17.12.20г\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

19ИСиП 1Д \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Основы алгоритмизации и программирования

ТЕМА: П\З **Работа с данными типа множество**

**Множество** — это структурированный тип данных, представляющий собой набор взаимосвязанных по какому-либо признаку или группе признаков объектов, которые можно рассматривать как единое целое. Каждый объект в множестве называется элементом множества.

Все элементы множества должны принадлежать одному из порядковых типов, содержащему не более 256 значений. Этот тип называется базовым типом множества. Базовый тип задается диапазоном или перечислением.

Область значений типа множество — набор всевозможных подмножеств, составленных из элементов базового типа. В выражениях на языке Паскаль значения элементов множества указываются в квадратных скобках: [1,2,3,4], ['а',‘b','с'], ['a'..'z'].

Если множество не имеет элементов, оно называется пустым и обозначается как []. Количество элементов множества называется его мощностью.

Множество может

ет принимать все значения базового типа. Базовый тип не должен превышать 256 возможных значений. Поэтому базовым типом множества могут быть byte, char, boolean и производные от них типы.

Множество в памяти хранится как массив битов, в котором каждый бит указывает является ли элемент принадлежащим объявленному множеству или нет. Максимальное число элементов множества 256, а данные типа множество могут занимать не более 32 байт.

Число байтов, выделяемых для данных типа множество, вычисляется по формуле:

**ByteSize = (max div 8) - (min div 8) + 1,**

где max и min — верхняя и нижняя границы базового типа данного множества.

Номер байта для конкретного элемента Е вычисляется по формуле:

**ByteNumber = (E div 8) - (min div 8),**

номер бита внутри этого байта по формуле:

**BitNumber = E mod 8**

Не имеет значения порядок записи элементов множества внутри конструктора. Например, [1, 2, 3] и [3, 2, 1] — это эквивалентные множества.

Каждый элемент в множестве учитывается только один раз. Поэтому множество [1, 2, 3, 4, 2, 3, 4, 5] эквивалентно [1..5].

Переменные множественного типа описываются так:  
**Var <идентификатор> : set of <базовый тип>;**

Например:

Var A, D : Set Of Byte;

B : Set Of 'a'..'z';

C : Set Of Boolean;

Нельзя вводить значения во множественную переменную процедурой ввода и выводить процедурой вывода.

Множественная переменная может получить конкретное значение только в результате выполнения оператора присваивания:  
**<множественная переменная> := <множественное выражение>;**

Например:

A : = [50, 100, 150, 200];

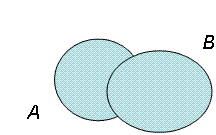
B : = ['m', 'n', 'k']; C : = [True, False];

D : = A;

Кроме того, выражения могут включать в себя операции над множествами.

**Операции над множествами**

**Объединением** двух множеств A и B называется множество, состоящее из элементов, входящих хотя бы в одно из множеств A или B. Знак операции объединения в Паскале «+».



Примеры:

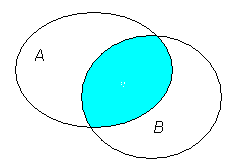
1) [1, 2, 3, 4] + [3, 4, 5, 6] => [1, 2, 3, 4, 5, 6]

2) []+[‘a’..’z’]+[‘A’..’E’, ‘k’]  => [‘A’..’E’, ‘a’..’z’]

3) [5<4, true and false] + [true]  => [false, true]

**Пересечением** двух множеств A и B называется множество, состоящее из элементов, одновременно входящих во множество A и во множество B.

Знак операции пересечения в Паскале «\*»



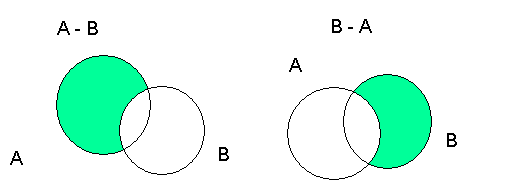
Примеры:

1) [1, 2, 3, 4] \* [3, 4, 5, 6] => [3, 4]

2) [‘a’..’z’]\*[‘A’..’E’, ‘k’]  => [‘k’]

3) [5<4, true and false] \* [true]  => []

**Разностью** двух множеств A и B называется множество, состоящее из элементов множества A, не входящих во множество B.



Примеры:

1a) [1, 2, 3, 4] - [3, 4, 5, 6] => [1, 2]

1b)  [3, 4, 5, 6] - [1, 2, 3, 4]  => [5, 6]

2a) [‘a’..’z’]-[‘A’..’E’, ‘k’]  => [‘a’..’j’, ‘i’..’z’]

2b) [‘A’..’E’, ‘k’] - [‘a’..’z’]  => [‘A’..’E’]

3a) [5<4, true and false] - [true] => [false]

3b) [true] - [5<4, true and false] => [true]

**Операция вхождения**. Это операция, устанавливающая связь между множеством и скалярной величиной, тип которой совпадает с базовым типом множества. Если x — такая скалярная величина, а M — множество, то операция вхождения записывается так: **x in M.**

Результат — логическая величина true, если значение x входит в множество M, и false — в противном случае.

Например, 4 in [3, 4, 7, 9] –– true, 5 in [3, 4, 7, 9] –– false.

