Оглавление

Введение 2

1. Логические константы и переменные 2

2. Операции сравнения 2

3. Логические операции 3

4. Логическое выражение 4

5. Условный оператор 5

6. Функция IIf 7

7. Оператор множественного ветвления ElseIf 7

8. Оператор выбора Select Case 9

9. Оператор безусловного перехода GoTo 12

10. Пример. Решение линейного уравнения 13

11. Пример. Программа-калькулятор 14

Приложение 1 18

Приложение 2 19

Приложение 3 19

Приложение 4 20

Приложение 5 20

Приложение 6 21

Приложение 7 21

Список литературы 23

Введение

Алгоритм, в котором последовательность выполнения некоторых предписаний зависит от выполнения проверяемых исполнителем условий, называется нелинейным. Очевидно, что подавляющее большинство алгоритмов являются нелинейными. Простейшим проявлением нелинейности является ветвление.

Ветвление представляет собой выбор пути решения задачи в соответствии с выполнением или невыполнением некоторого условия выбора. В Visual Basic 2005 ветвление реализуется с помощью условного оператора, а условие выбора записывается в форме логического (условного) выражения.

1. Логические константы и переменные

Логические константы и переменные – это константы и переменные, имеющие тип `Boolean`. `Boolean` – тип данных для хранения логических величин. Может иметь только два значения: `True` (Истина) и `False`(Ложь). При переводе числовых данных значений в логические значения ноль становится `False`, а все другие значения – `True`. При обратном преобразовании `False` становится нулем, а `True` – единицей. Идентификаторы `True` и `False` являются ключевыми словами Visual Basic 2005.

Как правило, логические переменные используются для хранения информации о состоянии какого-либо объекта. Например, отсортирован ли массив, найдено ли искомое значение и так далее.

2. Операции сравнения

Для сравнения двух однотипных величин в Visual Basic 2005 предусмотрено шесть различных операций сравнения. Все они имеют одинаковый приоритет. Знаки операций сравнений приведены в таблице 1.

Таблица 1

Знак операции сравнения	Пример использования	Описание операции
=	`a = b`	Равно. Операция возвращает значение `True`, если значения равны друг другу. Во всех остальных случаях результатом операции будет значение `False`.
< >	`a <> b`	Неравно. Операция возвращает значение `True`, если значения неравны друг другу. Иначе результатом операции будет значение `False`.

Оглавление

Знак операции сравнения	Пример использования	Описание операции
<	`a < b`	Меньше. Операция возвращает значение `True`, если значение переменной `a` меньше значения переменной `b`. Иначе результатом операции будет значение `False`.
<=	`a <= b`	Меньше или равно. Операция возвращает значение `True`, если значение переменной `a` меньше или равно значению переменной `b`. Иначе результатом операции будет значение `False`.
>	`a > b`	Больше. Операция возвращает значение `True`, если значение переменной `a` больше значения переменной `b`. Иначе результатом операции будет значение `False`.
>=	`a >= b`	Больше или равно. Операция возвращает значение `True`, если значение переменной `a` больше или равно значению переменной `b`. Иначе результатом операции будет значение `False`.

Обратите внимание, что если знак операции состоит из двух символов (например, неравно), то между символами пробел не ставится.

3. Логические операции

В Visual Basic 2005 реализованы четыре логических операции: `Not`, `And`, `Or`, `Xor`. Рассмотрим каждую их них.

`Not` – операция логического отрицания. Имеет высший приоритет среди логических операций. Изменяет логическое значение на противоположное. Отрицание Истины (`True`) будет Ложь (`False`). Отрицание Лжи (`False`) будет Истина (`True`).

`And` – операция логического умножения (логическое И). Имеет второй приоритет среди логических операций. Эта операция эквивалентна требованию одновременного выполнения обоих условий. Другими словами, операция `And` дает результат Истина (`True`), только если оба логических выражения, участвующих в операции, имеют значение Истина (`True`). Во всех остальных случаях результатом операции будет Ложь (`False`). Таким образом, Истина (`True`) И (`And`) Истина (`True`) будет Истина (`True`), остальное – Ложь (`False`).

