Лекция 12
БГТУ, ФИТ, ПОИТ, 2 семестр, Языки программирования

Основы теории формальных языков

Понятие языка, синтаксис, семантика. Формальные и естественные языки. Языки программирования

1. ОСНОВНЫЕ ПОНЯТИЯ

Формальный язык - это множество конечных слов над конечным алфавитом, например, язык программирования.

Лексика языка программирования – правила составления слов программы из символов языка (идентификаторы, константы, служебные слова, комментарии).

Синтаксис языка – система правил, определяющих допустимые конструкции языка программирования из слов языка (построение, порядок, составление).

Семантика (смысловое значение) – смысл, который закладывается в каждую конструкцию языка.

Алфавит – конечное непустое множество элементов языка.
Пример: $V = \{a,b\}$ – алфавит V, состоящий из двух символов a и b.

Цепочка – конечная последовательность символов языка.
Пример: ***abc*** — цепочка из трех символов.

2. СОГЛАШЕНИЯ

Алфавит будем обозначать заглавными буквами латинского алфавита:
$I, V, G, \ldots$

Символы алфавита будем обозначать строчными буквами латинского алфавита:
$a, b, c, \ldots$

Цепочки будем обозначать символами греческого алфавита:
$\alpha, \beta, \gamma, \delta, \lambda, \varepsilon, \omega, \ldots$
Пример: $\alpha = a_1 \ldots a_n$ — цепочка из n символов.

Пустую цепочку символов будем обозначать λ, ε.

3. ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

1). **Алфавитом** V называется конечное множество символов.

Пример: $V = \{a, b, c\}$, $I = \{a^n b^n\}$,

где n — натуральное число.

Алфавит задает язык *I*, состоящий из цепочек вида *ab, aabb, aaabbb* и т.д. Язык *I* представляет собой бесконечное множество цепочек, но его описание состоит всего из 8 символов, т.е. конечно.

2). **Цепочкой** α в алфавите I называется любая конечная последовательность символов этого алфавита.
Цепочка, которая не содержит ни одного символа, называется пустой цепочкой и обычно обозначается ε или λ.

Формальное определение цепочки символов в алфавите *I*:

a) ε – пустая цепочка в алфавите *I*;

b) если α – цепочка в алфавите I и a – символ этого алфавита, то αa – цепочка в алфавите *I*;

c) β – цепочка в алфавите I тогда и только тогда, когда она является таковой в силу утверждений a) и b).

Пример: $\alpha = abc$, $\beta = aaaa$, $\gamma = abcaaaa$.

3). **Длиной цепочки** α (обозначается $|\alpha|$) называется число составляющих ее символов.

Пример: $|\alpha| = 3$, $|\beta| = 4$, $|\gamma| = 7$. Длина пустой цепочки $|\varepsilon|=0$.

4). **Конкатенацией** (сцеплением) цепочек α и β называется цепочка $\gamma=\alpha\beta$, в которой символы данных цепочек записаны друг за другом.

Пример: $\alpha = abc$, $\beta = aaaa$, $\alpha\beta = abcaaaa$.

5). Для любой цепочки α справедливо утверждение $\alpha\varepsilon=\varepsilon\alpha$.

6). α^n называется **итерацией цепочки** α.

Справедливы следующие утверждения:

$\alpha^0 = \varepsilon$

$\alpha^n = \alpha^{n-1}\alpha = \alpha\alpha^{n-1}$ для $n \geq 1$.

Пример: $\alpha = ab$, $\alpha^3 = (ab)^3 = ababab$, $\alpha^0 = \varepsilon$

7). Цепочки α и β **равны** ($\alpha = \beta$), если они имеют один и тот же состав символов, одно и тоже количество символов $|\alpha| = |\beta|$ и тот же порядок следования символов.

8). **Реверсом** (обращением) цепочки α называется цепочка α^R, составленная из символов цепочки α, записанных в обратном порядке.

Пример. Пусть алфавит I = {a, b, c, d}, тогда для цепочек этого алфавита $\alpha=ab$ и $\beta=bcd$ будет справедливо $|\alpha|=2$, $|\beta|=3$, $\alpha\beta= abbcd$, $\alpha^2=abab$, $\beta^R=dcb$.

9). Пусть I – алфавит, тогда I^+ – множество всех цепочек, состоящих из символов алфавита I, исключая пустую цепочку (λ).

$$\boxed{\lambda \notin I^+}$$

10). Пусть I – алфавит, тогда I^* – множество всех цепочек, состоящих из символов алфавита I, включая пустую цепочку.

$$\boxed{\lambda \in I^*} \qquad \boxed{I^* = I^+ \cup \lambda}$$

11). **Языком *L(I)*** над алфавитом I называется произвольное множество цепочек из I^*, $L(I) \subseteq I^*$.

