Оглавление

Введение

Алгоритм, состоящий из одного или нескольких действий, которые должны быть выполнены строго последовательно, без всяких условий и в строгом соответствии с тем порядком, в котором записаны операторы программы, называется линейным.

Программа, реализующая линейной алгоритм, называется программой линейной структуры. Последовательность строк этой программы при выполнении сверху вниз соответствует шагам алгоритма решения задачи.

1. Оформление текста программы

В каждой строке программного кода помещается только один шаг алгоритма (один оператор). Если в одной строке надо поместить несколько операторов, то они разделяются двоеточием (:). Если оператор очень длинный, то его можно разбить на части, используя знаки переноса. Знак переноса состоит из двух символов: пробела и знака подчеркивания. Он ставится в любом месте оператора, где можно вставить пробел.

Оператор – это синтаксическая единица языка программирования, которая используется в программе для выполнения отдельного предписания (шага алгоритма). Операторы делятся на две группы: алгоритмические и функциональные. Алгоритмические операторы используются для организации последовательности выполнения действий. Функциональные операторы непосредственно реализуют действия, указанные в алгоритме.

Для пояснения текста программы используются комментарии. Комментарий начинается со знака апостроф (’) и продолжается до конца строки. Комментарии не выполняются, и их количество не влияет на скорость работы программы. Если нужно закомментировать большой фрагмент программного кода, то его выделить с помощью мыши и нажать кнопку **Comment out the selected lines** (), расположенную на панели инструментов **Text Editor**. Для удаления комментариев используется кнопка **Uncomment the selected lines** (), расположенная там же.

2. Понятие переменной

Переменная – это элемент программы, предназначенный для хранения данных в процессе выполнения программы. Переменная представляет собой зарезервированное место в оперативной памяти для временного хранения данных. Каждая переменная имеет имя и значение. Имя переменной уникально и не может меняться в процессе

Оглавление

выполнения программы. Значение переменной может многократно меняться в процессе выполнения программы.

Имя переменной – это строка символов, которая отличает эту переменную от других элементов программы. Иначе имя переменной называют идентификатор (от английского identify – распознавать, устанавливать идентичность). Имя переменной задается программистом. Оно должно подчиняться правилу имен и быть уникальным. Правило имен состоит из шести следующих пунктов.

1. В имени переменной можно использовать буквы[1], цифры и знак подчеркивания.
2. Первым символом имени должна быть буква.
3. Остальные символы имени – буквы, цифры и знак подчеркивания.
4. Имя переменной не должно содержать пробелы, скобки, знаки препинания и математических операций.
5. Длина имени не должна превышать 255 символов.
6. Имя переменной не должно совпадать ни с одним ключевым словом Visual Basic 2005.

Значение переменной – это данные, которые хранятся и обрабатываются в процессе выполнения программы. Способ обработки и хранения данных зависит от того, к какому типу данных принадлежит значение переменной.

3. Типы данных

Типом данных называется способ хранения и представления данных в компьютерной систем. В Visual Basic 2005 выделено 16 типов данных, которые делятся на две основные группы: числовые и нечисловые. Числовые типы данных предназначены для хранения и обработки чисел. Они в свою очередь делятся на целые типы и рациональные. Нечисловые типы данных предназначены для хранения нечисловой информации. В эту категорию попадают символьные типы данных, логические и прочие. Полная классификация типов данных приведена на рис. 1.

[1] Допускается использование в именах как латинских, так и русских букв. Но желательно ограничиться только латинским алфавитом, чтобы уменьшить количество ошибок в программе.

