Лекция 6. Применение макросов в MS Excel. Разработка макросов

# На этой лекции мы:

1. Узнаем, что такое макросы и область их применения;
2. Научимся записывать макросы с помощью макрорекордера;
3. Изучим основы языка Visual Basic for Applications и попробуем создать свой макрос с его помощью.
4. Научимся создавать пользовательские функции.

# Материалы к лекции:

Презентация.

Файл примера 1

Файл примера 2

Файл примера 3

# На прошлом уроке:

Мы узнали, что такое надстройки и как их устанавливать и настраивать. Также, мы познакомились с такими надстройками как: «Таблицы данных», «Диспетчер сценариев», «Подбор параметра», «Поиск решения» и «Пакет анализа».

# Основные термины:

Макросы - это программы на языке Visual Basic for Application, который интегрирован в Microsoft Office.

Макрорекордер — это небольшая программа, которая переводит действия пользователя на язык программирования VBA. Результатом служит программный модуль (Module).

Программный модуль - независимая и функционально законченная часть программы, оформленная в виде самостоятельного фрагмента кода, упакованная в макрос.

Редактор Visual Basic (Visual Basic Editor — VBE) – редактор, в котором создаются все программы на языке VBA. Этот редактор представляет собой отдельное приложение, которое доступно при работе с Excel - программа сама открывает редактор VBE, как только это понадобится пользователю.

Процедуры типа подпрограмма – это программный алгоритм действий, который не возвращает собственных значений (как функция), но выполняется, принимая аргументы и меняя в процессе значения каких-либо переменных.

Комментарии — это операторы, которые не выполняются и не компилируются, а предназначены только для описания части кода. Часто, если мы не хотим удалять определенные строки кода и не хотим, чтобы они выполнялись, мы можем прокомментировать их.

Методы – это функции (или процедуры), которые применимы для объектов класса, для которых они написаны.

Локальная переменная – переменная, которая инициализируется (для нее выделяется участок памяти, в котором уже может хранится какое-то значение) в памяти и использовать её значение можно внутри только этой процедуры, а по завершению процедуры переменная выгружается из памяти (обнуляется) и данные по ней теряются.

Тип данных — это множество допустимых значений, которые могут принимать переменные, принадлежащие к этому типу, а также перечень операций, которые можно осуществлять над этими данными.

Процедуры типа «функция»- это процедуры, которые возвращают значения и позволяют создавать пользовательские функции, которые можно использовать в дальнейшем, как и предустановленные в Excel.

# План лекции

[Введение](#_Toc108560458)

[Применение и запись макросов.](#_Toc108560459)

[Основы языка Visual Basic for APPlications (VBA)](#_Toc108560460)

[Пользовательские функции.](#_Toc108560461)

[Выводы](#_Toc108560462)

### Введение

Здравствуйте! Приветствую Вас на заключительном уроке курса «Электронные таблицы в аналитике»! Сегодня мы рассмотрим работу с макросами и основы языка Visual Basic for Applications.

На прошлом занятии мы узнали, что такое надстройки и как их устанавливать и настраивать. Также, мы познакомились с такими надстройками как: «Таблицы данных», «Диспетчер сценариев», «Подбор параметра», «Поиск решения» и «Пакет анализа».

Темой урока сегодня будет создание макросов с помощью макрорекордера и языка Visual Basic for Applications (VBA).

Мы изучим следующие аспекты:

* Макросы и область их применения.
* Запись макросов
* Основы синтаксиса языка Visual Basic for Applications (VBA);
* Разработка макросов и пользовательских функций;

### Применение и запись макросов.

Макросы — это программы на языке Visual Basic for Application, который интегрирован в Microsoft Office.

Рассмотрим задачи, которые можно решить с применением макросов:

* **Автоматизация повторяющихся или рутинных задач.**

Например, Вам необходимо отформатировать 100 таблиц определенным образом. Но, чтобы отформатировать одну таблицу нужно произвести около десяти действий. Макросы помогут свести число действий до одного – запуска макроса кнопкой или комбинацией клавиш: таким образом, форматирование будет производиться в 10 раз быстрее!

* **Обеспечение взаимодействия нескольких программ Microsoft Office.**

Например, из Access «сырые» данные поступают в таблицы Excel и форматируются в соответствии с корпоративным стилем компании. На основании таблицы составляется отчет (с включением диаграмм, графиков и прочего), сохраняется в документе Word и автоматически направляется с помощью Outlook по электронной почте.

* **Создание сложных алгоритмов поиска, которые недоступны обычными средствами Excel.**

Например, когда необходимо выполнять циклический поиск значения на странице.

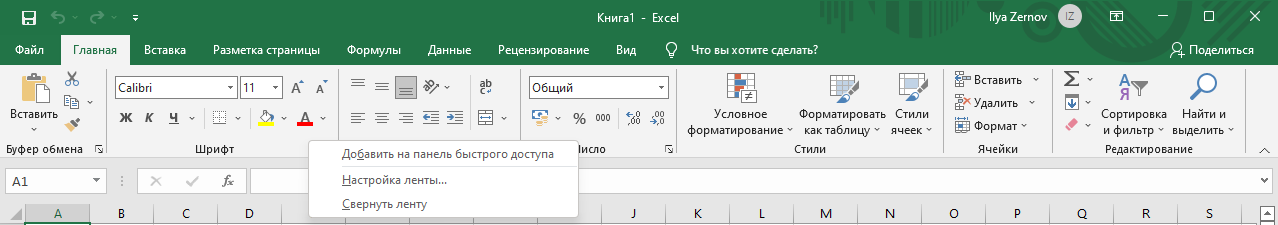
* **Создание новых функций в Excel.**

Например, специфические финансовые функции, которые применяются в рамках Вашей компании.