`Or` – операция логического сложения (логическое ИЛИ). Имеет низший приоритет среди логических операций. Эта операция соответствует требованию выполнения хотя бы одного из двух условия, но допускает и одновременное

Оглавление

выполнение обоих условий. Другими словами, операция `Or` дает результат Ложь (`False`), только в том случае, если оба логических выражения, участвующих в операции, имеют значение Ложь (`False`). Во всех остальных случаях результатом операции будет Истина (`True`). Таким образом, Ложь (`False`) ИЛИ (`Or`) Ложь (`False`) будет Ложь (`False`), остальное – Истина (`True`).

`Xor` – исключающее ИЛИ. Имеет такой же приоритет, как и операция `Or`. Но в отличие от нее требует, чтобы выполнялось только одно условие. Если оба логических выражения, участвующих в операции `Xor`, имеют одинаковые значения, то результатом операции будет Ложь (`False`). Если логические выражения имеют различное значение, то результатом операции `Xor` будет Истина (`True`).

Все возможные значения логических операций приведены в таблице 2.

Таблица 2

A	B	Логические операции			
		Not A	A And B	A Or B	A Xor B
True	True	False	True	True	False
True	False	False	False	True	True
False	True	True	False	True	True
False	False	True	False	False	False

4. Логическое выражение

Логическое (условное) выражение может быть представлено в четырех видах: логическая константа, логическая переменная, простое условие или сложное условие. Любое логическое выражение может иметь только одно из двух значений: Истина (`True`) или Ложь (`False`).

Логические константы и переменные рассмотрены в разделе 1.

Простое условие – это два выражения, между которыми стоит знак операции сравнения. В роли выражений могут выступать числа, числовые переменные, математические функции, арифметические выражения, строки, строковые переменные, строковые функции и строковые выражения. Оба выражения, участвующие в сравнении, обязательно должны принадлежать к одному и тому же типу. Если простое условие выполняется, оно имеет значение Истина (`True`). В противном случае оно имеет значение Ложь (`False`).

Сложное условие – это последовательность простых условий, логических переменных и логических констант, которые соединены между собой знаками

Оглавление

логических операций. Традиционно каждую составную часть сложного условия берут в круглые скобки, хотя в Visual Basic 2005 это является необязательным.

5. Условный оператор

Условный оператор в Visual Basic 2005 предназначен для организации ветвлений. Существует две формы синтаксиса условного оператора: однострочный оператор и многострочный оператор.

Как следует из названия, однострочный оператор всегда записывается в одну строку. Он используется в тех случаях, когда ветви алгоритма содержат небольшое количество действий, чаще всего одно. Однострочный условный оператор имеет следующий синтаксис.

```
If Условное Выражение Then Оператор1 Else Оператор2
```

Многострочный условный оператор записывается в несколько строк. Причем распределение ключевых слов по строкам является обязательным и не может быть изменено. Рассмотрим синтаксис многострочного условного оператора.

```
If Условное Выражение Then
    Группа Операторов 1
Else
    Группа Операторов 2
End If
```

В отличие от однострочного условного оператора, ветви многострочного оператора могут содержать не одно, а несколько действий. Поэтому данная форма условного оператора применяется гораздо чаще однострочной формы. Ее традиционно рекомендуют использовать начинающим программистам.

Так как принцип работы условного оператора не зависит от его синтаксической формы, то в дальнейшем мы не будем делать разницы между однострочной и многострочной формами. Рассмотрим логику работы условного оператора.

Условный оператор позволяет в определенный момент времени выбрать дальнейший путь выполнения алгоритма. Переход к выбранному пути называется условным переходом. Существует два вида условного перехода: одинарный или двойной.

Одинарный условный переход (другое его название – обход) предполагает, что действия есть только в одной ветви алгоритма, а другая его ветвь пуста, то есть не содержит ни одного действия. Условный оператор, реализующий такой переход, не

Оглавление

имеет части, которая начинается с ключевого слова `Else`. Если `Условное Выражение` имеет значение Истина (`True`), то выполняются операторы, стоящие после ключевого слова `Then`. Если `Условное Выражение` имеет значение Ложь (`False`), то выполнение условного оператора завершается и программа продолжает свою работу с оператора, стоящего после условного.

