



МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Прикладная математика»

Программирование в Delphi: введение в Object Pascal

Методические указания к лабораторной работе № 1
по курсам «Информатика», «Алгоритмические языки
и программирование»

Автор
Е.Н. Ладоса, Д.С. Цымбалов, О.В. Яценко,
Е.О. Власова

Ростов-на-Дону, 2018



Аннотация

Излагаются теоретические и технологические сведения, необходимые для создания консольных приложений в Delphi. Целью работы ставится выработка навыков программирования при помощи современных визуальных средств. Предназначены для студентов всех специальностей факультета «Информатика и вычислительная техника».

Автор

Доцент, к.т.н.
Ладоша Е.Н.

Старший преподаватель кафедры
«Электроника и электротехника»
Цымбалов Д.С.

Доцент, к.ф.-м.н.
Яценко О.В.

Студент ДГТУ
Власова Е.О.



Цель работы

Цель лабораторной работы – изучить элементы программирования языка Delphi: 1) переменные, константы и типы данных, 2) арифметические операторы и их приоритет, 3) встроенные математические функции и функции приведения типов. Самостоятельное выполнение предлагаемых задач и разбор контрольных вопросов призваны закрепить усвоенный материал, приобрести навыки программирования.

Введение

Язык программирования среды Delphi базируется на языке Object Pascal. Его основу составляет алфавит – набор латинских букв, арабских цифр, специальных символов и пробелов. Из символов составляются лексемы – смысловые единицы: **константы; имена (идентификаторы); ключевые слова; знаки операций; разделители.**

Составление программы заключается в наделении лексем определенным алгоритмическим (информационным) содержанием: для этих целей используется данные двух типов – **константы и переменные.**

Структура программы

Изучать язык Delphi целесообразно в рамках **Console Application** (Консольного приложения). **Консольное приложение** – разновидность Windows-приложения, которое, предоставляя полный доступ к функциям WinAPI, не имеет **графического интерфейса пользователя (Graphical User Interface – GUI)**. Консольная программа в Delphi состоит из **заголовка, разделов описаний и раздела операторов.**

```
[program имя; ]           {заголовок}
{$APPTYPE CONSOLE}      {директива компилятору, указывающая на
то, что компилятор должен создавать консольное приложение}
[разделы описаний]
begin
  [раздел операторов]
end.                      (* программа заканчивается точкой *)
```

квадратными скобками здесь и далее помечены
необязательные части

Программа может содержать комментарии, заключенные в фигурные скобки { } или в скобки вида (* *). Комментарием является также текст после ключевой последовательности end, завершающей текст программы.

В *разделе операторов* записываются исполняемые операторы программы. Ключ-

чевые слова `begin` и `end` служат для объединения операторов в **составной оператор** или т.н. **блок**.

Разделы описаний бывают нескольких видов: описание **модулей, констант, переменных, меток, типов, процедур и функций**.

Модуль – это подключаемая к программе библиотека ресурсов (подпрограмм, констант и т.п.).

Раздел описания модулей помещается первым и начинается с ключевого слова `uses`, за которым через запятую перечисляются все подключаемые к программе модули – как стандартные, так и оригинальные, например:

```
uses SysUtils, MyModule;
```

Число и порядок следования прочих описательных разделов произвольны. Концом каждого описательного раздела служит начало следующего. Допускается несколько однотипных разделов описаний, но рекомендуется сводить все однотипные описания в один раздел.

В разделе описания переменных необходимо определить все переменные, которые будут использоваться в основной программе.

Раздел описания констант служит для использования в программе имен констант вместо значений.

Раздел описания меток начинается с ключевого слова `label`, за которым через запятую следует перечисление всех меток, встречающихся в программе.

Метка – это либо имя, либо положительное число, не превышающее 9999. Метка ставится перед любым исполняемым оператором и отделяется от него двоеточием. Пример описания меток:

```
label 1, 2, error;
```

Метки служат для выполнения конкретного оператора, указанного в качестве аргумента оператора безусловного перехода `goto`.

Директивы компилятора

Строка, начинающаяся символами `{ $`, является не комментарием, а директивой компилятора – специальной командой, от которой зависит процесс компиляции и выполнения программы. Например, строка `{ $I-, Q+ }` отключает контроль правильности ввода-вывода, но включает контроль переполнения при вычислениях.

