Учебный курс

# Методы трансляции

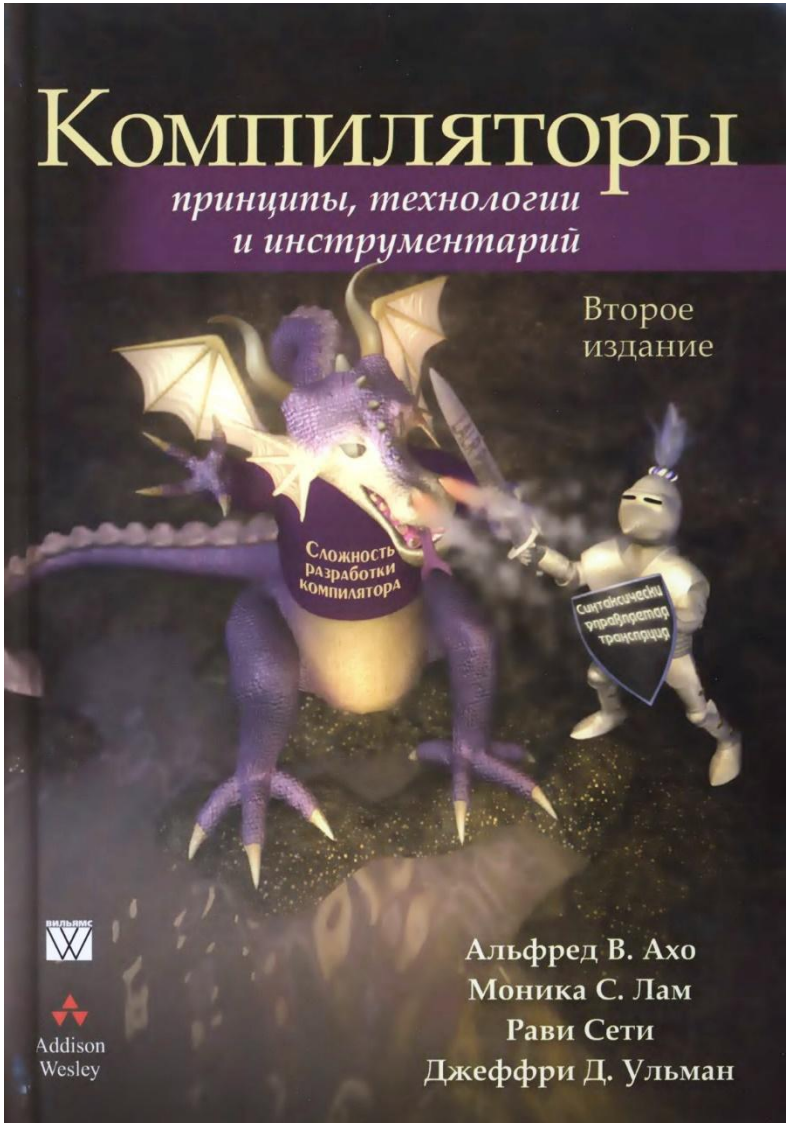
Языки и метаязыки.

Парадигмы языков программирования

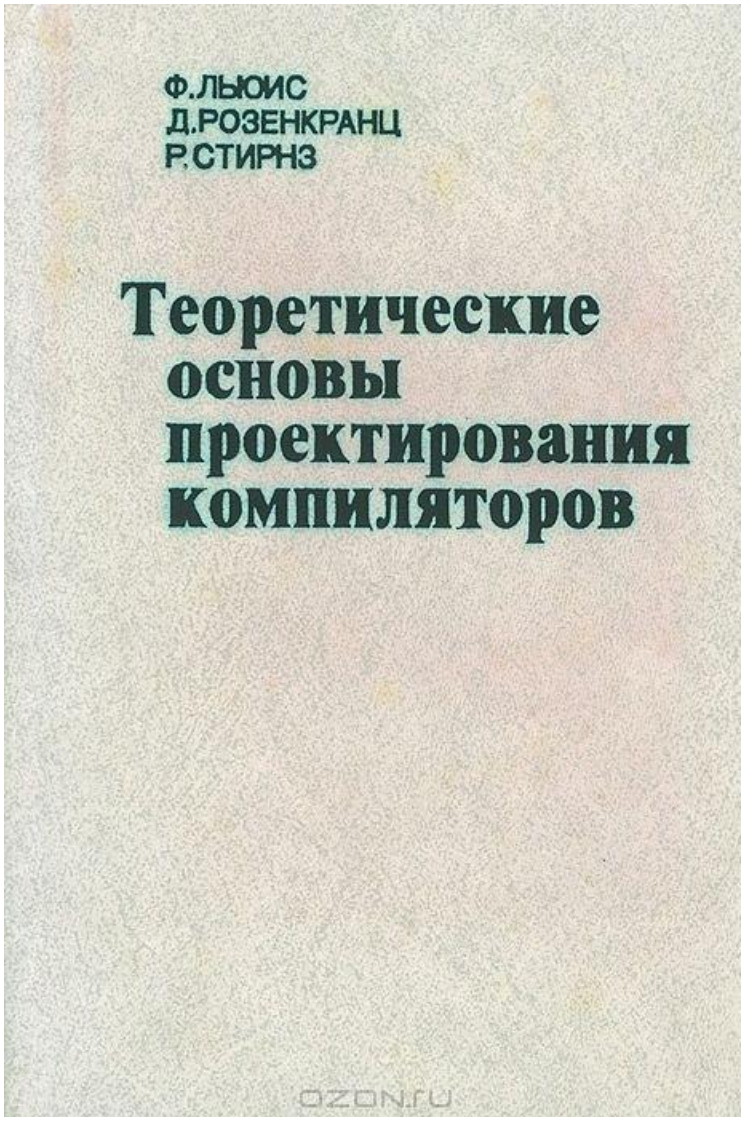
Возможности формального описания языка программирования

