Технология программирования

Блок 2. Основные понятия и операторы языка VB

Тема 3. Организация циклических структур, их виды и синтаксис

Цели изучения темы:

- ознакомить студентов с основами циклических структур.

Задачи изучения темы:

- сформировать знания о функционировании циклических структур VBA.

В результате изучения данной темы Вы будете
знать:

- виды циклов VBA;
- основные правила синтаксиса кода в VB;
- базовые операторы циклов и особенности их употребления;

уметь:

- использовать соответствующие задачам виды циклов;
- определять механизмы работы различных операторов и их виды;

владеть:

- основными правилами синтаксиса при использовании циклических структур;
- базовыми операторами циклов VB.

Учебные вопросы темы:

1. Циклы и их виды.
2. Организация циклических структур и их синтаксис.
3. Вложенные и бесконечные циклы.

Основные термины и понятия, которые Вам предстоит изучить:

цикл, интеракция цикла, счетчик, шаг, условие, тело цикла, циклическая структура, вложенные циклы, внутренние циклы, внешние циклы, бесконечные циклы.

Вопрос 1. Циклы и их виды

Решение многих практических задач сводится к выполнению вычислений по одним и тем же зависимостям, но при разных значениях входящих в них величин. Мы тратим много времени на повторяющиеся простые задачи. Например, такие как форматирование нескольких диапазонов, отображение нескольких листов, копирование и вставка в несколько рабочих книг, применение фильтров к нескольким таблицам или сводным таблицам, замена значений, обновление формул и т.д. Можете ли вы вспомнить несколько задач, в которых вам приходилось повторять один и тот же процесс снова и снова? Эти задачи чрезвычайно трудоемкие и скучные!

Вычислительный процесс, повторяющийся много раз, называется *циклическим*, а многократно повторяющиеся участки этого процесса называются циклами.

Процесс выполнения все операторов, заключенных в структуру цикла, один раз называется *итерацией цикла*. Мы можем использовать циклы в наших макросах VBA, чтобы очень быстро повторять действия. Задачи, выполнение которых вручную может занять несколько часов, могут быть выполнены за несколько секунд с помощью цикла.

Различают регулярные циклы с управляющим параметром (с известным числом повторений), условием окончания которого является достижение параметром цикла своего конечного значения; циклы итерационные, в которых условие повторения или окончания цикла задается по некоторому результату, например, пока не будет достигнута точность вычислений.

Блок операторов, находящийся между началом и концом цикла, называется *"тело цикла"*.

Реализуются циклы с помощью специальных операторов цикла.

Виды циклов

Название цикла	Вид
`For - Next`	С фиксированным количеством повторов. Выполняется заданное количество раз.
`While - Wend`	С предусловием. Если не верно условие, заданное на входе в цикл, может не выполниться ни разу.
`Do - Loop`	Группа циклов, которые могут быть и с предусловием, и с постусловием.

Вопрос 2. Организация циклических структур

Самой простой структурой цикла является ***фиксированный цикл***.

Оператор цикла ***For...Next***.

Часто при составлении макроса заранее известно количество повторений группы операторов, в таких случаях можно использовать инструкцию **For...Next**. Оператор **For...Next** используется для выполнения наборов операторов указанное число раз. Циклы **For** используют в качестве счетчика переменную, значение которой увеличивается или уменьшается при каждом выполнении цикла на указанное значение.

Синтаксис инструкции **For...Next**

For ***счетчик*** **=** ***начало*** **To** ***конец*** [step *шаг*]

[*операторы*]

Next [*счетчик*]

счетчик — обязательный элемент. Это должна быть числовая переменная. Она не может иметь тип Boolean или быть элементом массива.
начало — обязательный элемент, содержит начальное значение переменной *счетчик.*
конец — обязательный элемент, содержит конечное значение переменной *счетчик.*
шаг — элемент необязательный, это значение, на которое изменяется счетчик при каждом выполнении тела цикла. Если это значение не задано, по умолчанию *шаг* равен единице. *Шаг* может быть как положительным, так и отрицательным.
операторы — необязательный элемент. Один или несколько операторов между **For** и **Next**, которые выполняются указанное число раз.

