Оглавление

Введение

Microsoft Excel — электронные таблицы, которые предназначены для вычислений, анализа и построения разнообразных отчетов.

Сегодня этот программный продукт — абсолютный лидер среди аналогичных программ электронных таблиц по своей простоте и вместе с тем колоссальным возможностям.

Электронные таблицы Microsoft Excel разработаны компанией Microsoft и обновляются примерно раз в два года. В каждой новой версии Microsoft Excel расширяются возможности этой программы. Все, чему пользователь учится в одной программе, он найдет и в следующей версии, плюс что-то новое.

Даже самая совершенная программа не может охватить всего многообразия реальных ситуаций и задач. Чтобы вы могли самостоятельно решать такие задачи, в программе Microsoft Excel существуют макросы.

Как правило, в учебниках по программе Microsoft Excel об этом понятии говорится вскользь, и это неудивительно: к изучению макросов пользователи подходят уже подготовленными специалистами по Microsoft Excel. Чтобы создавать макросы, нужно видеть проблему, которая решается нестандартным способом.

Умение самостоятельно решать такие проблемы позволит вам не только почувствовать себя крутым специалистом, которых компании зазывают к себе на работу, но и самой организации сэкономить много финансовых средств, которые были бы потрачены на поиск программистов.

Программирование в среде Excel осуществляется на языке Visual Basic for Applications.

Слово *Basic* — это аббревиатура (Beginners Allpurpose Symbolic Instruction Code).

1

Введение в офисное программирование

Цель разработки

Офисное программирование предназначено для разработки приложений, используемых для автоматизации офисной деятельности. У приложений различное назначение: Microsoft Office, OpenOffice, LibreOffice, CorelDRAW Graphics Suite.

Цель — совершенствовать работу с документами под требования конкретного пользователя. Бывают макросы постоянного и разового применения. Вопрос в том, как выгоднее и быстрее обработать информацию: с помощью макроса или вручную.

Область применения

Макрос входит в состав документов: или одного, или встраивается в шаблон документа. В последнем случае макрос встраивается в каждый создаваемый документ. Макрос может изменять настройки документа или использоваться для обработки других документов. Может, например, изменять параметры шрифта, ориентацию страниц, создавать разделы, изменять ориентацию страниц в каждом из разделов, исправлять одни буквы на другие — і на и, ѣ на е, ѳ на ф в дореволюционных книгах и т. д. Как самостоятельная программа — отдельно от документа, рабочей книги, презентации — макрос не используется.

Язык программирования

В офисном программировании, как правило, используется язык Visual Basic for Applications (VBA). В некоторых приложениях (OpenOffice, LibreOffice, CorelDRAW Graphics Suite) используется версия языка StarBasic — в связи с другим набором объектов.

Язык Visual Basic for Applications — специализированная версия языка Visual Basic.

Среда разработки

Среда разработки — приложения Microsoft Office. Каждое имеет свой набор объектов: Документ ((Documents) в Word, Рабочий лист (WorkSheets) и Рабочая книга (Workbook) в Excel) и т. д.

Макросы создаются двумя способами: включением макрорекордера и ручным программированием. Макрорекордер записывает все действия, а программные строки записываются автоматически. Запись макроса вручную выполняется с помощью специального редактора, который загружается из риббона Разработчик.

Поддержка ООП

Язык VBA поддерживает объектно-ориентированное программирование. Документы, листы, книги, презентации, методы и события — это объекты. Например, многие цвета объектов имеют свои имена и/или константы, по ним эти цвета можно вызвать. Каждый элемент на форме и сама форма тоже объекты, у них есть свойства, которые можно изменить.

Объектно-ориентированное программирование — основа визуального программирования, то есть свойства объектов можно изменять не только в тексте программы, но и с помощью специальной панели. Но эта возможность может повлечь ошибки у неопытных пользователей. Поэтому многие преподаватели не разрешают студентам в начале обучения изменять свойства объектов на панели свойств, а использовать их только при обработке события инициализации формы.

Преимущества офисного программирования

Возможность использовать макрорекордер позволяет пользователю изменять документ, не привлекая программистов.

Пользователь может продолжать работать в привычном приложении, получая новые возможности.

Многие приложения (OpenOffice, LibreOffice, Mozilla Firefox) могут изменять возможности с помощью расширений и дополнений, их разрабатывают сторонние программисты. Для этих приложений организованы специальные интернет-магазины и библиотеки, можно скачать и установить такие дополнения.

2

Макросы. Использование макрорекордера

Назначение макросов

Макрос — это последовательность команд, которые выполняются в Microsoft Office, OpenOffice, LibreOffice, Corel Graphic Suite. В этих приложениях макросы можно создавать на языках Visual Basic for Applications, его версии StarBasic, многих других. Макросы — самостоятельные программные продукты, но способны выполняться только внутри одного из приложений. Они могут быть потенциально опасными, поэтому о существовании макросов обычно предупреждают в устаревших форматах: DOC, XLS и др. В новейших форматах (DOCX, XLSX) макросы сохранить невозможно: для этого предусмотрены специальные форматы (DOCM, XSLM). Буква *M* в формате говорит о специализации формата для хранения макросов. Такое разделение форматов по назначению в некоторой степени защищает пользователей от макровирусов при скачивании файлов из интернета.

Во многих приложениях для создания макросов используется язык Visual Basic for Applications (VBA) или его версии (StarBasic), но могут использоваться и другие языки: Python, BeanShell, JavaScript, Lisp (если это предусмотрено). Во многих приложениях, хотя об этом и не сообщается явно, используется язык запросов SQL.

Макросы нужны для автоматизации часто повторяющихся операций. На практике наибольшее применение макросов встречается в электронных таблицах. Поэтому в примерах будут использованы именно электронные таблицы (на примере Microsoft Excel). Рассмотрим несколько простейших примеров.

1. Отделу кадров часто приходится вводить информацию с персональными данными сотрудников: фамилию, имя, отчество. Для выбора ширины столбца можно разработать макрос.

2. В электронных таблицах нужно форматировать заголовки, шапку таблицы: размер шрифта, начертание и т. д.

3. Оформление бланков и отчетов. В столбцах можно создать формулы. Пользователю остается заполнить бланк или отчет числами, по которым немедленно будут сделаны расчеты.

4. Электронная библиотека Максима Мошкова предлагает множество дореволюционных книг. С помощью макрорекордера можно отредактировать текст в современном написании (без ятей и ижиц) и сделать книги удобными для чтения и конвертирования в электронные форматы (FB2, EPUB, PDF, DjVu).

Для создания макросов есть два способа: автоматическое создание команд по принципу магнитофона (авторекордер) и ручное создание в редакторе Visual Basic for Applications. Оба метода взаимосвязаны: в любой момент один может дополнить другой. Например, можно записать некоторую последовательность команд с помощью макрорекордера, а затем отредактировать полученный текст программы в редакторе VBA.

Запись макросов. Использование макрорекордера

Создадим пример. Введем пять записей с ФИО в трех столбцах (строку 7 не вводите), а затем включим макрорекордер, и для каждого столбца будет определена оптимальная ширина (рис. 1).

	A	B	C	D
1	Фамилия	Имя	Отчество	
2	Амирова	Дарья	Руслановна	
3	Гаджимур	Амина	Байрамбековна	
4	Копытина	Екатерин	Дмитриевна	
5	Айнуллин	Рушан	Раисович	
6	Коловато	Олеся	Александровна	
7	Байрамов	Тельман	Бюньямутдинович	

	A	B	C
1	Фамилия	Имя	Отчество
2	Амирова	Дарья	Руслановна
3	Гаджимурадова	Амина	Байрамбековна
4	Копытина	Екатерина	Дмитриевна
5	Айнуллин	Рушан	Раисович
6	Коловатова	Олеся	Александровна
7	Байрамов	Тельман	Бюньямутдинович

Рисунок 1. Столбцы до и после применения макроса

Алгоритм создания макроса: выполните команду **Разработчик ⇒ Запись макроса:** укажите имя нового макроса (рис. 2), создайте описание назначения макроса, нажмите на кнопку **ОК.**

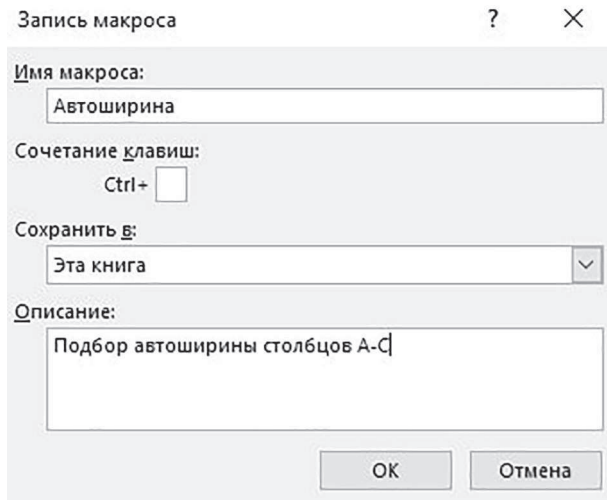

Рисунок 2. Диалоговое окно Запись макроса

Установите автоширину каждого из трех столбцов двойным щелчком между заголовками столбцов, выполните команду **Разработчик ⇒ Остановить запись.** Макрос готов, и его программный код можно просмотреть командой **Разработчик ⇒ Макрос ⇒ Изменить.** Появится следующий программный код:

```
Sub Автоширина()
'
' Автоширина Макрос
' Подбор автоширины столбцов
'

'
    Columns("A:A").EntireColumn.AutoFit
    Columns("B:B").EntireColumn.AutoFit
    Columns("C:C").EntireColumn.AutoFit
End Sub
```

Команда (AutoFit) устанавливает автоширину (Columns) для указанного столбца. Обратите внимание на имена ячеек (A:A, B:B, C:C) — автоширина подбирается по всему столбцу.

Обратите внимание на двойные компьютерные кавычки ("") — в тексте программ допускаются только такие кавычки. Другие типы кавычек (« », “ ”) не допускаются. Распространенная ошибка среди начинающих программистов — вместо одной двойной кавычки часто вводят две одиночные кавычки, найти такую ошибку очень сложно. Такие ошибки при использовании макрорекордера допущены не будут, так как он знает, какие кавычки допустимы, а вот при ручном написании текста программ ошибки с кавычками могут быть. Часто недопустимые кавычки оказываются в тексте программ при:

1) копировании текста программы из интернета;
2) при сканировании и оптическом распознавании текста, а потом при копировании этого текста и его вставке в свою программу.

Воспроизведение макроса

Добавьте новую строку с длинными ФИО. В примере (рис. 1) фамилия и отчество нового работника (строка 7) не умещаются по ширине столбцов. Выделите ячейку A1.

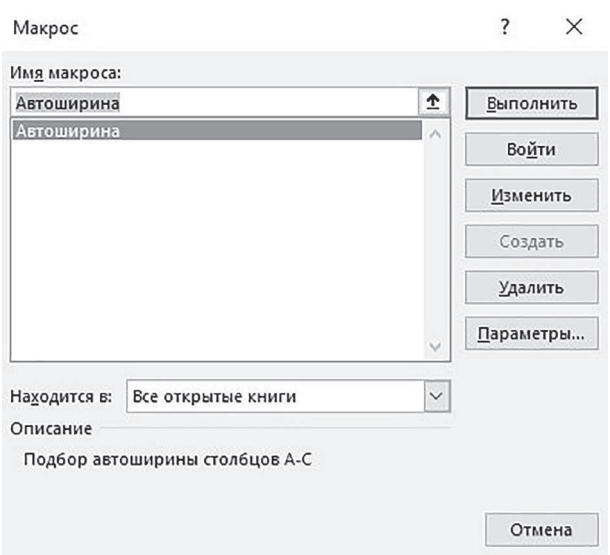

Рисунок 3. Выбор макроса

Для воспроизведения макроса выполните команду **Разработчик ⇒ Макрос ⇒ Выполнить.** В открывшемся диалоговом окне (рис. 3) выберите имя макроса, который необходимо воспроизвести, и щелкните по нему левой клавишей. После выбора макроса в списке он появляется в поле **Имя макроса**. Чтобы воспроизвести выбранный макрос, необходимо нажать на кнопку **Выполнить**.

