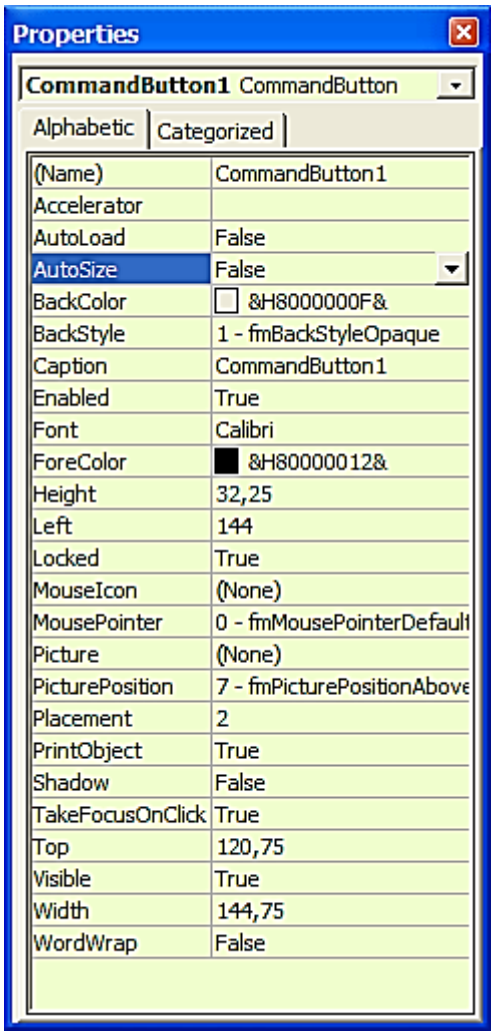


Рис. 3.10

Locked — если значение равно True, редактирование ЭУ в режиме конструктора заблокировано.

Большинство элементов ActiveX реагирует на такие события:

- **Click** — щелчок мышью;
- **DblClick** — двойной щелчок по элементу управления;

[51] В любой момент времени фокус может быть только на одном объекте.

- **KeyPress** — нажатие на любую клавишу клавиатуры (кроме F1—F12, Ctrl, Shift, Alt и клавиш управления курсором), когда элемент управления находится в фокусе ввода;
- **GotFocus**, **LostFocus** — смещение фокуса ввода на элемент управления или с него.

3.2.2 Элементов ActiveX, используемые в нестандартных формах и диалоговых окнах

1. **Label** (надпись) используется для отображения поясняющих надписей у элементов ActiveX, не обладающих свойством Caption.

   Важнейшее свойство — **Caption** (текст надписи).

2. **CommandButton** (кнопка), как правило, применяется для запуска Sub-процедур. Важнейшие свойства:
   - **Caption** — текст на кнопке;
   - **Picture** — рисунок на кнопке.

Важнейшее событие – **Click** (щелчок мышью)**.**
Методы этого элемента управления применяются редко.

3. **Checkbox** (флажок) позволяет пользователю выбрать один или несколько параметров. Флажок обычно имеет два состояния: установленное (True) и сброшенное (False), но его можно настроить на работу с выбором из трех альтернатив, добавив неопределенное состояние (Null).

Основные свойства:

- **Caption** — текст, отображаемый рядом с флажком;
- **TripleState** — если значение равно True флажок настроен на работу с выбором из трех альтернатив, в противном случае — только два;
- **Value** — задает и отображает состояния флажка;
- **LinkedCell** — позволяет автоматически сохранять значение свойства **Value** в указанной пользователем ячейке рабочего листа. Значением этого свойства является адрес ячейки (например, “B5” или “Лист2!G5”).

4. **OptionButton** (переключатель) функционально похож на элемент **CheckBox**. Эта опция так же имеет два состояния — установлено (True) и сброшено

(False). Однако, объединив несколько переключателей, вы сможете выбрать одно значение из нескольких взаимоисключающих значений данных, поскольку, когда один переключатель в группе устанавливается, остальные автоматически сбрасываются.

По умолчанию все переключатели на рабочем листе входят в одну группу. Для работы с несколькими параметрами переключатели необходимо объединить в несколько групп, задав одинаковое значение свойства **GroupName** для всех переключателей в каждой группе. Желательно при этом каждую поименованную таким образом группу окантовать рамкой.

Основные свойства:

- **Caption** — текст, отображаемый рядом с переключателем;
- **GroupName** — название группы, к которой принадлежит переключатель[52];
- **Value** — задает или возвращает состояние переключателя;
- **LinkedCell** — позволяет автоматически сохранять свойство Value переключателя в ячейке (при каждом изменении положения переключателя корректируется и значение в ячейке).

5. ToggleButton **(выключатель) действует идентично элементу** Checkbox**, но на экране монитора его состояние отображается иначе. Он выглядит как нажатая или отжатая кнопка. К этим двум положениям может быть добавлено третье — неопределенное.**

Основные свойства:

- Caption **— текст, отображаемый на кнопке выключателя;**
- Picture **— рисунок, отображаемый на кнопке выключателя;**
- Value **— задает и отображает состояние выключателя. Значение True означает, что выключатель нажат, False — отпущен, Null — положение не определено;**
- TripleState **— если значение этого свойства равно True, выключатель имеет три возможных положения, определяемые значениями True, False и Null;**

[52] Переключатели, помещенные на один рабочий лист, по умолчанию объединяются в группу, а их свойствам GroupName присваивается имя рабочего листа.

