SF EDUCATION

ПРОГРАММИРОВАНИЕ НА VBA

Циклы Do Loop

Содержание:

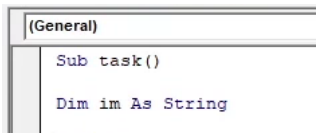
1. Элемент **While**
2. Структура **Until**
3. Несколько сабов в одном макросе
4. Ещё один пример – чем отличаются **While от Until**
5. Проблема бесконечного цикла
6. Выход из цикла

Переходим к следующему циклу, который называется **Do Loop** и использует некоторые элементы, а именно**: While и Until**.

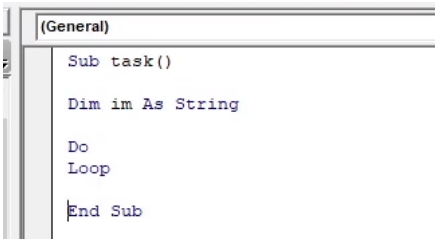
Элемент While

Чтобы написать этот цикл, нужно сделать следующее.

Создаём какую-то переменную, которая будет являться строковой:



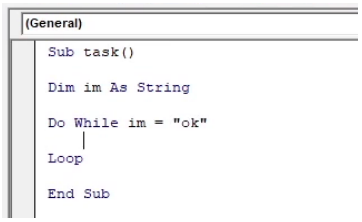
Задаём наш цикл вот таким образом:



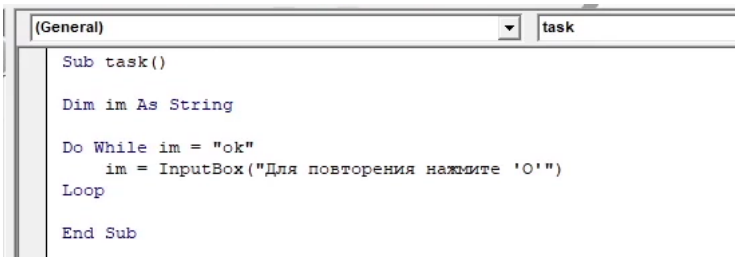
И для того, чтобы этот цикл работал, нужно использовать либо **While**, либо **Until**.

И **While, и Until** мы можем использовать либо в **Do**, либо в **Loop**.

Здесь пишем:

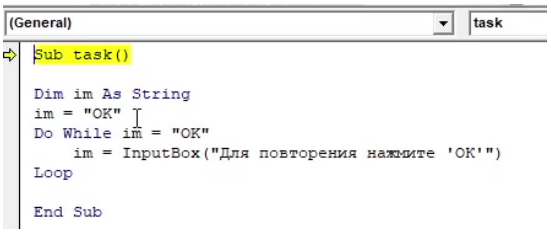


А дальше используем интересную вещь – **InputBox**. Наша переменная будет **InputBox**, где будет записан текст «**Для повторения нажмите Ok**».



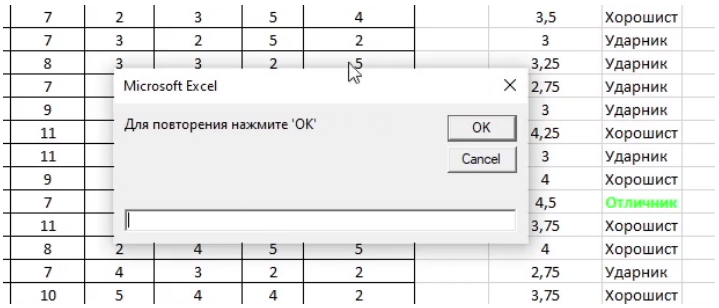
Пробуем запустить и пока ничего не происходит, поскольку **элемент im**, который мы запустили – пустой. Поэтому наш **Do While** не запускается.

Поступаем очень просто: изначально изменяем **переменную im** на значение **Ok**:



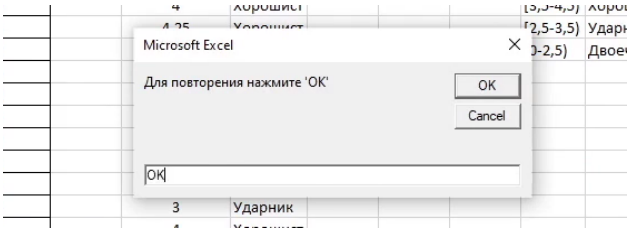
Теперь запускаем и видим, что мы зашли в наш цикл. Что это означает?

Во-первых, это окошечко **InputBox**:



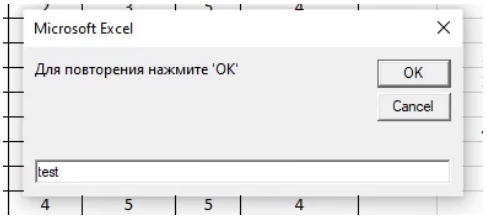
Если мы здесь будем записывать какие-либо значения, кроме **Ok**, наш цикл не продолжится.