Для начала необходимо настроить MS Excel. Для работы с макросами необходимо включить отображение вкладки «Разработчик» в ленте Excel, так как изначально она может не отображаться сразу после установки Excel.

Для этого выполним следующие действия:

1. В произвольном месте на ленте Excel нажмем правую кнопку мыши и нажмем на строку ***«Настройка ленты…».***



1. В открывшемся диалоговом окне ***«Параметры Excel»*** появится раздел ***«Настроить ленту».***
2. В отображенном в правой части окна списке установим флажок возле позиции ***«Разработчик».***
3. Нажимаем кнопку ***«ОК»*** и проверяем, появилась ли вкладка ***«Разработчик».***

Один из методов создания макросов – это непосредственно запись действий пользователя с помощью макрорекордера.

Макрорекордер — это небольшая программа, которая переводит действия пользователя на язык программирования VBA. Результатом служит программный модуль (Module).

Программный модуль - независимая и функционально законченная часть программы, оформленная в виде самостоятельного фрагмента кода, упакованная в макрос.

Данный способ имеет следующие преимущества:

* Не требует навыков программирования на VBA и знания его синтаксиса.
* Во многих случаях оптимальнее записать макрос, нежели писать его вручную в редакторе кода – так получается быстрее и можно избежать логических и синтаксических ошибок при написании программы.

Не лишен этот способ и недостатков:

* Случайная ошибка при записи макроса, вызывает необходимость ее повторить или корректировать полученный код в редакторе.
* Пишутся только те действия, которые происходят в окне Excel.
* Не может генерировать код, который включает в себя циклические структуры, а также присваивает переменные, выполняет условные операторы, отображает диалоговые окна и т.д.
* Не позволяет создавать новые функции в Excel.

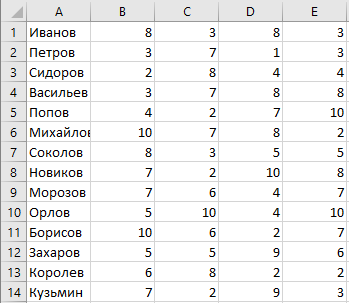
Рассмотрим пример.

#### Пример 1. Запись макроса.

Откройте файл примера 1 из материалов к лекции. Там вы увидите два листа с данными, которые необходимо отформатировать определенным образом. Например, нужно все фамилии выводить шрифтом Arial, курсив, размер 14 пт. Также необходимо сделать серую заливку этих ячеек.

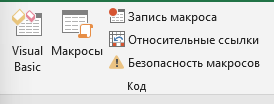
Таким образом нужно отформатировать все листы в файле и оптимизировать процесс с помощью макроса.

Запишем действия, которые производятся при форматировании первого листа с помощью макрорекордера и применим полученный макрос на втором листе.

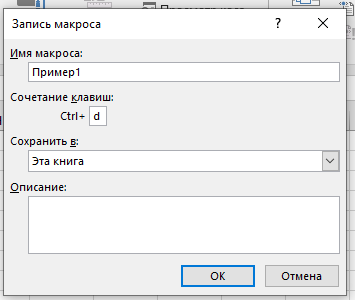


Выполним следующие действия:

1. В открывшейся книге перейдем на вкладку ***«Разработчик»*** и нажмем на кнопку запись макроса.



1. В открывшемся окне ***«Запись макроса»*** наблюдаем ряд параметров



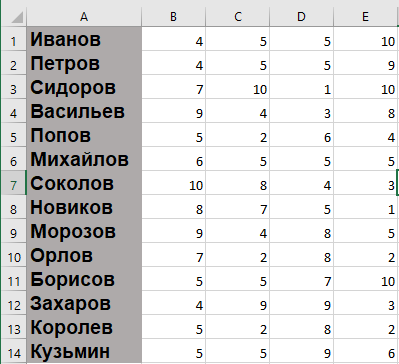
**Имя макроса.** Предоставляет возможность ввести название записываемого макроса. Вы можете использовать имя по умолчанию и изменить его позже, однако лучше сразу назвать макрос правильным именем. В нашем случае пусть будет Пример1.

**Комбинация клавиш.** Позволяет выполнить макрос с помощью комбинации клавиш. Добавим клавишу D, чтобы вызов макроса происходил с помощью комби нации Ctrl+D. Помните о том, что комбинация клавиш, назначенная макросу, переопределяет встроенную комбинацию клавиш (если она есть).

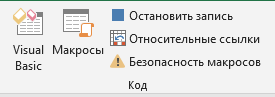
**Сохранить в.** Указывает Excel, где должен храниться макрос, который записывается. По умолчанию Excel помещает записанный макрос в модуль активной рабочей книги. По желанию можно записать его либо в новой рабочей книге, либо в личной книге макросов (Здесь можно сохранить макросы VBA, которые вы считаете особенно полезными. Эта рабочая книга называется Personal.xlsb и хранится в папке XLStart. Загрузка данной рабочей книги происходит после запуска Excel, причем вы сразу же получаете доступ к хранящимся там макросам. Книга Personal.xlsb скрыта, поэтому при обычной работе в Excel она не видна. Файл Personal.xlsb не существует до тех пор, пока не будет записан первый макрос.). Оставим данное поле пока без изменения.

