Лекция № 4

Константы.

Операции и знаки операций.

Массивы, инициализаторы коллекций и перечисления

Вопросы лекции:

1. Константы, правила их написания.
2. Операции, знаки операций и их приоритет.
3. Массивы, виды массивов.
4. Инициализаторы коллекций.
5. Перечисления
6. Разновидности процедур
7. Передача параметров
8. Лямбда-выражение

Константы

В отличие от переменной значение константы никогда не изменяется в процессе выполнения программ.

-34.07	числовая константа
3.7E+6	числовая константа
"Чтение и запись данных«	символьная константа
#12/06/2011#	константа даты
False	логическая константа

Visual Basic содержит большое количество встроенных констант для всех возможных случаев: цвета, клавиши, сообщения и т. п. Они имеют префикс **vb**. Для поиска констант определённой категории используется **Обозреватель объектов** (Object Browser), открываемый одноимённой кнопкой на панели инструментов или клавишей **F2**.

Константы называют **литеральными константами** (literal constants), если их абсолютное значение записывается **непосредственно** в программный код. **Это - режим неявного объявления**.

Кроме литеральных констант, VB позволяет создавать **именованные константы** (named constants). Такая константа, подобно переменной, имеет конкретное заданное ей **имя** и объявляется оператором **Const**. Присвоение значения именованной константе проводится также в строке объявления.

Константы

Синтаксис создания именованной константы:

Const имяКонстанты [As типДанных] = выражение

Как и в случае с литеральной константой, единственным способом изменить значение именованной константы является редактирование программного кода.

Именованные константы можно использовать для повышения **читабельности** сложных, трудно запоминаемых процедур или значений, которые трудно понять, а также для более простого обновления и сопровождения процедур и программ.

Если использовать константу как литеральную, то может оказаться затруднительным изменение программы, т. к. придется менять все **многократные вхождения** этой константы. При именованной константе достаточно изменить значение константы в **одном** месте, а именно, в операторе, объявляющем именованную константу.

Область действия констант определяется теми же правилами, что и переменных. При объявлении константы на модульном уровне можно дополнительно указать область её действия:

[Public или Private] Const имяКонстанты [As типДанных] = выражение

По умолчанию компилятор VB устанавливает режим **неявного** объявления констант

Операции и знаки операций

В VB используются следующие операции:

- арифметические;
- конкатенация;
- сравнения;
- логические.

Арифметические операции (математические операторы) - это всего 7 знаков: помимо 4-х основных: **+; -; *;** / имеются еще три дополнительных математических оператора:

^ - возведение в степень;

\ - целочисленное деление;

mod - первое значение тоже делится на второе, но в качестве результата получаем остаток от деления как целое число.

5.1\3 = 1 **37\7 = 5**

5.1 mod 3 = 2 **37 mod 7 = 2**

Если теперь выполнить операцию деления модуля на 7, то можно получить результат вычисления в дробной форме:

37 Mod 7 / 7 = 2/7 **или** **37 Mod 7 / 7 = 0,286.**

Конкатенация или сцепление строк - используется оператор **&**.

Операции и знаки операций

Сравнение значений (6 операций для сравнения числовых и строковых значений и 2 специальные операции Like (для строк) и Is (для объектов).

Результатом вычисления выражения, основанного на операциях сравнения всегда является True или False.

Операторы:

= (равенство); < (меньше, чем); <= (меньше, чем или равно);
> (больше, чем); >= (больше, чем или равно); <> (не равно).

Оператор **Like** означает подобие. Оба операнда должны быть значениями **String**. Если E1 совпадает с образцом, содержащимся в E2, то **True**.

Оператор **Is.** Если E1 ссылается на тот же самый объект, что и E2, то **True.**

Логические операции:

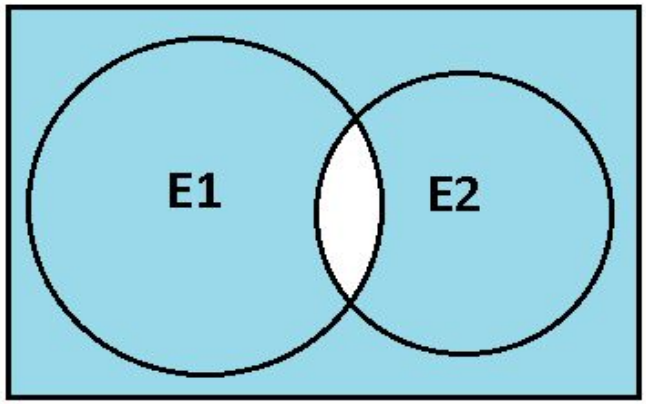
AND (конъюнкция: E1 And E2). Если имеют значение и E1, и E2 , то **True**;

