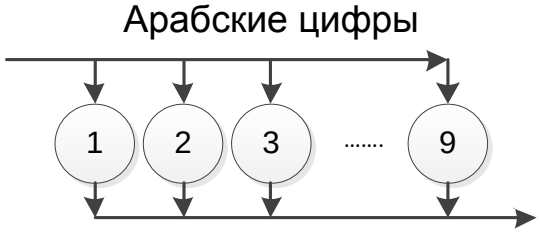
ОСНОВЫ ПАСКАЛЯ

1. Структура (состав) языка

В состав языка включается обычно **три составляющие**: алфавит, синтаксис и семантика. Кроме того в составе синтаксиса выделяют еще так называемую **лексическую структуру языка.**

Алфавит – это фиксированный для данного языка набор основных символов, из которого будут строиться все основные конструкции языка. Алфавит задается перечислением всех допустимых символов. Алфавит = {<буквы>, <цифры>, <пустые символы>, <спец. символы>}

Синтаксис задаёт (содержит) правила (для человека) образования допустимых (правильных) для данного языка конструкций и правила образования программы из этих конструкций. Синтаксис описывается с использованием специальных искусственных языков – аналитических (БНФ) и графических (синтаксические диаграммы). Например, определение «арабские цифры» имеет вид:



Описание синтаксиса на графическом языке

или Арабские цифры::= 0 | 1 | 2 | 3 | 4 | … | 9

Описание синтаксиса на аналитическом языке (БНФ)

Семантика – правила (для компилятора) толкования конструкций языка (с точки зрения возможности вычисления осмысленного результата).

Замечание: в программе могут быть конструкции, правильные с точки зрения синтаксиса (человека, написавшего программу по правилам грамматики) и неправильные с точки зрения семантики (компилятора, толкующего то, что написал человек):

```
Var
 i : integer;
 r : real;
begin
 r := 1.25;
 i := r;
end.
```

С точки зрения семантики языка Паскаль (по правилам т.н. совместимости типов) недопустимо присваивание вещественного значения переменной целого типа (у них в памяти абсолютно разные представления).

Задаётся семантика, в отличии от синтаксиса, не формально (т.е. не на искусственном языке), а посредством неформальных описаний на естественном языке.

2. Алфавит

Включает в себя буквы, цифры, спец. символы и пустые символы.

Буквы – допускается использование букв латинского алфавита (строчных и прописных, причем регистр букв в программах на Паскале не различается). Буквы русского алфавита не допускается использовать в конструкциях языка - можно только в составе строковых констант, символьных констант и комментариев). К буквам в Паскале относится и символ подчеркивания, часто выполняющий при записи сложных (многословных) имен роль пробела (типа *day_of_week*).

Цифры – арабские (от 0 до 9) или шестнадцатиричные (от *a* до *f*, или от *A* до *F*), где $a_{16}=10_{10}$, …$f_{16}=15_{10}$..

Пустые символы (их коды – от 0 до 32) – они названы так потому, что не отображаются на экране ЭВМ, но производят некоторое действие. Так, символ с кодом 7 при попытке вывести его на экран вызывает кратковременный гудок динамика, а символ с кодом 32 – пробел (пустое место в тексте на бумаге или на экране дисплея). При подготовке текста, чтобы подчеркнуть наличие в определенном месте пробела, для его обозначения используют символ ␣.

Спец. символы (их 22) – это + - * / = < > ( ) [ ] { } . , : ; ^ @ # $ ‘. Сюда входят знаки препинания, скобки, знаки операций, Кроме того, некоторые пары символов, рассматриваются как отдельные символы: >=, <=, .. – знак диапазона, <> - знак неравенства, (* и *) – вместо фигурных скобок для обозначения границ комментария, (. и .) – вместо квадратных скобок.

3. Лексическая структура языка

Минимальная конструкция языка, имеющая смысл, называется **лексемой** (аналог в обычном языке – слово). Фразы (аналог в обычном языке – предложение) на языке Паскаль записываются с использованием лексем и разделителей (между лексемами).