**Шиманский Валерий Владимирович**

# Литература

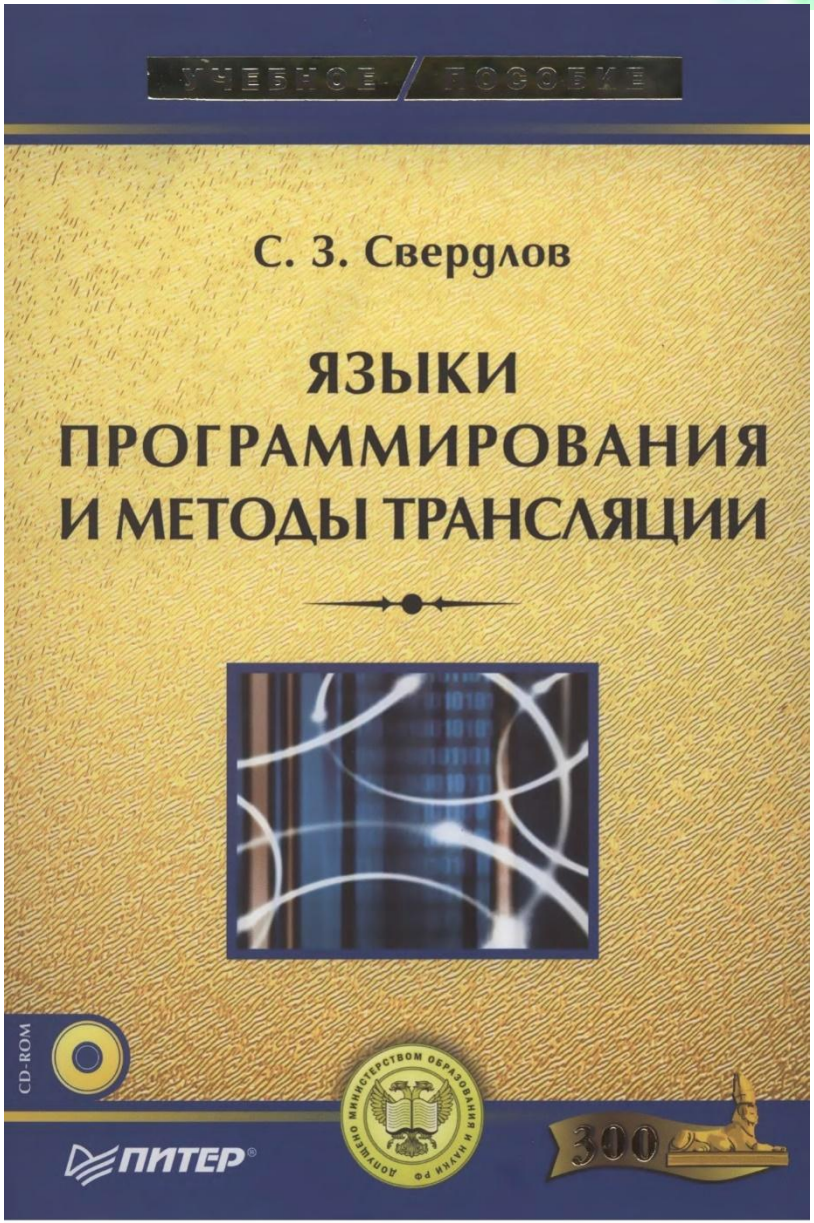



2-е изд. 2008 г.

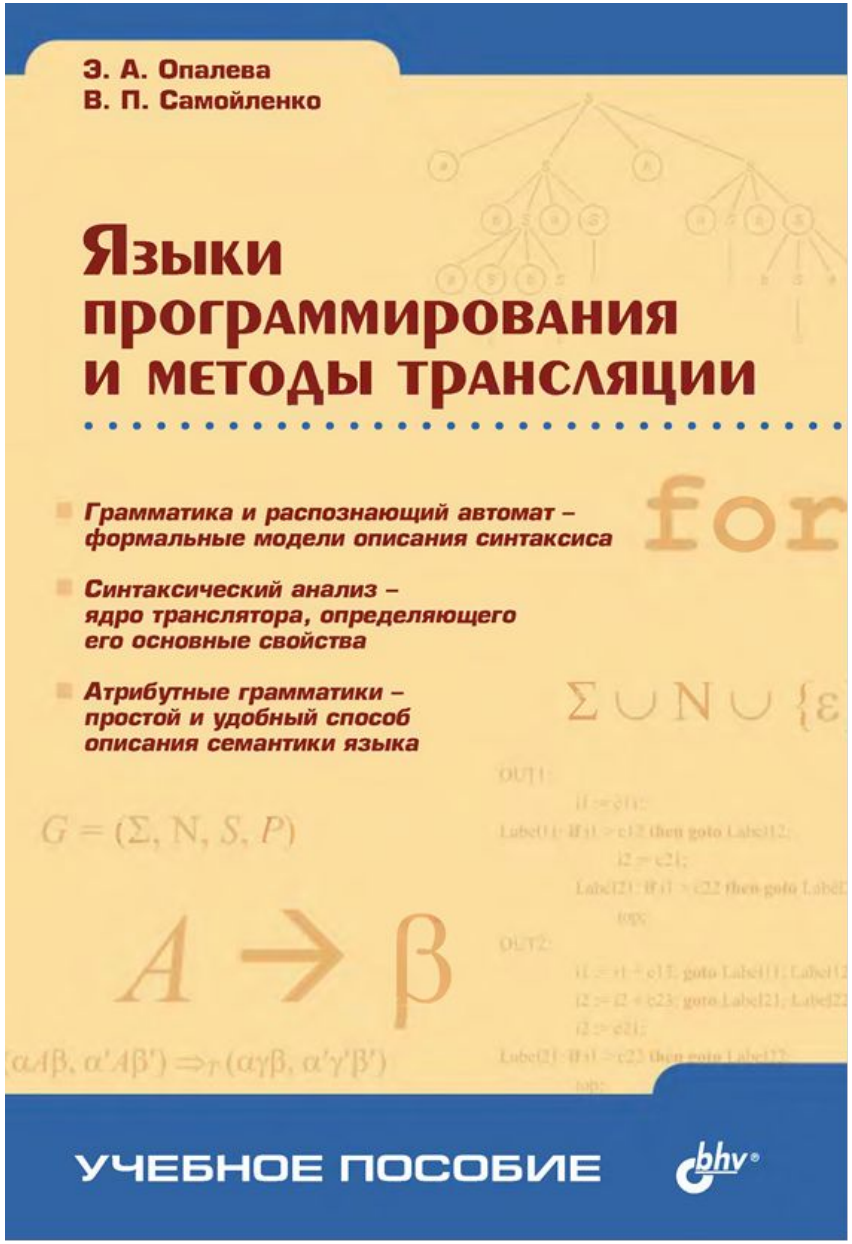



М. Мир 1979


М. ДМК-Пресс 2010




Учебное пособие Питер 2007




БХВ-Петербург 2005


Учебное пособие. Питер 2007

Альфред Тарский
(польск. Alfred Tarski)

**Метаязык** - язык, предназначенный для описания другого языка (национального, другой семиотической системы и т.д.). Впервые различение языка-объекта и метаязыка проведено **Давидом Гильбертом** (нем. David Hilbert; 1862-1943, немецкий математик-универсал), применительно к различению “математики” и “метаматематики” и без использования соответствующей терминологии.

Понятия «язык-объект» и «метаязык» были введены Альфредом Тарским и Рудольфом Карнапом в сер. 1930-х гг.

Понятие метаязыка используется:
- в лингвистике, при описании естественных языков;
- в математике и логике - при исследовании формальных языков исчислений;
- в информатике - метаданные, служащие для описания имеющихся

Мы будем понимать как средство изучения формализованных языков – логических и математических исчислений, или (в несколько иной формулировке) как формализованный или неформализованный язык, на котором формулируются утверждения метаматематики

**Парадокс лжеца** - утверждение «То, что я утверждаю сейчас - ложно»

Либо «Я лгу», либо «Данное высказывание — ложь». Если это высказывание истинно, значит, исходя из его содержания, верно то, что данное высказывание — ложь; но если оно — ложь, тогда то, что оно утверждает, неверно; значит, неверно, что данное высказывание — ложь, и, значит, данное высказывание истинно. Таким образом, цепочка рассуждений возвращается в начало.

Парадокс лжеца демонстрирует расхождение разговорной речи с формальной логикой, вводя высказывание, которое одновременно истинно и ложно.

Утверждение, составляющее парадокс лжеца, *в формальной логике не доказуемо и не опровержимо.*

*Поэтому считается, что данное высказывания вообще не является логическим утверждением.*

Пример Тарского: ***«Снег белый» - Утверждение из объектного языка, утверждение – «Снег белый истинно» - утверждение из метаязыка***

**Сумма внутренних углов любого треугольника равна 180°**

**Утверждение 1 истинно.**

**Утверждение 2 истинно.**

**Утверждение 3 истинно.**

Здесь первое утверждение написано на языке первого уровня, который позволяет формулировать теоремы планиметрии. Языком второго уровня (фраза № 2) пользуются при доказательстве теорем. Метаметаязык, которому принадлежит третье утверждение, — это язык, на котором написаны книги о теории

1. Языки программирования предназначены для облегчения программирования. Поэтому их операторы и структуры данных более мощные, чем в машинных языках.
2. Для повышения наглядности программ вместо числовых кодов используются символические или графические представления конструкций языка, более удобные для их восприятия человеком.
3. Для любого языка определяется:

- Множество символов, которые можно использовать для записи правильных программ (**алфавит**), основные элементы.
- Множество правильных конструкций программ (**синтаксис**).
- "Смысл" правильного блока конструкций программы (**семантика**).

