Кафедра «Автоматизированные станочные системы»

Dept. Of Automated Manufacturing Systems

Множества. Массивы

Троицкий Д.И.

Множество (set) – особый тип данных, строящийся на основе **перечислимого** типа

```
TYPE TS=SET OF CHAR;
```

Множество отвечает на вопрос: «Присутствует ли во множестве некоторое значение»

Множество – это не массив!

Нельзя обратиться к конкретному элементу множества, можно только узнать, есть он или нет

В памяти каждый элемент множества представляется одним битом (1 – элемент есть, 0 – элемента нет).
В множестве не может быть более 255 элементов

```
TYPE TS=SET OF INTEGER;
Set base type out of range
```

Допустимые базовые типы: BYTE, CHAR, диапазоны, перечислимые с числом элементов не более 255

Множества нельзя вывести на экран или ввести с клавиатуры

Как задаются множества

```
TYPE ts=SET OF CHAR;
VAR s:ts;
…
 s:=[‘A’,’B’,’C’,’D’..’H’];
```

Тогда в множестве s содержатся элементы:
A, B, C, D, E, F, G, H

Представление в памяти:

0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
>	?	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81

Операции над множествами

a IN b – проверяет наличие элемента a в множестве b:

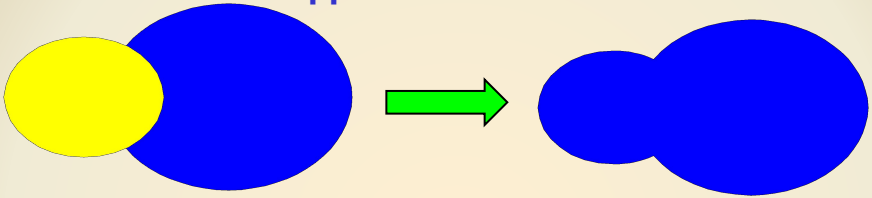
```
procedure TForm1.FormKeyPress(Sender: TObject; var Key:
Char);

BEGIN
 IF key IN [‘у’,’Y’,’д’,’Д’] THEN …
```

Операции над двумя множествами с одинаковым базовым типом:

объединение a + b
пересечение a*b
вычитание a-b

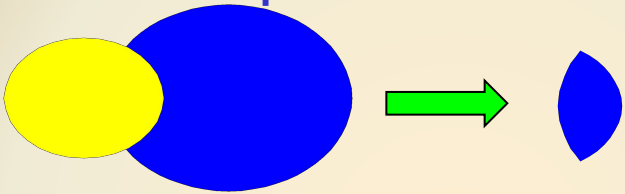
Объединение множеств



```
TYPE ts=SET OF CHAR;
VAR a,b,c:ts;
…
a:=[‘a’..’c’]; b:=[‘d’..’f’];
c:=a+b;
```

В с получим элементы a,b,c,d,e,f

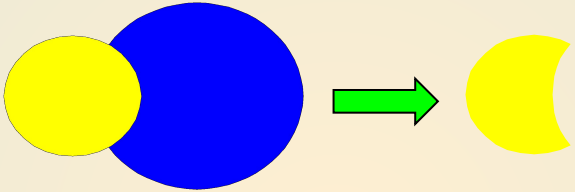
Пересечение множеств



```
TYPE ts=SET OF CHAR;
VAR a,b,c:ts;
…
a:=[‘a’..’d’]; b:=[‘d’..’f’];
c:=a*b;
```

В с получим элемент d

Вычитание множеств



```
TYPE ts=SET OF CHAR;
VAR a,b,c:ts;
…
a:=[‘a’..’d’]; b:=[‘d’..’f’];
c:=a-b;
```

В c получим элементы a,b,c

Массив (array) – самый распространенный сложный тип данных

Массив **однороден**: все его элементы имеют один и тот же **базовый** тип

Массив в памяти – структура данных с **прямым доступом** к каждому элементу (в отличие от файла на диске)

Нужный элемент в массиве находится по его **индексу**

Тип индекса также надо указывать

Описание массива=два типа данных:

1.Тип данных каждого элемента массива

2. Тип данных индекса массива.

```
TYPE TA=ARRAY[T1] OF T0
```

тип элемента

тип индекса

название типа

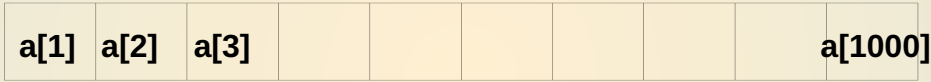
Примеры:

```
TYPE Td=ARRAY[1..10] OF REAL;
```

тип-диапазон

Выделение памяти под массив:

```
TYPE TA=ARRAY[1..1000] OF REAL;
VAR A:TA;
```



Элементы массива располагаются в памяти непрерывно, один за другим

Чтобы легко вычислять адрес каждого элемента

Зачем?

Общий размер памяти, занимаемый одним массивом, ограничен 64Кб

64Кб – максимальный размер сегмента памяти в IBM PC

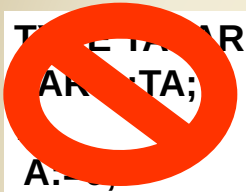
```
TYPE TA=ARRAY[1..50000] OF REAL;
Structure too large
```

Вычислим: 50000x8/1024=390.625Кб > 64Кб

Функция SIZEOF(тип) – возвращает объем памяти в байтах, занимаемый значением указанного типа

```
SIZEOF(REAL) □ 8
```

Нельзя сразу присвоить одно и то же значение всем элементам массива:



```
TYPE TA:ARRAY[1..10] OF REAL;
VAR a:TA;

A:=0;
```

Если нельзя, но очень хочется, то можно…

Обнуление массива без цикла:

```
TYPE TA:ARRAY[1..1000] OF REAL;
VAR a:TA;
…
FILLCHAR(a,SizeOf(TA),0);
```

Обращение к отдельному элементу массива – по его индексу: a[5]

Индексы можно вычислять

```
b:=a[I+1 DIV 4];
```

Что происходит при вычислении индекса?

Рассчитывается адрес ячейки памяти, начиная с которой лежит элемент массива с затребованным индексом

Адрес к-го элемента=адрес 1 элемента+ (к х размер элемента в байтах)

Многомерные массивы

Базовым типом массива может быть тоже массив:

```
TYPE TA1=ARRAY[1..20] OF REAL;
     TA2=ARRAY[1..10] OF TA1;
```

Получаем квадратную матрицу чисел 10x20

В памяти многомерный массив все равно хранится последовательно, по строкам или по столбцам

Размерностей может быть более двух

Статические и динамические массивы

Все обычные массивы – статические. Число элементов в них должно быть известно до начала выполнения программы

```
VAR a:WORD;
TYPE ta=ARRAY[1..a] OF REAL;
```

```
CONST Nmax=20;
TYPE ta=ARRAY[1..Nmax] OF REAL;
```