УРОК


# Синтаксис и данные в языке VBA

***Содержание урока***

- Основы синтаксиса VBA
- Как использовать константы
- Как объявлять программные переменные
- Типы данных, предусмотренные для сохранения числовых, текстовых и других значений
- Ранняя и поздняя привязка
- Статические и динамические массивы
- Пользовательские и перечислимые типы данных
- Область видимости переменных


В языке VBA содержится мощный набор инструментов, предназначенных для управления данными (и это важно, поскольку каждая программа Excel обрабатывает данные того или иного типа). В этом уроке рассматривается ряд вопросов, связанных с обработкой данных различных типов в приложениях Excel, в частности, будут описаны способы сохранения данных в процессе выполнения программы. Однако вначале мы рассмотрим основы синтаксиса языка VBA.

## Основы синтаксиса VBA

Понятие *синтаксиса* в языках программирования имеет много общего с аналогичным понятием обычных языков (русского, английского и т.п.). Так называется совокупность правил, определяющая порядок использования

и организации элементов данного языка. Например, одним из правил синтаксиса русского языка является следующее: предложение должно заканчиваться точкой. Поэтому прежде, чем приступить к написанию собственного кода VBA, необходимо ознакомиться с синтаксисом этого языка программирования.

Коды программы VBA состоят из *выражений* (*statements*). В каждой строке кода, как правило, располагается одно отдельное выражение. Но данное правило имеет исключения:

- несколько выражений могут быть расположены в одной строке и разделены двоеточием;
- использование символа продолжения строки (пробел, за которым следует символ подчеркивания) позволяет создавать выражение, занимающее несколько строк.

Не рекомендуется использовать первый прием для расположения нескольких выражений в одной строке, поскольку реальной пользы это не принесет, но сами коды могут при этом стать более сложными для чтения и восприятия.

Вторая же возможность, которая состоит в использовании символа продолжения строки, может быть действительно полезной. Некоторые строки кода оказываются слишком длинными и не помещаются целиком в пределах видимой области окна редактирования, что делает неудобным их чтение и редактирование. Чтобы разбить строку в определенной позиции, наберите пробел, затем символ подчеркивания ( _ ) и нажмите клавишу <Enter>. Не допускается использовать символ продолжения строки только внутри заключенных в кавычки строк. Рекомендуется продолжение каждой строки выделять отступом. Для VBA это не имеет никакого значения, но зато при чтении кодов вашей программы каждому будет понятно, что это продолжение предыдущей строки, а не начало новой.

## Комментарии

В кодах VBA *комментариями* (*comments*) называется текст, который игнорируется и никак не влияет на ход выполнения программы. Вы можете (и должны) использовать комментарии для добавления сопроводительных пояснений о том, как работает программа, значения каких аргументов требуется передавать процедурам. Советуем добавлять любые замечания, которые в будущем могут оказаться полезными вам или другим программистам.

Один из способов создания комментария состоит в использовании символа апострофа (единичные кавычки). Все, что будет набрано от апострофа и до конца текущей строки, считается комментарием. Символ апострофа может быть набран как вначале строки, так и внутри нее:

```
' Это комментарий
Dim MyWB As Workbook ' Это тоже комментарий
```

Для обозначения комментариев можно также использовать ключевое слово `REM` (образовано от слова “remark” — замечание). Обратите внимание, что слово `REM` обязательно должно быть указано в начале строки:

```
REM Это комментарий
Dim MyWB As Workbook REM А это не комментарий
```

Вы увидите, что в редакторе VBA комментарии отображаются другим цветом — по умолчанию зеленым.


Символ апострофа можно использовать с целью "перевода в разряд комментариев" отдельных фрагментов кода программы. Такая возможность оказывается полезной в тех случаях, когда требуется проверить, как будет работать программа без отдельных, уже набранных строк кода. Не удаляйте эти строки, а просто превратите их в комментарии. В дальнейшем, если вы захотите вновь вернуть строку "в строй", просто удалите набранный перед ней символ апострофа.

### Форматирование кодов

Так называемые *пробельные символы* (*white space*) в кодах программы (обычные пробелы, символы табуляции и пустые строки) игнорируются программой VBA, однако вы с успехом можете использовать их для визуального форматирования кодов, чтобы сделать их более удобными для чтения и восприятия. Добавляйте пустые строки, чтобы визуально отделить фрагменты кодов, функционально не зависящие друг от друга, а также добавляйте к выражениям большие или меньшие отступы, в зависимости от их отношения к другим выражениям. (Детально ознакомьтесь с материалом этого урока, а также урока 6, в котором изложена подробная информация о выражениях VBA, и смысл приведенных выше рекомендаций станет для вас более понятным.)

## Использование констант

*Константами* (*constant*) называются данные, которые не изменяются в ходе выполнения программы. В языке VBA различают константы двух типов: литеральные (literal) и символьные (symbolic).