Оглавление

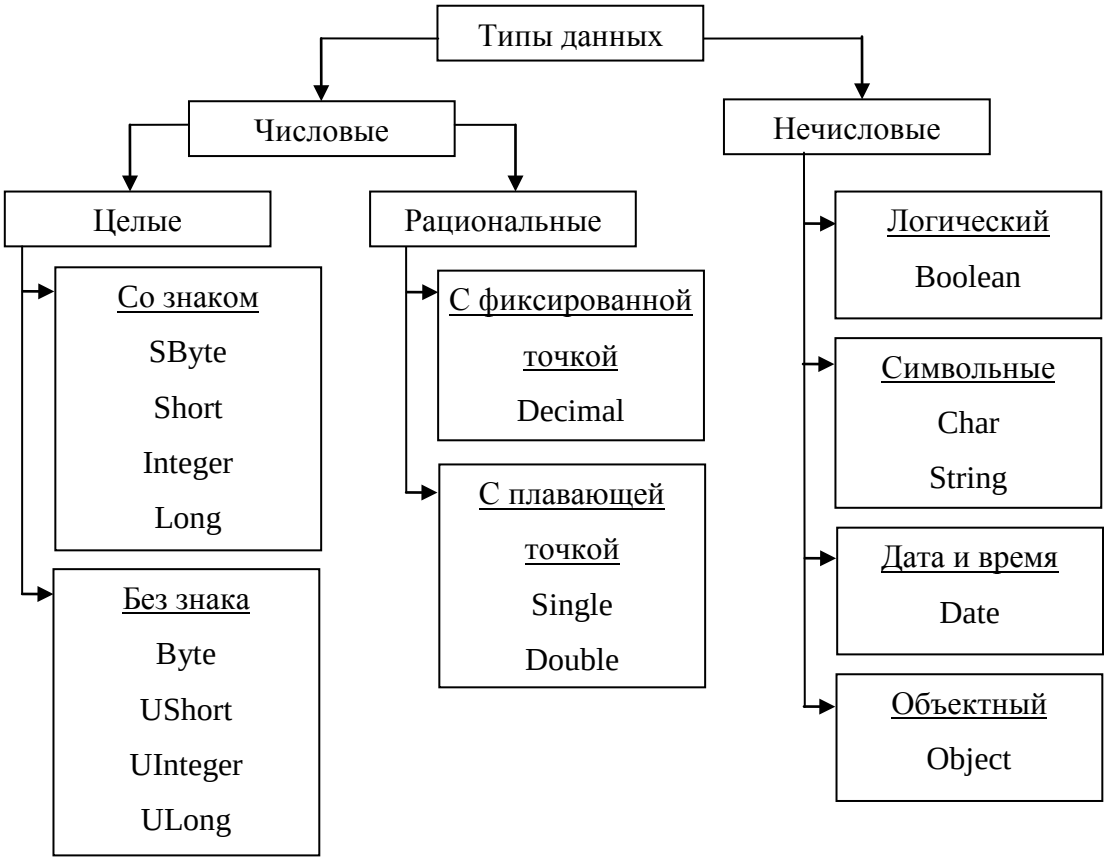

Рис. 1. Классификация типов данных

Рассмотрим характеристики каждого типа данных.

1. **Byte** – целое число без знака. Тип предназначен для хранения небольших целых неотрицательных чисел. Занимает 1 байт памяти. Диапазон значений от 0 до 255.
2. **UShort** – целое число без знака. Тип предназначен для хранения целых неотрицательных чисел. Занимает 2 байта памяти. Диапазон значений от 0 до 65 535.
3. **UInteger** – целое число без знака. Тип предназначен для хранения больших целых неотрицательных чисел. Занимает 4 байта памяти. Диапазон значений от 0 до 4 294 967 295.
4. **ULong** – целое число без знака. Тип предназначен для хранения очень больших целых неотрицательных чисел. Занимает 8 байт памяти. Диапазон значений от 0 до 18 446 744 073 709 551 615.
5. **SByte** – целое число со знаком. Тип предназначен для хранения небольших целых чисел, как отрицательных, так и положительных. Занимает 1 байт памяти. Диапазон значений от -128 до 127.

Оглавление

6. **Short** – целое число со знаком. Тип предназначен для хранения положительных и отрицательных целых чисел. Занимает 2 байта памяти. Диапазон значений от -32 768 до 32 767.
7. **Integer** – целое число со знаком. Тип предназначен для хранения больших целых чисел, как отрицательных, так и положительных. Занимает 4 байта памяти. Диапазон значений от -2 147 483 648 до 2 147 483 647.
8. **Long** – целое число со знаком. Тип предназначен для хранения очень больших целых чисел, как отрицательных, так и положительных. Занимает 8 байт памяти. Диапазон значений от -9 233 372 036 854 775 808 до 9 233 372 036 854 775 807.
9. **Single** – рациональное число одинарной точности с плавающей точкой. Число с плавающей точкой представляется в виде произведения мантиссы и 10 в некоторой степени. Мантисса – это рациональное число в диапазоне $(-10;10)$. Например, запись 1.234E2 соответствует рациональному числу $1.234\cdot10^2=1.234\cdot100=123.4$. Число с плавающей точкой может иметь и отрицательный показатель степени. Например, $4.56\text{E}-3=4.56\cdot10^{-3}=4.56\cdot0.001=0.00456$. Такая форма записи позволяет хранить в одной переменной как очень большие, так и очень маленькие числа. Одинарная точность означает, что в дробной части мантиссы 7 цифр. В Visual Basic 2005 дробная часть отделяется от целой точкой (а не запятой). Переменная этого типа занимает 4 байта памяти. Диапазон отрицательных чисел от -3.4028235E38 до -1.401298E-45. Диапазон положительных чисел от 1.401298E-45 до 3.4028235E38.
10. **Double** – рациональное число двойной точности с плавающей точкой. Двойная точность означает, что в дробной части мантиссы 17 цифр. Переменная этого типа занимает 8 байт памяти. Диапазон отрицательных чисел от -1.79769313486231570E308 до -4.94065645841246544E-324. Диапазон положительных чисел от 4.94065645841246544E-324 до 1.79769313486231570E308.
11. **Decimal** – рациональное число с фиксированной точкой. В отличие от чисел с плавающей точкой, числа данного типа не имеют множителя «десять в степени…» Это позволят избежать ошибок округления. Такие числа применяются для очень точных расчетов, например, финансовых. Дробная часть такого числа содержит 28 знаков. Занимает 16 байт памяти. Диапазон

Оглавление

значений от -7,9228162514264337593543950335 до 7,9228162514264337593543950335.

