

Объявление множеств

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 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.

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 ['a'..'я'] = ['a', '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'].

Разность двух множеств A и B ($A - B$) – это новое множество, состоящее из элементов множества A, не вошедших в множество B.

```
chs1 := ['a', 'e', 't'];  
chs2 := chs1 - ['e']           { ['a', 't'] }  
chs3 := ['m', 'n', 't'] - chs2 { ['m', 'n'] }
```

Манипулируя операциями над множествами, можно добавлять элементы к множествам или удалять их.

Для вставки и удаления элементов при работе с множествами в Pascal введены две процедуры:

include (имя_множества, элемент)

exclude (имя_множества, элемент)

Первая из них позволяет выполнить добавление одного элемента в указанное множество, а вторая удалить. Например:

```
include (chs1, 'g');    { аналогично chs1 + ['g'] }  
exclude (chs2, 'a');    { аналогично chs2 - ['a'] }
```

Другие операции над множествами

Над множествами можно выполнять четыре операции сравнения:
=, <>, >=, <=.

Два множества A и B равны ($A = B$), если каждый элемент множества A является элементом множества B и наоборот.

Два множества A и B не равны ($A \neq B$), если они отличаются хотя бы одним элементом.

Множество A является подмножеством множества B ($A \subseteq B$, или $B \supseteq A$), если каждый элемент из A присутствует в B.

Имеется также возможность выяснить, принадлежит ли данный элемент некоторому множеству. Для этого служит операция **in**. Пусть A – множество элементов некоторого базового типа, а x – переменная (константа, выражение) этого типа. Тогда выражение $x \text{ in } A$ истинно, если значение x является элементом множества A.

Все операции сравнения множеств, а также операция **in** возвращают логическое значение **true** или **false**.

В сложных выражениях над множествами операции имеют следующие приоритеты:

1. *
2. +, -
3. =, <>, <=, >=, in

Рассмотрим несколько примеров.

Пример. Пусть в вашем распоряжении имеется множество из трех монет разного достоинства: 1 р, 5 р, 10 р. Из этих монет можно составить следующие подмножества (их число равно $2^3 - 1 = 7$):

```
{1};  
{5};  
{10};  
{1, 5};  
{1, 10};  
{5, 10};
```

```
{1, 5, 10};  
{ }
```

Эти подмножества и будут принадлежать некоторому множеству, тип которого назовем **sum**. Сами элементы (монеты), из которых составляется подмножество, пусть принадлежат некоторому базовому типу, который назовем **monet**.

Опишем типы данных этого примера:

```
type  
    monet = (m1, m5, m10);  
    sum = set of monet;
```

Пример. Рассмотрим в качестве элементов базового типа сигналы от 4-х абонентов (ab1, ab2, ab3, ab4), поступающие на телефонную станцию. Обозначим базовый тип через **abonent**:

```
type abonent = (ab1, ab2, ab3, ab4),  
тогда комбинации сигналов можно описать переменной типа множество.  
Назовем этот тип sing:
```

```
sing = set of abonent;  
Тип sing описывает 16 комбинаций.
```

В общем виде тип множество описывается так:

```
type a = set of tc;  
здесь a – идентификатор типа (произвольный); tc – тип компонент множества  
называемый базовым типом.
```

Значение переменной типа множество изображается путем перечисления конкретных компонентов, разделенных запятыми и заключенных в квадратные скобки.

Пример. Пусть базовый тип **int** и тип **a** заданы так:

```
type  
    int = 1..3;  
    a = set of int;
```

Переменная *a* в этом случае может принимать восемь значений: [1], [2], [3], [1, 2], [1, 3], [2, 3], [1, 2, 3], []. Например, если переменная *b* имеет тип *a*, то можно присвоить ей одно из перечисленных выше значений: *b* := [1, 3]; *b* := [1, 3, 2]; и т.д.

Пример. Если базовый тип описывает набор двоичных бит, то можно получить их комбинацию. Пусть

```
type
  bin = (bit1, bit2, bit3);
  bts = set of bin;
```

Переменная типа *bts* может принимать восемь значений.

Таким образом, используя переменные типа *set*, можно работать с битовой информацией.

В качестве базового типа может использоваться любой простой тип, кроме *real*. Если задача требует использования множества, состоящего из большого числа элементов, то его можно представить как массив множеств, состоящих из допустимого числа элементов.