OR (дизъюнкция: E1 Or E2). Если имеется E1 или E2 или оба, то **True**;

NOT (отрицание: Not E1). Если E1 имеет значение, то False, если E1 отсутствует, то результат **– True**;

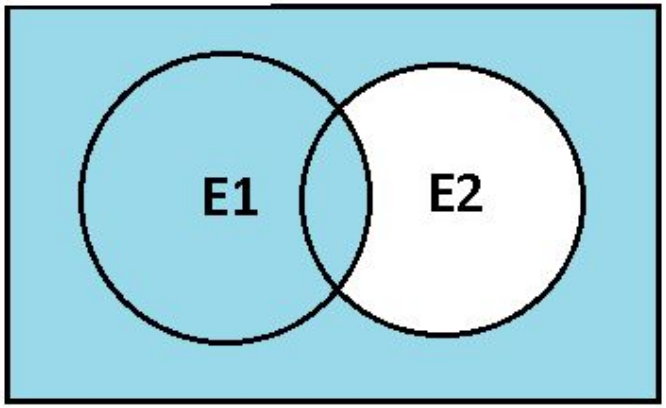
XOR (исключающее ИЛИ**):** E1 Xor E2. Если есть только E1 или только E2, то результат **- True**;

Операции и знаки операций



E1	E2	E1 Xor E2
1	1	0
1	0	1
0	0	1
0	1	1

Imp (импликация: E1 Imp E2). Если E1 – **False**, а E2 – **True**, то **False**. В остальных случаях результат **- True.** Эта операция называется операцией вовлечения и трактуется, как "если …, то…". Пример: "Если у фигуры все стороны равны, то это – квадрат".



E1	E2	E1 Imp E2	
1	1	1	Верно для квадрата
1	0	1	Верно для ромба
0	0	1	Верно для любой фигуры
0	1	0	Неверно

Приоритет операций

1. В первую очередь вычисляется выражение в круглых скобках.
2. Если скобки отсутствуют, то по умолчанию выполняются следующие категории операций:
3. sin(), cos(), abs()…;
4. ^;
5. отрицание (присвоение числу отрицательного значения);
6. *, /;
7. \ (целочисленное деление);
8. mod;
9. +, -;
10. & (конкатенация или сцепление строк);
11. операции сравнения;
12. логические операции.

Массивы

Массивы (класс Array) – это упорядоченные наборы переменных одного типа, доступ к которым осуществляется с помощью порядковых номеров, называемых **индексами**.

Каждая такая переменная называется **элементом массива**, а количество элементов в массиве называется **размером** массива. Размер массива ограничивается объёмом оперативной памяти и типом данных элементов массива.

Индекс элемента указывается в круглых скобках после имени массива, например, **strИмена(1), strИмена (3), strИмена (10)** являются элементами массива strИмена. Каждый из элементов можно использовать также как простую переменную.

Различаются **фиксированного** размера (статические) и **динамические** массивы.

Границы фиксированного массива устанавливаются при его объявлении, и при выполнении программы изменяться не могут.

Массивы

Объявление фиксированного массива зависит от области его видимости и осуществляется:

- **глобального массива** – с помощью оператора Public в разделе объявлений (Declaration) модуля;
- **массива модульного уровня** – с помощью оператора Private в разделе объявлений (Declaration) модуля;
- **локального массива** – с помощью оператора Dim внутри процедуры.

При объявлении фиксированного массива после его имени в круглых скобках указывается **верхняя граница** массива. Нижней границей массива всегда является нуль.

Например, нужно хранить фамилии 150 участников конференции.

Для этого объявляется одномерный фиксированный массив:

Dim strФамилия(149) As String

Для создания того же, но глобального массива нужно использовать оператор Public:

Public strФамилия(149) As String

Какой-либо, в частности восьмой элемент одномерного массива strФамилия (149) может быть обозначен:

strФамилия(7) = "Иванов"

Массивы

Массивы могут быть многомерными с размерностью - до 60 измерений (координат). Пример объявления **четырехмерного** массива типа Integer:

Dim intКомплектующие(34, 13, 4, 29) As Integer

При этом массив содержит: 35x14x5x30 = 73500 элементов.