*Литеральные константы* набираются непосредственно в кодах программы и могут представлять собой числа и строки (текстовые значения). Строковые константы должны заключаться в двойные кавычки. Например:

```
МояСтрока = "четыре"
МоеЧисло = 4
```

В данном случае значения "четыре" и 4 являются литеральными константами. Что касается числовых значений, VBA распознает их экспоненциальную запись и (с использованием префикса &H) шестнадцатеричную запись:

```
1.2E5 ' число 1,2, умноженное на 10 в пятой степени, т.е. 120000
&HFE      ' эквивалентно десятичному числу 254
```


Шестнадцатеричная система исчисления основана на использовании 16 цифр, тогда как привычная нам десятеричная система построена на применении десяти цифр. В шестнадцатеричной системе цифры от 0 до 9 обозначают первые десять цифр (как и в десятичной системе), а для обозначения последующих шести цифр используются первые шесть букв латинского алфавита (от A до F).

*Символьные константы* имеют собственные имена. Если в коде программы возникает необходимость в использовании значения такой константы, следует указать ее имя. Определяются символьные константы с помощью ключевого слова `Const`:

```
Const ИмяКонстанты = ЗначениеКонстанты
```

В качестве имени константы можно использовать любое допустимое в языке VBA имя (см. врезку "Правила выбора имен в языке VBA"). *ЗначениеКонстанты* — это литеральная, числовая либо строковая константа, определяющая значение создаваемой символьной константы. Например:

```
Const МоеИмя = "Питер"
Const МойВозраст = 30
```

Символьные константы обладают двумя преимуществами. Имя такой константы может (и должно) быть описательным, что значительно упрощает чтение кодов программы. Однако более важно следующее: если возникнет необходимость изменить значение константы в кодах программы, то достаточно будет сделать это только в одном месте — там, где она была объявлена.


Согласно общепринятому соглашению, имена констант набираются исключительно прописными буквами, тогда как имена других элементов, например, переменных, состоят из комбинаций прописных и строчных букв. Например, имя ПРОЦЕНТНАЯ_СТАВКА соответствует константе, а имя ПроцентнаяСтавка принадлежит переменной. Для программы VBA это правило не имеет никакого значения, однако, следуя ему, вы сможете довольно просто отличить в кодах программы имена констант от имен переменных.

**Правила выбора имен в языке VBA**

Эти правила в языке VBA распространяются на символьные константы, переменные, свойства и практически на любые другие элементы, которым в кодах программы могут присваиваться имена. Итак, в языке VBA имена должны соответствовать следующим требованиям:

- начинаться с букв;
- состоять не более чем из 255 символов;
- не совпадать с ключевыми словами VBA;
- не содержать в себе точек, пробелов, а также символов !, @, #, &, % и $.

Старайтесь давать переменным и прочим элементам описательные имена, по которым можно было бы определить их назначение в контексте программы. Поскольку использование пробелов не допускается, имена, состоящие из нескольких слов, можно образовать путем чередования строчных и прописных букв (например, `ПроцентнаяСтавка`) или вместо пробелов использовать символ подчеркивания (например, `Итог_2003`). Регистр символов не имеет значения, потому имена `Сумма`, `сумма` и `СУММА` считаются в языке VBA идентичными.

## Объявление и использование переменных

Как нетрудно догадаться, *переменные* содержат в себе значения, изменяемые в ходе выполнения программы. В языке VBA каждая переменная характеризуется своим *типом* (*type*), от которого зависит, какого рода данные в ней могут храниться. Создавая или объявляя переменную, вы определяете также ее тип. В простейшем случае синтаксис объявления переменной состоит из одного выражения, включающего в себя:

- ключевое слово `Dim`;
- имя переменной;
- ключевое слово `As`;
- название типа переменной.

Объявление переменной в кодах программы выглядит следующим образом:

```
Dim ИмяПеременной As тип
```

В одной строке можно объявить сразу несколько переменных:

```
Dim ИмяПеременной1 As тип1, ИмяПеременной2 As тип2, ...
```

Имена переменных должны быть составлены с учетом тех же правил, которые были рассмотрены ранее в этом уроке. Имя переменной должно быть уникальным в пределах ее области видимости. Подробнее об области видимости будет рассказано далее в этом уроке.

---

**Объявление переменных и выражение Option Explicit**

Недостатком языка VBA является то, что он позволяет программистам обходиться без объявления переменных. Такой порядок используется в языке VBA по умолчанию — вы просто добавляете новые переменные по мере необходимости, не заботясь об их объявлении или об использовании ключевого слова `Dim`. Если такой путь кажется вам наиболее простым и очевидным, что же в этом плохого?

Добавление переменных без их объявления приводит к возникновению ряда ошибок. Например, если вы случайно неправильно наберете имя переменной, программа просто воспримет это как добавление новой переменной с другим именем. В результате вы столкнетесь с некоторыми проблемами. Кроме того, если вы сами не объявите переменную, для нее автоматически будет выбран тип `Variant`, в результате доступные системные ресурсы будут использованы крайне нерационально.