Данные типа *set*

Данные типа *set* задаются путем перечисления значений, разделенных запятыми и заключенных в квадратные скобки.

Общий вид:

```
[expr1, expr2, ...exprn];
```

Здесь *expr_i* – выражение базового типа.

Порядок следования выражений несущественен. Непустой набор может быть также выражением вида:

```
[expr1];
[expr1..exprk];
[expr1, exprk..exprm];
```

Данные вида [*expr₁*..*expr_k*] соответствуют набору всех элементов базового типа от значения *expr₁* до *expr_k*.

Пример. [3 * 6 – 7..15 + 4] соответствует набору [11..19], т.е. [11, 12, 13, 14, 15, 16, 17, 18, 19].

Если окажется, что для [*i*..*j*], *i* > *j*, то такое множество интерпретируется как пустое, а в случае *i* = *j* – как множество, содержащее один элемент – *i*.

Пример. [3 * 6 – 7..5 + 6] эквивалентно [11].

Пример.

```
type color = (red, yellow, green, blue);
var mix: set of color;
.....
mix := [red, blue];
```

Пример.

```
type n = (1, 3, 5, 7, 9);
var k: set of n;
```

```
.....
k := [3..9];
```

В этом случае в *k* запишется комбинация [3, 5, 7, 9];

Операции над множествами
пт, 11/05/2010 - 17:09 — tech

К переменным типа *set* применимы следующие операции: =, <>, >=, <=, in, +, -, *.

Операции = и <> используются для проверки эквивалентности: два значения переменной типа *set* считаются равными, если они состоят из одних и тех же элементов.

Пример.

[1, 3] = [3, 1] возвращает true,

$[1..3] = [1, 2, 3]$ возвращает true,

$[1] \subsetneq [2]$ возвращает true,

$[1, 2, 3] = [1, 4, 3]$ возвращает false,

$[\text{red}, \text{blue}] = [\text{red}, \text{yellow}]$ возвращает false.

Операции $\supseteq$ и $\subseteq$ используются для проверки принадлежности одного множества другому: так, если множество a содержится во множестве b , то $a \subseteq b$ дает true.

Пример.

$[1, 2] \subseteq [1, 2, 3]$ дает true

Пустое множество $[\]$ содержится во всех множествах, т.е. всегда $[\] \subseteq [b]$ дает true.

Операция in используется для установления наличия определенного элемента в величине типа set. Так, если x есть элемент множества b , то $(x \text{ in } b)$ дает true. Общий вид:

$x \text{ in } a$;

здесь x – величина базового типа, a – величина типа set.

Пример.

$\text{red in } [\text{red}, \text{yellow}]$ возвращает true;

$\text{red in } [\text{blue}, \text{green}]$ возвращает false.

Замечание 1. Чтобы проверить, является ли значение n цифрой, удобно использовать операцию in следующим образом:

if $n \text{ in } [0..9]$ then ...

Замечание 2. Результат операции in может быть неопределенным в некоторых случаях.

Пример. Пусть

a : set of 1..50;

x : integer.

Если заслать в x число, большее максимального значения 50 (например, $x := 55$), то в этом случае результат операции $x \text{ in } a$ не всегда false.

К переменным типа set, относящимся к одному и тому же конкретному типу, применимы операции:

+ объединение;

* пересечение;

- дополнение.

Пусть a и b – операнды, имеющие один и тот же конкретный тип. Тогда $a + b$ представляет собой объединение множества элементов, входящих в a и b (одинаковые элементы не повторяются).

$a * b$ – пересечение множества элементов a и b (только те, которые есть в обоих множествах).

$a - b$ – множество элементов, которые есть в a , но отсутствуют в b .

Пример.

$[1, 3] + [1, 4] = [1, 3, 4]$;

$[1, 3] * [1, 4] = [1]$;

$[1, 3] - [1, 4] = [3]$.

Операция $a := a + x$ добавляет элемент x к множеству a . Если x уже имелся в a , то множество a не меняется. $a := a - x$ исключает x из a . Если x отсутствовал в a , то множество a не меняется. Операции над множествами
пт, 11/05/2010 - 17:09 — tech

К переменным типа set применимы следующие операции: $=$, $\subsetneq$, $\supseteq$, $\subseteq$, in , $+$, $-$, $*$.