Ключевое слово **Next** сообщает VBA о том, что достигнут конец цикла. Необязательная переменная *счетчик* после ключевого слова **Next** должна быть той же самой переменной *счетчик*, которая была задана после ключевого слова **For** в начале структуры цикла.

Инструкция **For...Next** работает следующим образом: начальное значение элемента *счетчик* сравнивается с конечным значением. Если *шаг* положителен и начальное значение меньше конечного или если *шаг* отрицателен и начальное значение больше конечного, то управление передается внутрь тела цикла. После выполнения всех

операторов в теле цикла значение *шаг* добавляется к текущему значению переменной *счетчик*. После этого операторы тела цикла либо выполняются еще раз (на основе того же условия, которое привело к начальному выполнению цикла), либо цикл завершается и выполнение продолжается с оператора, следующей за **Next**.

Пример: считать сумму натуральных чисел до 10

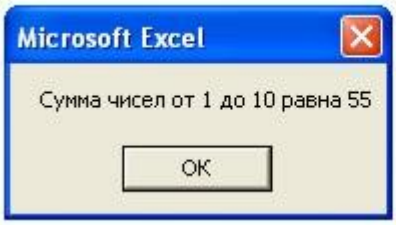
```
Dim i As Integer
Dim S As Integer
S = 0
For i = 1 To 10
  S = S + i
Next i
```

```
Dim i As Integer, SUM As Integer

SUM = 0

For i = 1 To 10
    SUM = SUM + i
Next i

MsgBox "Сумма чисел от 1 до 10 равна " & SUM
```



А теперь два варианта цикла **For...Next** с использованием шага цикла отличного от единицы:

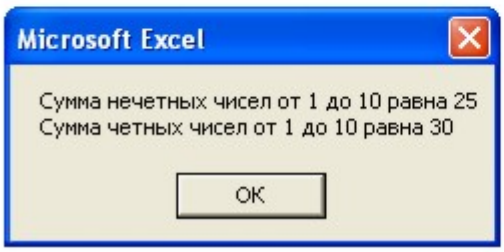
```
Dim i As Integer, SUM As Integer, SUM2 As Integer

SUM = 0: SUM2 = 0

For i = 1 To 10 Step 2
    SUM = SUM + i
Next i

For i = 10 To 1 Step -2
    SUM2 = SUM2 + i
Next i

MsgBox "Сумма нечетных чисел от 1 до 10 равна " & SUM & vbCr _
& "Сумма четных чисел от 1 до 10 равна " & SUM2
```



Обратите внимание! При уменьшении счетчика цикла **For...Next** цикл выполняется, пока переменная счетчика больше или равна конечному значению, а когда счетчик цикла увеличивается, цикл

выполняется, пока переменная счетчика меньше или равна конечному значению.

Допускается вложение циклов **For...Next** (один цикл **For...Next** располагается внутри другого). Счетчик каждого цикла должен иметь уникальное имя.

Как остановить цикл раньше?

Обычно цикл перебирает все элементы в коллекции, а затем переходит к следующей строке кода ниже следующей строки. Однако мы можем остановить цикл раньше с помощью оператора **Exit For**.

*Оператор цикла **While... Wend***

Этот цикл в VBA, чтобы сделать его совместимым со старым кодом Microsoft рекомендует использовать циклы *Do*, поскольку они более структурированы. Из MSDN: «Оператор Do… Loop обеспечивает более структурированный и гибкий способ выполнения циклов».

Цикл **While** – цикл с предусловием. Оператор имеет синтаксис

While ***условие***

[*операторы*]

Wend

Синтаксис инструкции **While...Wend** содержит следующие элементы:

условие — обязательный элемент. Числовое выражение или строковое выражение, которое имеет значение True или False. *операторы* — необязательный элемент. Один или несколько операторов, выполняемых, пока условие имеет значение True.

Этот оператор называют оператором цикла с предусловием. Выполняется оператор **While...Wend** следующим образом. Если *условие* имеет значение True, выполняются все операторы до инструкции **Wend**. Затем управление возвращается инструкции **While** и вновь проверяется *условие*. Если *условие* по-прежнему имеет значение True, процесс повторяется. Если оно не имеет значение True, выполнение возобновляется с инструкции, следующей за инструкцией **Wend**. В связи с этим элемент *условие* здесь является условием выполнения цикла. Циклы **While...Wend** могут иметь любую глубину вложенности.