12. **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. Объем памяти, занимаемой переменной этого типа, различен и зависит от операционной системы и особенностей установки Microsoft Visual Studio.
13. **Char** – этот тип предназначен для хранения одного символа в формате Unicode. Занимает 2 байта памяти.
14. **String** – тип данных для хранения текстовой информации. В одной переменной этого типа может храниться строка длиной от 0 до примерно 2 миллиардов символов в кодировке Unicode. Объем памяти для хранения таких переменных зависит от длины строки и может меняться в процессе выполнения программы.
15. **Date** – предназначен для хранения информации о дате и времени. Занимает 8 байт памяти. Минимальное значение даты – 1 января 0001 года. Максимальное значение даты – 31 декабря 9999 года. Минимальное значение времени – 0:00:00. Максимальное значение времени – 23:59:59.
16. **Object** – может хранить различные данные и менять их тип во время выполнения программы. Как правило, используется для хранения ссылок на объекты, в частности, на элементы управления. Объем памяти зависит от информации, хранящейся в переменной.

4. Объявление переменных

Объявление переменной означает указание имени и типа переменной перед ее использованием. Это можно сделать с помощью четырех различных операторов:

```
Dim Имя Переменной As Тип Данных
Private Имя Переменной As Тип Данных
Public Имя Переменной As Тип Данных
Static Имя Переменной As Тип Данных
```

Переменная, объявленная с помощью оператора `Dim`, называется локальной. Она доступна только в пределах области, содержащей оператор `Dim`. Как только выполнение программы выходит за границы этой области, объявленная переменная перестает существовать.

Оглавление

Объявление переменной с помощью оператора `Public` означает, что данная переменная является глобальной или, другими словами, общедоступной. Она доступна во всех областях, подпрограммах и модулях, входящих в состав проекта. Visual Basic 2005 не накладывает никаких ограничений на доступ к этой переменной. Оператор `Public` нельзя использовать внутри подпрограмм.

Оператор `Private` используется для объявления переменной, доступной только в пределах того модуля, в котором она объявлена. Такая переменная является глобальной в пределах своего модуля, но по отношению ко всему проекту она будет локальной, то есть недоступной из других его частей. Оператор `Private` нельзя использовать внутри подпрограмм.

Переменная, описанная как `Static`, является локальной, то есть доступной только в пределах той подпрограммы, где она объявлена. Но в отличие от `Dim`-переменной, она сохраняет свое значение и после завершения подпрограммы, хотя и недоступна из других частей проекта. Оператор `Static` можно использовать только внутри подпрограммы.

Если одновременно требуется объявить несколько переменных одного и того же типа, то их имена перечисляются через запятую. В конце перечисления запятую ставить не надо.

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

В пределах одного оператора можно описывать несколько переменных различного типа.

```
Dim s as String, d as Char
```

5. Оператор присваивания

Оператор присваивания является простейшим функциональным оператором. Он позволяет записать значение в переменную. Так же говорят, что оператор присваивания позволяет задать значение переменной. Оператор присваивания обозначается знаком равно (=). Слева от оператора указывается имя переменной, а справа – значение.

```
Имя переменной = Значение переменной
```

Значение переменной может быть задано явно, в виде выражения или с помощью другой переменной. При явном задании значения оператор присваивания имеет следующий вид.

```
a = 5.78
```

Оглавление

Если значение задается выражением, то справа от оператора присваивания записывается выражение, тип которого должен соответствовать типу переменной, стоящей слева от оператора присваивания. При выполнении такого оператора сначала вычисляется значение выражения, а потом вычисленное значение записывается в переменную.

```
b = 2 + a / 3
```

Если значение переменной задается с помощью другой переменной, то оператор присваивания выглядит так:

```
c = a
```

При этом происходит копирование значения переменной a в переменную c. И никогда наоборот. В таких случаях говорят, что

оператор присваивания работает справа налево.

Это означает, что значение переписывается из переменной, стоящей справа от знака равенства, в переменную стоящую слева от знака равенства.

В качестве примера рассмотрим фрагмент программы, которая меняет местами значения двух переменных: a и b. Полный текст программы приведен в приложении 1.

Для того чтобы поменять местами значения двух переменных нам понадобится третья переменная, в которой будет временно сохранено значение одной из исходных переменных. Поэтому в операторе Dim объявляются три переменных.

```
Dim a, b, c As Integer
```

Затем в третью переменную c сохраняется значение первой переменной a. Это делается, потому что при записи нового значения в переменную a, ее старое значение будет потеряно.

```
c = a
```

На место старого значения переменной a записываем значение переменной b.

```
a = b
```

А на место старого значения переменной b записываем значение, сохраненное в переменной c.

```
b = c
```

Таким образом, в результате трех операторов присваивания значение, которое изначально хранилось в переменной в переменной b, оказывается в переменной a. А

значение из переменной a перемещается в переменную b. Этот прием называется обмен значений двух переменных с использованием третьей. При использовании этого приема важно помнить, что все три переменные обязательно должны иметь одинаковый тип данных.

6. Константы

Константа – это величина, значение которой не может меняться в процессе выполнения программы. Значение константы определяется один раз. Оно остается неизменным в течение всего времени работы программы.

Объявление константы похоже на объявление переменной. Константы могут быть глобальными и локальными. Глобальные константы доступны из всех частей модуля, в котором они описаны. Локальные константы доступны только в той подпрограмме, где они были объявлены. Для описания константы используется следующая конструкция.

```
Const Имя константы As Тип данных = Значение
```

Например.

```
Const n As Integer = 20
Const FileName As String = "D:\text.txt"
```

7. Арифметические операции

В Visual Basic 2005 используется восемь арифметических операций. Они имеют различный приоритет, который определяет порядок вычисления арифметического выражения.