При двойном условном переходе действия находятся в обеих ветвях алгоритма. Условный оператор, реализующий двойной условный переход, содержит как часть `Then`, так и часть `Else`. Если `Условное Выражение` имеет значение Истина (`True`), то выполняются операторы, стоящие в части `Then`. Если `Условное Выражение` имеет значение Ложь (`False`), то выполняются операторы, стоящие в части `Else`. Заметим, что части `Then` и `Else` никогда не могут быть выполнены одновременно. После завершения работы условного оператора выполнение программы продолжается с оператора, стоящего сразу после него.

В качестве примера рассмотрим четыре разных варианта вычисления модуля числа. Исходное число `a` вводится с помощью функции `InputBox`. Модуль числа записывается в переменную `b`. Ее значение выводится с помощью функции `MsgBox`. Здесь мы рассмотрим несколько фрагментов программного кода. Полный текст программы приведен в приложении 1.

$$b = |a| = \begin{cases} a, \text{ если } a \geq 0 \\ -a, \text{ если } a < 0 \end{cases}$$

- Одинарный условный переход, однострочный условный оператор.

```
b = a
If a < 0 Then b = -a
```

  Сначала предполагаем, что модуль числа равен самому числу. Если исходное число – отрицательное, то меняем знак числа на противоположный.

- Одинарный условный переход, многострочный условный оператор.

```
b = a
If a < 0 Then
    b = -a
End If
```

  Здесь рассуждения полностью аналогичны предыдущему случаю. Разница заключается в форме записи условного оператора.

- Двойной условный переход, однострочный условный оператор.

Оглавление

```
If a >= 0 Then b = a Else b = -a
```

Если исходное число неотрицательное, то модуль равен самому числу. В противном случае для вычисления модуля исходного числа необходимо изменить его знак.

- Двойной условный переход, многострочный условный оператор.

```
If a >= 0 Then
    b = a
Else
    b = -a
End If
```

Логика работы этого варианта такая же, как и у предыдущего. Но вместо однострочного условного оператора используется многострочный условный оператор.

6. Функция IIf

В языке существует еще один способ организации ветвления – функция `IIf`. Она имеет следующий синтаксис.

```
IIf(Условное выражение, Значение1, Значение2)
```

Функция проверяет истинность `Условного Выражения`. Если `Условное Выражение` имеет значение Истина (`True`), то функция `IIf` возвращает `Значение1`. Если `Условное Выражение` имеет значение Ложь (`False`), то функция `IIf` возвращает `Значение2`.

В качестве примера опять рассмотрим задачу вычисления модуля числа. Исходное число обозначим `a`, а результат вычислений – `b`.

```
b = IIf(a >= 0, a , -a)
```

Значение, записанное в переменную `b`, будет зависеть от знака числа `a`. Если исходное число неотрицательное, то значение переменной `b` будет совпадать со значением переменной `a`. В противном случае значение переменной `b` будет противоположно по знаку значению переменной `a`, хотя и будет совпадать с ним по модулю. Таким образом, приведенный фрагмент программы позволяет найти модуль исходного числа с помощью функции `IIf`.

7. Оператор множественного ветвления ElseIf

При решении сложных задач часто возникает ситуация, когда определенное действие (или набор действий) нужно выполнить после проверки не одного, а нескольких условий. В таких случаях используют конструкцию множественного

Оглавление

условного перехода. В Visual Basic 2005 эта конструкция реализуется с помощью оператора множественного ветвления ElseIf, имеющего следующий синтаксис.

```
If Условное Выражение 1 Then
    Операторы 1
ElseIf Условное Выражение 2 Then
    Операторы 2
ElseIf Условное Выражение 3 Then
    Операторы 3
. . .
ElseIf Условное Выражение N Then
    Операторы N
Else
    Операторы Else
End If
```

Рассмотрим логику работы этого оператора. Если `Условное Выражение 1` имеет значение Истина (`True`), то выполняется блок `Операторы 1`, значения остальных условных выражений не проверяются и соответствующие блоки не выполняются. Если `Условное Выражение 1` имеет значение Ложь (`False`), то проверяется значение `Условного Выражения 2`. Если оно – Истина (`True`), то выполняется блок `Операторы 2`. Иначе проверяется значение `Условного Выражения 3` и так далее. Если все условные выражения имеют значение Ложь (`False`), то выполняются операторы, стоящие в части `Else`, ели она существует (в Visual Basic 2005 часть `Else` является необязательной). После завершения работы оператора множественного ветвления выполнение программы продолжается с оператора, стоящего после ключевого словосочетания `End If`. Обратите внимание, что выполниться может только одна ветвь оператора, даже если истинными являются сразу несколько `Условных Выражений`. В таких случаях реализуется первая из подходящих ветвей оператора.