Пример: выводить на экран случайные числа от 0 до 20 до тех пор, пока не будет выведено число больше 10.

```
Dim A As Integer
```

```
A = 1
While A < 10
   A = Int(Rnd() * 20)
   MsgBox A
Wend
```

*Оператор цикла **Do...Loop***

Оператор **Do...Loop** также используется для выполнения наборов операторов неопределенное число раз. Цикл **Do** можно использовать четырьмя способами, и это часто вызывает путаницу. Однако в каждом из этих четырех способов есть только небольшая разница. **Do** всегда в начале первой строки, а **Loop** всегда в конце последней строки.

Do

Loop

Мы можем добавить условие после любой строки:

Do [условие]

Loop

Do

Loop [условие]

Условию предшествует **While** или **Until**, которое дает нам эти четыре возможности:

Do While [условие]

Loop

Do Until [условие]

Loop

Do

Loop While [условие]

Do

Loop Until [условие]

При использовании **Do Loop** условию должно предшествовать **Until** или **While**. **Until** и **While**, по сути, противоположны друг другу. Они используются в VBA аналогично тому, как они используются в английском языке.

Например:

- Сушите одежду, пока не пойдет дождь
- Сушите одежду, пока не идет дождь

Другой пример:

- Оставайся в постели, пока не станет светло
- Оставайся в постели, пока темно

Еще один пример:

- повторять, пока число не станет больше или равно десяти
- повторить пока счет меньше десяти

Как видите, использование **Until** и **While** — это просто противоположный способ написания одного и того же условия.

Рассмотрим синтаксис всех версий оператора **Do Loop**:

Цикл Do ... While

Do While [***условие***]

[***операторы***]

Loop

Синтаксис инструкции **Do Loop** содержит следующие элементы:

условие — необязательный элемент. Числовое или строковое выражение, которое имеет значение True или False; ***операторы*** — один или несколько операторов, выполнение которых повторяется, пока условие не приобретет значение True.

Loop - ключевое слово, указывает на окончание тела цикла и обозначает место, из которого VBA возвращается в начало цикла для проверки условия. VBA выполняет цикл пока логическое выражение, представленное с помощью ***условие***, равно True.

При выполнении цикла **Do While** сначала тестируется логическое выражение (***условие***); если оно равно True - выполняется тело цикла. При достижении ключевого слова **Loop** управление опять передается в начало цикла и снова проверяется логическое выражение. Так происходит до тех пор, пока логическое выражение не станет False. Когда логическое выражение становится False - управление передается оператору, следующему за ключевым словом **Loop**.

Обратите внимание! Если логическое выражение равно False при первом выполнении цикла **Do While**, то управление сразу передается оператору, следующему за **Loop**, а операторы, находящиеся в теле

цикла соответственно пропускаются. Другими словами говоря, цикл **Do While** позволяет ни разу не выполнять операторы внутри него.

Пример: элементарный цикл **Do While**, подсчитывающий сумму цифр от 1 до 10:

```
Dim i As Integer, SUM As Integer

i = 1: SUM = 0

Do While i <= 10
    SUM = SUM + i
    i = i + 1
Loop

MsgBox ("Сумма цифр от 1 до 10 = " & SUM)
```

Цикл Do ... Until

Еще один цикл, проверяющий *условие* до выполнения цикла.

Do Until [***условие***]

[***операторы***]

Loop

VBA выполняет цикл пока логическое выражение, представленное с помощью ***условие***, равно False.

В остальном цикл **Do Until** полностью аналогичен циклу **Do While**.

Пример, использующий цикл **Do Until** для подсчета цифр от 1 до 10, будет выглядеть так:

```
Dim i As Integer, SUM As Integer

i = 1: SUM = 0

Do Until i > 10
    SUM = SUM + i
    i = i + 1
Loop

MsgBox ("Сумма цифр от 1 до 10 = " & SUM)
```

Для тестирования условий после выполнения тела цикла надо поместить логическое выражение в конец блока операторов, составляющих тело цикла, после ключевого слова **Loop**, которое сообщает о конце цикла.