При объявлении **динамического** массива его размер (в круглых скобках) не указывается, но может изменяться в процессе выполнения программы в соответствии с конкретными условиями.

Применение динамических массивов обеспечивает эффективное управление памятью, выделяя память под массив лишь на время, когда массив используется, а затем освобождая её. Например:

Dim datДниРожд( ) As Date

Для того чтобы в какой-то момент активизировать динамический массив, чтобы он получил **возможность** хранения данных, следует указать его размер с помощью оператора **ReDim**:

ReDim datДниРожд(intЧислоДнейРожд – 1)

т. е. предполагается, что значение переменной (intЧислоДнейРожд) уже задано ранее, а уменьшение числа элементов на 1 приводит число элементов массива в соответствие с их нумерацией **от нуля**. При выполнении оператора ReDim данные, размещенные в массиве ранее, стираются.

Массивы

Инициализация любого массива осуществляется поэлементно так же, как и переменной с помощью оператора присваивания. Но это можно сделать и при объявлении, поместив значения массива в фигурные скобки:

Dim strNames( ) As String ={"Андрей", "Владимир", "Иван"}

Dim intСчёт(,) As Integer = {{1, 2}, {3, 4}}

Копирование содержимого одного массива в другой, это делается с помощью оператора инициализации:

strМассивДругой() = strИсходныйМассив(…)

Массив, которому присваивается значение (МассивДругой), должен быть **динамическим**. Размеры его устанавливаются автоматически после присваивания.

Типы данных исходного массива и другого обязаны быть **совместимыми.**

С массивами производятся статистические функции, т. е. могут вычисляться итоговые функции, например:

Count – определение количества чисел в массиве;

Sum – суммирование элементов массива и т. п.

Массивы создаются на основе класса Array, который осуществляет следующие методы:

BinaryStarch – обеспечивает поиск в отсортированном массиве. Когда элемент найден, то возвращается его индекс;

Reserve – изменяет порядок следования элементов одномерного массива на обратный;

Sort – сортирует элементы одномерного массива по порядку и т. п.

Инициализаторы коллекций

Инициализатор коллекции предоставляет собой **сокращённый** синтаксис, позволяющий создать **литеральный массив** и заполнить его начальным набором значений. Такие конструкции полезны, когда необходимо создать список (последовательность) известных имён, состояний или числовых значений, которые используются для проверки правильности.

Инициализатор коллекции состоит из разделённого запятыми списка значений, заключённого в фигурные скобки. следуют фигурные скобки.

Примеры создания коллекций "Месяцы", "Номера", "Меню«:

```
Dim Месяцы() As String = {"Январь", "Февраль", "Март", "Апрель"}
Dim Номера() As Integer = {25, 37, 89, 66, 88, 101}
Dim Меню() As String = {"Дом", "Продукты", "Новости", "Контакты"}
```

Счёт элементов коллекции начинается с нулевого номера. Для вызова какого-либо элемента можно использовать процедуру с окном сообщений, в частности:

```
MsgBox(Месяцы(2))
MsgBox(Номера(1))
MsgBox(Меню(3))
```

Будут визуализироваться соответственно элементы "Март", 37 и "Контакты".

Перечисления

Перечисления представляют собой список взаимосвязанных констант. Для объявления перечислений используется оператор Enum … End Enum.

Перечисления могут быть объявлены только в разделе объявлений класса или модуля и могут иметь только целочисленные типы данных, например, Integer, Long и т. п. Если тип данных не указан, и элементам перечисления не присваиваются конкретные значения, то по умолчанию используется тип Integer, и счёт элементов идёт от нулевого значения. В ином случае конкретно указывается номер первого элемента.

Пример объявления перечисления:

```
Public Enum Дни_недели As Integer
Понедельник = 1 : Вторник : Среда : Четверг : Пятница : Суббота : Воскресенье
    End Enum
```

Для визуализации конкретного элемента используется процедура, например, нажатие кнопки:

```
Private Sub Button1_Click()
        Dim Статус As Дни_недели  =  Дни_недели.Четверг
        MsgBox(Статус & "-й день недели")
End Sub
```

Высвечивается сообщение: "4-й день недели".

Перечисления могут быть объявлены **только** в разделе объявлений класса или модуля, но не в процедуре.

Разновидности процедур

В общем случае в Visual Basic существуют следующие виды процедур:

- Sub;
- Function;
- Property.