В качестве примера рассмотрим программу, вычисляющую значение функции в некоторой точке *x*, заданной пользователем. Функция имеет следующий вид.

Оглавление

$$y = \begin{cases} 2x, \text{если } x \in (-\infty;\ -10) \\ x, \text{если } x \in [-10;\ -1) \\ 1/x, \text{если } x \in [-1;\ 1] \\ x^2, \text{если } x \in (1;\ \infty) \end{cases}$$

Полный текст программы приведен в приложении 2.

Первым шагом алгоритма является проверка области допустимых значений. Данная функция не имеет значения в точке $x = 0$. Потому первое условие анализирует значение аргумента функции. Если x попадает в область допустимых значений, то последовательно проверяется, к какой именно части принадлежит значение аргумента функции. В зависимости от этого вычисляется значение функции.

```
If x = 0 Then
    MsgBox("Функция не определена")
ElseIf x < -10 Then
    y = 2 * x
    MsgBox("y=" + Str(y))
ElseIf x < -1 Then
    y = x
    MsgBox("y=" + Str(y))
ElseIf x <= 1 Then
    y = 1 / x
    MsgBox("y=" + Str(y))
Else
    y = x ^ 2
    MsgBox("y=" + Str(y))
End If
```

8. Оператор выбора Select Case

Оператор выбора применяется для программирования выбора одной из нескольких взаимоисключающих возможностей (альтернатив), при этом условием выбора ветви алгоритма является значение некоторого выражения. Оператор выбора имеет следующий синтаксис.

```
Select Case Выражение
    Case Список Значений 1
        Операторы 1
    Case Список Значений 2
```

Оглавление

```
        Операторы 2
    . . .
    Case Список Значений N
        Операторы N
    Case Else
        Операторы Else
End Select
```

Список значений – это последовательность выражений, разделенных запятым. Выражение может быть представлено в виде константы, числа, строки, переменной, арифметического, логического или строкового выражения. Также Список Значений может включать выражения специального вида: диапазон и луч.

Выражение-диапазон позволяет задать диапазон возможных значений. Оно имеет следующий синтаксис.

```
Выражение To Выражение
```

Например, диапазон `1 To 5` определяет отрезок от 1 до 5, включая границы. В Visual Basic 2005 границы всегда включаются в диапазон.

Выражение-луч используется, когда надо определить полуинтервал возможных значений. Оно имеет следующий синтаксис.

`Is` *Знак операции сравнения* Выражение

Например, луч `Is > 10` задает множество чисел, превышающих 10.

Рассмотрим логику работы оператора выбора. Сначала вычисляется значение Выражения, стоящего в заголовке оператора. Если это значение совпадет с одним из элементов Списка Значений 1, то выполнятся блок Операторы 1. В противном случае анализируются элементы Списка Значений 2. В случае совпадения выполняется блок Операторы 2. Иначе анализируется Список Значений 3 и так далее. Если значение Выражения, стоящего в заголовочной части оператора, не совпало ни с одним значением из всех Списков Значений, то выполняется блок Операторы Else, стоящий в части `Case Else`. Часть `Case Else` является необязательной и может отсутствовать. В этом случае оператор `Select Case` не будет выполнять никаких действий, а выполнение программы продолжится с оператора, стоящего после ключевого словосочетания `End Select`. Обратите внимание, что всегда выполняется только одна ветвь оператора выбора. Даже если

значение заголовочного Выражения попадает в несколько Списков Значений. В таких случаях реализуется первая из подходящих ветвей оператора.

Разберем два примера использования оператора выбора.

Пример 1. Необходимо составить программу, которая для заданного числа выводит его характеристику: ноль, однозначное четное, однозначное нечетное, от 10 до 20, больше 20, отрицательное.

Полный текст программы приведен в приложении 3.

