# Глава 10. Множества

*– Они рисовали мышеловки, месяц, математику, множество… Ты когда-нибудь видела, как рисуют множество?*
*– Множество чего? – спросила Алиса.*
*– Ничего, – отвечала Соня. – Просто множество!*

*Льюис Кэрролл*
*«Алиса в стране чудес».*

Перейдём теперь к изучению следующего сложного типа данных – множества (**`set`**). Этот тип данных редко встречается в языках программирования, кроме Паскаля он есть, например в языке Python [i] [см. сноску в конце раздела]. Сначала рассмотрим множество как абстрактный тип данных стандарта Паскаля и сравним его с массивами и записями (см. Таблицу 10.1).

Таблица 10.1. Сравнение массивов, записей и множеств

| Массив | Запись | Множество |
|---|---|---|
| Является *переменной* | Является *переменной* | Является *переменной* |
| Состоит из *элементов* | Состоит из *полей* | Состоит из *элементов* |
| Каждый элемент переменная | Каждое поле переменная | Каждый элемент *константа* |
| Все элементы одного типа | Поля могут быть разных типов | Все элементы одного типа |
| Элементы не имеют имён | Все поля имеют имена | Элементы могут не иметь имён [1] |
| Элементы в массиве упорядочены | Поля в записи не упорядочены | Элементы не упорядочены |

Сначала обратите внимание на «не очевидные» свойства множества. Элементы, входящие в множество, нельзя менять, они не переменные, а *константы*. Элемент множества можно только поместить (добавить) в множество и убрать (удалить) из множества. При добавлении в множество элемента, который там уже есть, и при удалении из множества элемента, которого там нет, ошибка не фиксируется и множество, естественно, *не меняется*. Таким образом, если, например, в множестве целых чисел «изменить» элемент 2 на элемент 3, то это просто означает, что из множества удаляется элемент 2 и добавляется элемент 3. Соответственно, в Паскале для множества нет селектора элемента, аналогичного переменной с индексом `X[i]` или поля записи `X.Y`.

В конкретной реализации в языке Паскаль на математическое понятие множества накладывается много ограничений. В отличие от математики, где, например, существует множество, состоящее из столов и стульев в некоторой комнате, элементы множества в Паскале *одного типа*, он называется базовым типом множества.

Далее, как Вы знаете, для конечных множеств (а только такие и могут храниться в ограниченной памяти ЭВМ) понятие мощности множества сводится к числу его элементов. В Паскале для каждого типа множества при его описании определяется максимальное количество элементов, которые могут содержаться в множестве (полное множество). При порождении переменной типа множество именно под это число элементов и выделяется нужный объём памяти.

Для хранения каждого элемента в множестве достаточно одного бита («есть элемент в множестве» и «нет элемента в множестве»). Таким образом, для множества из 256 элементов надо отвести 256 бит=32*8 байта памяти. Переменные большóго размера тяжело обрабатывать и передавать из одной части программы в другую (например, возвращать как результат работы функции). Поэтому накладывается ограничение, что мощность множества в Паскале не может превышать число 256.[2] Итак, в качестве базового множества можно выбрать любой дискретный тип с числом элементов не более 256 и порядковые номера элементов которых не превышают 255.

Синтаксис описания типа множества:

---

[1] Так как элементы это константы, то им можно присвоить имена в разделе констант или при описании перечислимого типа.

[2] На современных ЭВМ размер так называемых векторных регистров сейчас достигает 256 и даже 512 бит. На этих регистрах можно эффективно хранить т обрабатывать такие множества.

```
<тип множества> ::= set of <базовый тип>;
<базовый тип> ::= <дискретный тип>
```

В Паскале определены константы-множества, их синтаксис:

```
<константа-множество> ::= [<секция>{,<секция>}...] | []
<секция> ::= {<выражение> | <выражение> ..<выражение>}
<выражение> ::= <выражение базового типа>
```

Как видим, константы типа множества заключаются в квадратные скобки, причём, естественно, существует пустое множество `[]`.

В стандарте языка Паскаль переменные-множества порождаются с неопределёнными значениями, а в языке Free Pascal переменные-множества статического класса порождаются пустыми, а переменные остальных классов порождаются «по настоящему» с неопределёнными значениями.

Примеры множеств:

```
var
  K: integer;
  X,Y: set of char;     { базовый тип char }
  Z: set of '0'..'9';  { базовый тип char }
  A: set of 0..99;      { базовый тип integer }
  B: set of 300..310; { ОШИБКА! ord(300)>256 }
begin {$R-}
  X:=Y; Z:=X;  { есть совместимость по присваиванию, ПОЧЕМУ ? }
  X:=A;  { ОШИБКА! Нет совместимости по присваиванию }
  K:=100;
  A:=[];  { пустое множество }
  A:=[3..1];  { пустое множество! }
  A:=[90,95,K-10..200];  { A=[90..99] }
  A:=[K-300];  { A=[56] ⚠ }
```

Засада! Отрицательные числа представляются в компьютере в так называемом *дополнительном коде*, так что `K-300=-200=256-200=56`. Это тема курса по архитектуре ЭВМ. Впрочем, если включить в программе режим `{$R+}` контроля выхода значений за допустимый диапазон, то при выполнении оператора `A:=[K-300]` будет зафиксирована ошибка. Аналогично

```
  A:=[-5..5];  { A=[251..5]=[]  Пустое множество ⚠ }
```

Рассмотрим теперь операции над множествами. Из математических операций над множествами в Паскале реализованы следующие:

1). Объединение множеств $X \cup Y$ обозначается в Паскале как `X + Y`.