Если в окошке записать **OK**:

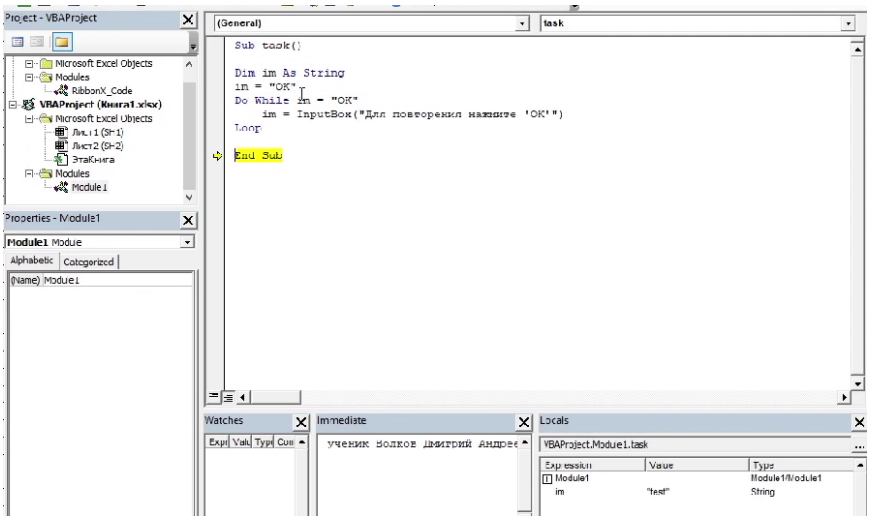


То мы войдём **в Loop** и ещё раз потом вернёмся в **Do**, поскольку **im** у нас до сих пор **OK**, то заходим ещё раз в наш **InputBox**.

Теперь, если мы здесь запишем **test**:



Запустим наш макрос и получим значение **im test.**



И соответственно, выходит из нашего цикла.

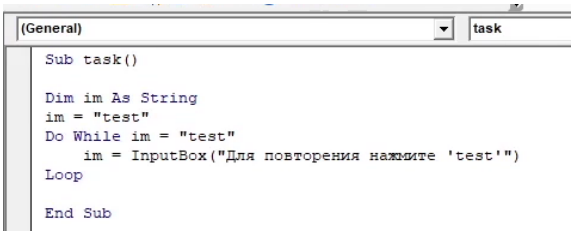
Do While выполняется до тех пор, пока im = OK. Это одно из четырёх возможных значений, что можно использовать:

- ✔ **Do While;**
- ✔ **Do Until;**
- ✔ **Loop While;**
- ✔ **Loop Until.**

Do While означает – выполняется до тех пор, пока im = OK.

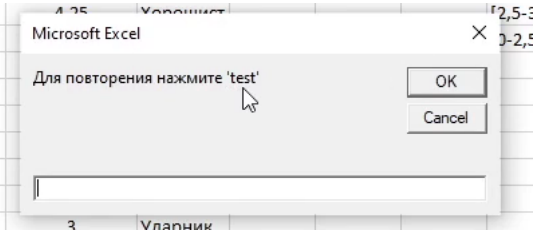
Здесь может быть «**больше или равно**», «**меньше или равно**», но главное **Do While** выполняется до тех пор, пока **im = OK.**

Можно написать совершенно **рандомные слова**:

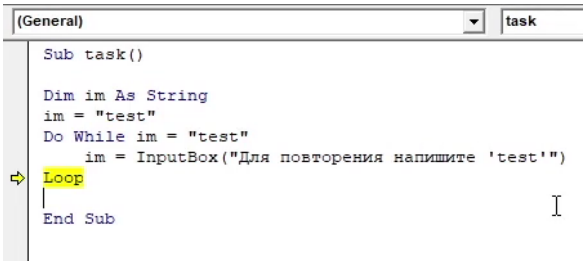

(General)
task
Sub task()
Dim im As String
im = "test"
Do While im = "test"
im = InputBox("Для повторения нажмите 'test'")
Loop
End Sub

Теперь запускаем наш макрос.

