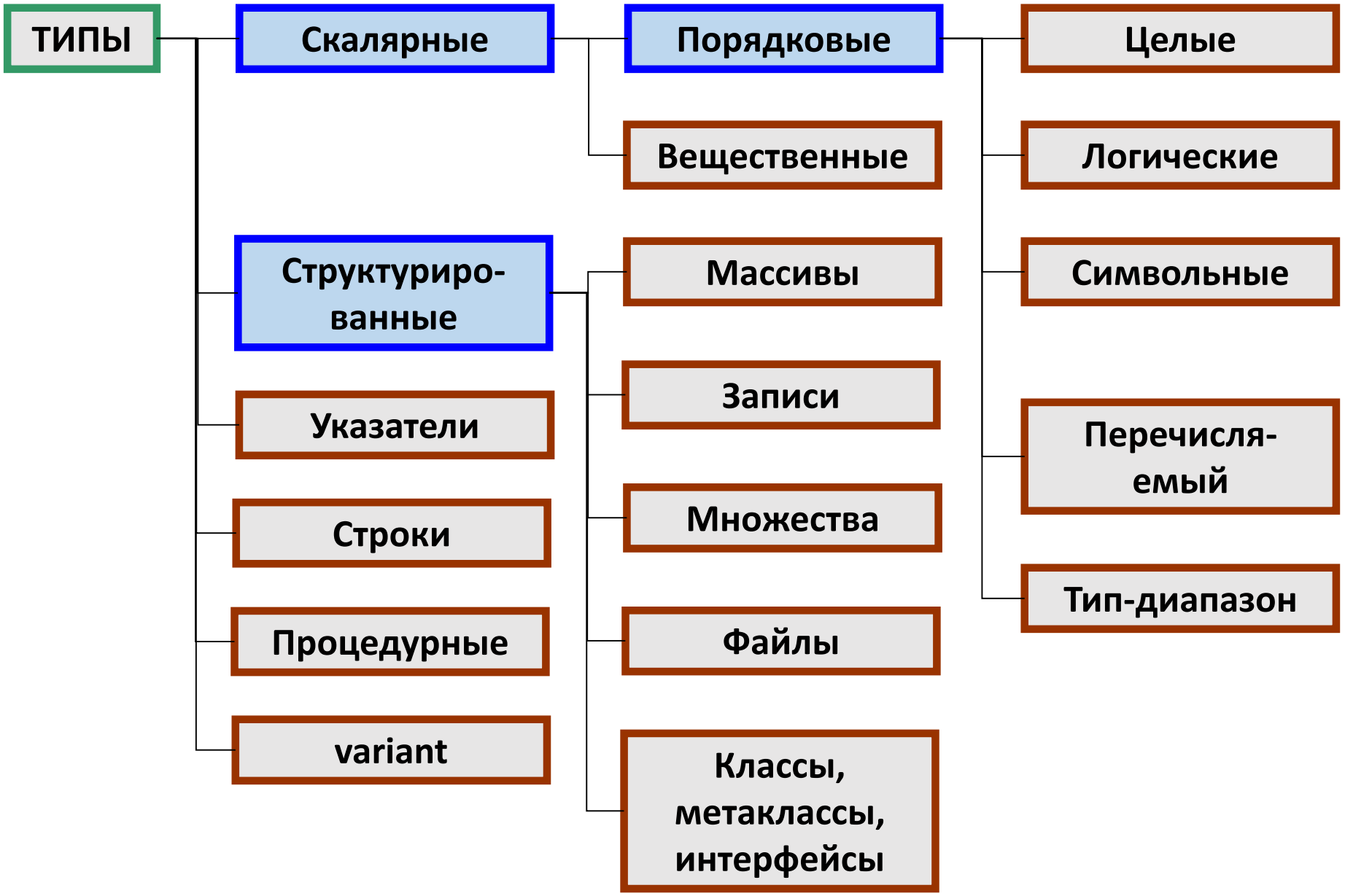
Глава 7. СТРУКТУРИРОВАННЫЕ ТИПЫ ДАННЫХ

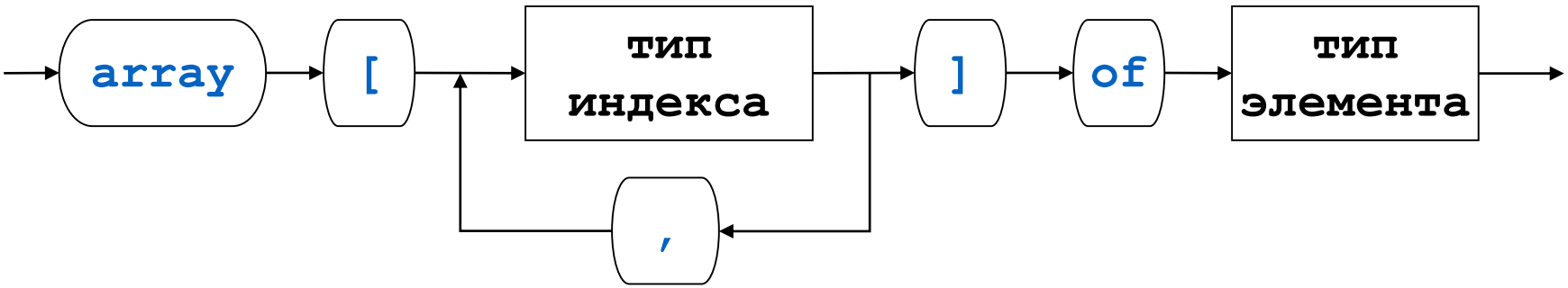
- Организация типов данных
- Массивы
- Записи, оператор присоединения
- Множества, операции над множествами
- Строки, стандартные процедуры и функции, работающие со строками
- Совместимость типов
- Явное и неявное преобразование типов

ТИПЫ ДАННЫХ ЯЗЫКА DELPHI PASCAL



МАССИВЫ

Массив – упорядоченная совокупность однотипных данных.



<тип индекса> – любой порядковый тип, размер которого не превышает 2 Гбайт

```
Type
   Vector = array [1..3] of Real;
                    {тип индекса – тип-диапазон}
Var
   R,V : Vector;
```

<u>или</u>

```
Var
   R,V : array [1..3] of Real;
   T : array [1..5] of Byte = (0,1,2,3,4);
```

МАССИВЫ

<тип элемента> массива – любой допустимый в Delphi Pascal тип кроме файла (в том числе и другой массив).

Многомерные массивы:

```
Type
   Matrix = array [0..5] of array [-2..2] of
            array [Char] of Real;
```

или

```
Type
   Matrix = array [0..5,-2..2,Char] of Real;
```

Доступ к элементам массива:

```
Var
   m : Matrix;
   N : Byte;
Begin
... m[1,0,'d'] := 5.2;
   N := 2;
   m[N-1][0]['n'] := 6.3; ...
End.
```

МАССИВЫ

Присваивание массивов:

```
Var
   a,b : array [1..5] of Real;
Begin
   ...
   a := b;
   ...
End.
```

При большом числе элементов массива возможно ограничение, связанное с максимальным объемом памяти, который отводится под глобальную переменную – 2 Гбайт.

```
Var
  Mas3D = array [1..1000,1..1000,1..1000] of Integer;
                                    {4 000 000 000 байт}
```

При компиляции в режиме, задаваемом ключом {$R+}, будет проверяться принадлежность значения индекса объявленному диапазону, и в случае нарушения границ будет выдано сообщение об ошибке.

