Лекция 17. Множества

Множества – это один из структурированных типов в Delphi.
Раньше мы рассматривали еще два структурированных типа – массивы и записи

Напоминаю:

Массив – совокупность **однородных** элементов, хранящихся в **смежных** ячейках памяти, имеющих общее **имя** и отличающихся по **индексам**.

Запись – совокупность **не**однородных элементов, хранящихся в **смежных** ячейках памяти, имеющих общее **имя** и отличающихся по **именам полей**.

Тип	**Массив**	**Запись**	**Множество**
Тип элементов	**Однотипные** элементы	Могут быть **разных** типов	**Однотипные не**повторяющиеся элементы **простых порядковых** типов с диапазонами значений (Ord) не более чем от 0 до 255 (byte, char, логический, перечисляемый, диапазон)
Имя	Общее для всех элементов	Общее для всех элементов	Общее для всех элементов
Доступ/Обращение к элементам	По **индексу** элемента Имя_массива[индекс]	По имени **поля** Имя_записи.Имя_поля	Нет (есть только **проверка его наличия**)
Память	Смежные ячейки памяти	Смежные ячейки памяти, но (по умолчанию) с выравниванием по 4 или 8 байт	Смежные ячейки памяти – один бит на 1 элемент + округление в большую сторону до целых байтов (по 8 бит) Но не более 256 элементов! (32 байта) Наибольшая эффективность при 4 байтах (32 значения-бита)

Таким образом,
Множество – совокупность не более чем 256 **не**повторяющихся **однородных простых** элементов **порядковых** типов, хранящихся в **смежных БИТАХ** памяти, имеющих общее **имя**.

Множество

(по этой схеме рассматривали остальные типы см **Types1.doc**)

1) (Множество –) **Пользовательский** тип (, а значит, как и для массива и записи, часто перед использованием необходимо определить тип в разделе type)

Например, для использования переменной типа запись в качестве параметра процедуры (функции) – определение своего типа обязательно (как и для статического массива)

2) Определение типа и описание переменных

Type //в разделе type с ключевым словом **set**
TByteSet = **set** of byte; // потенциально для хранения чисел от 0 до 255
TCharSet = **set** of char; // потенциально для хранения символов от #0 до #255 (по кодам)
TAlf = **set** of 'a'..'z';
TChislo = **set** of 200..255;

Если есть перечисляемый пользовательский тип
TDays = (sun, mon, tue, wed, thu, fri, sat);
то
TPerSet = set of **TDays**;

Var // теперь опишем переменные
set1, set2: TByteSet;
set3: set of 0..4; // в некоторых случаях можно и без определения типа
set4: TAlf;

3) Ввод значений

Значение каждого элемента добавляется в множество отдельно

```
set1:=[]; // пустое множество
Writeln('Введите кол-во элементов:');
Write('n=?'); readln(n);
For i:=1 to n do
begin
  Write('элемент=?'); readln(k);
  Include(set1, k);  // или с помощью объединения множеств set1:=set1+[k];
end;
```

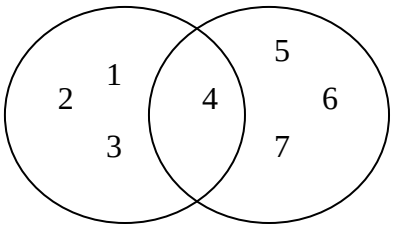
при несовпадении диапазона значения возможны ошибки, как и при введении отрицательных значений для переменной типа byte

4) **Присваивание** значений

Set4:=[] // пустое множество
set4:=['a','m'..'p']; // множество из символов a,m,n,o,p
set1:=set2; // одного типа TByteSet

5) Особые операции

Включения элемента во множество, **объединение** множеств
Set1:=set1+[5];
Include(set1, 5);
Set1:=set1+[2..7, 200];
Результатом сложения [1..4]+[4..7] будет [1..7]

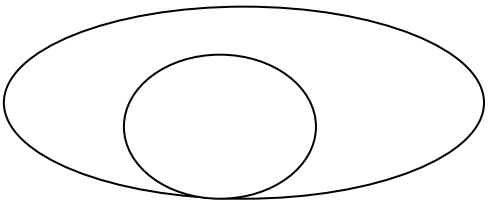


Исключения элемента(ов) из множества, **вычитание** множеств
Set1:=set1-[5];
Exclude(set1, 5);
Set1:=set1 – [2..7, 200];
Результатом вычитания [1..4]-[4..7] будет [1..3]

Пересечение множеств
Set1:=set1*[5,6];
Set2:=[1,3,6..12]*[4..8];
Результатом пересечения [1..4]*[4..7] будет [4]

Проверка наличия элемента в множестве ***IN***
If ‘z’ in set4 then
If not(‘z’ in set4) then
If [‘x’..‘z’] * set4 = [‘x’..‘z’] then

Операции **сравнения**
<, <=, =>, > вложено/содержит подмножество
=, <> // одинаковые, разные



6) **Вывод** значений
Через проверку наличия

```
Writeln(‘set1:’);
For i:=0 to 255 do
  If i in set1 then
      Write(i,’ ’);

Writeln(‘set4:’);
For ch:=’a’ to ‘z’ do
  If ch in set4 then
      Write(ch,’ ’);
```

