Глава 5. БАЗОВЫЕ ЭЛЕМЕНТЫ ЯЗЫКА ПРОГРАММИРОВАНИЯ (Object Pascal, Turbo Delphi)

- Синтаксис и семантика языка программирования
- Методы описания синтаксиса
- Идентификаторы
- Константы
- Адресация в оперативной памяти
- Переменные
- Типы
- Выражения и операции
- Скалярные типы данных (логический, целые, символьный, перечисляемый, тип-диапазон, вещественные)
- Арифметические и логические операции, операции отношения
- Приоритет операций
- Структура программы

СИНТАКСИС И СЕМАНТИКА

Синтаксис языка программирования – это форма, а **семантика** – смысл его выражений, операторов и программных единиц.

Формальное описание синтаксиса языка строится с использованием терминалов и нетерминалов. Конструкции языка, для которых требуются определения, называются нетерминальными символами или просто **нетерминалами** (например, <оператор>, <число>, <условный оператор> и т. д.). Число нетерминалов равно числу правил языка. Собственные символы (символы алфавита) и конструкции языка, для которых не требуется специальное описание, называются терминальными символами или **терминалами**.

Алфавит языка Delphi Pascal включает:
- прописные и строчные латинские буквы от **a** до **z** и от **A** до **Z** , а также знак подчеркивания
- арабские цифры от **0** до **9**
- специальные символы **+ - * / = , ‘ . : ; < > [ ] ( ) { } ^ @ $ # &** и пробел
- составные символы **:= <> .. <= >= (* *) (. .) //**

Терминалами также являются зарезервированные слова (**begin end array const if then else** и т.д.), в состав которых входят стандартные директивы (**absolute assembler external far forward near private virtual** и др.).

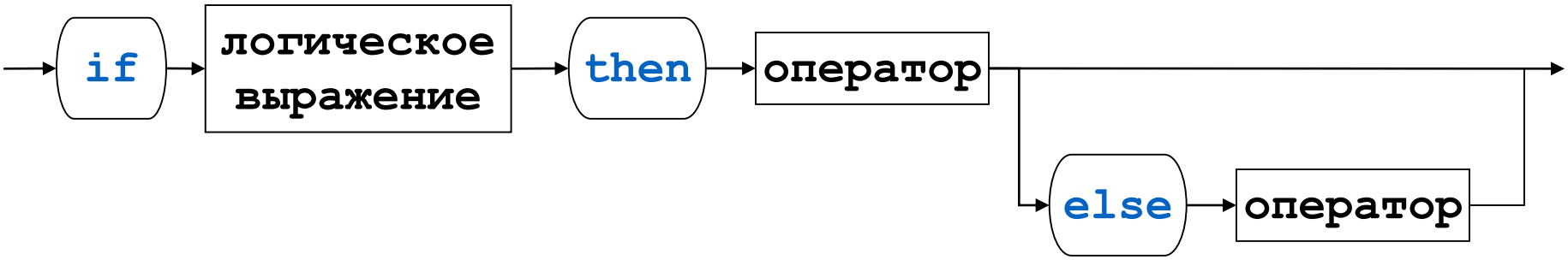
МЕТОДЫ ФОРМАЛЬНОГО ОПИСАНИЯ СИНТАКСИСА ЯЗЫКА

Форма Бэкуса-Наура (БНФ):

```
<цифра> → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<условный оператор> → if <логическое выражение> then
                        <оператор>
                      | if <логическое выражение> then
                        <оператор>
                        else <оператор>
```

Синтаксическая диаграмма (условный оператор):



ИДЕНТИФИКАТОРЫ

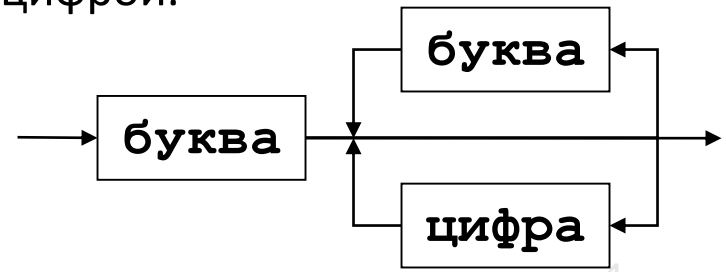
Текст программы состоит из **лексем**. Лексемы языка Delphi Pascal представлены такими категориями как специальные символы, идентификаторы, зарезервированные слова, метки, числовые константы и строковые константы. Две соседние лексемы должны отделяться друг от друга разделителями.