- LinkedCell **(связанная ячейка) позволяет автоматически сохранять значение параметра Value в ячейке с указанным адресом. Важно, что при каждом изменении положения выключателя корректируется значение в связанной с ним ячейке, а при редактировании содержимого ячейки меняется и состояние выключателя.**

6. **TextBox** (текстовое поле) используется для ввода текста или исходных данных в текстовом формате, которые после преобразования в числовой формат используются в процедуре. В этом же поле отображаются результаты выполненных расчетов.

   Основные свойства:

   - Text **— вводимый текст;**
   - LinkedCell **— связанная с текстовым полем ячейка рабочего листа;**
   - MultiLine **— если значение равно True, отображается текст из нескольких строк.**

7. **ListBox** (список) позволяет пользователю выбрать один или несколько вариантов из списка. Доступ ко всему списку обеспечивает полоса прокрутки на его правой границе.

   Основные свойства:

   - **List** — список строк;
   - **Value** – позиция выбранной строки;
   - **ListCount** — количество строк списка;
   - **ListFillRange** — диапазон рабочего листа, с которым связан список;
   - **LinkedCell** — ячейка рабочего листа, связанная со свойством Value.

Важнейшие метод Clear удаляет из списка все элементы, в результате чего он становится пустым. **Важнейшее событие Click** применяется для реагирования на выделение элемента щелчком мыши.

8. ComboBox **(поле со списком)** сочетает в себе функциональные возможности списка ListBox и текстового поля TextBox. В отличие от ListBox, в элементе управления ComboBox отображается только один элемент списка, зато он позволяет вводить значение, используя поле ввода, как это делается в элементе управления TextBox.

Свойства объекта ComboBox в значительной степени совпадают со свойствами объекта ListBox, включая List, ListCount, ListFillRange, и Value.

9. ScrollBar **(полоса прокрутки) и** SpinButton **(счетчик) применяются для подбора численных значений параметров. Из-за наличия бегунка, который можно перетаскивать мышью, работать с полосой прокрутки удобнее.**

Основные свойства:

- Max **— максимальное значение (целое положительное число или нуль), выдаваемое полосой прокрутки или счетчиком;**
- **Min** — минимальное значение (целое положительное число или нуль меньшее, чем значение параметра Max), выдаваемое полосой прокрутки или счетчиком;
- **SmallChange** — целое число, равное шагу изменения значения полосы прокрутки или счетчика при щелчке по стрелкам;
- **LargeChange** —целое число, равное шагу изменения значения при щелчке по самой полосе прокрутки[53];
- **Value** — текущее значение элемента управления ScrollBar или SpinButton;
- **LinkedCell** — адрес ячейки на рабочем листе, с которой связано свойство Value.

Примечание: Ограничения на значения параметров SmallChange и LargeChange можно обойти, разделив значение, выдаваемое полосой прокрутки или счетчиком, на нужное число.

10. Элемент **Image** (изображение) применяют для отображения рисунков растрового типа. У него множество свойств, таких как кадрирование, масштабирование, распределение по листу и др.

11. Элемент **Frame** (рамка) используется в диалоговых окон User Form для группирования элементов управления, но не применяется на рабочих листах**.** Там и на диаграммах следует использовать аналогичный Frame элемент управления формы **Рамка.**

[53] . У счетчика этого свойства нет

УПРАЖНЕНИЕ 25

Создать форму, которая поможет найти оптимальное решение, связанное с приобретением квартиры по разумной цене.

Дано:

- первый взнос у разных застройщиков равняется 15%, 20% и 25% от стоимости недвижимости;
- банки готовы предоставить кредит на 20 или 30 лет из расчета от 12% до 25% годовых.

Требуется, вычислив размер периодических платежей, определить приемлемую для семейного бюджета стоимость недвижимости.

1. Для конструирования формы «Ипотека» потребуется полоса прокрутки **ScrollBar** и пять переключателей **OptionButton**, которые надо разбить на две группы.
2. Один из возможных вариантов схемы расположения элементов ActiveX на рабочем листе представлен на рисунке 3.11. Там же показан диапазон H2:I16 с ячейками рабочего листа, связанными с шестью элементами управления формы «Ипотека».

