**Основы работы с модулями классов**

[**http://www.excelworld.ru/publ/vba/class\_module/class\_module/42-1-0-110**](http://www.excelworld.ru/publ/vba/class_module/class_module/42-1-0-110)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Создание приложения на vba путем написания процедур и функций стоит называть процедурным программированием. Написание участков кода, называемых процедурами и функциями, которые описывают какое-либо действие, и последующее последовательное исполнение этих кодов, является основным методом программирования на vba. При этом используется доступ к объектной модели Excel или других приложений. Данные и подпрограммы (функции и процедуры) функционально не связаны между собой. Это значит, что объявляя переменную «длина хвоста» на уровне модуля, нельзя задать ей различные значения для двух процедур.   При объектно-ориентированном программировании (**ООП**) подход иной. Данные и подпрограммы связаны между собой и описываются в классе.   **Класс** – сложный тип данных, в котором описывается структура объекта. Объект – основной строительный материал для написания программ. Отличные представители объектов – форма, лист, книга, диаграмма. Объект, созданный на основе класса, можно называть экземпляром класса.   Класс имеет следующую структуру:   **Поле** – элемент класса для хранения данных,   **Свойство** – элемент класса для хранения данных с возможностью их обработки,   **Метод** – аналог процедуры или функции,   **Событие** – сигнал при изменении состояния объекта, например исполнения метода или изменения данных.   Из всех принципов ООП в vba реализуемы только два: **Абстрагирование** и **инкапсуляция**.   **Инкапсуляция** позволяет скрыть механизм работы класса, оставив открытыми для других процедур только необходимые для работы с экземпляром класса.  **Абстракция** позволяет создать объект, наиболее приближенный к некомпьютерному прототипу. Например, создать объект «Кошка» со свойствами «количество лап» и «цвет хвоста».  **Создание класса**        Для создания класса в vba редакторе выберите в меню Insert строку Class Module. Назовите созданный класс путем переименования созданного модуля. В файле с примером он называется ExampleClass. А модуль, демонстрирующий использование этого класса называется ExClassManagement. В свойствах класса, кроме имени так же есть параметр Instancing. Указывается, будет ли виден класс из другой книги при установке ссылки на данную книгу. При установке Private (по умолчанию) класс виден только в данной книге, при установке PublicNotCreatable, класс не будет доступен из другой книги, однако экземпляр класса доступен будет, если он создан в данной книге.  **Создание экземпляра класса**       Класс - это всего лишь описание объекта. Для использования возможностей класса, необходимо создать экземпляр класса (объект). Существует несколько способов:   **Способ 1:**   Private Sub TestClass()  Dim cl As ExampleClass  Set cl = New ExampleClass End Sub  Данный способ корректен абсолютно   **Способ 2:**  Dim cl As ExampleClass  Private Sub TestClass()  Set cl = New ExampleClass End Sub  Этот способ отличается от первого способа тем, что экземпляр класса объявляется вне процедуры и работать с ним можно во всех процедурах модуля. При замене Dim на Public экземпляр класса доступен во всем проекте, если объявляется вне объектного модуля.    **Способ 3:**  Dim cl WithEvents As ExampleClass  Private Sub TestClass()  Set cl = New ExampleClass End Sub  Экземпляр класса объявляется с событиями, и если в классе описаны события объекта, они будут доступны. Работает только при объявлении в объектном модуле (модуль класса, формы, листа, книги)   **Способ 4:**   Private Sub TestClass()  Dim cl As New ExampleClass  End Sub  Так называемый неявный метод создания экземпляра класса. В этом случае объект создается при первом обращении к переменной cl. Наверное, предпочтительнее сначала объявлять переменную (выделяется память), а затем явно создавать объект.   **Уничтожение экземпляра класса**  Естественно, после использования экземпляра класса, необходимо очистить память. Делается это одним способом:   Set cl = Nothing  Если не деинициализировать экземпляр класс, то ничего глобально страшного происходить в современных офисных пакетах не будет. Но делать это нужно.   **Создание полей класса**  Созданный класс ExampleClass не имеет полей, свойств, методов, событий, поэтому и созданный на его основе объект (экземпляр класса) cl, так же бесполезен.   Поля это переменные класса, объявленные в его модуле. Поля бывают закрытые и открытые. Доступ к закрытым полям возможен только внутри модуля класса. Открытое поле, по сути – свойство класса, и при создании экземпляра класса оно будет доступно.   **Создание закрытого поля**:    Dim sBody As String  или   Private sBody As String  **Создание открытого поля**:    Public Head As String  И теперь свойство Head доступно у экземпляра класса cl.   В него можно записать   cl.Head = "FHead"  и прочитать   Debug.Print cl.Head  Создать поле с пользовательским типом данных не удастся.        Поля используются для хранения данных в объекте. Данные будут доступны, пока объект существует. Однако нельзя указать значение по-умолчанию для поля и нельзя сделать поле только для чтения.   **Создание свойства класса**        Свойство это способ доступа к данным внутри объекта. Выглядят как поля, однако, это функции (назовем их методами). Называются они Property Get для чтения данных из объекта, и Property Let для записи данных в объект. Есть еще третье Property Set для установки ссылки на другой объект. Но это можно сделать и при помощи Property Let, поэтому Property Set вещь бесполезная.   Синтаксис **Property Get**  ***[Public | Private | Friend] [Static] Property Get имя [(аргументы)] [As тип]*** *[произвольный код]  [имя=выражение]* ***[Exit Property]*** *[произвольный код]  [имя=выражение]* ***[End Property]***   |  |  | | --- | --- | | **Элемент** | **Описание** | | Public | Не обязательно. Делает метод открытым во всех модулях проекта | | Private | Не обязательно. Метод будет доступен только в модуле класса, то есть не будет виден как свойство | | Friend | Не обязательно. Метод будет виден во всех модулях проекта, но не будет  виден по ссылке на класс. то есть при конструкции Dim cl as new  ExampleClass; Dim cll as object; Set cll=cl в объекте cll свойство с  модификатором Friend видно не будет, а в объекте cl будет | | Static | Не обязательно. Значения объявленных локальных переменных внутри метода  сохраняются между обращениями. Не распространяется на переменные,  объявленные на уровне модуля класса | | имя | Обязательно. Имя метода будет являться и именем свойства. при этом оно  должно совпадать для методов Property Let или Property Set | | аргументы | Не обязательно. По сути, это аргументы функции. Имена и типы аргументов должны совпадать с аргументами метода Property Let | | тип | Не обязательно. Тип данных, возвращаемый функцией может быть Byte,  Boolean, Integer, Long, Currency, Single, Double, Date, String, Object,  Variant, пользовательским типом. Должен совпадать с типом данных  выражения | | произвольный код | Не обязательно. Любое количество строк кода, например математические операции с возвращаемым значением | | выражение | Значение, возвращаемое методом Property Get. Если не указать, метод  вернет пустую строку для типа String, False для Boolean, 0 для Integer и  так далее |   ***Замечания:***  *- Элемент****Exit Property****означает выход из метода, и по сути аналог****Exit Function****. В методе их может быть несколько.       - Внутри метода****Property Get****может быть сколько угодно процедур и  функций, но сам метод не может входить в состав других процедур и  функций, одноименных методов****Property Get****в модуле класса быть не может.