Выражения в Visual Basic

Выражения

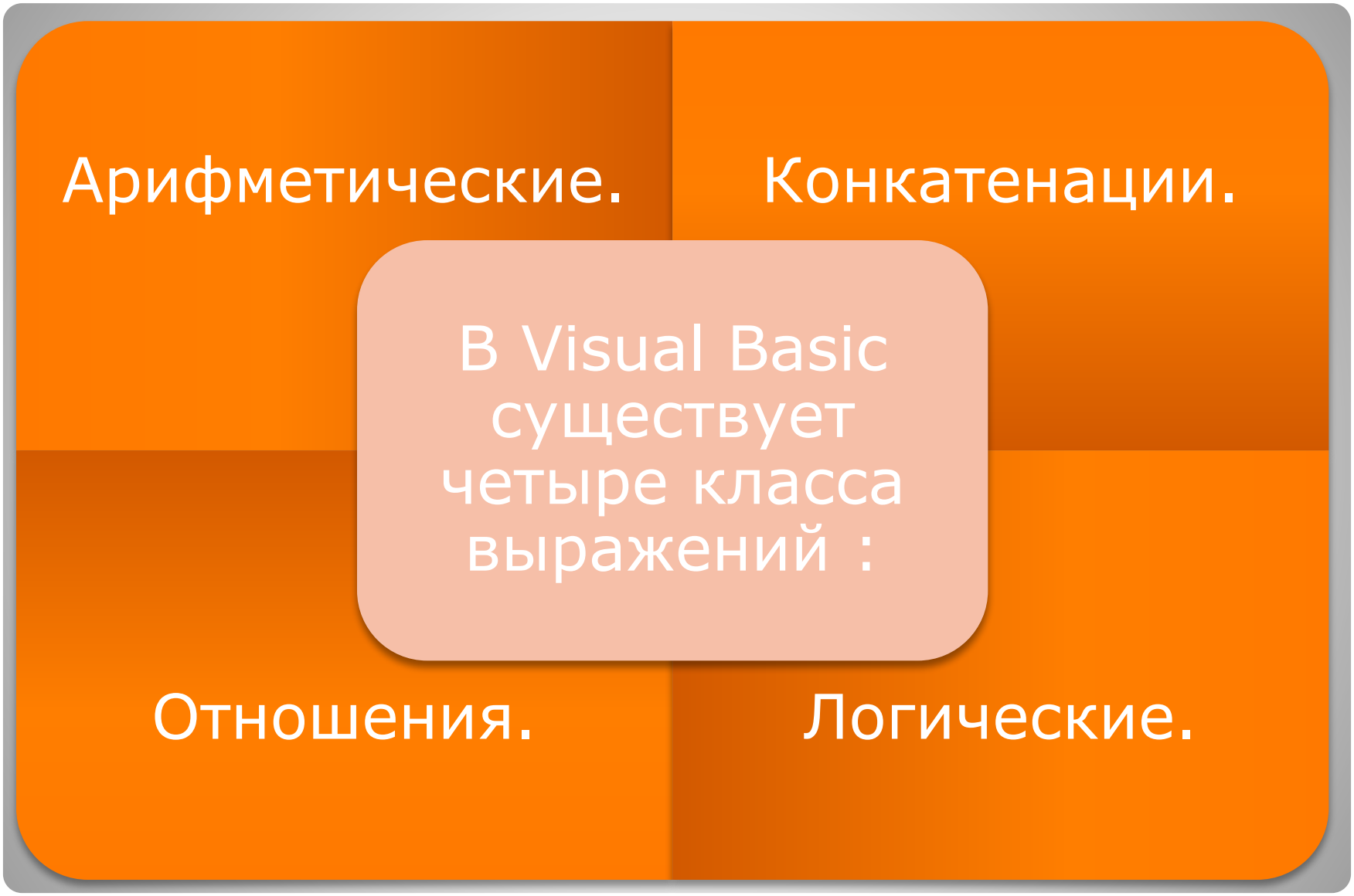
Выражение - это формула для вычисления величины.