Высшим приоритетом обладает операция возведения в степень. Она записывается следующим образом:

```
C = A ^ B
```

Значение переменной C получается в результате возведения значения переменной A в степень, показатель которой хранится в переменной B.

Следующая по старшинству операция – это перемена знака. Ее другое название – унарный минус. В результате выполнения оператора

```
C = -A
```

в переменную C будет записано значение переменной A с противоположным знаком.

Третья группа операций – мультипликативные операции. Она состоит из четырех следующих операций.

- Умножение:

```
C = A * B
```

Оглавление

- Деление:

```
C = A / B
```

- Целочисленное деление:

```
C = A \ B
```

Эта операция применяется только к целым числам. Она позволяет вычислить целую часть от деления числа A на число B. Например, `8 \ 3 = 2`.

- Остаток от деления:

```
C = A Mod B
```

Эта операция применяется только к целым числам. Она используется для определения остатка от деления значения переменной A на значение переменной B. Например, `7 Mod 3 = 1`. При использовании этой операции необходимо ставить пробелы до и после слова `Mod`.

Четвертая группа операций обладает саамы низким приоритетом. Она включает в себя аддитивные операции.

- Сложение: `C = A + B`
- Вычитание: `C = A – B`

В программировании достаточно часто используется прием, когда значение переменной изменяется на некоторую величину, и новое значение записывается обратно в ту же переменную вместо старого. Например: `k = k + 1`. В таких случаях можно использовать сокращенную запись арифметической операции: `k += 1`. Все виды сокращенных записей арифметических операций, используемых в Visual Basic 2005, приведены в таблице 1.

Таблица 1

Сокращенная запись	Полная запись	Описание
`k ^= Значение`	`k = k ^ Значение`	Значение переменной k возвести в указанную степень, и записать на место старого значения.
`k *= Значение`	`k = k * Значение`	Значение переменной k умножить на заданное число и записать на место старого значения.
`k /= Значение`	`k = k / Значение`	Значение переменной k разделить на заданное число и записать на место старого значения.
`k \= Значение`	`k = k \ Значение`	На место старого значения переменной k записать целую часть от деления старого значения переменной k на указанное число.

Оглавление

Сокращенная запись	Полная запись	Описание
`k += Значение`	`k = k + Значение`	К старому значению переменной k добавить указанное число и записать результат в переменную k.
`k -= Значение`	`k = k - Значение`	Из старого значения переменной k вычесть указанное число и записать результат в переменную k.

8. Математические функции

По определению функцией одного или нескольких аргументов называется некоторое правило, которое ставит в соответствие одному набору значений аргументов из области допустимых значений ровно одно значение самой функции. В Visual Basic 2005 аргументы функции всегда указываются в круглых скобках, даже если у функции один аргумент (в отличие от математической записи тех же функций).

Все математические функции, входящие в Visual Basic 2005 собраны в библиотеке `Math`. Поэтому названия всех функций начинаются с общей части «`Math.`». В таблице 2 приведены основные математические функции, существующие в Visual Basic 2005.

Таблица 2

Математическая запись	Запись в Visual Basic 2005	Описание
$y = \|x\|$	`y = Math.Abs(x)`	Функция вычисления модуля числа *x*.
	`y = Math.Sign(x)`	Функция определения знака числа *x*. Функция возвращает единицу, если аргумент положительный, минус единицу, если аргумент функции отрицательный, и ноль, когда аргумент равен нулю.
$y = \cos x$	`y = Math.Cos(x)`	Функция вычисления косинуса.
$y = \sin x$	`y = Math.Sin(x)`	Функция вычисления синуса.
$y = \mathrm{tg}\, x$	`y = Math.Tan(x)`	Функция вычисления тангенса.
$y = \arccos x$	`y = Math.ACos(x)`	Функция вычисления арккосинуса.
$y = \arcsin x$	`y = Math.ASin(x)`	Функция вычисления арксинуса.
$y = \mathrm{arctg}\, x$	`y = Math.ATan(x)`	Функция вычисления арктангенса.
$y = \ln x$	`y = Math.Log(x)`	Функция вычисления

Оглавление

Математическая запись	Запись в Visual Basic 2005	Описание
		натурального логарифма (логарифма по основанию *e*).
$y = \lg x$	`y = Math.Log10(x)`	Функция вычисления десятичного логарифма (логарифма по основанию 10).
$y = e^x$	`y = Math.Exp(x)`	Функция вычисления экспоненты (возведения числа *e* в степень *x*).
$y = \sqrt{x}$	`y = Math.Sqrt(x)`	Функция извлечения квадратного корня.
	`y = Math.Round(x)`	Функция математического округления. `Math.Round(2.3)` даст результат 2. `Math.Round(2.5)` даст результат 3.
	`y = Math.Truncate(x)`	Функция возвращает целую часть числа. `Math.Truncate(2.3)` дает результат 2. `Math.Truncate(2.9)` дает результат 2.

Кроме математических функций библиотека `Math` содержит две математические константы:

- `Math.PI` – число π = 3.1415926;
- `Math.E` – число *e* = 2.7182818.

9. Арифметическое выражение

В Visual Basic 2005 выделяют три вида выражений: арифметические, логические и строковые.