В качестве заголовочного выражения возьмем анализируемую переменную a. И рассмотрим все возможные ее значения. Каждый вариант будем оформлять в виде отдельного блока Case.

```
Select Case a
    Case 0
        MsgBox("Ноль")
```

Когда значений в списке несколько, они перечисляются через запятую.

```
    Case 2, 4, 6, 8
        MsgBox("Однозначное четное")
    Case 1, 3, 5, 7, 9
        MsgBox("Однозначное нечетное")
```

Для обработки отрицательных чисел воспользуемся выражением типа луч.

```
    Case Is < 0
        MsgBox("Отрицательное")
```

Числа от 10 до 20 зададим с помощью диапазона.

```
    Case 10 To 20
         MsgBox("От 10 до 20")
```

Все остальные числа определим, используя блок Case Else.

```
    Case Else
        MsgBox("Больше 20")
End Select
```

В результате мы получили оператор выбора, который решает поставленную задачу, то есть, анализируя значение переменной, выводит характеристику числа.

Пример 2. Требуется составить программу, определяющую, какое из трех введенных чисел равно пяти. Предполагается, что все три введенных числа различны.

Полный текст программы приведен в приложении 4.

Оглавление

Если в первом примере в качестве заголовочного выражения мы использовали переменную, то в данной задаче удобнее использовать значение. Так как требуется определить, какое из чисел равно пяти, то заголовочное выражение оператора выбора будет записано в виде числа 5. А списки значений будут содержать переменные, которые будут сравниваться со значением заголовочного выражения.

```
Select Case 5
    Case a
        MsgBox("a=5")
    Case b
        MsgBox("b=5")
    Case c
        MsgBox("c=5")
    Case Else
        MsgBox("Ни одно из чисел неравно 5")
End Select
```

9. Оператор безусловного перехода GoTo

Оператор безусловного перехода используется для передачи управления на некоторую заранее заданную строку программы. Рассмотрим синтаксис оператора.

```
GoTo Метка
```

`GoTo` – ключевое слово Visual Basic. `Метка` – это последовательность символов, заканчивающаяся двоеточием. Метка может начинаться как с буквы, так и с цифры. Она может содержать символы латинского и русского алфавитов, цифры и знак подчеркивания. В метке нельзя использовать скобки, пробелы, знаки пунктуации и арифметических операций. Традиционно рекомендуется составлять метки, пользуясь правилом имен. Метка всегда ставится на отдельной строке перед каким-либо оператором программного кода. Помеченный оператор будет выполняться сразу после оператора `GoTo`. Такой способ передачи управления называется безусловным переходом, так как он выполняется без проверки каких-либо условий. Передавать управление таким способом можно как вперед, так и назад по тексту программы.

Строго доказано, что программу любой сложности можно написать, не применяя оператора безусловного перехода. Поэтому широкое использование оператора `GoTo` считается «дурным тоном», так как они запутывают программу и затрудняют ее чтение.

Оглавление

Рассмотрим пример использования оператора безусловного перехода.

Требуется составить программу для ввода значения переменной n, которое должно находится в диапазоне от 3 до 20. При неправильном значении переменной программа должна выводить сообщение и требовать повторного ввода значения.

Полный текст программы приведен в приложении 5.

В начале программы ставим метку vvod. Она позволит нам повторить ввод значения переменной в случае ошибки.

```
vvod:
```

Затем вводим значение переменной n.

```
n = Val(InputBox("Введите число n от 3 до 20"))
```

Проверяем введенное значение.

```
If n < 3 Or n > 20 Then
```

Если оно меньше 3 или больше 20, то выводим сообщение об ошибке.

```
    MsgBox("Неправильное значение")
```

И передаем управление на оператор с меткой Vvod, чтобы обеспечить повторный ввод значения переменой и его проверку.

```
    GoTo Vvod
End If
```

10. Пример. Решение линейного уравнения

Рассмотрим программу решения уравнения вида $ax+b=c$. Значения параметров *a, b* и *c* задаются пользователем с клавиатуры и могут быть любыми.

Для начала проанализируем, какие возможны сочетания значений параметров, и каким при этом будет решение уравнения.

- Если коэффициент *a* неравен нулю, то решение уравнения можно найти по формуле $x=\frac{c-b}{a}$.
- Если коэффициент *a* равен нулю, то нужно получается выражение вида $b=c$. Если значения параметров *b* и *c* совпадают, то это выражение будет верным при любом *x*. Если значения параметров *b* и *c* различны, то выражение не имеет смысла.