Выражение содержит последовательность **операндов** и **операторов**.

Операнды могут содержать вызовы функций, переменные, константы, или другие выражения.

Операторы определяют действия, которые необходимо произвести над операндами.

Выражения в Visual Basic.



Выражения в Visual Basic.

Арифметические выражения

Выражения в Visual Basic.

Простейшие виды арифметических выражений

- Использование **констант**.
- Использование **переменных**.
- Использование **элементов массивов**.
- Использование **функций**.

*Другие арифметические выражения создаются из простейших форм, с использованием **скобок** и **арифметических операторов**.*

Арифметические выражения.

^ возведение в степень

Синтаксис: `результат = число^степень.`

Параметры:

результат - обязателен; любая числовая переменная

число - обязательно; любое числовое выражение

степень - обязательна; любое числовое выражение

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.
```

Арифметические операторы.

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**.

Возведение в степень.

* умножение

Синтаксис: *результат = число1 * число2.*

Параметры:

результат - обязателен; любая числовая переменная

число - обязательно; любое числовое выражение

Пример:

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

Арифметические операторы.

Замечания:

- тип результата обычно такой же, как и самый точный тип из двух чисел;

Порядок точности, от большего к меньшему:

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 одно из чисел - **Null**, то оно интерпретируется просто как обычный **0**.*

Умножение.

/ деление

Синтаксис: *результат = число1 / число2.*

Параметры:

результат - обязателен; любая числовая переменная

число - обязательно; любое числовое выражение

Пример:

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

Арифметические операторы.

Замечания:

- *Результат* обычно имеет тип Double;

есть исключения:

- *Если оба выражения имеют тип **Byte**, **Integer**, **Single**, то результат **Single**. Однако если размеры выражения не вписываются в рамки **Single**, происходит ошибка*;
- *Если оба 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 одно из чисел - **Null**, то оно интерпретируется просто как обычный **0**.*

Деление.

\ целочисленное деление

Синтаксис: *результат = число1 \ число2.*

Параметры:

результат - обязателен; любая числовая переменная

число - обязательно; любое числовое выражение

Пример:

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

Арифметические операторы.

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**, или **Long Variant**;
- если любое из выражений **Null**, то и результат **Null**;
- любое выражение, содержащее **Empty** интерпретируется как **0**.

Целочисленное деление.

Mod остаток от деления

Синтаксис: *результат = число1 Mod число2.*

Параметры:

результат - обязателен; любая числовая переменная

число - обязательно; любое числовое выражение

Пример:

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

Арифметические операторы.

Замечания:

- при делении числа с плавающей точкой округляются;

  Например: **19 Mod 6.7** равно **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**.

Остаток от деления.

+ сложение

Синтаксис: ***результат = выражение1 + выражение2***

Параметры:

результат **- обязателен; любая числовая переменная**

выражение **- обязательно; любое выражение**

Пример:

```
Dim MyResult, Var1, Var2,  Dim d As Date
MyResult = 4257.04 + 98112      ' Возвратит 102369.04
Var1 = "34”
Var2 = 6    ' Инициализируем смешанные переменные
MyResult = Var1 + Var2    ' Возвратит 40
Var1 = "34”
Var2 = “6"        ' Инициализируем переменные со строками
MyResult = Var1 + Var2 ' Возвратит "346" - произошла конкатенация а не сложение!
d = DateSerial(2008,11,14)' инициализация даты 14.11.2008
d = d + 7    ' теперь d содержит дату 21.11.2008 т.е. мы прибавили 7 дней
```

Арифметические операторы.

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

Сложение.

Замечания:

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 то происходит **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**, то возникает ошибка несовпадения типов ;
- если любое из выражений **Null**, то и результат **Null**;

Сложение.

Замечания:

Если оба выражения **Variant**, то применяются следующие правила:

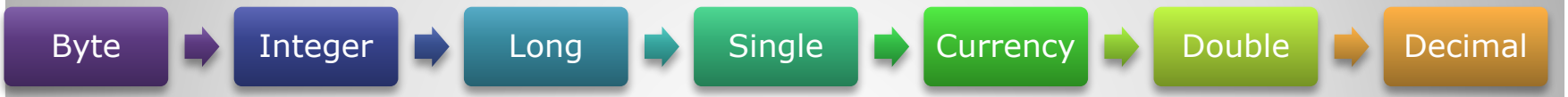
- если оба выражения **числа** - то они **складываются**;
- если оба выражения **строки** - то они **конкатенируются** ;
- если одно из выражений **число**, а другое **строка** - **происходит сложение**;

Сложение.

Замечания:

- Для обычного сложения тип данных результата обычно такой же, как и самый точный тип из двух чисел;

Порядок точности, от большего к меньшему:



есть исключения:

- *Если складываются **Single** и **Long**, то результат - **Double**;*
- *Если выражение с типом **Date** складывается с любым другим выражением, то результат - **Date**.*
- *Если одно или оба выражения **Null**, то результат тоже **Null**.*
- *Если оба выражения содержат **Empty**, результат **Integer**. Если только одно, то в качестве результата возвращается неизменённое второе выражение.*

Сложение.

- вычитание, смена знака

Синтаксис: *результат = выражение1 - выражение2*

или: *-выражение*

Параметры:

результат - обязателен; любая числовая переменная

выражение - обязательно; любое выражение

Пример:

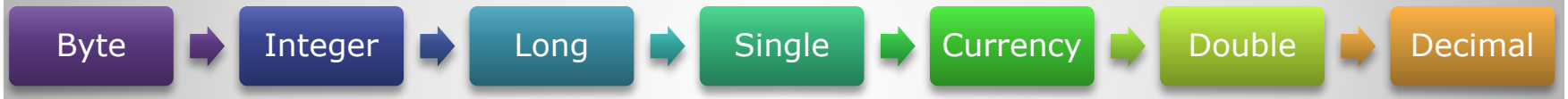
```
Dim MyResult,
Dim d1 As Date, Dim d2 As Date,
Dim razn As Long
MyResult = 4 - 2        ' Возвратит 2
MyResult = 459.35 - 334.90      ' Возвратит 124.45
d1 = DateSerial(1983,10,14)     ' инициализация даты 14.10.1983
d2 = DateSerial(2008,10,14)     ' инициализация даты 14.10.2008
razn = d2 – d1   ' разница в днях (9132)
```

Арифметические операторы.

Замечания:

- Оператор "–" используется для нахождения разницы между двумя числами, или для изменения знака выражения;
- Тип данных результата обычно такой же, как и самый точный тип из двух чисел.

Порядок точности, от большего к меньшему:



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**.*

Вычитание, смена знака.

Выражения отношения

Выражения в Visual Basic.

Выражения отношения

- Используются для сравнения выражений.
- Имеют 3 синтаксиса.

Синтаксис:

- *результат = выражение1 оператор_сравнения выражение2*
- *результат = объект1* ***Is*** *объект2*
- *результат = строка* ***Like*** *образец*

Параметры:

результат - **обязателен; любая численная переменная**

выражение - **обязательно; любое выражение**

оператор_сравнения - **обязателен; любой оператор сравнения**

объект - **обязателен; имя любого объекта**

строка - **обязательна; любое строковое выражение**

образец - **обязателен; любое строковое выражение, или диапазон букв и цифр**

Выражения отношения.

Список операторов сравнения и условия, по которым определяется результат выражения

Оператор	True	False	Null
<	***выр_1 < выр_2***	***выр_1 >= выр_2***	одно из выражений содержит **Null**
<=	***выр_1 <= выр_2***	***выр_1 > выр_2***	
>	***выр_1 > выр_2***	***выр_1 <= выр_2***	
>=	***выр_1 >= выр_2***	***выр_1 < выр_2***	
=	***выр_1 = выр_2***	***выр_1 <> выр_2***	
<>	***выр_1 <> выр_2***	***выр_1 = выр_2***	

Выражения отношения.

Замечания:

- Операторы **Is** и **Like** выполняют специфические функции, и их таблица сравнения отличается от приведённой;
- Когда сравниваются два выражения, не всегда можно определить, что будет сравниваться, **числа** или **строки**.

Выражения отношения.

Замечания:

Если оба выражения имеют тип, отличный от **Variant**:

- если оба выражения имеют численный тип - то происходит **сравнение чисел**;
- если оба выражения строки то происходит **сравнение строк**;
  меньшая строка та, первая и последующие буквы которой имеют 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 возникает 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 **""**.

Выражения отношения.

Замечания:

Если оба выражения 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**.

Выражения отношения.

Замечания:

- если **Single** сравнивается с **Double**, то **Double** округляется до точности **Single**;
- если **Currency** сравнивается с **Single** или **Double**, то **Single** или **Double** конвертируются в **Currency**;
- если **Decimal** сравнивается с **Single** или **Double**, то **Single** или **Double** конвертируются в **Decimal**.

Выражения отношения.

Замечания:

- При потере дробной части, выражения могут интерпретироваться как равные, хотя на самом деле, одно от другого будет отличаться (хоть и на маленькое значение):
  - ✔ Для **Currency** любая дробная часть меньшая, чем **.0001**, может быть утеряна;
  - ✔ Для **Decimal** значение **1E-28** может быть утеряно, или может произойти ошибка.

Выражения отношения.

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
MyResult = (Var1 > Var2)    ' Возвратит True.
Var1 = 5: Var2 = Empty
MyResult = (Var1 > Var2)    ' Возвратит True.
Var1 = 0: Var2 = Empty
MyResult = (Var1 = Var2)    ' Возвратит True.
```