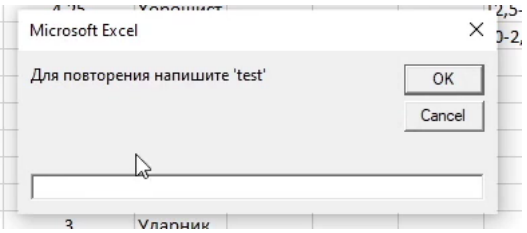
Таким образом, для повторения нужно нажать «**test**»:



Нам надо не «**нажать**», а «**написать**». Меняем:



Соответственно:



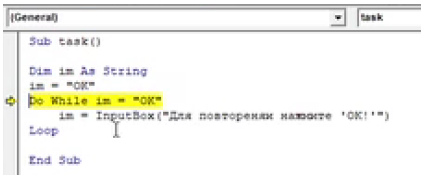
То есть, **если в ячейке ввода мы пишем какое-то другое слово** или используем заглавные буквы, то **мы выходим из цикла**. Очень придирчива к регистрам, к дополнительным элементам (пробелам и т. д.). Значение, которое мы вводим, **должно совпадать посимвольно с тем, что записано**.

Дальше мы пройдём **Do Until** и поймём, в чём разница между ними. После этого перейдём **к Loop While** и **Loop Until**.

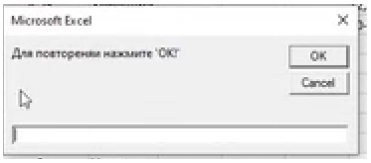
Сейчас мы познакомились с **Do While Loop**. Это в некотором роде **петля**, которая **выполняется до тех пор, пока условие в Do будет выполнимо**.

Приведём пример, когда можно поставить условие в **Loop.**

Ещё раз прогоним наш макрос:

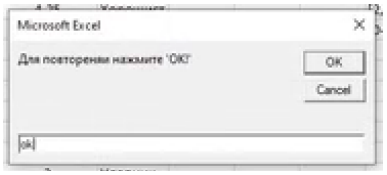


И посмотрим, что будет происходить:



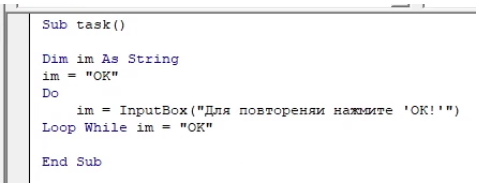
Для повторения нажать **ОК**. Пишем в строке **ОК**. Очень **важно вводить в правильном регистре.**

У нас **im = ОК**, значит мы будем опять входить внутрь нашего цикла и как только введём что-то другое, не удовлетворяющее регистру или написанию текста:



Тогда мы выйдем из этого макроса.

Если мы переместим **While:**



Что у нас изменится кардинально? Пройдём по тексту: мы **im** присваиваем **OK**, делаем считывание **im из InputBox**. Потом мы заходим в петлю **While** и, если пока **im будет равно OK**, мы будем это делать.

Проверим:

- ✔ Мы считываем значение и пишем **OK**;
- ✔ Зашли в **Loop**;
- ✔ Из **Loop** благодаря петле зашли в **Do**;
- ✔ И ещё раз считали. То есть, если мы здесь впишем другое значение, например, **test,** тогда **im** у нас становится **test**, в **Loop im не равно OK**, поэтому **заканчиваем макрос**.

Структура Until

Есть ещё одна структура, аналогичная **While**, а точнее её **полная противоположность – это Until.**

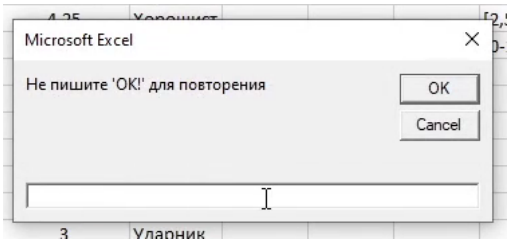
Поскольку это полная противоположность **While**, то нужно немного видоизменить текст:

```
Sub task()

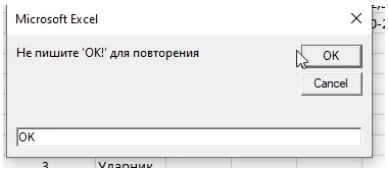
Dim im As String
im = "OK"
Do Until im = "OK"
    im = InputBox("Не пишите 'OK!' для повторения")
Loop

End Sub
```

Смотрим пошагово, что произойдёт. Мы зайдём в **OK**, зайдём в **Until,** далее у нас **im = OK**, поэтому заканчиваем. **Если im не OK**, а какая-то другая пустая строка, то написано:


Microsoft Excel
Не пишите 'OK!' для повторения
OK
Cancel

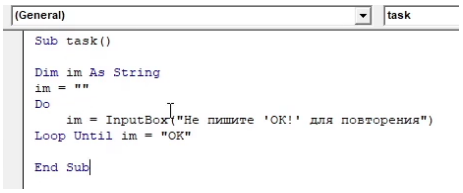
Напишем любое другое слово. И **Until у нас im не OK**, поэтому, если написать:



Мы получаем, что **im = OK**, поэтому мы выходим из цикла.