**Описание.** При желании в поле Описание можно ввести описание макроса. Введенный текст отображается в начале кода макроса в виде комментария. Можете заполнить произвольным сообщением.

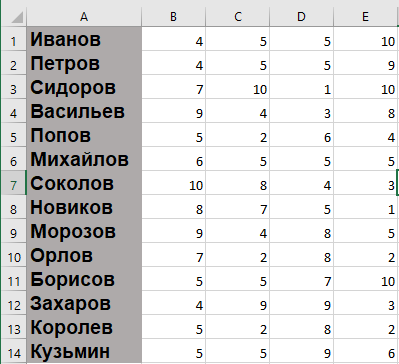
1. После заполнения полей нажмем кнопку ***«ОК».***
2. Выделим столбец с фамилиями и сделаем фон серым, шрифт жирным Arial, размер 14.
3. После чего выделим диапазон *A1:E14*.



1. Нажмите кнопку ***«Остановить запись».***

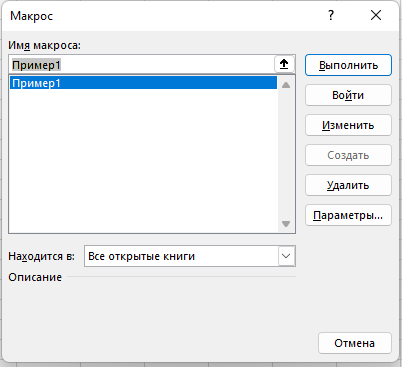


1. Применим записанный макрос с помощью комбинации клавиш Ctrl +D на втором листе



Можно выполнить макрос без использования комбинации клавиш. Нажмите кнопку Макросы во вкладке ***«Разработчик»***.

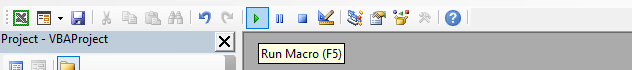
В открывшемся окне можно увидеть записанный нами макрос ***«Пример1».***



Теперь можно менять формат и выделять данный диапазон ячеек просто выполнив макрос.

Выполнить макрос можно следующими способами:

1. Нажатием на кнопку ***«Выполнить»*** в окне ***«Макрос».***
2. Нажатием кнопки ***«Run Macro»*** на панели инструментов в VBE. (О редакторе VBE Вы можете прочитать в следующем разделе)

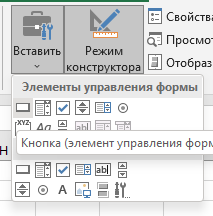


1. Нажатием горячих клавиш, назначенных в окне при создании макроса или уже после записи макроса нажатием кнопки ***«Изменить»*** в окне ***«Макрос».***
2. Нажатием созданной кнопки.

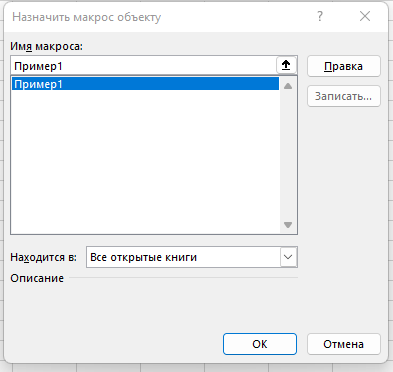
Давайте рассмотрим пример создания подобной кнопки.

Выполним следующие действия:

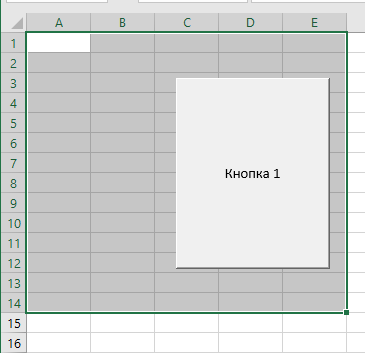
1. В вкладке ***«Разработчик»*** нажмем кнопку ***«Вставить»*** и в выпадающем меню выберем элемент управления ***«Кнопка».***



1. Выделим на листе произвольное поле и отпустим левую кнопку мыши после чего появится окно ***«Назначить объект макросу»***. В данном окне выберем Пример1 и нажимаем кнопку ОК.



1. Наблюдаем на рабочем поле созданную нами кнопку, нажатие на которую приводит к форматированию столбца с фамилиями и выделению диапазона *A1:E14*.