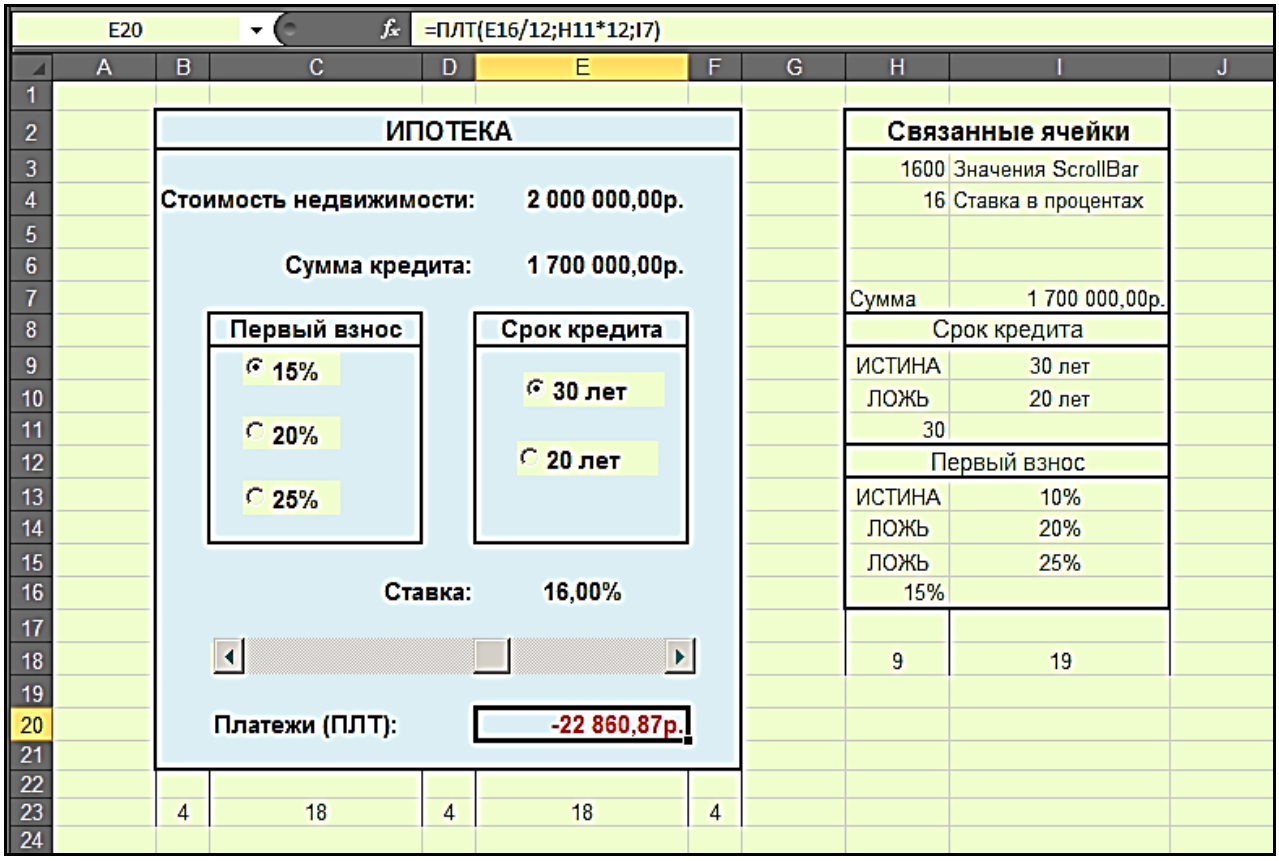


Рис. 3.11

3. Открыв в рабочей книге чистый лист, задайте команду ***Разработчик ⇒ Элементы управления ⇒ Вставить.***
4. Щелкнув по **OptionButton,** начертите мышью прямоугольник внутри рамки «Первый взнос».
5. Скопировав вставленный элемент управления, вставьте две его копии в рамку «Первый взнос», а две другие копии – в рамку «Срок кредита».
6. Если не удалось вручную выровнять элементы управления в группе, выделив их, задайте команду: ***Формат ⇒ Упорядочить ⇒ Выровнять*** и из открывшегося списка выберите нужный вариант выравнивания.
7. Выделив верхний переключатель в левой группе, задайте команду ***Разработчик ⇒ Элементы управления ⇒ Свойства*** и в окно «Properties» введите значения его параметров, представленных в таблице 3.1.

Таблица 3.1

	OptionButton1	**OptionButton2**	**OptionButton3**

Caption	15%	20%	25%
GroupName	Взнос	Взнос	Взнос
Height	15	15	15
LinkedCell	H13	H14	H15
Value	True	False	False
Width	45	45	45

8. Поочередно выделив **OptionButton2** и **OptionButton3,** выполните пункт 7 этого упражнения.
9. Убедитесь в том, что на рабочем листе отобразились значения ИСТИНА, ЛОЖЬ, ЛОЖЬ в ячейках диапазона H13:H15.
10. В ячейку H16 введите формулу: **=ЕСЛИ(H13;0,15;ЕСЛИ(H14;0,2;0,25)).**
11. Отменив режим конструктора, щелкните поочередно по переключателям первого взноса, контролируя при этом значение в ячейке H16.
12. В ячейку I7 введите формулу: **=E4-(E4*H16).**
13. В ячейку E4 введите любое положительное число и задайте для нее денежный формат.
14. Выделив в режиме конструктора верхний переключатель **OptionButton4** в левой группе, задайте команду ***Разработчик ⇒ Элементы управления ⇒ Свойства*** и в окно «Properties» введите значения его параметров, представленные в таблице 3.2.

Таблица 3.2

	OptionButton4	**OptionButton5**

Caption	30 лет	20 лет
GroupName	Срок	Срок
Height	15	15
LinkedCell	H9	H10
Value	True	False
Width	45	45

15. Выделив **OptionButton5,** выполните пункт 14 этого упражнения и убедитесь в том, что на рабочем листе отобразились значения ИСТИНА, ЛОЖЬ в ячейках H9 и H10.
16. В ячейку H11 введите формулу: **=ЕСЛИ(H9;30;20).**
17. Отменив режим конструктора, щелкните поочередно по переключателям срока кредита, контролируя при этом значение в ячейке H11.
18. Выделив в режиме конструктора полосу прокрутки, задайте команду ***Разработчик ⇒ Элементы управления ⇒ Свойства*** и в окно «Properties» введите значения её параметров **Max**, **Min, LargeChange** и **LinkedCell.** Автор использовал значения из таблицы 3.3.