Выражения отношения.

Is оператор сравнения

Синтаксис: *результат = объект1 Is объект2*

Параметры:

результат - обязателен; любая числовая переменная

объект - обязателен; имя любого объекта

Пример:

```
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.
```

Выражения отношения.

Замечания:

- Если ***объект1*** и ***объект2*** ссылаются на один и тот же объект, то результат - **True**, если нет, то **False**.
- Две переменные могут ссылаться на один и тот же объект несколькими путями.

Например:

```
Set A = B ' A ссылается на тот же объект, что и B
Set A = C
Set B = C ' A и B ссылаются на один и тот же объект-C
```

Оператор сравнения Is.

Like оператор сравнения строк

Синтаксис: *результат = строка Like образец*

Параметры:

результат – обязателен; любая числовая переменная
строка – обязательна; любое строковое выражение
образец – обязателен; любое строковое выражение, или диапазон букв и цифр

Пример:

```
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 ...
```

Выражения отношения.

Замечания:

- этот оператор можно использовать для проверки строки **String** на маску **Pattern**;
- если строка подходит под маску, то результат **True**, иначе – **False**;
- если одно из выражений **Null** - результат тоже **Null**;

Оператор сравнения строк Like.

Замечания:

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:

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

- если установлен тип **Text** (текстовое сравнение), то большие и маленькие буквы – равны:

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

Оператор сравнения строк Like.

Замечания:

В **маске** можно использовать следующие спец. символы:

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

- для того чтобы проверить принадлежность строки на маску, содержащую спец. символы нужно заключить из в квадратные скобки **[]**;
- всё, что находится в скобках не должно содержать никаких разделителей (пробелов, запятых и т.д.), иначе они тоже будут включены в последовательность.

Оператор сравнения строк Like.

Замечания:

- (**!**) 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;

  [A-Z] - *правильная маска,* **[Z-A] -** *неправильная маска*

- последовательность **[]** интерпретируется как пустая строка **""**.

Оператор сравнения строк Like.

Конкатенация строк

Конкатенация строк

- Чтобы соединить строки в Visual Basic , можно использовать всего 2 оператора это **&** и **+**.
- Оператор **+** описан выше.
- Поговорим об операторе **&**.

Конкатенация строк.

