Операции языка Visual Basic.NET

Последнее обновление: 30.10.2015

Visual Basic.NET поддерживает большинство стандартных операций, принятых и в других языках программирования. Рассмотрим все виды операций.

Арифметические операции

- `+`

  сложение двух чисел

- `-`

  вычитание двух чисел

- `*`

  умножение

- `^`

  возведение в степень

- `/`

  обычное деление

- `\`

  целочисленное деление двух чисел

- `Mod`

  Получение остатка от деления двух чисел

Примеры:

```
Dim x1 As Integer = 6 + 7 'Результат равен 13
```

```
Dim x2 As Integer = 6 - 7 'Результат равен -1
Dim x3 As Integer = 6 * 7 'Результат равен 42
Dim x4 As Integer = 12 / 6 'Результат равен 2
Dim x5 As Integer = 13 \ 6 'Результат равен 2, а остаток отбрасывается
Dim x6 As Integer = 13 Mod 6 'Результат (он же остаток от деления 13 на 6) ра
Dim x7 As Integer = 6 ^ 2 'Результат равен 36
```

Операции сравнения и логические операции

В Visual Basic.NET также имеются операторы сравнения:

- >

  Больше

- >=

  Больше или равно

- <

  Меньше

- <=

  Меньше или равно

- =

  Равно (в данном случае используется как знак сравнения на равенство двух значений)

- <>

  Не равно

Также в языке определены логические операторы, которые возвращают в отличие от арифметических операций значение типа Boolean:

- And

  Логическое умножение (логическое И)

- Or

  Логическое сложение (логическое ИЛИ)

- Xor

  Исключающее "или"

- Not

  Логическое отрицание

- AndAlso

  Сокращенный оператор And

- OrElse

  Сокращенный оператор Or

Оператор And возвращает результат True только в том случае, если все выражения истинны. В остальных случаях он возвращает False. Оператор Or возвращает True, если хотя бы одно из выражений истинно. Он возвращает False тогда, когда неверны все выражения. Оператор Xor возвращает True, если одно из выражений истинно, а другое ложно. В остальных случаях, если оба выражения либо истинны, либо ложны, возвращается False. Оператор Not возвращает True, если выражение ложно, и False, если выражение истинно. Примеры использования:

```
1  Dim x As Boolean = 6 > 2 And 2 < 4 'Результат True, так как и первое выражен
2  Dim y As Boolean = 6 > 2 And 2 > 4 'Результат False, так как только одно выр
3
4  Dim x1 As Boolean = 6 > 2 Or 2 < 4 'Результат True, так как хотя бы одно выр
5  Dim y1 As Boolean = 6 > 2 Or 2 > 4 'Результат True, так как опять же одно вы
6  Dim y2 As Boolean = 6 < 2 Or 2 > 4 'Результат False, так как оба выражения л
7
8  Dim x2 As Boolean = 6 > 2 Xor 2 < 4 'Результат False, так как оба выражения
9  Dim y2 As Boolean = 6 > 2 Xor 2 > 4 'Результат True, так как одно выражение
10
11 Dim x3 As Boolean = Not 2 < 4 'Результат False, так как выражение истинно
12 Dim y3 As Boolean = Not 2 > 4 'Результат True, так как выражение ложно
```

Операторы AndAlso и OrElse хотя возвращают тот же самый результат, что и And и Or, но все же между ними есть отличия. Так допустим у нас есть следующие два выражения, возвращающие идентичный результат :

```
1  Dim x As Boolean = 6 > 2 And 2 < 4
2  Dim x As Boolean = 6 > 2 AndAlso 2 < 4
```

В первом случае (в случае с `And`) программа сначала проверяет истинность первого выражения - `6 > 2` и не зависимости от результата проверки затем проверяет истинность второго выражения - `2 < 4`. Что у нас происходит в случае с `AndAlso`? Сначала также проверяется истинность первого выражения, и если оно истинно, тогда уже проверяется истинность второго. В случае если первое выражение ложно, тогда

нету смысла проверять истинность второго выражения, так как в любом случае итоговый результат будет False.

Все сказанное в отношении пары **And/AndAlso** характерно и для пары **Or/OrElse**: если оператор Or проверяет истинность правого и левого выражения, то оператор OrElse проверяет сначала истинность первого выражения, и только если оно False, проверяет на истинность второе выражение.

Поэтому в целях ускорения работы программы рекомендуется использовать **AndAlso** и **OrElse**. Однако иногда сокращенные логические операторы не могу употребляться: в поразрядных операциях. В таких операциях необязательно возвращается значение типа Boolean, оно может представлять и другой тип данных. Например:

```
Dim i As Integer
i = 4 And 5
```

Здесь числа 4 и 5 рассматриваются как двоичные:

число 4 в двоичной форме имеет представление 100,

а число 5 в двоичной форме имеет представление 101

Здесь действует та же логика, что и в логических операциях, только теперь сравниваются разряды чисел. В итоге это выражение возвращает нам число 4, потому что результат операции в двоичной форме будет иметь форму 100.

