Основы языка Object Pascal

11.1 Алфавит языка

Основными символами языка Object Pascal являются:

символы _ + -

26 больших и **26** малых латинских букв **A,B, …Y,Z, a,b, … y,z**

10 арабских цифр **0, 1, 2, 3, 4, 5, 6, 7, 8, 9**

специальные символы *** / = ^ < > ( ) [ ] { } . , : ; ' # $ @**

Буквы русского алфавита не входят в состав алфавита языка. Их использование допустимо только в строковых и символьных значениях.

Нет различий при использовании больших и малых букв в записи имен переменных, процедур, функций и меток. Их максимальная длина ограничена 126 символами.

11.2 Лексическая структура языка

В Object Pascal различают следующие основные классы лексем:

1. **Зарезервированные (служебные) слова.** Этот класс состоит из слов, построенных только с помощью букв алфавита. Служебные слова можно использовать только по прямому назначению, т. е. так, как их назначение определил разработчик языка. Ни в каком другом виде, например в качестве имен переменных, их использовать нельзя.

Например:**and, do, end, for, if, interface, not, procedure, then, until, while, begin, const**

2. **Идентификаторы или имена** предназначены для обозначения констант, переменных, типов, процедур, функций, меток. Они формируются из букв, цифр и символа "_" (подчеркивание). Длина имени может быть произвольной, однако компилятор учитывает имена по его первым 63 символам. Внутри имени не должно быть пробелов.

Object Pascal в именах не различает больших и малых букв. Так следующие имена будут идентичны:

SaveToFile, SAVETOFILE, savetofile, sAVEtOfILE.

3. **Разделители** используются с целью большего структурирования модуля, с тем чтобы повысить визуальное восприятие длинных текстов. К их числу можно отнести **; := (** .

4. **Комментарии** используют для пояснения отдельных фрагментов текста программы. Они представляют собой последовательность символов, заключенную в фигурные скобки { } или в разделители (* и *), а также последовательность символов, расположенных в строке справа от двух следующих друг за другом символов / - //.

Примеры комментариев:

{ Функция вычисления количества дней между двумя датами }

(* Функция вычисления количества дней между двумя датами *)

// Неправильный ответ

5. Пробел. Этот символ не имеет видимого изображения и служит для отделения лексем друг от друга в тех случаях, когда это необходимо. Обычно использование одного или нескольких рядом стоящих пробелов не искажает смысл программы.

11.3 Основные понятия языка

1. Значение – это постоянная величина или структурный комплекс постоянных величин, выраженных в явном виде. Значение не имеет имени.

Примеры значений:

-55.455051 { обыкновенное вещественное число},

'Расчет посадки с натягом' {строка символов}.

2. Константа – это ячейка памяти, в которой всегда хранится одно значение. Константы не могут быть изменены в ходе выполнения программы. В этом смысле константа отвечает общепринятому определению постоянной (неизменяемой) величины. Всякая константа должна быть описана, т. е. должно быть указано ее значение. Значение константы определяет ее тип. Описание констант производится в языковой конструкции начинающейся словом **const.**

Пример описания константы:

```
Const
e=0,001;
```

Необходимо отметить, что в языке существуют так называемые **типизованные константы**, которые в ходе прохождения программы могут быть изменены. Тип константы указывается в специальной языковой конструкции, начинающейся словом **Type** (тип).

Переменная отличается от константы или значения тем, что в процессе работы программы она может менять свое значение.

Переменная – это ячейка, в которой в каждый момент времени хранится одно значение или не хранится ничего. Переменная в любой момент времени может быть изменена программой. Всякая переменная должна быть описана. т .е. должен быть явно указан ее тип. Тип переменной указывается в специальной языковой конструкции, начинающейся словом **Var** (от английского variable).

Пример описания переменных:

Var	Var
x,y,z:real;	x:real; y:real; z:real;
s,n:integer;	s,n:integer; n:integer;

Тип – это структура и описание множества значений, которые могут быть присвоены переменной, начинающаяся словом **Type** (тип).

11.4 Система типов

В языке Object Pascal все переменные должны быть предварительно описаны. Это означает, что всякая переменная должна быть отнесена к какому-либо типу.

В ОР наиболее часто применяют следующие типы: **простые и составные**.

Простые типы

целые [Integer],
вещественные [Real],
логический (булевский) [Boolean],
символьные [Char],
строковые [String].

Целые типы

Эта группа типов охватывает множество целочисленных значений. Они отличаются друг от друга диапазоном допустимых значений и количеством занимаемой памяти.

