Простые FAQ (ЧАВО).

Для тех кто не знает что такое FAQ - поясняю, - это Часто Задаваемые Вопросы или ЧАВО (FAQ англ. Frequently Asked Questions).

Вопросы:

1. Как сохранить текст в файл?
2. Как прочитать текст из файла?
3. Как добавить, удалить элемент в списке (listbox)?
4. Как использовать элементы DirListBox, FileListBox, DriveListBox ?
5. Каким образом можно узнать какой элемент выбран в списке?
6. Как программно загрузить картинку в PictureBox или Image?
7. Можно ли в VB проиграть wav файл?
8. Как распечатать текст на принтере?
9. Как можно узнать текущее разрешение экрана?
10. Можно ли узнать где находится exe файл запущенной программы?
11. Как загрузить дополнительный элемент управления (.ocx)?

Ответы:

1. Как сохранить текст в файл?

Для сохранения текста в файл используйте такой код:

```
Open ПУТЬ_К_ФАЙЛУ for Output as #X
Print #X, ТЕКСТ_КОТОРЫЙ_НУЖНО_СОХРАНИТЬ
Close #X
```

Здесь всесто X нужно поставить номер (от 1 до 255). Также можно использовать функцию FreeFile.

Пример:

```
Open "C:\Temp\VB.txt" for Output as #1
Print #1, "Visual Basic - самый лучший язык программирования!"
Close #1
```

2. Как прочитать текст из файла?

Для чтения текста из файла используйте такой код:

```
Open ПУТЬ_К_ФАЙЛУ for Input as #X
Input #X, StrPerem
Close #X
```

Представленная процедура прочитает первую строчку из файла и запишет в переменную StrPerem. Для чтения всего текста, как правило, используется такой код:

```
Open ПУТЬ_К_ФАЙЛУ for Input as #X
While not(EOF (X))
Input #X, StrTemp
StrPerem = StrPerem & StrTemp
Loop
Close #X
```

Здесь всесто X нужно поставить номер (от 1 до 255). Также можно использовать функцию FreeFile.

3. Как добавить, удалить элемент в списке (listbox)?

Для добавления элемента в список используется метод AddItem.

Пример:

```
List1.AddItem "Меня добавили в список"
```

Для удаления используется метод RemoveItem с номером элемента, который нужно удалить.

Пример (удалит первый элемент в списке):

```
List1.RemoveItem 0
```

4. Как использовать элементы DirListBox, FileListBox, DriveListBox ?

Перечисленные элементы используются для навигации по диску. Например, для создания формы, в которой будет выбираться файл следует:

Создать все 3 элемента. Затем обработать событие Change у DirListBox, вписав в него такой вот код:

```
File1.Path = Dir1.Path
```

Теперь обработайте событие Change у DriveListBox:

```
Dir1.Path = Drive1.Drive
```

Вот и всё! Теперь запустите проект и походите по каталогам и дискам. Кстати имя текущего выбранного файла в FileListBox содержится в св-ве filename.

5. Каким образом можно узнать какой элемент выбран в списке?

Используйте свойство ListIndex.

Пример:

```
Form1.Caption = List1.ListIndex
```

6. Как программно загрузить картинку в PictureBox или Image?

Делается это так: `Obj.Picture = LoadPicture (ПУТЬ_К_ФАЙЛУ)`

Где Obj - имя PictureBox, Image или формы.

Пример:

```
Picture1.Picture = LoadPicture ("C:\Images\vblogo.jpg")
```

7. Можно ли в VB проиграть wav файл?

Можно! В VB всё можно! :) С помощью API ф-ции sndPlaySound из библиотеки winmm.dll (windows multimedia). Вот её объявление:

```
Declare Function sndPlaySound Lib "winmm.dll" Alias "sndPlaySoundA" (ByVal
lpszSoundName As String, ByVal uFlags As Long) As Long
```

Пример:

```
sndPlaySound "C:\youpath\vavchik.wav", 1
```

8. Как распечатать текст на принтере?

В Visual Basic есть такой объект Printer. С его помощью с принтером можно делать почти всё, что угодно. Например, вам нужно распечатать текст, находящийся в текстовом поле Text1. Для этого:

```
Printer.Print Text1.Text
Printer.EndDoc
```

9. Как можно узнать текущее разрешение экрана?

Используя свойства Height и Width объекта Screen.

Пример:

```
Form1.Caption = Screen.Height
' отобразится в твипах

Form1.Caption = Screen.Height / Screen.TwipsPerPixelY
' отобразится в пикселах
```

10. Можно ли узнать где находится exe файл запущенной программы?

Можно, использовав объект App. Кстати в нём хранятся и другая полезная инфа.

Пример:

```
Form1.Caption = App.Path
' на капшне отобразится путь, где находится запущенный exe
```

11. Как загрузить дополнительный элемент управления (.ocx)?

Нажмите правой кнопкой на панели с элементами управления (она находится слева) и выберите из контекстного меню команду Components... Далее отметьте галочками подключаемые элементы и нажмите OK.

Описание операторов VB

Здесь подробно описаны все арифметические операторы, операторы сравнения, логические, а также операторы конкатенации. Я попытался собрать всё то, что необходимо знать об этих операторах. Каждый оператор снабжён примером. Описания переведены мной со стандартного Help'a VB5.

Арифметические операторы

оператор ^ (возведение в степень)

Этот оператор предназначен для возведения числа в степень.

Синтаксис:

результат = *число* ^ *степень*

Параметры:

результат - обязателен; любая числовая переменная
число - обязательно; любое числовое выражение
степень - обязателена; любое числовое выражение

Замечания:

число может быть отрицательное, только в том случае, когда *степень* - целое число. Если в одном выражении используется несколько операторов ^, то вычисление происходит слева направо. Обычно тип результата - Double. Однако, если или *степень*, или *число* - Null выражение, то *результат* тоже Null.

Пример:

```
Dim MyValue
MyValue = 2 ^ 2 ' Возвратит 4.
MyValue = 2 ^ 3 ^ 3 ' Возвратит 512 (2^3=8, 8^3=512)
MyValue = (-5) ^ 3 ' Возвратит -125.
```

Советы:

Если вам необходимо возвести число в постоянную степень, то лучше использовать несколько операторов - умножений, чем один - возведение в степеръ, судите сами, цикл в миллион проходов с вычислением выражения проходил:

```
test1 = 2 ^ 8 ' 893мс
test1 = 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 ' 130мс!
```

оператор * (умножение)

Используется для перемножения двух чисел.

Синтаксис:

*результат = число1 * число2*

Параметры:

результат - обязателен; любая числовая переменная
число1 - обязательно; любое числовое выражение
число2 - обязательно; любое числовое выражение

Замечания:

Тип данных результата обычно такой же, как и самый точный тип из двух чисел. Порядок точности, от большего к меньшему - Byte, Integer, Long, Single, Currency, Double, Decimal. Правда, есть исключения:

- Если перемножаются Single и Long, то результат - Double
- Если тип данных *результата* - Long Single или Date, в который не помещается сам результат выражения, то результат конвертируется в Variant, содержащий Double.

Если *число1* или *число2* - Null, то оно интерпретируется просто как обычный 0.

Порядок точности в умножение отличен от тех, что используются в сложении и вычитании.

Пример:

```
Dim MyValue
MyValue = 2 * 2 ' Возвратит 4.
MyValue = 459.35 * MyValue ' Возвратит 495.35 * 4
```

оператор / (деление)

Используется для деления двух чисел и получить результат с плавающей точкой.

Синтаксис:

результат = *число1* / *число2*

Параметры:

результат - обязателен; любая числовая переменная
число1 - обязательно; любое числовое выражение
число2 - обязательно; любое числовое выражение

Замечания:

Результат обычно имеет тип Double. Правда, есть исключения:

- Если оба выражения выражения имеют тип Byte, Integer, Single, то результат Single. Однако, если размеры выражения не вписываются в рамки Single, происходит ошибка.
- Если оба выражения выражения имеют тип Variant, содержащий Byte, Integer или Single, то результат Single Variant. Однако, если размеры выражения не вписываются в рамки Single, то Double Variant.
- Если одно из чисел имеет тип Decimal, то и результат - Decimal.

Если *число1* или *число2* - Null, то оно интерпретируется просто как обычный 0.