Независимо от специфики языка **процесс трансляции можно считать функциональным преобразователем F, обеспечивающим однозначное отображение X в Y, где X - программа на исходном языке, Y - программа на выходном языке**. Поэтому сам процесс трансляции формально можно представить достаточно просто и понятно:

**Y = F(X)**

Формально каждая правильная программа X - это цепочка символов из некоторого алфавита **V**, преобразуемая в соответствующую ей цепочку Y, составленную из символов алфавита $\mathbf{V_1}$.

# Парадигмы языков программирования

Имеются четыре основные парадигмы языков программирования, отражающие вычислительные модели, с помощью которых описывается большинство существующих методов **программирования**:

- □ **императивная**;
- □ **функциональная**;
- □ **декларативная**;
- □ **объектно-ориентированная**.

***Императивные (процедурные) языки*** – это языки программирования, управляемые командами, или операторами языка.

В ***языках функционального программирования (аппликативных языках)*** вычисления в основном производятся путем применения функций к заданному набору данных.

$$функция_n\ (\ ...\ функция_2\ (функция_1\ (данные))\ ...\ )$$

**Программирование, как на императивных, так и на функциональных языках является *процедурным*.**

***Декларативные*** языки программирования – это языки программирования, в которых операторы представляют собой объявления или высказывания в символьной логике.

Концепция ***объектно-ориентированного программирования*** складывается из трех ключевых понятий: ***абстракция данных, наследование и полиморфизм***.

**Scripting language (язык сценариев) —**

высокоуровневый язык программирования для написания *сценариев* — кратких описаний выполняемых системой действий.

Сценарии обычно интерпретируются, а не компилируются.

По применению **сценарные** языки можно грубо разделить на 4 типа:

командно-сценарные;
прикладные сценарные;
языки разметки;
универсальные сценарные.

**Метаязык Хомского** имеет следующую систему обозначений:

- символ “®” отделяет левую часть правила от правой (читается как "порождает“ и "это есть");
- нетерминалы обозначаются буквой **A** с индексом, указывающим на его номер;
- терминалы - это символы используемые в описываемом языке;
- каждое правило определяет порождение одной новой цепочки, причем один и тот же нетерминал может встречаться в нескольких правилах слева.

Описание идентификатора на метаязыке Хомского будет выглядеть следующим образом:

| | | |
|---|---|---|
| **1. $A_1$ ® A** | **23. $A_1$ ® W** | **45. $A_2$ ® s** |
| **2. $A_1$ ® B** | 24. $A_1$ ® X | 46. $A_2$ ® t |
| **3. $A_1$ ® C** | 25. $A_1$ ® Y | 47. $A_2$ ® u |
| **….** | …. | ….. |
| **20. $A_1$ ® T** | 42. $A_2$ ® p | 64. $A_3$ ® $A_3A_1$ |
| **21. $A_1$ ® U** | 43. $A_2$ ® q | 65. $A_4$ ® $A_3A_2$ |

## Метаязык Хомского-Щутценберже

- символ “=” отделяет левую часть правила от правой (вместо символа “®”);
- нетерминалы обозначаются буквой A с индексом, указывающим на его номер;
- терминалы - это символы используемые в описываемом языке;
- каждое правило определяет порождение нескольких альтернативных цепочек, отделяемых друг от друга символом “+”, что позволяет, при желании, использовать в левой части только разные нетерминалы.

Введение возможности альтернативного перечисления позволило сократить описание языков. Описание идентификатора будет выглядеть следующим образом:

1. $A_1$=A+B+C+D+E+F+G+H+I+J+K+L+M+N+O+P+Q+R+S+T+U+V+W+X+Y+Z+a+b+c+d+e+f+g+h+i+j+k+l+m+n+o+p+q+r+s+t+u+v+w+x+y+z
2. $A_2$=0+1+2+4+5+6+7+8+9
3. $A_3=A_1+A_3A_1+A_3A_2$ ...

## Бэкуса-Наура формы (БНФ)

- металингвистическая связка "::=" отделяет левую часть правила от правой;
- металингвистические переменные обозначаются произвольной символьной строкой, заключенной в угловые скобки "<" и ">";
- терминальные символы (терминалы) - это символы, используемые в описываемом языке, в частности, ключевые слова;
- каждое правило определяет порождение нескольких альтернативных цепочек, отделяемых друг от друга металингвистической связкой - символом вертикальной черты "|".

Правила описания идентификатора с использованием БНФ:

- <буква> :: = A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z|a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z
- <цифра> :: = 0|1|2|3|4|5|6|7|8|9
- <идентификатор> ::= <буква> | <идентификатор><буква> |<идентификатор><цифра>

Правила можно задавать и раздельно:

- <идентификатор> :: = <буква>
- <идентификатор> :: = <идентификатор> <буква>
- <идентификатор> :: = <идентификатор> <цифра>

# Расширенная форма Бэкуса-Наура РБНФ (Augmented Backus-Naur Form ABNF)

Рассмотрим ее особенности на примере метаязыка Вирта для Модулы-2:

- Квадратные скобки "[" и "]" означают, что заключенная в них синтаксическая конструкция может отсутствовать.

- Фигурные скобки "{" и "}" означают ее повторение (возможно, 0 раз).

- Круглые скобки "(" и ")" используются для ограничения альтернативных конструкций.

- Сочетание фигурных скобок и косой черты "{/" и "/}" используется для обозначения повторения один и более раз.

# Синтаксические диаграммы Вирта

Предложены Никлаусом Виртом для описания синтаксиса языка Pascal и являются удобной графической формой представления РБНФ. Включают: прямоугольники, кружки или овалы, стрелки.

В прямоугольниках записываются имена металингвистических переменных, в кружках или овалах – основные символы языка, а стрелки определяют порядок сочетания металингвистических переменных и основных символов языка для образования определяемой конструкции языка.