Таблица 3.3

	ScrollBar1
Max	2000
Min	1000
LargeChange	100
LinkedCell	H3

19. В ячейку H4 введите формулу: **=H3/100.**

20. В ячейку E16 введите формулу: **=H4/100** и задайте для нее процентный формат.
21. Отменив режим конструктора, измените положение движка на полосе прокрутки, контролируя при этом значение ставки в ячейке E16.
22. Выделив ячейку E20, с помощью **Мастера функций** введите в нее аргументы функции ПЛТ().
23. Проверьте работоспособность созданной нестандартной формы. Задавая различные величины первого взноса и сроков кредита, плавно изменяйте значение процентной ставки.
24. С помощью стандартной функции ПЛТ () рассчитайте величину платежей при трех конкретных значениях суммы кредита, ставки и срока выплат (см. рис. 3.12).

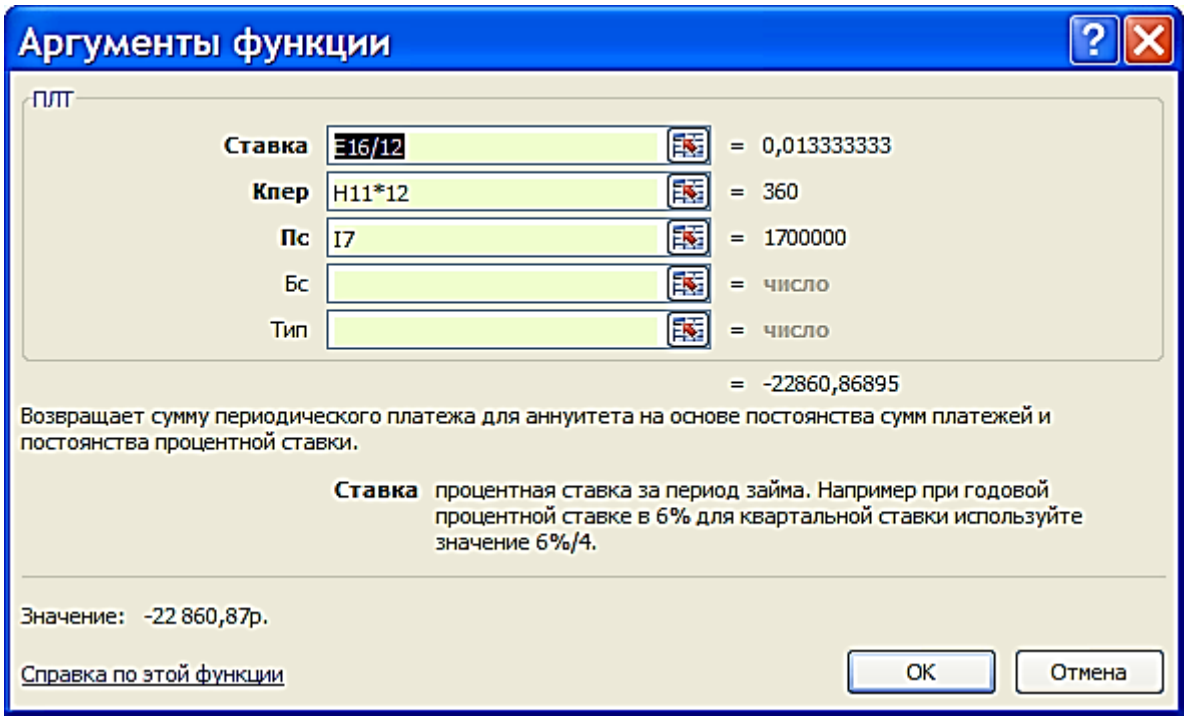


Рис. 3.12

3.3 Конструирование пользовательских диалоговых окон

Нестандартные диалоговые окна UserForm создают и применяют в тех случаях, когда требуется ввести ограниченный объем данных, необходимых для выполнения специфической задачи.

Поскольку пользовательские диалоговые окна модальные, пользователь может действовать только в пределах отображенного на экране диалогового окна.

Если необходимо что-либо выделять на рабочем листе при отображенном нестандартном диалоговом окне, поместите на него элемент управления RefEdit[54]**.**

3.3.1 Свойства объекта UserForm

1. **Name —** имя объекта UserForm, под которым он значится в программном коде процедур.
2. **Caption —** текст, отображаемый в строке заголовка объекта UserForm.
3. **ForeColor —** цвет текста в строке заголовка.
4. **BackColor —** цвет фона объекта UserForm.
5. **BorderStyle —** тип границы.
6. **Picture —** ссылка на растровое изображение, которое будет отображаться как фон объекта UserForm.
7. **PictureSizeMode** задает режим, позволяющий изменять масштаб изображения.
8. **Left** и **Top —** координаты верхнего левого угла объекта в пунктах.
9. **Height** и **Width —** высота и ширина объекта UserForm в пунктах.
10. **StartUpPosition —** местоположение формы при первом отображении её на экране. Допустимые значения:

- **Manual** — начальное значение не устанавливается;
- **CenterOwner** — выравнивание по центру объекта, к которому принадлежит форма;
- **CenterScreen** — выравнивание по центру экрана;
- **WindowsDefault —** положение верхнего левого угла экрана.