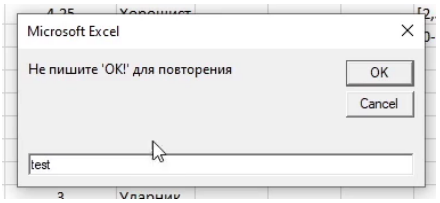
То есть, здесь мы должны написать **OK** для завершения цикла либо не написать **OK** для повторения цикла.

Аналогично, мы можем переместить условие:

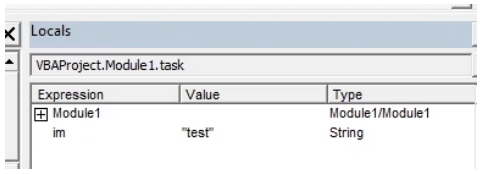


Мы будем выполнять считывание информации до тех пор, пока обратное действие **im = OK.**

Опять запускаем. Видим «**Не пишите OK для повторения»** цикла, значит напишем здесь, к примеру, **test**:

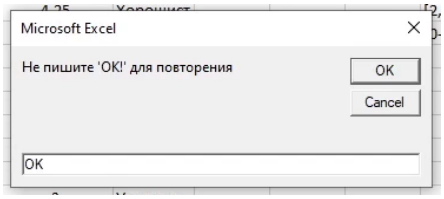


Получили, что **im равно test**:


Locals
VBAProject.Module1.task
Expression
Value
Type
Module1
Module1/Module1
im
"test"
String

Но поскольку, это не **OK**, мы ещё раз выполняем этот цикл.

Пишем теперь **OK**:



И выходим из цикла.

По сути, в структуру **Do Loop** в виде петли мы можем вписать либо в начало петли, либо в конец петли условие – **While или Until**.

While и Until – это полные противоположности:

- **While** говорит: **выполняй** пока **im = OK**;
- **Until** говорит: **не выполняй** пока **im = OK**.

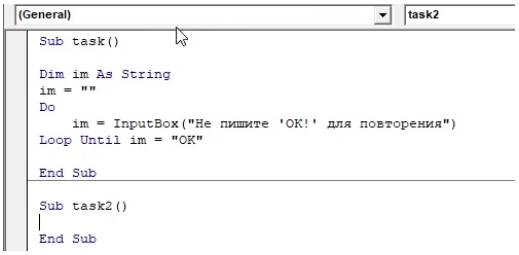
И какие-то действия, связанные либо с выполнением, либо с невыполнением данных структур мы дальше выполняем.

Несколько сабов в одном макросе

Мы можем в одном макросе **писать несколько сабов**. **До этого** мы рассматривал, **что один макрос – это один саб**. И если что-то нужно, то меняли внутри саба.

Мы можем написать рядом совершенно другой саб. Главное, чтобы **его идентификационный номер был уникальным**.

SF EDUCATION



Копируем всё и размещаем во втором сабе. И **делаем их различными**:

```
(General)                                   task2
Sub task()

Dim im As String
im = ""
Do
    im = InputBox("Не пишите 'OK!' для повторения")
Loop Until im = "OK"

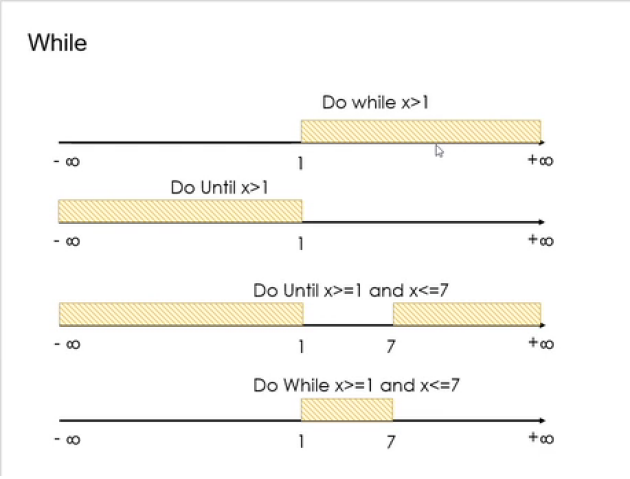
End Sub

Sub task2()
Dim im As String
im = ""
Do Until im = "OK"
    im = InputBox("Не пишите 'OK!' для повторения")
Loop
End Sub
```

Сабы разделены тоненькой горизонтальной чёрной линией. Соответственно, мы можем в одном макросе писать несколько сабов и выполнять их по очереди либо так, как захочется. **Они будут параллельны** и друг друга не касаться, если мы этого не просим внутри саба.

Ещё один пример – чем отличаются While от Until

Переходим к следующему этапу:



Первая прямая. Написано: выполняем, **если x>1**. Если мы находимся в элементе от **1 до +** ∞, мы будем выполнять структурные данные, либо циклы, либо параметры, которые необходимо выполнить, пока у нас **x>1.**

Вторая прямая. Если мы пишем: **Do Until**, то соответственно, мы пишем точно такое же условие **x>1**, только оно противоположно. То есть мы выполняем, пока x не больше 1. Если мы находимся от **-∞ до 1**, то выполняем всё, что нам требуется. Какие-то операции сложения, умножения и т. д.
Мы можем написать **Do While x<1 или Do Until x>1**.