*   Синтаксис**Property Let**  ***[Public | Private | Friend] [Static] Property Let имя ([аргументы,] значение)*** *[произвольный код]* ***[Exit Property]*** *[произвольный код]* ***[End Property]***   |  |  | | --- | --- | | **Элемент** | **Описание** | | Public | Не обязательно. Делает метод открытым во всех модулях проекта | | Private | Не обязательно. Метод будет доступен только в модуле класса, то есть не будет виден как свойство | | Friend | Не обязательно. Метод будет виден во всех модулях проекта, но не будет  виден по ссылке на класс. то есть при конструкции Dim cl as new  ExampleClass; Dim cll as object; Set cll=cl в объекте cll свойство с  модификатором Friend видно не будет, а в объекте cl будет | | Static | Не обязательно. Значения объявленных локальных переменных внутри метода  сохраняются между обращениями. Не распространяется на переменные,  объявленные на уровне модуля класса | | аргументы | Не обязательно. По сути, это аргументы функции. Имена и типы аргументов должны совпадать с аргументами метода Property Get | | значение | Не обязательно. Имя переменной метода, которой будет присваиваться  значения свойства. Тип данных значения должен совпадать с типом метода  Property Get | | произвольный код | Не обязательно. Любое количество строк кода, например проверка  указываемого значения. И конечно необходимо здесь передать значение из  локальной переменной (значение) метода во внешнюю переменную модуля  класса. А значение внешней переменной передать в выражение метода  Property Get |   ***Замечание:***       - *Аналогично методу****Property Get***  Синтаксис **Property Set**  ***[Public | Private | Friend] [Static] Property Set имя ([аргументы,] ссылка)*** *[произвольный код]* ***[Exit Property]*** *[произвольный код]* ***[End Property]***   |  |  | | --- | --- | | **Элемент** | **Описание** | | Public | Не обязательно. Делает метод открытым во всех модулях проекта | | Private | Не обязательно. Метод будет доступен только в модуле класса, то есть не будет виден как свойство | | Friend | Не обязательно. Метод будет виден во всех модулях проекта, но не будет  виден по ссылке на класс. то есть при конструкции Dim cl as new  ExampleClass; Dim cll as object; Set cll=cl в объекте cll свойство с  модификатором Friend видно не будет, а в объекте cl будет | | Static | Не обязательно. Значения объявленных локальных переменных внутри метода  сохраняются между обращениями. Не распространяется на переменные,  объявленные на уровне модуля класса | | аргументы | Не обязательно. По сути, это аргументы функции. Имена и типы аргументов должны совпадать с аргументами метода Property Get | | ссылка | Не обязательно. Имя переменной метода, которая будет ссылкой на объект. Тип данных ссылки должен быть object | | произвольный код | Не обязательно. Любое количество строк кода, например работа со свойствами объекта. И конечно необходимо здесь передать значение из  локальной переменной (значение) метода во внешнюю переменную модуля  класса. А значение внешней переменной передать в выражение метода  Property Get |   ***Замечание:****- Не забывайте, передача ссылки на объект происходит при помощи Set.*  **Примеры создания свойства в модуле класса Pacient:**  Private sHeight As Single  Public Property Get Height()  As Single Height = sHeight  End Property  Public Property  Let Height(ByVal sHeightValue As Single)  sHeight = sHeightValue  End Property  Свойство **Height** доступно для чтения и записи. при этом при записи и чтении никаких модификаций с данными не производится, то есть такая конструкция аналогична открытому полю (Public Height as Single), а потому создание ее смысла не имеет. Просто лишний код.  Private sHeight As Single  Public Property Get Height() As Single  Height = sHeight  End Property        Свойство доступно только для чтения. Метод **Property Let** не создавался. Предполагается, что данные переменной **sHeight** заданы в какой-либо процедуре модуля класса.  Const sHeight As Single = 2  Public Property  Get Height() As Single  Height = sHeight  End Property  Свойство доступно только для чтения и содержит константу. Единственный способ открытия константы из объектного модуля. Конструкция **Public Const sHeight As Single = 2** работать не будет.  