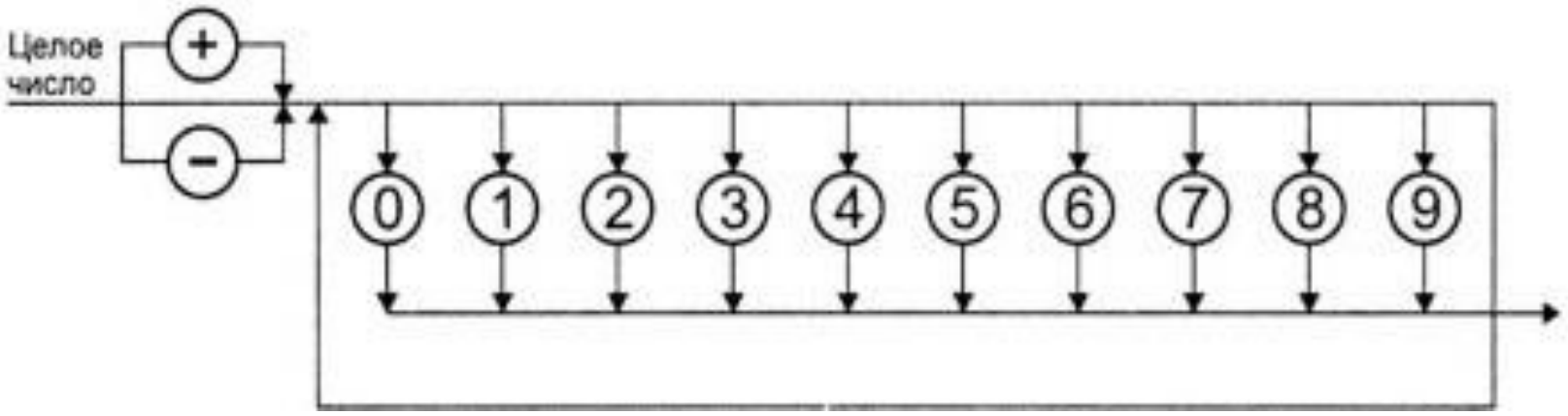
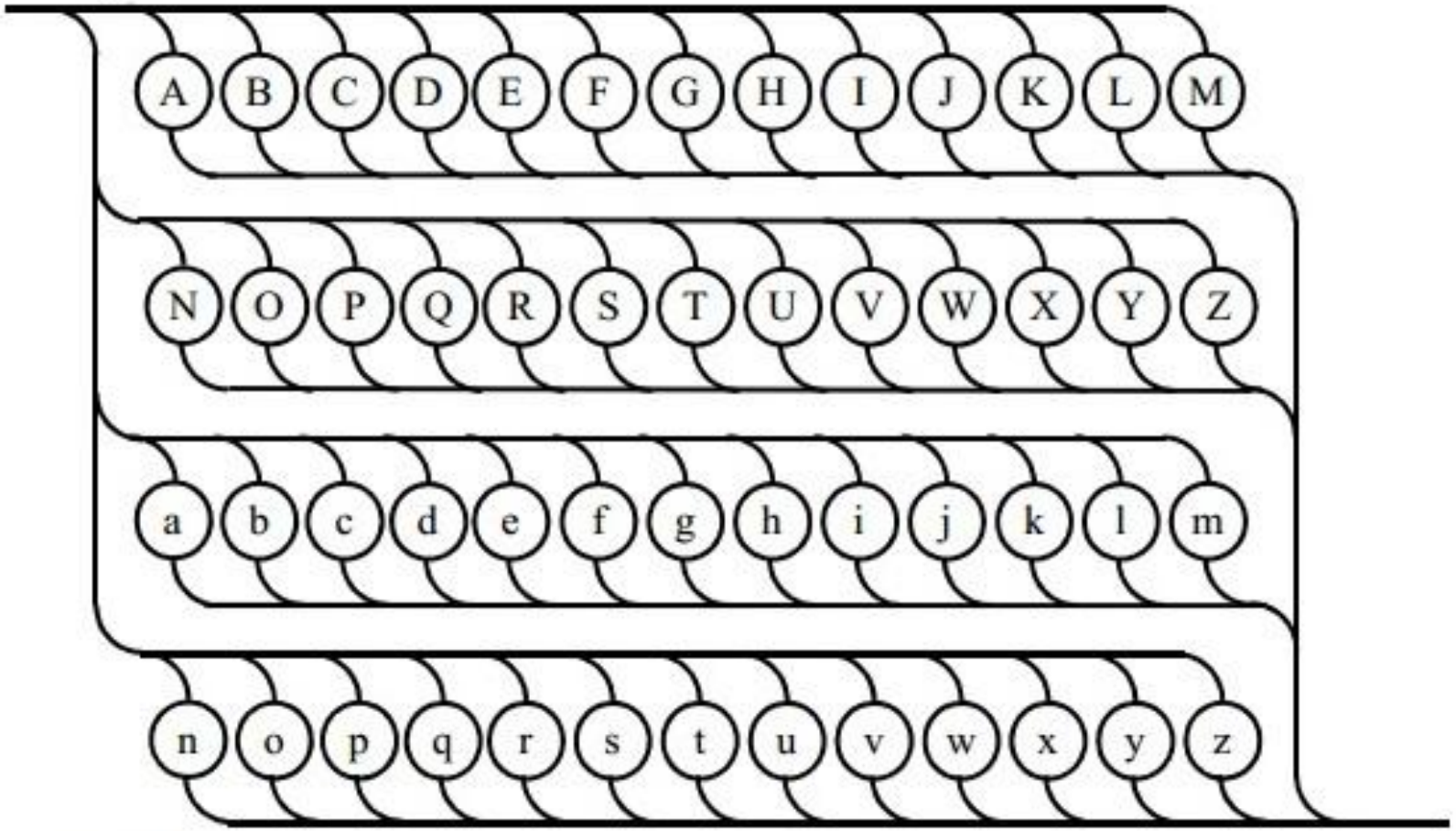
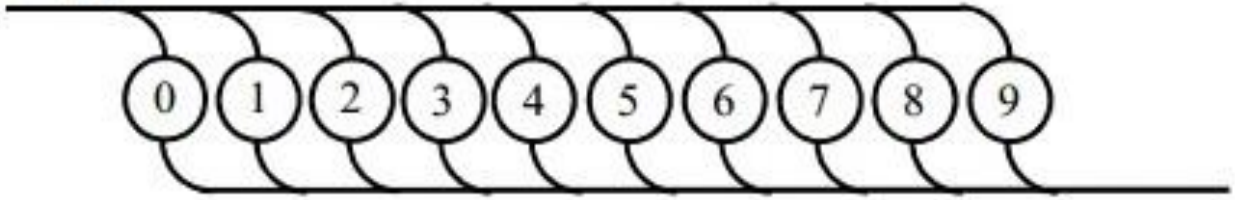


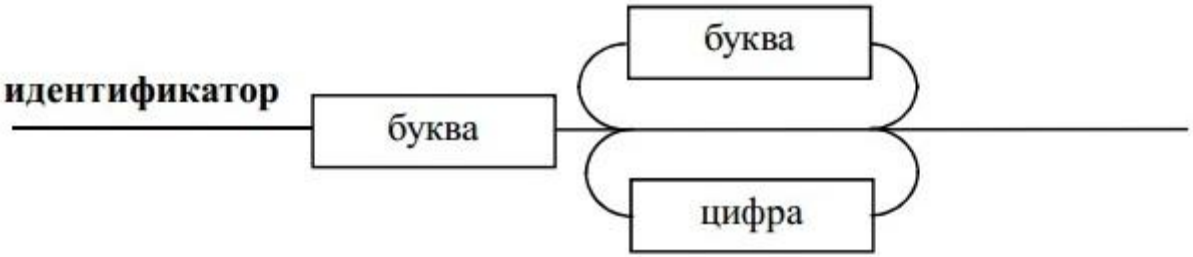

Рис. 1.2. Синтаксические диаграммы для определения множества целых чисел

**буква**


A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m
n o p q r s t u v w x y z