Примеры:

```
Dim MyValue
MyValue = 10 / 4 ' Возвратит 2.5.
MyValue = 10 / 3 ' Возвратит 3.333333.
```

оператор \ (целочисленное деление)

Используется для деления двух чисел и получить целочисленный результат.

Синтаксис:

результат = *число1* \ *число2*

Параметры:

результат - обязателен; любая числовая переменная
число1 - обязательно; любое числовое выражение
число2 - обязательно; любое числовое выражение

Замечания:

Перед тем, как происходит такое деление, выражения округляются до Byte, Integer или Long выражений. Обычно тип данных результата Byte, Byte variant, Integer, Integer variant, Long, or Long variant. Любая дробная часть обрезается.

Однако, если любое из выражений Null, то и результат Null. Любое выражение, содержащее Empty интерпретируется как 0.

Примеры:

```
Dim MyValue
MyValue = 11 \ 4 ' Возвратит 2.
MyValue = 9 \ 3 ' Возвратит 3.
MyValue = 100 \ 3 ' Возвратит 33.
```

оператор Mod (остаток от деления)

Используется для деления двух чисел и получение остатка от их деления.

Синтаксис:

результат = *число1* `Mod` *число2*

Параметры:

результат - обязателен; любая числовая переменная
число1 - обязательно; любое числовое выражение
число2 - обязательно; любое числовое выражение

Замечания:

При делении числа с правающей точкой округляются. Например, результат следующего выражения равен 5:

```
A = 19 Mod 6.7
```

Что здесь происходит? Сначала округляется число 6.7 до 7. Затем происходит деление, получаем 2.7.... Остаток от деления = 5. (2 * 7 = 14, 19 - 14 = 5).

Результат обычно имеет тип Byte, Byte variant, Integer, Integer variant, Long, или Variant содержащий Long.

Если любое из выражений Null, то и результат Null. Любое выражение, содержащее Empty интерпретируется как 0.

Примеры:

```
Dim MyResult
MyResult = 10 Mod 5 ' Возвратит 0.
MyResult = 10 Mod 3 ' Возвратит 1.
MyResult = 12 Mod 4.3 ' Возвратит 0.
MyResult = 12.6 Mod 5 ' Возвратит 3.
```

оператор + (сложение)

Используется для сложения двух чисел.

Синтаксис:

результат = выражение1 + выражение2

Параметры:

результат - обязателен; любая числовая переменная
выражение1 - обязательно; любое выражение
выражение2 - обязательно; любое выражение

Замечания:

Когда вы используете оператор +, вы не можете определить что произойдёт, сложение или конкатенация строк. Для конкатенации используйте оператор &, чтобы избежать недоразумений и сделать код более читабельным.

Если одно из выражений не Variant, то применяются следующие правила:

- Если оба выражения имеют численный тип (Byte, Boolean, Integer, Long, Single, Double, Date, Currency, или Decimal) - то происходит их сложение.
- Если оба выражения строки - конкатенация.
- Если одно из выражений имеет численный тип, а другое любое Variant значение, включая Null, то происходит сложение.
- Если одно из выражений строка, а другое любое Variant значение, то происходит конкатенация.
- Если одно из выражений содержит Empty, то возвращается второе, не изменённое выражение.
- Если одно из выражений имеет численный тип, а другое - строка, происходит ошибка несовпадения типов (Type mismatch).
- Если любое выражение Null - тогда и результат тоже Null.

Если оба выражения Variant, то применяются следующие правила:

- Если оба выражения числа - то они складываются.
- Если оба выражения строки - то они конкатенируются.
- Если одно из выражений число, а другое строка - происходит сложение.

Для обычного сложения тип данных результата обычно такой же, как и самый точный тип из двух чисел. Порядок точности следующий - Byte, Integer, Long, Single, Double, Currency, и Decimal. Есть исключения:

- Если складываются Single и Long, то результат - Double
- Если складываются выражение с типом Date, с любым другим выражением, то результат - Date.

Если одно или оба выражения Null, то результат тоже Null. Если оба из выражений содержат Empty, результат Integer. Если только одно, то в качестве результата возвращается не изменённое второе выражение.

Порядок точности в сложении и вычитании отличен от тех, что используются в умножении.

Примеры:

```
Dim MyNumber, Var1, Var2
MyNumber = 2 + 2 ' Возвратит 4.
MyNumber = 4257.04 + 98112 ' Возвратит 102369.04.

Var1 = "34"
Var2 = 6 ' Инициализируем смешанные переменные
MyNumber = Var1 + Var2 ' Возвратит 40.

Var1 = "34"
Var2 = "6" ' Инициализируем переменные со строками
MyNumber = Var1 + Var2 ' Возвратит "346" (произошла
' конкатенация, а не сложение!).
```

Советы:

Оператор сложение (+) можно использовать для сложения дат, т.е. переменных типа Date:

```
Dim d As Date
d = DateSerial(2002, 8, 15) ' иниализация даты 15.08.2002
d = d + 15 ' теперь d содержит дату 30.08.2002
' т.е. мы прибавили 15 дней
```

оператор - (вычитание, смена знака)

Используется для нахождения разницы между двумя числами, или, также, для изменения знака выражения.

Синтаксис:

результат = *выражение1* - *выражение2*

или

-*выражение*

Параметры:

результат - обязателен; любая числовая переменная
выражение - обязательно; любое выражение
выражение1 - обязательно; любое выражение
выражение2 - обязательно; любое выражение

Замечания:

В первом синтаксисе, оператор "-" необходим для нахождения разницы между двумя числами. Во втором синтаксисе, "-" используется для смены знака у *выражения*.

Тип данных результата обычно такой же, как и самый точный тип из двух чисел. Порядок точности следующий - Byte, Integer, Long, Single, Double, Currency, и Decimal. Есть исключения:

- Если в вычитании участвуют типы Single и Long, то результат - Double
- Если в вычитании используется выражение с типом Date,
  то результат - Date.
- Вычитание двух дат, даёт в результате Double.

Если одно или оба выражения Null, то результат тоже Null. Если одно из выражений Empty, то оно интерпретируется как 0.

Порядок точности в сложении и вычитании отличен от тех, что используются в умножении.

Примеры:

```
Dim MyResult
MyResult = 4 - 2 ' Возвратит 2.
MyResult = 459.35 - 334.90 ' Возвратит 124.45.
```

Советы:

Как и оператор сложение, оператор вычитание может быть применён для вычисления разницы (в днях) между двумя датами:

```
Dim d1 As Date
Dim d2 As Date
Dim razn As Long
d1 = DateSerial(1983, 10, 14)
d2 = DateSerial(2002, 8, 15)
razn = d2 - d1 ' разница в днях (6880).
```

Операторы сравнения

Используются для сравнения некоторых выражений. Имеют 3 синтаксиса:

Синтаксис:

результат = выражение1 операторсравнения выражение2
результат = объект1 Is *объект2*
результат = строка `Like` *образец*

Параметры:

результат обязателен; любая численная переменная
выражение обязательно; любое выражене
операторсравнения обязателен; любой оператор сравнения
объект обязателен; имя любого объекта
строка обязательна; любое строковое выражене.
образец обязателен; любое строковое выражение, или диапазон букв и цифр

Замечания:

Следующая таблица содержит список операторов сравнения и условия, по которым определяется результат выражения (True или False).

Оператор	True, если	False, если	Null, если
< (меньше чем)	*выражение1* < *выражение2*	*выражение1* >= *выражение2*	одно из выражений содержит Null
<= (меньше или равно)	*выражение1* <= *выражение2*	*выражение1* > *выражение2*	
> (больше чем)	*выражение1* > *выражение2*	*выражение1* <= *выражение2*	
>= (больше или равно)	*выражение1* >= *выражение2*	*выражение1* < *выражение2*	
= (равно)	*выражение1* = *выражение2*	*выражение1* <> *выражение2*	
<> (не равно)	*выражение1* <> *выражение2*	*выражение1* = *выражение2*	

Операторы Is и Like выполняют специфические функции, и их таблица сравнения отличается от приведённой (их мы рассмотрим ниже).

