Основы языка Pascal

Наумов Д.А., доц. каф. КТ

Программирование и алгоритмические языки, 2019

Содержание лекции

1. Основные элементы языка

2. Концепция типов данных
   - Организация данных в программах
   - Слабая и сильная типизация

Структура приложения

```
program ProgramName;

uses <Подключение модулей>;

const
  <Определение констант>;

type
  <Описание типов данных>;

var
  <Описание переменных>;

label
  <Описание меток>;

<Описания/определения процедур и функций>;

begin
  <Операторы>
end.
```

Основные элементы языка

Алфавит языка:

- латинские буквы A, B, C,. . . , x, y, z;
- цифры 0, 1, 2,. . . , 9;
- специальные символы +, -, /, =, <, >, [,], ., (,), ;, :, {, }, $, #, _, @, ‘, ^.

Комментарии:

```
{Комментарий может выглядеть так!}
(*Или так.*)
//А если вы используете такой способ,
//то каждая строка должна начинаться
//с двух символов «косая черта».
```

Основные элементы языка

Идентификатор

совокупность букв, цифр и символа подчёркивания.

- начинается с буквы или символа подчёркивания;
- используется для именования различных объектов (констант, переменных, меток, типов данных, процедур, функций, модулей, классов) языка;
- не может содержать пробел;
- прописные и строчные буквы в именах не различаются;
- каждое имя (идентификатор) должно быть уникальным и не совпадать с ключевыми словами.

Типы данных

Данные хранятся в памяти компьютера и могут быть самых различных типов

- целые;
- вещественные числа;
- символы;
- строки;
- массивы...

Тип данных определяет:

- структуру хранения данных в памяти;
- область значений, которые могут принимать данные этого типа;
- множество операций, применимых к данным этого типа.

Переменная

величина, которая может изменять свое значение.

- идентификатор (имя) служит для обращения к области памяти, в которой хранится значение;
- значение переменной можно изменить;
- перед использованием любая переменная должна быть описана.

Описание переменной:

```
1  var
2    идентификатор1, идентификатор2,...,идентификаторN: тип;
```

```
var
ha: integer;     //Объявлена целочисленная переменная.
hb, c: real;     //Объявлены две вещественные переменные.
```

Константа

величина, которая не может изменять свое значение.

Определение константы:

```
1  var
2    идентификатор = константное_выражение;
```

```
const
h=3;        //Целочисленная константа.
bk=-7.521;    //Вещественная константа.
c='abcde';    //Символьная константа.
```

Программирование языке низкого уровня требует точного знания:

- как данные представлены в виде последовательности битов;
- какие машинные команды должны применяться для реализации требуемых операций.

Язык высокого уровня обеспечивает следующие возможности:

- на объекты данных ссылаются с помощью определенных пользователем имен, а не конкретных адресов памяти;
- объекты данных связаны с типом, определяющим множество значений, которые могут приниматься объектами этого типа, и множество операций, которые могут применяться к объектам этого типа.