Макрос: создание заголовка таблицы

Рассмотрим другой пример, в котором в ячейке A5 вводится заголовок таблицы. Создадим макрос, который должен выполнять следующие действия:

- вводить текст *Отчет за II квартал*;
- устанавливать размер шрифта в 20 пунктов;
- форматировать текст полужирным и курсивным начертанием;
- устанавливать автоширину столбца (рис. 4).

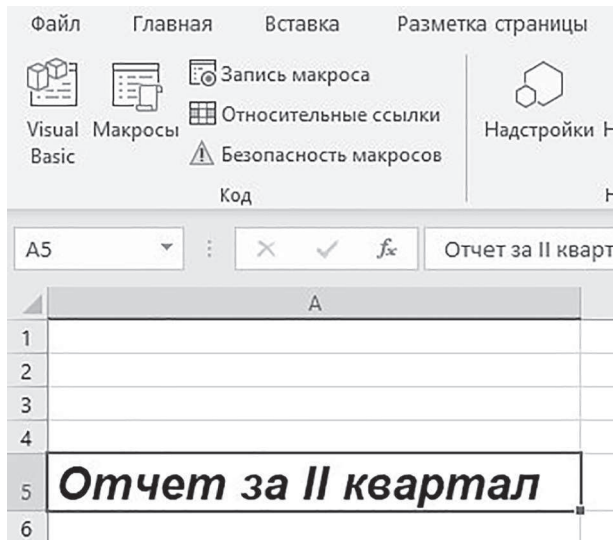

Рисунок 4. Оформленный заголовок

Алгоритм создания макроса в этом примере ничем не отличается от предыдущего: выполняем команду **Разработчик ⇒ Макрос ⇒ Начать запись**, задаем имя новому макросу,

вводим неформатированный текст, форматируем ячейку полужирным и курсивным начертанием, устанавливаем автоширину столбца, выполняем команду **Разработчик ⇒ Макрос ⇒ Остановить запись.** Макрос готов. Его программный код можно просмотреть командой **Разработчик ⇒ Макрос ⇒ Макросы** и в открывшемся диалоговом окне **Макрос** нажать на кнопку **Изменить.** Появится следующий программный код:

```
Sub Макрос2()
'
' Макрос2 Макрос
' Макрос записан 25.11.2022 (Шитов)
'

'
    Columns("A:A").EntireColumn.AutoFit
    Range("A5").Select
    ActiveCell.FormulaR1C1 = "Отчет за II квартал"
    With ActiveCell.Characters(Start:=1,
Length:=19).Font
        .Name = "Arial Cyr"
        .FontStyle = "полужирный курсив"
        .Size = 20
        .Strikethrough = False
        .Superscript = False
        .Subscript = False
        .OutlineFont = False
        .Shadow = False
        .Underline = xlUnderlineStyleNone
        .ColorIndex = xlAutomatic
    End With
    Rows("5:5").Select
End Sub
```

Из всего программного кода отметим только значения булева типа, которые могут принимать следующие значения: или *True* (*истина*), или *False* (*ложь*). Опция True означает, что данный параметр включен, а опция False — параметр выключен.

Как видно из текста программных кодов, есть повторяющиеся команды:

```
Columns("A:A").EntireColumn.AutoFit
```

Даже если мы ничего не понимаем в программировании, все равно понятно, что для столбца (Columns) устанавливается автоширина (AutoFit), пример работы с которой рассмотрен чуть выше.

В тексте этой программы задан текст, который мы должны вводить в ячейке. С одной стороны, удобно для ввода повторяющихся и не изменяющихся записей. Но в этом примере в следующем квартале текст надписи, наверное, будет «Отчет за III квартал», и нам придется вручную изменять его. Учтем, что человек часто забывает исправлять подобные ошибки, поэтому в нашем примере правильнее вообще не вводить текст. Для этого в программном коде удалим текст между кавычками, оставив только кавычки: **""**. Будет проводиться форматирование ячейки, а требуемый текст можно потом ввести вручную, уже с заданными параметрами форматирования.

Создайте пустую строку перед последней строкой макроса End Sub. Для этого поместите курсор перед этой строкой, нажмите на клавишу Enter на клавиатуре. Вставьте в пустой строке команду:

```
ActiveCell.FormulaR1C1 = ""
```

Снова выполните макрос: теперь ширина столбца A изменяется, а потом текст удаляется, но ширина столбца не изменяется.

Воспроизведение макроса

Выполните команду **Разработчик ⇒ Макрос ⇒ Макросы**. В открывшемся диалоговом окне выберите имя макроса и нажмите на кнопку **Выполнить**.

Внешне работа макроса не очень видна, так как он только подготовил ячейки для ввода информации: отформатировал шрифт, но текста в ячейке нет. Если теперь начнем вводить текст, он будет вводиться с полужирным и курсивным начертанием, размером 20 пунктов и другими выбранными параметрами.

Макрос: создание балансового отчета

Рассмотрим более сложный пример — макрос по составлению болванки для создания балансового отчета. В интернете нашли шаблон бюджета и сохранили под именем *Балансовый отчет.xlt* (рис. 5).

Создайте новую книгу, задайте ей имя (у нас это Книга2). Имена рабочих книг специально оговариваются, так как в тексте программы будут фигурировать два имени: Книга2 и *Балансовый отчет.xlt*. Была открыта книга *Балансовый отчет.xlt*. Выполните команду **Вид ⇒ Расположить** и в открывшемся диалоговом окне **Расположение окон** установите переключатель **Расположить ⇒ Сверху вниз.**

1. Выполните команду **Разработчик ⇒ Макрос ⇒ Начать запись**. Задайте имя макросу. Можете оставить его по умолчанию — Макрос1.

2. Введите в ячейку F1 текст: Балансовый отчет. Выделите эту ячейку, измените размер шрифта на 14 пунктов. Выделите текст и задайте ему полужирное и курсивное начертание. Выделите ячейки с A1 по F1. На риббоне **Главная** нажмите на кнопку **Объединить и поместить в центре** ().

3. В ячейку A3 введите текст: Описание. Выделите ячейки с A3 по E5. Выполните команду **Главная ⇒ Формат ⇒ Формат ячеек**. В открывшемся диалоговом окне **Формат ячеек** перейдите на вкладку **Выравнивание**. Установите флажок в индикаторе **Объединение ячеек**. В раскрывающемся списке **Выравнивание по вертикали** выберите значение **По верхнему краю**. Нажмите **ОК** для закрытия окна **Формат ячеек.**

4. В ячейке G3 введите текст: Начальный баланс. Выделите ячейки G3, H3, I3. Выполните команду **Главная ⇒ Формат ⇒ Формат ячеек**. В открывшемся диалоговом окне **Формат ячеек** перейдите на вкладку **Выравнивание**. Установите флажок в индикаторе **Объединение**

ячеек. Нажмите **ОК** для закрытия окна **Формат ячеек**.

5. Выделите ячейку A7 и введите текст: Дата.

6. В ячейку B7 введите текст: Описание статьи. Выделите ячейки с B7 по G7. На риббоне **Главная** нажмите на кнопку **Объединить и поместить в центре** ().

7. В ячейках H7, I7 и J7 введите соответственно текст: Получено, Платеж и Баланс.

8. В ячейке G18 введите текст: Получено. Выполните команду **Главная ⇒ Формат ⇒ Формат ячеек.** В открывшемся диалоговом окне **Формат ячеек** перейдите на вкладку **Выравнивание**. Установите флажок в индикаторе **Объединение ячеек**. В раскрывающемся списке **Выравнивание по горизонтали** выберите значение **По правому краю (отступ)**.

9. В ячейке G19 введите текст: Платежи. Выполните команду **Главная ⇒ Формат ⇒ Формат ячеек.** В открывшемся диалоговом окне **Формат ячеек** перейдите на вкладку **Выравнивание.** Установите флажок в индикаторе **Объединение ячеек**. В раскрывающемся списке **Выравнивание по горизонтали** выберите значение **По правому краю (отступ).**

10. В ячейке G20 введите текст: Текущий баланс. Выполните команду **Главная ⇒ Формат ⇒ Формат ячеек.** В открывшемся диалоговом окне **Формат ячеек** перейдите на вкладку **Выравнивание.** Установите флажок в индикаторе **Объединение ячеек**. В раскрывающемся списке **Выравнивание по горизонтали** выберите значение **По правому краю (отступ)**.

11. Выделите ячейки с H18 по H20. На риббоне **Главная** нажмите на треугольную кнопку, раскрывающую список кнопки **Цвет заливки** (). Выберите светло-бирюзовый.

12 Выделите ячейки с H8 по H17. На риббоне **Главная** нажмите на треугольную кнопку, раскрывающую список кнопки **Цвет заливки**. Выберите серый 25%.

13 Выделите ячейки с I8 по I17. На риббоне **Главная** нажмите на треугольную кнопку, раскрывающую список кнопки **Цвет заливки**. Выберите светло-желтый.

14 Выделите ячейки с J8 по J17. На риббоне **Главная** нажмите на треугольную кнопку, раскрывающую список кнопки **Цвет заливки**. Выберите бирюзовый.

На этом можем отключить создание макроса.

После создания макроса нужно выполнить команду **Разработчик ⇒ Макрос ⇒ Остановить запись.** Воспроизводить этот макрос не нужно, так как он все равно работать пока не будет.

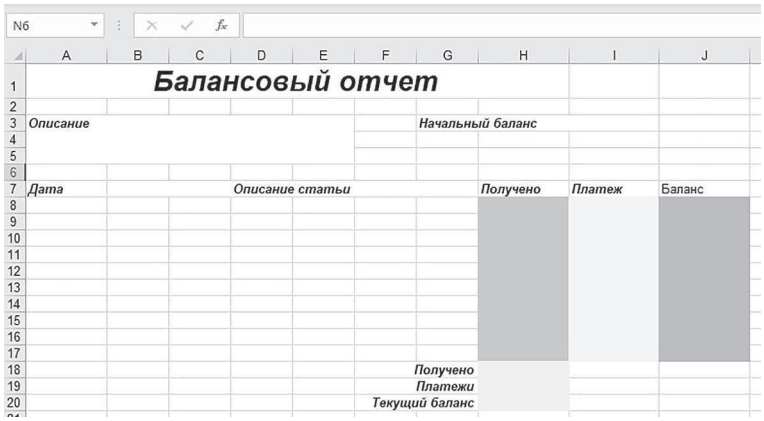

Рисунок 5. Балансовый отчет, созданный с помощью макроса

При создании макроса мы умышленно забыли некоторые вопросы. Например, форматирование ячеек в диапазоне с A8 по J17 и с H18 по H20; суммирование в ячейках с H18 по H20; выравнивание по центру надписи *Дата* и т. д. Форматирование должно было производиться разными числовыми форматами. В этом макросе мы этого не сделали, чтобы показать в дальнейшем, как производить изменения макроса. Рано или поздно

проблема изменения макросов все равно возникнет, поэтому легче изменить существующий макрос, чем создавать новый. Тем более что при создании нового макроса можно внести ошибки.

Макрос готов. Его программный код можно просмотреть командой **Разработчик ⇒ Макрос ⇒ Макросы** и в открывшемся диалоговом окне **Макрос** нажать на кнопку **Изменить.** Появится программный код.