Операции $=$ и $\subsetneq$ используются для проверки эквивалентности: два значения переменной типа set считаются равными, если они состоят из одних и тех же элементов.

Пример.

$[1, 3] = [3, 1]$ возвращает true,

$[1..3] = [1, 2, 3]$ возвращает true,

$[1] \subsetneq [2]$ возвращает true,

$[1, 2, 3] = [1, 4, 3]$ возвращает false,

$[\text{red}, \text{blue}] = [\text{red}, \text{yellow}]$ возвращает false.

Операции $\geq$ и $\leq$ используются для проверки принадлежности одного множества другому: так, если множество a содержится во множестве b , то $a \leq b$ дает true.

Пример.

$[1, 2] \leq [1, 2, 3]$ дает true

Пустое множество $[]$ содержится во всех множествах, т.е. всегда $[] \leq [b]$ дает true.

Операция `in` используется для установления наличия определенного элемента в величине типа `set`. Так, если x есть элемент множества b , то $(x \text{ in } b)$ дает true. Общий вид:

$x \text{ in } a$;

здесь x – величина базового типа, a – величина типа `set`.

Пример.

`red in [red, yellow]` возвращает true;

`red in [blue, green]` возвращает false.

Замечание 1. Чтобы проверить, является ли значение n цифрой, удобно использовать операцию `in` следующим образом:

`if n in [0..9] then ...`

Замечание 2. Результат операции `in` может быть неопределенным в некоторых случаях.

Пример. Пусть

a : set of 1..50;

x : integer.

Если заслать в x число, большее максимального значения 50 (например, $x := 55$), то в этом случае результат операции $x \text{ in } a$ не всегда false.

К переменным типа `set`, относящимся к одному и тому же конкретному типу, применимы операции:

+ объединение;

* пересечение;

- дополнение.

Пусть a и b – операнды, имеющие один и тот же конкретный тип. Тогда $a + b$ представляет собой объединение множества элементов, входящих в a и b (одинаковые элементы не повторяются).

$a * b$ – пересечение множества элементов a и b (только те, которые есть в обоих множествах).

$a - b$ – множество элементов, которые есть в a , но отсутствуют в b .

Пример.

$[1, 3] + [1, 4] = [1, 3, 4]$;

$[1, 3] * [1, 4] = [1]$;

$[1, 3] - [1, 4] = [3]$.

Операция $a := a + x$ добавляет элемент x к множеству a . Если x уже имелся в a , то множество a не меняется. $a := a - x$ исключает x из a . Если x отсутствовал в a , то множество a не меняется. Операции над множествами
пт, 11/05/2010 - 17:09 — tech

К переменным типа `set` применимы следующие операции: $=$, $\langle \rangle$, $\geq$, $\leq$, `in`, $+$, $-$, $*$.

Операции $=$ и $\langle \rangle$ используются для проверки эквивалентности: два значения переменной типа `set` считаются равными, если они состоят из одних и тех же элементов.

Пример.

$[1, 3] = [3, 1]$ возвращает true,

$[1..3] = [1, 2, 3]$ возвращает true,

$[1] \langle \rangle [2]$ возвращает true,

$[1, 2, 3] = [1, 4, 3]$ возвращает false,

`[red, blue] = [red, yellow]` возвращает false.

Операции $\geq$ и $\leq$ используются для проверки принадлежности одного множества другому: так, если множество a содержится во множестве b , то $a \leq b$ дает true.

Пример.

$[1, 2] \leq [1, 2, 3]$ дает true

Пустое множество $[]$ содержится во всех множествах, т.е. всегда $[] \leq [b]$ дает true.

Операция `in` используется для установления наличия определенного элемента в величине типа `set`. Так, если `x` есть элемент множества `b`, то $(x \text{ in } b)$ дает true. Общий вид:

$x \text{ in } a$;

здесь `x` – величина базового типа, `a` – величина типа `set`.

Пример.

`red in [red, yellow]` возвращает true;

`red in [blue, green]` возвращает false.

Замечание 1. Чтобы проверить, является ли значение `n` цифрой, удобно использовать операцию `in` следующим образом:

`if n in [0..9] then ...`

Замечание 2. Результат операции `in` может быть неопределенным в некоторых случаях.

