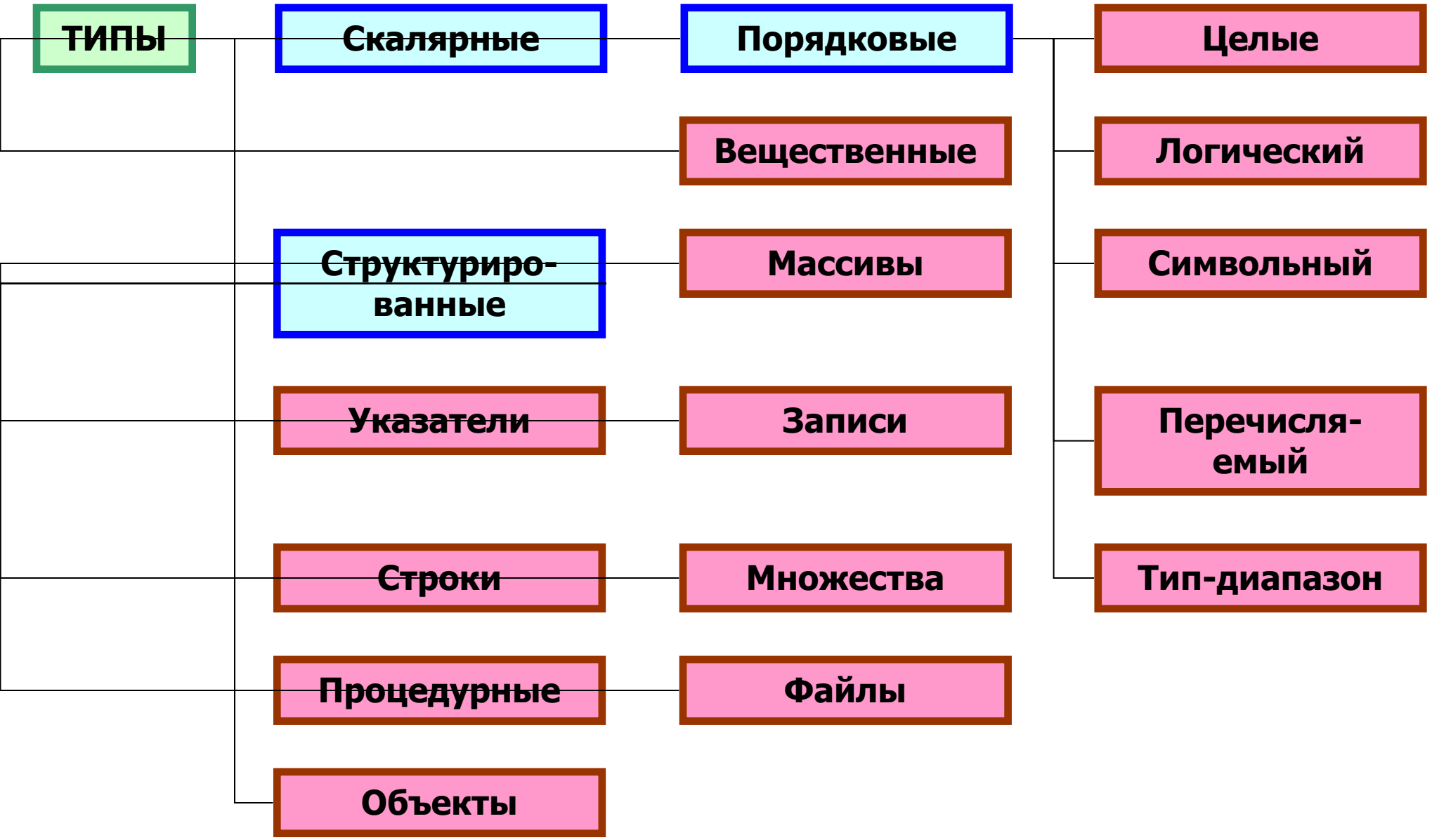
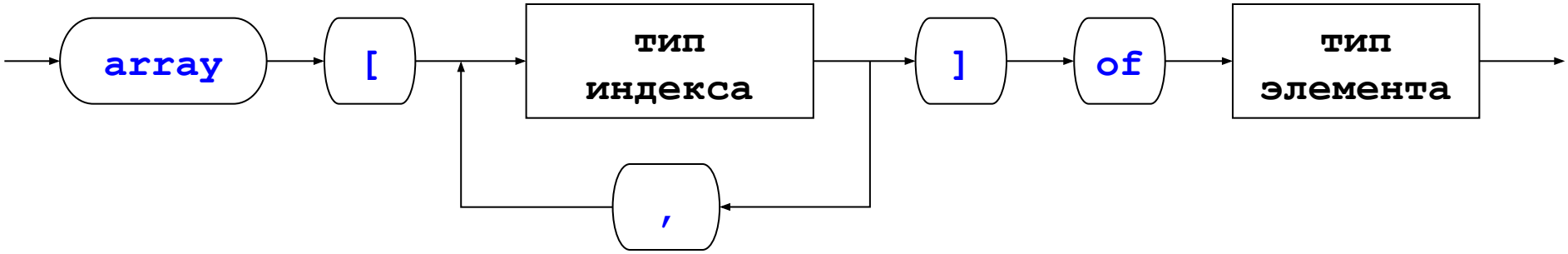
Глава 7. СТРУКТУРИРОВАННЫЕ ТИПЫ ДАННЫХ

- **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
Скалярные
Порядковые
Целые
Логический
Символьный
Перечисля-емый
Тип-диапазон
Вещественные
Структуриро-ванные
Массивы
Записи
Множества
Файлы
Указатели
Строки
Процедурные
Объекты

Массивы

Массив - упорядоченная совокупность однотипных данных.