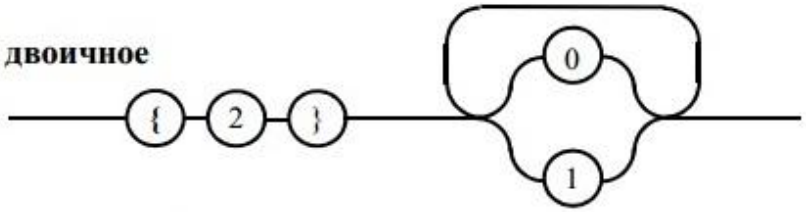
**цифра**

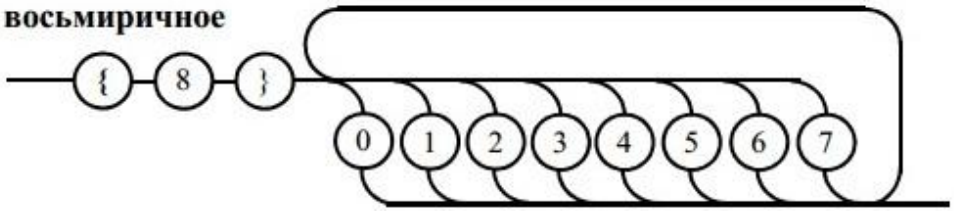

0 1 2 3 4 5 6 7 8 9

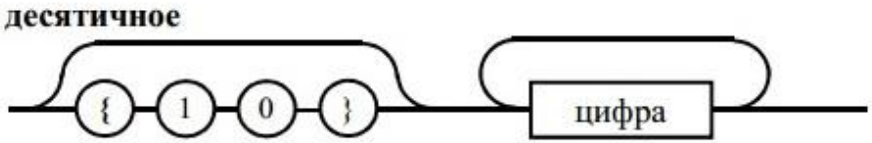

идентификатор
буква
буква
цифра

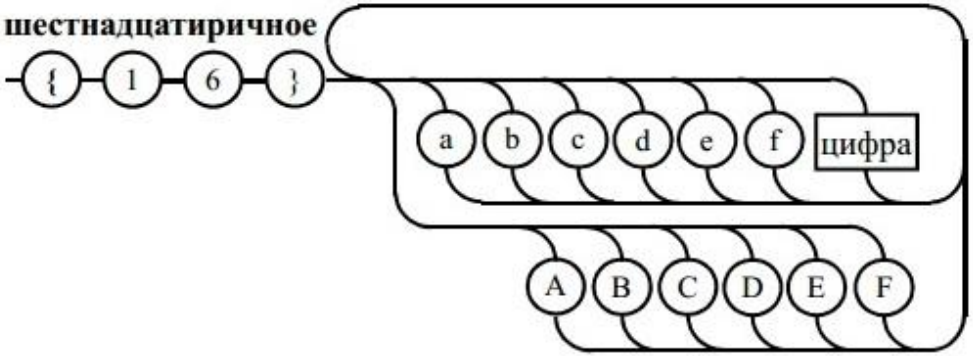

число
целое
действительное

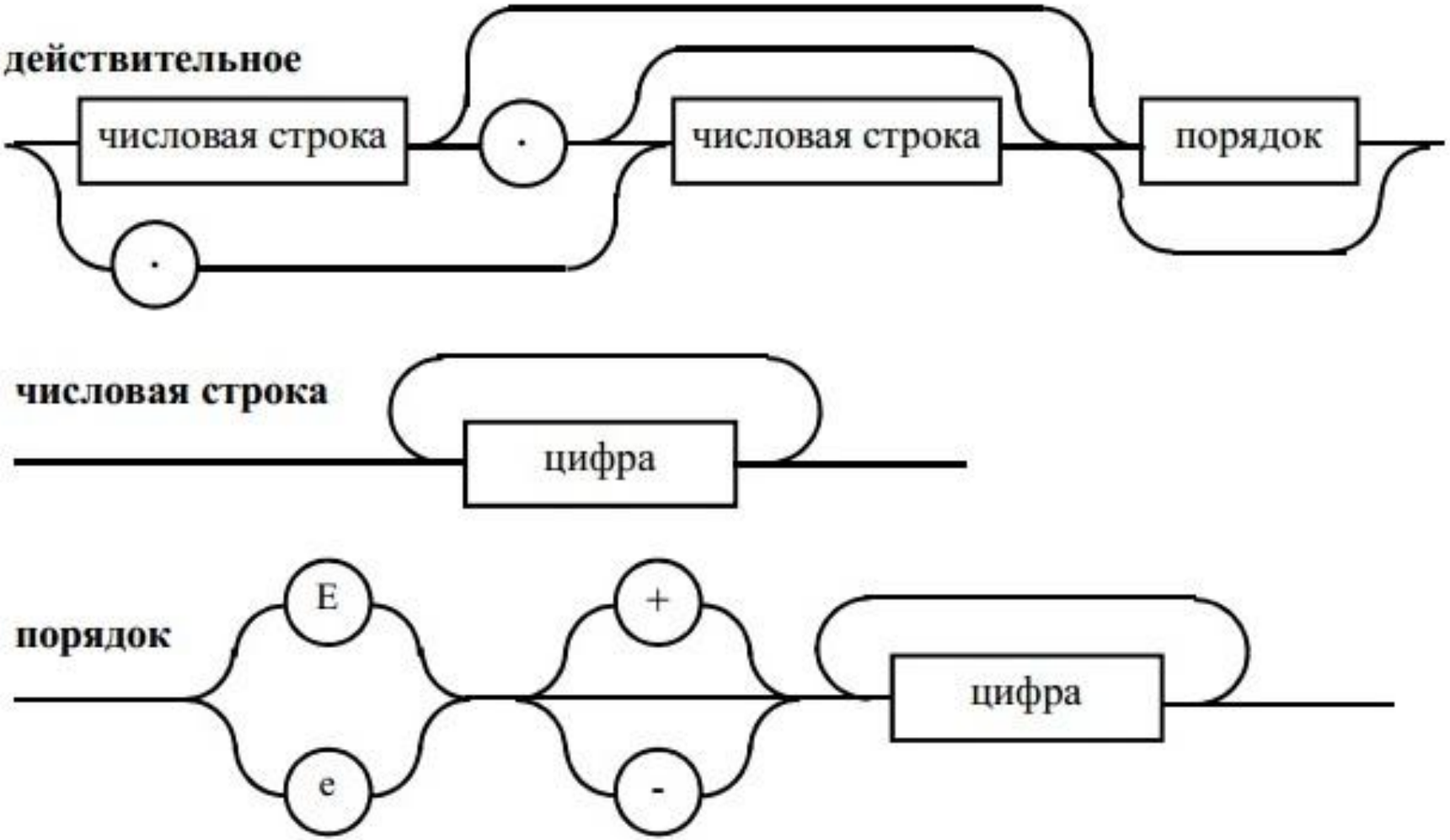

целое
двоичное
восьмиричное
десятичное
шестнадцатиричное


двоичное
{
2
}
0
1

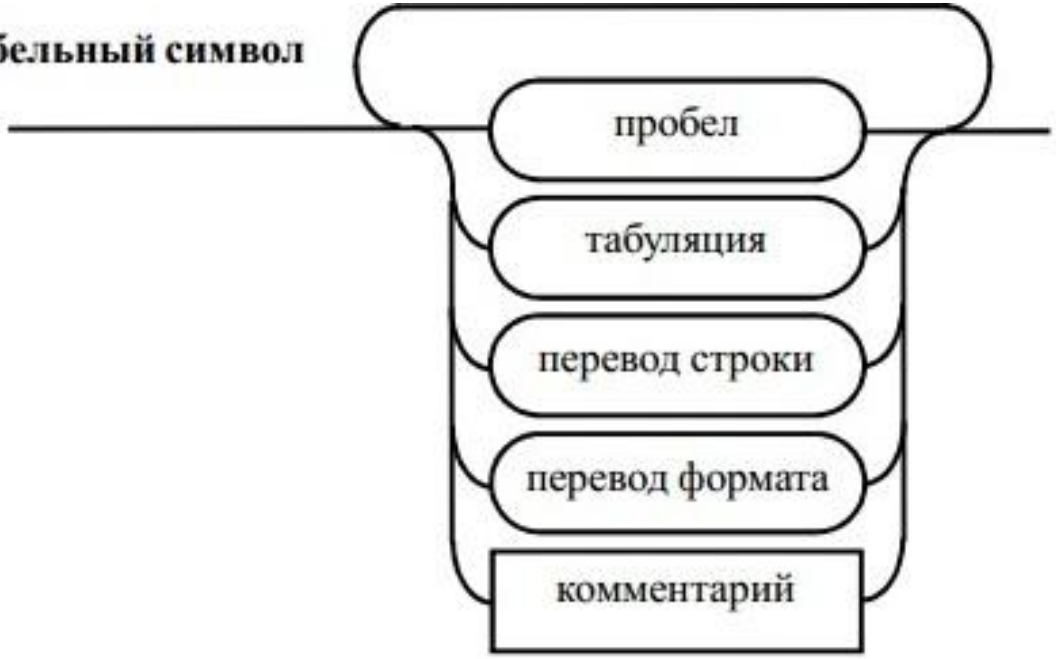

восьмиричное
{
8
}
0
1
2
3
4
5
6
7


десятичное
{
1
0
}
цифра

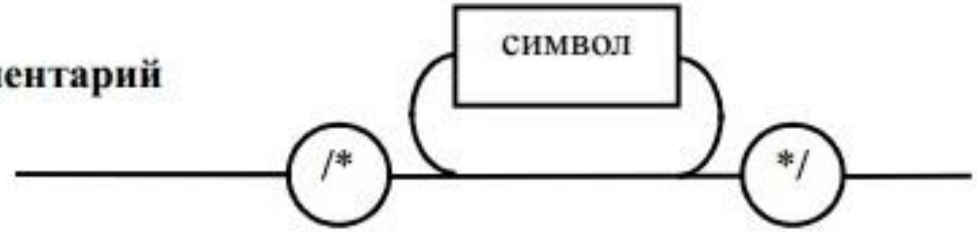

шестнадцатиричное
{
1
6
}
a
b
c
d
e
f
цифра
A
B
C
D
E
F


действительное
числовая строка
.
числовая строка
порядок
.
числовая строка
цифра
порядок
E
e
+
-
цифра