Когда сравниваются два выражения, не всегда можно определить, что будет сравниваться, числа или строки. Ниже показано, как будет вычислятся результат, если оба выражения имею тип, отличный от Variant:

- Если оба выражения числа (Byte, Boolean, Integer, Long, Single, Double, Date, Currency, или Decimal), то происходит сравнение чисел.
- Если оба выражения строки, то происходит сравнение строк. (меньшая строка та, первая и последующие буквы которой имею меньший ascii код).
- Если одно из выражений число, а другое Variant, который может быть трактован как число, то происходит сравнение чисел.
- Если одно из выражений число, а другое Variant строка, которая не может быть трактована как число, то происходит ошибка (Type mismatch).
- Если одно из выражений строка, а другое любое Variant значение (даже Null), то происходит строковое сравнение.
- Если одно из выражений Empty, а другое число, то происходит сравнение чисел, где Empty рассматривается как 0.
- Если одно из выражений Empty, а другое строка, то происходит сравнение строк, где Empty рассматривается как пустая строка "".

Если и первое выражение и второе имеют тип Variant, то выражения сравниваются, согласно тем типам данных, которые содержит Variant:

- Если оба Variant выражения содержат числа, то происходит сравнение чисел.
- Если оба Variant выражения содержат строки, то происходит сравнение строк.
- Если одно из Variant выражений содержит число, а другое строку, то числовое выражение меньше строкового.
- Если одно из Variant выражений Empty, а другое число, то Empty рассматривается как 0.
- Если одно из Variant выражений Empty, а другое строка, то Empty рассматривается как пустая строка "".
- Если оба выражения Empty, то они рассматриваются как равные.

Когда Single переменная сравнивается с Double, то Double округляется до точности Single.

Если Currency сравнивается с Single или Double, то Single или Double конвертируются в Currency. Точно так же, при сравнении Decimal с Single или Double, то Single или Double конвертируются в Decimal. Для Currency любая дробная часть меньшая, чем .0001, может быть утеряна. Для Decimal это значение 1E-28, или может произойти ошибка. Таким образом, при потере дробной части, выражения могут интерпретироваться как равные, хотя на самом деле, одно от другого будет отличаться. (хоть и на маленькое значение).

Примеры:

```
Dim MyResult, Var1, Var2
MyResult = (45 < 35) ' Возвратит False.
MyResult = (45 = 45) ' Возвратит True.
MyResult = (4 <> 3) ' Возвратит True.
MyResult = ("5" > "4") ' Возвратит True.

Var1 = "5": Var2 = 4 ' в VB можно использовать двоеточие,
' для разделения операторов.
MyResult = (Var1 > Var2) ' Возвратит True.

Var1 = 5: Var2 = Empty
MyResult = (Var1 > Var2) ' Возвратит True.

Var1 = 0: Var2 = Empty
MyResult = (Var1 = Var2) ' Возвратит True.
```

оператор сравнения Is

Этот оператор используется для сравнения объктных переменных.

Синтаксис этого оператора приведён выше.

Замечания:

Если объект1 и объект1 ссылаются на один и тот же объект, то результат - True, если нет, то False. Две переменные могут ссылается на один и тот же объект несколькими путями. В следующем примере, A ссылается на тот же объект, что и B:

```
Set A = B
```

Следующий пример делает так, что переменные A и B ссылаются на один и тот же объект - C:

```
Set A = C
Set B = C
```

Примеры:

```
Dim MyObject, YourObject, ThisObject, _
OtherObject, ThatObject, MyCheck

Set YourObject = MyObject ' создаём ссылки на объекты
Set ThisObject = MyObject
Set ThatObject = OtherObject
MyCheck = YourObject Is ThisObject ' Возвратит True.
MyCheck = ThatObject Is ThisObject ' Возвратит False.
' Предполагаем, что MyObject <> OtherObject
MyCheck = MyObject Is ThatObject ' Возвратит False.
```

оператор сравнения строк - Like

Оператор сравнения строк Like используется для сравнения строк.

Синтаксис этого оператора уже рассмотрен выше.

Замечание:

Этот оператор можно использовать для проверки строки String на маску Pattern. Это очень мощный оператор, почти аналог регулярных выражений в Perl.

Итак, работает этот ператор следующим образом. Если строка подходит под маску, то результат True. Если нет - False. Если одно из выражений Null - результат тоже Null.

Поведение оператора Like зависит от установленного по умолчанию типа сравнения строк. (оператор Option Compare).

Если установлен тип Binary (т.е. двоичное сравнение), то строки сравниваются согласно их Ascii кодам (в разных кодировках она разная). Обычно используется такая последовательность:

A < B < E < Z < a < b < e < z < А < К < Я < а < к < я

Если установлен тип Text (текстовое сравнение). При таком сравнении последовательность отличается от предыдущей, здесь большие и маленькие буквы - равны:

(A=a) < (А=а) < (B=b) < (E=e) < (К=к) < (Z=z) < (Я=я)

Самое главная функция оператора Like - это проверка на принадлежность строки какой-нибудь маске. В маске можно использовать следующие спец. символы:

? Любой отдельный символ
* Ноли или более символов
Любая цифра (0–9).
[charlist] Любой отдельный символ, попадающий в список charlist
[!charlist] Любой отдельный символ, непопадающий в список charlist

Здесь небольшое замечание. Для того чтобы проверить принадлежность строки на маске, содержащую спец. симолы (т.е. проверить, к примеру, есть ли в строке симолы [,?,#,],*), то нужно заключить из в квадратные скобки []. Просто так ставить отдельную скобку [или], нельзя.

При указании списка символов, можно использовать тире (–). Например, чтобы задать последовательность от A до Z, нужно использовать маску [A-Z]. Всё, что находится в скобках не должно содержать никаких разделителей (пробелов, запятых и т.д.), иначе они тоже будут включены в последовательность.

Есть и другие важные правила при проверки по маске:

- (!) знак в начале списка символов говорит о том, что нужно искать символы, не входящие в этот список. Если вам необходимо найти сам знак !, то нужно поставить скобки [!].
- (–) используется для задания диапазона символов.
- Когда задаётся диапазон символов, то он должен быть возрастающим по ASCII кодам. Т.е. [A-Z] правильная маска, а [Z-A] нет.
- Последовательность [] интерпретируется как пустая строка "".

Примеры:

```
Dim MyCheck
MyCheck = "aBBBa" Like "a*a" ' Возвратит True.
MyCheck = "F" Like "[A-Z]" ' Возвратит True.
MyCheck = "F" Like "[!A-Z]" ' Возвратит False.
MyCheck = "a2a" Like "a#a" ' Возвратит True.
MyCheck = "aM5b" Like "a[L-P]#[!c-e]" ' Возвратит True.
MyCheck = "BAT123khg" Like "B?T*" ' Возвратит True.
MyCheck = "CAT123khg" Like "B?T*" ' Возвратит False.

myString = "312T-87GD-8922"

If myString Like "###[A-Z]-##[A-Z][A-Z]-####" Then ...
```

Операторы конкатенации строк

Вообще-то, чтобы соединить строки в Visual Basic, можно использовать всего 2 оператора. Это & и +. Оператор + описан выше. Поговорим об операторе &.

оператор конкатенации строк - &

Используется для конкатенации двух выражений.

Синтаксис:

результат = выражение1 & выражение2

результат обязателен; Любая String или Variant переменная
выражение1 обязательно; Любое выражение
выражение2 обязательно; Любое выражение

Замечания:

Если в выражение не строка, то она конвертируется в String Variant. Тип данных результата - String только тогда, когда оба выражения имеют тип String. Иначе результат String Variant. Если оба выражения Null, то результат тоже Null. Однако, если только одно из выражений содержит Null значение, то оно интерпретируется как пустая строка "". Empty также интерпретируется как пустая строка "".

Примеры:

```
Dim MyStr
MyStr = "Hello" & " World"
' Возвратит строку "Hello World".
MyStr = "ПроВерКА" & 123 & " ПроВерка"
' Возвратит строку "ПроВерКА 123 ПроВерка".
```

Логические операторы

Это самая интересная группа операторов. При программировании вы обязаны знать их работу и применение (причём не только на Visual Basic).

В Visual Basic их 6 штук. Рассмотрим каждый оператор подробно.

оператор And

Используется для совершения логичкого умножения над двумя выражениями.