Замечание: Правильнее говоря, разделители являются особым типом лексем вместе со знаками операций и ключевыми (служебными) словами. Эти особого вида лексемы заранее известны (задаются) компилятору (т.н. терминальные символы грамматики языка - знаки операций, ключевые слова, ...).

В качестве разделителей используются пустые и спец. символы. При этом между двумя соседними лексемами должен находиться хотя бы один разделитель (терминальный символ). Сами лексемы строятся из символов алфавита.

Пример:

- лексемы, - разделители (при этом у лексем разные типы: for, to, do –ключевые слова, i – идентификатор, 1 и 10 – константы).

Правило грамматики Паскаля для FOR имеет вид:

Анализ правильности отдельных слов (лексем) в программе выполняет программа- ***parser (сканер)*** (строит и передает синтаксическому анализатору таблицу кодов лексем, где коды лексем располагаются в прядке следования лексем в программе), а анализ правильности предложений (операторов) языка выполняет программа – ***scanner (синтаксический анализатор)*** (на основе таблицы кодов лексем и синтаксических правил проверяет правильность слов и их порядок в предложении - см. ЛР2).

В Паскале имеются следующие **классы лексем**: служебные слова, идентификаторы, метки константы, комментарии, директивы.

Служебные (зарезервированные) слова (begin, end, type, var ….)– за ними в языке закреплены определенные функции и они не должны использоваться в других целях. То есть нельзя, например, использовать служебные слова в качестве идентификаторов в программе:

```
Type:=10;
Begin:=a;
```

или

```
Program str;
...
begin
  str(...);
end.
```

сообщение компилятора «Ожидается точка»

Ошибка: здесь имя программы совпало с именем стандартной процедуры Паскаля STR.

Идентификаторы – используются в программах на Паскале для *обозначения имен* следующих конструкций: именованных констант, переменных, процедур, функций, меток, модулей, классов, объектов, программ, полей в записи. В Паскале ограничений на длину идентификатора не накладываются, но только первые 63 символа считаются значимыми. Идентификатор может начинаться только с буквы, за которой могут следовать (в любом порядке) буквы или цифры или знаки подчеркивания. Нельзя в идентификаторах использовать буквы русского алфавита, специальные символы и пробелы. В Паскале не различаются регистры букв (прописные или строчные). Идентификаторы могут быть простыми и составными. Составные идентификаторы

записываются с использованием символа ‘.’ , который разделяет одну часть сложного имени от другой, например,

aaa.bbb
ccc.bbb
ddd.bbb

Здесь 3 составных имени - aaa, ccc и ddd - это т.н. записи (из полей). В составном имени ccc.bbb то, что слева от точки указывает, какому конкретному объекту (записи) принадлежит поле bbb

Метки – существует два способа записи меток: 1) в виде целых чисел без знака в диапазоне от 0 до 9999. 2) в виде правильных идентификаторов (имен). Метки используются в программах для обозначения точек перехода и отделяются от операторов двоеточием:

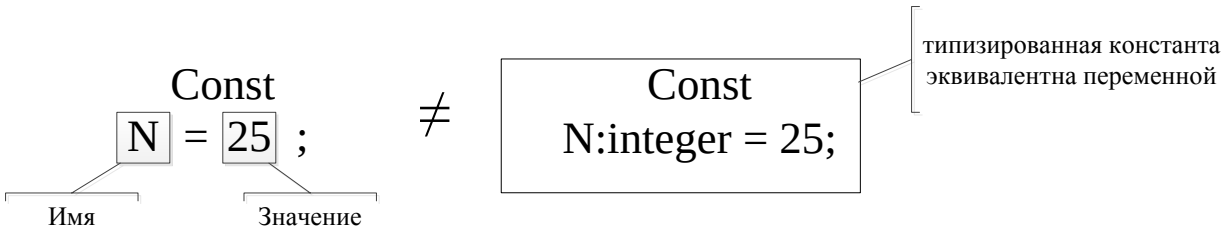
```
goto metka;   — должна быть описана в секции Label
…..           — метка в операторе безусловного перехода
…..           — оператор, на который происходит безусловный переход
metka: a:=a+1;
```