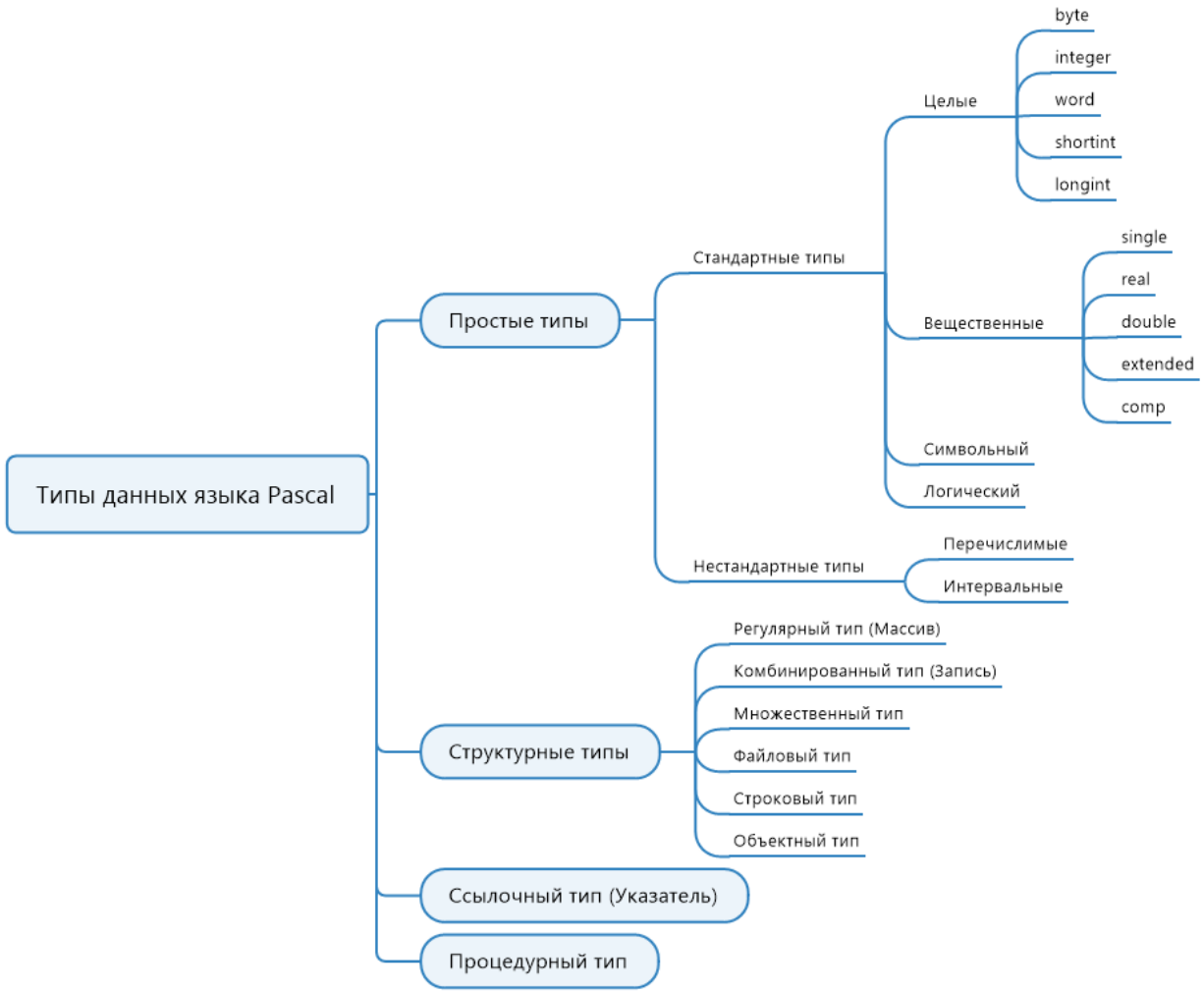
Хороший механизм типов является ключевым фактором при обеспечении надежности языка программирования, что имеет первостепенную важность при программировании.

В общем случае в языке программирования должно быть *множество предопределенных типов данных* и *набор механизмов для спецификации типов*, определяемых пользователем.

Типы данных

1. простые;
   - нет внутренней структуры;
   - могут содержать лишь одно значение;
   - доступные операции предопределены;
2. структурные
   - состоят из других простых и (или) структурных типов;
   - могут содержать составные значения;
   - могут инкапсулировать поведение.

Типы данных


Типы данных языка Pascal
Простые типы
Стандартные типы
Целые
byte
integer
word
shortint
longint
Вещественные
single
real
double
extended
comp
Символьный
Логический
Нестандартные типы
Перечислимые
Интервальные
Структурные типы
Регулярный тип (Массив)
Комбинированный тип (Запись)
Множественный тип
Файловый тип
Строковый тип
Объектный тип
Ссылочный тип (Указатель)
Процедурный тип

Целочисленные типы данных FreePascal:

Тип	Диапазон	Размер, байт
Byte	0...255	1
Word	0...65535	2
LongWord	0...4294967295	4
ShortInt	−128...127	1
Integer	−2147483648...2147483647	4
LongInt	−2147483648...2147483647	4
Smallint	−32768...32767	2
Int64	$-2^{63}...2^{63}$	8
Cardinal	0...4294967295	4

Описание целочисленных переменных:

```
var
b: byte; i, j: integer; W: word; L_1, L_2: longint;
```

Внутреннее представление вещественного числа:

То же самое, но в двоичном представлении:

-118.625 118d = 1110110b

$0.625d = 1 \cdot 2^{0} + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} = 0.101b$

-1110110.101b

$$-1.110110101 * 2^{+6}$$

Знак мантиссы — Мантисса — Порядок (экспонента)

Константы вещественного типа:

- в формате с фиксированной точкой;
- в формате с плавающей точкой.