Идентификатор (имя) – это строка символов, используемая для идентификации некоторой сущности в программе (переменной, метки, подпрограммы, параметра и т.д.). Идентификаторы могут быть стандартные и пользовательские. Стандартные идентификаторы служат для обозначения заранее определенных разработчиками языка типов данных, констант, процедур и др.

Правила построения идентификаторов:

- идентификаторы могут иметь произвольную длину, но значащими будут только первые 255 символов;
- всегда начинается буквой, за которой могут следовать буквы и цифры (символ подчеркивания считается буквой, пробелы и специальные символы недопустимы);
- имена меток могут начинаться как буквой, так и цифрой.

Синтаксическая диаграмма конструкции <идентификатор>



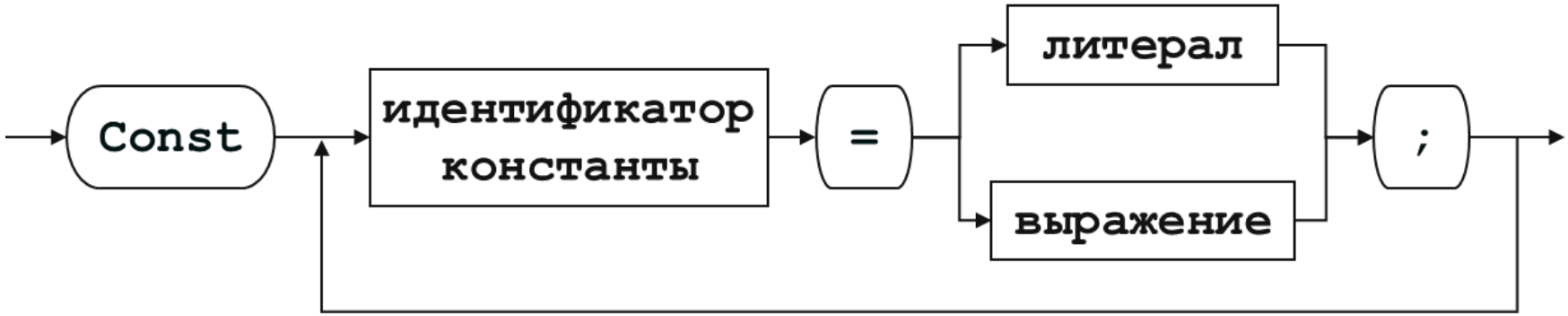
КОНСТАНТЫ

Языки программирования интерпретируют данные как константы или переменные.

Константа – элемент данных, значение которого не изменяется в процессе выполнения программы.

Литерал представляет собой значение константы, записанное непосредственно в программе (например, в выражении 2 + 5.1*x использованы два литерала "2" и "5.1").

Поименованная константа объявляется в инструкции секции описания констант Const.




```
Const
   Ln10 = 2.3026;
   Ln10R = 1 / Ln10;
   X = MaxInt div SizeOf(Real);
```

КОНСТАНТЫ

Значениями констант в языке Delphi Pascal могут быть:

- Целые числа (от -9 223 372 036 854 775 808 (-2^{63}) до 9 223 372 036 854 775 807 ($2^{63}-1$)).
- Вещественные числа. Может использоваться формат с фиксированной точкой или формат с плавающей точкой (a = 61.2; b = 3.14e5; c = -72E-3).
- Шестнадцатеричные числа (в диапазоне от $0000 0000 0000 0000 до $FFFF FFFF FFFF FFFF).
- Логические константы (True, False).
- Символы. Записываются в апострофах, кроме того допускается использование записи символа путем указания его внутреннего кода, которому предшествует символ # (Znak1 = 'd'; Znak2 = 'ф'; Znak3 = #90).
- Строки символов. Любая последовательность символов, заключенная в апострофы. Можно составить из кодов нужных символов с предшествующими каждому коду знаком #.
- Конструкторы множеств. Список элементов множества, обрамленный квадратными скобками (a = [1,2,4..7,12]; b = [red,blue,green]; c = [True]).
- Признак неопределенного указателя - NIL.