Синтаксис:

результат = *выражение1* `And` *выражение2*

результат обязателен; Любая числовая (включая Boolean) переменная
выражение1 обязательно; Любое выражение
выражение2 обязательно; Любое выражение

Замечания:

Следующая таблица показывает как работает оператор And:

Если выражение1 =	, а выражение2 =	То результат =
True	True	True
True	False	False
True	Null	Null
False	True	False
False	False	False
False	Null	False
Null	True	Null
Null	False	False
Null	Null	Null

Оператор And также используется для проверки битов числа. Для битов оператор And работает следующим образом (смотреть слева направо)

0 0 **0**
0 1 **0**
1 0 **0**
1 1 **1**

Результат выделен жирным шрифтом.

Примеры:

```
Dim A, B, C, D, MyCheck
A = 10: B = 8: C = 6: D = Null
MyCheck = A > B And B > C ' Возвратит True.
```

```
MyCheck = B > A And B > C ' Возвратит False.
MyCheck = A > B And B > D ' Возвратит Null.
MyCheck = A And B ' Возвратит 8 (битовое сравнение).
```

Последний пример рассмотрим подробнее. Число 10 представляется в виде битов следущим образом (как тетрада, т.е. 4 бита):

1010

А число 8 вот так:

1000

В результате работы оператора And, согласно вышеприведённой таблице мы получим:

1000

Т.е. 8. Для чего мы это делали? Мы делали это для того, чтобы проверить, установлен ли четвёртый бит у числа А? Получив В, мы убедились в том, что этот бит установлен.

оператор Or

Используется для совершения логичкого сложения двух выражений.

Синтаксис:

результат = выражение1 `Or` *выражение2*

результат обязателен; Любая числовая (включая Boolean) переменная
выражение1 обязательно; Любое выражение
выражение2 обязательно; Любое выражение

Замечания:

Следующая таблица показывает как работает оператор Or:

Если выражение1 =	, а выражение2 =	То результат =
True	True	True
True	False	True
True	Null	True
False	True	True
False	False	False
False	Null	Null
Null	True	True
Null	False	Null
Null	Null	Null

Оператор Or используется для установки определённых битов числа. Для битов оператор Or работает следующим образом (смотреть слева направо)

0 0 **0**
0 1 **1**
1 0 **1**
1 1 **1**

Результат выделен жирным шрифтом.

Примеры:

```
Dim A, B, C, D, MyCheck
A = 10: B = 8: C = 6: D = Null
MyCheck = A > B Or B > C ' Возвратит True.
MyCheck = B > A Or B > C ' Возвратит True.
MyCheck = A > B Or B > D ' Возвратит True.
MyCheck = B > D Or B > A ' Возвратит Null.
MyCheck = A Or 5 ' Возвратит 15:
```

Давайте последний пример рассмотрим подробнее. Число 10 представляется в виде битов следущим образом (как тетрада, т.е. 4 бита):

1010

А число 5:

0101

В результате работы оператора Or, согласно вышеприведённой таблице мы получим:

1111

Т.е. 15. Как видите, оператор Or очень легко и удобно использовать не только в выражениях сравнения, но и для установки определённых битов числа.

оператор Xor

Используется для совершения логичкого отрицания двух выражений.

Синтаксис:

результат = *выражение1* `Xor` *выражение2*

результат обязателен; Любая числовая (включая Boolean) переменная
выражение1 обязательно; Любое выражение
выражение2 обязательно; Любое выражение

Замечания:

Следующая таблица показывает как работает оператор Xor:

Если выражение1 =	, а выражение2 =	То результат =
True	True	False
True	False	True

False	True	True
False	False	False

Оператор Xor используется для инвертирования определённых битов числа. Для битов оператор Xor работает следующим образом (смотреть слева направо)

0 0 **0**
0 1 **1**
1 0 **1**
1 1 **0**

Результат выделен жирным шрифтом. Xor отличается от Or, только тем, что когда оба бита единицы, Xor выдаёт 0. Оператор Xor интересен тем свойством, то при его двойном применении он выдаёт то же число. Это часто используют в криптографии.

Примеры:

```
Dim A, B, C, D, MyCheck
A = 10: B = 8: C = 6: D = Null
MyCheck = A > B Xor B > C ' Возвратит False.
MyCheck = B > A Xor B > C ' Возвратит True.
MyCheck = B > A Xor C > B ' Возвратит False.
MyCheck = B > D Xor A > B ' Возвратит Null.
MyCheck = A Xor B ' Возвратит 2
```

Интересным примером использования оператора Xor является обмен значениями двух численных переменны:

```
Dim a As Long, b As Long
a = 4
b = 7
a = a Xor b
b = a Xor b
a = a Xor b
```

Теперь переменная a содержит значение переменной b, и наоборот. Подробнее данный фокус описан здесь.

оператор Not

Используется для совершения логичкого инвертирования двух выражений.

Синтаксис:

результат = `Not` *выражение*

результат обязателен; Любая числовая (включая Boolean) переменная
выражение обязательно; Любое выражение

Замечания:

Следующая таблица показывает как работает оператор Not:

Если выражение =	То результат =
True	False
False	True
Null	Null

Оператор Not инвертирует все биты *выражения.* Для битов оператор Not работает следующим образом (смотреть слева направо):

0 **1**
1 **0**

Результат выделен жирным шрифтом.

Примеры:

```
Dim A, B, C, D, MyCheck
A = 10: B = 8: C = 6: D = Null
MyCheck = Not(A > B) ' Возвратит False.
MyCheck = Not(B > A) ' Возвратит True.
MyCheck = Not(C > D) ' Возвратит Null.
MyCheck = Not A ' Возвратит -11 (все биты инвертированы)
```

Рассмотрим подробнее последний пример. Число 10 представляется в виде битов следущим образом (как байт, т.е. 8 битов):

0000 1010

После инвертирования всех битов получим:

1111 0101

А это и есть -11.

оператор Eqv

Используется для совершения логической эквивалентности двух выражений.

Синтаксис:

результат = *выражение1* `Eqv` *выражение2*

результат обязателен; Любая числовая (включая Boolean) переменная
выражение1 обязательно; Любое выражение
выражение2 обязательно; Любое выражение

Замечания:

Следующая таблица показывает как работает оператор Eqv:

Если выражение1 =	, а выражение2 =	То результат =
True	True	True

True	False	False
False	True	False
False	False	True

Для битов оператор Eqv работает следующим образом (смотреть слева направо)

0 0 **1**
0 1 **0**
1 0 **0**
1 1 **1**

Примеры:

```
Dim A, B, C, D, MyCheck
A = 10: B = 8: C = 6: D = Null
MyCheck = A > B Eqv B > C ' Возвратит True.
MyCheck = B > A Eqv B > C ' Возвратит False.
MyCheck = A > B Eqv B > D ' Возвратит Null.
MyCheck = A Eqv B ' Возвратит -3
```

оператор Imp

Используется для совершения логической импликации двух выражений.

Синтаксис:

результат = выражение1 `Imp` *выражение2*

результат обязателен; Любая числовая (включая Boolean) переменная
выражение1 обязательно; Любое выражение
выражение2 обязательно; Любое выражение

Замечания:

Следующая таблица показывает как работает оператор Imp:

Если выражение1 =	**, а выражение2 =**	**То результат =**
True	True	True
True	False	False
True	Null	Null
False	True	True
False	False	True
False	Null	True
Null	True	True
Null	False	Null
Null	Null	Null

Для битов оператор Imp работает следующим образом (смотреть слева направо)

0 0 **1**
0 1 **1**
1 0 **0**
1 1 **1**

Примеры:

```
Dim A, B, C, D, MyCheck
A = 10: B = 8: C = 6: D = Null
MyCheck = A > B Imp B > C ' Возвратит True.
MyCheck = A > B Imp C > B ' Возвратит False.
MyCheck = B > A Imp C > B ' Возвратит True.
MyCheck = B > A Imp C > D ' Возвратит True.
MyCheck = C > D Imp B > A ' Возвратит Null.
MyCheck = B Imp A ' Возвратит -1
```

Описание ошибок VB5

Ниже представлена таблица всех ошибок VB5 (только без SQL и Jet ошибок) из раздела trappable errors, т.е. отлавливаемые ошибки.