Процедура типа Sub

Процедура типа Sub не возвращает значения и наиболее часто используется для обработки какого-либо события. Её можно помещать в стандартные модули, модули форм и классов. Она имеет следующий синтаксис:

```
[Уровень доступности] Sub ИмяПроцедуры(Аргументы)
    Операторы
End Sub
```

С помощью параметра **Уровень доступности** (аналогично области видимости для переменной) указывается, доступна ли процедура другим частям программы.

Разновидности процедур

Уровень доступности может принимать следующие значения:

Public – процедура доступна в проекте, в котором определена;

Private – процедура доступна только в том классе или модуле, в котором она определена;

Protected – такие (защищённые) процедуры доступны внутри класса, в котором они объявлены, а также в производных от данного класса;

Friend – такие (дружественные) процедуры доступны только внутри той **сборки**, в которой они объявлены.

Сборка – полностью самостоятельная единица приложения, которая обычно соответствует всей программе, поэтому данный модификатор можно воспринимать как указание видимости в **пределах программы**;

Protected Friend – доступность процедуры расширяется не только на сборку, но и на производные классы.

Параметр **Аргументы** процедуры используется для объявления передаваемых в процедуру переменных.

Процедуры типа Sub разделяются на процедуры **обработки событий** и **общие** процедуры.

Разновидности процедур

Процедуры обработки событий (event procedure) связаны с объектами, размещёнными в формах VB. Или с самой формой и выполняются при возникновении **события**, с которым они связаны. Для события, связанного с формой, процедура обработки события Sub имеет следующий синтаксис:

Private Sub ИмяФормы_ИмяСобытия(аргументы) Handles ИмяСобытия

Операторы

End Sub

Например, имя процедуры загрузки формы Form1 будет выглядеть, как Form1_Load( ), а для щелчка на форме - Form1_Click( ). Эти названия создаются автоматически средой разработки в шаблоне процедуры. Имя формы можно изменять в окне Свойства (Properties) путём изменения значения свойства Name.

Аналогично создаются шаблоны процедур обработки события **для любого элемента управления** формы. Внутри процедур помещаются операторы программного кода, создаваемые разработчиком, а в конце – оператор окончания процедуры End Sub.

Разновидности процедур

Общие процедуры – это серия операторов VB между ключевыми словами Sub и End Sub. При вызове процедуры эти операторы выполняются до тех пор, пока не встретится ключевое слово End Sub или Exit Sub.

В общем случае процедуры могут вызываться как другими процедурами, так и по своему имени, например, макрос VBA, который вызывается по имени либо из приложения (Word, Excel), либо из редактора VBA.

Процедуры Sub выполняют определённые действия, но **не возвращают значения**. Они могут содержать аргументы, объявляющие входящие в процедуру переменные, константы, выражения.

Процедура типа Sub может быть объявлена внутри модуля, класса или структуры. По умолчанию (при отсутствии оператора Public) она имеет **общий доступ.** Для ограничения уровня доступности используются ключевые слова Private или Static.

Вызов процедуры Sub имеет следующий синтаксис:

[Уровень доступности] ИмяПроцедуры(аргумент1, аргумент2,… аргументN),

При этом ключевое слово, обозначающее область видимости, не является обязательным. При вызове процедуры из другого модуля программы необходимо указать адресную ссылку на имя модуля, содержащего процедуру, например, при вызове из **Form1**:

[Уровень доступности] Form1.ИмяПроцедуры(аргумент1, аргумент2,… аргументN)

Разновидности процедур

Процедура типа Function

Процедуры Function в отличие от процедур Sub могут **возвращать значение** в вызывающую процедуру. Синтаксис Function имеет следующий вид:

```
[Уровень доступности] Function ИмяПроцедуры(Аргументы) [As Тип]
    Операторы
End Function
```

В качестве уровня доступности может быть указано: Public, Private, Protected, Friend, Protected Friend. Процедуры Function, как и переменные, имеют тип, задаваемый ключевым словом As. Если тип процедуры не задан, то по умолчанию ей присваивается тип Object. Тип процедуры определяет тип возвращаемого ею значения.

Возвращаемым значением называется значение, которое функция передаёт обратно в вызвавшую её программу. Функция может вернуть значение двумя способами:

- значение присваивается самому имени функции один или несколько раз в процессе выполнения процедуры. Управление и, соответственно, возвращаемое значение, не будет передано в программу, вызвавшую функцию до тех пор, пока не выполнится End Function или Exit Function;
- использованием оператора Return (возврат), чтобы определить возвращаемое значение с немедленной передачей управления программе, вызвавшей функцию.