& оператор конкатенации

Синтаксис: *результат = выражение1 & выражение2*

Параметры:

результат – обязателен; любая String или Variant переменная

выражение – обязательно; любое выражение

Пример:

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

Конкатенация строк.

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**;
- если только одно из выражений содержит **Null** значение, то оно интерпретируется как пустая строка **""**;
- **Empty** также интерпретируется как пустая строка **""**.

Оператор конкатенации .

Логические выражения

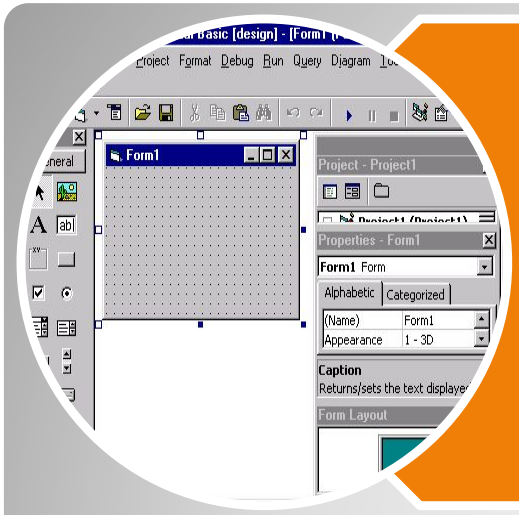
Выражения в Visual Basic.

Простейшие формы логических выражений

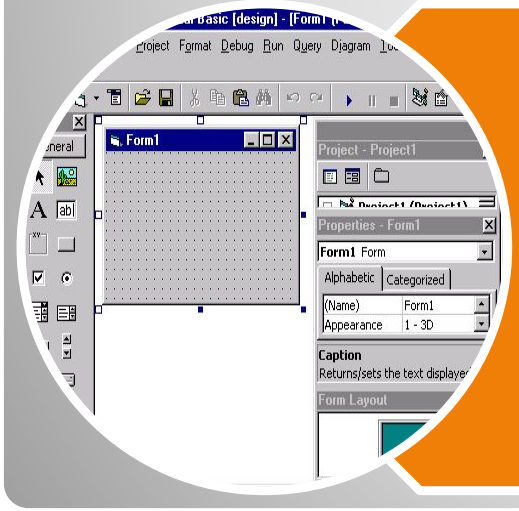
- Логические **константы**.
- Ссылки на **логические переменные**.
- Ссылки на элементы **логических массивов**.
- Ссылки на **логические функции**.
- Выражения отношения.

*Другие логические выражения состоят из простейших логических форм, приведенных выше, с использованием скобок и **логических операторов**.*

Логические выражения.



В Visual Basic 6 логических операторов.



Рассмотрим каждый оператор подробно.

Логические выражения.

And логическое умножение

Синтаксис: *результат = выражение1 And выражение2*

Параметры: *результат* – обязателен; любая числовая переменная
выражение – обязательно; любое выражение

Пример:

```
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 1010 and 1000 = 1000).
```

Логические выражения.

Следующая таблица показывает, как работает оператор **And** :

выр.1	выр.2	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.

Оператор **And** может быть также использован для проверки битов числа.

Для битов оператор **And** работает следующим образом:

bit_1	bit_2	результат
0	0	0
0	1	0
1	0	0
1	1	1

Оператор And.

Or логическое сложение

Синтаксис: *результат = выражение1* Or *выражение2*

Параметры: *результат* – обязателен; любая числовая переменная
выражение – обязательно; любое выражение

Пример:

```
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 C > B  ' Возвратит False.
MyCheck = A > B Or B > D  ' Возвратит True.
MyCheck = B > D Or B > A  ' Возвратит Null.
MyCheck = A Or 5
      ' Возвратит 15 (битовое сравнение 1010 or 0101 = 1111).
```

Логические выражения.

Следующая таблица показывает, как работает оператор **Or**:

выр.1	выр.2	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.

Оператор **Or** может быть также использован для установки определённых битов числа.