№	English language	Русский язык
3	Return without GoSub	Оператор Return без GoSub
5	Invalid procedure call	Неверный вызов процедуры
6	Overflow	Переполнение
7	Out of memory	Не хватает памяти
9	Subscript out of range	Индекс массива вышел за допусти-- мые пределы
10	This array is fixed or temporarily locked	Массив зафиксирован или временно недоступен
11	Division by zero	Деление на ноль
13	Type mismatch	Неверный тип
14	Out of string space	Нет места для строки
16	Expression too complex	Выражение слишком сложное
17	Can't perform requested operation	Не могу выполнить запрашиваемую операцию
18	User interrupt occurred	Произошло прерывание по указу пользователя
20	Resume without error	Оператор Resume без ошибки
28	Out of stack space	Нет места для стека
35	Sub, Function, or Property not defined	Процедура, функция или свойство не определено
47	Too many DLL application clients	Слишком много клиентов DLL приложения
48	Error in loading DLL	Ошибка при загрузке DLL
49	Bad DLL calling convention	Неверный вызов DLL
51	Internal error	Внутренняя ошибка
52	Bad file name or number	Неверное имя файла или неверный номер
53	File not found	Файл не найден
54	Bad file mode	Неверный тип доступа к файлу
55	File already open	Файл уже открыт
57	Device I/O error	Ошибка устройства ввода вывода
58	File already exists	Файл уже существует
59	Bad record length	Неверный размер записи
61	Disk full	Диск переполнен
62	Input past end of file	Чтение файла не возможно, т.к. достигнут его конец

63	Bad record number	Неверный номер записи
67	Too many files	Слишком много файлов
68	Device unavailable	Устройство не доступно
70	Permission denied	Доступ запрещён
71	Disk not ready	Диск не готов
74	Can't rename with different drive	Нельзя переименовать файл на другой носитель (т.е. оператором Name)
75	Path/File access error	Ошибка доступа к файлу или каталогу
76	Path not found	Каталог не найден
91	Object variable or With block variable not set	Ссылка на объект или блок оператора With не установлен
92	For loop not initialized	Не найдено начало цикла
93	Invalid pattern string	Неверная маска
94	Invalid use of Null	Неверное использование Null
97	Can't call Friend procedure on an object that is not an instance of the defining class	Не могу вызывать процедуру Friend не являющуюся экземпляром класса
98	A property or method call cannot include a reference to a private object, either as an argument or as a return value (Error 98)	Обращение к свойство или методу не может включать ссылку на Private объект. Этот объект также не может быть в качестве аргумента или возвращаемого значения.
298	System DLL could not be loaded	Системная DLL не может быть загружена
320	Can't use character device names in specified file names	Не могу использовать в качестве имени файла название порта
321	Invalid file format	Неверный формат файла
322	Can’t create necessary temporary file	Не могу создать необходимый временный файл
325	Invalid format in resource file	Неверный формат файла ресурсов
327	Data value named not found	Значение не можен быть найдено
328	Illegal parameter; can't write arrays	Недопустимый параметр; не могу записать массив
335	Could not access system registry	Не могу получить доступ к реестру
336	ActiveX component not correctly registered	ActiveX компонент неправильно зарегистрирован
337	ActiveX component not found	ActiveX компонент не найден
338	ActiveX component did not run correctly	ActiveX компонент не правильно запущен (?)
360	Object already loaded	Объект уже загружен
361	Can't load or unload this object	Не могу загрузить или выгрузить объект
363	ActiveX control specified not found	ActiveX компонент не найден
364	Object was unloaded	Объект был выгружен

365	Unable to unload within this context	В данном случае невозможно выгрузить объект
368	The specified file is out of date. This program requires a later version	Данный файл устарел. Эта программа требует более новой версии
371	The specified object can't be used as an owner form for Show	Данный объект не может быть использован как родитель формы для её показа
380	Invalid property value	Неверное значение свойства
381	Invalid property-array index	Неверный индекс свойства-массива
382	Property Set can't be executed at run time	Процедура установки свойства не может быть запущена во время работы программы
383	Property Set can't be used with a read-only property	Процедура установки свойства не может быть использована для свойств, доступных только для чтения
385	Need property-array index	Необходим индекс для свойства-массива
387	Property Set not permitted	Процедура установки свойства не разрешена
393	Property Get can't be executed at run time	Процедура чтения свойства не может быть запущена во время работы программы
394	Property Get can't be executed on write-only property	Процедура чтения свойства не может быть использована для свойств, доступных только для записи
400	Form already displayed; can't show modally	Форма уже показана; не могу показать форму модально
402	Code must close topmost modal form first	Код должен сначала закрыть модальную форму самого высшего уровня
419	Permission to use object denied	Запрещено использование данного объекта
422	Property not found	Свойство не найдено
423	Property or method not found	Свойство или метод не найден
424	Object required	Необходим объект
425	Invalid object use	Неверное использование объекта
429	ActiveX component can't create object or return reference to this object	ActiveX компонент не может создать объект или возвратить ссылку на этот объект
430	Class doesn't support Automation	Класс не поддерживает технологию Automation
432	File name or class name not found during Automation operation	Имя файла или класса не найдено в процессе операции Automation
438	Object doesn't support this property or method	Объект не поддерживает данное свойство или метод

440	Automation error	Ошибка Automation
442	Connection to type library or object library for remote process has been lost	Связь с библиотекой типов или библиотекой объектов для удалённого процесса была утеряна
443	Automation object doesn't have a default value	Automation объект не имеет значения по умолчанию
445	Object doesn't support this action	Объект не поддерживает данную операцию
446	Object doesn't support named arguments	Объект не поддерживает названные аргументы
447	Object doesn't support current locale setting	Объект не поддерживает текущие локальные настройки
448	Named argument not found	Названный аргумент не найден
449	Argument not optional or invalid property assignment	Аргумент обязателен или неверное присваивание свойству
450	Wrong number of arguments or invalid property assignment	Неверное количество аргументов или неверное присваивание свойству
451	Object not a collection	Объект - это вам не коллекция :)
452	Invalid ordinal	Неверное числительное
453	Specified DLL function not found	Данная функция DLL не найдена
454	Code resource not found	Ошибка для макинтошей (это такие компы разноцветные)
457	This key is already associated with an element of this collection	Этот ключ уже ассоциирован с элементом коллекции
458	Variable uses a type not supported in Visual Basic	Переменная использует неподдер-живаемый Visual Basic'ом тип
459	This component doesn't support the set pf events	Единственный раз в жизни я увидел опечатку в Microsoft'овской документации! Слева вместо pf должно стоять of! А ошибка переводится примерно так: Не для всех компонентов можно использовать оператор WithEvevents и Implements.
460	Invalid Clipboard format	Неверный формат буфера обмена
461	Specified format doesn't match format of data	Данный фармат не совпадает с форматом данных
480	Can't create AutoRedraw image	Не могу создать AutoRedraw изображение
481	Invalid picture	Неверное изображение
482	Printer error	Ошибка принтера
483	Printer driver does not support specified property	Драйвер принтер не поддерживает данное свойство
484	Problem getting printer information from the system. Make sure the printer is set up correctly	Проблема при чтении информации принтера в системе. Убедитесь, что принтер правильно установлен

485	Invalid picture type	Неверный тип изображения
486	Can't print form image to this type of printer	Не могу распечатать изображение формы на таком типе принтера
520	Can't empty Clipboard	Не могу очистить буфер обмена
521	Can't open Clipboard	Не могу открыть буфер обмена
735	Can't save file to TEMP directory	Не могу сохранить файл в директорию TEMP (временную)
744	Search text not found	Искомый текст не найден
746	Replacements too long	Текст для замещение слишком длинный
31001	Out of memory	Не хватает памяти
31004	No object	Нет объекта
31018	Class is not set	Класс не установлен
31027	Unable to activate object	Не могу активизировать объект
31032	Unable to create embedded object	Не могу создать внедрённый объект
31036	Error saving to file	Ошибка записи в файл
31037	Error loading from file	Ошибка чтения из файла

Описание событий VB

Здесь размещены подробные описания всех событий всех основных элементов управления, встроенных в Visual Basic. Сюда не входят события подключаемых элементов управления, таких например, как Microsoft Common Controls и т.п. Многие события снабжены примерами их использования. Всё переводилось мной со стандартного Help'a VB5.