Редактирование макроса

Первый макрос, который мы создали (изменение ширины столбцов), не требует никаких изменений: он очень простой. Второй (форматирование заголовка) выявил недостаток: он не объединяет и не центрирует заголовок. Идти путем, которым мы шли в первом макросе, — устанавливать автоширину — неправильно: будет расширен весь столбец, а мы этого совсем не хотим.

Чтобы заставить компьютер поработать вместо нас и показать, что же нам нужно делать, создадим небольшой макрос. Запустите запись макроса (**Разработчик ⇒ Макрос ⇒ Начать запись**), назовите его любым именем (после того, как посмотрим текст программы и скопируем ее в наш макрос (2), все равно удалим).

Выделите ячейки с A5 по E5. Нажмите на кнопку **Объединить и поместить в центре**, которая находится на инструментальной панели **Форматирование.** Остановите макрос. Выполните команду **Разработчик ⇒ Макрос ⇒ Макросы.** В открывшемся диалоговом окне выберите имя макроса, который мы создали только что, и нажмите на кнопку **Изменить.**

Выделите весь программный код этого макроса, кроме первых строк с комментарием и последней строки End Sub, и скопируйте его в буферную память (сочетание Ctrl + C). Откройте макрос по форматированию заголовка (для этого выполните команду **Разработчик ⇒ Макрос ⇒ Макросы**. В открывшемся диалоговом окне выберите имя макроса и нажмите на кнопку **Изменить**). Найдите последнюю строку макроса End Sub и установите перед ней (End Sub) указатель мыши, нажмите

Enter, чтобы создать одну пустую строку. Установите указатель мыши в начало пустой строки и нажмите Shift + Insert для вставки содержимого буферной памяти. Программный код макроса по объединению и центрированию строки будет дописан в конец макроса заголовка. Найдите строку Rows("5:5"). Select (она находилась в первоначальном коде перед командой End Sub) и закомментируйте ее. Установить комментарий в VBA означает установить символ одиночной кавычки (**'**). Все, что находится правее от одиночной кавычки, будет считаться комментарием, а не программой. Таким образом, строка Rows("5:5").Select не будет учитываться при работе программы.

Удалите макрос по объединению и центрированию ячеек. Для этого выполните команду **Разработчик ⇒ Макрос ⇒ Макросы**. В открывшемся диалоговом окне выберите имя макроса и нажмите **Удалить.** Программа спросит вас о подтверждении удаления. Нажмите **Да**.

Полный программный код макроса по форматированию, объединению и центрированию ячеек будет такой:

```
Sub Макрос()
'
' Макрос2 Макрос
' Макрос записан 22.11.2022 (Шитов)
'

'
    Columns("A:A").EntireColumn.AutoFit

Range("A5").Select
    ActiveCell.FormulaR1C1 = ""
    With ActiveCell.Characters(Start:=1,
      Length:=19).Font
        .Name = "Arial Cyr"
        .FontStyle = "полужирный курсив"
        .Size = 20
        .Strikethrough = False
        .Superscript = False
        .Subscript = False
        .OutlineFont = False
```

```
        .Shadow = False
        .Underline = xlUnderlineStyleNone
        .ColorIndex = xlAutomatic
    End With
    '  Rows("5:5").Select         так
                    оформляется комментарий (')
    Range("A5:E5").Select
    With Selection
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlBottom
        .WrapText = False
        .Orientation = 0
        .AddIndent = False
        .IndentLevel = 0
        .ShrinkToFit = False
        .ReadingOrder = xlContext
        .MergeCells = False
    End With
    Selection.Merge
End Sub
```

Рассмотрим следующий (созданный нами ранее) макрос.

При выполнении макроса Балансовый отчет необходимо открывать документ, с которого мы рисовали наш макрос. Так как для прокрутки экрана требовалось переключаться с одной книги на другую, эти действия также были записаны в макрос. Нужно найти начало этой ошибки и превратить программные строки в комментарий — и эти команды выполняться не будут. Чтобы оформить строку комментарием, нужно в самом начале строки ввести одиночную кавычку.('). Комментарием необходимо оформить следующие программные строки:

```
'  Windows("Балансовый отчет.xlt").Activate
'  ActiveWindow.SmallScroll Down:=5
'  Windows("Книга2").Activate
```

Разберем эти строки.

В первой происходит активация книги *Балансовый отчет.xlt*. Во второй — задается величина прокрутки на линейке

прокрутки. В третьей строке происходит активация документа Книга2. Книга уже активирована, значит, эта команда не нужна.

Найдем следующий фрагмент, в котором происходит переключение на книгу *Балансовый отчет.xlt*, а затем на Книга2. Сделайте эти программные строки комментарием. Необходимо первым символом в программной строке (точнее — в любом месте строки перед текстом программы в этой строке) сделать символ (').

```
'    Windows("Балансовый отчет.xlt").Activate
'    ActiveWindow.SmallScroll Down:=12
'    Windows("Книга2").Activate
```

Просмотрите свой программный код. Если встретятся обращения к книге *Балансовый отчет.xlt*, а затем к книге Книга2, оформите и эти строки как комментарий.

Может возникнуть вопрос: а почему нельзя их удалить — вместо того чтобы оформлять комментарием? В принципе, можно сделать и так, но человеку свойственно забывать собственные ошибки. Поэтому, если у вас возникнет такая же ошибка в будущем, можно открыть текст программы рассмотренного выше макроса и вспомнить, как эту ошибку исправить.

Продолжим анализ нашего программного кода. При создании макроса нам пришлось постоянно перемещаться с помощью линейки прокрутки. Это было необходимо по многим причинам, например, чтобы переместиться в книге Книга2 то вверх, то вниз. Мы создавали сложный документ, таких передвижений было не избежать. В нашем коде это помечалось командами такого типа:

```
ActiveWindow.SmallScroll Down:=7
ActiveWindow.ScrollRow = 7
```

Если запустим макрос с такими программными строками, увидим, как в процессе выполнения макроса лист книги дергается то вниз, то вверх. Поэтому такие программные строки также необходимо оформить в виде комментария:

```
'    ActiveWindow.SmallScroll Down:=7
'    ActiveWindow.ScrollRow = 7
'    ActiveWindow.SmallScroll Down:=1
'    ActiveWindow.ScrollRow = 7
'    ActiveWindow.ScrollRow = 5
'    ActiveWindow.ScrollRow = 4
'    ActiveWindow.ScrollRow = 3
'    ActiveWindow.ScrollRow = 2
'    ActiveWindow.ScrollRow = 1
'    ActiveWindow.ScrollRow = 2
'    ActiveWindow.ScrollRow = 3
'    ActiveWindow.ScrollRow = 4
'    ActiveWindow.ScrollRow = 5
'    ActiveWindow.ScrollRow = 6
'    ActiveWindow.ScrollRow = 7
'    ActiveWindow.ScrollRow = 8
```

В тексте программы есть еще одна строка такого же типа. Ее также необходимо оформить в виде комментария:

```
'    ActiveWindow.SmallScroll Down:=-3
```

После внесенных изменений в текст программы макроса нам не нужно заранее открывать файл *Балансовый отчет.xlt*, и лист книги Книга2 не дергается при работе макроса.

При создании макроса Балансового отчета мы как бы случайно (а на самом деле, конечно, не случайно) забыли сделать много другой работы:

- отформатировать ячейки в столбце «Дата» числовым форматом **Дата**;
- отформатировать ячейки в столбцах Получено, Платеж, Баланс числовым форматом **Финансовый** или **Денежный**;
- отформатировать итоговые ячейки с H18 по H20 числовым форматом **Финансовый** или **Денежный**;
- оформить рамками таблицы;

- написать формулы расчета сумм значений в ячейках с H18 по H20 и т. д.

Как объяснить? Во-первых, наш макрос и так довольно сложен для начинающих пользователей. Мы только что рассматривали неточности в ходе его создания. Если усложнить макрос при его создании, был риск не создать его никогда: число ошибок и неточностей на каждом шаге только бы увеличивалось. Во-вторых, постоянно придется изменять макросы, и нужно научиться этому.

На самом деле ничего сложного в усовершенствовании макроса нет. Вы можете сделать это самостоятельно и убедиться: это действительно так. При доработке макроса форматирования заголовка было так: создали отдельный самостоятельный макрос, который делает только то, что мы упустили из виду при создании исходного макроса (форматирование ячеек в столбце Дата числовым форматом Дата). Затем открываем этот макрос, копируем программные строки (это строки без строк комментария и последней строки End Sub) и вставляем их в наш исходный макрос перед строкой End Sub. Так по очереди можем устранить все недостатки исходного макроса.

Пусть вас не смущает длина программы: на выполнение макроса это не сильно влияет. Даже длинный программный код, который мы первоначально создавали, идет довольно быстро, а после усовершенствования стал идти значительно быстрее.

Мы рассмотрели, как исключить отдельные программные строки из процесса выполнения макроса. Это несложно. Сложнее — добавить программные строки в текст ранее созданного.

Одно из решений — создать макрос заново и внести изменения. Рассмотрим недостатки этого метода. Во-первых, при создании нового макроса вы можете внести в него не только новые конструкционные изменения, но и новые ошибки. Во-вторых, должны постоянно помнить о последовательности выполнения операций. Нет гарантий, что не забудете хотя бы один шаг. Наконец, при создании такого большого и сложного макроса, как Балансовый отчет, опять придется удалять ненужные программные строки (которые были созданы при просмотре разных документов, прокрутке линейки и т. д.). Поэтому после создания макроса таким

путем вам придется опять тратить много времени на его анализ и вычищать новые мелкие конструкционные недостатки.

Наконец, нет никакой гарантии, что через некоторое время опять что-то не изменится в законодательстве.

Поэтому такой путь не является оптимальным. Самое простое решение — добавлять программные строки в уже готовый макрос. Ранее созданный макрос мы сохранили под именем Макрос1.

Приступая к доработке, сначала запускаем макрос создания балансового отчета. Включите запись нового макроса, например под именем МакросФорматБаланс. Выполните следующие действия.

1. Выделите ячейки с A8 по A17. Выполните команду **Главная ⇒ Формат ⇒ Формат ячеек**. В окне **Формат ячеек** на вкладке **Число** в списке **Числовые форматы** выберите значение **Дата**. В списке **Тип** выберите формат в виде ДД.ММ.ГГ. Нажмите на кнопку **ОК**.

2. Выделите ячейки с H8 по H17. Выполните команду **Главная ⇒ Формат ⇒ Формат ячеек.** В окне **Формат ячеек** на вкладке **Число** в списке **Числовые форматы** выберите значение **Финансовый.** Нажмите **ОК.**

3. Выделите ячейки с I8 по I17. Выполните команду **Главная ⇒ Формат ⇒ Формат ячеек**. В окне **Формат ячеек** на вкладке **Число** в списке **Числовые форматы** выберите значение **Финансовый**. Нажмите **ОК**.

4. Выделите ячейки с J8 по J17. Выполните команду **Главная ⇒ Формат ⇒ Формат ячеек**. В окне **Формат ячеек** на вкладке **Число** в списке **Числовые форматы** выберите значение **Финансовый.** Нажмите **ОК**.

5. Выделите ячейки с H18 по H20. Выполните команду **Главная ⇒ Формат ⇒ Формат ячеек**. В окне **Формат**

ячеек на вкладке **Число** в списке **Числовые форматы** выберите значение **Финансовый.** Нажмите **ОК.**

6. Выделите ячейки с H4 по H6. Выполните команду **Главная ⇒ Формат ⇒ Формат ячеек.** В окне **Формат ячеек** на вкладке **Число** в списке **Числовые форматы** выберите значение **Финансовый.** Нажмите **ОК.**

7. Выделите ячейку H18. На инструментальной панели **Стандартная** нажмите на кнопку **Автосумма** (∑). Установите указатель левой мыши в ячейку H8, нажмите левую клавишу мыши и, не отпуская ее, протащите до ячейки H17 включительно. Отпустите мышь. Так мы указываем диапазон суммирования. Опять нажмите на кнопку **Автосумма**.