Разновидности процедур

Преимущество первого способа состоит в том, что имени функции можно присвоить предварительное значение, которое затем в процессе выполнения процедуры легко изменить.

Если функция возвращает массив, то внутри этой функции невозможен доступ к отдельным элементам массива.

Рассматривается процедура, вычисляющая площадь квадрата и обеспечивающая вызов функции:

```
Module Module1
    Function Площадь(ByVal A As Single) As Single
        Площадь = A ^ 2
    End Function
    Sub Main()
        Dim Квадрат As Single
        Квадрат = Площадь(400)
        MsgBox("Площадь квадрата" & " = " & Квадрат)
    End Sub
End Module
```

Для вызова функции используется процедура Sub Main().

Передача параметров

Переменные, передаваемые процедуре, называются **параметрами процедуры.** По умолчанию они имеют тип **Object**. VB обеспечивает задание типа параметров с помощью ключевого слова **As**.

Передача параметров может задаваться двумя способами:

- по значению (**By value**);
- по ссылке (**By reference**).

В случае передачи в процедуру по значению (оператор ByVal) в качестве переменной передаётся не сама переменная, а её строковое **значение**.

При передаче параметров по ссылке (оператор ByRef) процедура получает доступ к области памяти, в которой эта переменная хранится, в результате чего при изменении **параметра** в процедуре происходит **возвращение** прежнего значения переменной, например:

```
Module Module1
    Sub Тест(ByVal strA As String, ByRef strB As String)
        strA = "Строка по значению"
        strB = "Строка по ссылке"
    End Sub
```

Передача параметров

```
Sub Main()
        Dim strA As String
        strA = "Привет"
        Dim strB As String
        strB = "Здравствуй, мир!"
        MsgBox(strA & ", " & strB) 'визуализируются обе локальные переменные
        Тест(strA, strB)
        MsgBox(strA & ", " & strB) 'визуализируется локальная переменная по значению  'и модульная переменная - по ссылке
    End Sub
End Module
```

Переменная strA, являющаяся параметром модульной процедуры Тест, передаётся по значению (ByVal), а переменная strB передаётся по ссылке (ByRef).

При выполнении процедуры Sub Main() используются локальные переменные, имеющие те же имена, что и модульные переменные, т. е. в окне сообщения появляется текст: "Привет, Здравствуй, мир!".

Передача параметров

Когда же выполняется переменная Тест(strA, strB), то переменная strA, передаваемая по значению, будет инициализироваться, как и указано: по локальному значению, присвоенному в процедуре Sub_Main(), а переменная strB локального значения не принимает, т. к. имеет **ссылку** на значение, присвоенное в модуле. Таким образом, в случае переменной strB **возвращается** прежнее значение, и в окне сообщения появляется текст: "Привет, Строка по ссылке".

Объявление и присвоение переменной значения (как и для константы) может осуществляться в одной строке, например, для предыдущего кода:

```
Dim refstrB As String = "Здравствуй, мир!"
```

Это обеспечивает большую компактность и ясность кода.

Лямбда-выражение

Лямбда-выражение вводится с целью большей компактности кода при передаче значения, возвращаемого функцией. Выражение представляет собой некоторую безымянную функцию Function(x). При применении выражения используются различные варианты записи функции, например:

```
Dim x As Integer
    Private Sub Button1_Click()
        Dim A = Function(x) x * 20
        MsgBox(A(5))       'результат: 5 * 20 = 100
        Console.WriteLine(A(15))       'результат - в окне "Вывод": 15 * 20 = 300
    End Sub

    Private Sub Button2_Click()
        Dim B = Function(x)
                Return x * 10 'лямбда-выражение используется с функцией Return (возврат)
                              'оператор Function(x) опускается
            End Function
        MsgBox(B(5))          'результат: 5 * 10 = 50
        Debug.WriteLine(B(15)) 'результат - в окне "Интерпретация": 15 * 10 = 150
    End Sub
```

Лямбда-выражение

```
Private Sub Button3_Click()
    Dim C = Function(x As Integer) x ^ 4
    MsgBox(C(5)) 'результат: x ^ 4 = 625
    Console.WriteLine(C(2)) 'результат - в окне "Вывод": x ^ 4 = 16,
End Sub
```

В третьем варианте объявление переменной x осуществляется не в разделе объявлений модуля, а непосредственно в процедуре, что обеспечивает ещё большую компактность кода.