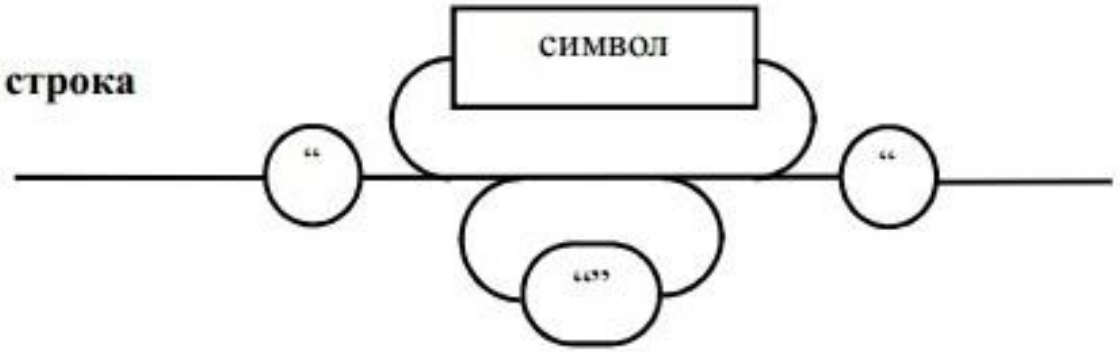
**Пробельный символ**



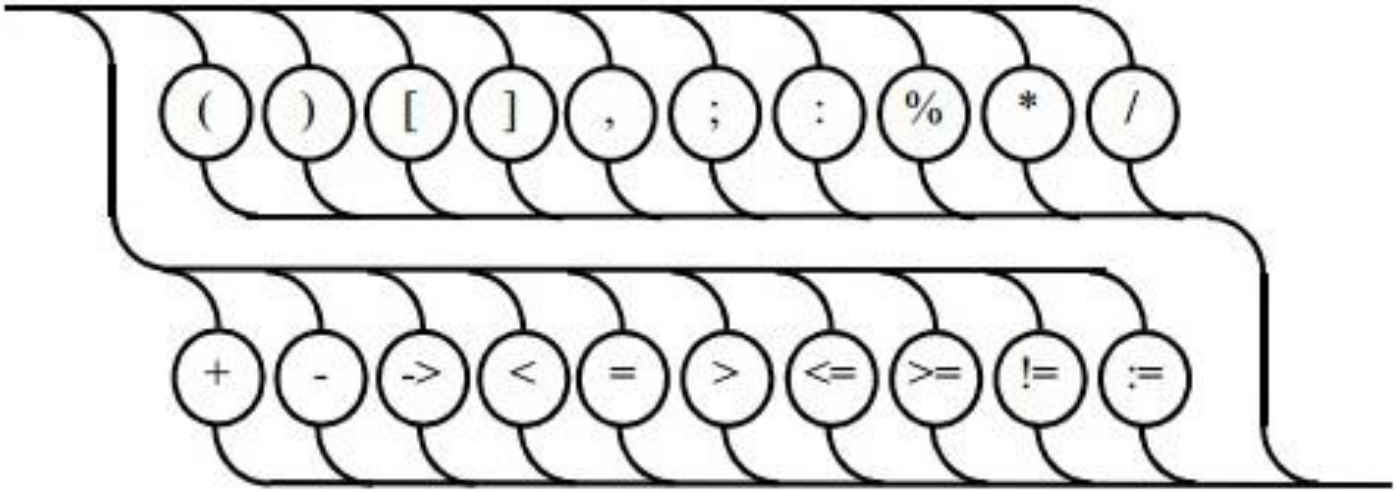

**комментарий**



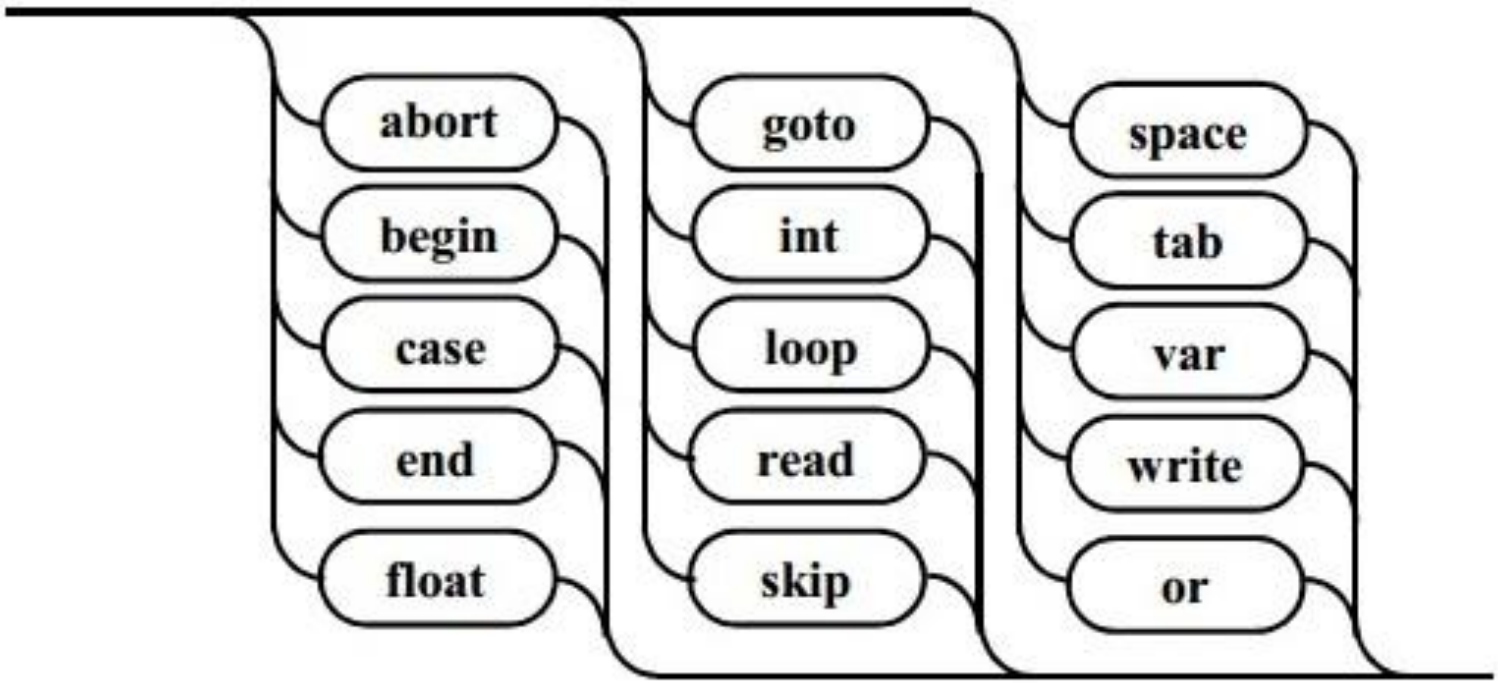

**строка**



**разделитель**




**ключевое слово**



*Если* A - множество, то его *элементы* – это есть объекты *a*, для которых $a \in A$. Знак $\in$ означает принадлежность множеству A. Итак, $A=\{a_1, a_2, \ldots a_n, \}$ – множество, $a_i \in A$ *элемент* множество. *Отрицание этого утверждения записывается* $a \notin A$. *Если* A содержит конечное число элементов, то A называется *конечным* множеством. Символ  обозначает пустое множество, т.е. множество, в котором нет элементов.

Диаграмма Венна

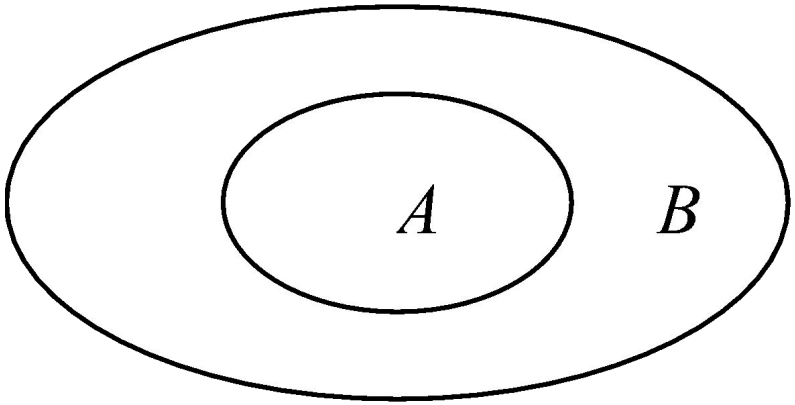



Диаграмма Венна для включения множеств

$A \cup B = \{x \mid x \in A \text{ или } x \in B\}$ -