Private sHeight As Single  Public Property  Let Height(ByVal sHeightValue As Single)  sHeight = sHeightValue  End Property  Свойство только для записи. при попытке его прочитать, появится ошибка.  Private sHeight As Single  Public Property  Get Height() As Single  Height = sHeight  End Property  Public Property  Let Height(ByVal sHeightValue As Single)  Select Case sHeightValue  Case Is > 250  sHeight = 250  Case Is < 50  sHeight = 50  Case Else  sHeight = sHeightValue  End Select End Property  Происходит проверка данных при записи значения.  Private sHeight As Single  Public Property  Get Height() As Single  Height = sHeight / 100  End Property  Public Property  Let Height(ByVal sHeightValue As Single)  sHeight =  sHeightValue  End Property  Пример произвольного кода в методе **Property Get**. В свойство **Height** записывается значение в сантиметрах, а читается в метрах.  Private sHeight As Single  Public Property Get Height(ByVal Scales As Integer) As Single  Select Case Scales  Case 1  Height = sHeight  Case 2  Height = sHeight / 10  Case 3 Height = sHeight / 1000  End Select  End Property  Public Property Let Height(ByVal Scales As Integer, ByVal sHeightValue As Single)  Select Case Scales  Case 1  sHeight = sHeightValue  Case 2  sHeight = sHeightValue \* 10  Case 3 sHeight = sHeightValue \* 1000  End Select  End Property  Свойство со аргументом. Аргумент **Scales** указывает, в каких единицах измерения записывается рост, а в каких читается.  Private sHeight As Single  Private sWeight As Single  Public Property Let Height(ByVal sHeightValue As Single)  sHeight = sHeightValue  End Property  Public Property Let Weight(ByVal sWeightValue As Single)  sWeight = sWeightValue  End Property  Property Get IMT() As Single  If sHeight <> 0  Then IMT = Round(sWeight / sHeight ^ 2, 1)  End Property        Свойства **Height** и **Weight** только для записи, свойство **IMT** только для чтения и содержит индекс массы тела, рассчитанный на основе веса и роста. Расчет происходит всякий раз при обращении к свойству **IMT**.   **Методы класса**       В созданном классе можно создавать процедуры и функции. Они будут видны в экземпляре класса как методы, если указаны как Public и не видны, если указаны как Private. Все процедуры и функции, которые не планируется использовать как методы, должны быть Private. Этого требует принцип инкапсуляции.   Их создание ничем не отличается от создания процедур и функций в обычном модуле. Процедуры, как и функции, могут содержать обязательные и не обязательные аргументы. Однако вызов их отличается.   |  |  | | --- | --- | | **Код в классе** | **Вызов метод** | | Public Function IMT(sWeight As Single, sHeight As Single) As Single  If sHeight <> 0  Then IMT = Round(sWeight / sHeight ^ 2, 1)  End Function | Dim oPac As Pacient  Set oPac = New Pacient  Debug.Print oPac.IMT(113, 1.83)  Set oPac = Nothing | | Public IMT As Single  Public Sub CalcIMT(sWeight As Single, sHeight As Single)  If sHeight <> 0  Then IMT = Round(sWeight / sHeight ^ 2, 1)  End Sub | Dim oPac As Pacient  Set oPac = New Pacient  oPac.CalcIMT sWeight:=113, sHeight:=1.83  Debug.Print oPac.IMT  Set oPac = Nothing |   В первом случае значение индекса массы тела возвращает функция, во втором открытое поле **IMT**.   Классический вариант:   |  |  | | --- | --- | | **Код в классе** | **Вызов метод** | | Private sHeight As Single  Private sWeight As Single  Private sIMT As Single  Public Property Get Height() As Single  Height = sHeight  End Property  Public Property Let Height(ByVal sHeightValue As Single)  sHeight = sHeightValue  End Property  Public Property Get Weight() As Single  Weight = sWeight  End Property  Public Property Let Weight(ByVal sWeightValue As Single)  sWeight = sWeightValue  End Property  Public Property Get IMT() As Single  IMT = sIMT  End Property  Public Sub CalcIMT()  If sHeight <> 0  Then sIMT = Round(sWeight / sHeight ^ 2, 1)  End Sub | Dim oPac As Pacient  Set oPac = New Pacient  oPac.Height = 1.83  oPac.Weight = 113  oPac.CalcIMT  Debug.Print oPac.IMT  Set oPac = Nothing |    Свойства**Height** и **Weight** для чтения и записи, свойство**IMT** только для чтения. Индекс массы тела рассчитывается методом **CalcIMT**.  **События класса**        Созданный класс уже имеет два скрытых события:  **Class\_Initialize**  - Происходит при создании экземпляра класса. В этом событии удобно указывать значения свойств и переменных по-умолчанию. **Class\_Terminate** - Происходит при уничтожении экземпляра класса. Экземпляр класса уничтожается, когда процедура, в которой он был объявлен, завершает свою работу. Или после явной деинициализации экземпляра класса:  **Set oPac=Nothing**   Добавление собственных событий в класс, которые будут происходить при определенных условиях, не составляет особых сложностей. Единственное условие – экземпляр класса с событиями должен (может) быть объявлен только в объектном модуле (модуль класса, формы, листа, книги) на уровне модуля.   Private WithEvents  oPac As Pacient  А в самом модуле класса указывается событие:   Синтаксис: ***[Public] Event****имя[(аргументы)]*   |  |  | | --- | --- | | **Элемент** | **Описание** | | Public | Не обязательно. Делает событие открытым во всех объектных модулях проекта. По-умолчанию, все пользовательские события Public | | имя | Имя события, которое будет видно в экземпляре класса | | аргументы | Не обязательно. События могут иметь аргументы, которым можно передавать значения или ссылку на объект |   ***Замечание:****Событие должно указываться в самом начале модуля класса*       В месте, где должно происходить событие указывается триггер с синтаксисом:   ***RaiseEven****t имя[(значение аргументов)]*   |  |  | | --- | --- | | **Элемент** | **Описание** | | имя | Обязательно. Имя события, для которого указывается триггер | | значение аргументов | Не обязательно. Если при указании события были указаны аргументы, здесь через запятую указываются их значения |         Вот пример кода расчета индекса массы тела в форме.   ***Код в классе:***  Public Event IMTCalculated(IMTValue As Single)  Private sHeight As Single  Private sWeight As Single  Private sIMT As Single  Public Property Get Height() As Single  Height = sHeight  End Property  Public Property Let Height(ByVal sHeightValue As Single)  sHeight = sHeightValue  End Property  Public Property Get Weight() As Single  Weight = sWeight  End Property  Public Property Let Weight(ByVal sWeightValue As Single)  sWeight = sWeightValue  End Property  Public Sub CalcIMT()  If sHeight <> 0 Then sIMT = Round(sWeight / sHeight ^ 2, 1)  RaiseEvent IMTCalculated(sIMT)  End Sub  ***Вызов метода в форме:***  Private WithEvents oPac As Pacient  Private Sub CommandButton1\_Click()  Set oPac = New Pacient  oPac.Height = 1.83  oPac.Weight = 113  oPac.CalcIMT  End Sub  Private Sub oPac\_IMTCalculated(IMTValue As Single)  Me.Label1.Caption = IMTValue  End Sub  Событие **IMTCalculated** срабатывает после расчета индекса массы тела, и в форме значение аргумента**IMTValue** присваивается надписи. Таким образом, отпадает необходимость в создании отдельного свойства**IMT**.  **Практическое применение модулей классов**        Большинство разработчиков никогда не используют модули классов в своих программах. Но есть задачи, которые нельзя решить без написания пользовательских классов:   |  |  | | --- | --- | | **Задача** | **Пример файла** | | Выполнение одной процедуры несколькими контролами формы | [http://www.excelworld.ru/button/my_button/Download_file.gif](http://www.excelworld.ru/Author/ratboy/ClassModule/neskolko_kontrolov_s_odnoj_proceduroj.xls) | | Управление событиями на уровне приложения | [http://www.excelworld.ru/button/my_button/Download_file.gif](http://www.excelworld.ru/Author/ratboy/ClassModule/upravlenie_sobytijami_na_urovne_prilozhenija.xls) |               Есть случаи, когда использовать пользовательские классы удобнее, чем процедуры и функции:   |  |  | | --- | --- | | **Задача** | **Пример файла** | | Сокрытие деталей реализации компонента для упрощения работы с ним и  создание собственной библиотеки компонентов | [http://www.excelworld.ru/button/my_button/Download_file.gif](http://www.excelworld.ru/Author/ratboy/ClassModule/proverka_orfografii.xls) | | [http://www.excelworld.ru/button/my_button/Download_file.gif](http://www.excelworld.ru/Author/ratboy/ClassModule/nejroset.zip) |   Желаю успехов в разработке ratboy [ondister@gmail.com](mailto:ondister@gmail.com) | |