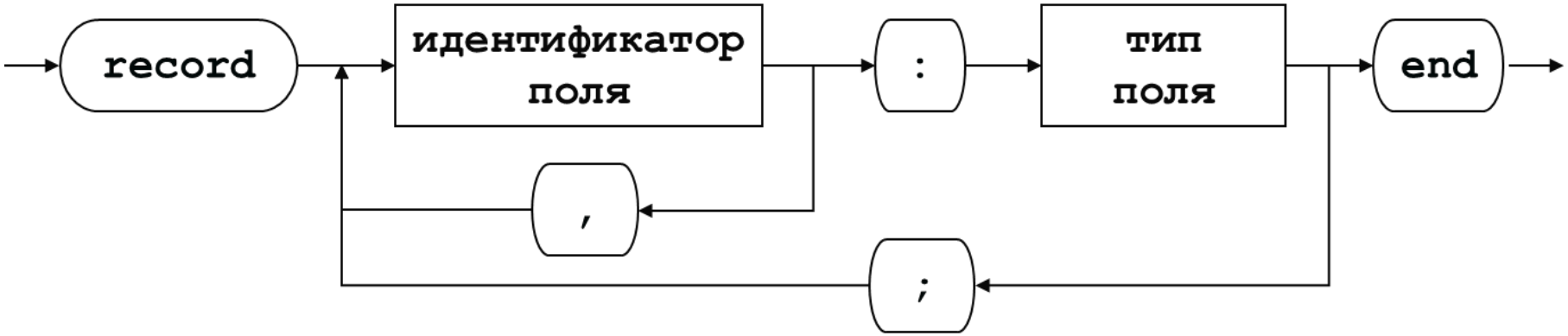
ДИНАМИЧЕСКИЕ МАССИВЫ

```
Var
   a : array of Real;
   N,i : Byte;
Begin
   N := 5;                          {a → 0}
   SetLength(a,N);                  {a → (0,0,0,0,0)}
   for i := 0 to N-1 do
      a[i] := i*2;                  {a → (0,2,4,6,8)}
   SetLength(a,N-2);                {a → (0,2,4)}
   SetLength(a,N+1);                {a → (0,2,4,0,0,0)}
End.
```

Переменная типа динамического массива – это указатель (значение – адрес).

ЗАПИСИ

Запись – структура данных, состоящая из фиксированного числа разнотипных компонент, называемых **полями записи**.



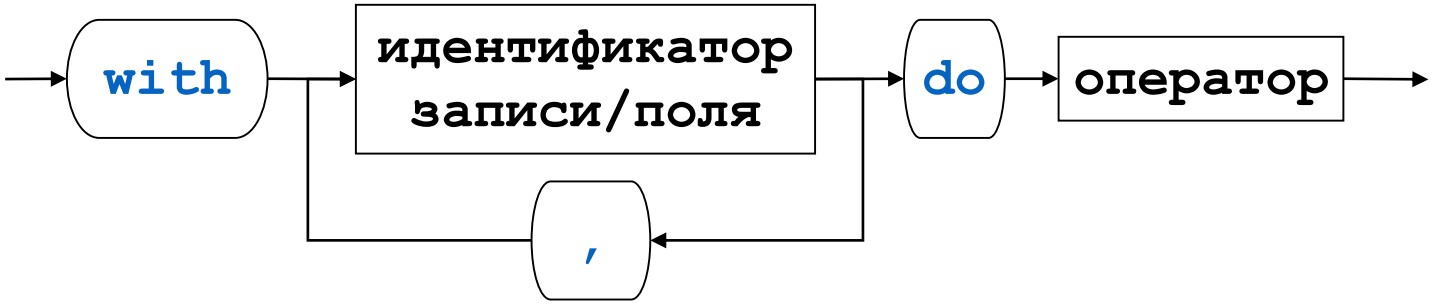

```
Type
   Data = record
          X,Y : Integer;
          Z   : Char
          end;
Var    D1,D2 : Data;
Begin
...    D1.X := 10;
...    D2.Z := 'n';
...    D2 := D1;          {присваивание записей}
End.
```

ЗАПИСИ

Полем записи может быть другая запись (вложенные структуры):

```
Var
   D : record
         X : Integer;
         R : record
               RX : Integer;
               RZ : Char
               end
         end;
Begin
   ...
   D.R.RX := 2;
   ...
End.
```