Пример. Пусть

`a: set of 1..50`;

`x: integer`.

Если заслать в `x` число, большее максимального значения 50 (например, `x := 55`), то в этом случае результат операции `x in a` не всегда false.

К переменным типа `set`, относящимся к одному и тому же конкретному типу, применимы операции:

`+` объединение;

`*` пересечение;

`-` дополнение.

Пусть `a` и `b` – операнды, имеющие один и тот же конкретный тип. Тогда

`a + b` представляет собой объединение множества элементов, входящих в `a` и `b` (одинаковые элементы не повторяются).

`a * b` – пересечение множества элементов `a` и `b` (только те, которые есть в обоих множествах).

`a - b` – множество элементов, которые есть в `a`, но отсутствуют в `b`.

Пример.

$[1, 3] + [1, 4] = [1, 3, 4]$;

$[1, 3] * [1, 4] = [1]$;

$[1, 3] - [1, 4] = [3]$.

Операция `a := a + x` добавляет элемент `x` к множеству `a`. Если `x` уже имелся в `a`, то множество `a` не меняется. `a := a - x` исключает `x` из `a`. Если `x` отсутствовал в `a`, то множество `a` не меняется. Операции над множествами
пт, 11/05/2010 - 17:09 — tech

К переменным типа `set` применимы следующие операции: `=`, `<>`, `>=`, `<=`, `in`, `+`, `-`, `*`.

Операции `=` и `<>` используются для проверки эквивалентности: два значения переменной типа `set` считаются равными, если они состоят из одних и тех же элементов.

Пример.

$[1, 3] = [3, 1]$ возвращает true,

$[1..3] = [1, 2, 3]$ возвращает true,

$[1] <> [2]$ возвращает true,

$[1, 2, 3] = [1, 4, 3]$ возвращает false,

$[\text{red}, \text{blue}] = [\text{red}, \text{yellow}]$ возвращает false.

Операции `>=` и `<=` используются для проверки принадлежности одного множества другому: так, если множество `a` содержится во множестве `b`, то `a <= b` дает true.

Пример.

$[1, 2] <= [1, 2, 3]$ дает true

Пустое множество [] содержится во всех множествах, т.е. всегда [] <= [b] дает true.

Операция in используется для установления наличия определенного элемента в величине типа set. Так, если x есть элемент множества b, то (x in b) дает true. Общий вид:

x in a;
здесь x – величина базового типа, a – величина типа set.

Пример.

red in [red, yellow] возвращает true;
red in [blue, green] возвращает false.

Замечание 1. Чтобы проверить, является ли значение n цифрой, удобно использовать операцию in следующим образом:

if n in [0..9] then ...

Замечание 2. Результат операции in может быть неопределенным в некоторых случаях.

Пример. Пусть

a: set of 1..50;
x: integer.
Если заслать в x число, большее максимального значения 50 (например, x := 55), то в этом случае результат операции x in a не всегда false.

К переменным типа set, относящимся к одному и тому же конкретному типу, применимы операции:

+ объединение;
* пересечение;
- дополнение.

Пусть a и b – операнды, имеющие один и тот же конкретный тип. Тогда a + b представляет собой объединение множества элементов, входящих в a и b (одинаковые элементы не повторяются).

a * b – пересечение множества элементов a и b (только те, которые есть в обоих множествах).

a – b – множество элементов, которые есть в a, но отсутствуют в b.

Пример.

[1, 3] + [1, 4] = [1, 3, 4];
[1, 3] * [1, 4] = [1];
[1, 3] - [1, 4] = [3].

Операция a := a + x добавляет элемент x к множеству a. Если x уже имелся в a, то множество a не меняется. a := a – x исключает x из a. Если x отсутствовал в a, то множество a не меняется. Операции над множествами
пт, 11/05/2010 - 17:09 — tech

К переменным типа set применимы следующие операции: =, <>, >=, <=, in, +, -, *.

Операции = и <> используются для проверки эквивалентности: два значения переменной типа set считаются равными, если они состоят из одних и тех же элементов.

Пример.

[1, 3] = [3, 1] возвращает true,
[1..3] = [1, 2, 3] возвращает true,
[1] <> [2] возвращает true,
[1, 2, 3] = [1, 4, 3] возвращает false,
[red, blue] = [red, yellow] возвращает false.