<тип индекса> – любой порядковый тип кроме *LongInt* и типов-диапазонов с базовым типом *LongInt*.

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

или

```
Var
   R,V : array [1..3] of Real;
```

<тип элемента> массива – любой допустимый в Turbo 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.
```

Присваивание массивов:

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

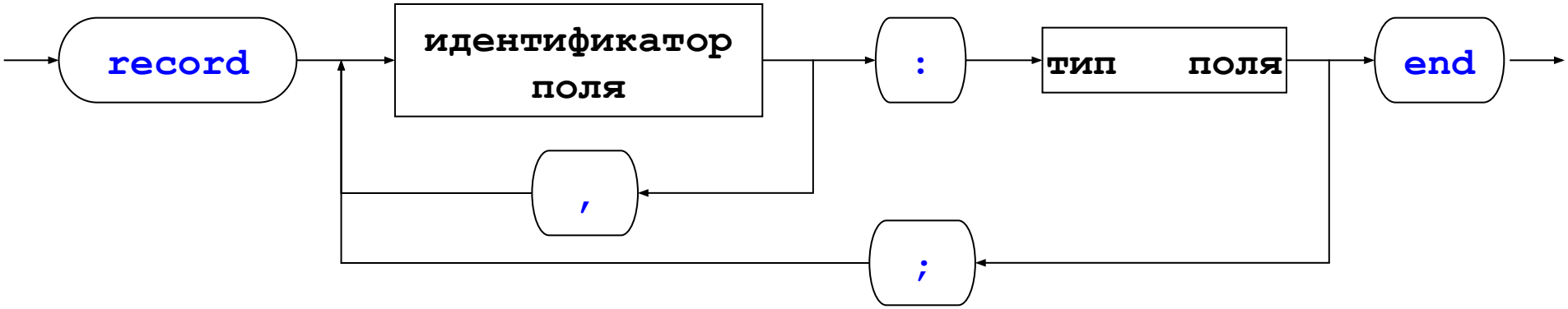
При большом числе элементов массива наступают ограничения, связанные с максимальным объемом памяти, отводимой под глобальные переменные – *сегмент данных* объемом 64 К.

```
Var
   Dim100x200 = array [1..100,1..200] of Real
{100x200x6 байт = 120 000 байт}
```

При компиляции в режиме, задаваемым ключом {$R+}, будет проверяться принадлежность значения индекса объявленному диапазону, и в случае нарушения границ будет будет выдано сообщение об ошибке (*Range check Error*).

Записи

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

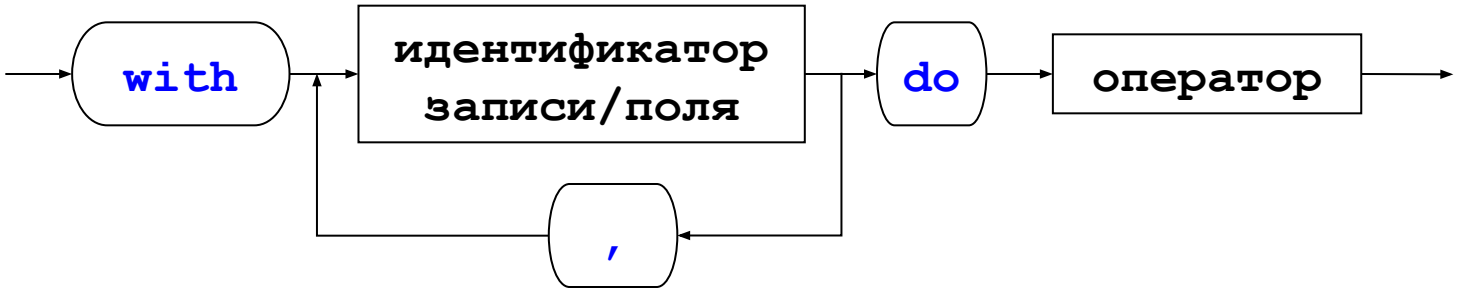



```
Type
   Data = record
        X,Y : Integer;
        Z   : Char
        end;
VarD1,D2 : Data;
Begin
...    D1.X := 10;
...D2.Z := 'n';
...    D2 := D1;    {присваивание записей}
End.
```

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.
```

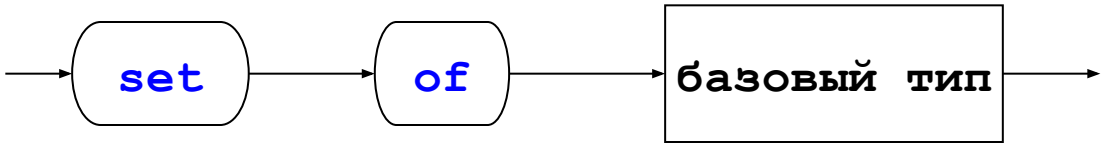
Оператор присоединения:




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

Множества

Множество – это структурированный тип данных, представляющий собой неупорядоченную совокупность неповторяющихся элементов. Количество элементов, входящих во множество, в Pascalabc не оговаривается, множество может быть и *пустым*.