метка отмечает место перехода

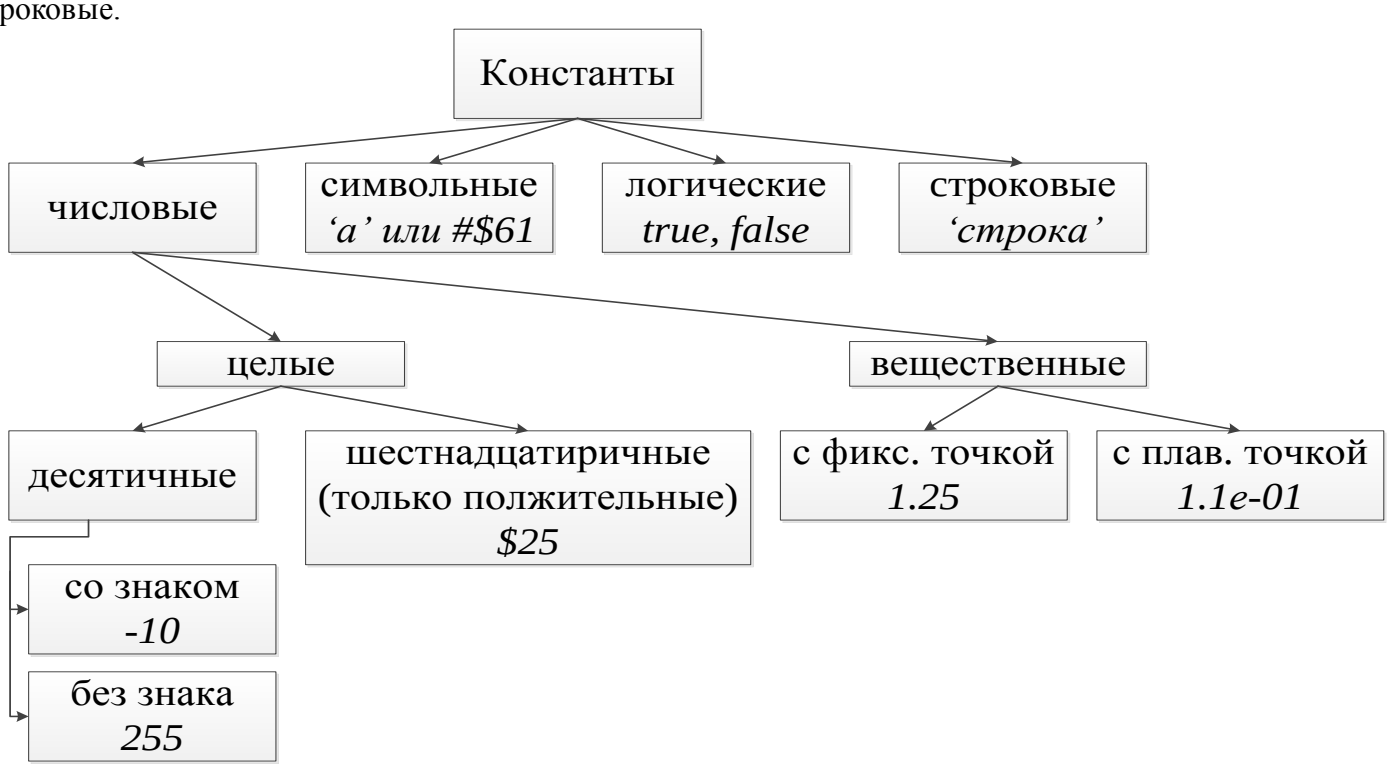
Константы – Простая константа используется для явного (непосредственного) представления ***значения*** (за каждой константой закрепляется одно значение) в программе. Существует два вида простых, т.е. нетипизированных, констант:

- без имени (литерал, строковое представление константы): c:= a + **2**;
- с именем: Const N=2.

Для повышения понятности в программе вместо значения константы лучше использовать осмысленное имя:



Константы делятся на числовые (целые и вещественные), символьные, логические и строковые.