Теперь можно перейти к разработке интерфейса программы. Так как значения коэффициентов мы будем вводить с помощью функции InputBox, а вывода результатов будем использовать функцию MsgBox, то на форме располагается только кнопка «Старт», запускающая нашу программу.

Оглавление

Следующий шаг – реализация алгоритма. Первым шагом является описание переменных. Так как значения коэффициентов могут быт любыми, то переменные a, b, c должны иметь тип Single. Значение переменной x определяется в результате деления, поэтому она тоже должна иметь тип Single.

```
Dim a, b, c, x As Single
```

С помощью функции InputBox вводим значения коэффициентов уравнения. Так как коэффициенты являются числами, то при их вводе необходимо использовать преобразование Val.

```
a = Val(InputBox("Введите коэффициент a"))
b = Val(InputBox("Введите коэффициент b"))
c = Val(InputBox("Введите коэффициент c"))
```

Анализируем значение коэффициента a.

```
If a = 0 Then
```

Если оно равно нулю, то проверяем равенство двух других коэффициентов и выводим соответствующее сообщение.

```
    If b = c Then
        MsgBox("Решение - любое число")
    Else
        MsgBox("Нет корней")
    End If
Else
```

Если коэффициент a неравен нулю, то вычисляем корень уравнения и выводим его. Так как корень уравнения является числом, то для его вывода необходимо использовать преобразование Str.

```
    x = (c - b) / a
    MsgBox("x=" + Str(x))
End If
```

Программа закончена. Теперь ее необходимо сохранить и запустить. Текст программы приведен в приложении 6.

11. Пример. Программа-калькулятор

Рассмотрим программу, имитирующую работу калькулятора. Исходными данными являются два числа и знак операции. Программа должна выполнять следующие действия: сложение (+), вычитание (-), умножение (*), деление (/),

Оглавление

вычисление остатка от деления (%), целочисленное деление (\). Также необходимо реализовать повтор ввода исходных данных и вычислений по желанию пользователя.

Так же как в предыдущей программе для ввода исходных данных будем использовать функцию `InputBox`, а дл вывода результатов – функцию `MsgBox`. Поэтому на форме расположен только один элемент управления – это кнопка, запускающая программу.

Разработку программы начнем с описания переменных. Переменные `a` и `b` предназначены для хранения пары чисел, над которыми выполняется операция. Так как исходные числа могут быть любыми, то для их хранения необходимо использовать рациональный тип данных. Например, `Single`. В переменной c будет храниться результат арифметической операции. Так как все действия мы будем выполнять над рациональными числами, то результат тоже будет рациональным числом. Поэтому переменная `c` тоже будет иметь тип `Single.`

```
Dim a, b, c As Single
```

Переменная `znak` предназначена для хранения знака операции. Знак операции – это символ. Поэтому переменная `znak` имеет тип `Char.`

```
Dim znak As Char
```

Переменная `otvet` предназначена для хранения результата функции `MsgBox.` Эта переменная позволит нам определить, желает ли пользователь повторить вычисления.

```
Dim otvet As Integer
```

В начале программы ставим метку `nachalo.` Она позволит нам организовать повтор ввода значений и вычисления результата.

```
nachalo:
```

Вводим исходные числа. Как всегда при вводе числовой информации используем преобразование `Val.`

```
a = Val(InputBox("Введите число a"))
b = Val(InputBox("Введите число b"))
```

Вводим знак операции. Так как знак операции является символом, то при его вводе дополнительных преобразований не требуется.

```
znak = InputBox("Введите знак операции (+ - * / % \")
```

Анализируем знак операции. В зависимости от его значения выполняем те или иные вычисления. Обратите внимание, что символы указываются в кавычках.

```
Select Case znak
```

Оглавление

```
        Case "+"
```

Здесь выполняется операция сложения.

```
            c = a + b
            MsgBox("c=" + Str(c))
        Case "-"
```

Здесь выполняется операция вычитания.

```
            c = a - b
            MsgBox("c=" + Str(c))
        Case "*"
```

Здесь выполняется операция умножения.

```
            c = a * b
            MsgBox("c=" + Str(c))
        Case "/"
```

Здесь выполняется операция деления. Но перед вычислением необходимо проверить, равен ли знаменатель нулю.

```
            If b = 0 Then
```

Если в переменной `b` хранится ноль, то деление невозможно. Поэтому вместо ответа выводится сообщение.

```
                MsgBox("Ошибка! Деление на ноль")
            Else
```