Операции >= и <= используются для проверки принадлежности одного множества другому: так, если множество a содержится во множестве b, то a <= b дает true.

Пример.

[1, 2] <= [1, 2, 3] дает true
Пустое множество [] содержится во всех множествах, т.е. всегда [] <= [b] дает true.

Операция `in` используется для установления наличия определенного элемента в величине типа `set`. Так, если `x` есть элемент множества `b`, то `(x in b)` дает `true`. Общий вид:

`x in a;`

здесь `x` – величина базового типа, `a` – величина типа `set`.

Пример.

`red in [red, yellow]` возвращает `true`;

`red in [blue, green]` возвращает `false`.

Замечание 1. Чтобы проверить, является ли значение `n` цифрой, удобно использовать операцию `in` следующим образом:

`if n in [0..9] then ...`

Замечание 2. Результат операции `in` может быть неопределенным в некоторых случаях.

Пример. Пусть

`a: set of 1..50;`

`x: integer.`

Если заслат в `x` число, большее максимального значения 50 (например, `x := 55`), то в этом случае результат операции `x in a` не всегда `false`.

К переменным типа `set`, относящимся к одному и тому же конкретному типу, применимы операции:

+ объединение;

* пересечение;

- дополнение.

Пусть `a` и `b` – операнды, имеющие один и тот же конкретный тип. Тогда `a + b` представляет собой объединение множества элементов, входящих в `a` и `b` (одинаковые элементы не повторяются).

`a * b` – пересечение множества элементов `a` и `b` (только те, которые есть в обоих множествах).

`a - b` – множество элементов, которые есть в `a`, но отсутствуют в `b`.

Пример.

`[1, 3] + [1, 4] = [1, 3, 4];`

`[1, 3] * [1, 4] = [1];`

`[1, 3] - [1, 4] = [3].`

Операция `a := a + x` добавляет элемент `x` к множеству `a`. Если `x` уже имелся в `a`, то множество `a` не меняется. `a := a - x` исключает `x` из `a`. Если `x` отсутствовал в `a`, то множество `a` не меняется. Операции над множествами
пт, 11/05/2010 - 17:09 — tech

К переменным типа `set` применимы следующие операции: `=`, `<>`, `>=`, `<=`, `in`, `+`, `-`, `*`.

Операции `=` и `<>` используются для проверки эквивалентности: два значения переменной типа `set` считаются равными, если они состоят из одних и тех же элементов.

Пример.

`[1, 3] = [3, 1]` возвращает `true`,

`[1..3] = [1, 2, 3]` возвращает `true`,

`[1] <> [2]` возвращает `true`,

`[1, 2, 3] = [1, 4, 3]` возвращает `false`,

`[red, blue] = [red, yellow]` возвращает `false`.

Операции `>=` и `<=` используются для проверки принадлежности одного множества другому: так, если множество `a` содержится во множестве `b`, то `a <= b` дает `true`.

Пример.

`[1, 2] <= [1, 2, 3]` дает `true`

Пустое множество `[]` содержится во всех множествах, т.е. всегда `[] <= [b]` дает `true`.

Операция `in` используется для установления наличия определенного элемента в величине типа `set`. Так, если `x` есть элемент множества `b`, то `(x in b)` дает `true`. Общий вид:

$x \text{ in } a$;

здесь x – величина базового типа, a – величина типа `set`.

Пример.

`red in [red, yellow]` возвращает `true`;

`red in [blue, green]` возвращает `false`.

Замечание 1. Чтобы проверить, является ли значение n цифрой, удобно использовать операцию `in` следующим образом:

`if n in [0..9] then ...`

Замечание 2. Результат операции `in` может быть неопределенным в некоторых случаях.

Пример. Пусть

a : set of 1..50;

x : integer.

Если заслать в x число, большее максимального значения 50 (например, $x := 55$), то в этом случае результат операции $x \text{ in } a$ не всегда `false`.

К переменным типа `set`, относящимся к одному и тому же конкретному типу, применимы операции:

$+$ объединение;

$*$ пересечение;

$-$ дополнение.