1) **целые** – десятичные и шестнадцатиричные. Десятичные записываются с помощью арабских цифр и знака перед числом. Шестнадцатиричные – строятся с использованием шестнадцатиричных цифр, которым предшествует символ $, причем шестнадцатиричные константы м.б. только положительными, т.е. -$25 является правильной константой, а. $-25 является неправильной.
2) **вещественные** – имеют две формы записи: с фиксированной точкой и плавающей точкой. С фиксированной точкой – 1.25 . С плавающей точкой: 1.25E-05 (читается как 1.25 на 10 в степени минус 05).

3) **символьные** имеют две формы записи: 1) 'c' 2) #ЦЦ, где ЦЦ – код символа (в 10-чной или 16-ричной системе счисления). Соответствие между символом и его кодом представлено в кодовых таблицах. Для ПЭВМ обычно используется таблица ASCII (American Standard Code for Information Interchange). В этой таблице записаны 256 символов с порядковыми номерами 0..255 (цифры, большие и малые буквы, знаки операций, знаки препинания и др.). Порядковый номер символа в таблице ASCII - это код данного символа.

В кавычки заключаются изображения символов, которые можно ввести с клавиатуры. С помощью решетки и номера обычно обозначают неотображаемые символы, например,. #32 - символ с кодом 32 (пробел). Тот же самый символ можно обозначить с помощью шестнадцатиричного кода символа = #$20.

Замечание. В Паскале и строковые константы и символьные константы заключаются в одинарные кавычки.

4) **логические**: true (истина) и false (ложь).
5) **строковые** - представляют из себя последовательность символов (кроме символов с кодами 13_{10} и 10_{10} - это конец строки), заключенную слева и справа в кавычки.

В отличие от конструкции языка в строке допустимы символы русского алфавита.

Можно разрывать длинную строку и в разрыве ставить символ #:

' Строка 1' #7 ' Строка 2'

Комментарии: в правилах Паскаля комментарий начинается и заканчивается фигурными скобками:

{Комментарий}.

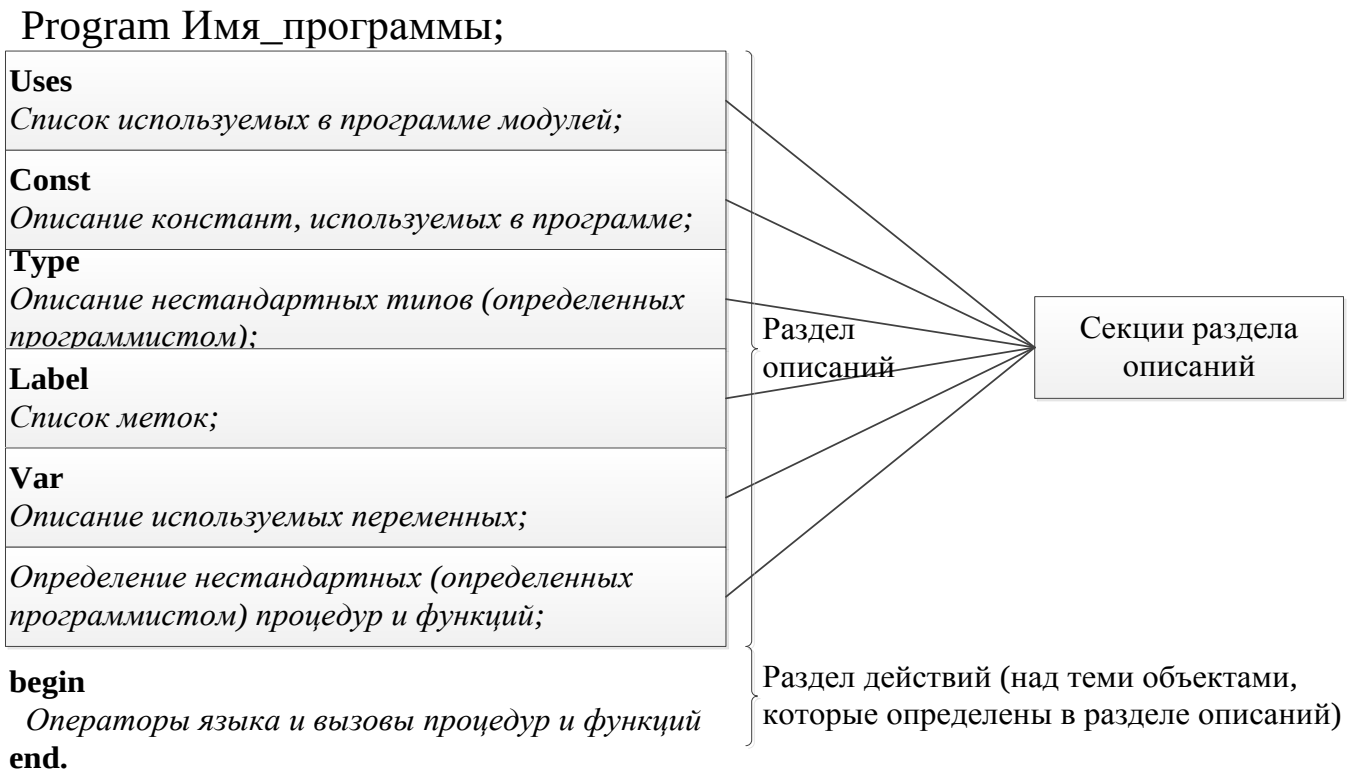
Вместо фигурных скобок можно использовать (* и *):

