Лекция 2

Любая программа (приложение) предназначена для обработки ***данных***.

Концепция данных

Обрабатываемые данные подразделяются на

- Переменные
- Константы.

Константы – это данные, значения которых установлены в разделе констант (const), и не изменяются во время выполнения программы.

Синтаксис объявления констант:

```
Const     имя1=значение;
          . . .
          имяn=значение;
```

Например

```
const e=2.71;
    n = 1000;
    slovo = 'константа';
```

Имя – последовательность из латинских букв, цифр. Первый символ – не цифра.

Переменные – это данные, которые объявляются в разделе переменных (var), получают значения во время выполнения приложения, и значения переменных могут быть изменены.

```
var   имя1: тип1;
      Имя2, имя3,  . . . , имяn : тип2;
        . . .
```

Любое данное характеризуется именем и типом возможных значений.

При этом тип константы распознается во время компиляции по указанному значению константы.

Тип переменной связывается с именем переменной во время компиляции. Значение переменной изменяется во время выполнения программы. Первое назначение значения переменной называют *инициализацией* переменной, и компилятор обычно отслеживает, чтобы инициализация состоялась. Значение переменной изменяется при помощи оператора присваивания.

Синтаксис оператора присваивания:

```
Имя := выражение;
```

Например

```
var   a, b : real;
      simb : char;
begin
    a := 1.5;  b:=3;   simb:='a';
```

Концепция типов данных

Тип данных определяет множество допустимых значений, а также множество операций, применимых к значениям данного типа. Причем результат операций должен быть того же типа.

Язык Pascal характеризуется разветвленной структурой типов данных В языке предусмотрен механизм создания новых типов на основании уже имеющихся.

Предопределенные типы данных

1. Целый тип (integer)

Множество значений целого типа состоит из целых чисел. Количество значений этого типа ограничено (в отличие от бесконечного множества целых чисел в математике) и зависит от возможностей компьютера.

Операции, допустимые над значениями целого типа:

Сложение (+), вычитание (–), умножение (*) – привычные операции с числами. Результат операции деления целого на целое может быть нецелым. Например, результатом деления целого 2 на целое 4 является вещественное 0,5. Поэтому деление значений целого типа представлено двумя операциями:

Операция a div b дает в результате частное, а операция a mod b – остаток деления a на b. Таким образом, если $c = a \; div \; b$, $d = a \bmod b$, то $a = b \times c + d$.

Константы целого типа записываются при помощи цифр и знаков +, –.

2. Вещественный тип (real)

Множество значений вещественного типа состоит из вещественных чисел.

Над значениями целого типа можно выполнять обычные арифметические операции: сложение (+), вычитание (–), умножение (*), деление (/). При записи вещественных констант в виде десятичной дроби принято в качестве разделителя ставится точку, а не запятую.

3. Логический тип (boolean)

Множество значений этого типа содержит 2 значения true (истина) и false (ложь).

Операции, определенные над значениями типа boolean, рассмотрим позже.

4. Символьный тип (char)

Значениями этого типа являются символы, заключенные в апострофы.

5. Тип строка (string)

Строка – это набор символов в апострофах, воспринимаемый как единое целое.

Строки можно складывать при помощи знака +.

Рассмотрим пример.

```
var  str1, str2, str3, str4 : string;
begin
  str1 := ‘Добрый’;  str2 := ‘день’;
  str3 := str1 + str2;     {‘Добрыйдень’}
  str4 := str1 +’  ‘ + str2;     {‘Добрый  день’}
```

Рассмотрим некоторые стандартные функции, которыми мы будем пользоваться.

1. StrToInt (S: string): integer; – преобразует строку S в целое число.
2. StrToFloat (S: string): real; – преобразует строку S в вещественное число.
3. IntToStr (value: integer): string; – преобразует целое выражение value в строку.
4. FloatToStr (value: real): string; – преобразует вещественное выражение value в строку.

Арифметические выражения

Запись, содержащая имена констант и переменных арифметического типа (это типы real и integer), вызовы стандартных функций, знаки арифметических операций и скобки, называется арифметическим выражением. Значением арифметического выражения является число.

В программе арифметическое выражение записывается в строку.

Например, $\frac{a}{b}$ будет записано в виде a / b.

Порядок выполнения операций, входящих в арифметическое выражение, подчиняется следующий (правило приоритетов):

1. Вычисление стандартных функций.
2. Мультипликативные операции (*, / , div, mod).
3. Аддитивные операции (+, –,).

Изменить порядок выполнения операций можно при помощи скобок.

Любой язык программирования имеет набор стандартных функций, которые существенно облегчают работу программиста.

Синтаксис вызова стандартной функции:

Имя_функции (аргумент)

В качестве аргумента может быть любое выражение допустимого типа.

Приведем список стандартных математических функций, имеющихся в языке Pascal (классическом).

Обращение	**Тип параметра**	**Тип результата**	**Примечание**
Abs(x)	Real, Integer	Тип аргумента	Модуль аргумента
Sqrt(x)	Real	Real	Корень квадратный из аргумента
Sqr(x)	Real	Real	Аргумент в квадрате
Ln(x)	Real	Real	Натуральный логарифм
Exp(x)	Real	Real	Экспонента (e^x)
Sin(x)	Real	Real	Синус, угол в радианах
Cos(x)	Real	Real	Косинус, угол в радианах
Arctan(x)	Real	Real	Арктангенс, значение в радианах

Рассмотрим пример. Построить арифметические выражения для вычисления

$$a = \frac{\sqrt{x+y} + tg5z}{\cos(x^3 + 5y) + \dfrac{2x}{x-y}}, b = \frac{e^{-xz^2} + \sqrt[3]{\left|z^2 - y\right|}}{2xz}.$$

Если пользоваться приведенными выше функциями, получим

```
a:=(sqrt(x+y)+sin(5*z)/cos(5*z))/(cos(x*x*x+5*y)+2*x/(x-y));
b:=(exp(-x*sqr(z))+exp(ln(abs(z*z-y))/3)/(2*x*z);
```

В Lazarus включен модуль Math, в котором набор стандартных математических функций существенно расширен. Некоторые функции приведены в следующей таблице:

Обращение	**Пояснение**
ArcCos(X)	Арккосинус
ArcSin(X)	Арксинус
Tan(X)	Тангенс
Cotan(X)	Котангенс
CosH(X)	Косинус гиперболический
SinH(X)	Синус гиперболический
TanH(X)	Тангенс гиперболический

Log10(X)	Десятичный логарифм
Log2(X)	Двоичный логарифм
LogN(Base,X)	Логарифм от X по основанию Base
Power(Base,Exponent)	Возведение Base в вещественную степень Exponent
IntPower(Base,Exponent)	Возведение Base в целочисленную степень Exponent

Заметим, что для применения любой из этих функций нужно в текст модуля **unit** в список модулей оператора **uses** добавить **Math** (дополнить список используемых модулей).

Используя эти функции, мы можем записать выражения иначе:

*a:=(sqrt(x+y)+tan(5*z))/(cos(x*x*x+5*y)+2*x/(x-y));*

*b:=(exp(-x*z*z)+Power(abs(z*z-y),1/3))/(2*x*z);*