Пусть a и b – операнды, имеющие один и тот же конкретный тип. Тогда $a + b$ представляет собой объединение множества элементов, входящих в a и b (одинаковые элементы не повторяются).

$a * b$ – пересечение множества элементов a и b (только те, которые есть в обоих множествах).

$a - b$ – множество элементов, которые есть в a , но отсутствуют в b .

Пример.

$[1, 3] + [1, 4] = [1, 3, 4]$;

$[1, 3] * [1, 4] = [1]$;

$[1, 3] - [1, 4] = [3]$.

Операция $a := a + x$ добавляет элемент x к множеству a . Если x уже имелся в a , то множество a не меняется. $a := a - x$ исключает x из a . Если x отсутствовал в a , то множество a не меняется. Операции над множествами
пт, 11/05/2010 - 17:09 — tech

К переменным типа `set` применимы следующие операции: $=$, $\langle \rangle$, $\geq$, $\leq$, `in`, $+$, $-$, $*$.

Операции $=$ и $\langle \rangle$ используются для проверки эквивалентности: два значения переменной типа `set` считаются равными, если они состоят из одних и тех же элементов.

Пример.

$[1, 3] = [3, 1]$ возвращает `true`,

$[1..3] = [1, 2, 3]$ возвращает `true`,

$[1] \langle \rangle [2]$ возвращает `true`,

$[1, 2, 3] = [1, 4, 3]$ возвращает `false`,

$[\text{red}, \text{blue}] = [\text{red}, \text{yellow}]$ возвращает `false`.

Операции $\geq$ и $\leq$ используются для проверки принадлежности одного множества другому: так, если множество a содержится во множестве b , то $a \leq b$ дает `true`.

Пример.

$[1, 2] \leq [1, 2, 3]$ дает `true`

Пустое множество $[]$ содержится во всех множествах, т.е. всегда $[] \leq [b]$ дает `true`.

Операция `in` используется для установления наличия определенного элемента в величине типа `set`. Так, если x есть элемент множества b , то $(x \text{ in } b)$ дает `true`. Общий вид:

$x \text{ in } a$;

здесь x – величина базового типа, a – величина типа `set`.

Пример.

red in [red, yellow] возвращает true;
red in [blue, green] возвращает false.

Замечание 1. Чтобы проверить, является ли значение n цифрой, удобно использовать операцию in следующим образом:

if n in [0..9] then ...

Замечание 2. Результат операции in может быть неопределенным в некоторых случаях.

Пример. Пусть

a: set of 1..50;

x: integer.

Если заслат в x число, большее максимального значения 50 (например, x := 55), то в этом случае результат операции x in a не всегда false.

К переменным типа set, относящимся к одному и тому же конкретному типу, применимы операции:

+ объединение;

* пересечение;

- дополнение.

Пусть a и b – операнды, имеющие один и тот же конкретный тип. Тогда a + b представляет собой объединение множества элементов, входящих в a и b (одинаковые элементы не повторяются).

a * b – пересечение множества элементов a и b (только те, которые есть в обоих множествах).

a – b – множество элементов, которые есть в a, но отсутствуют в b.

Пример.

[1, 3] + [1, 4] = [1, 3, 4];

[1, 3] * [1, 4] = [1];

[1, 3] - [1, 4] = [3].

Операция a := a + x добавляет элемент x к множеству a. Если x уже имелся в a, то множество a не меняется. a := a - x исключает x из a. Если x отсутствовал в a, то множество a не меняется. Операции над множествами

пт, 11/05/2010 - 17:09 — tech

К переменным типа set применимы следующие операции: =, <>, >=, <=, in, +, -, *.

Операции = и <> используются для проверки эквивалентности: два значения переменной типа set считаются равными, если они состоят из одних и тех же элементов.

Пример.

[1, 3] = [3, 1] возвращает true,

[1..3] = [1, 2, 3] возвращает true,

[1] <> [2] возвращает true,

[1, 2, 3] = [1, 4, 3] возвращает false,

[red, blue] = [red, yellow] возвращает false.

Операции >= и <= используются для проверки принадлежности одного множества другому: так, если множество a содержится во множестве b, то a <= b дает true.

Пример.

[1, 2] <= [1, 2, 3] дает true

Пустое множество [] содержится во всех множествах, т.е. всегда [] <= [b] дает true.