### Основы языка Visual Basic for APPlications (VBA)

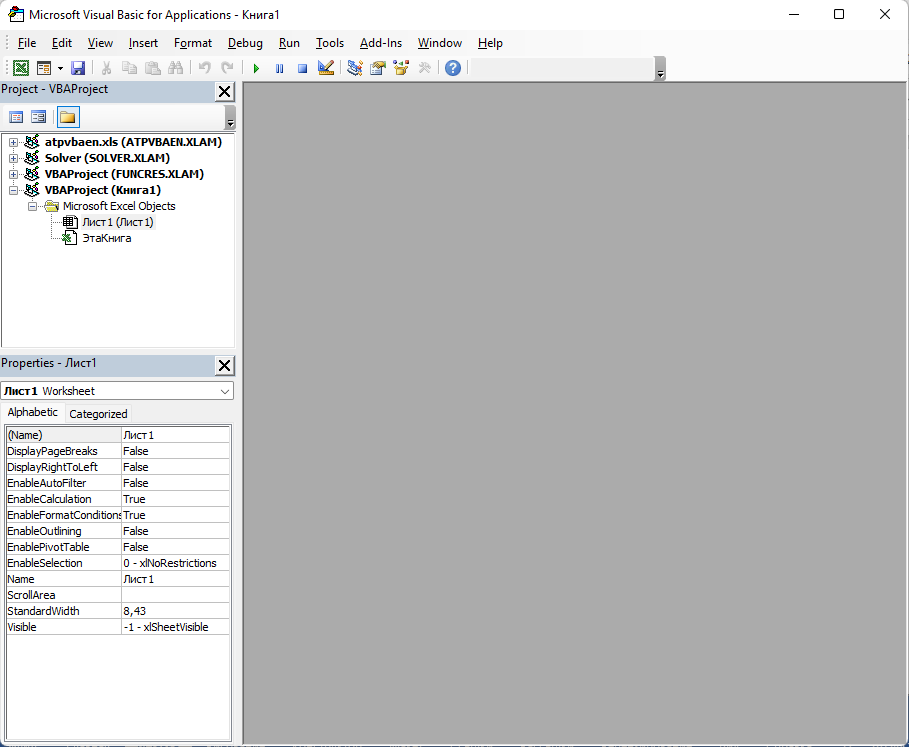
Visual Basic for Applications- это упрощенная версия Visual Basic, интегрированная в линейку продуктов Microsoft Office и ряда других программ.

Все программы на языке VBA создаются с помощью редактора Visual Basic (Visual Basic Editor — VBE). Этот редактор представляет собой отдельное приложение, которое доступно при работе с Excel - программа сама открывает редактор VBE, как только это понадобится пользователю. Единственное, что сделать не получится — это запустить редактор VBE отдельно, поскольку сначала должен быть открыт MS Excel (или другое приложение в которое он интегрирован).

Запустим VBE одним из двух способов:

* нажатием комбинации клавиш *Alt+F11*;
* выбором команды ***Разработчик→ Код→ Visual Basic***.

В зависимости от версии Microsoft Office и от индивидуальных настроек пользователя окно редактора VBE может изменяться. Рассмотрим основные компоненты окна редактора для версии Excel 2019 года.



* **Строка меню.**

Несмотря на то, что в Excel применяется интерфейс в виде ленты, редактор Visual Basic использует классические меню и панели инструментов. Строка меню VBE содержит команды, используемые для управления различными компонентами VBE. Например, Вы можете просмотреть модуль кода для рабочего листа выбрав Лист1 в окне ***«Project»*** и выполнив команду ***View→ Code→ Visual Basic***.



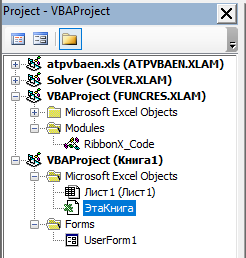
* **Панели инструментов.**

Стандартная панель инструментов, которая по умолчанию находится под строкой меню. Это одна из шести панелей инструментов, используемых в VBE (строка меню также считается панелью инструментов). Панели инструментов VBE работают, как и в Excel: можно задавать специальные настройки для панелей инструментов, перемещать их, отображать другие панели инструментов и т.д.



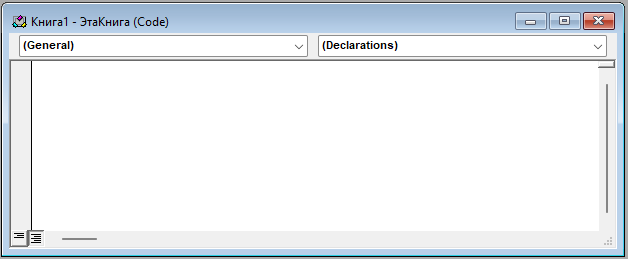
* **Окно Project Explorer.**

В окне Project Explorer отображается древовидная структура всех открытых в данный момент в Excel рабочих книг (включая надстройки и скрытые рабочие книги). Каждая рабочая книга известна как проект. Если в редакторе Visual Basic окно Project Explorer не отображено, нажмите *Ctrl+R*.



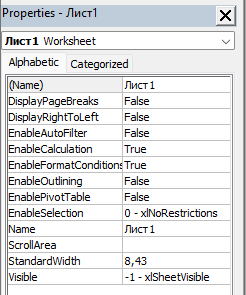
* **Окно кода.**

В окне кода (которое иногда называется Module) содержится код VBA. Для каждого элемента проекта представлено собственное окно кода. Чтобы просмотреть код объекта, дважды щелкните мышью на этом объекте в окне Project Explorer. Например, чтобы просмотреть код объекта Лист1, дважды щелкните на элементе Лист1 в окне Project Explorer. Если вы не создавали для него VBA-код, открывшееся окно будет пустым.