Третья прямая. Также мы можем прописывать два и более условий. Если нам нужно одновременное выполнение условий, то можно использовать оператор and, то есть выполнение одного условия и выполнение другого условия.

Четвёртая прямая. Тут сказано, что мы сначала выполняем цикл **Do Loop While** пока **x >=1 и x <= 7**. Это такой диапазон данных от 1 до 7 на числовой прямой. Если x будет находиться в этом диапазоне, например, 5,5, 6, 13,4:3, мы будем находиться в этом диапазоне.
Если x будет находиться вне диапазона, мы не будем выполнять больше наш цикл **Do Loop**. Это полная **противоположность Do Until.** Если **x >= 1 и x <= 7**, то мы не выполняем, а в противоположном случае – выполняем.

Стоит заметить один тонкий момент, **касающийся «ровно» в этой схеме**.

Берём первую прямую. Мы пишем пока **x>1** и если **x=1**, то мы не выполняем, потому что 1 не больше 1. Но на второй прямой условие будет выполнено, потому что x не больше 1.

На третьей прямой сказано, что **x >= 1 и x <= 7** и используется структура **Until.** Значит, если **x=1**, то у нас структура не выполнится, потому, что x будет больше либо равен 1 и поэтому не выполнится условие, что 1 не больше либо равно 1.

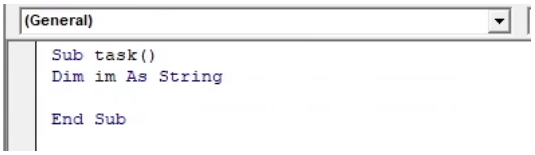
На четвёртой **прямой While и если x >=1 и x <= 7,** то условие выполнится, потому что 1 больше или равно 1.

Ещё раз напоминаю. **While** мы выполняем пока есть определённые условия, а **Until** мы выполняем в противовес этим условиям.

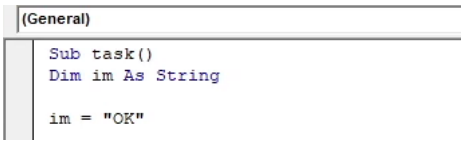
Проблема бесконечного цикла

Напишем наш старый макрос, к которому пришли.

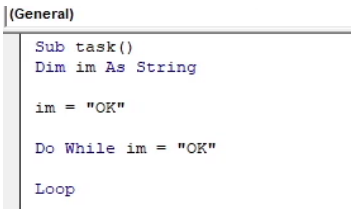
Создадим переменную **im**, которая является стринговой:



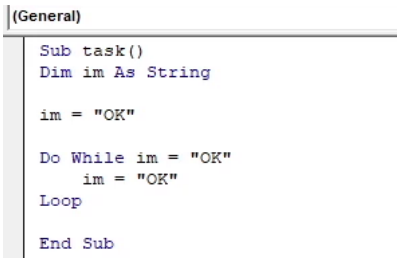
Придадим этой переменной значение **OK** изначально:



И теперь, соответственно:



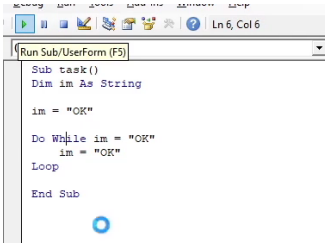
Теперь запросим данные:



Прогоним наш цикл. В таком виде наш цикл становится бесконечным. Если нажать **F8**, то становится видно, что этот цикл стал бесконечным и никогда не перейдёт в **End Sub**.

Мы очень часто не можем остановить выполнение этого цикла. Это очень коварный момент.

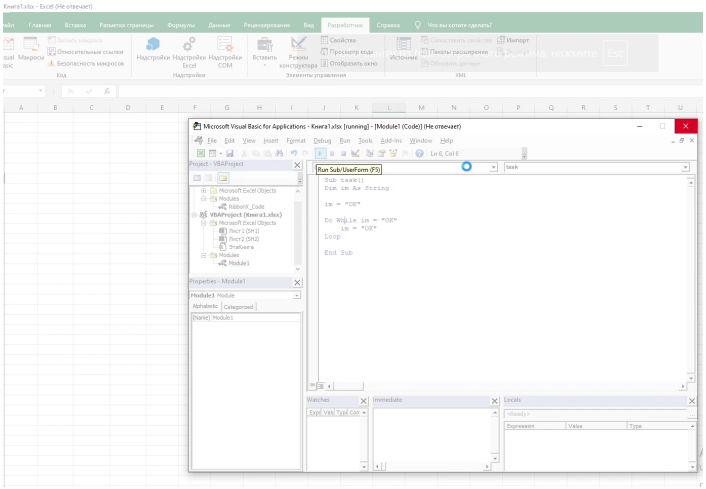
Устроим демонстрацию. Останавливаем нынешний цикл и нажимаем кнопку:



И видим вместо курсора **кружок**. Это означает, что **идёт выполнение цикла**. Но мы прекрасно понимаем, что **цикл никогда не закончится**, потому, что пока **im равно OK, то выполняем im равно OK**. То есть, мы делаем следующее действие пока **im равно OK** и в действиях мы говорим, что **im равно OK**.