это множество, содержащее все элементы A и B

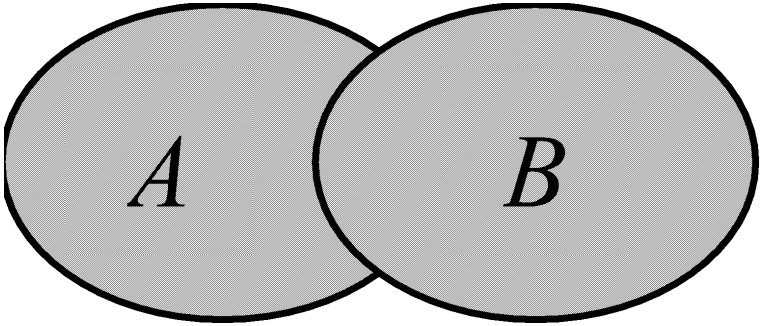



*Диаграмма Венна для объединения множеств*

$A \cap B = \{x \mid x \in A \text{ и } x \in B\}$ -
это множество, состоящее из всех тех элементов, которые принадлежат обоим множествам *A* и *B*

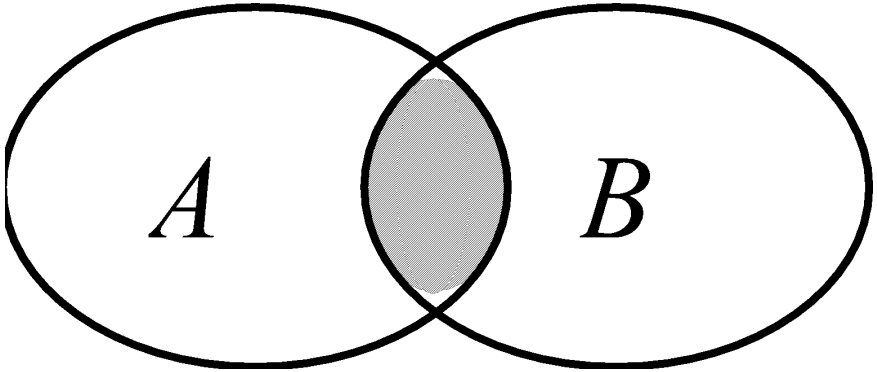



*Диаграмма Венна для* пересечений *множеств*

A – B -
это множество элементов A, не принадлежащих B

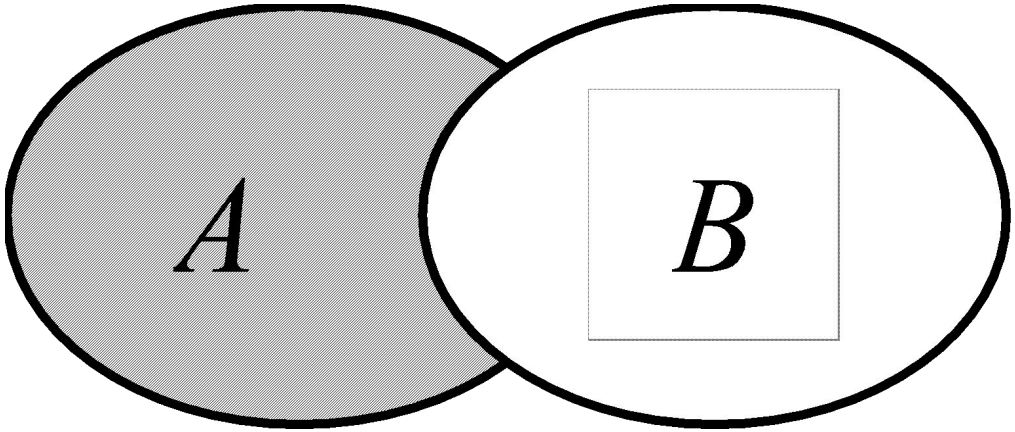



Диаграмма Венна для разности множеств

*Декартово произведение A и B*
$A \times B = \{(a,b| \ a \in A \ и \ b \in B\}$

*Пример.*
Если $A=\{1,2\}$, $B=\{2,3,4\}$,
то $A \times B=\{(1,2), (1,3), (1,4), (2,2), (2,3), (2,4)\}$

**Отношением из *A* в *B* называется любое подмножество множеств $A \times B$.**

Если $A=B$, то отношение задано (определено) на $A$.