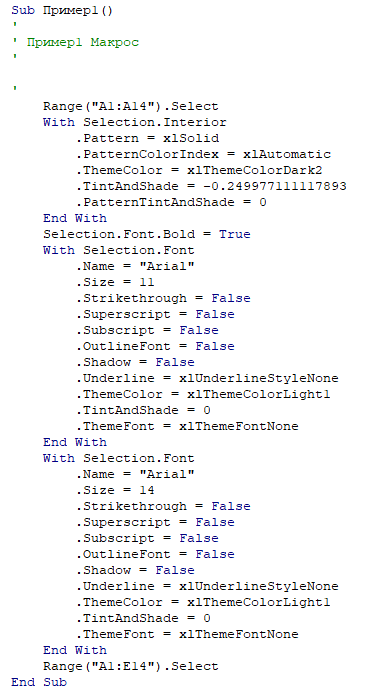
* **Окно свойств (Properties).**

В окне свойств содержатся описание свойств объекта. Чтобы просмотреть свойства объекта, щелкните мышью на этом объекте в окне Project Explorer. Например, чтобы просмотреть код объекта Лист1, щелкните на элементе Лист1 в окне Project Explorer.



При использовании VBA макрос создается в виде подпрограммы в редакторе VBE. Данный способ не имеет ограничений по функционалу, но требует знания языка и, в некоторых случаях, является довольно трудоемким.

Давайте взглянем на структуру макроса, который мы записали в первом примере. Откроем окно ***«Макрос»***, выберем наш макрос ***«Пример1»*** нажмем кнопку ***«Изменить»***. В открывшемся редакторе мы увидим листинг нашего скрипта.



Разберем код.

* Sub Пример1() — ключевое слово Sub объявляет имя, аргументы и код, которые формируют тело процедуры типа подпрограмма – это программный алгоритм действий, который не возвращает собственных значений (как функция), но выполняется, принимая аргументы и меняя в процессе значения каких-либо переменных. Пример подпрограммы: выполнение последовательности действий для форматирования ячеек. Пример функции – функция суммирования, которая в явном виде возвращает значение, которое подставляется вместо вызова функции.

End Sub —означает конец процедуры типа подпрограмма.

* ' Пример1 Макрос— комментарии — это операторы, которые не выполняются и не компилируются, а предназначены только для описания части кода. Часто, если мы не хотим удалять определенные строки кода и не хотим, чтобы они выполнялись, мы можем прокомментировать их. Комментарии начинаются с апострофа (‘).
* **Range("A1:A14").Select**—метод позволяющий выбрать ячейку или диапазон. Методы – это функции (или процедуры), которые применимы для объектов класса, для которых они написаны. Например, для класса диапазонов ячеек в данном случае можно применить метод, представляющий из себя выбор (выделение) этого диапазона. Давайте рассмотрим, как осуществляется работа с ячейкой.

|  |  |
| --- | --- |
| Пример | |
| Range("A1:E14") | обращается к диапазону A1:E14 |
| Range("A1") | обращается к ячейке A1 |

**Range** это функция с помощью которой можно получить доступ к ячейкам в VBA. На вход данная функция берет адреса ячеек, на выходе возвращает множество ячеек.

Альтернативный способ обращения в ячейке функция **Cells**. Данная функция возвращает диапазон только из одной ячейки и берёт строку и столбец, как аргументы. Использование данной функции бывает удобно, когда адреса ячеек участвуют в расчетах и могут изменяться в зависимости от результатов работы.

|  |  |
| --- | --- |
| Пример | |
| Cells(1, 1) | обращается к ячейке A1 |
| Cells(10, 2) | обращается к ячейке B10 |

В случае если изменяются диапазоны удобно комбинирование **Range** и **Cells**.

|  |  |
| --- | --- |
| Пример | |
| Range(.Cells(1, 1), .Cells(10, 1)) | обращается к диапазону A1:A10 |

Часто бывает необходима работа с столбцом с помощью функции **Columns** или строкой с помощью функции **Rows**.

|  |  |
| --- | --- |
| Пример | |
| Columns(2) | обращается к столбцу B |
| Columns("B:F") | обращается к столбцу от B до F |
| Rows(5) | обращается к строке 5 |

* Оператор With... End With - выполняет последовательность операторов, которые многократно ссылаются на единственный объект или структуру. Это используется для упрощения синтаксиса доступа операторов (методов или функций) к членам объекта или структуры. При использовании структуры можно только считывать значения членов или вызвать методы.

|  |  |
| --- | --- |
| With | Выражение, результатом которого является объект. |
| Один или несколько операторов, которые могут ссылаться на члены объекта. | |
| End With | |

* Точка (**.**) – также является оператором класса, структуры, интерфейса или перечисления и используется для доступа к ним. Элемент может быть полем, свойством, событием или методом. Это наглядно видно в нашем примере, где мы меняем свойства наполнения ячеек или шрифта.

Теперь попробуем написать свой макрос.

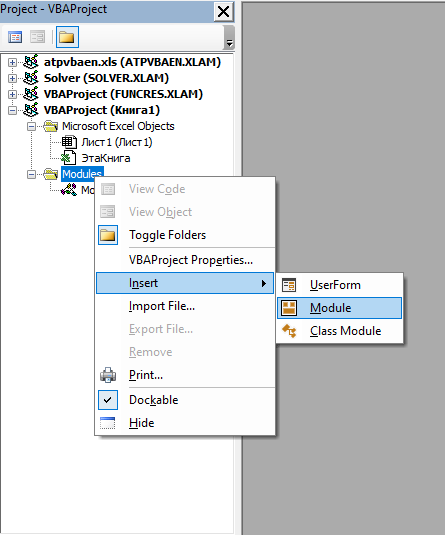
#### Пример 3. Заполнение ячеек произвольным значением в шахматном порядке.