Это ловушка и мы никогда не закончим этот цикл. Выйти из этой ситуации будет небезопасно и это серьёзная проблема.
Если у вас **клавиатура старого образца**, то на ней есть кнопка **Break**. Нужно на неё нажать либо нажать **Ctrl+ Break**. Тогда выполнение цикла и всех текущих процессов в Excel **завершится**.

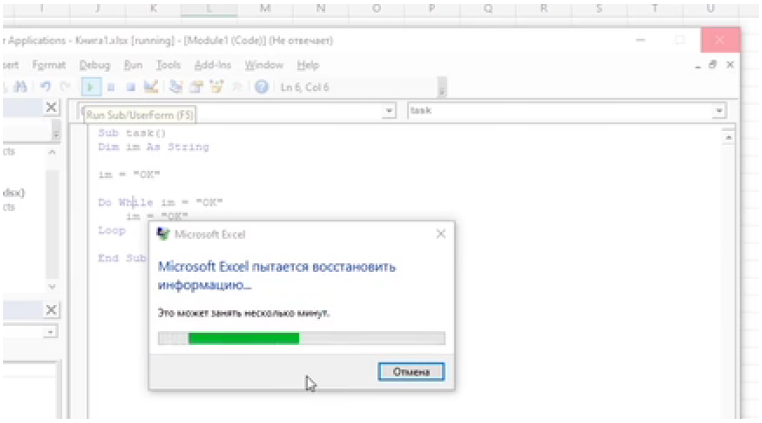
Но если кнопки **Break** на клавиатуре нет, то у вас большие проблемы и вы не сможете просто так выйти из цикла. Единственное спасение – нажать **esc**. Тогда рабочий экран становится почти белым, программы «**Не отвечают**», и мы ничего делать на компьютере не можем.



Независимо от нас происходит выполнение циклов. Единственный способ всё это завершить – **закрыть Excel**, по-другому выйти из этой ситуации нельзя.

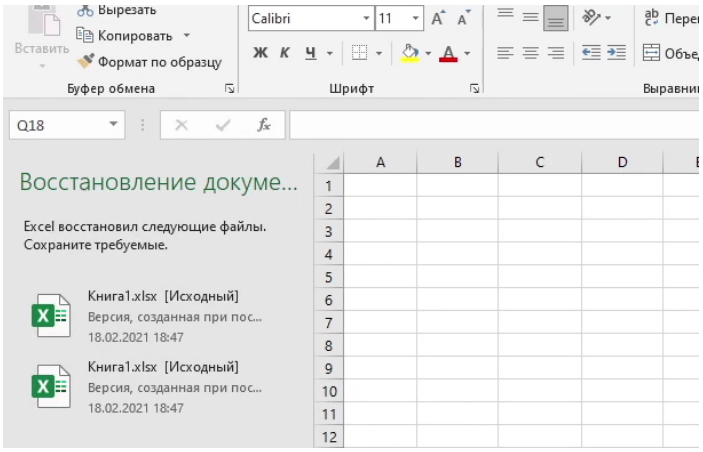
Иногда некоторым везёт, и выскакивает окошко с текстом, что вы попали в ловушку и можете завершить этот цикл. Но это чистое везение и сейчас – не повезло.

Соответственно, **единственный способ – это закрыть программу и ждать окончания**:

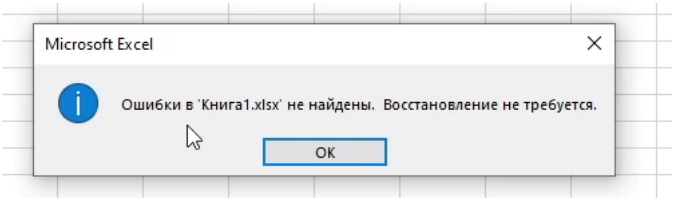


Либо через «**Диспетчер задач**» закрыть эту программу.

Глобальная проблема заключается в том, что при открытии этой же Книги появляются исходные версии, которые невозможно восстановить.