Переменные и типы данных

Переменная – это величина, которая может принимать различные значения. С каждой переменной ассоциировано некоторое имя и ячейка оперативной

памяти с определенным адресом. **Диапазон значений** переменной ограничен и определяется ее **типом**. Переменные могут содержать как **числовые данные** (т.е. числа), так и могут содержать и **символьные данные** (буквы, цифры и др. символы).

Данные этих двух видов обрабатываются компьютером по-разному. Хранение символьных данных осуществляется относительно просто, для этого нужны только два типа данных: **символы** и **строки**. **Строка** – это последовательность символов, которая хранится в строковой переменной (т.е. в переменной типа строки), а **символ** – это неделимая часть строки. В то же время для числовых переменных используется большее количество типов данных, так как числа могут носить разный характер. Числа могут быть **целыми** или **вещественными**.

В табл. 1 и 2 перечислены типы данных Object Pascal, объем требуемой для их хранения памяти и диапазон допустимых значений.

Таким образом, для хранения одних и тех же чисел можно использовать различные типы данных. Ниже рассмотрим использование "стандартных" типов данных – Integer, Real, Boolean, Char и String.

Таблица 1

Логические и цифровые типы данных

Типы данных	Диапазон значений	Размер	Количество десятичных значащих цифр
Shortint	От -128 до 127	8 бит со знаком	
Smallint	От -32768 до 32767	16 бит со знаком	
Integer	От -2147483648 до 2147483647	32 бит со знаком	
Int64	От -2^{63} до $2^{63}-1$	64 бит со знаком	
Byte	От 0 до 255	8 бит без знака	
Word	От 0 до 65535	16 бит без знака	
Longword	От 0 до 4294967295	32 бит без знака	
Boolean	true или false	1 байт	
WordBool	true или false	2 байта	
LongBool	true или false	4 байта	
Real48	От $2,9 \cdot 10^{-39}$ до $1,7 \cdot 10^{38}$	6 байт	От 11 до 12
Single	От $1,5 \cdot 10^{-45}$ до $3,4 \cdot 10^{38}$	4 байта	От 7 до 8
Real	От $5,0 \cdot 10^{-324}$ до $1,7 \cdot 10^{308}$	8 байт	От 15 до 16
Extended	От $3,6 \cdot 10^{-4951}$ до $1,1 \cdot 10^{4932}$	10 байт	От 19 до 20
Comp	От $-2^{63}+1$ до $2^{63}-1$	8 байт	От 19 до 20
Currency	От -922337203685477.5808 до 922337203685477.5807	8 байт	От 19 до 20

Таблица 2

Логические и цифровые типы данных

Типы данных	Максимальная длина	Необходимый объем памяти
Char	Один символ ANSI	1 байт
WideChar	Один символ Unicode	2 байта
ShortString	255 символов ANSI	От 2 до 256 байт
String	231 символов ANSI	От 4 байт до 2 Гбайт
WideString	230 символов Unicode	От 4 байт до 2 Гбайт

Порядковый тип данных

Среди типов данных особо выделяются **порядковые** типы. Такое название можно обосновать двояко:

- Каждому элементу **порядкового** типа может быть сопоставлен уникальный (порядковый) номер. Нумерация значений начинается с нуля. Исключение - типы данных `shortint`, `integer` и `longint`. Их нумерация совпадает со значениями элементов.
- Кроме того, на элементах любого **порядкового** типа определен порядок (в математическом смысле этого слова), который напрямую зависит от нумерации. Таким образом, для любых двух элементов **порядкового** типа можно точно сказать, который из них меньше, а который – больше.

Для величин **порядковых** типов определены функции и процедуры, представленные в табл. 3:

Таблица 3

Стандартные подпрограммы, обрабатывающие порядковые типы данных

Функция/Процедура	Назначение
Ord (X: Char): Byte;	Функция возвращает порядковый номер значения переменной X
Pred (X: Integer): Integer;	Функция возвращает значение, предшествующее X
Succ (X: Integer): Integer;	Функция возвращает значение, следующее за X
Inc (var X: Integer);	Процедура возвращает значение, следующее за X
Inc (var X: Integer; N: Integer);	Процедура возвращает N-е значение, следующее за X
Dec (var X: Integer);	Процедура возвращает значение, предшествующее X
Dec (var X: Integer; N: Integer);	Процедура возвращает N-е значение, предшествующее X
Low (var X: Type): Integer;	Функция возвращает наименьшее значения величины типа X
High (var X: Type): Integer;	Функция возвращает наибольшее значения величины типа X

На первый взгляд кажется, будто результат применения процедуры **inc**(x) полностью совпадает с результатом применения функции **succ**(x).