АДРЕСАЦИЯ

Оперативная память представляет собой последовательность ячеек емкостью 1 байт (8 бит). Доступ к любой ячейке осуществляется путем указания ее адреса.

MS-DOS. Память разбита на **сегменты** – непрерывные области памяти размером 64 К (65536 байт). Начало любого сегмента отстоит от начала предыдущего на 16 байт. **Смещение** – номер ячейки памяти, отсчитанный относительно той ячейки, с которой начинается сегмент.

Логический адрес конкретной ячейки: **`сегмент : смещение`**

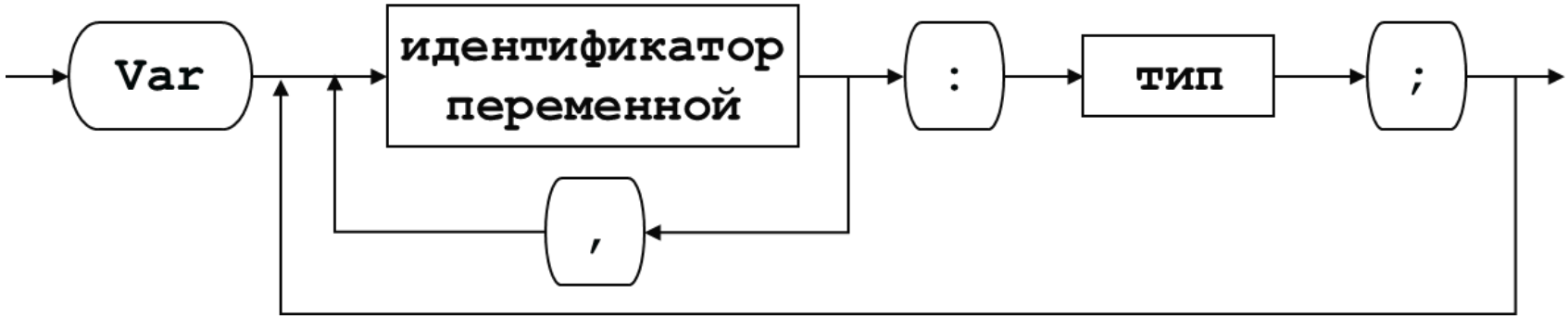
`физический адрес = сегмент*10h + смещение`

Например, ячейка с физическим адресом `00012h` может иметь логический адрес `0001h : 0002h` либо `0000h : 0012h`.

Windows. Приложения Windows имеют доступ ко всей памяти компьютера. Используется "страничная" модель памяти. Под адресом будем понимать целое число размером 32 бита (4 байта). Например, `1111FFFFh`

ПЕРЕМЕННЫЕ

Переменной называется поименованная область в памяти компьютера, выделяемая для хранения конкретных данных, значение которой может изменяться в ходе выполнения программы. Переменные интерпретируются однозначным способом – с помощью явного указания ее типа в специальном разделе описаний. Тип соответствует количеству байт, которые должны быть выделены под эту переменную в соответствии с теми значениями, которые сможет принимать переменная.



Возможна инициализация переменных в их объявлениях:

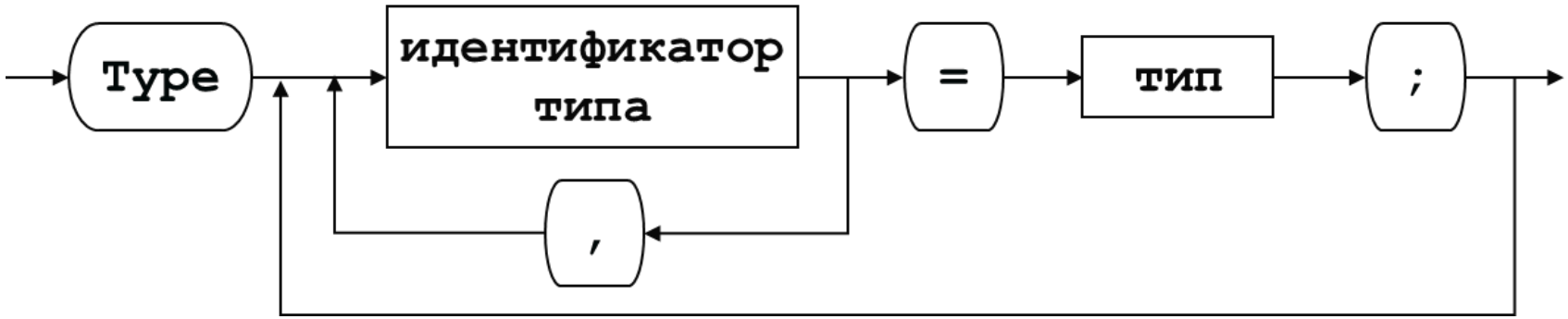
```
Var
   S : string = 'Start';
```

Типизированная константа – переменная с заранее определенным значением.

```
Const
   A : Real = 5.6;
```

ТИПЫ

Типы с уникальными именами описываются в специальной секции раздела описаний.




```
Type
   LightType = (red, yellow, green);
Var
   A, B, StreetLight : LightType;
```

Если значение не структурное, то есть не распадается на компоненты, то оно называется **скаляром**, а сам тип, описывающий такие значения – **скалярный тип**.

Параметры-атрибуты переменной: имя, адрес, значение, тип, время жизни, область видимости.

ВЫРАЖЕНИЯ, ОПЕРАЦИИ

Выражение задает порядок выполнения действий над элементами данных и состоит из операндов (константы, переменные, обращения к функциям), знаков операций и круглых скобок. **Операции** определяют действия, которые надо выполнить над операндами. Операция – атрибут типа.

```
(X + sin(Y) - 10)*4
```

`X,sin(Y),10,4` – операнды, `"+","-","*"` – операции.

Унарные операции относятся к одному операнду и всегда записываются перед ним (not, @, +, -).

Бинарные операции выражают отношение между двумя операндами и записываются между ними (X + Y).

ЛОГИЧЕСКИЙ ТИП

Значениями переменных логических (булевских) типов могут быть логические константы True, False.

```
Var
   X : Boolean;
```

Логические типы: **Boolean** (1 байт), ByteBool (1 байт), WordBool (2 байта), LongBool (4 байта).
Логические переменные могут выступать операндами в логических операциях.

Операнд 1	Операнд 2	and (конъюнкция)	or (дизъюнкция)	xor (исключающее ИЛИ)	not (отрицание)
False	False	False	False	False	-
True	False	False	True	True	-
False	True	False	True	True	-
True	True	True	True	False	-
True	-	-	-	-	False
False	-	-	-	-	True

ЦЕЛОЧИСЛЕННЫЕ ТИПЫ

Название	Длина, байт	Диапазон значений
Byte	1	0 .. 255
ShortInt	1	-128 .. +127
Word	2	0 .. 65535
SmallInt	2	-32768 .. +32767
Cardinal (LongWord)	4	0 .. 4 294 967 295
Integer (LongInt)	4	-2 147 483 648 .. +2 147 483 647
Int64	8	-2^{63} .. $+2^{63} - 1$

К операндам целых типов могут быть применены арифметические операции.

Результат операций сложения "+", вычитания "-", умножения "*", целочисленного деления **div**, возвращения остатка от целочисленного деления **mod** двух переменных X и Y относится к целому типу **Integer** (либо к Int64, если X или Y имеет тип Int64). Результат операции деления "/" относится к наиболее мощному вещественному типу Extended.

```
Var X : Byte; Y : ShortInt;
... Z := X + Y; {Z → Integer}
```

ЦЕЛОЧИСЛЕННЫЕ ТИПЫ

Арифметические операции **сдвига** K shl N и K shr N оперируют с двоичным (битовым) представлением целых чисел. Они восстанавливают в качестве результата значение, полученное путем сдвига на N позиций влево или вправо числа K, представленного в двоичном виде. Тип результата – Integer.