№	Тип	Диапазон значений	Размер памяти
1.	ShortInt	–128 .. 127	1 байт
2.	SmallInt	–32768 .. 32767	2 байта
3.	LongInt	–2147483648 .. 2147483647	4 байта
4.	Int64	–2^63 .. 2^63–1	8 байтов
5.	Byte	0...255	1 байт
6.	Word	0...65535	2 байта
7.	LongWord	0 .. 4294967295	4 байта

Так если значения переменной будут только положительными, то можно ее отнести к одному из типов Byte, Word, LongWord. Если известно также, что ее значения никогда не выйдут за 255 (например, если переменная предназначена для хранения номера месяца текущего года), то лучше использовать тип Byte. При этом память будет расходоваться наиболее экономно.

Над целыми значениями можно выполнять четыре обыкновенных арифметических действия: **сложение (+), вычитание (-), умножение (*), деление (/) и два дополнительных действия: деление нацело (div) и взятие остатка от деления (mod).** При выполнении деления результатом будет вещественное значение, во всех остальных операциях – целое.

Пример описания:

```
Var
ai,sum:integer;
```

Вещественные типы

Эта группа типов охватывает вещественные значения.

Вещественные типы не могут быть использованы: в качестве индексов массивов; в операторах For и Case.

№	Тип	Диапазон значений	Значащих цифр в мантиссе	Размер памяти
1.	Real48	2.9 x 10^–39 ... 1.7 x 10^38	11 – 12	6 байтов
2.	Single	1.5 x 10^–45 ... 3.4 x 10^38	7 – 8	4 байта
3.	Double	5.0 x 10^–324 ... 1.7 x 10^30	15 – 16	8 байтов
4.	Extended	3.6 x 10^–4951 ... 1.1 x 10^4932	19 – 20	10 байтов
5.	Comp	-2^63+1 ... 2^63 -1	19 – 20	8 байтов
6.	Currency	-922337203685477.5808... 922337203685477.5807	19 – 20	8 байтов

При описании вместо Real48 можно указывать Real.

Пример описания:

Var

ai,sum:Real;

Вещественные значения можно изобразить:

- в форме с фиксированной десятичной точкой;
- в форме с плавающей десятичной точкой.

Первая форма представления вещественного значения представляет привычное число, в котором целая и дробная части разделены десятичной точкой, например:

12.455 -988.45 -8.0

Вторая форма предназначена для записи очень больших или очень маленьких по абсолютной величине значений, когда их представление в форме с фиксированной точкой затруднительно или невозможно. Такое значение изображают в виде:

<значение с фиксированной точкой > E <порядок>

Пример:

-45.2E6 (то же, что -45,2 10^{6}) 5.245E-12 (то же, что 5,24 10^{-12})

Порядок таких чисел должен быть всегда целым числом.

Логический (булевский) тип

Логические переменные имеют тип boolean. Такая переменная занимает один байт памяти и может иметь одно из двух возможных значений – **True** (истина) или **False** (ложь).

Пример:

```
Var
b1, Ti : boolean;
```

Символьный тип

Типы AnsiChar и WideChar описывают множество отдельных символов языка, включая буквы русского алфавита. AnsiChar описывает множество из 256 ASCII-кодов и занимает один байт памяти, WideChar описывает множество Unicode – универсальное множество кодов и занимает два байта памяти. Тип AnsiChar эквивалентен базовому типу Char прежних версий языка.

Пример:

```
Var
Ch,k:AnsiChar;
Char_Massivr: array[1..100] of Char;
```

Символьное значение представляют в виде символа, заключенного с обеих сторон в апострофы. Для изображения самого апострофа его удваивают (последний пример), например:

'h' 'X' '#' '$' ''''

Строковые типы

Этот тип во многом схож с типом Array of Char, т. е. массивом символов. Отличие состоит в том, что переменная этого типа может иметь динамическое количество символов (от нуля до верхней границы), в то время как массив символов всегда статичен и имеет одинаковое количество символов.

№	Тип	Длина строки	Занимаемая память
1.	ShortString	0 – 256 символов	(Кол-во символов) х 1 байт
2.	AnsiString	0 – 2 Гб символов	(Кол-во символов) х 1 байт
3.	WideString	0 – 2 Гб символов	(Кол-во символов) х 2 байта

Максимальная длина строковой переменной должна быть указана явно.

Размер строки на единицу больше ее объявленной длины, т. к. в ее нулевом байте содержится фактическая длина строки. Длину в нулевом байте можно принудительно менять.