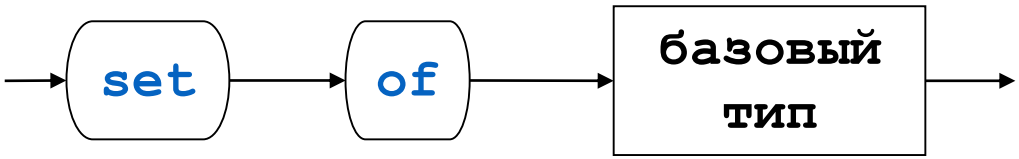
ОПЕРАТОР ПРИСОЕДИНЕНИЯ




```
with D do
   begin
      R.RX := 2;          {D.R.RX}
      with R do
         RZ := 'f';       {D.R.RZ}
   end;
```

МНОЖЕСТВА

Множество – это структурированный тип данных, представляющий собой неупорядоченную совокупность неповторяющихся элементов. Количество элементов, входящих во множество, может меняться в пределах от 0 до 256 (множество может быть пустым).



<базовый тип> – любой порядковый тип с числом элементов, не превышающим 256 (с кодами в диапазоне 0..255).

```
Type
   TypeSet1 = set of Char;
   TypeSet2 = set of 0..9;

   VideoType = (Hercules,CGA,EGA,VGA,SVGA);
   TypeSet3 = set of VideoType;
```

МНОЖЕСТВА

Значением переменной множественного типа является множество, которое определяется с помощью конструктора множества, представляющего собой перечисление элементов базового типа в квадратных скобках.

```
Var
   Set1, Set2 : set of Byte;
   Set3 : set of 'a'..'f';
   X : Integer;
Begin
   ...
   Set1 := [3..10,12];
   Set3 := ['a','d'];
   X := 5;
   Set1 := [X+2,4,9];
   Set3 := [];
   Set2 := [9,7,9,4];
   ...
End.
```

ОПЕРАЦИИ НАД МНОЖЕСТВАМИ

Set1 = [0..3,6] Set2 = [3..9]

* – **пересечение множеств**, результат содержит элементы, общие для обоих множеств (Set1 * Set2 = [3,6]);

+ – **объединение множеств**, результат содержит элементы первого множества, дополненные недостающими элементами второго (Set1 + Set2 = [0..9]);

- – **разность множеств**, результат содержит элементы первого множества, которые не принадлежат второму (Set1 - Set2 = [0,1,2]);

= – **проверка эквивалентности**, возвращает True, если оба множества эквивалентны;

<> – **проверка неэквивалентности**, возвращает True, если множества неэквивалентны;

<= – **проверка вхождения**, возвращает True, если первое множество включено во второе;

>= – **проверка вхождения**, возвращает True, если второе множество включено в первое;

in – **проверка принадлежности** (E in S), возвращает True, если значение E входит в множество S и принадлежит базовому типу этого множества (3 in Set1 = True, 2*2 in Set1 = False).

МНОЖЕСТВА

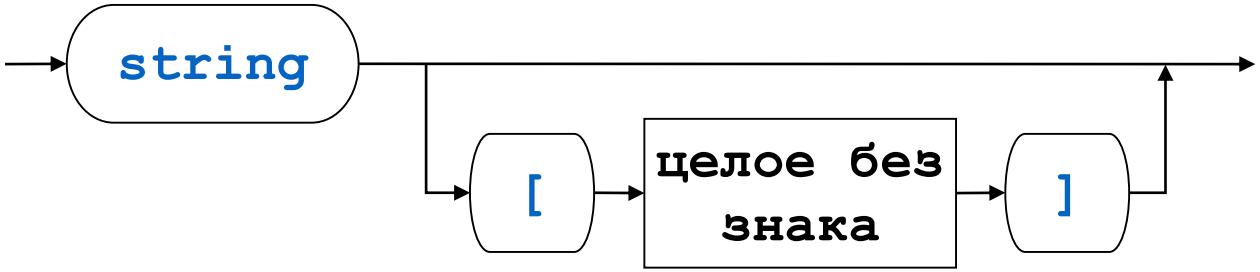
Процедуры, параметром которых является множество:

INCLUDE (S,I) – включает новый элемент I в множество S (включаемый элемент должен принадлежать базовому типу множества S).
EXCLUDE (S,I) – исключает элемент I из множества S.

```
Var
   Set1 : set of 1..10;
   I : Byte;
Begin
   ...
   Set1 := [2,3,4];
   Include(Set1,2*3);
   for I := 1 to 10 do
      if I in Set1 then Writeln(I);
   Writeln(SizeOf(Set1));
   Set1 := Set1 + [12];
   ...
End.
```

СТРОКИ

Тип String используются для обработки текстов и трактуется как цепочка символов. **Строка** – динамический (переменной длины) массив, состоящий из символов. Максимально возможная длина строки 255 символов. Тип объявляется как String[N], где N - максимальное число символов в строке.




```
Var
   S32 : String[32];
   S255 : String[255]; {String}
Begin
   S32 := 'Это строка';
   {ORD(S32[0]) = 10}

   S32[3] := 'a';
   S32 := S32 + '!!!';
End.
```

ПРОЦЕДУРЫ И ФУНКЦИИ ДЛЯ РАБОТЫ СО СТРОКАМИ

LENGTH (S : String) : Integer – возвращает длину строки (функция);

CONCAT (S1, S2,…,Sn : String) : String – возвращает конкатенацию (слияние) строк S1,…,Sn (функция);

COPY (S : String; Start, Len : Integer) : String – возвращает подстроку длиной Len, начинающуюся с позиции Start строки S (функция);

DELETE (Var S : String; Start, Len : Integer) – удаляет из S подстроку длиной Len, начинающуюся с позиции Start строки S (процедура);

INSERT (SubS : String; Var S : String; Start : Integer) – вставляет в S подстроку SubS, начиная с позиции Start (процедура);

POS (SubS, S : String) : Integer – ищет вхождение подстроки SubS в строке S и возвращает номер первого символа SubS в S или 0, если SubS нет в S (функция);

ПРОЦЕДУРЫ ПРЕОБРАЗОВАНИЯ

STR (X; Var S : String) – преобразует числовое значение X в строковое S, возможно задание формата для X (Str(X:F:n,S), где F – общая ширина поля, n – количество символов в дробной части для вещественных чисел);

VAL (S : String; Var X; Var Code : Integer) – преобразует строковое значение S (строку цифр) в значение числовой переменной (X – целое или вещественное, параметр Code содержит ноль, если преобразование прошло успешно, в противном случае он содержит номер позиции в строке, где обнаружен ошибочный символ, при этом X не меняется).

Операции отношения (=, <>, >, <, >=, <=):
Результат - логическая константа (*True, False).* Сравнение строк выполняется последовательно слева направо с учетом внутренней кодировки символов до первого несовпадающего символа.

'aBcd' = 'ab' (результат *False*);
'aBcd' > 'ab' (результат *False*);
'aBcd' < 'ab' (результат *True*).

Строковые типы Delphi Pascal: ShortString, ANSIString, String, WideString, PChar.

СОВМЕСТИМОСТЬ ТИПОВ

Delphi Pascal требует соблюдения правил совместимости типов в ряде случаев: при использовании оператора присваивания, при выполнении операций отношения, при подстановке переменных или значений в вызовы процедур и функций и т.д.

Два типа совместимы, если они тождественны (идентичны). Типы считаются **тождественными**, если:

1. Они описаны вместе, либо одним и тем же идентификатором типа:

```
Type
      T1 = Boolean;
      T2 = Boolean;
      T3,T4 = array [1..2] of Real;
```

2. Типы описаны как эквивалентные

```
Type
      T1 = array [1..2] of Real;
      T2 = T1;
      T3 = T2;
```

СОВМЕСТИМОСТЬ ТИПОВ

Типы совместимы (гарантирует работу операций отношения, подстановку значений или переменных в параметры функций и процедур), если:

- оба типа являются тождественными;
- оба типа являются вещественными;
- оба типа являются целыми;
- один тип является поддиапзоном другого;
- оба типа являются поддиапазонами одного и того же базового типа;
- оба типа являются множествами, составленными из одного и того же базового типа;
- один тип является строковым, а другой символьным или строковым;
- один тип является указателем, а другой указателем или ссылкой;
- оба типа являются классами, ссылками на класс или интерфейсами, причем один тип унаследован от другого;
- ...