В противном случае вычисляется частное чисел `a` и `b` и выводится результат.

```
                c = a / b
                MsgBox("c=" + Str(c))
            End If
        Case "\"
```

Операция целочисленного деления выполняется по таким же правилам, как и простое деление. Так как исходные переменные рациональные, то перед выполнением целочисленного деления Visual Basic 2005 автоматически произведет математическое округление чисел до целых.

```
        If b = 0 Then
                MsgBox("Ошибка! Деление на ноль")
            Else
                c = a \ b
                MsgBox("c=" + Str(c))
```

Оглавление

```
            End If
        Case "%"
```

При вычислении остатка от деления действуют те же правила, что и при делении. Исходные числа, так же как и в предыдущем случае, автоматически округляются до целых.

```
            If b = 0 Then
                MsgBox("Ошибка! Деление на ноль")
            Else
                c = a Mod b
                MsgBox("c=" + Str(c))
            End If
        Case Else
```

Если выполнение программы попало в этот блок, значит, знак операции введен с ошибкой. Поэтому здесь не выполняются никакие вычисления, а выводится поясняющее сообщение.

```
            MsgBox("Неизвестный знак операции")
    End Select
```

После окончания вычислений спрашиваем пользователя, желает ли он выполнить еще один расчет, и предлагаем ему два варианта ответа: «Да» и «Нет». Для этого в функции `MsgBox` второй параметр задаем в виде `32 + 4`. `32` определяет пиктограмму со знаком вопроса, а `4` – набор кнопок «Да» и «Нет». Результат функции `MsgBox` записываем в переменную `otvet` для дальнейшего анализа.

```
    otvet = MsgBox("Выполнить еще один расчет?", 32 + 4)
```

Анализируем ответ пользователя. Если он нажал кнопку «Да», то в переменную `otvet` будет записано число 6. В этом случае надо передать управление в начало программы. Для этого мы заранее поставили там метку `nachalo.`

```
    If otvet = 6 Then
        GoTo nachalo
    End If
```

Программа закончена. Ее текст приведен в приложении 7.

Оглавление

Приложение 1

Вычислить модуля рационального числа с использованием различных видов условных переходов и условных операторов.

```
Dim a, b As Single
a = Val(InputBox("введите исходное число"))

'Одинарный условный переход, однострочный условный оператор
b = a
If a < 0 Then b = -a
MsgBox("Одинарный условный переход, однострочный условный
оператор" + vbNewLine + "b=" + Str(b))

'Одинарный условный переход, многострочный условный оператор
b = a
If a < 0 Then
    b = -a
End If
MsgBox("Одинарный условный переход, многострочный условный
оператор" + vbNewLine + "b=" + Str(b))

'Двойной условный переход, однострочный условный оператор
If a >= 0 Then b = a Else b = -a
MsgBox("Двойной условный переход, однострочный условный
оператор" + vbNewLine + "b=" + Str(b))

'Двойной условный переход, многострочный условный оператор
If a >= 0 Then
    b = a
Else
    b = -a
End If
MsgBox("Двойной условный переход, многострочный условный
оператор" + vbNewLine + "b=" + Str(b))
```

Оглавление

Приложение 2

Вычислить значение функции в точке x, заданной пользователем.

$$y = \begin{cases} 2x, \text{если } x \in (-\infty;\ -10) \\ x, \text{если } x \in [-10;\ -1) \\ 1/x, \text{если } x \in [-1;\ 1] \\ x^2, \text{если } x \in (1;\ \infty) \end{cases}$$

```
Dim x, y As Single
x = Val(InputBox("Введите значение переменной x"))
If x = 0 Then
    MsgBox("Функция не определена")
ElseIf x < -10 Then
    y = 2 * x
    MsgBox("y=" + Str(y))
ElseIf x < -1 Then
    y = x
    MsgBox("y=" + Str(y))
ElseIf x <= 1 Then
    y = 1 / x
    MsgBox("y=" + Str(y))
Else
    y = x ^ 2
    MsgBox("y=" + Str(y))
End If
```

Приложение 3

Составьте программу, которая для заданного числа выводит его характеристику: ноль, однозначное четное, однозначное нечетное, от 10 до 20, больше 20, отрицательное.

```
Dim a As Integer
a = Val(InputBox("Введите число"))
Select Case a
    Case 0
        MsgBox("Ноль")
    Case 2, 4, 6, 8
```