Операция in используется для установления наличия определенного элемента в величине типа set. Так, если x есть элемент множества b, то (x in b) дает true. Общий вид:

x in a;

здесь x – величина базового типа, a – величина типа set.

Пример.

red in [red, yellow] возвращает true;

red in [blue, green] возвращает false.

Замечание 1. Чтобы проверить, является ли значение n цифрой, удобно использовать операцию `in` следующим образом:

if n in [0..9] then ...

Замечание 2. Результат операции `in` может быть неопределенным в некоторых случаях.

Пример. Пусть

a : set of 1..50;

x : integer.

Если заслать в x число, большее максимального значения 50 (например, $x := 55$), то в этом случае результат операции x in a не всегда false.

К переменным типа `set`, относящимся к одному и тому же конкретному типу, применимы операции:

+ объединение;

* пересечение;

- дополнение.

Пусть a и b – операнды, имеющие один и тот же конкретный тип. Тогда $a + b$ представляет собой объединение множества элементов, входящих в a и b (одинаковые элементы не повторяются).

$a * b$ – пересечение множества элементов a и b (только те, которые есть в обоих множествах).

$a - b$ – множество элементов, которые есть в a , но отсутствуют в b .

Пример.

$[1, 3] + [1, 4] = [1, 3, 4]$;

$[1, 3] * [1, 4] = [1]$;

$[1, 3] - [1, 4] = [3]$.

Операция $a := a + x$ добавляет элемент x к множеству a . Если x уже имелся в a , то множество a не меняется. $a := a - x$ исключает x из a . Если x отсутствовал в a , то множество a не меняется. Операции над множествами

пт, 11/05/2010 - 17:09 — tech

К переменным типа `set` применимы следующие операции: `=`, `<>`, `>=`, `<=`, `in`, `+`, `-`, `*`.

Операции `=` и `<>` используются для проверки эквивалентности: два значения переменной типа `set` считаются равными, если они состоят из одних и тех же элементов.

Пример.

$[1, 3] = [3, 1]$ возвращает true,

$[1..3] = [1, 2, 3]$ возвращает true,

$[1] <> [2]$ возвращает true,

$[1, 2, 3] = [1, 4, 3]$ возвращает false,

$[red, blue] = [red, yellow]$ возвращает false.

Операции `>=` и `<=` используются для проверки принадлежности одного множества другому: так, если множество a содержится во множестве b , то $a <= b$ дает true.

Пример.

$[1, 2] <= [1, 2, 3]$ дает true

Пустое множество $[]$ содержится во всех множествах, т.е. всегда $[] <= [b]$ дает true.

Операция `in` используется для установления наличия определенного элемента в величине типа `set`. Так, если x есть элемент множества b , то $(x \text{ in } b)$ дает true. Общий вид:

$x \text{ in } a$;

здесь x – величина базового типа, a – величина типа `set`.

Пример.

`red in [red, yellow]` возвращает true;

`red in [blue, green]` возвращает false.

Замечание 1. Чтобы проверить, является ли значение n цифрой, удобно использовать операцию `in` следующим образом:

if n in $[0..9]$ then ...

Замечание 2. Результат операции in может быть неопределенным в некоторых случаях.

Пример. Пусть

a : set of $1..50$;

x : integer.

Если заслат в x число, большее максимального значения 50 (например, $x := 55$), то в этом случае результат операции $x \text{ in } a$ не всегда false.

К переменным типа `set`, относящимся к одному и тому же конкретному типу, применимы операции:

+ объединение;

* пересечение;

- дополнение.

Пусть a и b – операнды, имеющие один и тот же конкретный тип. Тогда $a + b$ представляет собой объединение множества элементов, входящих в a и b (одинаковые элементы не повторяются).

$a * b$ – пересечение множества элементов a и b (только те, которые есть в обоих множествах).

$a - b$ – множество элементов, которые есть в a , но отсутствуют в b .

Пример.

$[1, 3] + [1, 4] = [1, 3, 4]$;

$[1, 3] * [1, 4] = [1]$;

$[1, 3] - [1, 4] = [3]$.

Операция $a := a + x$ добавляет элемент x к множеству a . Если x уже имелся в a , то множество a не меняется. $a := a - x$ исключает x из a . Если x отсутствовал в a , то множество a не меняется.