14.12.20г\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Основы алгоритмизации и программирования

ТЕМА: П\З **Составление программы (ЧАСТЬ 2)**

СОЗДАНИЯ ПРОГРАММЫ

Изначально создание программы можно разделить на следующие этапы:

1. Разработка алгоритма работы программы.
2. Дизайн пользовательского интерфейса. Размеры и свойства окна программы, расположение элементов управление внутри окна (окошки, кнопочки, менюшки и другие) и так далее.
3. Написание логики программы. Это именно та часть, которая реагирует на события от нажатия кнопок, события операционной системы и выводит результат своей работы на экран.

Понятие множества. Объявление множества.

Множество – это фундаментальное понятие не только математики, но и всего окружающего мира. Возьмите прямо сейчас в руку любой предмет. Вот вам и множество, состоящее из одного элемента.

В широком смысле, **множество – это совокупность объектов (элементов), которые понимаются как единое целое** (по тем или иным признакам, критериям или обстоятельствам). Причём, это не только материальные объекты, но и буквы, цифры, теоремы, мысли, эмоции и т.д.

Обычно множества обозначаются большими латинскими буквами  (как вариант, с подстрочными индексами: ** и т.п.), а его элементы записываются в фигурных скобках, например:

 – множество букв русского алфавита;
 – множество натуральных чисел;

ну что же, пришла пора немного познакомиться:
 – множество студентов в 1-м ряду

… я рад видеть ваши серьёзные и сосредоточенные лица =)

Множества  и  являются конечными (состоящими из конечного числа элементов), а множество  – это пример бесконечного множества. Кроме того, в теории и на практике рассматривается так называемое пустое множество:

 – множество, в котором нет ни одного элемента.

Пример вам хорошо известен – множество  на экзамене частенько бывает пусто =)

Принадлежность элемента множеству записывается значком , например:

 – буква «бэ» принадлежит множеству букв русского алфавита;
 – буква «бета» **не** принадлежит множеству букв русского алфавита;
 – число 5 принадлежит множеству натуральных чисел;
 – а вот число 5,5 – уже нет;
 – Вольдемар не сидит в первом ряду (и тем более, не принадлежит множеству  или  =)).

В абстрактной и не очень алгебре элементы множества обозначают маленькими латинскими буквами  и, соответственно, факт принадлежности оформляется в следующем стиле:

 – элемент  принадлежит множеству 

В языке программирования Pascal существует понятие множества, имеющее смысл некоторого собрания элементов, одно и того же базового типа. Базовый тип определяет перечень всех элементов, которые вообще могут содержаться в данном множестве. В качестве базового типа может выступать любой простой порядковый тип. Но вещественные числа (**real** не порядковый тип) и строки (не простой и не порядковый тип) не могут быть элементами множества.

Размер множества в Turbo Pascal всегда ограничен некоторым предельно допустимым количеством элементов. Во множествах допускаются только такие элементы, порядковые значения которых не выходят за границы 0..255. Для целочисленных множеств это означает, что в них могут присутствовать только числа от 0 до 255. Отрицательные элементы множеств в Turbo Pascal не допускаются. Поэтому базовыми типами не могут быть типы **shortint, integer, longint**. Если же необходимо множество целочисленных о

объектов, то базовый тип должен объявлен как диапазон типа **byte**. Для множеств, содержащих символы, подобных затруднений нет, поскольку базовым типом для них является **char** (а в нем 256 значений с порядковыми номерами от 0 до 255).

В математике для обозначения множества используют фигурные скобки (например, {4, 7, 12}), в Паскаль — квадратные (например, [1, 3, 5]). Порядок элементов во множестве не имеет значения. Так, записав [3, 6, 9] или [9, 3, 6], мы будем иметь дело с одним и тем же множеством. Более того, многократное повторение одного и того же элемента не меняет множество. Например, [4, 7, 3] и [3, 7, 4, 4] – это одно и то же множество.

По форме записи объявление переменной типа множество сходно с объявлением одномерного массива:

**var**

 имя: **set** **of** тип;

Например, объявление переменной ch, рассматриваемой как множество с базовым типом **char**, имеет вид:

**var**

 ch: **set** **of** **char**;

В отличие от элементов массива, элементы множества не упорядочены и не имеют индексов.

Можно сначала объявить тип множества, а потом использовать его для объявления переменных:

**type**

 t\_ch = **set** **of** **char**;

**var**

 ch1, ch2: t\_ch;

Довольно часто в качестве базового типа множества используется тип перечисления или некоторый его диапазон:

**type**

 week\_days = (Mon, Tue, Wed, Thu, Fri);

**var**

 work\_days: **set** **of** week\_days;

 lett: **set** **of** 'A'..'Z';

Объявление переменной-множества не дает ей определенного значения.

**18+**

Построение множества

Чтобы во множестве появились элементы, необходимо выполнить оператор присваивания, в левой части которого стоит имя переменной-множества, а в правой — конструктор множества или некоторое выражение над множествами.

*Конструктор множества* — это заключенный в квадратные скобки перечень элементов, разделенных запятыми. В качестве элементов могут использоваться диапазоны значений:

**type**

 week\_days = (Mon, Tue, Wed, Thu, Fri);

**var**

 work\_days: **set** **of** week\_days;

 lett: **set** **of** 'A'..'Z';

**begin**

 work\_days := [Mon, Wed, Thu];

 lett := ['C', 'E'..'M', 'Z']

**end**.

Следует помнить, что при задании множества порядок его элементов безразличен, но при задании диапазона такой порядок важен.

Множество, в котором нет элементов, называется пустым (или нуль-множеством). В языке программирования Паскаль обозначается квадратными скобками, между которыми нет элементов:

work\_days := [ ];

Множество может быть объявлено типизированной константой, для чего в описании после знака равенства следует указать конструктор множества. Например:

**const** lett: **set** **of** ['а'..'я'] = ['а', 'е', 'и', 'о', 'у', 'ы', 'э', 'ю', 'я'];

В данном случае оператор описывает множество, элементами которого могут быть буквы русского алфавита, с записью в него начального значения, которое представляет собой множество гласных букв.

Конструируя множества, можно использовать и переменные при условии, что их текущие значения попадают в диапазон базового типа множества. Так, если ch1 и ch2 имеют тип **char**, то допустима следующая последовательность операторов:

ch1 := 'A';

ch2 := 'K';

chs := [ch1, ch2, 'M'];

В результате получится множество ['A', 'K', 'M'].

Элементы множества нельзя вводить и выводить. Для организации ввода-вывода элементов множества следует использовать вспомогательные переменные. В то же время можно использовать множества как элементы типизированных файлов.

Действия над множествами

**Объединение, пересечение и разность множеств**

Над множествами выполнимы объединение (+), пересечение (\*) и разность (-).

*Объединение* двух множеств A и B (A + B) – это новое множество, состоящее из элементов, принадлежащих множеству A или B, либо тому и другому одновременно.

**var**

 chs1, chs2, chs3: **set** **of** **char**;

**begin**

 chs1 := ['a', 'b', 'd'];

 chs2 := ['m', 'd', 'e'];

 chs3 := chs1 + chs2 + ['k', 'n'];

**end**.

Результат: chs3 = ['a', 'b', 'd', 'm', 'e', 'k', 'n'].

*Пересечение* двух множеств A и B (A \* B) – это множество, состоящее из элементов, одновременно принадлежащих множествам A и B.

chs3 := chs1 \* chs2;

Результат: chs3 = ['d']

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