Если же необходимость объявления переменных активизирована, VBA автоматически распознает неправильно набранные имена переменных как "необъявленные", что избавляет вас от многих проблем, о которых было сказано выше. Кроме того, поскольку вам придется явно объявлять каждую переменную, вы сможете подобрать такие типы данных, которые наилучшим образом будут подходить для сохранения обрабатываемых значений.

Чтобы активизировать необходимость объявления переменных, добавьте выражение Option Explicit в каждом модуле перед началом всех процедур. Вы можете активизировать также эту возможность (что я вам настоятельно рекомендую), выбрав команду Tools⇨Options (Сервис⇨Параметры). В открывшемся диалоговом окне на вкладке Editor (Редактор) установите флажок опции Require Variable Declaration (Объявление переменных обязательно).

---

*Осталось 20 минут*

## *Числовые переменные*

Числовые типы данных VBA могут быть разделены на две категории.

- **Целочисленные.** Числа, не содержащие дробной части (например, –5, 1, 1 234 и т.п.).
- **С плавающей запятой.** Числа, содержащие дробную часть (например, 1,01 или –0,06.).

К каждой из этих категорий относится три разных типа, отличающихся диапазоном допустимых значений и (это касается типов с плавающей запятой) точностью определения принимаемых значений. Обобщенная информация об этих типах представлена в табл. 4.1. При объявлении переменных определиться с тем, какой тип выбрать — целочисленный или с плавающей запятой — несложно. Во всех случаях, когда это возможно, следует отдавать предпочтение именно первому из них, поскольку значения целочисленных типов занимают меньше памяти и быстрее обрабатываются (хотя, справедливости ради, следует заметить, в мощных современных компьютерах вы эту разницу вряд ли заметите). В пределах же одной категории следует выбирать тот тип данных, диапазон и точность которого окажутся достаточными для сохранения и обработки значений, передаваемых этой переменной. Обратите внимание, что тип данных `Currency` специально предназначен для представления денежных значений.

*Таблица 4.1. Числовые типы данных VBA*

| *Тип* | *Категория* | *Диапазон* | *Точность* |
|---|---|---|---|
| `Byte` | Целочисленный | от 0 до 255 | нет |
| `Integer` | Целочисленный | −32 768 до 32 767 | нет |
| `Long` | Целочисленный | −2 147 483 648 до 2 147 483 647 | нет |
| `Single` | С плавающей запятой | −3,4×1038 до 3,4×1038 * | 6 знаков |
| `Double` | С плавающей запятой | −1,79×10308 до 1,79×10308 * | 14 знаков |
| `Currency` | С плавающей запятой | −9,22×1011 до 9,22×1011 * | 4 знаков |

* Приблизительные значения


При объявлении числовых переменных им изначально присваивается нулевое значение.

## Строковые переменные

Переменные типа `String` (строковые переменные, или строки) используются для хранения и обработки текстовых данных. *Строки переменной длины* могут включать любое количество символов (приблизительно до двух миллиардов). Вам не придется беспокоиться о длине строки, поскольку она автоматически будет увеличиваться до размеров, достаточных для сохранения передаваемых данных. При объявлении переменных этого типа используется ключевое слово `String`:

```
Dim Имя As String
```

Другой тип строковых переменных в языке VBA называется *строками фиксированной длины*. Максимальная длина таких строковых переменных определяется в момент их объявления, она может колебаться в пределах от 1 до 64 000 символов. Чтобы объявить строковую переменную фиксированной длины, после ключевого слова `String` наберите символ * (звездочка) и затем число, соответствующее максимальному количеству символов:

```
Dim СтрокаФиксированнойДлины As String * 12
```

Если строковым переменным фиксированной длины присваиваются строки, состоящие из большего количества символов, чем объявлено для этой переменной, лишние символы просто отбрасываются. Например:

```
Dim строка As String * 5
строка = "New York"
```

В результате выполнения этого кода переменная `строка` будет содержать значение “New Y” (помните, что пробелы также считаются символами).


Некоторые строки, например “1234”, могут выглядеть как числовые значения. Для VBA, однако, это просто строка, она не может быть использована в числовых вычислениях. Проверить, что последовательность цифр сохраняется именно как строка, можно, начав ее с нуля. Если цифры сохранены в виде строки, ноль будет отображаться.


В VBA включены инструменты, которые позволяют преобразовывать строковые значения в числовые. Эти инструменты будут рассмотрены в уроке 10.

## Переменные даты

В языке VBA термином *дата* обозначаются как собственно значения даты, так и значения времени. Даты отображаются в числовом виде как значения с плавающей запятой. Целая часть такого значения определяет дату как количество дней, прошедших, начиная с 30 декабря 1899 года (предшествующие этой дате дни определяются с помощью отрицательных чисел), а дробная часть указывает время дня как дробную часть от 24-часового интервала. К счастью, вам никогда не придется непосредственно столкнуться с этим числовым представлением даты и времени, поскольку в VBA представлены мощные инструменты, которые позволяют работать с датами, представленными в более привычном и удобном формате. Подробнее о них будет рассказано в уроке 8.