8. Выделите ячейку H19. Нажмите **Автосумма,** укажите диапазон суммирования и опять нажмите на кнопку **Автосумма.**

9. Выделите ячейку H20. Нажмите **Автосумма**, укажите диапазон суммирования и опять нажмите на кнопку **Автосумма.**

Отключите запись макроса МакросФорматБаланс. Выполните команду **Разработчик ⇒ Макрос ⇒ Макросы**. Выделите макрос МакросФорматБаланс. В окне **Макрос** нажмите **Изменить**, чтобы просмотреть текст созданной нами программы:

```
Sub МакросФорматБаланс()
'
' МакросФорматБаланс Макрос
' Макрос записан 27.11.2022 (Шитов)
'

'
    Range("A8:A17").Select
    Selection.NumberFormat = "dd/mm/yy;@"
    Range("H8:H17").Select
    Selection.NumberFormat = "_($* #,##0.00_); _($*
(#,##0.00);_($* ""-""??_);_(@_)"
```

```
    Range("I8:I17").Select
    Selection.NumberFormat = "_($* #,##0.00_); _($*
(#,##0.00);_($* ""-""??_);_(@_)"
    Range("J8:J17").Select
    Selection.NumberFormat = "_($* #,##0.00_); _($*
(#,##0.00);_($* ""-""??_);_(@_)"
    Range("H18:H20").Select
    Selection.NumberFormat = "_($* #,##0.00_); _($*
(#,##0.00);_($* ""-""??_);_(@_)"
    Range("H4:H6").Select
    Selection.NumberFormat = "_($* #,##0.00_); _($*
(#,##0.00);_($* ""-""??_);_(@_)"
    Range("H18").Select
    Selection.FormulaR1C1 = "=SUM(R[-10]C:R[-1]C)"
    Range("H19").Select
    Selection.FormulaR1C1 = "=SUM(R[-11]C[1]:R[-2]
C[1])"
    Range("H20").Select
    Selection.FormulaR1C1 = "=SUM(R[-12]C[2]:R[-3]
C[2])"
End Sub
```

Задача — добавить текст программы только что созданного макроса МакросФорматБаланс к тексту программы Макрос1, который создает Балансовый отчет. Для этого нужно выделить исполняемую часть макроса МакросФорматБаланс, вырезать ее (или скопировать, если текст макроса МакросФорматБаланс потребуется для дальнейшей работы) и вставить в текст макроса Макрос1. Исполняемой частью макроса являются программные строки, за исключением строк Sub [Имя макроса], End Sub, строк с комментарием. Из них Sub [Имя макроса] и End Sub нельзя копировать ни в коем случае. Строки с комментарием не влияют ни на что, поэтому их не копируем только для экономии места.

В данном случае фрагмент, который мы хотим добавить к макросу Макрос1, начинается со строки Range("A8:A17").Select и заканчивается строкой Selection.FormulaR1C1 = "=SUM(R[-12]C[2]:R[-3]C[2])" включительно.

Откройте текст программы макроса Макрос1 командой **Разработчик ⇒ Макрос ⇒ Макросы.** Выделите Макрос1. В окне **Макрос** нажмите на кнопку **Изменить.** Установите курсор в начало последней строки End Sub, нажмите на Enter для создания пустой строки. Из макроса МакросФорматБаланс скопируйте исполняемый фрагмент программы. Это можно сделать так: выделите строки, которые собираетесь скопировать, и нажмите на сочетание клавиш Ctrl + C (или Ctrl + X, если необходимо эти строки вырезать из макроса). На пустую строку в макросе Макрос1 вставьте скопированный в буферную память фрагмент. Для этого можно нажать на сочетание Shift + Insert (или Ctrl + V).

Для вырезания, копирования и вставки фрагмента в программе Microsoft Visual Basic имеются также специальные кнопки Cut, Copy, Paste и команды раздела меню Edit с этими же именами: Cut, Copy, Paste. Лично я отдаю предпочтение сочетанию клавиш, так как это осуществляется только с использованием клавиатуры и без участия мыши.

В данном случае мы решили вставлять фрагмент программы в самый конец макроса Макрос1. Форматирование происходит после того, как Балансовый отчет уже создан. Но если вы будете создавать макрос для других целей, ничто не мешает вам вставлять фрагмент программы в другие части макроса — в начало, в середину или в другое место. Нужно только понимать, что вставляемый фрагмент может разорвать выполнение созданного ранее макроса. Пока вы не умеете определять место вставки в тело макроса, и мы вставляем наш фрагмент в конец ранее созданного.

Создав макрос, мы вдруг вспомнили, что забыли оформить Балансовый отчет рамками. Это обычная ситуация — вам не раз придется дорабатывать свой макрос. Постоянно изменяется ситуация, для которой мы изначально создавали программы. Появляются другие отделы-заказчики или организации-заказчики, которые начинают понимать нужность и простоту наших отчетов, изменяется законодательство и т. д.

Приступая к доработке макроса, сначала запускаем макрос создания балансового отчета. Затем включаем запись нового макроса, например под именем МакросРамки. Выделите заполняемые ячейки в Балансовом отчете с A1 по J20. Выполните

команду **Главная ⇒ Формат ⇒ Формат ячеек.** В окне **Формат ячеек** перейдите на вкладку **Граница** и нажмите на кнопку **Внешние** и **Внутренние.** Нажмите **ОК**. Щелкните по любой ячейке для снятия ранее сделанного выделения (я щелкнул по G21, как вы увидите позже). Отключите запись макроса МакросРамки. Выполните команду **Разработчик ⇒ Макрос ⇒ Макросы**. Выделите макрос МакросРамки. В окне **Макрос** нажмите на кнопку **Изменить**, чтобы просмотреть текст созданной программы:

```
Sub МакросРамки()
'
' МакросРамки Макрос
' Макрос записан 27.11.2022 (Шитов)
'

'
Range("A1:J20").Select
    Selection.Borders(xlDiagonalDown).LineStyle =
xlNone
    Selection.Borders(xlDiagonalUp).LineStyle =
xlNone
    With Selection.Borders(xlEdgeLeft)
        .LineStyle = xlContinuous
        .Weight = xlThin
        .ColorIndex = xlAutomatic
    End With
    With Selection.Borders(xlEdgeTop)
        .LineStyle = xlContinuous
        .Weight = xlThin
        .ColorIndex = xlAutomatic
    End With
    With Selection.Borders(xlEdgeBottom)
        .LineStyle = xlContinuous
        .Weight = xlThin
        .ColorIndex = xlAutomatic
    End With
    With Selection.Borders(xlEdgeRight)
        .LineStyle = xlContinuous
        .Weight = xlThin
```

```
        .ColorIndex = xlAutomatic
    End With
    With Selection.Borders(xlInsideVertical)
        .LineStyle = xlContinuous
        .Weight = xlThin
        .ColorIndex = xlAutomatic
    End With
    With Selection.Borders(xlInsideHorizontal)
        .LineStyle = xlContinuous
        .Weight = xlThin
        .ColorIndex = xlAutomatic
    End With
Range("G21").Select
End Sub
```

Осталось только вырезать (или скопировать) текст исполняемой программы и вставить этот фрагмент в макрос Балансового отчета. Чтобы открыть текст макроса, опять выполните в Microsoft Excel команду **Разработчик ⇒ Макрос ⇒ Макросы**. Выделите Макрос1. В окне **Макрос** нажмите на кнопку **Изменить** и перейдите в самый конец текста программы. Установите курсор перед последней строкой (End Sub) и нажмите на Enter для создания пустой строки. В макросе МакросРамки выделите и вырежьте текст программы (начиная со строки Range("A1:J20").Select и заканчивая строкой Range("G21").Select включительно). Вырезать фрагмент программы можно с использованием сочетания Ctrl + X. Вставьте вырезанный фрагмент в подготовленную ранее пустую строку в макросе Макрос1. Это можно сделать с помощью сочетания клавиш Shift + Insert (или Ctrl + V). Если макрос МакросРамки более не нужен, то и сохранять его не нужно. Если он потребуется в дальнейшем, из него нужно не вырезать фрагмент программы, а копировать путем выделения и использования клавиш Ctrl + C.

Поработайте с готовым Балансовым отчетом. Вводите в ячейки данные, изменяйте их, удаляйте. Балансовый отчет реагирует на все изменения, отражая их как в одной ячейке, так и в итоговых.

Макрос окончательно готов. После нескольких дополнительных доработок (удаление и вставка новых фрагментов в основной

макрос) вы получили навыки корректировки макросов. Что касается корректировки текста макроса, этому будет посвящена вся оставшаяся часть книги.

Использование личной книги макросов

При создании макроса всегда предлагаются три варианта сохранения макроса.

Личная книга макросов — будет доступен для любой книги Excel.

Новая книга — макрос создается для новой книги Excel.

Эта книга — создается только для этой книги Excel. Для других этот макрос будет недоступен.

Чтобы макрос был доступен во всех книгах, необходимо сохранить его в Личной книге макросов.

Пока в ней не сохранен ни один макрос, книга недоступна. Поэтому сохраните в Личной книге макросов любой макрос.

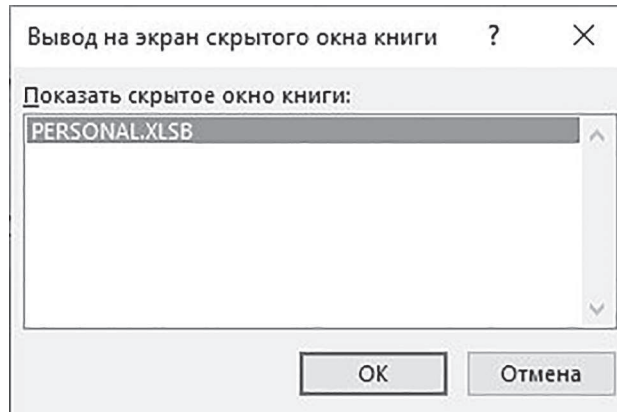

Рисунок 6. Отображение скрытого окна Personal.xlsb

С открытием существующей или созданием новой книги Личная книга макросов всегда присутствует, но по умолчанию скрыта. Чтобы ее открыть, необходимо выполнить команду **Вид** ⇒ **Отобразить окно**. Откроется диалоговое окно **Вывод**

на экран скрытого окна книги (рис. 6). Выберите в списке **Показать скрытое окно книги** имя **Personal** и нажмите **ОК**.

Первоначально в Личной книге макросов всего один модуль, но можно добавить и новые модули, и новые рабочие листы.

Вполне вероятна ситуация, когда макрос мог быть создан в текущей книге или в новой книге, а не в Личной книге макросов. Поэтому может потребоваться перетащить этот макрос в Личную книгу.