Цикл Do .. Loop While

Конструкция цикла, проверяющая свое условие после выполнения цикла. Цикл с постусловием.

Синтаксис:

Do

[*операторы*]

Loop While [*условие*]

VBA выполняет цикл пока логическое выражение, представленное с помощью ***условие***, равно True.

При выполнении цикла **Do Loop While** сначала выполняются операторы тела цикла, затем по достижении ключевого слова **Loop** тестируется логическое выражение (***условие***); если оно равно True - управление передается в начало тела цикла и цикл повторяется снова. Так происходит до тех пор, пока логическое выражение не станет False. Когда логическое выражение становится False - управление передается оператору, следующему за строкой **Loop While...**

Обратите внимание! Даже если при первом выполнении цикла **Do Loop While** логическое выражение равно False тело цикла всё равно будет выполнено. Другими словами говоря, независимо от значения логического выражения, представленного с помощью ***условие***, этот цикл всегда выполняется, по крайней мере, один раз.

Пример программы, использующий цикл **Do Loop While** для подсчета цифр от 1 до 10, будет выглядеть так:

```
Dim i As Integer, SUM As Integer
i = 1: SUM = 0
Do
    SUM = SUM + i
    i = i + 1
Loop While i <= 10
MsgBox ("Сумма цифр от 1 до 10 = " & SUM)
```

Цикл Do .. Loop Until

Еще один цикл, тестирующий условие детерминанта после выполнения цикла.

Синтаксис:

Do

[*операторы*]

Loop Until [*условие*]

VBA выполняет цикл пока логическое выражение, представленное с помощью ***условие***, равно False.

В остальном цикл **Do Loop Until** полностью аналогичен циклу **Do Loop While**.

Пример программы, использующий цикл **Do Loop Until** для подсчета цифр от 1 до 10, будет выглядеть так:

```
Dim i As Integer, SUM As Integer
i = 1: SUM = 0
Do
    SUM = SUM + i
    i = i + 1
Loop Until i > 10
MsgBox ("Сумма цифр от 1 до 10 = " & SUM)
```

Следует сказать, что представленные выше примеры даны чисто в познавательных целях, чтобы можно было понять саму суть организации неопределенных циклов с проверкой условий до и после выполнения тела цикла. Сама же гибкость неопределенных циклов здесь явно не просматривается.

Вопрос 3. Вложенные и бесконечные циклы

Вложенные циклы

Внутри одного цикла могут находиться один или несколько других циклов. В этом случае охватывающий цикл называется *внешним*, а вложенные в него циклы – *внутренними*.

Правила организации как внешнего, так и внутренних циклов аналогичны правилам организации простого цикла. Параметры внешнего и внутреннего циклов изменяются не одновременно, т.е. при одном значении параметра внешнего цикла параметр внутреннего последовательно принимает все возможные значения. При организации вложенных циклов необходимо следить за тем, чтобы область действия внутреннего цикла не выходила за область действия внешнего цикла.

Бесконечный цикл

Даже если вы никогда не писали код в своей жизни, я уверен, что вы слышали фразу «бесконечный цикл». Это цикл, в котором условие никогда не будет выполнено. Обычно это происходит, когда вы забыли обновить счетчик.

Следующий код показывает бесконечный цикл:

```
Dim cnt As Long
cnt = 1
'это бесконечный цикл
Do While cnt <> 5
Loop
```

В этом примере cnt установлен в 1, но он никогда не обновляется. Поэтому условие никогда не будет выполнено — cnt всегда будет меньше 5.

Использование **For** безопаснее для подсчета, поскольку оно автоматически обновляет счет в цикле.

Контрольные вопросы:

1. Что такое цикл и зачем он используется?
2. Какие бывают виды циклов?
3. Чем регулярный цикл отличается от итерационного?
4. Какие команды выполняют итерационный цикл?
5. Чем отличаются циклы с пред- и пост- условием?
6. Какой цикл в VBA считается устаревшим?
7. Какие есть правила и ограничения при использовании вложенных циклов?
8. Что такое бесконечный цикл?