Откройте пустой лист Excel. Задача заключается в том, чтобы заполнить ячейки произвольным значением в шахматном порядке на этом листе.

Для решения задачи напишем подпрограмму на языке VBA.

Выполним следующие действия:

1. Откроем редактор VBE.
2. В окне Project Explorer добавим модуль.



1. В открывшемся окне напишем Sub Пример3(). После чего программа автоматически добавит End Sub.
2. Объявим перемененные необходимые для решения нашей задачи. В нашем случае необходимо определить количество строк и столбцов, которые необходимо заполнить, а для работы цикла заведем два счетчика для их перебора. Синтаксис объявления переменных приведен в таблице ниже:

|  |  |  |  |
| --- | --- | --- | --- |
| оператор области действия | Имя\_переменной | As | тип данных |

Разберем подробнее операторы объявления переменных.

**Оператор области действия**

Dim- данный оператор используется для объявления переменной, значение которой будет храниться только в той процедуре, внутри которой данная переменная объявлена. Во время запуска процедуры такая переменная инициализируется (для нее выделяется участок памяти в котором уже может хранится какое-то значение) в памяти и использовать её значение можно внутри только этой процедуры, а по завершению процедуры переменная выгружается из памяти (обнуляется) и данные по ней теряются. Переменную, объявленную подобным образом еще называют локальной переменной. Для нашей задачи воспользуемся оператором Dim.

|  |
| --- |
| Альтернативные операторы: |
| **Static** - данный оператор используется для объявления переменной, значение которой предполагается использовать внутри конкретной процедуры, но не теряя значения данной переменной по завершении процедуры. Такая переменная инициализируется в памяти при первом запуске процедуры, в которой она объявлена. По завершении процедуры данные по переменной не выгружаются из памяти, однако они не доступны в других процедурах. Как только Вы запустите процедуру с этой переменной еще раз - данные по такой переменной будут доступны в том виде, в котором были до завершения процедуры. Выгружается из памяти такая переменная только после закрытия проекта (книги с кодом).  **Public** - данный оператор используется для объявления переменной, значение которой будет доступно в любой процедуре проекта (в обычных модулях, модулях класса, модулях форм, модулях листов и книг). Переменная, объявленная подобным образом, должна быть объявлена вне процедуры - в области объявлений. Такая переменная будет называться глобальной. |

**Имя\_переменной**

В качестве имен переменных можно использовать буквы (регистр роли не играет – basic не чувствителен к регистру) и числа, также допускается использование символа нижнего подчеркивания, но первой в имени переменной всегда должна быть буква. Длина имени не должна превышать 254 символов. Нельзя использовать в качестве имен для переменных зарезервированные константы редактора VBA (например Sub, Integer и т.п.). Назовем одну из наших переменных Str от слова string.

**As**

As-оператор присваивания.

**Тип данных**

Тип данных — это множество допустимых значений, которые могут принимать переменные, принадлежащие к этому типу, а также перечень операций, которые можно осуществлять над этими данными. Типы переменных используемые в VBA приведены в таблице ниже:

|  |  |  |
| --- | --- | --- |
| Тип данных | Объем занимаемой памяти | Пределы значений |
| Byte | 1 байт | Целые числа от 0 до 255 |
| Boolean | Зависит от реализации платформы | True или False |
| Integer | 2 байта | Целые числа от (-32768) до 32767 |
| Long | 4 байта | Целые числа от (-2147483648) до 2147483647 |
| Single | 4 байта | От (–3.402823Е+38) до (–1.401298Е-45) и от 1.401298Е-45 до 3.402823Е+38 |
| Double | 8 байт | От ±1.79769313486232Е+308  до ±4.94065645841247Е-324 |
| Currency | 8 байт | От (–922337203685477.5808) до 922337203685477.5807 |
| Date | 8 байт | От 01.01.100 до 31.12.9999 |
| String | 10 байт + длина строки | От 0 до 65400 символов для фиксированных строк и чуть более 2 млрд. для строк переменной длины |
| Object | 4 байта | Любой объект |
| Array | В зависимости от количества и размера элементов | - |
| Variant | 16+ байт | Любой из встроенных типов данных |

Выберем тип данных integer и объявим переменную Str и напишем следующую строку кода:

|  |
| --- |
| Dim Str As Integer |

Таким образом мы объявили одну из четырех нужных нам переменных.

Объявим еще две переменные: количество столбцов (Col от слова column) и счетчик строк (counterStr) следующим образом:

|  |
| --- |
| Dim Col As Integer, counterStr As Integer |

Таким образом, в одной строке можно объявить несколько переменных.

И еще одним способом объявим переменную счетчик столбцов:

|  |
| --- |
| Dim counterСol% |

Некоторые типы данных можно объявлять подобным образом. Перечень типов, к которым применим данный метод объявления:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ! - Single | # - Double | $ - String | % - Integer | & - Long | @ - Currency |

Присвоим начальные значения переменным. Присваивание осуществляется оператором =. Напишем следующие строки кода:

|  |
| --- |
| Str = 10  Col = 10  counterStr = 1  counterCol = 1 |

Таким образом мы декларируем, что будем заполнять десять строк и десять столбцов начиная с первых столбца и строки.