Для этого откройте книгу, в которой хранится макрос. Выполните команду **Вид ⇒ Отобразить окно**. Выберите в диалоговом окне **Вывод на экран скрытого окна книги** значение **Personal.xlsb**, нажмите **ОК**. Перейдите в книгу, где хранится макрос, не входящий в Личную книгу. Выполните команду **Разработчик ⇒ Макрос ⇒ Макросы**. В диалоговом окне **Макрос** нажмите на кнопку **Изменить**, после чего войдете в модуль редактора Visual Basic, где хранится текст программы. Выделите все строки программы, включая первую строку **Sub** Имя_Макроса() и последнюю **End Sub**. Нажмите Ctrl + X, чтобы вырезать текст программы. Можно воспользоваться и другими командами вырезания, но так быстрее. Перейдите в Личную книгу макросов. Откройте редактор Visual Basic. В окне проектов **Project — VBAProject** найдите проект VBAProject(Personal.xlsb). Если слева от имени этого проекта находится символ раскрытия дерева вхождений в виде плюса (+), щелкните по этому знаку, чтобы его развернуть. Список проекта будет развернут, а плюс станет минусом (–). Выделите модуль **Module**. На инструментальной панели редактора Visual Basic есть кнопка со списком Insert Имя_элемента (вторая кнопка слева) (). Раскройте список и выберите значение **Module**. После этого модуль **Module**, который был в проекте, будет переименован в Module1, а новый модуль станет Module2. Новый модуль после его создания раскрывается автоматически. (Если автоматически не раскрывается или нужно раскрыть какой-либо модуль, щелкните по нему дважды левой клавишей мыши). Нажмите Shift + Insert (или Ctrl + V) для вставки содержимого буферной памяти, а поскольку в буферной памяти находится текст программы, который мы вырезали из предыдущего макроса, то и будет вставлен текст этой программы. Если

вы пользуетесь другими способами вставлять содержимое буферной памяти, можете вставить программу своим способом. Осталось перейти в Excel, выделить книгу Personal, если она не выделена, и сохранить книгу, в которой раньше был макрос. Но перед сохранением нужно обязательно выделить книгу *Personal.xlsb* и выполнить команду **Вид ⇒ Скрыть окно**, чтобы скрыть Личную книгу макросов.

Если в дальнейшем будете выполнять этот макрос, увидите в диалоговом окне ссылку на Личную книгу макросов *Personal.xlsb*. Этот макрос принадлежит уже не отдельной Книге, а Личной книге макросов.

Относительные и абсолютные ссылки в макросах

По умолчанию в макросах используются абсолютные ссылки. Но можно использовать и относительные.

Чтобы сравнить макросы, использующие абсолютные и относительные ссылки, создадим два макроса — первый с абсолютными, второй с относительными ссылками. Примеры будут одинаковые, чтобы их можно было сравнить.

Так как мы создавали макросы только с абсолютными ссылками, никакие дополнительные действия не нужны.

Чтобы записать первый макрос, выполните следующие действия.

1. Перейдите в ячейку A1, как обычно при открытии нового листа.
2. Выполните команду **Разработчик ⇒ Макрос ⇒ Начать запись**.
3. Сохраните макрос под именем МакросЗоопарк.
4. В ячейку A1 введите: План поставки экзотических животных в зоопарк *Алиса и Базилио*.

5. В ячейку C3 введите: 1 полугодие.

6. В ячейку D3 введите: *2* полугодие.

7. В ячейку A4 введите: Свинорогий спиногрыз.

8. В ячейку A5 введите: Пупырчатый камнедав.

9. В ячейку A6 введите: Чешуйчатый грибожуй.

10. В ячейку A8 введите: Итого.

11. Выделите ячейки с A1 по K1.

12. Выполните команду **Разработчик ⇒ Макрос ⇒ Остановить запись**.

Если теперь выполнить команду **Разработчик ⇒ Макрос ⇒ Макросы**, в открывшемся диалоговом окне **Макрос** выбрать записанный макрос МакросЗоопарк и нажать на кнопку **Изменить**, увидим такой программный код:

```
ActiveCell.FormulaR1C1 = _
        "План поставки экзотических животных
         в зоопарк Алиса и Базилио"
    Range("C3").Select
    ActiveCell.FormulaR1C1 = "1 полугодие"
    Range("D3").Select
    ActiveCell.FormulaR1C1 = "2 полугодие"
    Range("A4").Select
    ActiveCell.FormulaR1C1 = "Свинорогий спиногрыз"
    Range("A5").Select
    ActiveCell.FormulaR1C1 = "Пупырчатый камнедав"
```

```
    Range("A6").Select
    ActiveCell.FormulaR1C1 = "Чешуйчатый грибожуй"
    Range("A8").Select
    ActiveCell.FormulaR1C1 = "Итого:"
    Range("A1:K1").Select
```

Чтобы сравнить полученный макрос с абсолютными ссылками с другим макросом с относительными ссылками, необходимо получить этот макрос. Выполните следующие действия.

1. Перейдите в ячейку A1, как обычно бывает при открытии нового листа.
2. Выполните команду **Разработчик ⇒ Макрос ⇒ Начать запись**.
3. Сохраните макрос под именем МакросЗоо2.
4. На риббоне **Разработчик** нажмите на кнопку **Относительные ссылки**.
5. Выполните действия с п. 4 по п. 12, которые выполнялись при создании макроса с абсолютными ссылками.

Если теперь выполнить команду **Разработчик ⇒ Макрос ⇒ Макросы**, в открывшемся диалоговом окне **Макрос** выбрать записанный макрос МакросЗоо2 и нажать на кнопку **Изменить**, то увидим такой программный код:

```
ActiveCell.FormulaR1C1 = _
        "План поставки экзотических животных
в зоопарк Алиса и Базилио"
    ActiveCell.Offset(2, 2).Range("A1").Select
    ActiveCell.FormulaR1C1 = "1 полугодие "
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "2 полугодие "
    ActiveCell.Offset(1, -3).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Свинорогий спиногрыз"
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Пупырчатый камнедав"
```

```
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Чешуйчатый грибожуй"
    ActiveCell.Offset(2, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Итого:"
    ActiveCell.Offset(-7, 0).Range("A1:K1").Select
```

Различия в двух макросах видны с первого взгляда. Во-первых, набор самих команд совершенно другой. Во-вторых, в первом макросе адреса ячеек указываются в явном виде, а во втором макросе ячейки указываются в виде адресов смещения курсора.

Макрос, созданный на основе абсолютных ссылок, ориентируется по адресам ячеек, указанных в макросе. Создавая первый макрос, я специально сделал небольшую ошибку, установил курсор в ячейку A1 до включения записи макроса. Если мы установим курсор в ячейку A1 и выполним команду **Разработчик ⇒ Макрос ⇒ Макросы**, в открывшемся диалоговом окне **Макрос** выберем записанный макрос МакросЗоопарк и нажмем на кнопку **Выполнить**, все действия будут выполнены правильно. Если мы установим курсор в любую другую ячейку, кроме A1, в эту ячейку будет записана строка: *План поставки экзотических животных в зоопарк Алиса и Базилио* (курсив мой), далее, в ячейках, адреса которых находятся в макросе МакросЗоопарк, будут выведены остальные значения. Последняя команда в обоих макросах — выделение ячеек с A1 по K1. Если в первом макросе макрос начал выполняться не с ячейки A1, такого выделения не будет.

Во втором макросе нет адресов конкретных ячеек, вывод значений происходит в матрице со сдвигом относительно предыдущей ячейки, с которой производилось выполнение макроса.

В макросе на основе относительных ссылок невозможна ситуация, которая сложилась в примере с ячейкой A1. Если макрос, созданный на основе абсолютных ссылок, должен выполняться с ячейки A1, для макроса, созданного на основе относительных ссылок, совершенно все равно, с какой ячейки начнется выполнение. Просто адрес этой ячейки станет отправной точкой, от которой будет организован вывод данных. Рассмотрим это подробнее на основе адресов ячеек, на основании которых строился второй макрос.

Перед записью макроса курсор находился в ячейке A1. Следующая запись создавалась в ячейке C3, то есть со сдвигом в матрице 2 ячейки вниз и 2 ячейки вправо. Поэтому и адрес сдвига в этой строке записан как (2, 2). Далее запись создавалась в ячейке D3, то есть со сдвигом относительно ячейки C3 вниз на 0 ячеек и вправо на 1 ячейку. Поэтому адрес сдвига будет (0, 1). Далее запись происходит в ячейке A4, а это сдвиг относительно ячейки D3 вниз на одну ячейку и влево на 3 ячейки. А что такое сдвиг влево? Это отрицательный сдвиг, так как является возвратом. Следовательно, адрес сдвига будет (1, –3).

Теперь, когда мы поняли получение адреса сдвига в матрице ячеек (как положительного, так и отрицательного), можем сказать, что первое число в адресе указывает на вертикальный сдвиг, а второе — на горизонтальный.

Рассмотрим различие команд в обоих макросах.

Метод **Offset** производит сдвиг ячеек относительно предыдущей ячейки. Как это происходит, мы изучили только что.

Синтаксис этого метода следующий:

expression.**Offset**(***RowOffset***, ***ColumnOffse***)

Где:

expression — обязательный аргумент, выражение, которое возвращает объект **Range**;

RowOffset — необязательный аргумент, тип **Variant**, количество строк (положительное, отрицательное или 0 (нуль)), на которое диапазон должен быть перемещен. Положительные величины — перемещение вниз, отрицательные величины — перемещение вверх. Значение по умолчанию — 0;

ColumnOffset — необязательный аргумент, тип **Variant**, количество столбцов (положительное, отрицательное или 0 (нуль)), на которое диапазон должен быть сдвинут. Положительные величины являются сдвигом вправо, отрицательные величины — сдвиг влево. Значение по умолчанию — 0.

Так как при создании макросов можно применять и абсолютные, и относительные ссылки, возникает вопрос: какие ссылки лучше использовать. Нужны и те и другие в зависимости от того, какие вопросы требуется решить. Если требуется, чтобы заполнялись строго указанные ячейки, лучше применять абсолютные ссылки. Если необходим какой-то маневр и некоторая свобода действий, лучше использовать относительные.

3

Интерфейс редактора Visual Basic

Интерфейс редактора Visual Basic состоит из сложного и взаимосвязанного набора составных частей. Он включает в себя.

- Строка меню.
- Инструментальные панели.
- Окно проектов.
- Окно кода Code.
- Окно выполнения.
- Окно формы.
- Панель элементов управления Toolbox.
- Окно Watch.

Кроме этих элементов интерфейса есть много других второстепенных составных частей.

Строка меню

В разделе меню **File** представлены команды по сохранению редактируемого макроса, его печати, операций импорта и экспорта, удалению компонента приложения — формы, рабочего листа, диаграммы, модуля и т. д.

В разделе меню **Edit** — команды по редактированию, поиску, замене, сдвигу текста относительно левого края. Здесь же представлены информационные команды по методам, функциям, константам и т. д.

В разделе меню **View** — команды по открытию или закрытию окон.

В разделе меню **Insert** представлены команды по вставке в приложение формы, модуля, класса модуля и т. д.

В разделе меню **Format** представлены команды по форматированию элементов управления, в частности, по выравниванию, группировке.

В разделе меню **Debug** представлены команды по отладке приложения.

В разделе меню **Run** — команды по компилированию приложения.

В разделе меню **Tools** представлены команды по настройке параметров и опций редактора VBA.

В разделе меню **Add-Ins** представлена команда создания менеджера приложения.

В разделе меню **Window** — команды по организации окон в редакторе VBA.

В разделе меню **Help** — команды по справочной системе VBA.

Инструментальные панели

В верхней части приложения под меню находятся панели инструментов. Их можно настраивать так же, как и любую инструментальную панель.

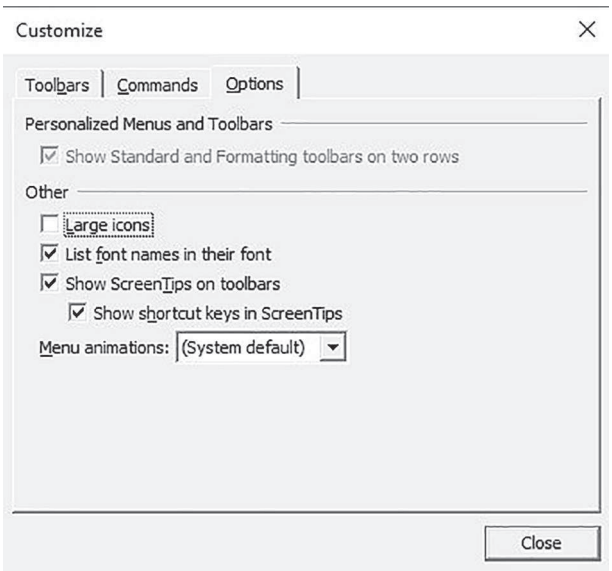

Рисунок 7. Диалоговое окно Customize

По умолчанию открыта только одна инструментальная панель — **Standard**. Чтобы открыть другие, нужно подвести указатель мыши к любой кнопке или свободной области инструментальной панели или полосы меню (она также считается инструментальной панелью) и щелкнуть правой клавишей. Открывается контекстное меню с именами основных инструментальных панелей и командой **Customize**. При выполнении команды **Customize** открывается одноименное диалоговое окно.