Арифметическое выражение – это последовательность чисел, констант, переменных, функций и других арифметических выражений, заключенных в круглые скобки, которые соединены между собой знаками арифметических операций. Переменные, входящие в состав арифметического выражения должны иметь числовой тип. Функции, входящие в состав арифметического выражения должны возвращать числовое значение.

При вычислении значения арифметического выражения сначала выполняются действия в круглых скобках, затем все остальные арифметические операции в

Оглавление

соответствии со своими приоритетами. При равенстве приоритетов операции выполняются слева направо.

Составим арифметическое выражение для вычисления значения следующего выражения.

$$y = \frac{3\sin^2 5x + 5\cos 2x}{\ln\left|x^3\right| - \sqrt{e^5}}$$

Обратите внимание, что аргументы всех функций берутся в круглые скобки. Так же скобками оформляются числитель и знаменатель дроби. Символ подчеркивания означает перенос выражения на следующую строку.

```
y = (3 * Math.Sin(5 * x) ^ 2 + 5 * Math.Cos(2 * x)) _
/(Math.Log(Math.Abs( x ^ 3 ))- _
Math.Sqrt(Math.Exp(5)))
```

10. Окно ввода (InputBox)

Для ввода значений переменных традиционно используются текстовые поля. При этом для каждой переменной отводят свое отдельное текстовое поле. Но в ряде случаев такой подход оказывается достаточно неэффективным. Например, когда требуется задать большое количество различных значений. В таких случаях можно использовать специальное окно ввода (рис. 2). Оно вызывается с помощью функции `InputBox`. Рассмотрим ее параметры.

```
InputBox( Приглашение, Заголовок, Начальное значение)
```

`Приглашение` представляет собой строку текста, которая отображается в окне ввода. Для пользователя она является подсказкой, указывающей, какую информацию он должен ввести в поле ввода (рис. 2).

`Заголовок` – это строка текста, определяющая надпись, которая будет выводиться в заголовке окна (рис. 2).

Начальное значение – это значение, которое отображается в поле ввода значения (рис. 2). Если начальное значение является числом, то при передаче его в функцию `InputBox` необходимо использовать преобразование `Str`.

Оглавление

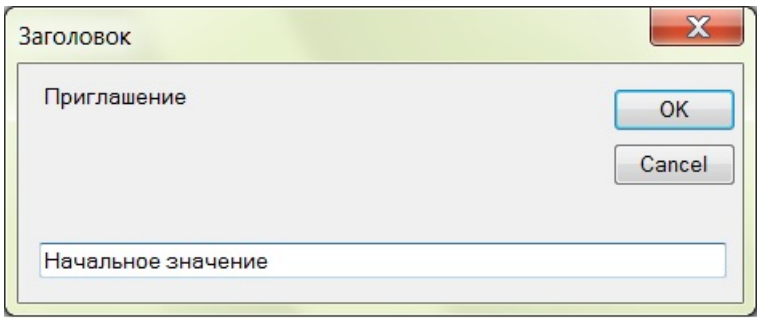

Рис. 2. Окно ввода `InputBox`

Все аргументы функции `InputBox` являются строками. Их значения должны задаваться в кавычках. Параметр `Приглашение` является обязательным. Он всегда присутствует при вызове функции `InputBox`. Два других параметра можно пропустить. Запятые при этом не ставятся. Если надо пропустить только параметр `Заголовок`, а параметр `Начальное значение` необходимо оставить, тогда между первым и третьим параметрами ставятся две запятые.

Результат функции `InputBox` всегда имеет тип `String`. Если функция `InputBox` используется для ввода числовых значений, то ее результат необходимо преобразовать с помощью функции `Val`.

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

При вызове функции `InputBox` на экране поверх формы появляется окно ввода (рис. 2). Работа программы приостанавливается до тех пор, пока окно не будет закрыто. После ввода информации пользователь должен нажать кнопку `OK` или клавишу `Enter`. Тогда окно ввода исчезнет, а переменной будет присвоено введенное значение.

Пример использования окна ввода рассмотрен в разделе 13.

11. Окно вывода сообщения (MsgBox)

Как правило, в приложениях окна вывода сообщений используются для того, чтобы дать пользователю какие-либо указания или задать вопрос, на который возможен один из стандартных ответов. Для вывода таких окон предназначена функция MsgBox. Рассмотрим ее параметры.

```
MsgBox(Текст сообщения, Параметры окна, Заголовок)
```

`Текст сообщения` – надпись, которую увидит пользователь в окне сообщения. Максимальная длина текста – 1024 символа. Текст сообщения всегда берется в кавычки.

Оглавление

`Заголовок` – это строка текста, определяющая надпись, которая будет выводиться в заголовочной части окна.

`Параметры окна` определяют внешний вид окна: тип пиктограммы и набор кнопок. Этот параметр является целым числом, которое, как правило, записывается в виде суммы двух целых параметров. Первый из них определяет вид пиктограммы (тип сообщения). Возможные значения этого параметра приведены в таблице 3. Второе слагаемое определяет набор кнопок, которые будут выведены в нижней части окна. Возможные значения этого параметра приведены в таблице 4.