7) Пример программы

Какие общие **латинские буквы** в двух заданных словах (длиной не более 20 символов)?

```
----------------------------------------
program PrSet;
{$APPTYPE CONSOLE}

Uses
  SysUtils;

Type
  Tset= set of 'a'..'z';
  Tstr= string[20];

Procedure vsebukvy(const slovo: Tstr; out bukvy: Tset);
Var i: byte;
Begin
  Bukvy:=[];
  For i:=1 to length(slovo) do
   If slovo[i] in ['a'..'z'] then
     Bukvy:= Bukvy + [slovo[i]];
End;


Var
  Bukvy1, Bukvy2, BukvyRes: Tset;
  Slovo1, Slovo2: Tstr;
  Ch: char;
begin
  writeln('Slovo 1 = ?');  readln(slovo1);
  writeln('Slovo 2 = ?');  readln(slovo2);
```

```
  slovo1:=LowerCase(slovo1);
  slovo2:=LowerCase(slovo2);

  vsebukvy(slovo1, Bukvy1);
  vsebukvy(slovo2, Bukvy2);
  BukvyRes:= Bukvy1 * Bukvy2;

  Writeln('Obschie: ');
  If BukvyRes =[] then writeln('net takih!')
  Else
  For ch:='a' to 'z' do
    If ch in BukvyRes then
       Write(ch,' ');

  readln
end.
-----------------
```

Какие **общие цифры** в двух заданных целых числах?

```
---------------------------------------
program PrSet2;
{$APPTYPE CONSOLE}

Type
  Tset= set of 0..9;

Procedure vsecifry(n: integer; out cifry: Tset);
Var i: byte;
Begin
  n:=abs(n);
  repeat
    cifry:= cifry + [ n mod 10 ];
    N:=n div 10;
  Until n=0;
End;


Var
  cifry1, cifry2, cifryRes: Tset;
  N1, N2: integer;
  i: byte; // или 0..10

begin
  writeln('N1 = ?');  readln(N1);
  writeln('N2 = ?');  readln(N2);

  vsecifry (N1, cifry1);
  vsecifry (N2, cifry2);
  cifryRes:= cifry1 * cifry2;

  Writeln('Obschie: ');
  If cifryRes =[] then writeln('net takih!')
  Else
  For i:=0 to 9 do
    If i in cifryRes then
```

```
      Write(i,' ');

  readln
end.
-----------------
```

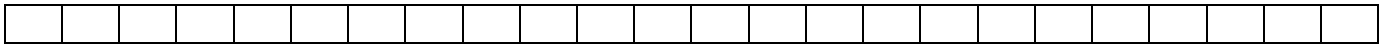
Недостатки:

1) ограниченный **диапазон** значений
2) только по одному значению – **есть/нет** – без дублей

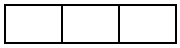
Преимущества перед массивом:

1) меньше памяти

a: array [1..24] of Boolean; // 24 раза по 1(2)байта по 8 бит



b: set of 1..24; //3 байта по 8 бит



2) короче условие

Пусть есть целое число k

if **(k = 3) or (k = 5) or (k >= 7) and (k <= 12)** then

if **k in [3, 5, 7..12]** then // так условие намного проще и короче

Тема: Константы сложных типов (массивы, записи, множества)

```
program Proj_const;
{$APPTYPE CONSOLE}

Type
  vector = array [1..5] of char;
  matrix = array [1..2, 1..3] of integer;

Const
  v: vector = ('a', 'k', 'z', '.', 'w');
  m: matrix = ( (2, -5, 20),
                (0, 10, -2) );
//---------------------------------------
Type
  Trec = record
      a,b: integer;
      c: real;
  end;

Const
  rec: Trec = (a: 5; b: 15; c: 1e-7);
//---------------------------------------
```

```
Type
  Tchisla = set of 0..9;
  TAlf = set of 'A'..'Z';
Const
  chisla : Tchisla = [3,5,7..9];
  chisla2 : set of 0..9 = [2..4, 8];
  bukvy : TAlf = [];
//----------------------------------------

Type
  TFigVid = (treug, krug, kvadr, pryam);

  TFig = record
    fig: TFigVid;
    case TFigVid of
       treug: (a,b,c: integer;);
       krug : (r: real;);
       kvadr: (aa: real;);
       pryam: (d,s: real;);
  end;

  Tmas = array [1..5] of TFig;

Const
  mas: Tmas = ( (fig: treug; a: 3; b:4; c:5),
                (fig: krug; r: 2.5),
                (fig: krug; r: 5.0),
                (fig: kvadr; aa: 3.0),
                (fig: pryam; d: 12; s: 4.5) );
//----------------------------------------

Type
  Trecmas = record
      n: integer;
      mas: array [1..4] of record
                              x,y: real;
                           end;
  end;

Const
  a: Trecmas = (n: 2;
                mas: ( (x: 12.1; y: 10),
                       (x: -5; y: 23.5),
                       (x: 0; y: 0),
                       (x: 0; y: 0) )
                );
//----------------------------------------

var b: Trecmas; i: integer;
begin
  b:=a;
  for i:=1 to b.n do
    writeln(b.mas[i].x :5:1, b.mas[i].y :5:1);
  readln;
end.
```