Вещественные типы данных FreePascal:

Тип	Диапазон	Кол-во знач-х цифр	Размер, байт
`Single`	$1.5E45 \ldots 3.4E+38$	7–8	4
`Real`	$2.9E-39 \ldots 1.7E+38$	15–16	8
`Double`	$5.0E-324 \ldots 1.7E+308$	15–16	8
`Extended`	$3.4E-4932 \ldots 3.4E+4932$	19–20	10
`Comp`	$-2^{63} \ldots 2^{63}$	19–20	8
`Currency`	$-922337203685477.5808 \ldots 922337203685477.5807$	19–20	8

Логический тип данных

Константы логического типа:

- True - логическая истина;
- False - логическая ложь.

Логические типы данных FreePascal:

Тип	Размер, байт
Boolean	1
ByteBool	1
WordBool	2
LongBool	4

Перечисляемый тип данных относится к нестандартным порядковым типам и определяется набором идентификаторов, с которыми могут совпадать значения переменных этого типа.

```
1  type TDay = (MO, TU, WE, TH, FR, SA, SU);
2  var  D: TDay;
3  (* ORD (WE) = 2 *)
```

Максимальная мощность перечисляемого типа - 256 значений.

```
(* типы эквивалентны по внутреннему представлению...*)
1  type TColors = (black, red, white );
2       TOrdenal=(one, two, three);
3       TDays=(monday, thesday, wednesday);
4  var  col: TColor; num: TOrdenal; day: TDays;
(* допустимые операторы присваивания *)
5  col := black; num := two; day := monday;
(* недопустимые операторы присваивания *)
6  col := two; day := black;
```

Для перечисляемых типов определены стандартные функции PRED, SUCC и ORD, имеющие тот же смысл, что и для стандартных скалярных типов.

```
1  type TColors = (black, red, white);
2  (* SUCC(red) = white *)
3  (* PRED(red) = black *)
4  (* ORD(red) = 1 *)
```

- значения перечисляемого типа должны быть определены только идентификатором (именем);
- нельзя присваивать переменной значение из описания другого типа;
- недопустимо описание двух и более перечисляемых типов с совпадающими значениями;
- нельзя значения перечисляемого типа использовать для ввода и вывода.

```
1 type TColor1 = (red, yellow, blue);
2      TColor2 = (green, blue, gray);
```

Ограниченный тип данных относится к нестандартным порядковыми типам, образуется на основе порядковых типов, называемых базовыми, путём ограничения диапазона значений этих типов заданием минимального и максимального значений.

```
1  type TDay = (MO, TU, WE, TH, FR, SA, SU);
2       TNumber = 10..25;
3       TChars = 'c'..'x';
4       TWeekDays = SA..SU;
```

- базовым типом для создания ограниченного типа может быть любой порядковый тип;
- два символа ".."рассматриваются как один символ, поэтому между ними не допустимы пробелы;
- необходимо, чтобы левая граница диапазона не превышала его правую границу.

Особенности слабой типизации

- операция, которая может восприниматься машиной как корректная, может быть некорректной на абстрактном уровне программы;

```
1  var c: char;
2  c := 10;
```

- для сохранения корректности предусмотрено выполнение операции преобразования типа;

```
1  var x,y: real;
2  var i,j,k: integer;
3  i := x;
4  k := y - j;
```

- увеличение гибкости, обеспечиваемое слабой типизацией, является слишком дорогой ценой за резкое уменьшение ясности программ и необходимость дополнительного контроля во время работы компилятора.

Особенности сильной типизации

- каждый объект обладает уникальным типом;
- тип определяет множество значений и множество операций;
- тип присваиваемого значения и тип объекта данных, которому производится присваивание, должны быть эквивалентны;
- применяемая к объекту операция должна принадлежать множеству операций, определяемому типом объекта.

```
1  var x: real;
2      i: integer;
3      b: boolean;
4      c: char;
5  i := 'A' (*разные типы в левой и правой частях*)
6  x := i;
7  i := i or 10; (*недопустимая операция*)
```

Преимущество сильной типизации: программисту разрешается определять при описании типа свои собственные типы.

```
1  type
2    TAge = 0..200;
3    TIndex = 1..10;
4  var
5    age, total_age : TAge;
2    i: TIndex;
3  begin
4    total_age := 0;
4    for i := 1 to 10 do begin
5      readln(age);
6      total_age := total_age + age;
7    end;
8    writeln(total_age);
9  end.
```

Имеются две различные основы для вычисления эквивалентности типов данных:

- структурная эквивалентность: два объекта принадлежат эквивалентным типам, если у них одинаковая структура;
- именная эквивалентность: два объекта принадлежат эквивалентным типам, если они описаны с помощью одного и того же типа.