$2\ \text{shl}\ 7 = 256;\quad (2)_{10} = (10)_2;\quad (100000000)_2 = (256)_{10}.$

$161\ \text{shr}\ 2 = 40;\quad (161)_{10} = (10100001)_2;\quad (101000)_2 = (40)_{10}.$

При использовании логических операций применительно к целым числам осуществляется побитное (поразрядное) сравнение операндов. Тип результата – наименьший целый, включающий типы операндов. Двоичные цифры результата образуются по следующим правилам:

Операнд 1	Операнд 2	and	or	xor	not
0	0	0	0	0	-
1	0	0	1	1	-
0	1	0	1	1	-
1	1	1	1	0	-
1	-	-	-	-	0
0	-	-	-	-	1

$12\ \text{or}\ 22 = 30;\quad (12)_{10} = (01100)_2;\quad (22)_{10} = (10110)_2;\quad (11110)_2 = (30)_{10}.$

$X : \text{Byte} = 5;\quad (X)_2 = 00000101;\quad X := \text{not}\ X = (11111010)_2 = \underline{(250)_{10}}.$

$X : \text{ShortInt} = 5;\quad (X)_2 = 0\ 0000101;\quad X := \text{not}\ X = (1\ 1111010)_2 = \underline{(-6)_{10}}.$

СИМВОЛЬНЫЕ ТИПЫ

Значениями переменных символьного типа является множество всех символов компьютера. Каждому символу приписывается целое число (код).
В основе кодировки лежит 7-битный код **ASCII** (American Standard Code of Information Interchange). Символы с кодами 128..255 отводятся под национальный алфавит – 1-байтовые кодировки CP866 (DOS), KOI8-R, ANSI CP1251 (Win)). Unicode – 2 байта.

```
Var
   X : Char;
...
X := 'ф';
X := #244;      {CP1251}
```

Символьные типы – **Char**, ANSIChar (1 байт); WideChar (2 байта).

Применимы встроенные функции:

CHR(X) – преобразует выражение X : Byte в символ и возвращает этот символ;

UPCASE(X) – возвращает прописную букву, если X – строчная латинская буква, в противном случае (например, если X – русская буква) возвращает сам символ X.

ПЕРЕЧИСЛЯЕМЫЙ ТИП

Перечисляемый тип задается перечислением тех значений, которые он может получать. Значения перечисляемого типа должны иметь синтаксис идентификаторов. Во внутреннем представлении значения перечисляемого типа кодируются целыми числами, начиная от нуля.

```
Type
   Colors = (red, white, blue);
Var
   Color1, Color2 : Colors;
Var
   Color3 : (red, white, blue);
...
   Color3 := red;
```

ТИП-ДИАПАЗОН

Тип-диапазон – это подмножество базового типа, в качестве которого может выступать любой целый, логический, символьный или перечисляемый тип. Тип-диапазон задается границами своих значений внутри базового типа.

```
Type
   Digit1 = '1'..'9'; {базовый тип – символьный}
   Digit2 = 1..9; {базовый тип – целый}
Var
   LatCht : 'A'..'Z'; {базовый тип – символьный}
```

Стандартные функции:

HIGH(X) – возвращает максимальное значение типа-диапазона.

LOW(X) – возвращает минимальное значение типа-диапазона.

ПОРЯДКОВЫЕ ТИПЫ

Рассмотренные скалярные типы (целые, символьные, логические, перечисляемый, тип-диапазон) относятся к классу **порядковых типов**. Все значения порядкового типа можно упорядочить, с каждым из них можно сопоставить целое число – порядковый номер значения.

К любому из порядковых типов применима функция **ORD(X)**, которая возвращает порядковый номер значения X.
Целые типы: ORD(X) = X (кроме Int64).
Логический тип Boolean: ORD(False) = 0, ORD(True) = 1.
Символьные типы: ORD(X) = код символа X.
Перечисляемый тип: ORD(X) = целое число 0..65535 в соответствии с положением X в списке.
Тип-диапазон: ORD(X) определяется свойствами базового типа.