Тип данных `Date` специально предназначен для обработки информации о дате и времени. Чтобы указать значение даты, наберите его в любом из широко распространенных форматов между двумя символами решетки (#). Например:

```
Dim день1 As Date, день2 As Date, день3 As Date
день1 = #Март 7, 2004#
день2 = #1/1/2004#
день3 = #Июль 4, 2004#
```

## Переменные-объекты

Переменные типа `Object` могут содержать ссылку на любой объект. Такая возможность используется нечасто, поскольку при объявлении переменных требуется указывать, значения какого именно типа они будут принимать. Например:

```
Dim ОбщийОбъект As Object
Dim КонкретныйОбъект As Worksheet
```

Объявленная таким образом переменная `ОбщийОбъект` может содержать ссылку на самые различные объекты, тогда как переменная `КонкретныйОбъект` содержит только ссылку на объекты `Worksheet` (т.е. на рабочие листы). На первый взгляд может показаться, что лучше объявлять переменные с использованием типа данных `Object`, поскольку он предоставляет более широкие возможности. Однако это не совсем так, и вот почему.

Использование типа данных `Object` называется *поздней привязкой* (*late binding*). Программа изначально не знает, ссылку на объект какого именно типа будет содержать переменная. Это становится известно только после того, как программа начинает работать и переменной, наконец, присваивается какое-то значение. Данное обстоятельство несколько замедляет работу программы, но не это самое главное. Гораздо важнее, что в таком случае недоступной будет возможность Auto List редактора VBA.

Если вы объявляете переменную с использованием какого-то конкретного объектного типа, это называется *ранней привязкой* (*early binding*). VBA изначально знает, ссылки на объекты какого именно типа будет содержать такая переменная, что позволяет в процессе редактирования кодов программы использовать возможность Auto List, отображающую список методов и свойств объектов данного типа. Это может быть действительно удобно.

Объявление переменных-объектов определенного типа не следует путать с таким действием, как инициализация переменных. Например:

```
Dim ws As Worksheet
```


Чтобы активизировать возможность Auto List, выберите команду Tools⇨Options (Сервис⇨Параметры) и на вкладке Editor (Редактор) открывшегося диалогового окна установите флажок опции Auto List Members (Автоматическое отображение списка компонентов). Если данная возможность активна, при наборе названия переменной-объекта и следующей за ним точки редактор отобразит список методов и свойств объектов этого типа, из числа которых можно выбрать требуемый компонент.

После выполнения этой строки создается переменная `ws`, затем указывается, что она будет ссылаться на объекты типа `Worksheet`. Однако в настоящий момент она ни на что не ссылается, так как содержит значение `Nothing`. Чтобы инициализировать данную переменную, необходимо передать ей конкретную ссылку, например:

```
Set ws = Worksheet.Add
```

Теперь переменная `ws` ссылается на новый рабочий лист, созданный в результате вызова метода `Add`.


Ключевое слово `Set` должно использоваться каждый раз, когда переменной присваивается ссылка на объект.

Как правило, ссылка на объект извлекается из коллекции — это либо уже существующий элемент коллекции, либо он создается в результате вызова метода `Add`. В некоторых ситуациях, однако, требуемой коллекции может и не быть, тогда для создания объекта необходимо использовать ключевое слово `New`. Синтаксис подобного выражения выглядит следующим образом:

```
Set ПеременнаяОбъект = New НазваниеОбъекта
```

В одном выражении можно также совместить объявление с инициализацией переменной:

```
Dim ПеременнаяОбъект As New НазваниеОбъекта
```

## Переменные типа Boolean

Переменные, объявленные с использованием типа `Boolean`, могут принимать одно из двух значений — `True` (ИСТИНА) и `False` (ЛОЖЬ). Такие переменные и свойства (иногда называемые логическими) в кодах программы часто используются для представления и обработки данных: да/нет, принять/отказать и т.п. Логическим переменным в момент объявления автоматически присваивается начальное значение `False`. В языке предусмотрены ключевые слова `True` и `False` для обозначения соответствующих логических значений:

```
Dim Обновить As Boolean
Обновить = True
```

## Тип Variant

В VBA тип `Variant` имеет настолько же широкие возможности, как и тип, используемый по умолчанию. Более того, если в момент объявления переменной ее тип не указывается, она создается как переменная типа `Variant`:

```
Dim x'Объявление переменной X типа Variant
```

Можно также непосредственно указать этот тип данных:

```
Dim x As Variant
```

Переменные, объявленные с использованием типа `Variant`, могут принимать значения практически всех типов, однако они не могут использоваться для представления строк фиксированной длины и для работы с типами, определенными пользователем. Таким образом, переменная типа `Variant` может содержать целочисленные значения, числа с плавающей запятой, строки или ссылки на объекты. Переменная `Variant` содержит целый массив. Недостаток этого типа состоит в том, что для сохранения его значений требуется больше памяти, а для их обработки — большее количество системных ресурсов. Следовательно, вы следует злоупотреблять использованием типа `Variant` лишь потому, что в таком случае вам можно не беспокоиться о выборе более специфического типа данных. Ограничьте применение данного типа только теми ситуациями, в которых его универсальность действительно оказывается востребованной. Приведем приблизительный перечень подобных ситуаций.