|  |  |
| * [1](http://www.excelworld.ru/publ/vba/class_module/class_module/42-1-0-110) * [2](http://www.excelworld.ru/publ/vba/class_module/class_module/42-1-0-110) * [3](http://www.excelworld.ru/publ/vba/class_module/class_module/42-1-0-110) * [4](http://www.excelworld.ru/publ/vba/class_module/class_module/42-1-0-110) * [5](http://www.excelworld.ru/publ/vba/class_module/class_module/42-1-0-110)   Категория: [Модули классов](http://www.excelworld.ru/publ/vba/class_module/42) | Добавил: [ratboy](javascript://) (31.03.2012) | Автор: ratboy [E](javascript://) | |
|  |  |
| Просмотров: **16758** | Комментарии: **8** | Рейтинг: **5.0**/**9** | |

|  |  |
| --- | --- |
| Всего комментариев: **8** |  |
| Порядок вывода комментариев:         |  | | --- | | **0**  http://s5.ucoz.net/img/icon/thumbu_.png http://s5.ucoz.net/img/icon/thumbd_.png  **[1](http://www.excelworld.ru/publ/vba/class_module/class_module/42-1-0-110" \l "ent62)**    **[ratboy](javascript://)**   (06.04.2012 07:26)     Замечательно. В примере с полями лишний тег <br> |  |  | | --- | | **0**  http://s5.ucoz.net/img/icon/thumbu_.png http://s5.ucoz.net/img/icon/thumbd_.png  **[2](http://www.excelworld.ru/publ/vba/class_module/class_module/42-1-0-110" \l "ent63)**    [**Serge\_007**](javascript://)   (06.04.2012 09:40)  [[http://www.excelworld.ru/avatar/81/076655.gif](javascript://)](javascript://)   Убрал) |  |  | | --- | | **0**  http://s5.ucoz.net/img/icon/thumbu_.png http://s5.ucoz.net/img/icon/thumbd_.png  **[3](http://www.excelworld.ru/publ/vba/class_module/class_module/42-1-0-110" \l "ent66)**    **Серго**   (07.04.2012 12:42)     Сложно для моего понимания, но буду разбираться, спасибо! |  |  | | --- | | **0**  http://s5.ucoz.net/img/icon/thumbu_.png http://s5.ucoz.net/img/icon/thumbd_.png  **[4](http://www.excelworld.ru/publ/vba/class_module/class_module/42-1-0-110" \l "ent112)**    **Сергей**   (01.10.2012 15:33)     Спасибо, огромное!  До этого не понимал чем могут быть полезны класс модули.. Теперь понимаю что они лишают такооого гемороя!))  Я давно искал способ задания событий для массива контролов! |  |  | | --- | | **0**  http://s5.ucoz.net/img/icon/thumbu_.png http://s5.ucoz.net/img/icon/thumbd_.png  **[5](http://www.excelworld.ru/publ/vba/class_module/class_module/42-1-0-110" \l "ent261)**    **Ещё один Сергей.**   (17.07.2013 08:36)     Супер! Спасибо за труды. Очень полезно и доходчиво. Будем учиться! |  |  | | --- | | **0**  http://s5.ucoz.net/img/icon/thumbu_.png http://s5.ucoz.net/img/icon/thumbd_.png  **[6](http://www.excelworld.ru/publ/vba/class_module/class_module/42-1-0-110" \l "ent353)**    **[SkyPro](javascript://)**   (08.10.2013 23:09)  [[http://www.excelworld.ru/avatar/73/341553.jpg](javascript://)](javascript://)   Очень полезная статья.  Осталось только усвоить прочитанное :) |  |  | | --- | | **0**  http://s5.ucoz.net/img/icon/thumbu_.png http://s5.ucoz.net/img/icon/thumbd_.png  **[7](http://www.excelworld.ru/publ/vba/class_module/class_module/42-1-0-110" \l "ent442)**    **ratboy**   (11.02.2014 22:09)     Не много. Больше на уяснения понятий, что такое нейросети и как их едят... |  |  | | --- | | **0**  http://s5.ucoz.net/img/icon/thumbu_.png http://s5.ucoz.net/img/icon/thumbd_.png  **[8](http://www.excelworld.ru/publ/vba/class_module/class_module/42-1-0-110" \l "ent743)**    **Alex**   (17.01.2015 13:32)     Спасибо за работу. Много интересного выделил для себя. | | |
|  | |
|  | |

Добавлять комментарии могут только зарегистрированные пользователи.