Стандартные функции:
PRED(X) – возвращает предыдущее значение порядкового типа;
ORD(PRED(X)) = ORD(X) - 1.
SUCC(X) – возвращает следующее значение порядкового типа;
ORD(SUCC(X)) = ORD(X) + 1;

```
Var C,D : Char;
begin C := 'f'; D := PRED(C); end.
```

ВЕЩЕСТВЕННЫЕ ТИПЫ

Обработка вещественного числа выполняется с некоторой конечной точностью, зависящей от его внутреннего формата (типа).

Название	Длина, байт	Количество значащих цифр	Диапазон значений
Real	**8**	**15...16**	**$\pm 5.0 \cdot 10^{-324}$ … $\pm 1.7 \cdot 10^{308}$**
Real48	6	11…12	$\pm 2.9 \cdot 10^{-39}$ … $\pm 1.7 \cdot 10^{38}$
Single	4	7...8	$\pm 1.5 \cdot 10^{-45}$ … $\pm 3.4 \cdot 10^{38}$
Double	8	15...16	$\pm 5.0 \cdot 10^{-324}$ … $\pm 1.7 \cdot 10^{308}$
Extended	10	19...20	$\pm 3.6 \cdot 10^{-4932}$ … $\pm 1.1 \cdot 10^{4392}$
Comp	8	19...20	-2^{63} … 2^{62}
Currency	8	19…20	–922 337 203 685 477.5808 … 922 337 203 685 477.5807

ПРЕДСТАВЛЕНИЕ ВЕЩЕСТВЕННЫХ ЧИСЕЛ В ПАМЯТИ

Структура вещественных чисел в памяти компьютера соответствует представлению в формате с плавающей точкой:

s	e	m
знак (1 бит)	порядок(d бит)	мантисса (r бит)

1 + d + r = 8N, где N – число байт, отводимых под переменную данного типа (Single – d = 8 бит, r = 23 бита; Extended – d = 15 бит; r = 64 бита).

s = 0, если знак числа "+"; s = 1, если знак "-";
e – задает истинный порядок числа $t = e - (2^{d-1} - 1)$;
m – задает мантиссу (дробную часть) $m_1 = m \cdot 2^{-r}$ $(0 \le m_1 < 1)$.

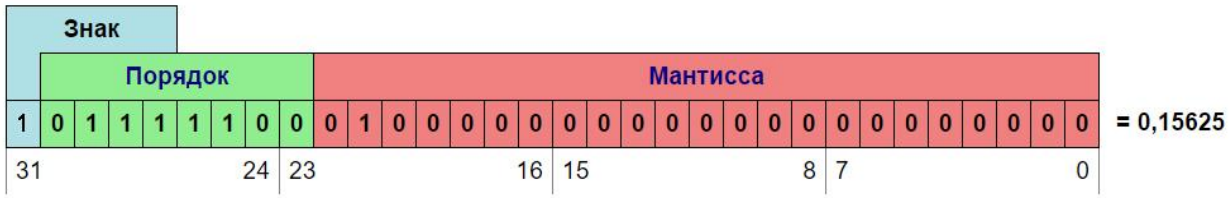
Записанное число может быть определено по формуле:

$$(-1)^s \cdot (1+m_1) \cdot 2^t \leftarrow \textbf{нормализованная запись}$$

Пример: $(-0.15625)_{10}$ как тип Single – N = 2 байта (s = 1, d = 8, r = 23).

$(-0.15625)_{10} = (-0.00101)_2 = (-1.01)_2 \cdot 2^{-3}$, т.е. $t = (-3)_{10}$, $m_1 = (0.01)_2$ => s = 1,

e = $-3 + (2^7 - 1) = (124)_{10} = (0111\ 1100)_2$, m = $m_1 \cdot 2^{23} = (10\ 0000\ 0000\ 0000\ 0000\ 0000)_2$



ВЕЩЕСТВЕННЫЕ ТИПЫ

Вещественные типы введены в расчете на **арифметический сопроцессор** – устройство, которое подключается непосредственно к центральному процессору (или интегрировано в ЦП) и предназначено для выполнения операций над числами в формате с плавающей точкой и длинными целыми числами. Сопроцессор всегда обрабатывает числа в формате типа Extended (остальные вещественные типы получаются усечением результатов до нужных размеров).

```
Var
  E1,E2 : Extended; E3 : Double; Result : Single;
  ...
  Result := E1*E2/E3;
```

Тип **Currency** используется для представления денежных единиц. В памяти он хранится как масштабированное в 10 000 раз 8-байтовое целое (минимизируются ошибки округления). Совместим с другими вещественными типами.