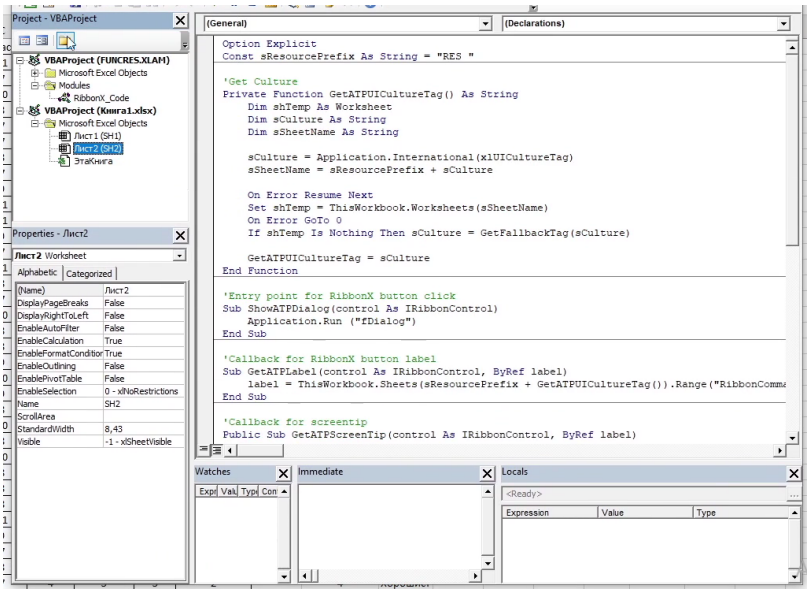


Если мы нажимаем «**Показать восстановленные элементы**», то видим:



И как будто, в Книге не было проблем. А макрос и Книга – это две разные вещи. В самой Книге проблем нет, а в макросе – есть. Но **макрос мы не сохранили**. Поэтому, здесь написано, что восстановление не потребуется, поскольку в Книге нет проблем.

Поэтому книги можем закрыть. Зайти в **Visual Basic**, в наше дерево:



И увидеть, что **макросы не сохранились**. Это очень плохо.

Мы писали макрос, совершили ошибку и **получили бесконечное использование макроса – бесконечный цикл**. И макрос не сохранился.

Поэтому, прежде чем запускать макрос, прежде чем нажать на кнопку Play или F5, нужно пройти макрос пошагово через F8 либо сохранить Книгу с макросами, как говорилось ранее. И только после этого можно запускать.

Мы всего этого здесь делать не будем по причине того, что нам макрос можно написать достаточно быстро, в течение 1 минуты.

Но если макрос пишется в течение нескольких дней, и вы не сохраняете, что написали, то это может обернуться большой проблемой. Поэтому:

- **Сохраняем макрос – запускаем макрос.**

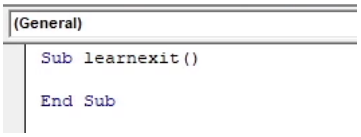
Либо выполняем его пошагово. Есть ещё **кнопка Pause, кнопка Reset** и всегда можно сбросить выполнение макроса в пошаговом режиме.

Выход из цикла

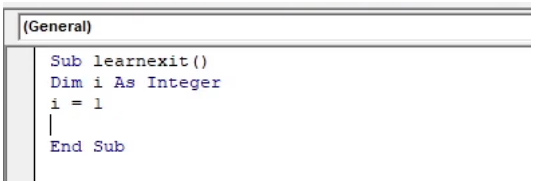
В обычном режиме (**Run Sub**) вы не сможете его остановить, если не повезёт. Это некий дисклеймер, но мы можем себя предостеречь.

Это можно сделать, прописав в каждом макросе выполнение определённой структуры. Посмотрим пример и будем учиться выходить из цикла.

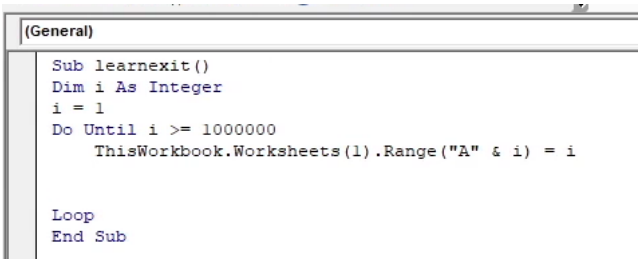
Создаём новый саб.



Создаём **переменную i** и присваиваем ей значение 1.