Используя данную операцию, можно не только работать с элементами множества, но и, даже если в решении задачи явно не используются множества, некоторые логические выражения можно записать более лаконично.

1) Натуральное число n является двухзначным. Вместо выражения **(n >= 10) and (n <=99)** можно записать **n in [10..99]**.

2) Символ c является русской буквой. Вместо выражения **(c >= ‘А’) and (c <= ‘Я’) or (c>=‘а’) and (c<=‘п’) or (c>=‘р’) and (c<=‘я’)** пишем **c in [‘А’.. ‘Я’, ‘а’.. ‘п’, ‘р’.. ‘я’]** и т.д.

Добавить новый элемент в множество можно с использованием операции объединения. Например, a:= a+[5] Для этих же целей в Turbo Pascal 7.0 предназначена процедура Include: **include (M, A)** M – множество, A – переменная того же типа, что и элементы множества M. Тот же пример можно записать так: Include (a, 5)

Исключить элемент из множества можно с помощью операции «разность множеств». Например, a:= a-[5] Для этих же целей в Turbo Pascal 7.0 предназначена процедура Exclude: **exclude (M, A)** M – множество, A – переменная того же типа, что и элементы множества M. Тот же пример можно записать так: Exclude (a, 5)

Рассмотрим несколько примеров использования множеств при решении задач.

**Задача 1.** В городе имеется n высших учебных заведений, которые производят закупку компьютерной техники. Есть шесть компьютерных фирм: «Диалог», «Avicom», «Нэта», «Сервер», «Декада», «Dega.ru». Ответить на следующие вопросы:  
1) в каких фирмах закупка производилась каждым из вузов?  
2) в каких фирмах закупка производилась хотя бы одним из вузов?  
3) в каких фирмах ни один из вузов не закупал компьютеры?

Решим задачу с использованием множеств. Для удобства дальнейших манипуляций в порядке следования занумеруем компьютерные фирмы, начиная с единицы. Занесём информации о месте закупок компьютеров каждым из вузов в отдельное множество.

Ответ на первый вопрос можно получить, выполнив пересечение всех таких множеств.

Ответ на второй вопрос – результат объединения множеств.

И, наконец, на последний – разность множества всех фирм и множества фирм, где хотя бы один вуз делал покупки.

Описание множественного типа данных Паскаля

**Type** <имя\_типа>= **set** **of** <базовый\_тип>

**Пример множественного типа данных Паскаля**

**Type** symbol= **set** **of** char; {описан множественный тип symol из букв}  
**Var** letter, digits, sign: symbol; {описаны три переменные множественного типа}

Для того чтобы придать переменной множественного типа значение, используют **конструктор множества** – перечисление элементов множества через запятую в квадратных скобках. Например,

sign:= ['+', '-'];

Конструктор множества может содержать диапазон значений базового типа. Тогда во множества включаются все элементы диапазона. Например,

digits:= ['0' .. '9'];  
letter:= ['a' .. 'z'];

Обе формы конструирования множеств могут сочетаться. Например,

letter:= ['a' .. 'z', 'A' .. 'Z'];

Конструктор вида [] обозначает пустые множества.

В программе можно использовать множественны тип как константы, в этом случае их определяют следующим способом:

**Const** YesOrNo= ['Y', 'y', 'N', 'n'];

Можно множественный тип определить как типизированную константу:

Const digits: **set** **of** char= ['0' .. '9'];

При описании множественного тип как констант допускается использование знака “+” (слияние множеств). Например,

**Const** Yes= ['Y', 'y']; No= ['N', 'n'];  
YesOrNo= Yes+ No;

Операции над множественными типами Паскаля

С множественными типами Паскаля можно выполнять действия объединения, исключения и пересечения.

**Объединение множественных типов** содержит элементы, которые принадлежат хотя бы одному множеству, при этом каждый элемент входит в объединение только один раз. Операция объединения множеств обозначается знаком '+'.

**Пример множественных типов Паскаля**

**Type** symbol= **set** **of** char;  
**Var** small, capital, latin: symbol;  
………………  
small:= ['a' .. 'z'];  
capital:= ['A' .. 'Z'];  
latin:= small + capital; {образованы множества латинских букв путем объединения множеств small и capital}

Возможно объединять множественные типы и отдельные элементы. Например,

small:= ['c' .. 'z'];  
small:= small + ['a'] +['b'];

***Исключение*** определяется как разность множественных типов, в котором из уменьшаемого исключаются элементы, входящие в вычитаемое. Если в вычитаемом есть элементы, отсутствующие в уменьшаемом, то они никак не влияют на результат. Операция исключения обозначается знаком '-'.

**Пример исключения множественных типов Паскаля**

letter:= ['a' .. 'z']; {множества букв латинского алфавита}  
glasn:= ['a', 'e', 'o', 'u', 'i', 'y']; {множества гласных букв}  
soglasn:= letter-glasn; {образовано множества согласных букв путем исключения из множества всех букв множества гласных букв}

**Пресечение множественных типов**– множества, содержащие элементы, одновременно входящие в оба множества. Операция пересечения множеств обозначается знаком '\*'.

Преподователь \_\_\_\_\_\_\_\_\_\_\_\_\_\_Дузаев И.К.