1. Для того чтобы начать пробегать по строкам и столбцам нам необходимо создать два цикла For… Next, вложенных друг в друга. Синтаксис цикла For… Next в VBA приведен ниже.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| For | Имя\_переменной\_счётчика | **=** | Начальное значение | To | Конечное значение | Step\* | Значение шага |
| Оператор | | | | | | | |
| Exit For\* | | | | | | | |
| Операторы | | | | | | | |
| Next | Имя\_переменной\_счётчика | | | | | | |

\*так отмечены необязательные атрибуты.

Exit For – это оператор экстренного выхода из цикла (или досрочного завершения его работы) – это бывает необходимо, если получено необходимое значение или выполнено необходимое условие и дальнейшие итерации являются лишними.

Напишем следующие строки кода:

|  |
| --- |
| For counterStr = 1 To Str Step 1  For counterCol = 1 To Col Step 1  Next counterCol  Next counterStr |

1. Теперь введем критерии заполнения ячеек с помощью операторов ветвления If...Then...Else. Синтаксис ветвления:

|  |  |  |
| --- | --- | --- |
| If | Условие | Then |
| Операторы | | |
| Else\* | | |
| Операторы | | |
| End If | | |

\*так отмечены необязательные атрибуты.

Напишем следующие строки кода в тело цикла:

|  |
| --- |
| If ((counterStr Mod 2) = 1 And (counterCol Mod 2) = 1) Or ((counterStr Mod 2) = 0 And (counterCol Mod 2) = 0) Then  Cells(counterStr, counterCol) = 1  End If |

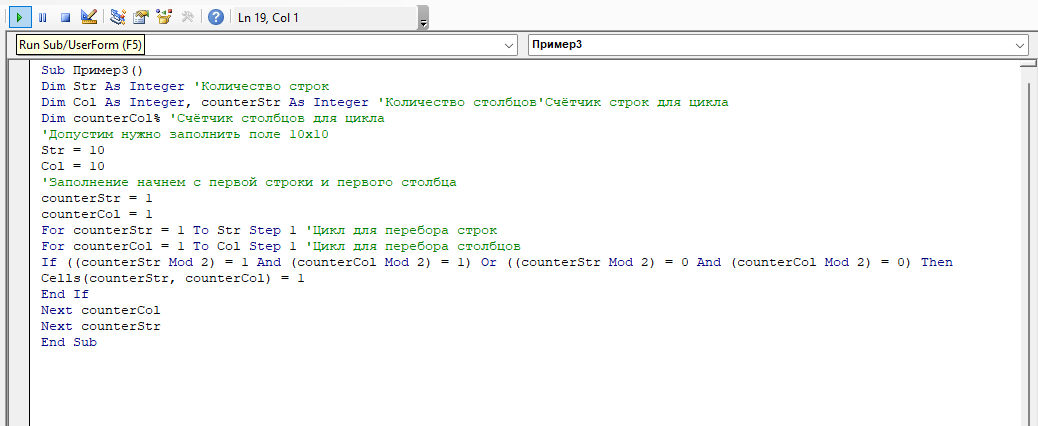
Данные строки значат:

Если (номер строки нечетный И номер столбца нечетный) ИЛИ (номер строки четный и номер столбца четный), то запишем в ячейку единицу.

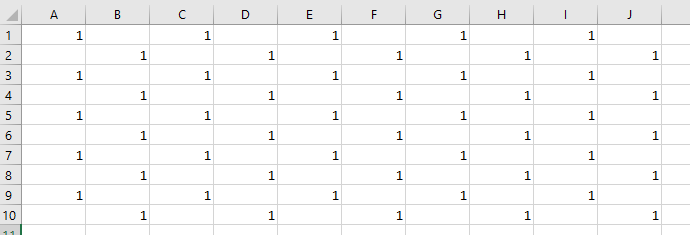
|  |  |
| --- | --- |
| Логические операторы в VBA | |
| Not | Отрицание (Не) |
| And | Умножение (И) |
| Or | Сложение (ИЛИ) |
| Xor | Исключающее сложение (ИЛИ НЕ) |
| Egv | Эквивалентность |
| Imp | Импликация |

Таблица истинности и приоритетность операций соответствует таковым в Excel.

1. Запустим полученный код:



1. Можно увидеть результат выполнения на листе Excel:



Данная программа является простым примером, и, логично предположить, что создание более сложных макросов на языке VBA требует практики и изучения программирования в целом.

### Пользовательские функции.

Помимо процедур типа «подпрограмма», которые мы рассмотрели выше, с помощью VBA можно создавать процедуры типа «функция». Эти процедуры возвращают значения и позволяют создавать пользовательские функции, которые можно использовать в дальнейшем, как и предустановленные в Excel.