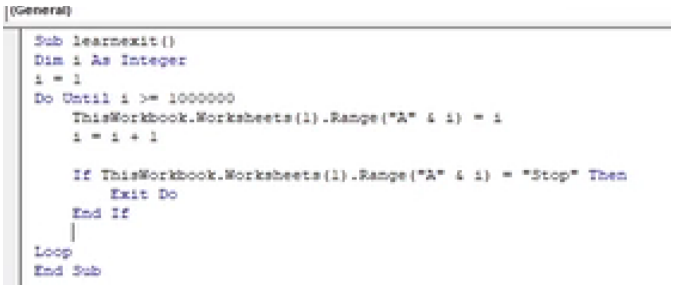


Дальше мы будем выполнять **Do Loop**. И пропишем, что пока **i** меньше миллиона:

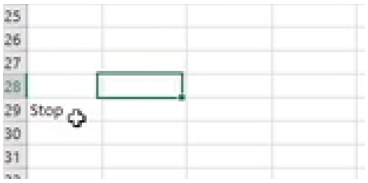


В ячейку A1 мы впишем значение 1, в ячейку A2 мы впишем значение 2, в ячейку A3 мы впишем значение 3. Но **нам нужно и увеличивать**, поскольку это не **For**, где мы прописываем **Next i, а While Until**, где нет автоматического увеличения i.

Нам нужно это **i** увеличить на единицу. Пишем: если мы в этой Книге, в ячейке встретим какое-то значение, мы встретим слово, к примеру, **Stop,** тогда мы закончим выполнение нашего макроса. Соответственно, закрываем структуру:



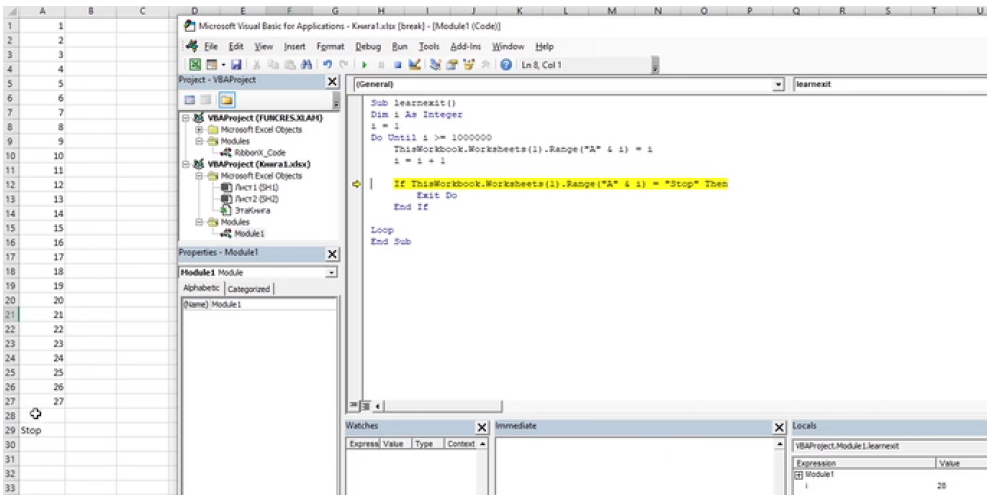
Stop мы запишем, например, на этой строке:



Запускаем наш макрос F5 и оно быстро прокрутилось.

Теперь пошагово выполним:

- Сначала i на единице, в A1 записываем 1 и увеличиваем на 1;
- A2 не равно 100, поэтому мы не заходим в if;
- Возвращаемся ещё раз к **Do Until**, меньше 1 миллиона;
- A2 равно 2, присваиваем и увеличиваем на 1;
- И так делаем пока не доберёмся до 27 строки.

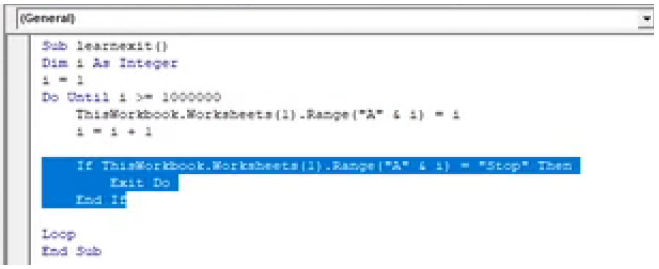


Сейчас **i** у нас 28. На 28 – не **Stop**, поэтому мы заходим ещё раз в **Until**, 28 меньше миллиона, выполняем. Соответственно A28 становится 28 и становится 29.

Но A29 равно 100 и мы заходим в наш цикл и выходим из цикла Do и заканчиваем.

Точно также, если мы, к примеру, изменим наш **Stop** до 50 и опять прогоним наш макрос, то значение увеличится до 50 строки. После уже не пойдут.

Если мы уберём вот эту величину:



И будем увеличивать всё время **i**. То придём опять к бесконечному циклу.



Прогоним наш макрос и посмотрим, что происходит. Мы понимаем, что i никогда не будет больше равно 0 и цикл будет идти и идти.

Будет перегрузка данных. То есть была **ошибка в наших значениях**. И нам предлагается л**ибо завершить цикл, либо исправить значение на последнем шаге**, на котором мы находимся.

Сейчас мы находимся на шаге 32767. Сейчас нам повезло, но нам может не повезти, и мы опять потеряем все внутренности нашего макроса.