<базовый тип> – любой порядковый тип .

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

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

Значением переменной множественного типа является множество, которое определяется с помощью *конструктора множества*, представляющего собой перечисление элементов базового типа через запятую 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).

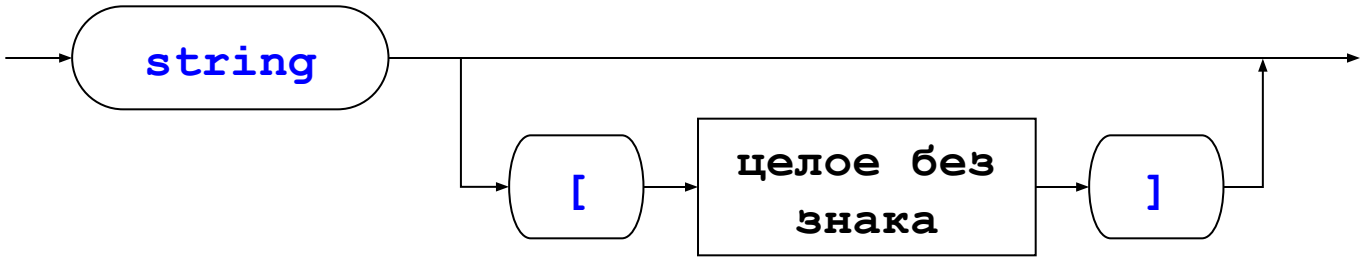
Процедуры, параметром которых является множество:

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 Pascalabc.net не оговаривается . Тип объявляется как string или *String[N],* где N - максимальное число символов в строке.




```
Var
      s:=string;
   S32 : String[32];
   S255 : String[255]; {String}
Begin
      s:= ‘dfhtgerj’;
   S32 := 'Это строка';
   S32[3] := 'a';
   S32 := S32 + '!!!';
End.
```

Процедуры и функции для работы со строками:

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

CONCAT (S1, S2,…,Sn) : 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 (Var S, SubS : String; Start : Integer) – вставляет в S подстроку SubS, начиная с позиции Start (процедура);

POS (SubS, S : String) : Byte – 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*).

Turbo 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;
```

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.

Совместимость по присваиванию.

Переменной *X* (тип *Type1*) может быть присвоено значение *Y* (тип *Type2*) (X := Y) если:

- *Type1* и *Type2* – тождественные типы, и не один не является файловым типом (или структурным типом, содержащим компонент с файловым типом);
- *Type1* и *Type2* – совместимые типы (в смысле, рассмотренном ранее), относящиеся к порядковым, и значения типа *Type2* попадают в диапазон возможных значений *Type1*;
- *Type1* и *Type2* – вещественные типы и значения типа *Type2* попадают в диапазон возможных значений *Type1*;
- *Type1* – вещественный тип, *Type2* – целочисленный тип;
- *Type1* и *Type2* – строковые типы;
- *Type1* – строковый тип, *Type2* – символьный тип;
- *Type1* и *Type2* совместимые множества и все члены значения множества типа *Type2* попадают в диапазон возможных значений *Type1*;
- *Type1* и *Type2* совместимые адресные типы;

(Тип объекта *Type2* совместим по присваиванию с типом объекта *Type1*, если *Type2* находится в области типа объекта *Type1*. Тип ссылки *Ptr2*, указывающий на тип объекта *Type2*, совместим по присваиванию с типом ссылки *Ptr1*, указывающим на тип объекта *Type1*, если *Type2* находится в области типа объекта *Type1*).

Явное преобразование типов.

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.

В *операции приведения типа* используется функция преобразования, которая совпадает с именем типа, к которому должна быть приведена переменная. При приведении типов переменных необходима их совместимость в машинном представлении.

```
Type  M2Word = array[1..2] of Word;
      M4Byte = array[1..4] of Byte;
Var   V1 : M2Word;V2 : M4Byte;
   V3 : LongInt;  V4 : Integer;
Begin
   V3 := 100;       V1 := M2Word(V3);
   V2 := M4Byte(V3);  V4 := Integer(V1[1]); End.
```

Неявное преобразование типов:

- реализуется в числовых выражениях, составленных из вещественных и целочисленных переменных, последние автоматически преобразуются к вещественному типу, и все выражение в целом приобретает вещественный тип;
- 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 либо абсолютный адрес, либо имя ранее определенной переменной.

```
Var
   w : LongInt absolute $3DA0:$0055;
   x : Real;
   y : array [1..3] of Integer absolute x;
```