[54] RefEdit используется тогда, когда пользователь должен выделить диапазон ячеек на листе.

3.3.2 Методы объекта UserForm

1. **Show —** загружает экранную форму в оперативную память компьютера и отображает ее на экране.
2. **Hide —** закрывает форму, не удаляя ее из оперативной памяти компьютера.
3. **Move —** изменяет местоположение экранной формы.
4. **PrintForm —** печатает изображение формы.

3.3.3 Соглашения об именах

Выполняя предыдущее упражнение, вы наверняка обратили внимание на то, что при размещении элементов ActiveX на рабочем листе Excel по умолчанию устанавливается свойство Name, в котором присутствует порядковый номер создания элемента этого типа. Например, имя OptionButton1 было присвоено первому созданному переключателю, второму – OptionButton2 и т.д. Поскольку при разработке нестандартных форм установленное по умолчанию имя элемента ActiveX не вызывает никаких проблем, мы их оставили без изменения.

В процессе конструирования нестандартных диалоговых окон вам придется писать программный код как минимум для одной процедуры, где придется неоднократно ссылаться на свойство Name различных управляющих элементов. Замените их на более короткие. Для переименования объектов и переменных в среде Windows существует соглашение об именах, называемое «венгерской нотацией». Суть его в том, что имя объекта начинается с короткого префикса, одинакового для однотипных объектов (см. таб. 3.4).

Таблица 3.4

ЭУ ActiveX	**Префикс**
Label	**Lbl**
TextBox	**Txt**
ComboBox	**Cbo**
ListBox	**Lst**
CheckBox	**Chk**
OptionButton	**Opt**
ToggleButton	**Tgl**
CommandButton	**Cmd**

ScrollBar	**Scr**
SpinButton	**Sp n**

За префиксом следует либо собственно имя, отображающее суть элемента управления, либо его порядковый номер .

УПРАЖНЕНИЕ 26

Создать нестандартное диалоговое окно для вычисления размера комиссионных, которые придется заплатить за полученную на предприятии ссуду при разных способах её погашения (помесячно или ежегодно).

1. Открыв в рабочей книге новый лист, создайте форму (см. рис.) для отображения в ней введенных в диалоговое окно исходных данных и полученных результатов вычислений

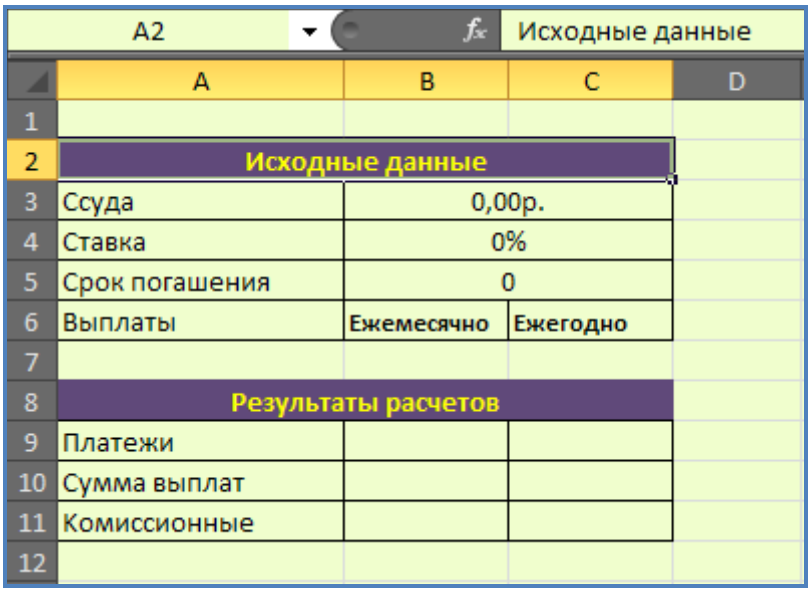


Рис. 3.13

2. Задайте команду ***Разработчик ⇒ Код ⇒ Visual Basic.***
3. В редакторе Visual Basic, задав команду ***Insert ⇒ UserForm,*** откройте окно «Properties» и задайте значения параметров Height=420 и Width=240.
4. Если одновременно с объектом **UserForm1** в окне редактора не отобразилась панель «Toolbox» с элементами управления ActiveX, задайте команду ***View ⇒ Toolbox.***
5. Поместите на объект **UserForm1:**
   - шесть полей для надписей **Label1 – Label6;**

- шесть текстовых полей **TextBox1 – TextBox6;**
- два переключателя **OptionButton1** и **OptionButton2;**
- две кнопки **CommandButton1 и CommandButton2;**
- рамку **Frame**

Примечания

1) Один из возможных вариантов схемы размещения представлен на рисунке 3.14.
2) Для ускорения работы можно, поместив первое поле, скопировать его и далее вставлять и передвигать копии.
3) Выделив группу полей и задав следующие команды:
   - ***Format ⇒ Align,*** вы сможете выровнять элементы этой группы;
   - ***Format ⇒ Make Same Size*** удобна для подгонки размеров выделенных элементов;
   - ***Format ⇒ Horizontal/Vertical Spacing,*** можно выровнять расстояния между полями.