Синтаксис написания функции выглядит следующим образом:

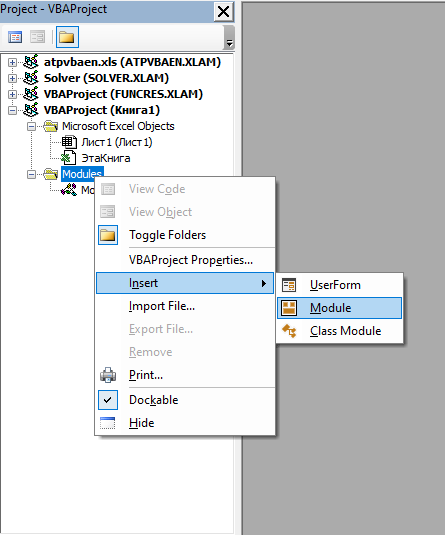
|  |  |  |
| --- | --- | --- |
| Function | Имя | (СписокАргументов)[As ТипДанных] |
| Операторы | | |
| Имя = выражение | | |
| Exit Function\* | | |
| Операторы | | |
| Имя = выражение | | |
| End Function | | |

#### Пример 4. Функция деления

Попробуем создать функцию, которая проверяет, можно ли разделить первый ее аргумент на второй и выводит результат деления, в случае наличия такой возможности.

Выполним следующие действия:

1. Откроем редактор VBE.
2. В окне Project Explorer добавим модуль.



1. В открывшемся окне напишем:

|  |
| --- |
| Function Деление(divisible As Variant, divider As Variant) |

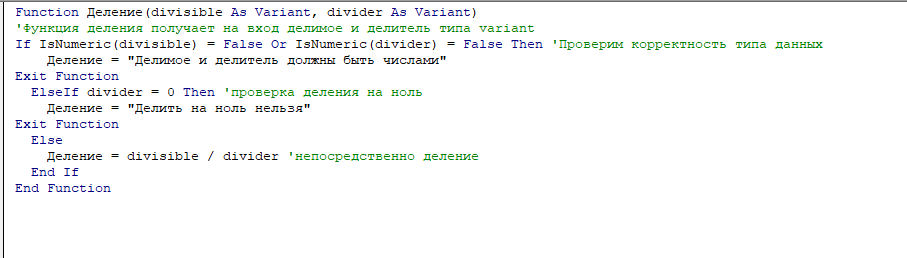
После чего программа автоматически добавит строку End Function.

В скобках мы указали что на вход данной функции получает делимое(divisible) и делитель (divider). Тип данных назначим Variant для максимальной универсальности функции.

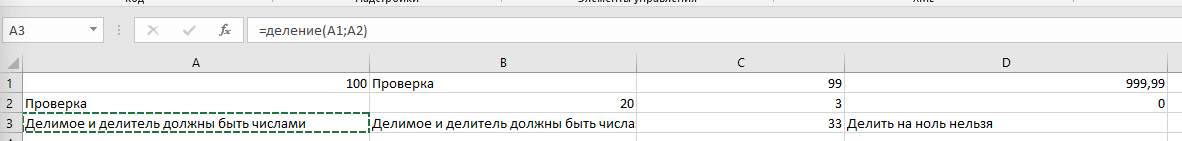
1. Рассмотрим все возможные варианты входных данных и последствия их поступления. Мы знаем, что делить можно только числа, что делить на ноль нельзя. Опишем данные ограничения через операторы ветвлений:

|  |
| --- |
| If IsNumeric(divisible) = False Or IsNumeric(divider) = False Then 'Проверим корректность типа данных  Деление = "Делимое и делитель должны быть числами"  Exit Function  ElseIf divider = 0 Then 'проверка деления на ноль  Деление = "Делить на ноль нельзя"  Exit Function  Else  Деление = divisible / divider 'непосредственно деление  End If |

1. Сохраним получившийся проект.



1. Перейдем к таблице и проверим работу функции. В ячейках A1:D1 пишем делимое (число и строковое значение), в A2:D2 - делители (ненулевое число, нуль и строковое значение). И в ячейках A3:D3 запишем формулу с пользовательской функцией деление(). Получаем результат:



Таким образом мы рассмотрели возможность создания функций в Excel.

### Выводы

На этом уроке мы познакомились c базовыми возможностями VBA в MS Excel.

Исходя из полученных знаний и опыта можно сделать следующие выводы:

* Макросы позволяют оптимизировать процесс решения рутинных задач.
* VBA может существенно расширить возможности Excel.
* VBA позволяет создавать пользовательские функции которых недостает в штатном наборе Excel.

Описанное в данной лекции является лишь малой частью возможностей VBA, который дополнительно позволяет создавать и программировать формы (специальные структуры для корректного ввода данных для их последующей обработки) и реализует в себе принципы объектно-ориентированного программирования.

# Итоги курса «Электронные таблицы в аналитике»

Во время изучения курса электронные таблицы в анализе данных мы научились:

* Создавать и форматировать таблицы
* Использовать формулы для расчетов
* Применять инструменты визуализации данных
* Использовать инструменты для анализа данных, такие как фильтрация, поиск, сводные таблицы и другие
* Методам работы с надстройками и научились их устанавливать
* Применять, записывать и разрабатывать макросы на языке VBA.

Спасибо за внимание к данному курсу!

До новых встреч!

# Список литературы

1. **Клименко Андрей, Комолова Нина Владимировна.** Программирование на VBA в Excel 2019, Санкт-Петербург : БХВ-Петербург, 2020. стр. 496. 978-5-9775-6593-6.

2. **Alexander M., Kusleika R.** Excel 2019 Power Programming with VBA*.* Wiley, 2019. стр. 756, 978-1-119-51492-3.

3. **Справочник по VBA для Office [В Интернете] https://docs.microsoft.com/ru-ru/office/vba**