Окно **Customize** можно открыть также командой **View** ⇒ **Toolbars** ⇒ **Customize** (Вид ⇒ Инструментальные панели ⇒ Настройка) (рис. 7).

На вкладке Commands предлагаются кнопки по разделам, примерно соответствующим меню. Если нужна кнопка, которой нет на инструментальных панелях, подхватите ее с вкладки Commands и перетащите на одну из открытых панелей. Указатель мыши будет показывать точку вставки. Отпустите мышь.

Окно проектов

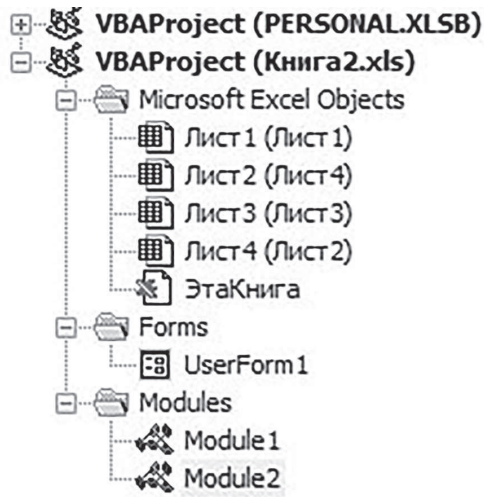

Рисунок 8. Окно проектов

В окне проекта отображаются все элементы приложения: формы, модули, классы и т. п., сгруппированные по категориям (рис. 8). В VBA все разрабатываемые приложения называются проектами. Проект содержит несколько групп компонентов (рабочие книги, листы, формы, модули и т. п.). Все приложения VBA строятся по модульному принципу, поэтому и объектный код состоит не из одного большого файла, а из нескольких частей. Несколько приложений могут объединяться в группы.

Для записи проекта выберите команду **File ⇒ Save КнигаN**. После этого можно задать конкретное имя проекта, который сохраняется не как самостоятельный, а как проект, входящий в какую-то Книгу Excel.

Форма

Форма — это панель, на которой расположены все остальные элементы управления (рис. 9). Элементы управления располагаются либо на форме, либо на рабочем листе. Располагать элементы управления на форме удобнее, чем на рабочем листе. Форму можно открыть, когда это удобно, и закрыть после того, как выполнены действия, запускаемые с этой формы.

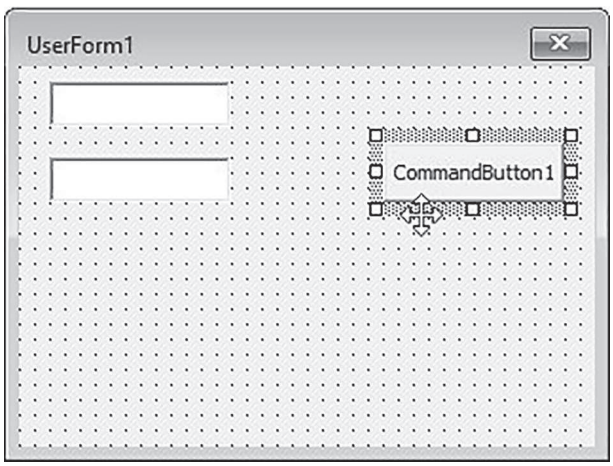

Рисунок 9. Форма

На форме расположена сетка, позволяющая точно расположить элементы управления относительно друг друга. Размер сетки определяется из вкладки **General** диалогового окна **Options**. Узлы сетки примагничивают объекты, располагающиеся на форме.

Чтобы отменить выделение всех элементов управления, щелкните по пустому месту на форме. Чтобы выбрать элементы управления в контейнере, сначала отмените выделение контейнера.

Создание экранных форм VBA

В какой бы программе вы ни работали, в ней, как правило, имеются диалоговые окна. В этом разделе мы научимся создавать такие диалоговые окна.

Вставка новой формы

В открывшемся редакторе Visual Basic на инструментальной панели **Standard** найдите кнопку с раскрывающимся списком **Insert UserForm** (это имя устанавливается по умолчанию). Раскройте список. Выберите значение **UserForm**. Этим вы создадите форму, на которую будете помещать элементы управления.

Другой способ вставить форму — выполнить команду **Insert ⇒ UserForm**.

Первой новой форме по умолчанию присваивается имя **UserForm1**. Каждая следующая форма в приложении будет увеличивать свое имя на единицу. Имя формы можно менять в окне свойств **Properties**. Для этого на вкладке **Alphabetic** найдите самую верхнюю строку (**Name**). В поле свойства напишите новое имя формы (если это необходимо) английскими или русскими буквами.

В окне **Properties** есть вкладки **Alphabetic** и **Categorized**. Разница между ними в том, что на вкладке **Alphabetic** все свойства рассортированы по алфавиту, а на вкладке **Categorized** — по категориям. По умолчанию открыта вкладка **Alphabetic**. С какой вкладкой работать, решает пользователь.

Многие путают имя формы с заголовком формы. Имя формы (**Name**) — имя, под которым происходит обращение к форме в программе, и под этим же именем форма фигурирует в окне проектов. Заголовок формы (**Caption**) — текст в системной полосе формы.

На вставленной форме нет ни одного элемента управления, так как программа не знает, какие из них конкретно вам потребуются. На форме есть разметочная сетка, на выходной форме она не будет видна. Обычно устанавливаемые элементы управления привязывают к верхнему левому углу ячейки сетки.

Поверхность формы может быть контейнером для вывода на нее картинки. При этом картинку можно выводить в виде обоев, то есть повторяющейся непрерывной картинки. Выбирают ее с помощью свойства **Picture**, при нажатии на которое в правой части визуализируется кнопка с многоточием. При нажатии на нее выводится обычное диалоговое окно для выбора картинки.

В одном приложении может быть несколько форм. Они открываются и закрываются командами, которые необходимо написать.

Рассмотрим, как создать приложения (создаются две формы). И на первой, и на второй будет по одной командной кнопке **CommandButton**, этого вполне достаточно. Кнопка на форме **UserForm1** будет открывать **UserForm2**, а кнопка на **UserForm2** — закрывать форму **UserForm2**. Форма **UserForm1** в нашем приложении будет основной, а форма **UserForm2** — вспомогательной. Поэтому совершенно естественно, что управление открытием формы **UserForm2** будет находиться на **UserForm1**. Закрыть **UserForm2** можно как с первой, так и со второй формы. Но так как необходимо задействовать и форму **UserForm2**, мы будем управлять закрытием этой формы с нее же.

Обработчик щелчка кнопки, расположенной на форме **UserForm1,** будет такой:

```
Private Sub CommandButton1_Click()
UserForm2.Show
End Sub
```

```
Обработчик щелчка кнопки на форме UserForm2 будет
такой:
Private Sub CommandButton1_Click()
Unload UserForm2
End Sub
```

Рассмотрим эти две процедуры.

В первой форма **UserForm2** открывается методом **Show**. Существует несколько способов открыть форму. Но в нашем случае наиболее подходящий — с помощью метода **Show**. Имя загружаемой формы указано в явном виде.

При открытии методом **Show** первая форма будет недоступна до тех пор, пока не закроем **UserForm2**. Поэтому мы поступили логично, управляя закрытием формы **UserForm2** с нее же.

Во второй процедуре используется команда **Unload**, выгружающая форму из памяти. Есть два способа указать выгружаемую форму: указать ее имя явно или вместо имени — слово **Me** (то есть Я или Меня). Так как управление выгрузкой происходит с формы **UserForm2**, можно использовать оба варианта. При выгрузке **UserForm2** с формы **UserForm1** нужно явно указать имя **UserForm2**. Если, пытаясь выгрузить форму **UserForm2** (при другом методе, не **Show**), использовать **Me**, вместо **UserForm2** будет выгружена форма **UserForm1**.

Команда **Unload** не является обязательной: закрыть форму можно щелчком по кнопке **Закрыть** на системной полосе. Но чтобы приложение выглядело доработанным, конечно, необходимо предусматривать закрытие с помощью специальных кнопок или щелчков, например, по форме.

В некоторых программах, которые я видел у своих знакомых, программирующих на VBA, выгрузка формы происходит не щелчком по кнопке на форме, а щелчком по самой форме, то есть они пишут следующий код:

```
Private Sub UserForm_Click()
Unload UserForm2
End Sub
```

В этом случае при щелчке по форме она закрывается. Несмотря на относительную простоту этой части приложения (нет лишней кнопки на форме), этот вариант не очень удобен: любой щелчок (например, случайный мимо кнопки, которая должна выполнять какой-нибудь расчет, выборку или сортировку) приведет к закрытию формы и, скорее всего, к потере произведенных настроек.

Кроме метода **Show** имеется оператор **Load**. Но он не отображает форму визуально, а только загружает ее в память. Для визуализации формы все равно нужен метод **Show**. Например:

```
Private Sub UserForm_Initialize()
Load UserForm2
UserForm2.Show
End Sub
```

Эта процедура записана для формы **UserForm1**. При загрузке **UserForm1** она не видна, видна форма **UserForm2**. Только после удаления **UserForm2** становится видна форма **UserForm1**.

Как правило, при загрузке формы записывают и оператор **Load** для загрузки ее в память, и метод **Show** для визуализации. Это необходимо, чтобы прорисовка формы происходила быстрее.

Рассмотрим метод **Hide**, который скрывает объекты, в данном случае форму. Метод скрывает форму от визуализации, но не удаляет ее из памяти. В следующем примере мы будем закрывать форму **UserForm2** щелчком по ней. Щелкните дважды по форме **UserForm2** для создания первоначального программного кода и впишите текст, чтобы процедура выглядела следующим образом:

```
Private Sub UserForm_Click()
UserForm2.Hide
End Sub
```

Обратите внимание, форма скрывается гораздо быстрее, чем закрывается: ее не нужно выгружать из памяти. Различие во времени заметно даже на быстродействующем компьютере,

хотя в этом примере мы пользовались на обеих формах всего одной-двумя кнопками. На сложных формах время будет существенно отличаться в зависимости от того, какой метод вы выбираете, **Hide** или **Unload**.

Вместо записи:

UserForm2.Hide

можно использовать запись:

Me. Hide

Она закрывает текущую (активную) форму.

Очень часто метод **Hide** используется при запуске процедуры, после которой форма временно не нужна. Например:

```
Private Sub CommandButton2_Click()
Worksheets("Лист1").Range("A17").Select
Selection.Formula = "=Sum((A1):(A15))"
MsgBox "Расчет произведен!"
Me. Hide
End Sub
```

В этом примере производится расчет суммы ячеек с A1 по A15, результат заносится всегда в ячейку A17, после чего сообщается, что расчет произведен. Форма становится невидимой, но из памяти не выгружается. Чтобы открыть невидимую форму, можно использовать метод Show, при этом время на визуализацию формы, загруженной в память, минимально.