Таблица 3

Число	Тип сообщения	Пиктограмма	Константа Visual Basic 2005
0	Без пиктограммы	Нет	Нет
16	Критическое сообщение		`MsgBoxStyle.Critical`
32	Вопрос		`MsgBoxStyle.Question`
48	Предупреждение		`MsgBoxStyle.Exclamation`
64	Информация		`MsgBoxStyle.Information`

Таблица 4

Число	Набор кнопок	Константа Visual Basic 2005
0	OK	`MsgBoxStyle.OkOnly`
1	OK, Отмена	`MsgBoxStyle.OkCancel`
2	Прервать, Повтор, Отмена	`MsgBoxStyle.AbortRetryIgnore`
3	Да, Нет, Отмена	`MsgBoxStyle.YesNoCancel`
4	Да, Нет	`MsgBoxStyle.YesNo`
5	Повтор, Отмена	`MsgBoxStyle.RetryCancel`

При вызове функции MsgBox на экране поверх формы появляется дополнительное окно с текстом сообщения. Например, следующая строка программного кода приведет к появлению окна, представленного на рис. 3.

```
MsgBox("Деление на ноль", 16, "Ошибка")
```

Оглавление

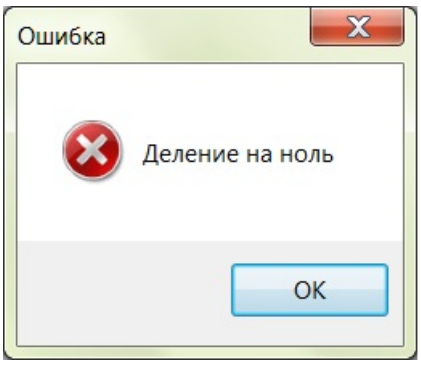

Рис. 3. Окно вывода сообщения

После вывода окна сообщения работа программы приостанавливается до тех пор, пока пользователь не закроет окно. Функция возвращает код кнопки, которую нажал пользователь. Это значение можно использовать в программе для реализации соответствующих действий. Для этого необходимо результат функции записать в некоторую целую переменную.

```
Dim Otvet As Byte
Otvet = MsgBox ("Повторить вычисления?", 32 + 4, _
"Вопрос")
```

Все возможные варианты результата работы функции `MsgBox` приведены в таблице 5.

Таблица 5

Значение	Кнопка	Константа Visual Basic 2005
1	OK	`MsgBoxResult.Ok`
2	Отмена	`MsgBoxResult.Cancel`
3	Прервать	`MsgBoxResult.Abort`
4	Повторить	`MsgBoxResult.Retry`
5	Пропустить	`MsgBoxResult.Ignore`
6	Да	`MsgBoxResult.Yes`
7	Нет	`MsgBoxResult.No`

12. Пример. Вычисление площади треугольника

В качестве примера программы линейной структуры рассмотрим задачу вычисления площади треугольника. Треугольник задан длинами сторон: *a*, *b*, *c*. Предполагается, что длины отрезков заданы таким образом, что из них можно построить треугольник. Для вычисления площади треугольника будем использовать формулу Герона

$$s = \sqrt{p(p-a)(p-b)(p-c)},$$

где p – это полупериметр треугольника.

Оглавление

$$p = \frac{a+b+c}{2}$$

Первым шагом является разработка экранной формы приложения. На форму поместим четыре метки и для каждой определим значение свойства `Text`. Первая метка – Сторона A, вторая – Сторона B, третья – Сторона C, четвертая – Площадь. Затем на форму помещаем четыре текстовых поля. Каждому из них даем имя (определяем значение свойства `Name`). Первое поле имеет имя txtA, второе – txtB, третье – txtC, четвертое – txtS. Теперь на форму надо поместить кнопку. Свойство `Text` этой кнопки имеет значение «Вычислить», а свойство `Name` – btStart. Разработанная форма приведена на рис. 4.

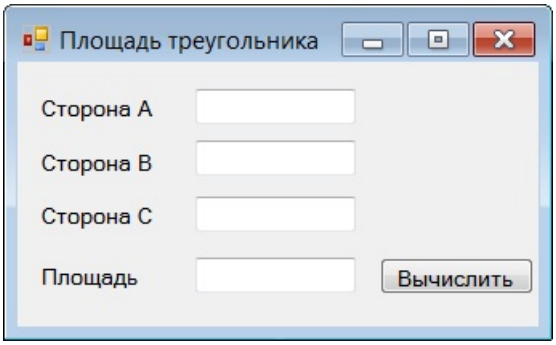

Рис. 4. Экранная форма для задачи вычисления площади треугольника

Для того чтобы связать программный код с кнопкой, дважды щелкнем левой кнопкой мыши по кнопке. В открывшемся окне редактора программного кода набираем текст программы.

Сначала объявим все необходимые переменные. Для решения задачи нам необходимо знать длины сторон треугольника. Это будут переменные `a`, `b`, `c`. Еще потребуется переменная для хранения полупериметра треугольника. Назовем ее `p`. Результатом работы программы будет площадь треугольника. Соответствующую переменную назовем `s`. Все переменные будут иметь тип `Single`, поэтому для их описания можно использовать один оператор `Dim`.

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

Следующий этап – ввод исходных данных. Для нашей задачи исходными данными являются длины сторон треугольника. Это значения переменных `a`, `b`, `c`. Так как все эти переменные имеют числовой тип, то при вводе их значений из текстовых полей необходимо использовать преобразование `Val`.