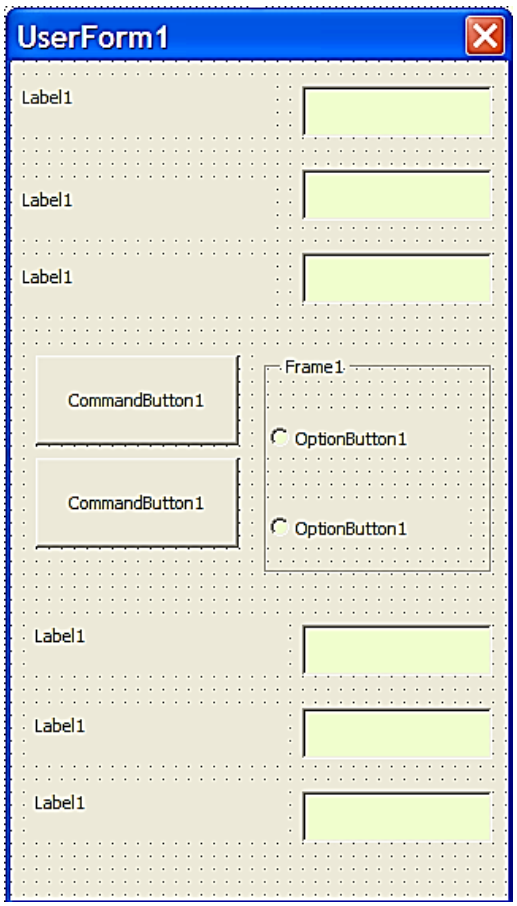


Рис 3.14

Таблица 3.5

6. Поочередно выделив поля для надписей, переименуйте их и замените значение параметра **Caption** на те, что представлены в таблице 3.5.
7. Переименовывая остальные элементы, замените значение параметра Caption у

Lbl1	**Размер ссуды (р.)**
Lbl2	**Ставка (%)**
Lbl3	**Срок (лет)**
Lbl4	**Платежи**

двух кнопок, а у переключателей задайте значения трех параметров: **Caption**, **GroupName** и **Value**.

8. Исходные данные для вычислений, вводимые пользователем в текстовые окна

Lbl5	**Сумма выплат**
Lbl6	**Комиссионные**
Opt1	**Ежемесячно**
Opt2	**Ежегодно**
Cmd1	**Вычислить**
Cmd2	**Очистить**

Txt1, Txt2 и **Txt3,** должны отображаться в ячейках B3, B4 и B5 формы на рабочем листе (см. рис. 3.13).

9. Результаты расчетов выводятся в текстовые окна **Txt4, Txt5** и **Txt6** диалогового окна и в отформатированном виде в ячейки формы (диапазон B9:C11).
10. Из контекстного меню кнопки **Cmd1** задайте команду **View Code.** Откроется окно **Code** с заголовком процедуры (см. рис. 3.15).

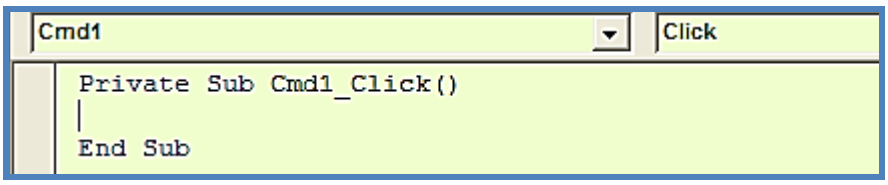


Рис.3.15

11. Напечатайте нижеприведенный программный код процедуры **Cmd1** (реакция на событие **Click** – щелчок курсором мыши по кнопке «Вычислить»).

В процедуре **Sub Cmd1_Click(),** которая выполняет расчеты по щелчку по кнопке «Вычислить» использованы встроенные функции VBA:

- **Pmt –** аналог финансовой функции **ПЛТ();**
- **Val** преобразует вводимые с экрана данные в числовой формат;
- **Str** преобразует числовые результаты вычислений в строковый формат для вывода их на экран монитора.

```
Private Sub Cmd1_Click()
```

```
Dim ПЛТ, ПС, Вып, Ком As Single
Dim Ставка, Срок As Integer
ПС = Val(Txt1.Text)                ' размер ссуды
Ставка = Val(Txt2.Text) / 100  'ставка в процентах
Срок = Val(Txt3.Text)              'срок погашения ссуды
Cells(3, 2) = ПС
Cells(4, 2) = Ставка
Cells(5, 2) = Срок
        If Opt1.Value Then
                ПЛТ = Pmt(Ставка / 12, Срок * 12, -ПС)  ' ежемесячные платежи
                Вып = ПЛТ * Срок * 12
                Cells(9, 2) = ПЛТ
                Cells(10, 2) = Вып
                Cells(11, 2) = Вып - ПС
        Else
                ПЛТ = Pmt(Ставка, Срок, - ПС)    'ежегодные платежи
                Вып = ПЛТ * Срок
                Cells(9, 3) = ПЛТ
                Cells(10, 3) = Вып
                Cells(11, 3) = Вып - ПС
        End If
      Ком = Вып - ПС             'комиссионные
Txt4.Text = Str(ПЛТ)
Txt5.Text = Str(Вып)
Txt6.Text = Str(Ком)
End Sub
```