Однако разница между ними проявляется на границах допустимого диапазона. Функция **succ**(*x*) неприменима к максимальному элементу типа, а вот процедура **inc**(*x*) не выдаст никакой ошибки, но, действуя по правилам машинного сложения, прибавит очередную единицу к номеру элемента. Номер, конечно, выйдет за пределы диапазона и за счет усечения превратится в номер минимального значения диапазона. Т.е., процедуры **inc**() и **dec**() воспринимают любой порядковый "замкнутым в кольцо": за последним значением следует первое.

Именованное, объявление и использование переменных

В языке Object Pascal имена переменных должны начинаться с буквы, принадлежащей английскому алфавиту, или с символа подчеркивания `_`. За первым символом может следовать любое количество букв английского алфавита, цифр или символов подчеркивания, однако только первые 255 символов принимаются транслятором во внимание. Для компилятора Delphi имена переменных **не чувствительны к регистру**. Например, если объявить переменную `todayDate`, то в исходном коде ее можно использовать под именем `todayDATE`, компилятор воспримет их как одну и ту же переменную.

Для создания описательных имен символ подчеркивания почти не используется. Вместо этого слова, входящие в переменную, выделяются тем, что первая буква второго и последующих слов набирается в верхнем регистре, например `TodayDate`, `TotalSales` или `CostOfGoodsSold`.

- Чтобы назначить переменной какой-либо тип данных, ее нужно **объявить** с этим типом.

Ключевым словом называется зарезервированное слово, распознаваемое компилятором как часть языка. Использование ключевых слов в качестве имен недопустимо. Ниже приведен список наиболее часто встречающихся ключевых (зарезервированных) слов:

<code>and</code>	<code>goto</code>	<code>set</code>
<code>array</code>	<code>implementation</code>	<code>shl</code>
<code>begin</code>	<code>in</code>	<code>shr</code>
<code>case</code>	<code>interface</code>	<code>string</code>
<code>const</code>	<code>label</code>	<code>then</code>
<code>div</code>	<code>mod</code>	<code>text</code>
<code>do</code>	<code>nil</code>	<code>to</code>
<code>downto</code>	<code>not</code>	<code>type</code>
<code>else</code>	<code>of</code>	<code>unit</code>
<code>end</code>	<code>or</code>	<code>until</code>
<code>file</code>	<code>pointer</code>	<code>uses</code>
<code>far</code>	<code>procedure</code>	<code>var</code>
<code>for</code>	<code>program</code>	<code>while</code>
<code>forward</code>	<code>record</code>	<code>with</code>



function repeat xor

Для объявления переменных и назначения им типов в Object Pascal используется ключевое слово `var`. Объявление сообщает компилятору имя и тип переменной, а также объем памяти, который нужно выделить для хранения объявленной переменной. Таким образом, `var` является **невыполняемым оператором** вида

```
var имя_переменной: тип_данных;
```

Например, следующая строка кода объявляет переменную `myNumber` как целую:

```
var myNumber: Integer;
```

Имя переменной и ее тип разделены двоеточием (:). Такое написание обусловлено **синтаксисом** оператора `var`. Точка с запятой в Object Pascal является **разделителем операторов**.

В одном операторе `var` можно объявить несколько переменных одного типа, например оператор

```
var number1, number2: Integer;
```

объявляет две переменные типа `Integer` с именами `number1` и `number2`. Таким образом, несколько переменных одного и того же типа можно объявить в одном операторе, синтаксис которого имеет вид:

```
var имя_переменной, [имя_переменной1] : тип_данных;
```

Самый общий синтаксис оператора `var` имеет вид:

```
var
```

```
имя_переменной1, [имя_переменной2] : тип_данных1;
```

```
имя_переменной21, [имя_переменной22] : тип_данных2;
```

В следующем **фрагменте кода** (неполной части исходного кода) объявляются несколько переменных:

```
var
```

```
dollars:        Integer;
```

```
cents:         Integer;
```

```
cost:          Real;
```

```
myMessage:    String;
```

Переменные делятся на **локальные** и **глобальные**. Переменные, объявленные в процедурах и функциях, являются **локальными** и существуют только во время выполнения соответствующих процедур и функций. Переменные, объявленные в разделе `var` основной программы, являются **глобальными**. **Глобальной** переменной можно присваивать начальное значение при ее описании в разделе `var`:

```
var имя_переменной1 : имя_типа1 = начальное_значение;
```


Локальные переменные инициализировать таким способом нельзя.