С помощью поразрядной операции Xor удобно применять примитивное шифрование:

```
Dim x As Byte = 102 'Пусть это будет ключ - в двоичной форме 1100110
Dim y As Byte = 45  'Значение, которое надо зашифровать - в двоичной форме 10
Dim z As Byte = y Xor x
Console.WriteLine(z) 'Результатом будет число 1001011 или 75
'Обратная операция - расшифровка
y = z Xor x
Console.WriteLine(y) 'Результатом будет исходное число 45
```

Операции сдвига

Еще один класс операций представляют операции поразрядного сдвига << и >>. Они имеют следующий синтаксис:

результат = число << (или >>) число разрядов

Сдвинем число 64 на два разряда влево и мы получим 256

```
Console.WriteLine(64 << 2)
```

Теперь сдвинем число 64 вправо также на два разряда и мы получим 16:

```
Dim num1  = 64 >> 2
Console.WriteLine(num1)
```

Несмотря на то, что с первого взгляда кажется, что данные операторы редко используются. нельзя не отметить их практическую ценность. Поскольку каждый сдвиг на один разряд вправо приравнивается к делению на 2, а сдвиг влево - к умножению на 2, то операции сдвига можно использовать вместо деления/умножения на степени двойки, тем более с точки зрения архитектуры компьютера операции сдвига будут отрабатывать быстрее, чем операции умножения и деления на 2.

Операции присваивания

Ну и в конце мы рассмотрим операции присваивания.

- =

  Присваивание

- ^=

  Присваивание после возведения в степень

- *=

  Присваивание после умножения

- /=

  Присваивание после деления

- \=

  Присваивание после целочисленного деления

- +=

  Присваивание после сложения

- -=

  Присваивание после вычитания

- >>=

  Присваивание после сдвига вправо

- <<=

  Присваивание после сдвига влево

- &=

  Присваивание после конкатенации двух значений

Большинство операций присваивания представляют сокращенную форму других операций. Так, выражение

```
x1=x1 + 7
```

эквивалентно следующему:

```
x1+=7
```

Подобным образом работают и другие операторы присваивания. Отдельно скажем про конкатенацию. Оператор &= предназначен для сцепления двух строк:

```
Dim s As String = "hello"
s&=" world" 'В итоге переменная s будет иметь значение "Hello world"
```

Если же эту операцию применить к числам, то они сначала будут преобразованы в строки, а затем эти строки будут подобным образом объединены:

```
Dim s As Integer = 6
s&=7 'В итоге переменная s будет иметь значение 67
```

Назад **Содержание** **Вперед**

Д **Деник Б**
6 месяцев назад edited

Доброго времени суток не подскажете что за операция '_' т.е. нижнее подчеркивание, перевожу код из basic на java, вот отрывок математического выражения

...Cos(B) * da _ + (N ^ 2 / a ^ 2 + 1)...

Не могу понять что за '_' между 'da' и '+'

0 0 Ответить • Поделиться ›

Metanit Модератор → Деник Б
6 месяцев назад

это часть названия переменной (или параметра)

0 0 Ответить • Поделиться ›

Д **Деник Б** → Metanit
6 месяцев назад

Нет это не так, переменная 'da' названа без '_', вообще это какой то знак перехода на новую строку или что то типа того, для java его достаточно просто удалить, оказалось так

0 0 Ответить • Поделиться ›

Metanit Модератор → Деник Б
6 месяцев назад

да, он может служить в качестве перевода кода на новую строку, но в этом случае он должен писаться отдельно от "da". А так "da_" - это вполне валидное название переменной

0 0 Ответить • Поделиться ›

Mnham
6 лет назад edited

Xor - исключающее "или"

Xor - исключающее или
Not - логическое отрицание

0 0 Ответить • Поделиться ›

Metanit Модератор → Mnham
6 лет назад
поправил

0 0 Ответить • Поделиться ›

E

Egor
7 лет назад
В VB.net есть оператор OrElse, а OrAlso не встречал.

0 0 Ответить • Поделиться ›

Metanit Модератор → Egor
7 лет назад
поправил

0 0 Ответить • Поделиться ›

A

Alexandr
8 лет назад
В тексте же можно использовать символ сложения, вместо амперсанда. Или есть разница? (для текста)

0 0 Ответить • Поделиться ›

Metanit Модератор → Alexandr
8 лет назад
по большому счету нет, хотя раньше по крайней мере больше использовался

Помощь сайту

YooMoney:
410011174743222

Qiwi:
qiwi.com/n/METANIT

Перевод на карту
Номер карты:
4048415020898850

Вконтакте | Телеграм | Twitter | Помощь сайту

Контакты для связи: metanit22@mail.ru