В Visual Basic структура обработчика событий имеет следующую схему:

```
[Private | Public] Sub Объект_ИмяСобытия ([arglist])

End Sub
```

Внутри таких процедур и содержится основной код Visual Basic. Остальное находится в пользовательских функциях и процедурах.

Выберите интересующее вас событие:

- Validate

Событие	Причина возникновения
`Activate,Deactivate`	Это событие имеет только форма. **Activate -** вызывается в тот момент, когда форма становится активной (получает фокус). Однако, если перейти к другому приложению Windows, а затем вернуться к своему, то это событие не произойдёт. Оно срабатывает только при переключении между формами внутри программы. **Deactivate** - Событие, обратное событию Activate. Вызывается при потере фокуса формы.
`Change`	**ComboBox** — меняется текст в текстовой части элемента. Происходит только тогда, когда свойство Style установлено в 0 (Dropdown Combo) или 1 (Simple Combo) и юзер изменяет текст или вы меняете его в коде программы. **DirListBox** — Меняется выбранная директория. Происходит когда юзер делает двойной клик на новой директории или когда меняется свойство Path в коде. **DriveListBox** — Меняется выбранное устройство. Происходит когда юзер меняет устройство, выбрав его из списка, или когда меняется свойство Drive в коде. **HScrollBar** и **VScrollBar** (горизонтальная и вертикальная прокрутки) — Подвинулся скрол. Происходит когда юзер передвинул и отпустил полоску прокрутки или меняется свойство Value в коде. **Label** — Меняется содержимое метки. Происходит когда меняется свойство Caption в коде. **PictureBox** — Меняется содержимое PictureBox. Происходит при смене свойства Picture. (а также при использовании LoadPicture, прим.eax) **TextBox** — Меняется текст в текстовом поле. Происходит при смене текста юзером или при смене свойства Text в коде.
`Click`	Происходит когда юзер нажимает и отпускает кнопку мышки над оъектом. Оно также может происходить при смене *некоторого значения* объекта. Для формы такое собитие выполняется при клике на свободном месте формы, или по отключённому элементу управления (т.е. когда его Enabled = False). Вообще событие происходит и для правой кнопки мыши и для левой. Но для элементов **CheckBox**, **CommandButton**, **Listbox**, и **OptionButton** происходит только при нажатии

	левой кнопки мыши. Для **ComboBox** или **ListBox** оно происходит и при клике мышой и также при смене текущего элемента клавишами курсора. Происходит при нажатии на "ПРОБЕЛ" у элементов **CommandButton**, **OptionButton**, или **CheckBox**, когда те имею фокус. При нажатии на ENTER при фокусе на элементе **CommandButton** (то бишь гашей кнопке) и когда установлено свойство Default. Происходит при нажатии на ESC когда форма имеет Cancel кнопку - **CommandButton** с установленым свойством Cancel. Также происходит при нажатии на горячую последовательность. Например, если кнопка имеет Caption - "&Go", то при нажатии Alt+G запуститься событие. Также, событие Click может быть сгенерировано в следующих случаях в коде: Установка значения Value для **OptionButton** и **CheckBox**.
`DblClick`	Происходит при двойном клике по объекту. Для формы происходит при двойном клике по форме, а также по отключённому объекту. Для других элементов: Двойной клик по объекту левой кнопкой. Двойной клик по элементу в **ComboBox**, когда Style = 1. Или также в **FileListBox**, **ListBox**, **DBCombo**, или **DBList**.
`DragDrop`	Происходит при завершении операции перетаскивания (Drag&Drop). В обработку события передаются 3 аргумента - координаты курсора (**X**,**Y**), где был отпущен объект, и ссылка на объект (**Source**), который был перетащен.
`DragOver`	Происходит когда объект перетаскивается над получателем, но кнопка ещё не отпущена. Имеет 4 параметра. **Координаты курсора**, **ссылка** на объект, и текущее состояние (**State** As Integer): 0 = Enter (вошёл) (источник вошёл в область объекта). 1 = Leave (покинул) (источник ушёл из этой области). 2 = Over (над) (произошёл сдвиг в пределах области).
`Error`	Происходит только при работе с **базами данных**. Коркретно: при ошибке в доступе к данным при выполнении кода.

	Имеет 2 параметра: **dataerr** - номер произошедшей ошибки **response** - номер, соответствующий выбранному в настройках (Settings): Если vbDataErrContinue, то response = 0 (Продолжить) Если vbDataErrDisplay, то response = 1 (Default) Показать сообщение об ошибке.
GotFocus	Происходит, когда объект получает фокус, или при нажатии кнопки Tab или кликом по объекту, а также при запуске метода SetFocus в программе. Форма получает фокус только тогда, когда все видимые элементы отключены (Enabled = False).
Initialize	Это событие имеет только форма. Обрабатывается первым и один раз. Visual Basic вызывает его при первом создании формы. Здесь обычно размещают код, для инициализации нужных переменных в программе.
ItemCheck	Происходит когда у ListBox контрола свойство Style установлено в 1 (checkboxes) и выбран или сброшен флажок у какого-либо элемента (item) в контроле. Передаётся один параметр: **index** - номер элемента, который был выбран в ListBox.
KeyDown, KeyUp	Происходит когда юзер нажимает (KeyDown) или отпускает (KeyUp) клавишу, в то время как объект имеет фокус. Чтобы получить код клавиши, используйте событие KeyPress. Параметров нет.
KeyPress	Происходит когда юзер нажамает и отпускает клавишу на клавиатуре. Событие имеет один параметр: **keyascii** - код нажатой клавиши. Например, если нажать на клавишу "1" (основного ряда), то keyascii будет равен 49. Если нажать ESC, то 27, и и.д. Если присвоить этой переменной 0, то нажатая буква (символ) не появится в текстом поле. Иногда это бывает очень удобным. Например, можно сделать так, чтобы в текстовое поле можно было вводить только цифры. Вот пример: Таблицу кодов клавиш вы можете посмотреть здесь.
LinkClose LinkError	LinkClose: Происходит когда DDE соединение закрывается.

LinkNotify LinkOpen	Может произойти в любое время. LinkError: Происходит при возникновении ошибки в ходе связи DDE. LinkNotify: Происходит когда у источника меняются данные, на которые установлена ссылка DDE. LinkOpen: Происходит при создании инициализации DDE связи с источником. За подробностями пожалуйста обращайтесь к справочникам, т.к. эти события используются очень редко.
Load	Это событие имеет только форма. Обрабатывается при загрузке формы в память. Происходит после события Initalize. Обычно код испольняется один раз. При запуске программы, это событие обрабатывается автоматически для той формы, которая загружается первой. Это событие может выполняться несколько раз. Т.е. если вы выгрузите форму оператором Unload, а затем вновь загрузите оператором Load или методом Show, то это событие будет выполнено.
LostFocus	Происходит при потере объектом фокуса, или при нажатии Tab юзером, или при использовании метода SetFocus для другого объекта.
MouseDown, MouseUp	Происходят когда юзер нажимает (MouseDown) или отпускает (MouseUp) кнопку мыши. Имеют 4 передаваемых параметра: **button** - содержит integer - номер кнопки мыши. С помощью этого события можно опередилить какую кнопку нажал пользователь. Для этого используюся константы: vbLeftButton (левая), vbMiddleButton (средняя), vbRightButton (правая). Проверить можно примерно так: `If button = vbRightButton Then ...` **shift** - содержит integer - указывающее на состояние клавиш Shift, Alt и Ctrl. Бит 0 - Shift, бит 1 - Ctrl, бит 2 - Alt. Для опредения факта нажатия клавиш Ctrl и Alt можно использовать такой код: `If (Shift And (vbCtrlMask Or vbAltMask)) = (vbCtrlMask Or vbAltMask) Then ...` Т.е. мы проверяем содержит ли переменная Shift биты 1 и 2 (оператором And). Скобки везде обязательны. vbShiftMask, vbAltMask, vbCtrlMask - это обычные константы, содержащие