Значения типа **Comp** – вещественные представления "больших" целых чисел. Оставлен для обратной совместимости с ранними версиями языка.

Возможные ситуации, когда результат арифметических операций не удовлетворяет ограничениям на диапазон (или форму) представляемых чисел:

– переполнение (`Var X,Y:SmallInt;...X:=20000; Y:=15000; X:=X+Y;`)
– "исчезновение порядка".

ОПЕРАЦИИ ОТНОШЕНИЯ

Операции отношения выполняют сравнение двух совместимых операндов и определяют, истинно значение выражения или ложно. Определены для скалярных типов, строк, множеств и др. Результат – True или False.

Операция	Название	Выражение	Результат
=	Равно	A = B	*True*, если A равно B
<>	Не равно	A <> B	*True*, если A не равно B
>	Больше	A > B	*True*, если A больше B
<	Меньше	A < B	*True*, если A меньше B
>=	Больше или равно	A >= B	*True*, если A больше или равно B
<=	Меньше или равно	A <= B	*True*, если A меньше или равно B
in	Принадлежность	A in M	*True*, если A принадлежит множеству M

ПРИОРИТЕТ ОПЕРАЦИЙ

Операции	Приоритет
@, not	1 (высший)
*, /, div, mod, and, shr, shl	2
+, -, or, xor	3
>, <, <>, =, <=, >=, in	4 (низший)

Правила для определения старшинства операций:

- операнд, находящийся между двумя операциями с различными приоритетами, связывается с операцией, имеющей более высокий приоритет;
- операнд, находящийся между двумя операциями с равными приоритетами, связывается с операцией, которая находится слева;
- выражение, заключенной в скобки, перед выполнением вычисляется, как отдельный операнд.

Результат операции @ – адрес операнда.

СТРУКТУРА ПРОГРАММЫ

```
Program Example;
{$APPTYPE CONSOLE} {директива консольного приложения}
Uses SysUtils;     //подключение модулей (библиотек)
Label
   1,m1,Stop;
Const
   MaxN: Word = 100;
   Kod = $20;  {Шестнадцатеричная константа}
Type
   Matrix = array [1..Kod] of Real;
Var
   A,B : Integer; C : Integer = 5;
   Result : Matrix;
Procedure <имя>;
   begin <тело процедуры> end;
Function <имя> : <тип>;
   begin <тело функции> end;
BEGIN
   <операторы>;
END.
```

СТРУКТУРА ПРОГРАММЫ

Программа состоит из заголовка, раздела описаний и раздела операторов. В строке может быть несколько операторов (объявлений). Разделитель – символ ";".

Любой подраздел раздела описаний может отсутствовать, если в нем нет необходимости. Подразделы **`Label, Type, Const, Var`** могут следовать друг за другом в любом порядке и встречаться в разделе описаний сколько угодно раз. При этом обязательно соблюдение правила: все, что в программе используется, должно быть перед этим описано.

Комментарии – это произвольный поясняющий текст в любом месте программы, заключенный в фигурные скобки { }, или между двойными символами (* *), или после символа //.

Директивы компилятора используются программистом для управления режимами компиляции (включать или выключать контроль ошибок, изменять распределение памяти и т.д.). Они заключаются в фигурные скобки и имеют отличительный признак $. При использовании директив-переключателей указывается буква-ключ с последующим знаком "+" (включить режим) или "-" (выключить). Например, {$R-} – отключить проверку принадлежности заданному диапазону. Директивы-параметры устанавливают режимы и параметры компиляции. Например, {$APPTYPE CONSOLE} – директива консольного приложения.