Основные элементы управления

На панели **Toolbox** находятся 15 элементов управления, которые загружаются при создании нового приложения:

- **Label** — надпись или метка;
- **TextBox** — текстовое поле;
- **ComboBox** — раскрывающийся список;
- **ListBox** — линейный список;
- **CheckBox** — индикатор с флажком;
- **OptionButton** — переключатель;

- **ToggleButton** — кнопка с эффектом нажатого состояния;
- **Frame** — рамка;
- **CommandButton** — командная кнопка;
- **TabStrip** — страница с закладкой;
- **MultiPage** — многостраничное окно;
- **ScrollBar** — полоса прокрутки;
- **SpinButton** — счетчик;
- **Image** — изображение;
- **RefEdit** — поле интервала.

Label — надпись или метка

Элемент управления **Label** предназначен для вывода текста, как символьного, так и числового. Текст может быть задан заранее и оставаться неизменным, а может меняться в зависимости от того, какая команда была выполнена. Изменение текста с клавиатуры и другими ручными способами не допускается.

Текст задается в свойстве **Caption**. Первоначально он задан так же, как и имя, — **Label1**. Порядковый номер элемента управления указывается сзади него автоматически. Следовательно, если мы возьмем одну метку **Label**, она получает имя **Label1**, если вторую — **Label2**, и т. д. Если вы не хотите, чтобы указанный текст показывался в метке, сотрите его либо впишите туда свой. После компиляции он отразится в метке. Все параметры шрифта находятся в свойстве **Font**. Чтобы открыть его, надо щелкнуть на значке «…» слева от этого свойства. Если свойство **Font** не выделено, то и значок «…» слева от этого свойства не виден. Значок показывает, что после нажатия на него будет открыто какое-то диалоговое окно. После нажатия на этот значок откроется знакомое нам диалоговое окно **Шрифт**, в котором можно выбрать **Шрифт**, **Размер**, **Начертание**, а также **Видоизменение** шрифта — **Подчеркнутый** или **Зачеркнутый**. Здесь же можно просмотреть образец выбранного. Нажмем на кнопку **ОК** — загрузятся сделанные изменения, и закроется окно **Шрифт**.

Свойство **Autosize** определяет авторазмер. Изначально установлено в **False**, что запрещает устанавливать авторазмер. Если надпись в метке слишком длинная, она продолжается на следующих строках. При установке свойства **Autosize** в **True** ширина

элемента управления **Label** подстраивается под ширину текста надписи.

Первоначальное имя элемента управления задается в свойстве **(Name)**. Если посмотреть на это имя сразу после установки элемента управления на форму, это будет **Label1**. Имя можно оставлять заданное, можно менять. При изменении используются как английские, так и русские буквы, а также цифры и символ подчеркивания (_). Другие символы задавать нельзя. Если случайно задать другие недопустимые символы, появится сообщение об ошибке и недопустимый символ будет удален из имени автоматически.

Свойство **WordWrap** определяет допустимость переноса продолжения строки. Чтобы иметь такую возможность, установите свойство **Autosize** в **False**, а свойство **WordWrap** — в **True**. Здесь нужны подробности. Если установим **Autosize** в **True**, а **WordWrap** — в **False**, надпись будет продолжаться вправо, даже если ширина элемента управления меньше этой надписи. Если установим **Autosize** в **True и WordWrap** — в **True**, длинная надпись не будет увеличиваться вправо, а будет переноситься на другие строки при достижении ширины элемента управления. То есть свойство **WordWrap** как бы затеняет свойство **Autosize**. Автоширина хотя и включена, но не работает. Если же **Autosize** установлена в **False**, независимо от значения свойства **WordWrap** ширина элемента управления не изменяется, а не умещающаяся часть надписи обрезается. Например, нужно вывести надпись «Сарстройтранссарай». Но по ширине умещаются только символы «Сарстройтранс». Следовательно, эти символы и будут выведены в надписи, а *сарай* обрежется. Поэтому при работе с элементом управления **Label** необходимо следить за свойством **Autosize**.

Свойство **BorderStyle** определяет наличие рамки вокруг надписи: выберете значение **fmBorderStyleNone** — рамки не будет, значение **fmBorderStyleSingle** — вокруг надписи появится рамка. Цвет рамки определяет **BorderColor**. При выделении этого свойства в левой части появляется маленькая кнопка с треугольным значком. Нажав на нее, увидим список цветовых констант. Если ни один из вариантов вас не устраивает, есть другие способы управления цветом, например функция RGB.

Цвет текста определяется свойством **ForeColor**. Цвет фона — свойством **BackColor**. Выбираем цветовое решение так же, как и в **BorderColor**.

Выравнивание текста надписи управляется свойством **TextAlign**. По умолчанию выбрано значение **fmTextAlignLeft**, то есть выравнивание по левому краю. Другие значения этого свойства: **fmTextAlignCenter** — выравнивание по центру и **fmTextAlignRight** — выравнивание по правому краю.

Свойство **SpecialEffect** определяет внешний вид элемента управления **Label**:

- **fmSpecialEffectFlat** — специального эффекта нет;
- **fmSpecialEffectRaised** — элемент управления приподнят над формой;
- **fmSpecialEffectSunken** — элемент управления вдавлен в форму;
- **fmSpecialEffectEtched** — рамка вокруг элемента управления утоплена в форму;
- **fmSpecialEffectBump** — рамка вокруг элемента управления приподнята над формой.

Свойство **Enabled** определяет доступность элемента управления. При значении **True** он доступен, а при **False** — недоступен.

Свойство **Visible** определяет видимость элемента управления. При значении **True** элемент управления виден на форме или на другой панели, а при значении **False** — невидим.

Свойства **Width** и **Height** определяют ширину и высоту элемента управления в пикселах. Эти свойства можно менять как вручную, задавая соответствующие значения в полях, так и с помощью белых маркеров, окружающих элемент управления при его выделении.

Свойство **Picture** определяет рисунок (точнее, значок-пиктограмму) на элементе управления **Label**. Если выделить это свойство, в левой части появляется маленькая кнопка с многоточием «…». Нажав на нее, откроем диалоговое окно **Load Picture**, с помощью которого можно выбрать изображение в надписи (если, конечно, у вас есть изображения такого размера).

Свойство **PicturePosition** определяет порядок выравнивания пиктограммы в надписи и текста надписи:

- **fmPicturePositionLeftTop** — изображение слева, текст надписи справа в верхнем углу;
- **fmPicturePositionLeftCenter** — изображение слева, текст надписи справа и по центру;
- **fmPicturePositionLeftBottom** — изображение слева, текст надписи справа в нижнем углу;
- **fmPicturePositionRightTop** — изображение справа, текст надписи слева в верхнем углу;
- **fmPicturePositionRightCenter** — изображение справа, текст надписи слева и посередине;
- **fmPicturePositionRightBottom** — изображение справа, текст надписи слева в нижнем углу;
- **fmPicturePositionAboveTop** — изображение вверху, текст надписи под изображением справа;
- **fmPicturePositionAboveCenter** — изображение вверху, текст надписи под изображением в центре;
- **fmPicturePositionAboveBottom** — изображение вверху, текст надписи под изображением в нижней части;
- **fmPicturePositionBelowTop** — изображение внизу, текст надписи над изображением в верхней части;

- **fmPicturePositionBelowCenter** — изображение внизу, текст надписи над изображением в центре;
- **fmPicturePositionBelowBottom** — изображение внизу, текст надписи над изображением в нижней части;
- **fmPicturePositionCenter** — и текст надписи, и изображение посередине, перекрывая друг друга.

Свойство **ControlTipText** определяет надпись (подсказку, хинт), которая будет появляться при подводе указателя мыши к данному элементу управления.

Свойство **MousePointer** определяет вид указателя мыши при подводе к данному элементу управления:

- **fmMousePointerDefault** — по умолчанию;
- **fmMousePointerArrow** — стрелка;
- **fmMousePointerCross** — тонкий черный крестик;
- **fmMousePointerIBeam** — I-образный указатель;
- **fmMousePointerSizeNESW** — двунаправленная стрелка, из нижнего левого угла в верхний правый угол;
- **fmMousePointerSizeNS** — двунаправленная стрелка, снизу вверх;
- **fmMousePointerSizeNWSE** — двунаправленная стрелка, из верхнего левого угла в нижний правый угол;
- **fmMousePointerSizeWE** — двунаправленная стрелка, справа налево;
- **fmMousePointerUpArrow** — тонкая черная стрелка, направленная снизу вверх;
- **fmMousePointerHourGlass** — песочные часы;

- **fmMousePointerNoDrop** — черный круг, перечеркнутый из верхнего левого угла в нижний правый угол;
- **fmMousePointerAppStarting** — белая стрелка с песочными часами;
- **fmMousePointerHelp** — белая стрелка с черным вопросительным знаком;
- **fmMousePointerSizeAll** — черная крестообразная стрелка;
- **fmMousePointerCustom** — пользовательский указатель.

Свойство **MouseIcon** позволяет выбрать значок иконки, который будет появляться при подведении указателя мыши к элементу управления и вместо указателя мыши, если в свойстве **MousePointer** выбрано значение **fmMousePointerCustom**. То есть вместо указателя мыши по умолчанию будет появляться пользовательский указатель.

Свойство **TabStop** определяет отключение элемента управления из обхода клавишей Tab. Если это свойство установлено в **True**, элемент управления будет выделен при нажатии на клавишу Tab в порядке очередности, установленном в списке элементов управления в диалоговом окне **Tab Order**. Если **TabStop** установлен в **False**, этот элемент управления отключается от обхода клавишей Tab. Для элементов управления **Label** это актуально: этот элемент управления предназначен только для вывода текста, ему не нужно передавать фокус, поэтому ему по умолчанию устанавливают **TabStop** в **False**.

Этот элемент управления мы рассмотрели первым, поэтому пришлось подробно остановиться на свойствах. Многие из них будут встречаться в других элементах управления. Там, где они будут повторяться, рассматривать их заново не станем: можете просмотреть их назначение и возможные значения в этом элементе управления.

В элементе управления **Label** выводится строковая информация, то есть с типом данных **String**.

TextBox — текстовое поле

Элемент управления **TextBox** предназначен для ввода и вывода на него как однострокового, так и многострокового текста самого разного характера — строк, чисел, символов.

В текстовом поле, как это обычно делается в среде Windows, можно также выделять текст теми же способами, что и в других программах, с помощью мыши или клавиатуры. С помощью клавиатуры это можно сделать, например, нажав на клавишу Shift и удерживая ее, а затем нажимая клавиши ⇦, ⇨, ⇧, ⇩.

У внешнего оформления **BorderStyle** два подсвойства: если выбрано значение **fmBorderStyleNone**, рамки вокруг текстового поля нет; если **fmBorderStyleSingle** — вокруг текстового поля появляется рамка. Для вывода какой-либо первоначальной информации есть свойство **Text**. Принцип его действия аналогичен свойству **Caption** элемента управления **Label**. Есть свойство **Value**, оно дублирует текст свойства **Text** и по идее должно хранить число. Но так как в VBA имеется тип данных **Variant**, тип данных определяется автоматически как наиболее подходящий.

Многие свойства имеют такое же назначение и такие же значения, что и элемент управления **Label**. Например, **BackColor** определяет цвет фона текстового поля, но появилось и новое свойство **BackStyle**, которое определяет прозрачность цвета фона. Свойство **BackStyle** может принимать значения:

- **fmBackStyleTransparent** — цвет фона игнорируется и принимается цвет фона родителя (Parent), на котором находится данный элемент управления. Если этот элемент управления находится на форме, принимает цвет формы, так как форма в данном случае и есть родитель;

- **fmBackStyleOpaque** — принимается цвет фона, заданный свойством **BackColor**.

Свойство **MultiLine** задает многострочность в элементе управления. Если выбрано значение **False**, разрешается только одна строка. Если выбрано **True**, строк может быть больше.