	маски битов: `vbShiftMask = 1 (нулевой бит) Bin: 00000001` `vbCtrlMask = 2 (первый бит) Bin: 00000010` `vbAltMask = 4 (второй бит) Bin: 00000100` **x**, **y** - координаты курсора мыши того места, где произошло событие MouseUp или MouseDown. Координаты всегда зависят от координытной системы, задаваемой свойствами ScaleHeight, ScaleWidth, ScaleLeft, ScaleTop объекта.
`MouseMove`	Происходит когда курсор мыши изменяет своё положение над объектом. Т.е. когда курсор стоит на месте, событие не происходит. При каждом сдвиге курсора - срабатывает. Параметры те же, что и у MouseDown и MouseUp.
`ObjectMove`	Происходит немедленно после того, как OLE контейнер сдвигается или меняет размеры пока объект активен. Передаваемые параметры: **left** - координата левой грани OLE контейнера **top** - координата верхней грани OLE контейнера **width** - ширина OLE контейнера **height** - высота OLE контейнера
`OLECompleteDrag`	Происходит когда компонент - источник "брошен" на компонент - цель, информируя объект о том, что либо завершён, либо отменён процесс перетаскивания. Имеет один параметр: **effect** - привожу оригинал: A long integer set by the source object identifying the action that has been performed, thus allowing the source to take appropriate action if the component was moved (such as the source deleting data if it is moved from one component to another). The possible values are listed in Settings. A Settings может быть: vbDropEffectNone = 0 - операция "бросания" (Drop) была отменена. vbDropEffectCopy = 1 - показывать значок копирования данных. vbDropEffectMove = 2 - "сбрасываемые" данные являются ссылкой на оригинальные данные.
`OLEDragDrop`	Это событие происходит когда на объект сбрасываются OLE данные. Например, происходит при перетаскивании на объект файлов из проводника, или перетаскивании изображения из Internet Explorer и т.п. Замечание: Это событие происходит только в том случае,

	когда свойство OLEDropMode установлено в 1 (Manual). Имеет очень много параметров: **data** - объект типа DataObject. Имеет 4 метода и одно свойство - коллекцию файлов, перетаскивамых на объект. Для получения данных можно использовать метод GetData. Чтобы узнать формат данных, перекинутых на объект можно использовать метод GetFormat. **effect** Long - привожу оригинал - set by the target component identifying the action that has been performed (if any), thus allowing the source to take appropriate action if the component was moved (such as the source deleting the data). The possible values are listed in Settings. **button** - то же, что и в событии MouseDown(Up). **shift** - то же, что и в событии MouseDown(Up). **x**,**y** - то же, что и в событии MouseDown(Up). Пример. Обработаем перетащенные на форму файлы. Не забудьте установить свойство OLEDropMode в 1. `Private Sub Form_OLEDragDrop _` `(Data As DataObject, Effect As Long, _` `Button As Integer, Shift As Integer, _` `X As Single, Y As Single)` `' перетаскиваются файлы?` `If Data.GetFormat(vbCFFiles) = True Then` `Dim c As Long` `' пройдёмся по всей коллекции` `For c = 1 To Data.Files.Count` `' покажем имя перетащенного файла` `MsgBox "Был перетащен файл: " & Data.Files(c)` `Next c` `End If` `End Sub`
`OLEDragOver`	Происходит при перетаскивании над объектом. Парметры - комбинация парамтров событий OLEDragDrop и обычного DragOver.
`OLEGiveFeedback`	Происходит после каждого события OLEDragOver. OLEGiveFeedback позволяет источнику обеспечивать визуальную "отдачу" пользователю, такое, как изменение курсора мыши (посмотрите на него при перетаскивании файлов в проводнике с нажатым Ctrl, - видите значок "+", вот это и есть та "отдача"?), которое будет говорить пользователю о том, что происходит. Параметры - effect и **defaultcursors** - boolean - разрешает или запрещает использование курсора по умолчанию. Если True - то

	используется курсор по умолчанию, если False, то используется курсор, указанный пользователем в свойстве MousePointer объекта Screen.
OLEStartDrag	Происходит тогда, когда начинается перетаскивание (т.е. когда перетаскиваемый объект (файл, например) появляется над объектом приёмником). Имеет 2 параметры: **data** - то же что и у события OLEDragDrop. **allowedeffects** - Long - содержит поддерживаемые источником эффекты. Возможные значения описаны в Setting'cax.
Paint	Происходит когда часть или весь объект появляется на экране после сдвига или изменения размера. Событие происходит также при сдвиге окна, которое закрывает объект. Разберём чуть подробнее: Событие Paint полезно, если вы используете графические методы объекта (Line, PSet...) в коде. С помощью этого события вы можете убедиться в том, что данные перерисовались, когда это необходимо. Событие Paint вызывается при запуске метода Refresh. Если AutoRedraw установлено в True, то перерисовка происходит автоматически, таким образом, это событие теряет свою необходимость. Если свойство ClipControls установлено в False, то графические методы в процедуре Paint воздействуют только на видимую часть формы; иначе, графические методы перерисовывают всю открытую часть формы (т.е. ту, которая не перекрыта такими элементами, как Image, Label, Line, и Shape). Используя метод Refresh в событии Resize вызывает перерисовку всего объекта каждый раз, когда происходит изменение его размеров (Resize). Замечание: Используя событие Paint для определённых задач, может произойти каскадирование событий (Т.е. просто напросто произойдёт рекурсия, когда Paint будет вызывать сам себя, и произойдёт переполнение стека). Чтобы этого избежать, нужно придерживаться следующих правил: · Избегать вызова события Paint при сдвиге или изменении размеров объекта. · Внутри события Paint изменять свойства, которые могут вызвать событие Paint. Такие, например, как BackColor. · Включать метод Refresh метод внутрь Paint.

PathChange	Происходит при смене пути, установкой свойства FileName или Path в коде. Замечание: Вы можете использовать это события, чтобы сообщить FileListBox'y о том, что путь у DirListBox изменился.
PatternChange	Происходит при изменении маски (такой, как "*.*") установкой свойств FileName или Pattern в коде.
QueryUnload	Это событие имеет только форма. Происходит перед закрытием формы или приложения. Если закрывается MDI форма, то это событие происходит сначала для формы контейнера, и лишь потом для дочерних форм. Это событие происходит перед событием Unload. В обработчик данного события передаётся два параметра: cancel - integer - если в обработчике события присвоить переменной cancel значение True, то выгрузка формы будет отменена. Если оставить там False, то форма благополучно выгрузится. unloadmode - константа. Содержит значение - почему происходит выгрузка формы. Может принимать: vbFormControlMenu = 0 - юзер выбрал команду Close (Закрыть) в меню приложения (слева вверху в заголовке формы). vbFormCode = 1 - произошёл вызов оператора Unload в коде программы. vbAppWindows = 2 - Windows завершает работу. vbAppTaskManager = 3 - закрытие приложения происходит через Ctr+Alt+Del. vbFormMDIForm = 4 - дочерняя MDI форма закрывается, т.к. закрывается главная. Замечания: Вообще, это событие обычно применяется для проверки завершённости некоторых действий. Или также, например, здесь можно спросить юзера, действительно ли он хочет выйти из приложения, или это произошло случаянно. Следать это можно примерно так: `Private Sub Form_QueryUnload _` `(Cancel As Integer, _` `UnloadMode As Integer)`

	`` Dim rez As VbMsgBoxResult rez = MsgBox("Вы действительно" & _ " хотите выйти?", _ vbQuestion + vbYesNo) If rez = vbNo Then Cancel = 1 End Sub ```
Reposition	Происходит когда запись становится текущей записью. Замечания: Когда Data контрол загружен, первая запись в объекте RecordSet становится текущей, и вызывается это событие. Когда бы юзер не кликнул любую кнопку на контроле Data, передвигаясь с записи на запись, или вы используете одно из методов Move в коде (такие, как MoveFirst, MoveNext, FindFirst..), или любое другое свойство, которое может изменить текущую запись - происходит событие Reposition. Событие Validate происходит перед этим событием.
Resize	Происходит когда объект первый раз появляется или когда меняется состояние окна (например, при свёртывании и развёртывании окна). А также при смене размеров окна. Это событие удобно использовать, если вы хотите сделать "растягивающийса" интерфейс. Т.е. когда все элементы на форме меняют свои размеры, в зависимости от текущих размеров формы. Код для изменения размеров этих элеметов как раз можно поместить в это событие.
Scroll	Происходит тогда, когда сдвигается полоса прокрутки (ScrollBar). Замечания: Для DBGrid это событие происходит, когда двигается вертикальная или горизонтальная полоса прокрутки. Для ComboBox это событие происходит при сдвиге скролбара в выпадающем списке. Вы можете использовать это события для синхронизации изменения положения полоски прокрутки и другими объектами, с которыми эта полоска связана. К примеру, в обработку этого события вставить код, который будет синхронизировать текущую позицию с другим элеметом. Поместите на форму HScroll и вставьте такой код: ``` Private Sub HScroll1_Scroll() Form1.Caption = HScroll1.Value End Sub ``` Теперь в заголовок формы будет меняться в соответствии со