- Для представления данных, которые на разных этапах выполнения программы могут интерпретироваться как числовые и как текстовые.
- Для представления аргументов процедуры, которым могут передаваться значения разных типов.
- Для сохранения данных из ячеек рабочего листа, если изначально не известно, значения каких именно типов могут содержаться в этих ячейках.


О методах работы с аргументами процедур будет рассказано уроке 7, а вопросы, связанные с извлечением значений из ячеек рабочего листа, рассматриваются в уроке 10.


## Работа с массивами

Массивы позволяют сохранять в одном месте целые наборы элементов данных. Массив имеет собственное имя, а доступ к отдельным его элементам осуществляется с помощью числовых индексов. В момент создания массива указывается какой-то определенный тип данных VBA, и все элементы этого массива соответствуют именно этому типу данных. Массивы удобны, поскольку позволяют сохранять в одном месте наборы взаимосвязанных значений. Предположим, что программа должна обработать итоговые значения объемов продаж по каждому из 12 месяцев прошедшего года. Вы можете создать массив из 12 элементов и значение данного показателя за январь сохранить, например, как первый элемент этого массива, значение по февралю — как второй элемент и т.д. Использование массивов для работы с данными во многих случаях упрощает задачу написания кодов программы и делает их более удобными для чтения и восприятия. В VBA предусмотрено использование массивов двух типов — статических и динамических.

### Статические массивы

Общее количество элементов *статического массива* строго определено. Размер массива (количество его элементов) определяется в момент его объявления, о он остается неизменным в течение всего процесса выполнения программы. Синтаксис объявления массива имеет следующий вид:

```
Dim ИмяМассива(n) As Тип
```

*ИмяМассива* должно удовлетворять стандартным правилам присвоения имен языка VBA, а *Тип* может быть любым предусмотренным в языке VBA типом данных. Размер массива определяется параметром *n*. В действительности размер

массива будет на единицу бо́льшим числа *n*, поскольку нумерация элементов массивов VBA начинается с индекса 0. Поэтому массив, объявленный как

```
Dim МойМассив(50) As Integer
```

состоит из 51 элемента, все они будут пронумерованы индексами от 0 до 50. Вы получите доступ к элементам массива, если укажете имя массива и заключенное в скобки значение индекса:

```
МойМасив(0) = 1
МойМассив(25) = 73
```

В качестве индекса могут указываться целочисленные переменные или константы:

```
Dim idx As Integer
idx = 12
МойМассив(idx) = 44  ' Равнозначно использованию записи
МойМассив(12)
```

Если вы не хотите, чтобы отсчет индексов массива начинался с нуля, можете воспользоваться одним из двух вариантов. Один из них состоит в добавлении выражения `Option Base 1` в начале модуля за пределами всех процедур. В результате для всех массивов данного модуля отсчет индексов будет начинаться с единицы, а не с нуля. Другой вариант является более гибким и заключается в объявлении массива с использованием ключевого слова `To`:

```
Dim ИмяМассива(НижняяГраница To ВерхняяГраница) As Тип
```

Значения *НижняяГраница* и *ВерхняяГраница* должны быть целочисленными, причем первое обязательно должно быть меньше второго. Например:

```
Dim МойМассив(20 To 40) As Long
```

Массивы с одним индексом называются *одномерными*. Чтобы создать массив с двумя или несколькими индексами, укажите их в момент объявления массива.

```
Dim Двухмерный(10, 10) As Single
```

Такой массив состоит из 121 элемента, он начинается элементом `Двухмерный(0, 0)` и заканчивается элементом `Двухмерный(10, 10)`. Первая размерность включает в себя 11 элементов, вторая — еще 11. Общее количество элементов массива вычисляется как произведение числа 11 на число 11. При объявлении многомерных массивов также можно использовать ключевое слово `To`:

```
Dim ШахматнаяДоска (1 To 8, 1 To 8) As String
```

В VBA не предусмотрено никаких ограничений на максимально допустимое количество элементов массива и на количество его размерностей. На практике, однако, и первая, и вторая характеристика ограничены доступными объемами системной памяти и дискового пространства, но эти ограничения вряд ли когда-нибудь вам смогут помешать. В случае, если ваша система работает довольно медленно по причине недостаточного количества доступной памяти или дискового пространства, работа с массивами больших размеров может усугубить ситуацию.