Запомните: *перед первым использованием каждая переменная обязательно должна быть инициализирована.*

Для присвоения переменным значений в Object Pascal используется оператор присваивания (`:=`). В следующем фрагменте кода значение переменной `count` увеличивается на 2:

```
count := count + 2;
```

Общее правило гласит: *результат вычисления выражения, стоящего в правой части оператора `:=`, сохраняется в переменной, стоящей в левой части этого оператора.* Таким образом, чтобы присвоить значение 1 переменной `count`, необходимо написать `count:=1`, но ни в коем не `1:=count`, потому что числовой константе 1 ничего присвоить нельзя. Оператор присваивания `:=` можно представлять себе как стрелку, указывающую влево (\leftarrow) и означающую "присвоить этой переменной это значение".

Константы

Константа – хранящееся в компьютере число или строка, значение которого (которой) остается неизменным на протяжении всего времени выполнения программы.

Как и переменная, константа хранится в ячейке оперативной памяти с определенным адресом. Константы делятся на **именные** и **неименные**. **Именная** константа может быть определена с помощью любого математического или строкового выражения. Во время компиляции имя константы просто замещается в программе ее значением. Числа и строки, записанные в исходном коде, называются **неименными** константами. Тип **неименной** константы определяется по правилам:

- любая последовательность цифр (возможно, предваряемая знаком "-" или "+" или разбиваемая одной точкой) воспринимается компилятором как неименованная константа – число (целое или вещественное) (см. табл. 4);
- любая последовательность символов, заключенная в апострофы, воспринимается как неименованная константа – строка, если требуется представить сам апостроф, он дублируется (см. табл. 4);
- символьные неименованные константы записываются в одной из трех форм (см. табл. 4): 1) символ заключенный в апостроф, 2) десятичный код символа, предваряемый знаком #, 3) буква, предваряемая знаком ^ (управляющий символ).

Таблица 4

*Знак \$ перед числом указывает о шестнадцатеричном представлении

10

сания нетипизированных именованных констант:

```
const
  n = -10;
  m = Int64(10000000000);
  mmm = n*100;
  x = 2.5;
  c = string('z');
  s = 'string';
  b = true;
```

Чтобы по имени можно было отличить константу от переменной, имена констант следует составлять из прописных букв с символами подчеркивания, например `days_per_week`.

Изменить **именную** константу довольно легко, для этого достаточно изменить одну строку исходного кода, в котором определено ее значение. В то же время изменять **неименную** константу весьма неудобно: нужно найти и модифицировать каждую строку кода, в которой она используется. Поэтому для облегчения модификации программы рекомендуется использовать **именные** константы.

Как мы уже знаем, константы – это данные программы, которые не могут изменять свое значение во время выполнения программы. Но в этом правиле имеется исключение – **типизированные константы**. Их описание производится по следующему шаблону:

```
const имя_константы : тип_константы = начальное_значение;
```

В следующем фрагменте кода объявляется несколько **типизированных констант**:

```
const
  n: integer = -10;
  x: real = 2.5;
  c: char = 'z';
  b: boolean = true;
```

Значение типизированных констант можно изменять во время выполнения программы, но при условии, что исполнена директива компилятора `{ $J+ }`. Если была выполнена директива `{ $J- }`, то изменять значение **типизированных констант** нельзя, и они превращаются в обычные именованные константы.

Арифметические операторы

Как и в математике, арифметические **операторы** в компьютерных программах выполняют определенные математические операции над переменными. Например, операция суммирования обозначается знаком `+` как в математике, так и в исходном коде программы. Однако умножение в математике и в компьютерных программах обозначается по-разному: в математике знаком `×`, а в програм-

мах — знаком $*$. По-разному обозначается и операция деления: в математике символом $:$ или \div , а в программах — символом $/$. В этих **бинарных операторах** используются два **операнда**. Операндом называется **переменная** или **выражение**, значение которого оператор использует для вычисления результата операции. Кроме **бинарных** существуют **унарные** операторы. В них используется не два, а один операнд. В табл. 5 перечислены арифметические операторы Object Pascal, типы операндов и результатов операций. В столбце с примерами в качестве операндов используются переменные x и y .