Строковое значение изображают в виде последовательности символов, заключенной в апострофы. Пустую строку изображают двойным апострофом.

Примеры значений строковых типов:

'Иванов И.И.' " 'Газета"ИЗВЕСТИЯ"' 'Строка символов'

Примеры описания переменных строковых типов:

```
Var
  s1,s2:ShortString[12];
  st1,st2:AnsiString[580];
  ChMassiv: array [1..15] of String;
```

11.5 Описание переменных

Описание переменной или группы переменных начинается словом **Var.**

Общий вид описания переменных одного типа:

<переменные> : <тип>;

Переменными могут быть объявлены не только переменные простых типов. Ниже будут рассмотрены переменные более сложных – структурных – типов. Более того, переменными могут быть объявлены структуры структур, примером которых являются классы.

```
Var.

 I, n, m:integer ;
k, L:integer; s, sum:real;
```

11.6 Описание констант

В Object Pascal различается два вида констант – обыкновенные и типизованные. Описание констант следует после слова Const.

Обыкновенные константы

Описание константы строится по правилу:

<имя константы> = <выражение или значение>;

Пример:

```
Const
Par = 12899;
StarString = '* * * * * * * * * * * * * *';
Log10 = 2.302585;
Log10_Invert = 1/Log10;
```

Тип константы определяется автоматически по виду ее значения.

Типизованные константы

Это специальный тип констант, которые отличаются от обыкновенных констант тем, что при их описании необходимо указывать тип.

Простые типизованные константы. Общий вид константы:

<имя константы> : <тип> = <значение>;

Пример:

```
Const
CurrentPosition: Word = 11000;
LastLetter: Char = 'z';
HeadOfModule: String[26] = 'Начало программного модуля';
```

Значение типизованных констант можно изменять в ходе выполнения программы, и они могут быть использованы в качестве Var-параметра процедуры или функции. В этой связи типизованные константы по сути являются переменными с начальным значением.

Типизованные константы типа "массив". Этот тип констант позволяет обозначить постоянной величиной целый массив однотипных значений.

Типизованные константы типа "запись". Это комбинированный тип констант, основанный на конструкциях типа Record (Запись), которые состоят из полей.

Типизованные константы типа "множество". Эти константы могут быть построены как подмножества базовых или производных от них типов.

11.7 Описание пользовательских типов

Ранее уже приводились примеры описания переменных, в которых их тип указывался в Var-секции.

Описание секции типов начинается словом **Type**.

В таблице дан пример двух идентичных способов описания переменных t, u, n.

Явный способ описания переменных	Описание переменных с предварительным описанием их типа
Var t,u,n:(Mon, Tue,Wed, Thu,Fri,Sat,Sun);	Type DaysOfWeek = (Mon, Tue,Wed, Thu,Fri,Sat,Sun); Var t,u,n: DaysOfWeek;

Следует отдавать предпочтение способу описания переменных с предварительным объявлением их типа в секции **Type**. Такой способ позволяет:

а) конкретизировать тип;

б) четко выделить множество переменных этого типа;

в) повысить уровень структурированности программы;

г) снизить вероятность путаницы в типах, когда переменные фактически того же типа объявлены разными способами;

д) уменьшить объем текста за счет возможности быстрой ссылки на ранее определенный тип, особенно в тех ситуациях, когда этот тип используется для порождения новых типов, переменных, функций и пр. в других секциях или модулях.

Пример:

```
Type
t1 = byte;
Var
p1,p2: t1;
```

11.8 Структурные типы

Структурные типы представляют собой совокупность значений одного или нескольких различных типов. Их наличие позволяет программисту конструировать производные типы практически любой сложности, что резко расширяет возможности языка.

К числу структурных относятся следующие типы:

- множественные типы [Set],
- регулярные типы (массивы) [Array],
- комбинированные типы (записи) [Record],
- файловые типы [File],
- классы [Class].

Регулярные типы (массивы).

Массив – это структура языка Object Pascal, представляющая собой упорядоченную совокупность элементов одного типа.

Следует различать два вида массивов: массив-тип и массив-переменную.

Массив-тип. Синтаксис массива-типа

Одномерный массив

Type

<имя массива> = Array [<диапазон индексов1] of <тип элемента>;

Двухмерный массив

Type

<имя массива> = Array [<диапазон индексов1>, <диапазон индексов2>] of <тип элемента>;

Всякий массив имеет **размерность**. Размерность определяется количеством индексов, которые заключены в квадратные скобки [..].