	сменой положения полоски прокрутки. Кстати, избегайте использования фукнции MsgBox в этом событии!
Terminate	Это событие имеет только форма. Происходит когда все ссылки на экземпляры форм, MDI форм, элементы управления, классы удалены из памяти (установкий переменной в Nothing). Это событие выполняется после события Unload, и выполняется самым последним в программе. Можете использовать его по своему усмотрению.
Timer	Это событие имеется только у элемента управления **Timer**. Оно происходит через определённый интервал времени, указанный в его свойтве Interval. Timer невидим для пользователя, и полезен для внутренних процессов программы. Его можно использовать когда необходимо, чтобы некоторый код программы выполнился через определённый промежуток времени. Например, с его помощью можно сделать часы. Для этого достаточно поместить на форму элемент Timer, установите его свойство Interval в 1000 (1 сек = 1000 мс), и вставить следующий код: `Private Sub Timer1_Timer()` `    Form1.Caption = Time` `End Sub` Функция Time возвращает текущее системное время. Таким образом, каждую секунду будет выполнять событие Timer, и следовательно изменение заголовка формы. Примечание: количесво таймеров на форме не ограничего.
Unload	Это событие имеет только форма. Происходит когда форма выгружается из памяти (оператором Unload, например, или нажатием на крестик). В дальнейшем она может быть загружена оператором Load. При перезагрузке формы ВСЕ ЭЛЕМЕНТЫ УПРАВЛЕНИЯ ИНИЦИАЛИЗИРУЮТСЯ ЗАНОВО, т.е. все значения, которые были в них - теряются.

<table>
<tr><td>Updated</td><td>Происходит когда меняются данные (при работе с БД).

Имеет параметр Code - integer - описывает, как объект был изменён (описано в настройках). Может принимать следующие константы:

vbOLEChanged = 0 - Данные объекта изменились
vbOLESaved = 1 - Данные объекта были сохранены приложением
vbOLEClosed = 2 - Файл, на который ссылается объект был закрыт
vbOLERenamed = 3 - Файл, на который ссылается объект был переименован</td></tr>
<tr><td>Validate</td><td>Это событие нужно для проверки введённых данных. Например, в TextBox. Это событие выполняется только тогда, когда свойство объекта CausesValidation = True.

Рассмотрим пример, как можно проконтролировать данные, введённые в текстовое поле. Поместите на форму TextBox и вставьте код:

<pre>Private Sub Text1_Validate _
(Cancel As Boolean)
    If Not (IsNumeric(Text1.Text)) Then
        Text1.ForeColor = vbRed
        MsgBox "Введите в TextBox числа"
        Cancel = True
    Else
        Text1.ForeColor = vbButtonText
    End If
End Sub</pre>
Этот код не примет введённый в TextBox текст, пока тот не будет являться числом. Если установить параметр Cancel в True, то фокус вернётся обратно TextBox.

Это событие происходит при потере фокуса у объекта.</td></tr>
</table>

Функции и процедуры в VB

Процедуры и функции представляют собой отдельные блоки, из которых складывается код программы, каждая процедура выполняет какую-то задачу или ее часть.

Процедуры обработки событий после вызова постоянно находятся в ожидании событий.

Кроме процедур обработки событий в программу можно включить процедуры и функции не связанные с событиями. Они выполняют отдельные действия и могут быть использованы неоднократно. Назовем их общими. Процедуры общего назначения вызываются на выполнение в коде программы. Использование процедур экономит время и позволяет избежать лишних ошибок. Функции отличаются от процедур тем, что возвращают какое-то значение.

Под процедурой или функцией понимается последовательность операций, которую нужно многократно выполнять в различных местах приложения. При этом требуемый блок команд записывается в коде только один раз, после чего к нему можно обращаться из любой части программы.

Функция – это подпрограмма, которую вызывают, чтобы выполнить какие-то расчеты или проверки. Когда она завершает работу, то возвращает управление вызывающей программе и передает ей результат расчета.

Процедура – это тоже подпрограмма. Ее тоже вызывают, чтобы выполнить какие-то действия, но от нее не требуется возвращать основной программе какие-либо значения.

Синтаксис объявления процедуры и функции:

Код:

```
[Public/Private][Static] Sub <Имя процедуры>(<Параметры>)
<Операторы>
End Sub

Function <Имя функции> [As тип]
<Операторы>
End Function
```

Процедуры, объявленные с ключевым словом Public, можно вызвать в любом модуле приложения (каждая форма – это отдельный модуль).

Процедуры объявленные как Private, можно вызывать только в текущем модуле.

Слово Static означает, что все переменные, объявленные в процедуре, будут статическими, т.е. их значения сохраняются между вызовами.

Параметры обеспечивают связь процедуры с приложением. Это данные, передаваемые в процедуру при вызове.

Процедуры обработки событий. Вызываются в том, случае если произошло какое-либо событие. При этом существенным является как имя элемента, та и вид события, который с ним произошел.

Пользовательские процедуры. Группы операторов, создаваемые разработчиком для выполнения определенных задач и не зависящие от текущего состояния приложения или произошедших в тот или иной момент событий.

Встроенные функции. Определенные наборы команд, имеющиеся в языке Visual Basic и в предназначенные для вычисления тех или иных значений на основании исходных данных. Встроенными являются, в частности, как математические, так и строковые функции (Abs, Cos, Sin, Mid, Len и т.д.)

Пользовательские функции. Группы операторов, аналогичные пользовательским процедурам.

Однако между ними есть ряд отличий.

Основные отличия функции от процедуры состоят в следующем.

1. Функция имеет тип (аналогично переменной) и может возвращать в программу значение, которое присваивается функции при помощи оператора:

<Имя функции> = значение

2. Вызов функции, как правило, осуществляется посредством указания в правой части какого-либо оператора ее имени и параметров. С другой стороны, процедура вызывается при помощи отдельного оператора:

Call <Имя процедуры> (Параметры)
Или
<Имя процедуры> (Параметры)

Если при вызове процедуры используется ключевое слово Call, то список параметров должен быть указан в скобках. Если же процедура вызывается без использования Call, то ее параметры перечисляются без скобок.

Необходимо отметить, что вызываемая процедура может не иметь параметров. В этом случае (если использовалось служебное слово Call) после имени процедуры следует ставить пустые скобки.

Пользовательские процедуры обычно используются при необходимости выполнения одной и тоже последовательности операций. Например, в программе требуется неоднократно вводить в цикле значения массива arrA, состоящего из пяти элементов. В этом случае заполнение массива лучше всего оформить в виде процедуры.

Команда Add Procedure меню Tools позволяет добавить процедуру или функцию.

Пусть процедура Cir вычерчивает эллипс с координатами x, y, которые передаются в процедуру как параметры. Создавая процедуру Cir командой Add Procedure, нужно указать имя процедуры и выбрать область видимости Public или Private.

Завершив диалог, получим объявление процедуры:

Код:

```
Private Sub Cir()
…
End Sub
```

Теперь нужно вписать параметры в скобки и написать текст процедуры. В списке параметров рекомендуется указывать тип переменных.

Код:

```
Private Sub Cir(x As Integer, y As Integer)
Circle (x,y),500,,,2
End Sub
```

Автор: BiT
Источник: www.bit.pirit.info/forum/

Ссылки по теме

Вызов функций по указателю

Техника программирования сложных окон в Visual Basic

Создание программы на Visual basic для вывода случайного числа в заданном интервале чисел

Visual Basic и Системный Реестр Windows

Встроенные функции VisualBasic

Вся документация по Visual Basic