СОВМЕСТИМОСТЬ ПО ПРИСВАИВАНИЮ

Переменной *X* (тип *T1*) может быть присвоено значение *Y* (тип *T2*) (т.е., X := Y) если:

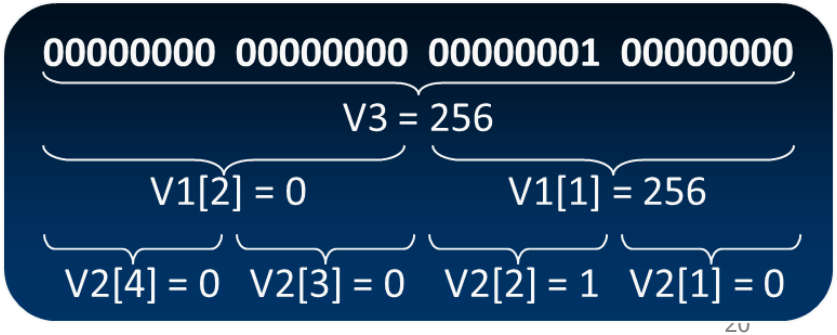
- *T1* и *T2* – тождественные типы, и не один не является файловым типом (или структурным типом, содержащим компонент с файловым типом);
- *T1* и *T2* – совместимые типы (в смысле, рассмотренном ранее), относящиеся к порядковым, и значения типа *T2* попадают в диапазон возможных значений *T1*;
- *T1* и *T2* – вещественные типы и значения типа *T2* попадают в диапазон возможных значений *T1*;
- *T1* – вещественный тип, *T2* – целочисленный тип;
- *T1* и *T2* – строковые типы;
- *T1* – строковый тип, *T2* – символьный тип;
- *T1* и *T2* совместимые множества и все члены значения множества типа *T2* попадают в диапазон возможных значений *T1*;
- *T1* и *T2* совместимые адресные типы;
- *T1* и *T2* оба являются классами, ссылками на классы или интерфейсами, при этом *T2* унаследован от *T1*;
- ...

ЯВНОЕ ПРЕОБРАЗОВАНИЕ ТИПОВ

1. Может быть реализовано посредством использования специальных функций:
TRUNC(x) – преобразует значение вещественного типа в значение целого типа, отбрасывая дробную часть;
ROUND(x) – преобразует значение вещественного типа в значение целого типа, округляя его до ближайшего целого;
ORD(x) – преобразует значение порядкового типа в его номер;
CHR(x) – преобразует код символа в сам символ и др.

2. При приведении типа переменной используется функция, которая совпадает с именем типа, к которому должна быть приведена переменная. Необходимо равенство длин внутреннего представления в памяти обоих типов.

```
Type  M2Word = array [1..2] of Word;
      M4Byte = array [1..4] of Byte;
Var   V1 : M2Word;      V2 : M4Byte;
      V3 : LongInt;     V4 : SmallInt;
Begin
      V3 := 256;
      V1 := M2Word(V3);
      V2 := M4Byte(V3);
V4 := SmallInt(V1[1]); End.
```



НЕЯВНОЕ ПРЕОБРАЗОВАНИЕ ТИПОВ

1. При несовпадении типов правой и левой частей оператора присваивания для совместимых типов автоматически выполняется неявное преобразование результата выражения к типу переменной, указанной в правой части (например, деление целых чисел $\rightarrow$ Extended).

2. Происходит, если одна и та же область памяти попеременно трактуется как содержащая данные то одного, то другого типа (совмещение в памяти данных разного типа).

Совмещение данных в памяти, в частности, возможно при размещении данных разного типа по одному и тому же абсолютному адресу. Для размещения переменной по нужному абсолютному адресу она описывается с последующей стандартной директивой Absolute, за которой помещается имя ранее определенной переменной.

```
Var
      x : Real;
      y : array [1..2] of Integer absolute x;
```