Пример описания одномерного массива

```
Type
Mas=array [1 .. 10] of  Real;
Var
 a:mas;
```

описана структура одномерного массива вещественных элементов (Real), в котором индекс может изменяться в диапазоне целых значений от 1 до 10. Его элементами являются вещественные типы a[1], a[2], a[3], …, a[9], a[10].

Пример описания двухмерного массива

```
Type
Ma= array [1 .. 10, 1..10] of  integer;
Var
 b,c:ma;
```

описана структура одномерного массива целых элементов (integer), в котором первый и второй индексы могут изменяться в диапазоне целых значений от 1 до 10. Его элементами являются вещественные типы b,c [1,1], b,c[1,2], b,c[1,3], …b,c [1,10], b,c[2,1], b,c[2,2], b,c[2,3], …b,c [2,10], … b,c [10,10].

Массив-переменная. Синтаксис массива-переменной

Одномерный массив

Var
<имя массива> :Array [<диапазон индексов1] of <тип элемента>;

Двухмерный массив

Var
<имя массива>:Array [<диапазон индексов1>, [<диапазон индексов2>] of <тип элемента>;

Массив-переменная отличается от массива-типа тем, что все его элементы – это отдельные независимые переменные, которые могут содержать различные значения одного типа.

Пример описания одномерного массива

```
Var
A:array[1 .. 10] of Real;
```

Пример описания двухмерного массива

```
Var
b,c : array [1 .. 10, 1..10] of integer;
```

Дать примеры

Комбинированные типы (записи).

Запись – это объединение элементов разных типов. Как и в массивах, следует различать запись-тип и запись-переменную. Один элемент записи называется полем.

Запись-тип. Синтаксис записи-типа:

<имя записи> = Record
<имя поля 1> : <тип>;
<имя поля 2> : <тип>;
…
<имя поля N> : <тип>;
<вариантная часть >
End;

Записи очень удобны для описания и хранения разнотипных данных о каких-либо однотипных структурах.

Примером могут служить сведения о студентах. Запись включает поля: Фамилия, Имя, Отчество, Год рождения, Группа, Год поступления в вуз, Курс. Такие структуры являются однотипными и могут быть описаны следующим образом:

```
Type
prop=array [1..1000] of integer;
Stud=Record {Сведения о студенте как запись }
Fam:String[40]; {ФИО как строка из 40 символов }
Name:String[20];{Имя как строка из 20 символов }
Otch:String[30];{Отчество как строка из 30 символов }
BirthYear: integer; {Год рождения как целое типа integer }
Group:String[8];{Группа как строка из 8 символов }
ElectYear: integer; {Год поступления как целое типа integer }
Curs:integer;{Курс как целое типа integer }
Ocenka:array[1..5] of integer; {Оценка  как массив целых элементов}
Colchas:prop;
End;
Var
Tabl:stud;
```

Запись-переменная. Синтаксис записи-переменной:

Var
<имя записи> : Record
<имя поля 1> : <тип>;
<имя поля 2> : <тип>;
...
<имя поля N> : <тип>;
<вариантная часть >
End;

т.е. синтаксисы записи переменной и записи типа, как и у массивов отличаются разделом в котором производится описание (var или type)и символом разделителем (":" и "="). Если описание произведено в секции **Type** то для использования данного типа необходимо создать переменную этого типа в разделе **var** (см. примеры)

```
Var
Tabl:Record
Fam:String[40];
Name:String[20];
Otch:String[30];
BirthYear: integer;
Group:String[8];
ElectYear: integer;
Curs:integer;
Colchas:array[1..1000] of integer;
Ocenka:array [1..5] of integer;

End;
```

Доступ к полям записей. Переменная, представляющая поле, конструируется из имени записи и поля, отделенного друг от друга десятичной точкой.

Например

```
Tabl.Fam='Иванов‘:
Tabl. Name=‘Иван’;
```

Доступ к полям записей с помощью оператора присоединения With.

Для упрощения обращения к полям одной и той же записи можно использовать оператор With.

With <значение>**do**
Begin
<оператор1>
…
<оператор N>
End;

Пример:

Фрагмент программы без with

```
…
Tabl.Fam='Иванов‘:
Tabl. Name=‘Иван’;
Tabl. Otch=‘Петрович’;
...
End;
```

С with

```
...
With tabl do
        Begin
        Fam='Иванов‘:
        Name=‘Иван’;
Otch=‘Петрович’;
End;
…
```

11.11 Совместимость типов

Необходимым условием корректного вычисления выражений или выполнения операторов присваивания является совместимость типов входящих в них компонент.