(* То же комментарий *).

В отличие от других языков Паскаль не допускает вложения комментариев с ограничителями одного типа друг в друга (с ограничителями разных типов - можно).

Директивы (компилятору) - на Паскале директивы записываются как особая форма комментария: {$I+} {$N+} В директиве за левой скобкой идет знак доллара, за ним текст самой директивы. Имеется три вида директив: 1) ключевые директивы 2) директивы параметров 3) директивы условной компиляции. Ключевые директивы содержат указание компилятору включить (I+) или выключить (I-) определенные свои возможности. Директивы параметров определяют значения различных параметров, используемых при компиляции (директива M определяет размер памяти, отводимой компилируемой программе). Директивы условной компиляции позволяют в зависимости от задания тех или иных условий компилировать или исключать из компиляции фрагменты программы (м.б. удобно для отладки программ).

4. Структура программной единицы



В отличие от стандартного Паскаля в Турбо Паскале (и Free Pascal) фразу "Program Имя;" можно не писать.

В случае использования модулей необходимо перечислить их после Uses через запятую. Если в программе используется Uses, оно должно быть сразу после слова Program, или, если такого нет, идти первым.

Секция описания меток имеет следующий вид:

Label

метка$_1$, метка$_2$, ..., метка$_n$;

На количество и порядок других секций (Const, Type, Var) в разделе описаний в Турбо Паскале ограничений не накладывается, т.е. может быть несколько секций описания переменных, описания типов, идущих в произвольном порядке.

Но есть 2 правила: 1) недопустимы повторные описания одного и того же имени в разных секциях; 2) все нестандартное, что в секции описаний используется**, должно быть выше по тексту программы (выше точки использования) определено.**

Например, в нижеприведенном случае имеется ошибка – N должно быть определено выше места использования:

```
Var
 v: 1..N;
Const
 N = 2;
```

Структуру, подобную приведенной структуре программы, имеют также и другие программные единицы, к которым также относятся **модули, процедуры и функции**. Отличие в записи модуля процедуры и функции (от программы), будет в основном в первых строчках:

unit имя;

procedure Имя (список формальных параметров);

function Имя (список формальных параметров);

и в последней строчке будет отличие:

программа заканчивается end .

модуль тоже заканчивается end .

процедура и функция заканчиваются end ;

5. Стиль записи программ на языке Паскаль

Чтобы программа легко читалась, ее текст следует располагать ступенчато с помощью пробелов (отступов) и пустых строк. При этом надо помнить два простых правила:

- конструкции языка (описания, операторы, блоки) одинакового уровня вложенности надо размещать с выравниванием влево по одной вертикальной линии.
- конструкции языка с более высоким уровнем вложенности (вложены в другие конструкторы) надо смещать вправо на 1-2 позиции относительно конструкций более низкого уровня вложенности.

Что такое ***уровень вложенности***? Если некоторая конструкция языка является частью другой (вложена ей внутрь), то ее уровень вложенности на 1 больше, чем у внешней.