Таблица 5

Арифметические операторы Object Pascal

Операция	Оператор	Типы операндов	Тип результата	Пример
Унарный плюс	$+$ (унарный)	Целый, вещественный	Целый, вещественный	$+x$
Унарный минус	$-$ (унарный)	Целый, вещественный	Целый, вещественный	$-x$
Умножение	$*$	Целый, вещественный	Целый, вещественный	$x*y$
Деление	$/$	Целый, вещественный	Вещественный	x/y
Целочисленное деление	<code>div</code>	Целый	Целый	$x \text{ div } y$
Деление по модулю два	<code>mod</code>	Целый	Целый	$x \text{ mod } y$
Суммирование	$+$	Целый, вещественный		$x+y$
Вычитание	$-$	Целый, вещественный		$x-y$

В Object Pascal результат операции деления ($/$) всегда имеет тип `Extended`, независимо от типов операндов. Если переменные x и y имеют тип `Integer`, то результат операции x/y имеет тип `Extended`. Для прочих арифметических операций справедливы следующее правила:

- если хоть один операнд имеет вещественный тип, то результат операции имеет тип `Extended`;
- результат операции над целыми операндами имеет тип `Int64`, если хоть один операнд имеет тип `Int64`, в противном случае результат имеет тип `Integer`.

Целочисленное деление (оператор `div`) используется для деления двух целых чисел, причем возвращаемый результат содержит только целую часть от отношения, дробная часть отбрасывается. Например, оператор `5 div 2` возвращает результат 2. Оператор деления по модулю `mod` возвращает целый остаток

деления двух целых чисел. Оператор $5 \bmod 2$ возвращает результат 1. Математически оператор $x \bmod y$ эквивалентен операции $x - (x \operatorname{div} y) * y$ как для положительных, так и для отрицательных операндов.

Выражения и порядок вычислений

Выражения – это правило вычисления значений. В выражении участвуют операнды, объединенные знаками операций. Операндами выражения могут быть константы, переменные и вызовы функций. Операции выполняются в определенном порядке в соответствии с приоритетом. Для изменения порядка вычислений операций используются круглые скобки, уровень вложенности которых практически не ограничен. Если же скобок нет, то сначала исполняются операции с более высоким приоритетом, затем – с менее. Последовательность равноприоритетных операций выполняется "слева направо".

Таблица 6

Приоритеты операций языка Object Pascal

Операция	Оператор	Приоритет
Унарные операции	$+$, $-$, not , $@$, $^$, $\#$	Первый (высший)
Операции, эквивалентные умножению	$*$, $/$, div , mod , and , shl , shr , as	Второй
Операции, эквивалентные сложению	$+$, $-$, or , xor	Третий
Операции сравнения	$=$, $<>$, $>$, $<$, $<=$, $>=$, in , is	Четвертый

Замечание: Вызов любой функции имеет более высокий приоритет, чем все внешние относительно этого вызова операции. Выражения, являющиеся аргументами вызываемой функции, вычисляются в момент вызова

Приведение типов и функции преобразования типов

Object Pascal является **сильно типизированным языком**. Это означает, что типы данных строго различаются, а при вычислении выражений строго выполняются определенные правила и ограничения на использование различных типов. Преимущество сильной типизации состоит в том, что транслятор получает возможность правильно обрабатывать данные и тщательно проверять исходный код, исключать **ошибки в процессе выполнения** программы. Чтобы понять принципы сильной типизации, выполните следующий фрагмент кода:

```
var
    myChar:   Char;
    myByte:   Byte;
begin
    myChar := 'A';
    myByte := myChar; {Строка с синтаксической ошибкой!}
end;
```

Переменные обоих типов (Char и Byte) для хранения своих значений занимают по одному байту памяти. Компилятор генерирует сообщение об ошибке

потому, что согласно правилам сильной типизации Object Pascal нельзя присваивать значение типа Char переменной типа Byte и наоборот.