Если какая-то процедура попытается обратиться к несуществующему элементу массива, это вызовет ошибку выполнения программы. Например, если массив объявлен как `Dim МойМассив(20) As Integer`, то при введении выражения `МойМассив(21)`, `МойМассив(30)` и т.п., возникнет ошибка.

## Динамические массивы

Размер *динамических массивов* не является фиксированным. В ходе выполнения программы он может увеличиваться и уменьшаться по мере необходимости. Динамический массив объявляется с помощью пустых скобок:

```
Dim ДинамическийМассив() As Тип
```

Прежде чем приступить к использованию массива, необходимо определить его размер с помощью выражения ReDim:

```
ReDim ДинамическийМассив(размер)
```

Аргумент *размер* определяет как размерность массива, так и количество элементов в каждой размерности. Например:

```
Dim Динамический1() As String
Dim Динамический2() As Integer
Dim Динамический3() As Object
...
ReDim Динамический1(100)          ' 1 размерность, 101 элемент
ReDim Динамический2(-5 To 5)      ' 1 размерность, 11 элементов
                                  ' с -5 по 5
ReDim Динамический2(5, 5, 5)      ' 3 размерности, всего 216
                                  ' элементов
```

Выражение ReDim не позволяет изменять тип динамического массива, однако с его помощью вы можете изменять размер этого массива любое количество раз. Как правило, при использовании выражения ReDim все прежние значения массива теряются. Чтобы сохранить уже имеющиеся данные (по возможности), добавьте к выражению ReDim ключевое слово Preserve:

```
ReDim Preserve Динамический(100)
```

Существуют следующие ограничения на сохранение данных с помощью ключевого слова Preserve:

- если размер массива уменьшается, значения отбрасываемых элементов теряются;
- для многомерных массивов вы можете изменять только объем последней размерности;
- вы не можете изменять количество размерностей.

В листинге 4.1 представлен пример использования динамического массива. Чтобы применить данную программу на практике, расположите курсор над верхней ячейкой столбца числовых значений и затем запустите программу. Программа пройдет вниз по ячейкам столбца и скопирует все его значения, сохранив их как элементы массива. Для каждой очередной ячейки программа будет использовать выражение ReDim, с помощью которого добавляются новые элементы массива. Когда будет достигнута пустая ячейка, программа просуммирует значения всех элементов массива. В результате будет получено итоговое значение, которое затем помещается в ячейку под исходным столбцом значений.

Ниже перечислены действия, которые необходимо выполнить для создания и выполнения данной программы.

1. В Excel на рабочем листе наберите столбец числовых значений.
2. Нажмите клавиши <Alt+F11>, чтобы открыть редактор VBA.
3. В окне Project Explorer дважды щелкните на названии рабочего листа, в котором вы набирали числовые значения. Откроется окно редактора кодов.

4. В открывшемся окне наберите коды, представленные в листинге 4.1.
5. Опять вернитесь к окну Excel и разместите курсор над верхней ячейкой столбца.
6. Нажмите клавиши <Alt+F8>, чтобы открыть диалоговое окно Макрос.
7. Выберите макрос SumColumn и затем щелкните на кнопке Выполнить.

В данной программе используются некоторые элементы языка VBA, с которыми вы пока еще мало знакомы. Возможно, вы сами догадаетесь об их назначении. Сосредоточьте свое внимание на представленном далее способе использования динамического массива.

***Листинг 4.1. Пример использования динамического массива***

```
Public Sub SumColumn()
Dim data() As Single
Dim total As Single
Dim count As Integer
Dim i As Integer
Dim finished As Boolean
count = 0

Do
     count = count + 1
     ReDim Preserve data(count)
     data(count) = ActiveCell.Value
     ActiveCell.Offset(1, 0).Activate
     If ActiveCell.Value = "" Then finished = True
Loop Until finished

For i = 1 To Ubound(data)
     total = total + data(i)
Next

ActiveCell.Value = total

End Sub
```

**Определение размеров массива**

В VBA существуют две функции, которые позволяют определить размеры любого массива. В частности, они предоставляют возможность определить наименьший и наибольший допустимый индекс массива:

```
UBound(ИмяМассива, размерность);
LBound(ИмяМассива, размерность).
```

Функция UBound возвращает наибольший, а функция LBound — наименьший допустимый индекс заданного массива. Аргумент *размерность* является необязательным, он определяет, индекс какой размерности массива требуется вернуть. По умолчанию его значение равно единице. Например:

```
Dim МойМассив(1 To 5, 10 To 25)
x = LBound(МойМассив)' Возвращает 1
x = UBound(МойМассив, 2)   ' Возвращает 25
```

## Определенные пользователем типы

*Определенные пользователем* (т.е. программистом) типы (далее для удобства мы будем обозначать их аббревиатурой ОПТ) служат инструментом определения специальных элементов данных, которые наилучшим образом подходят для обработки той или иной специфической информации. Такие типы могут содержать от двух и более элементов. Каждый элемент может быть либо переменной, либо массивом. Для определения соответствующего типа используйте выражение Type…End Type:

```
Type НазваниеТипа
     ИмяЭлемента1 As Тип
     ИмяЭлемента2 As Тип
     ...
End Type
```