Если $R$ отношение из $A$ в $B$ и $(a, b) \in R$, то пишут ***aRb***.

Множество $A$ называют **областью определения**, $B$ **- множеством значений**.

Пусть $A$ – множество, $R$ – отношение на $A$.
Тогда $R$ называют:

- ***рефлексивным***, если $aRa$ для всех пар из $A$;
- ***симметричным***, если $aRb$ влечет $bRa$ для всех $a$ и $b$ из $A$;
- ***транзитивным***, если $aRb$ и $bRc$ влекут $aRc$ для $a, b, c$ из $A$.

Рефлексивное, симметричное и транзитивное отношение называют отношением ***эквивалентности***.

Отношение эквивалентности, определенное на $A$, заключается в том, что оно разбивает множество $A$ на непересекающиеся подмножества, называемые **классами эквивалентности**.

Рассмотрим отношение сравнения по модулю $N$, определенное на множестве неотрицательных чисел.

$a$ сравнимо с $b$ по модулю $N$

a=b(modN), т.е. $a-b=kN$ ($k$ - целое)

Пусть $N$=3, тогда множество {0, 3, 6,…, 3$n$,…} будет одним из классов эквивалентности, т.к. $3n=3m(mod3)$ для целых $n$ и $m$. Обозначим его через [0].

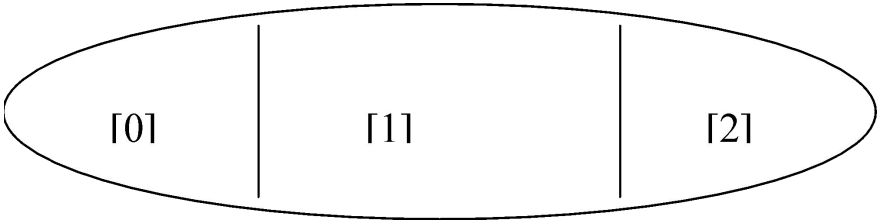
[0]={0, 3, 6,…, 3$n$,…}

Другие два:

[1]={1, 4, 7,…, 3$n$+1,…};

[2]={2, 5, 8,…, 3$n$+2,…}.

Объединение этих трех множеств дает множество неотрицательных целых чисел




Классы эквивалентности отношения сравнения по модулю 3

# Замыкание отношений

k – степень отношения R на A ($R^k$ ) определяется:

1) $aR^1b$ тогда и только тогда, когда aRb;
2) $aR^ib$ для i>1 тогда и только тогда, когда существует такое $c \in A$, что aRc и $cR^{i-1}$

Транзитивное замыкание отношения множества R на A ($R^+$) определяется так: $aR^+b$ тогда и только тогда, когда $aR^ib$ для некоторого $i \geq 1$.

Расшифровка понятия:

$aR^+b$, если существует последовательность $c_1, c_2, \ldots, c_n$, состоящая из 0 или более элементов принадлежащих A, такая, что $aRc_1$, $aRc_2, \ldots aRc_{n-1}, aRc_n, c_nRb$ . Если n=0, то aRb.

Рефлексивное и транзитивное замыкание отношения R (R*) на множестве A определяется следующим образом:

1) $aR^*a$ для всех $a \in A$;
2) $aR^*b$, если $aR+b$;
3) в R* нет ничего другого, кроме того, что содержится в 1) и 2).

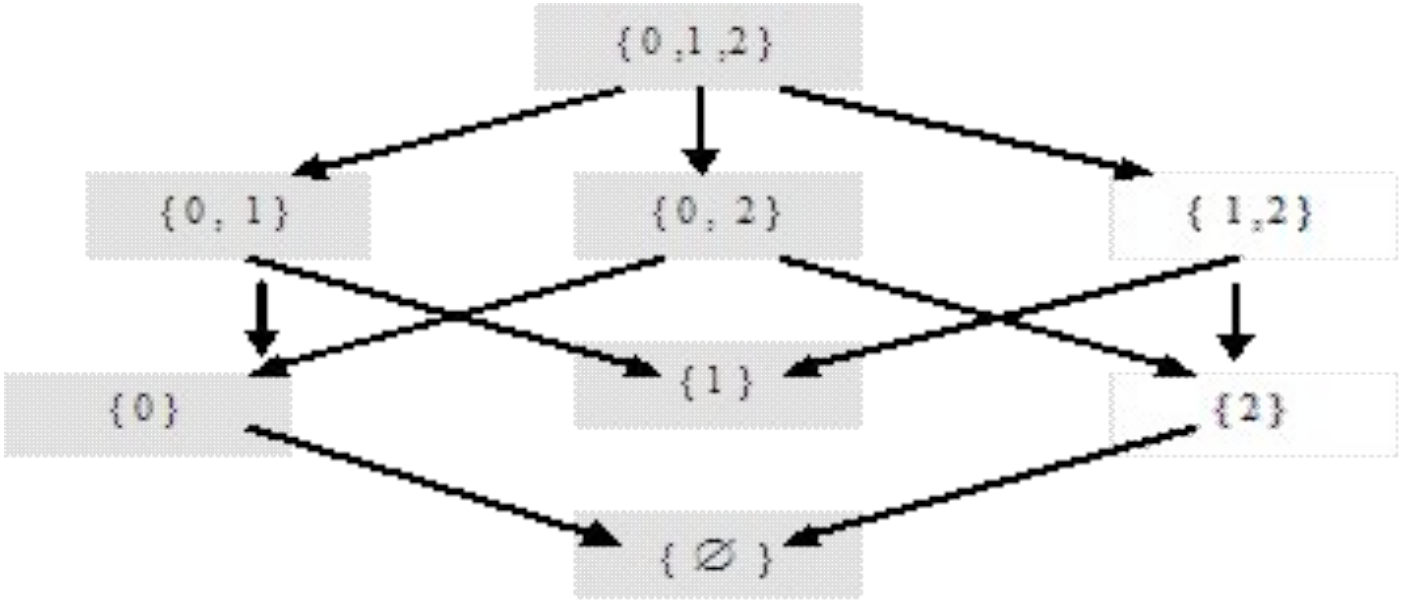
**Частичным порядком** на множестве *A* называют отношение *R* на *A* такое, что:

1) *R* – транзитивно;
2) для всех $a \in A$ утверждение *aRa* ложно, т.е. отношение *R* иррефлексивно.

*Пример.*

S= {$e_1$, $e_2$, …, $e_n$}, - множество, состоящее из *n* элементов, и пусть *A*=*P*(*S*). Положим *aRb* для любых *a* и *b* из *A* тогда и только тогда, когда $a \not\subset b$. Отношение *R* называется частичным порядком.

Для случая *S*={0, 1, 2} имеем



**Рефлексивным частичным порядком** называется отношение $R$, когда

1) $R$ – транзитивно;
2) $R$ – рефлексивно;
3) если $aRb$, то $a=b$.

Последнее свойство называется *антисимметричностью.*

Каждый частичный порядок можно графически представить в виде ориентированного ациклического графа.

Линейный порядок $R$ на множестве $A$ – это такой частичный порядок, что, если $a$ и $b \in A$, то либо $aRb$, либо $bRa$, либо $a=b$. Удобно это понять из следующего.

Пусть $A$ представлено в виде последовательности $a_1, a_2, \ldots, a_n$, для которых $a_i R a_j$ тогда и только тогда, когда $i<j$.

Аналогично определяется рефлексивный линейный порядок.

Из традиционных систем отношение < (меньше) на множестве неотрицательных целых чисел – это линейный порядок, отношение ≤ - рефлексивный линейный порядок.