Самой внешней конструкцией является программа, текст которой начинается с заголовка программы (со слова Program). Заголовки секций, такие как Uses, Label, Type, Const, Var, а также заголовки процедур и функций в разделе описаний, слова begin и end, отмечающие основной раздел действий программы, располагаются (смещаются) вправо на 1-2 позиции относительно позиции, с которой начинается слово Program. В свою очередь содержимое секций Uses, Label, Type, Const, Var, тела процедур и функций в разделе описаний, выполняемые действия между словами begin и end располагаются со смещением вправо, относительно позиции, с которой начинается заголовок соответствующей секции.

Внутри секций Type и Var надо по возможности так разместить текст, чтобы значки «:» и «=» располагались на одной вертикали. Описание каждой переменной и каждого типа должно занимать отдельную строку и заканчиваться комментарием.

Например:

```
Program P1;
  uses crt;
  type
    t1= array[1..20] of char;          {Комментарий}
    t2= string[15];                    {Комментарий}
    t3= record a:byte; b: char; end;   {Комментарий}
  var
    v1:t1;                             {Комментарий}
    v2:t2;                             {Комментарий}
    v3:t3;                             {Комментарий}
  begin
    a:=2;.....
  end.
```

6. Типы данных в Паскале

6.1 Понятие данных (см. то, что было до Паскаля)

Данные – статические модели объектов предметной области (только их свойства). Виды данных – исходные, промежуточные и выходные. Данные – константы и данные – переменные.

6.2 Концепция типа данных (см. то, что было до Паскаля)

Тип данных настраивает исполнителя алгоритма на работу с данными разного типа. Это достигается за счет тог, что понятие типа данных определяет следующие 3 элемента:

1) список возможных значений данных конкретного типа;
2) внутреннее представление данных этого типа в памяти ЭВМ;
3) список возможных операций (действий) над данными конкретного типа.

Настройка исполнителя на работу с данными обеспечивается указанием для каждого из данных его имени и типа. Имя служит синонимом места расположения данных в памяти, а тип данных указывает компилятору как обрабатывать то, что лежит по конкретному адресу.

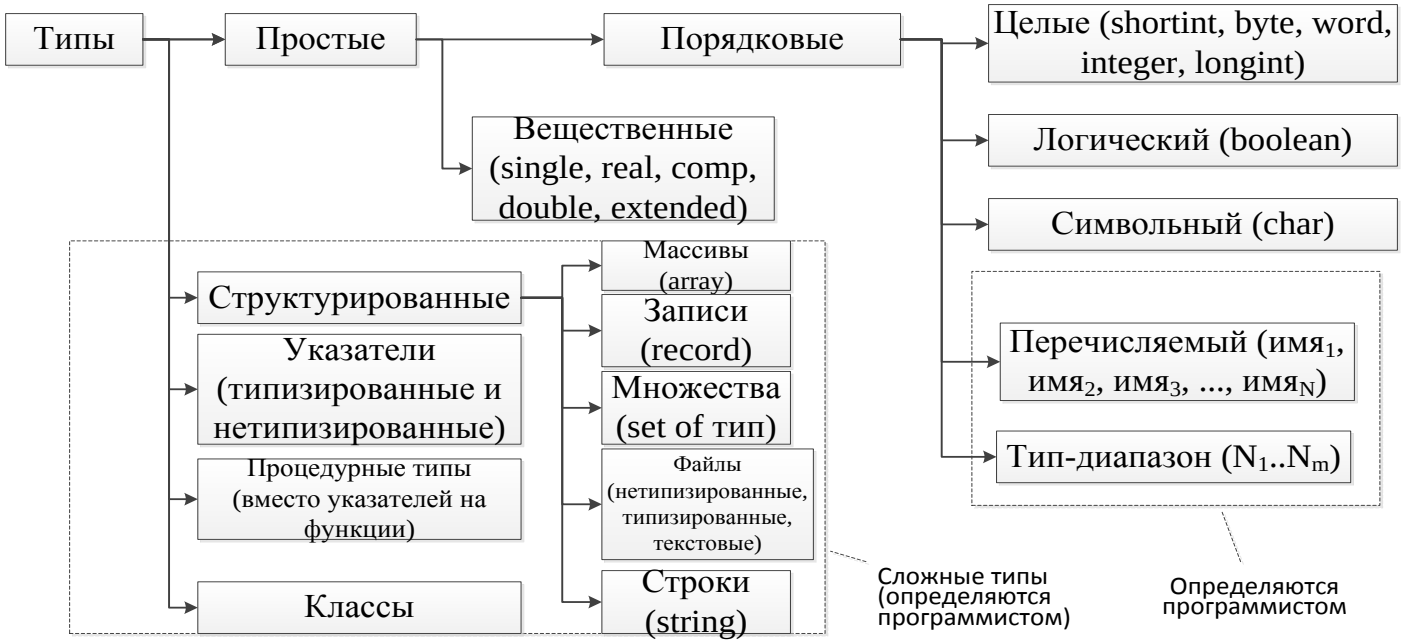
Типы могут быть простыми (скалярными) и сложными (структурированными), а также стандартными (в том числе простыми или определенными(доопределенными) программистом) и нестандартными.