12. Из контекстного меню кнопки **Cmd2** задайте команду **View Code.**
13. В окне **Code** напечатайте нижеприведенный программный код процедуры **Cmd2** (реакция на событие **Click** – щелчок курсором мыши по кнопке «Очистить»).

```
Private Sub Cmd2_Click()
```

```
    Txt1.Text = " "
    Txt2.Text = " "
    Txt3.Text = " "
End Sub
```

14. Задав команду ***Insert ⇒ UserForm,*** напечатайте в окне **Code** представленный ниже программный код процедуры загрузки и отображения созданного диалогового окна.

```
Sub Загрузить_Комисс()
    UserForm1.Show
End Sub
```

15. Для вызова диалогового окна «Выплаты по ссуде» поместите на рабочий лист элемент управления формы **Кнопка** и в окне «Назначить макрос объекту» назначьте ей процедуру Sub Загрузить_Комисс().
16. Введите ваши исходные данные и, выполнив вычисления, проверьте их точность с помощью встроенной в Excel финансовой функции ПЛТ().

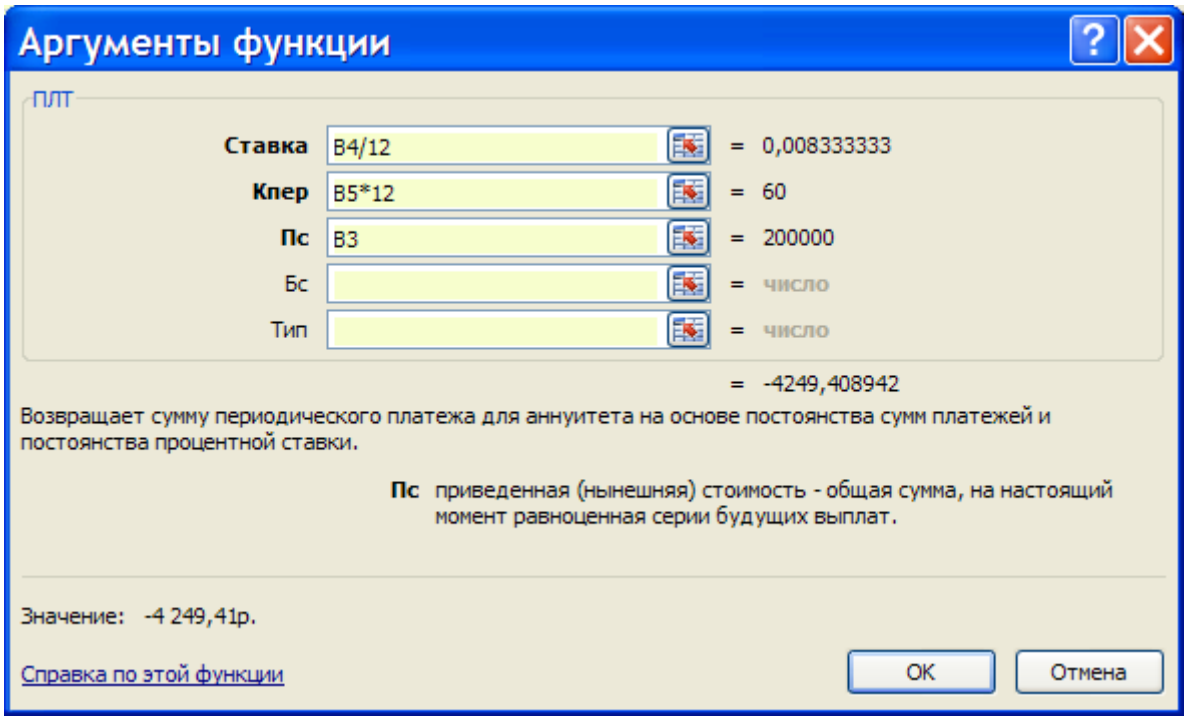


Рис.3.16

17. На рисунке 3.17 представлено сконструированное диалоговое окно и пользовательская форма с отчетом о проделанной работе.