Оглавление

```
        MsgBox("Однозначное четное")
    Case 1, 3, 5, 7, 9
        MsgBox("Однозначное нечетное")
    Case Is < 0
        MsgBox("Отрицательное")
    Case 10 To 20
         MsgBox("От 10 до 20")
    Case Else
        MsgBox("Больше 20")
End Select
```

Приложение 4

Составьте программу, определяющую, какое из трех введенных чисел равно пяти. Предполагается, что все три введенных числа различны.

```
Dim a, b, c As Integer
a = Val(InputBox("Введите число a"))
b = Val(InputBox("Введите число b"))
c = Val(InputBox("Введите число c"))
Select Case 5
    Case a
        MsgBox("a=5")
    Case b
        MsgBox("b=5")
    Case c
        MsgBox("c=5")
    Case Else
        MsgBox("Ни одно из чисел неравно 5")
End Select
```

Приложение 5

Составьте программу для ввода значения переменной n, которое должно находится в диапазоне от 3 до 20. При неправильном значении переменной программа должна выводить сообщение и требовать повторного ввода значения.

```
Dim n As Integer
vvod:
```

Оглавление

```
n = Val(InputBox("Введите число n от 3 до 20"))
If n < 3 Or n > 20 Then
    MsgBox("Неправильное значение")
    GoTo Vvod
End If
```

Приложение 6

Составьте программу для решения уравнения вида $ax+b=c$. Значения параметров a, b и c задаются пользователем с клавиатуры и могут быть любыми.

```
Dim a, b, c, x As Single
a = Val(InputBox("Введите коэффициент a"))
b = Val(InputBox("Введите коэффициент b"))
c = Val(InputBox("Введите коэффициент c"))
If a = 0 Then
    If b = c Then
        MsgBox("Решение - любое число")
    Else
        MsgBox("Нет корней")
    End If
Else
    x = (c - b) / a
    MsgBox("x=" + Str(x))
End If
```

Приложение 7

Программа, имитирующая работу калькулятора. Вводятся два числа и знак операции. Программа должна выполнять следующие действия: сложение (+), вычитание (-), умножение (*), деление (/), вычисление остатка от деления (%), целочисленное деление (\). Также необходимо реализовать повтор ввода исходных данных и вычислений по желанию пользователя.

```
Dim a, b, c As Single
Dim znak As Char
Dim otvet As Integer

nachalo:
```

Оглавление

```
a = Val(InputBox("Введите число a"))
b = Val(InputBox("Введите число b"))
znak = InputBox("Введите знак операции (+ - * / % \")
Select Case znak
    Case "+"
        c = a + b
        MsgBox("c=" + Str(c))
    Case "-"
        c = a - b
        MsgBox("c=" + Str(c))
    Case "*"
        c = a * b
        MsgBox("c=" + Str(c))
    Case "/"
        If b = 0 Then
            MsgBox("Ошибка! Деление на ноль")
        Else
            c = a / b
            MsgBox("c=" + Str(c))
        End If
    Case "\"
        If b = 0 Then
            MsgBox("Ошибка! Деление на ноль")
        Else
            c = a \ b
            MsgBox("c=" + Str(c))
        End If
    Case "%"
        If b = 0 Then
            MsgBox("Ошибка! Деление на ноль")
        Else
            c = a Mod b
            MsgBox("c=" + Str(c))
```

Оглавление

```
        End If
    Case Else
         MsgBox("Неизвестный знак операции")
End Select
otvet = MsgBox("Выполнить еще один расчет?", 32 + 4)
If otvet = 6 Then
    GoTo nachalo
End If
```

Список литературы

1. Волчёнков Н.Г. Программирование на Visual Basic 6: В 3-х ч. Часть 1. – М.: ИНФРА-М, 2000. – 286 с.
2. Шевякова Д.А., Степанов А.М., Карпов Р.Г. Самоучитель Visual Basic 2005 / под общ. ред. А.Ф. Тихонова. – СПб.: БХВ-Петербург, 2007. – 576 с.
3. Богданов М.Р. Visual Basic 2005 на примерах. – СПб.: БХВ-Петербург, 2007. – 592 с.

Оглавление