2). Пересечение множеств $X \cap Y$ обозначается в Паскале как `X * Y`.

3). Разность (вычитание) множеств $X \setminus Y$ обозначается в Паскале как `X - Y`.

4). Принадлежность элемента множеству $p \in X$ обозначается в Паскале как `p in X`.

5). В языке Free Pascal реализована также операция получения симметрической разности (symmetrical difference) двух множеств $X \Delta Y$ она обозначается как `X >< Y`.

Далее, как известно, над скалярными типами в Паскале есть операции отношения `=`, `<>`, `>`, `<`, `>=` и `<=`. Для двух множеств (одного типа!) определены все из этих отношений, кроме `>` и `<`.[2] Итак, `X >= Y` обозначает $X \supseteq Y$, а `X <= Y` обозначает $X \subseteq Y$.

Заметим, что в Паскале не реализованы отдельные операции добавления в множество элемента и удаления элемента из множества, вместо этого предлагается использовать операции объединения и

---

[1] В описании синтаксиса множеств квадратные скобки `[]` используются как символы языка Паскаль, а не как служебные символы метаязыка БНФ (для задания необязательности конструкции).

[2] В некоторые реализации Паскаля (например, в PascalABC.NET) над множествами реализованы и операции `>` и `<`, но в стандарте Паскаля и в языке Free Pascal их (по непонятным причинам) нет.

разности множеств. Например, для удаления из множества X символа 'A' следует использовать оператор `X:=X-['A']`, а для добавления `X:=X+['A']`. Здесь учащиеся часто для добавления элемента в множество пытаются использовать оператор `X:=X+'A'`, что является синтаксической ошибкой.

Впрочем, в языке Free Pascal вместо `X:=X-['A']` можно использовать стандартную процедуру `Exclude(X,'A')`, а вместо `X:=X+['A']` использовать `Include(X,'A')`.

В качестве примера рассмотрим следующую задачу. Надо ввести текст до точки и вывести все символы цифр, входящие в этот текст. Можно предложить следующее решение с использованием множеств:

```
  var c: char; M: set of '0'..'9';
begin M:=[];
  repeat read(c); M:=M+[c]
  until c='.';
  Write('В текст входят цифры: ');
  for c:='0' to '9' do
    if c in M then Write(c:3);
  Writeln
end.
```

Обратите внимание, что элементы в множестве Паскаля никак не упорядочены, поэтому, чтобы напечатать все символы цифр, входящие в множество M, приходится делать полный перебор этого множества по возрастанию элементов.

Множества-константы часто используются в программах, например, логическое условие принадлежности символа c множеству символов-цифр, описанное как `(c<='9') and (c>='0')` более компактно записывается в виде `c in ['0'..'9']`.

Работа с множествами менее эффективна, чем с массивами, например, предыдущую задачу можно было бы решить так:

```
var c: char; M: array['0'..'9'] of boolean;
begin
  for c:='0' to '9' do M[c]:=false; { M:=[] }
  repeat read(c);
    if c in ['0'..'9'] then M[c]:=true
  { или M[c]:=M[c] or (c in ['0'..'9'])
    это лучше, так как нет условного оператора if,
    который очень «не любит» компьютер }
  until c='.';
  Write('В текст входят цифры: ');
  for c:='0' to '9' do if M[c] then Write(c:3);
  Writeln
end.
```

Эта программа выполняется быстрее, но требует и больше памяти, т.к. переменная M как множество занимает один байт, а как массив занимает 10 байт. Впрочем, так как в программе на языке Free Pascal адрес переменной в памяти обычно выравнивает на границу 16 байт, то отвести под переменную меньше 16 байт просто так не получится 🐷.

В качестве примера работы с множествами в языке Free Pascal, напишем функцию с именем card, чтобы она выдавала мощность множества типа `set of byte`:

```
type SetB=set of byte;
function card(x: SetB): byte;
  var i: integer;
begin card:=0;
  for i:=0 to 255 do
{ можно «красиво» for i in [0..255] do }
      if i in x then ⚠ inc(card)
```

```
{ в стандарте Паскаля такой фокус не пройдёт }
end;
```

---

[i] Для продвинутых читателей. Реализация множеств в языке Python много богаче, чем в Паскале и по возможностям более похоже на множества в математике. Скажем, множесво может включать элементы разных типов (правда, только *иммутабельных*, см раздел 3.3.3), например:

```
f=["a",True,123,3.14159]
```

В отличие от Паскаля (и математики), однако, множество в языке Python (как и, например, массив) есть упорядоченный объект, все входящие в него элементы пронумерованы (с нуля), т.е. `f[0]→"a"`, `f[3]→3.14159` и т.д.