В многострочном поле для перехода на новую строку можно использовать клавишу Enter. Но следует помнить, что для некоторой кнопки, возможно, установлено свойство **Default = True**. Поэтому нажатие клавиши Enter вызовет срабатывание этой кнопки. Тогда для перехода на новую строку надежнее использовать Ctrl + Enter или Shift + Enter. Если одна из кнопок имеет свойство **Default = True**, а свойство **MultiLine** в элементе управления **TextBox** равно **False**, при нажатии Ctrl + Enter или Shift + Enter будет немедленно, без всяких предупреждений, выполнено действие, предусмотренное при нажатии на кнопку.

Есть свойство **EnterKeyBehavior**, в котором при установленном **True** переход на новую строку осуществляется простым нажатием на Enter. Каждое нажатие на клавишу Enter или на Ctrl + Enter или Shift + Enter переводит курсор на новую строку до тех пор, пока вы явно не укажете на необходимость выхода за пределы элемента управления, например щелчком по форме или любому другому элементу управления. Приведу пример. Допустим, есть элемент управления **TextBox**, в котором свойства **MultiLine** и **EnterKeyBehavior** установлены в **True,** и кнопка, где установлено свойство **Default = True**. При прочих равных условиях нажатие на Enter запустит действие, заложенное в этой кнопке. Если мы находимся в элементе управления **TextBox**, нажатие на Enter добавит новую строку. Чтобы нажать на кнопку, в которой установлено свойство **Default = True**, нужно или непосредственно нажать на эту кнопку, или щелкнуть по форме или любому другому элементу управления (кроме элемента управления **TextBox**). После этого каждое нажатие на Enter будет запускать действия, заложенные в кнопку, где установлено свойство **Default = True**. Кнопки будем изучать немного позднее, там и познакомимся ближе с понятием **Default = True**. Кнопка с таким свойством и значением может быть выбрана только одна.

Независимо от того, какое значение выбрано в свойстве **MultiLine**, **ScrollBars** определяет наличие линеек прокрутки в окне элемента управления **TextBox**. Необходимо помнить, что линейки прокрутки появляются только в том случае, если текстовая строка не умещается в том направлении, в котором вы выбрали линейку. Здесь можно выбрать одно из следующих значений:

- **fmScrollBarsNone** — линейки прокрутки отсутствуют;
- **fmScrollBarsHorizontal** — имеется горизонтальная линейка прокрутки;
- **fmScrollBarsVertical** — имеется вертикальная линейка прокрутки;
- **fmScrollBarsBoth** — имеются и горизонтальная, и вертикальная линейки прокрутки.

Еще раз напоминаю, что заданные линейки автоматически появляются только при необходимости. Если линейки прокрутки не нужны, то есть поле не заполнено по ширине или по высоте, линейки не видны.

Элемент управления **TextBox** — один из самых распространенных. Практически в каждом приложении приходится указывать программе номер листа, имя книги, начальную или конечную ячейку и т. д.

Для нашего примера перенесите на форму две кнопки **CommandButton** и два текстовых поля **TextBox**. Назначение примера следующее: при нажатии на кнопку **CommandButton1** будет открыт Лист3, а сам Лист3 переместится в списке листов на первое место. При нажатии на кнопку **CommandButton2** в текстовом поле **TextBox1** будет выдано число листов в книге (включая диаграммы на отдельных листах и т. д.), во втором текстовом поле **TextBox2** будет выдано имя первого листа в списке листов. Программный код для первой кнопки будет такой:

```
Private Sub CommandButton1_Click()
Sheets("Лист3").Select
Sheets("Лист3").Move Before:=Sheets(1)
End Sub
```

Программный код для второй кнопки будет такой:

```
Private Sub CommandButton2_Click()
TextBox1.Text = ActiveWorkbook.Sheets.Count
TextBox2.Text = ActiveWorkbook.Sheets(1).Name
End Sub
```

ComboBox — раскрывающийся список

Раскрывающийся список **ComboBox** — это комбинированный список, комбинация двух элементов управления: списка со значениями и текстового поля. Поля со списком используются в том случае, если заранее трудно определить значения, которые нужно включить в список, или список содержит слишком много элементов. В таком списке нужное значение можно не только выбирать, но и вводить непосредственно в поле ввода. Новое значение после ввода автоматически помещается в список.

Раскрывающийся список **ComboBox** более компактен, чем похожий на него линейный список **ListBox**.

В списке **ComboBox** разрешается выбрать только одно значение из этого списка. Множественный выбор, то есть одновременный выбор сразу нескольких значений, в списке **ComboBox** не разрешен.

Свойство **Text** содержит текст последнего выбранного элемента раскрывающегося списка.

Свойство **ListRows** задает число строк, которые видны в раскрывающемся списке **ComboBox** без прокрутки. Если в списке меньше **ListRows** строк, полоса прокрутки не устанавливается, так как она не нужна.

Свойство **ColumnCount** отображает число столбцов в элементе управления **ComboBox**. По умолчанию в раскрывающемся списке один столбец, но можно сделать и несколько.

Свойство **DropButtonStyle** определяет внешний вид кнопки, расположенной в раскрывающемся списке. Здесь можно выбрать следующие значения:

- **fmDropButtonStylePlain** — нет никакого значка;
- **fmDropButtonStyleArrow** — треугольный значок, установлен по умолчанию;
- **fmDropButtonStyleEllipsis** — знак многоточия;

- **fmDropButtonStyleReduce** — символ подчеркивания (_).

Свойство **SpecialEffect** определяет внешний вид элемента управления **ComboBox**:

- **fmSpecialEffectFlat** — специального эффекта нет;
- **fmSpecialEffectRaised** — элемент управления приподнят над формой;
- **fmSpecialEffectSunken** — элемент управления вдавлен в форму;
- **fmSpecialEffectEtched** — рамка вокруг элемента управления утоплена в форму;
- **fmSpecialEffectBump** — рамка вокруг элемента управления приподнята над формой.

Свойство **ListStyle** определяет стиль выбора записей. Вы можете выбрать один из двух стилей представления для **ComboBox**. Если **fmListStylePlain**, каждый пункт находится в отдельной строке; пользователь выбирает пункт, выделяя одну строку. Если стиль **fmListStyleOption**, кнопка выбора или индикатор с флажком появляется в начале каждой строки. С этим стилем пользователь выбирает пункт, щелкая кнопку выбора или индикатора с флажком.

ListBox — линейный список

Линейный список **ListBox** позволяет выбирать из списка один или несколько элементов. Можно добавлять новые элементы или удалять существующие. Если не все могут одновременно отобразиться в поле линейного списка, в нем автоматически появляются полосы прокрутки.

Свойство **Multiselect** — это возможность множественного выбора значений в линейном списке **ListBox**. Если множественный выбор запрещен, можно выбрать в списке только одно значение, если разрешен — одновременно выбрать несколько значений. Максимальное число выбранных значений при этом ограничено только их числом в списке.

Свойство **Multiselect** может принимать следующие значения:

- **fmMultiselectSingle** — множественный выбор невозможен. Щелчком мыши или нажатием клавиши пробела в списке можно выбрать только один элемент;
- **fmMultiselectMulti** — простой множественный выбор. Элементы списка выбираются щелчком мыши или нажатием клавиши пробела.
- **fmMultiselectExtended** — расширенный множественный выбор. Можно выбрать несколько элементов с помощью мыши или клавиш управления курсором с использованием клавиш Shift и Ctrl. При использовании мыши без клавиатуры можно выделить только один элемент в списке.

При множественном выборе свойство **Text** содержит текст последнего выбранного элемента линейного списка.

Свойство **ListStyle** определяет стиль выбора записей. Можно выбрать один из двух стилей представления для **ListBox**. Каждый обеспечивает различные пути, чтобы выбирать пункты в списке. Если **fmListStylePlain**, каждый пункт находится в отдельной строке; выбираем пункт, выделяя одну или более строк. Если **fmListStyleOption**, кнопка выбора или индикатор с флажком появляются в начале каждой строки. С этим стилем выбираем пункт, щелкая кнопку выбора или индикатора с флажком. Индикаторы с флажками появляются при условии, что свойство **MultiSelect** установлено в **True**.

Сочетание свойств **ListStyle** и **MultiSelect** дает различные комбинации представления строк на элементе управления **ListBox**.

Для **ListBox** значение свойства **Text** должно соответствовать существующему значению в списке.

Кроме свойств, которые находятся в окне **Properties**, существует еще несколько, которых в этом окне нет. Они задаются программным путем. Рассмотрим их и создадим примеры с использованием этих свойств.

Текущее количество элементов в списке сохраняется в свойстве **ListCount**.

Свойство **ListCount** только для чтения. **ListCount** — количество колонок, которые вы можете переместить. **ListRows** — максимум, чтобы отображаться сразу. **ListCount** всегда на единицу больше, чем самая большая величина для свойства **ListIndex**, поскольку индексные номера начинаются с 0 и счет пунктов начинается с 1. Если ни один пункт не выбран, **ListCount** — 0 и **ListIndex** — 1.

Свойство **Selected** возвращает или устанавливает состояние выбора пунктов в **ListBox**. Синтаксис этого свойства следующий:

object.**Selected**(*index*) [= *Boolean*]

Где:

object — обязательный аргумент, возвращаемый объект;
index — обязательный аргумент, целое с диапазоном от 0 до числа, на единицу меньше, чем количество пунктов в списке;
Boolean — необязательный аргумент, выбор пункта;
True — если выбор сделан и **False** — если выбор не сделан.

Выбранное свойство полезно, когда пользователь может делать множественный выбор. Это свойство можно использовать, чтобы определять выбранные колонки при множественном выборе списка значений. Это свойство можно использовать, чтобы выбирать или отменять выбор колонки в списке. Значение по умолчанию — в текущем состоянии выбора **ListBox**. **Value** или **ListIndex** рекомендованы при единственном выборе списка. В этом случае **ListIndex** возвращает индекс выбранного пункта. При множественном выборе **ListIndex** возвращает индекс строки, содержащейся в пределах прямоугольника фокуса — независимо от того, какое значение действительно выбрано. Если линейный список свойства **MultiSelect** установлен в **None**, только одно значение может быть установлено в свойстве **Selected** в **True**.

Нам потребуется на форме один элемент управления **ListBox**, одно текстовое окно **TextBox**, несколько кнопок

CommandButton. Назначение примера следующее: в **ListBox** создадим список, с которым будем проводить какие-то манипуляции. В случае необходимости — выводить полученное значение в текстовое окно **TextBox**, а кнопками будем инициировать выполнение процедур.

Вначале создадим записи в списке **ListBox1**. Напишите обработчик щелчка первой кнопки:

```
Private Sub CommandButton1_Click()
ListBox1.AddItem "Иванов Петр Сидорович"
ListBox1.AddItem "Штирлиц Максим Исаевич"
ListBox1.AddItem "Попандопуло Кузьма Егорович"
ListBox1.AddItem "Голубков Леонид Мавродьевич"
ListBox1.AddItem "Воробьянинов Аполидевк Харонович"
End Sub
```

Конечно, это не лучший способ занести записи в список. Гораздо проще делать это при создании формы, то есть с применением события **Activate**. Но мы еще не изучали события и напишем таким способом, который нам известен.

Используем метод **AddItem**, который добавляет строку в список.

Вторая кнопка будет удалять все записи из списка. Для этого воспользуемся методом **Clear**. Обработчик второй кнопки будет такой:

```
Private Sub CommandButton2_Click()
ListBox1.Clear
End Sub
```

Первое свойство, которое изучим, — **List(i).** Оно возвращает текст элемента списка по его индексу, где i — номер индекса. Нумерация индексов начинается с 0. Поэтому программный код

```
Private Sub CommandButton3_Click()
TextBox1.Text = ListBox1.List(3)
End Sub
```

возвращает нам выделенную строку в линейном списке после нажатия третьей кнопки. Возвращается текст в записи с индексом 3 (List(3)).