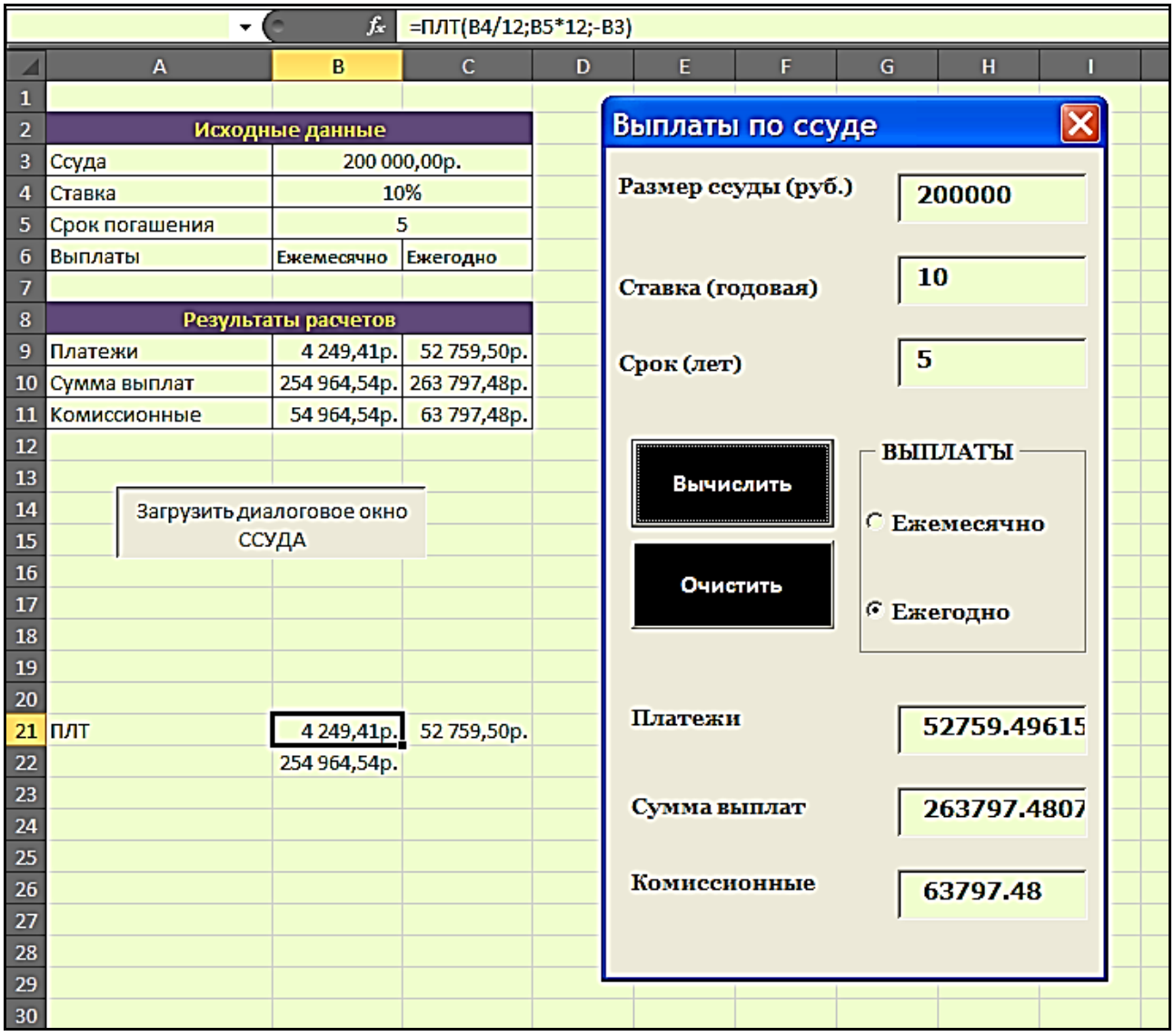

=ПЛТ(B4/12;B5*12;-B3)
Исходные данные
Ссуда
200 000,00р.
Ставка
10%
Срок погашения
5
Выплаты
Ежемесячно
Ежегодно
Результаты расчетов
Платежи
4 249,41р.
52 759,50р.
Сумма выплат
254 964,54р.
263 797,48р.
Комиссионные
54 964,54р.
63 797,48р.
Загрузить диалоговое окно ССУДА
ПЛТ
4 249,41р.
52 759,50р.
254 964,54р.
Выплаты по ссуде
Размер ссуды (руб.)
200000
Ставка (годовая)
10
Срок (лет)
5
Вычислить
Очистить
ВЫПЛАТЫ
Ежемесячно
Ежегодно
Платежи
52759.49615
Сумма выплат
263797.4807
Комиссионные
63797.48

Рис. 3.17

Список литературы

Гарнаев А.Ю. Microsoft Office Excel 2010. Разработка приложений. СПб., 2011.

Кашаев С. М. Офисные решения с использованием Microsoft Excel 2007 и VBA. СПб., 2009.

Михеев Р.Н. VBA и программирование в MS Office для пользователей. СПб., 2006.

Слепцова Л.Д. Программирование на VBA в Microsoft Office 2010. М., 2010.

Уокенбах Дж. Microsoft Excel 2010. Библия пользователя. М., 2012.

Приложение 1

Встроенные функции языка VBA, использованные в упражнениях

Abs() Возвращает абсолютное значение переданного ей числа.

Array() Автоматически создаёт массив нужного размера и типа.

Date() Возвращает текущую системную дату.

Fix () Отбрасывает дробную часть числа.

FV () Возвращает будущую стоимость вклада.

Impt() Возвращает величину выплаты за указанный период.

InputBox() Обеспечивает прием информации от пользователя.

Int () Округляет дробное число до меньшего ближайшего числа.

MsgBox () Обеспечивает взаимодействие[55] с пользователем.

Now() Возвращает текущее системное время и дату

Nper() Возвращает количество периодов выплаты по вкладу.

PV () Возвращает приведенную (текущую) стоимость вклада

Pmt() Возвращает величину выплат по кредиту.

Rnd() Вместе с инструкцией Randomize формирует случайные числа.

Time() Возвращает текущее системное время.

[55] Информирует пользователя

Приложение 2

Список сокращений

MS – Microsoft

VBA – Visual Basic for Application (язык программирования для приложений пакета Microsoft Office System)

Sub – Subroutine (подпрограмма)

Dim – Dimension (размерность)

NB! – Обратить внимание!

кн. ОК – кнопка okay

ПБД – панель быстрого доступа

ЭУ – элемент управления формы (старый) и элемент ActiveX (новый)