Каждый элемент объявляется с учетом стандартных правил объявления переменных. При этом для элементов могут быть выбраны любые доступные типы данных. Элемент может быть также массивом либо экземпляром другого объявленного вами типа. Приведем пример определения ОПТ, предназначенного для представления информации о сотрудниках:

```
Type СотрудникИнфо
     Имя As String
     Фамилия As String
     Код As String * 10
     Стаж As Date
     Страховка As Boolean
     Оклад As Currency
End Type
```

Посмотрим, что происходит при выполнении каждой строки этого кода.

- Первой строкой начинается определение данного типа.
- Во второй строке объявляется первая переменная ОПТ — Имя типа String.
- В третьей строке объявляется вторая переменная ОПТ — Фамилия типа String.
- В последующих строках объявляются другие переменные ОПТ.
- Последней строкой завершается определение данного типа.

Код определения ОПТ должен помещаться в модуле за пределами всех процедур. После того, как ОПТ будет определен, вы можете использовать его, как и другие стандартные типы VBA, при объявлении переменных. Например:

```
Dim Сотрудник As СотрудникИнфо
Dim ВсеСотрудники(100) As СотрудникИнфо
```

Для получения доступа к элементам (переменным) ОПТ используется синтаксис *имя.элемент*. Например:

```
Сотрудник.Имя = "Степан"
Сотрудник.Фамилия = "Петров"
ВсеСотрудники(1).Имя = "Никодим"
```

## Перечислимые типы

Перечислимые типы данных состоят из наборов символьных констант, указанных программистом. Переменные, объявленные с использованием перечис-

лимого типа, могут принимать только одно из предустановленных значений. Создание и работа с такими типами удобна в тех случаях, когда природа самих данных такова, что переменные должны принимать только некоторые строго определенные значения. Например, это могут быть названия дней недели. Используя перечислимый тип, содержащий названия семи дней недели, вы точно будете уверены, что соответствующая переменная не примет какое-то нестандартное неверное значение.

Перечислимые типы определяются с помощью выражения Enum…End Enum:

```
Enum ИмяТипа
     Элемент1
     Элемент2
     ...
End Enum
```

Например:

```
Enum Наполнитель
     Ваниль
     Шоколад
     Земляника
     Кофе
End Enum
```

Каждой константе перечислимого типа присваивается числовое значение. Если вы сами не укажите эти значения, они будут присвоены автоматически, начиная с нулевого и далее в порядке возрастания. Так, в предыдущем примере константа Ваниль получила значение 0, константа Шоколад — значение 1 и т.д. Если это необходимо, можете сами указать требуемые значения:

```
Enum Наполнитель
     Ваниль = 1
     Шоколад = 4
     Земляника = 7
     Кофе = 16
End Enum
```

Перечислимый тип должен быть определен на уровне модуля за пределами всех процедур (подобно определяемым пользователем типам). После этого такой тип можно использовать при объявлении переменных:

```
Dim Мороженое As Наполнитель
Dim Ассортимент(50) As Наполнитель
```

При работе с переменным перечислимых типов Auto List редактора VBA отображает перечень констант данного типа, и это действительно удобно, поскольку вы можете просто выбрать требуемую константу из списка и не набирать ее вручную.

## Область видимости переменных

Переменные, объявленные в программе VBA, остаются доступными для использования только в пределах определенной части данной программы. Этот факт определяется таким понятием, как *область видимости переменных*, согласно которому каждая переменная доступна только в пределах своей области видимости. Для других частей программы эта переменная, по сути, не существует. Вы можете работать с двумя одинаковыми переменными, которые имеют разную область видимости, и они будут абсолютно независимы друг от друга.

В VBA различают три уровня видимости.

- **Уровень процедуры.** Область видимости ограничена одной конкретной процедурой VBA.
- **Уровень модуля.** Область видимости ограничена одним конкретным модулем (и распространяется на все процедуры этого модуля).
- **Уровень проекта.** Область видимости распространяется на весь проект.

Чтобы создать переменную с областью видимости на уровне процедуры, объявите ее с помощью выражения `Dim` внутри требуемой процедуры. Всегда ограничивайте область видимости переменной уровнем процедуры, если не существует каких-либо веских причин для использования более широкой области видимости. Большинство переменных вашей программы должны иметь именно такую область видимости.