Оглавление

```
a = Val(txtA.Text)
b = Val(txtB.Text)
c = Val(txtC.Text)
```

Теперь можно вычислить полупериметр. Для этого складываем значения переменных `a`, `b`, `c` и полученную сумму делим на 2. Значение арифметического выражения запишем в переменную `p`.

```
p = (a + b + c) / 2
```

Затем вычисляем площадь треугольника по формуле Герона. Результат запишем в переменную `s`.

```
s = Math.Sqrt(p * (p - a) * (p - b) * (p - c))
```

Заключительное действие в нашей программе – вывод полученного результата. Так как площадь треугольника является числом, то при его выводе необходимо использовать преобразование `Str`.

```
txtS.Text = Str(s)
```

Теперь наша программа закончена. Сохраняем ее с помощью команды **Save All** из меню **File** и запускаем наше приложение. Для этого надо нажать клавишу **F5** или выбрать команду **Start Debugging** из пункта меню **Debug**. На экране появится окно приложения. Введем в три первых текстовых поля необходимые значения и нажмем кнопку «Вычислить». В четвертом текстовом поле появится вычисленный результат. Пример работы программы приведен на рис. 5. Полный текст этой программы представлен в приложении 2.

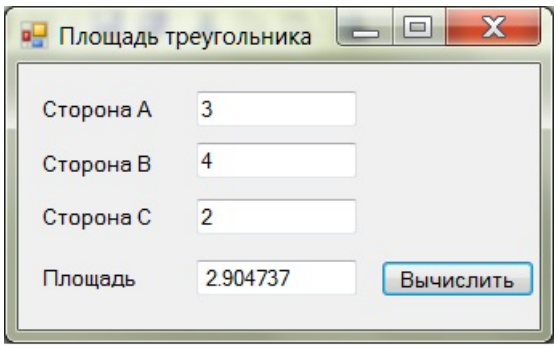

Рис. 5. Пример работы программы вычисления площади треугольника

13. Пример. Выделение цифр заданного числа

Рассмотрим еще одну программу линейной структуры. Ее целью является выделение всех цифр четырехзначного числа с двумя цифрами в дробной части.

Начнем с разработки интерфейса приложения. Для ввода исходного числа будем использовать окно ввода `InputBox`. Все значения, вычисленные в процессе выполнения программы, будут отображаться в окне списка. Поэтому на форму необходимо поместить окно списка. Дадим ему имя `lstResult`. Для запуска программы нам потребуется кнопка. Назовем ее btStart и зададим свойству Text значение «Старт». Разработанная форма приведена на рис. 6.

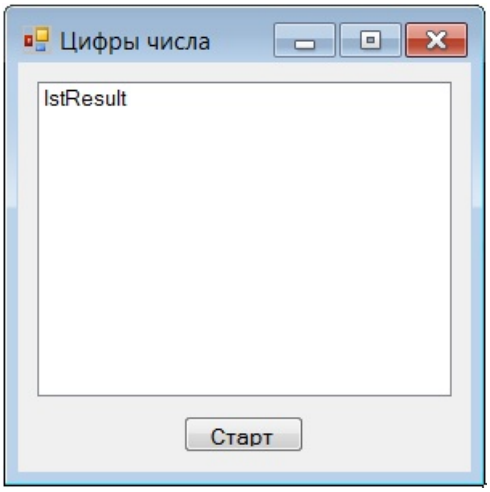

Рис. 6. Внешний вид приложения для нахождения цифр числа

Переходим к разработке программного кода. Первым шагом будет описание переменных, необходимых для решения задачи. Исходное число имеет дробную часть, поэтому для его хранения будем использовать тип `Single`. Переменную назовем `chislo`.

```
Dim chislo As Single
```

Каждая цифра исходного числа должна храниться в отдельной переменной. Так как любая цифра попадает в диапазон от 0 до 9, то для ее хранения вполне достаточно типа `Byte`. Нам потребуется четыре переменных для хранения цифр целой части (назовем их `c1, c2, c3, c4`) и две переменных для хранения цифр дробной части (их имена – `d1, d2`).

```
Dim c1, c2, c3, c4, d1, d2 As Byte
```

Для удобства вычисления выделим переменные для хранения целой и дробной части числа. Так как целая часть числа находится в диапазоне от 1000 до 9999, то для этих переменных будем использовать тип `Integer`. Имена этих переменных соответственно `celoe` и `drobnoe`.

```
Dim celoe, drobnoe As Integer
```

Оглавление

Работа программы начинается с очистки окна списка. Это делается для того, чтобы результаты предыдущего запуска программы не мешали пользователю.

```
lstResult.Items.Clear()
```

Следующий шаг – ввод исходного числа. Для этого будем использовать функцию `InputBox`. Так как исходная информация является числом, то при ее вводе необходимо использовать преобразование `Val`.

```
chislo = Val(InputBox("Введите четырехзначное число с
двумя цифрами после запятой"))
```

Для контроля правильности работы программы выведем исходное значение в окно списка. Единственный аргумент метода `Items.Add` запишем в виде суммы. Первое слагаемое – поясняющий текст, второе – исходное число. Как обычно, при выводе числовой информации будем использовать преобразование `Str`. Знак плюс в данном случае означает не математическое сложение, а соединение двух строк в одну.

```
lstResult.Items.Add("Исходное число:" + Str(chislo))
```