Работа с данными стандартных типов осуществляется без каких-либо дополнительных описаний (кроме указания имени типа) с помощью операторов языка, а для работы с *нестандартными* типами надо: 1) определить их тип в секции type 2) написать подпрограммы, реализующие действия с этими данными, 3) для выполнения действий надо в нужный момент вызывать соответствующие подпрограммы, 4) подпрограммы для размещения данных в памяти.

Необходимо чётко различать структуры данных и структуры хранения (структуры представления данных в памяти ЭВМ).

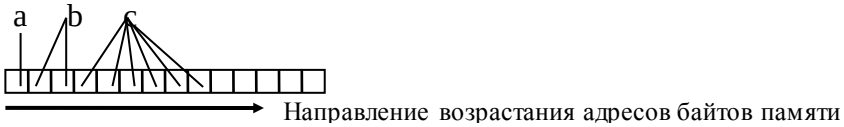
6.3 Классификация типов данных в Турбо Паскале

На Паскале структура типов имеет следующий вид:



В программе для каждого элемента данных должны быть известны имя и тип. В соответствии с этим определением компилятор выделяет память для каждого имени в соответствии с типом. Причем, память выделяется в том порядке, в котором переменные описаны в разделе описания. Например:

```
Var
  a: byte;     {занимает 1 байт}
  b: integer;  {занимает 2 байт}
  c: Real;     {занимает 6 байт}
```



Стандартные типы данных, как уже говорилось, бывают простые и определенные программистом. С одной стороны **типы, определенные программистом,** обычно записываются в секции описания типов в следующей форме:

```
Type
  имя = определение типа;
```

Например:

```
Type
  M = record
        a, b: byte:
      end;
```

С другой стороны можно было бы, как говорят, **сконструировать (объявить) тип переменной прямо в секции описания переменных** по следующему шаблону:

```
Var
  имя1: описание типа;
```

Например:

```
Var
  v1: array [1..10] of char;
```

Переменная v1 относится к тому типу, который был определен при объявлении переменной, но с v1 не будет совместим никакой другой тип.

Константы (простые) объявляются в секции описания констант следующим образом:

```
Const
  N = 2;              целого типа
  C = 'c';        символьного типа
  D = 1.28;       вещественный тип
  E = $AD;        целого типа
  F = 'abcd';     строкового типа
```

В отличие от описания переменных, **при описании простых констант тип явно не указывается**, но он легко угадывается, исходя из формы записи констант (для простых констант).

В Паскале есть так называемые **типизированные константы**, для которых обязательно указывается тип. Форма их записи следующая:

```
Const
  N: byte = 2;
  C: char = 'c';
```

Хотя типизированные константы описываются в секции описания констант, они не являются на самом деле константами: значение типизированных констант, в отличие от простых констант, вы можете менять в программе сколько угодно раз. По своей сути типизированные константы - это **переменные с заданные начальными значениями,** причем начальные значения присваиваются еще на стадии компиляции.

Обычные константы используются для того, чтобы сделать программу более читаемой и легче модифицируемой. Это достигается за счет того, что в программе **вместо значений констант используются их осмысленные имена.**