Для битов оператор **Or** работает следующим образом:

bit_1	bit_2	результат
0	0	0
0	1	1
1	0	1
1	1	1

Оператор Or.

Xor логическое отрицание

Синтаксис: *результат = выражение1 Xor выражение2*

Параметры: *результат* – обязателен; любая числовая переменная
выражение – обязательно; любое выражение

Пример:

```
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 (инвертирование битов 1010 xor 1000 = 0010).
```

Логические выражения.

Следующая таблица показывает, как работает оператор **Xor**:

выр.1	выр.2	результат
True	True	False
True	False	True
False	True	True
False	False	False

Xor *отличается от* **Or**, *только тем, что когда оба бита единицы,* **Xor** *выдаёт* **0**.

Оператор Xor.

Оператор **Xor** может быть также использован для инвертирования определённых битов числа.

Для битов оператор **Xor** работает следующим образом:

bit_1	bit_2	результат
0	0	0
0	1	1
1	0	1
1	1	0

Оператор **Xor** *интересен тем свойством, то при его двойном применении он выдаёт то же число. Это часто используют в криптографии.*

Оператор Xor.

Интересным примером использования оператора **Xor** является обмен значениями двух численных переменных:

```
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*

Оператор Xor.

Not логическое инвертирование

Синтаксис: *результат = Not выражение*

Параметры: *результат* – обязателен; любая числовая переменная
выражение – обязательно; любое выражение

Пример:

```
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
          ' Not 00001010 = 11110101)
```

Следующая таблица показывает, как работает оператор **Not**:

выр.	результат
True	False
False	True
Null	Null

Оператор Not.

Оператор **Not** может быть также использован для инвертирования всех битов числа.

Для битов оператор **Not** работает следующим образом:

bit_1	результат
0	1
1	0

Оператор Not.

Eqv логическая эквивалентность

Синтаксис: *результат = выражение1 Eqv выражение2*

Параметры: *результат* – обязателен; любая числовая переменная
выражение – обязательно; любое выражение

Пример:

```
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 (сравнение битов 00001010 eqv 00001000 =
11111101)
```

Следующая таблица показывает, как работает оператор **Eqv**:

выр.1	выр.2	результат
True	True	True
True	False	False
False	True	False
False	False	True

Оператор Eqv.

Оператор **Eqv** может быть также использован для побитного сравнения чисел.

Для битов оператор **Eqv** работает следующим образом:

bit_1	bit_2	результат
0	0	1
0	1	0
1	0	0
1	1	1

Оператор Eqv.

Imp логическая импликация

Импликация - вовлечение, включение

Синтаксис: *результат = выражение1 Imp выражение2*

Параметры: *результат* – обязателен; любая числовая переменная
выражение – обязательно; любое выражение

Пример:

```
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 (импликация битов 00001000 imp 00001010 = 11111111)
```

Логические выражения.

Следующая таблица показывает, как работает оператор **Imp**:

выр.1	выр.2	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

Оператор Imp.

Для битов оператор **Imp** работает следующим образом:

bit_1	bit_2	результат
0	0	1
0	1	1
1	0	0
1	1	1

Оператор Imp.

Приоритеты операторов

Когда в одном выражении встречаются арифметические, логические операторы и операторы отношения, они выполняются со следующими приоритетами :

1. Арифметические (высший)
2. Отношения (средний)
3. Логические (низший)

Приоритеты операторов.

Порядок следования приоритетов операторов Visual Basic

приоритет убывает с верху вниз и слева направо

рицание (-)	Неравно (<>)	
ие и деление (*, /)	Меньше чем (<)	Or
ленное деление (\)	Больше чем (>)	Xor
от деления (Mod)	Меньше или равно (<=)	Eqv
ычитание (+,-)	Больше или равно (>=)	Imp
катенация (&)	Удовлетворение маске (Like)	
	Принадлежность к типу (Is)	

Если в выражении встречаются операторы из разных категорий, то вычисляются они в порядке столбцов слева направо

Приоритеты операторов.