Выделим целую и дробную части числа. Целая часть выделяется с помощью функции `Math.Truncate`.

```
celoe = Math.Truncate(chislo)
```

Для получения дробной части надо из исходного числа вычесть целую часть. Так как дробная часть содержит две цифры, полученную разность надо умножить на 100. Результат вычисления округлим, чтобы целая и дробная части обрабатывались одинаково.

```
drobnoe = Math.Round((chislo - celoe) * 100)
```

Выведем целую часть числа в окно списка.

```
lstResult.Items.Add("Целая часть числа:" + _ Str(celoe))
```

Начинаем выделять цифры целой части. Сначала выведем в окно списка поясняющий текст.

```
lstResult.Items.Add("Цифры целой части")
```

Проще всего выделить последнюю цифру числа. Для этого достаточно взять остаток от деления этого числа на 10.

```
c4 = celoe Mod 10
```

Чтобы выделить остальные цифры, найдем целую часть от деления исходного числа на 10. Результат запишем в ту же переменную. При реализации этой операции будем использовать сокращенную запись арифметической операции.

```
celoe \= 10
```

Оглавление

В результате выполнения этой операции в переменной `celoe` вместо четырехзначного числа окажется трехзначное. Аналогично выделим все остальные цифры числа.

```
c3 = celoe Mod 10
celoe \= 10
c2 = celoe Mod 10
c1 = celoe \ 10
```

Полученные значения выведем в окно списка, используя преобразование `Str`.

```
lstResult.Items.Add(Str(c1))
lstResult.Items.Add(Str(c2))
lstResult.Items.Add(Str(c3))
lstResult.Items.Add(Str(c4))
```

Аналогичным образом выделим цифры дробной части. Сначала выведем дробную часть с поясняющим текстом.

```
lstResult.Items.Add("Дробная часть числа:" + _
Str(drobnoe))
lstResult.Items.Add("Цифры дробной части")
```

Потом выделим цифры дробной части.

```
d2 = drobnoe Mod 10
d1 = drobnoe \ 10
```

И выведем полученные результаты в окно списка, используя преобразование `Str`.

```
lstResult.Items.Add(Str(d1))
lstResult.Items.Add(Str(d2))
```

Наша программа закончена. Теперь ее необходимо сохранить и запустить. Пример работы программы представлен на рис. 7. Полный текст программы приведен в приложении 3.

Оглавление

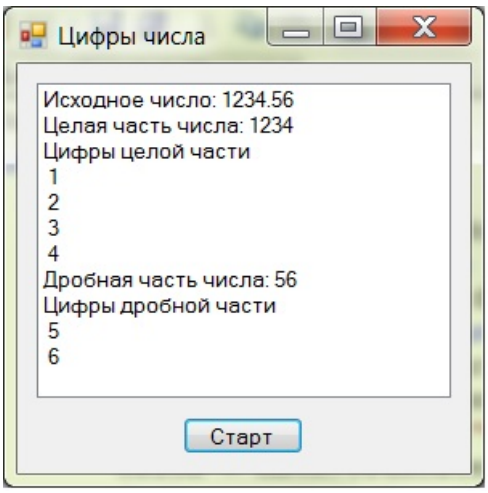

Рис. 7. Пример работы программы определения цифр числа

Приложение 1

Поменять местами значения двух переменных с использованием третьей.

```
Dim a, b, c As Integer
a = Val(txtA.Text)
b = Val(txtB.Text)
c = a
a = b
b = c
txtA.Text = Str(a)
txtB.Text = Str(b)
```

Приложение 2

Треугольник задан длинами сторон. Вычислить площадь треугольника.

```
Dim a, b, c, p, s As Single
a = Val(txtA.Text)
b = Val(txtB.Text)
c = Val(txtC.Text)
p = (a + b + c) / 2
s = Math.Sqrt(p * (p - a) * (p - b) * (p - c))
txtS.Text = Str(s)
```

Приложение 3

Дано четырехзначное число с двумя цифрами в дробной части. В окно списка вывести цифры этого числа.

```
Dim chislo As Single
```

Оглавление

```
Dim c1, c2, c3, c4, d1, d2 As Byte
Dim celoe, drobnoe As Integer
lstResult.Items.Clear()
chislo = Val(InputBox("Введите четырехзначное число с двумя
цифрами после запятой"))
lstResult.Items.Add("Исходное число:" + Str(chislo))
celoe = Math.Truncate(chislo)
drobnoe = Math.Round((chislo - celoe) * 100)
lstResult.Items.Add("Целая часть числа:" + Str(celoe))
lstResult.Items.Add("Цифры целой части")
c4 = celoe Mod 10
celoe \= 10
c3 = celoe Mod 10
celoe \= 10
c2 = celoe Mod 10
c1 = celoe \ 10
lstResult.Items.Add(Str(c1))
lstResult.Items.Add(Str(c2))
lstResult.Items.Add(Str(c3))
lstResult.Items.Add(Str(c4))
lstResult.Items.Add("Дробная часть числа:" + Str(drobnoe))
lstResult.Items.Add("Цифры дробной части")
d2 = drobnoe Mod 10
d1 = drobnoe \ 10
lstResult.Items.Add(Str(d1))
lstResult.Items.Add(Str(d2))
```

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