В языке Pascal принят принцип именной эквивалентности типов, устанавливающий, что два типа T1 и T2 эквивалентны, если выполняется одно из следующих условий:

- T1 и T2 — одно и то же имя типа;
- Тип T2 описан с использованием типа T1 равенством вида type T2=T1; или последовательностью подобного вида равенств.

```
1  type T1 = integer;
2     T3 = T1;
3     T2 = T3;
4     T4 = 1..10;
5     T5 = 1..10;
```

Выражение

задаёт порядок выполнения действий над данными и состоит из операндов (констант, переменных, обращений к функциям), круглых скобок и знаков операций.

Операнд

это аргумент операции.

Каждая операция:

- имеет знак операции;
- определяет количество операндов;
- определяет допустимый тип операндов;
- имеет приоритет;
- имеет ассоциативность.

Операции

Операция	Действие	Тип операндов	Тип результата
+	сложение	целый/вещественный	целый/вещественный
+	сцепление строк	строковый	строковый
-	вычитание	целый/вещественный	целый/вещественный
*	умножение	целый/вещественный	целый/вещественный
/	деление	целый/вещественный	вещественный
div	целочисленное деление	целый	целый
mod	остаток от деления	целый	целый
not	арифметическое/логическое отрицание	целый/логический	целый/логический
and	арифметическое/ло-	целый/логичес-	целый/логический

Операции

shl	сдвиг влево	целый	целый
shr	сдвиг вправо	целый	целый
in	вхождение в множество	множество	логический
<	меньше	не структурированный	логический
>	больше	не структурированный	логический
<=	меньше или равно	не структурированный	логический
>=	больше или равно	не структурированный	логический
=	равно	не структурированный	логический
<>	не равно	не структурированный	логический

Стандартные функции

Обозначение	Тип результата	Тип аргументов	Действие
`abs(x)`	целый/вещественный	целый/вещественный	модуль числа
`sin(x)`	вещественный	вещественный	синус
`cos(x)`	вещественный	вещественный	косинус
`arctan(x)`	вещественный	вещественный	арктангенс
`pi`	без аргумента	вещественный	число π
`exp(x)`	вещественный	вещественный	экспонента e^x
`ln(x)`	вещественный	вещественный	натуральный логарифм
`sqr(x)`	вещественный	вещественный	квадрат числа
`sqrt(x)`	вещественный	вещественный	корень квадратный
`int(x)`	вещественный	вещественный	целая часть числа
`frac(x)`	вещественный	вещественный	дробная часть числа
`round(x)`	вещественный	целый	округление числа
`trunc(x)`	вещественный	целый	отсекание дробной части числа
`random(n)`	целый	целый	случайное число от 0 до n

Оператор (команда, инструкция)

наименьшая автономная часть языка программирования.

- оператор присваивания;
- операторы ввода-вывода;
- составной оператор;
- условные операторы;
- циклические операторы;
- оператор вызова процедуры.

Синтаксическая форма оператора присваивания:

```
1  L-выражение := R-выражение;
```

- L-выражение в результате вычисления должно указывать на переменную;
- R-выражение в результате вычисления должно быть совместимым по присваиванию с L-выражением;

Этапы выполнения оператора присваивания:

1. вычисляется значение R-выражение (в соотвествии с приоритетом операций);
2. вычисляется L-выражение;
3. значение R-выражения приводится к типу L-выражения;
4. в ячейку памяти (вычисленное на шаге 2) записывается вычисленное на шаге 3 значение.

Операторы вывода:

```
1  Write(Вып1:Фрмт1, Вып2:Фрмт2,...);
2  Writeln(Вып1:Фрмт1, Вып2:Фрмт2,...);
```

- Вып1, Вып2,... - выражения, которые будут выведены;
- Фрмт1, Фрмт2,... - формат вывода может задавать ширину поля (выражение целого типа) для вывода значений выражения и количество разрядов (выражение целого типа) для вывода выражения вещественного типа в формате с фиксированной точкой.

Операторы ввода:

```
1  Read(Выражение1, Выражение2, ...);
2  Readln(Выражение1, Выражение2, ...);
```