Однако иногда в программе необходимо выполнить операцию, запрещенную правилами типизации. Обойти правила типизации можно средствами **приведения типов**. Приведением типов называется преобразование типов в отношении промежуточных результатов вычислений. Обратите внимание: преобразуются только типы промежуточных результатов, типы объявленных переменных всегда остаются неизменными. Общий синтаксис приведения типов имеет вид:

тип_данных (выражение);

Например, оператор `Integer('A')` приводит тип символа A к целому. Значение любой переменной можно привести к любому типу при условии, что исходный и результирующий типы занимают одинаковый объем памяти и не смешиваются целые и вещественные числа.

Теперь можно исправить наш предыдущий пример так:

```
var
  myChar: Char;
  myByte: Byte;
begin
  myChar := 'A';
  myByte := Byte(myChar); {Синтаксическая ошибка исправлена!}
end;
```

Операция приведения типов не преобразует типы значений между вещественными и целыми числами. Однако это можно сделать с помощью встроенных процедур и функций преобразования типов Object Pascal, перечисленных в таблице 7.

Таблица 7

Процедуры и функции преобразования типов

Функция/Процедура	Назначение
CompToCurrency (Value: Comp) : Currency;	Преобразует значение Comp в значение Currency
CompToDouble (Value: Comp) : Double;	Преобразует значение Comp в значение Double
CurrencyToComp (Value: Currency; var Result: Comp);	Преобразует значение Currency в значение Comp
DoubleToComp (Value: Double; var Result: Comp);	Преобразует значение Double в значение Comp
Int (X: Extended) : Extended;	Возвращает целую часть вещественного значения
Round (X: Extended) : Int64;	Округление значения x до ближайшего целого значения Int64. Если x находится точно между двумя целыми, то результат всегда является четным числом
Trunc (X: Extended) : Int64;	Усекает вещественное значение до целого (отбрасывает дробную часть)

Функция/Процедура	Назначение
Chr (X: Byte): Char;	Возвращает символ с порядковым значением X (ASCII)

Выражения, в которых операнды имеют разные числовые типы данных, называются **смешанными арифметическими выражениями**. Их использование в Object Pascal допускается. Например, следующий фрагмент кода не содержит ошибки:

```
var
  num1, result: Real;
  num2: Integer;
begin
  num1 := 7.2;
  num2 := 3;
  result := num1 * num2; {Смешанное арифметическое выражение}
end;
```

В предыдущем коде num1 является вещественным числом, а num2 – целым. Таким образом, num1 * num2 является смешанным арифметическим выражением. Из табл. 7 видно, что для явного преобразования целого типа в вещественный никакой встроенной процедуры нет. Неявное приведение типов данных можно отключить, если указать директиву компилятора {\$R+}, которая принуждает компилятор всегда проверять границы и диапазоны.

Встроенные математические функции

Встроенные математические функции перечислены в табл. 8.

Таблица 8

Встроенные математические функции

Функция	Название
Abs (x) ;	Абсолютное значение x
Exp (X: Real): Real;	Экспонента
Frac (X: Extended): Extended;	Дробная часть
ArcTan (X: Extended): Extended;	Арктангенс угла X*
Cos (X: Extended): Extended;	Косинус угла X*
Sin (X: Extended): Extended;	Синус угла X*
Ln (X: Real): Real;	Логарифм натуральный числа X
Sqr (X: Extended): Extended;	Квадрат числа X
Sqrt (X: Extended): Extended;	Корень квадратный числа X
Pi (): Extended;	3,141592653589792123852
Random (): Extended;	Генерирует случайное число в диапазоне от 0 до 1
Random (const ARange: Integer): Integer;	Генерирует случайное число в диапазоне от 0 до ARange
Randomize	Инициализирует генератор случайных чисел

*Значение угла X в радианах

Примеры записи математических выражений при помощи встроенных математических функций:

Математическая запись	Запись на Object Pascal
$Tg(x)$	$\text{Sin}(x) / \text{Cos}(x) ;$
$\text{ArcSin}(x)$	$\text{ArcTan}(x / \text{Sqrt}(1 - \text{Sqr}(x))) ;$
$\text{ArcCos}(x)$	$\text{Pi} / 2 - \text{ArcTan}(x / \text{Sqrt}(1 - \text{Sqr}(x))) ;$
$\text{ArcCtg}(x)$	$\text{Pi} / 2 - \text{ArcTan}(x)$
x^y	$\text{exp}(y * \ln(x))$
$\text{Log}_x(y)$	$\ln(y) / \ln(x)$
$x^2 - 7x + 6$	$\text{Sqr}(x) - 7 * x + 6$
$\frac{ x - y }{1 - xy }$	$(\text{Abs}(x) - \text{Abs}(y)) / (1 + \text{Abs}(x * y))$

Контрольные задания

- Ниже приведены два столбца чисел. Сопоставьте представления форматов Object Pascal и математического для этих чисел:

7,9985	1E6
π	0.3278282E5
-1/3	720
10^6	1*2*3*4*5*6
-1/10000000	74
32782,82	7.9985
$\sqrt{2}$	-0.3333
-0,3(3)	-1E-7
6!	0.1414E1
LXXIV	1E1
10	3.1416
74	1.4142

- Запишите на языке Object Pascal следующие формулы (используя только функции из табл. 8):

а) $\frac{-b + \sqrt{b^2 - 4ac}}{2a}$, б) $\{[(ax - b)x + c]x - d\}x - e$,

в) $\frac{1}{3} \left(\left(\frac{\sin^2(x) - \cos^2(x)}{\sin \frac{x+y}{2}} - e^{|\cos x| + \sin x} \right) \ln x - \sqrt{x-1} \right)$, г) $\left(1 + \frac{x}{2!} + \frac{y}{3!} + \frac{z}{4!} \right)$,

$$\text{д) } \frac{1 + \sin^2(x+y)}{2 + \left| \frac{x-2x}{(1+x^2y^2)} \right|} + x, \text{ е) } \frac{1}{a + \frac{1}{b+c+\frac{1}{d}}} - \frac{a}{2d + \frac{c}{2b + \frac{1}{f}}},$$

$$\text{ж) } \tan^2\left(\frac{x^2}{2} - 1\right) + \frac{2\cos(x - \pi/6)}{1/2 + \sin^2 y}, \text{ з) } 2\log_2 \frac{x}{15}, \text{ и) } \sqrt[8]{x^8 + 8^x} + \arcsin x.$$

3. Определите тип следующих числовых выражений:

- а) `1+sqrt(4)+sin(0)+0.0`; б) `sqr(2.0)+sqrt(121)-succ(-11)`;
 в) `pred(32700)+round(10/3)`; г) `trunc(sin(0))+1`;
 д) `succ(round(5/2)-pred(3))`.

Контрольные вопросы

1. Что такое переменная? Чем она отличается от константы?
2. Что есть и как используется оператор присваивания?
3. Почему так важно инициализировать переменные?
4. Какие типы данных предназначены для хранения чисел, а какие – для хранения символов?
5. Чем именованные константы отличаются от неименованных? Какие из них предпочтительнее использовать?
6. Что такое целочисленное деление и деление по модулю? Какие операторы выполняют эти операции?
7. Каким символом заканчивается текст программы?
8. Из каких двух основных частей состоит программа?
9. Какое служебное слово используется для описания раздела констант?
10. Каким служебным словом начинается описание переменных?
11. С какой целью в программах используют комментарии?
12. Где в программе можно написать комментарий?
13. Какими символами ограничивается текст комментария?
14. Как влияет комментарий на выполнение программы?
15. Можно ли аргумент стандартной тригонометрической функции задать в градусах?
16. Для какого типа данных определены операции `+`, `-`, `*`, `/`?
17. Можно ли при записи идентификаторов использовать буквы русского алфавита?
18. Допустимо ли присваивание переменной целого типа значения выражения вещественного типа?
19. Каков приоритет выполнения арифметических операций?



20. Что такое выражение, операция, операнд?

Список использованной литературы

1. *Фаронов В.В.* Delphi 3. Учебный курс. М.: «Нолидж», 1998. 400 с.
2. *Галисеев Г.В.* Программирование в среде Delphi 8 for .NET. М.: Издательский дом «Вильямс», 2004. 304 с.
3. *Павловска Т.А.* Паскаль. Программирование на языке высокого уровня. СПб.: Питер, 2003. 393 с.
4. *Абрамов С.А. и др.* Задачи по программированию. М.: Наука, 1988. 224 с.