Чтобы создать переменную с областью видимости на уровне модуля, объявите ее, используя выражение `Dim`, внутри этого модуля за пределами всех процедур. Такой уровень видимости применяется в тех случаях, когда доступ к данной переменной необходим сразу нескольким процедурам этого модуля (подобная ситуация возникает при работе с процедурами свойств, о которых будет рассказано в уроке 26). Тот же эффект может быть достигнут в том случае, если вместо слова `Dim` используется ключевое слово `Private`:

```
Private Total As Integer
```

Область видимости на уровне проекта бывает двух видов. Если переменная объявляется с помощью ключевого слова `Public`, ее область видимости распространяется на все модули данного проекта, а также на все открытые в данный момент проекты VBA. Если в начале модуля будет добавлено выражение Option Private Module, область видимости переменной, объявленной с помощью ключевого слова `Public`, ограничивается пределами данного проекта.


Если области видимости двух переменных с одинаковым именем пересекаются, приоритет имеет та переменная, область видимости которой меньше.

При выборе области видимости для переменной следует использовать как можно более ограниченную область. Больш*ая* часть переменных программы имеет область видимости на уровне процедуры. Изолируя переменные в пределах процедуры, вы исключаете тем самым возможность непреднамеренного изменения их значений и существенно уменьшаете вероятность возникновения ошибок. Если необходимо передать информацию от одной процедуры к другой, используйте возможности возвращения процедурами значений и передачи процедурам значений аргументов. Прибегайте к помощи переменных с более широкой областью видимости только лишь в крайних случаях, при этом необходимо внимательно следить за корректностью их использования.


## Обзор

Практически все программы Excel создаются с целью обработки какой-то информации. Возможности VBA в этом плане практически неограничены. Различные типы данных могут быть использованы для работы с самыми разными значениями, в том числе с числовыми значениями, с текстом, с датами и со ссылками на объекты. В данном уроке были рассмотрены следующие вопросы.

- Основы синтаксиса VBA, в том числе правила использования комментариев.
- Способы применения символьных и литеральных констант.

- Правила VBA о присвоении имен переменным.
- Способы объявления переменных.
- Типы числовых, строковых и объектных переменных.
- Область видимости переменных.
- Способы создания и использования статических и динамических массивов.
- Использование определенных пользователем и перечислимых типов.

## Проверьте себя

1. Как разбить выражение VBA на несколько строк? (См. раздел "Основы синтаксиса VBA".)
2. Назовите два способа создания комментариев. (См. подраздел "Комментарии".)
3. Назовите преимущества использования символьных констант. (См. раздел "Использование констант".)
4. Требуется ли применять выражение Option Explicit при написании программ? Поясните ответ. (См. врезку "Объявление переменных и выражение Option Explicit".)
5. Какой тип данных предназначен для работы с денежными значениями? (См. подраздел "Числовые переменные".)
6. Какое выражение используется для создания определенного пользователем типа? (См. подраздел "Определенные пользователем типы".)
7. Когда необходимо использовать тип данных `Object`? (См. подраздел "Переменные-объекты".)
8. Что означает термин *область видимости переменных*? (См. раздел "Область видимости переменных".)

ЧАСТЬ

# Пятница. Вечер. Обзор

1. Что такое программа?
2. Как запустить редактор VBA из Excel?
3. Как вставить новый модуль в проект VBA?
4. Назовите простейший способ копирования кодов из одного проекта VBA в другой.
5. Что такое свойство?
6. Как создается абсолютная ссылка на ячейку?
7. В чем различие между свойствами и методами объектов?
8. Всегда ли при вызове метода необходимо передавать аргументы в том порядке, в котором они были объявлены?
9. Какой оператор VBA применяется для организации прохождения по всем членам коллекции?
10. В каких случаях в программе, работающей в Excel, необходимо использовать ключевое слово `Application`?
11. Как можно сохранить копию рабочей книги под новым именем, оставив при этом название исходной книги без изменений?
12. Какой метод следует использовать для вывода текущей рабочей книги на печать?
13. Как получить ссылку на активный рабочий лист текущей рабочей книги?
14. Каким образом в кодах программы можно изменить название рабочего листа (которое отображается на его корешке)?
15. Как отменить отображение диалогового окна Excel о подтверждении удаления рабочего листа, если команда на удаление указывается в кодах программы?
16. Для каких целей служит ключевое слово `Nothing`?
17. Как в кодах VBA определить, сколько именно листов содержится в данный момент в текущей рабочей книге?
18. Какое назначение свойства `Workbook.CreateBackup`?
19. Как получить полное имя файла (с указанием пути) рабочей книги?
20. Если в Excel открыто несколько рабочих книг, как сделать активной требуемую книгу?

21. Как определить, сохранялась ли рабочая книга после внесения в нее последних изменений?
22. Какие ограничения (если они есть) существуют на использование символа продолжения строки в кодах VBA?
23. Какими правилами VBA регулируется применение отступов в кодах программы?
24. Каким образом в языке VBA фиксируется информация о датах?
25. Массив VBA был объявлен таким образом: `Dim Data(100) As Integer`. Из какого количества элементов он состоит?
26. Как в кодах программы определить размер массива?
27. Что такое *область видимости* переменной?
28. Как следует объявить переменную, область видимости которой должна распространяться на весь проект?