Совместимость по вычислению

Вычисление выражений возможно только при соблюдении следующих условий.

Типы операций и операндов эквивалентны.

Например, нельзя применять арифметические операции к логическим переменным и, наоборот, логические операции – к арифметическим переменным.

Совместимость по присваиванию

Оператор присваивания считается корректным, если тип переменной, расположенной в его левой части, совместим с типом выражения, расположенного в правой части.

11.12 Выражения

Вычислительная система выполняет вычислительные и управляющие операции по командам, которые представлены в программе с помощью операторов.

Большинство таких операторов строится с использованием выражений, которые в практике программирования играют большую роль, определяя способ и порядок преобразования данных.

Выражения состоят из операндов (значений, констант, переменных, функций), соединенных с помощью операций. Для изменения порядка выполнения операций могут быть использованы круглые скобки. Наиболее важную роль играют арифметические, логические и строковые выражения.

Арифметические выражения.

Арифметическим называется выражение состоящее из операндов и арифметических операций.

Операндом называется любая компонента, к которой применяется операция.

Операндом может быть, например, значение, константа, переменная или выражение заключенное в скобки.

В выражениях можно применять круглые открывающие и закрывающие скобки. При этом количество открывающих скобок должно быть равно количеству закрывающих скобок.

При вычислении выражения операции выполняются в строго определенной последовательности в соответствии с их приоритетом. Порядок выполнения операций можно изменить скобками.

В языке Object Pascal существует шесть арифметических операций. Учитывая, что арифметические операции образуют подмножество множества всех операций языка, в табл. 11.6 показано абсолютное значение приоритета каждой операции.

Операция	Наименование	Приоритет
+	Сложение	2
-	Вычитание	2
*	Умножение	1
/	Деление	1
div	Деление нацело	1
mod	Остаток от целочисленного деления	1

При вычислении выражения его тип определяется типами операндов.

Операция	Тип операндов	Тип результата
+	Integer, real	integer, real
–	Integer, real	integer, real
*	Integer, real	integer, real
/	Integer, real	real
div	Integer	integer
mod	Integer	integer

Пример:

4*5 = 20, 6/5 = 1.2, 8+7 = 15, 7-3 = 4, 16 div 5 = 3, 16 mod 5 = 1.

Порядок выполнения операций определяется приоритетом операций и расположением внутренних выражений, заключенных в круглые скобки. Все операции в арифметическом выражении выполняются слева направо.

Пример:

Выражение:

15 * ((25/5-5*9 + (j-8) * 7.55) / 8.67)

8 2 5 3 6 1 4 7

Логические выражения

Логическое выражение состоит из других выражений, (арифметических, строковых и др.), значений, констант, переменных, функций, **логических операций** и **логических отношений**.

Результатом вычисления логического выражения может быть одно из двух логических значений: True (истина) или False (ложь).

В языке существует четыре логических операций. Приоритет операций показан в табл. 11.8.

Операция	Наименование	Приоритет
Not	Отрицание	3
And	Конъюнкция	4
Or	Дизъюнкция	5

Значения элементарных логических выражений, поясняющих назначение этих операций, приведены в табл. 11.9

A	B	not A	A and B	A or B
True	True	False	True	True
True	False	False	False	True
False	True	True	False	True
False	False	True	False	False

В табл. 11.10 представлены логические отношения.

Отношение	Наименование
>	Больше
<	Меньше
>=	Больше или равно
<=	Меньше или равно
=	Равно
<>	Не равно

Все отношения равноприоритетны. Порядок выполнения операций при вычислении логического выражения следующий:

- сначала вычисляются арифметические выражения;
- затем – отношения;
- в последнюю очередь вычисляются логические операции.

Выражение:	Not ((x > 6 + 8 * 2) and (y < 7) or (z > 7)) and (x <> y)								
Порядок:	9	3	2	1	6	4	7	5	10	8

Строковые выражения

Строковые выражения, частными случаями которых могут быть пустой символ ‘ ‘ или одиночный символ (например ‘A’), строятся из строковых или символьных значений, констант, переменных и строковых функций при помощи строковой операции **конкатенации** (**присоединения**).

Эта операция обозначена символом + (плюс). Скобки в строковых выражениях не применяются.

Пример:

Выражение: 'Object '+'Pascal '+' для Delphi‘

Результат: 'Object Pascal для Delphi‘

Для организации перехода на следующую строку используется код клавиши “Enter” #13

Выражение: ‘Object ‘+’Pascal ‘ # 13‘для Delphi‘

Результат: 'Object Pascal
для Delphi‘