Пример: $I = \{a, b, c\}$,

$L_1(I) = \{ab, ac, bc\}$, $L_2(I) = \{a, b, c\}$, $L_3(I) = \{aa, bb, cc\}$, …

12). Язык $L_1(I)$ является подмножеством языка $L_2(I)$, если каждая цепочка, входящая в язык L_1, входит в язык L_2; язык L_2 включает язык L_1.

$$\boxed{L_1(I) \subseteq L_2(I) \Leftrightarrow (\forall \alpha \in L_1(I) \Rightarrow \alpha \in L_2(I))}$$

13). Языки $L_1(I)$ и $L_2(I)$ совпадают (эквивалентны), если язык $L_1(I)$ включает язык $L_2(I)$ и язык $L_2(I)$ включает язык $L_1(I)$.

$$\boxed{L_1(I) = L_2(I) \Leftrightarrow (L_1(I) \subseteq L_2(I) \wedge L_2(I) \subseteq L_1(I)).}$$

4. ФОРМАЛЬНЫЕ ГРАММАТИКИ. ОСНОВНЫЕ ПОНЯТИЯ.

Язык L можно определить тремя способами:

a) перечислением всех цепочек языка;
b) указанием способа (алгоритма) порождения цепочек;
c) определением метода (алгоритма) распознавания цепочек.

a) язык L может быть задан с помощью перечисления цепочек, если количество цепочек конечно.

b) способ порождения (задания) цепочек языка L называется грамматикой языка L.

   Например, грамматика языка
   $L = \{a^n b^n\}$, где n — натуральное число, задает язык, состоящий из цепочек вида ab, aabb, aaabbb и т.д.

c) алгоритм (программа), распознающий цепочки языка L, называется распознавателем.

Пример: $I = \{0,1\}$, $L_1(I) = \{0,1,00,01,10,11\}$, $L_2(I) = \{0^n 1^n, n > 0\}$, тогда распознаватель L_2 – алгоритм (программа), проверяющий, что цепочка начинается с 0 и содержит одинаковое количество 0 и 1.

Лексика языка программирования – множество цепочек языка.
Синтаксис языка – набор формальных правил, определяющий конструкции (последовательности цепочек) языка.
Семантика языка – набор неформальных правил, которые описываются словесно (например, в руководстве программиста). *Пример*: переменную надо объявить до ее применения.

Чтобы создать язык программирования, следует определить:

- множество допустимых символов (алфавит);
- формально описать множество правильных программ;
- задать семантические правила языка.

Множество допустимых символов может быть проверено.

Определить множество формально правильных программ можно с помощью алгоритма-распознавателя. Распознаватель строится на основе формального описания языка – его формальной грамматики.

Семантические правила могут быть реализованы в виде эвристических алгоритмов (алгоритм, не имеющий строгого обоснования, но дающий приемлемое решение) или в виде словесного (неформального) описания правил языка.

5. ОПРЕДЕЛЕНИЕ ПОРОЖДАЮЩЕЙ ГРАММАТИКИ.

$G = \langle T, N, P, S \rangle$ – грамматика языка (порождающая грамматика) –
это четверка, где:
T – множество терминальных символов,
N – множество нетерминальных символов,
P – множество правил (продукций) грамматики,
S – начальный символ грамматики.

T – множество терминальных символов (терминалы, алфавит языка) – это символы языка, определяемые грамматикой. Терминалы будем обозначать строчными символами.
N – множество нетерминальных символов (нетерминалы) – символы, применяемые в продукциях P (символы, определяющие слова, понятия, конструкции языка). $\boxed{N \cap T = \varnothing}$
Нетерминалы будем обозначать прописными буквами.
P – множество правил вида $\alpha \to \beta$ говорят, α порождает β,
где $\alpha \in (N \cup T)^+$, $\beta \in (N \cup T)^*$.
S – стартовый символ грамматики, $S \in N$.

> *Пример*: $G = \langle \{a, b, +\}, \{S, A, B\}, P, S \rangle$,
> где правила $P = \{S \to A + B, S \to B + A, A \to a, B \to b\}$.
> Порождаемый язык $L = \{a + b, b + a\}$

$V = N \cup T$ – **словарь** грамматики $G = \langle T, N, P, S \rangle$.

6. ЦЕПОЧКИ ВЫВОДА.

Выводом называется процесс порождения предложения языка на основе правил грамматики языка. Записывается $\alpha \Rightarrow \beta$

Цепочка $\beta = \delta_1\gamma\delta_2$ называется непосредственно выводимой из цепочки $\alpha = \delta_1\omega\delta_2$ в грамматике $G = \langle T, N, P, S \rangle$,

где $V = N \cup T$, δ_1, γ, $\delta_2 \in V^*$, $\omega \in V^+$,

если в грамматике существует правило: $\omega \to \gamma$.

Цепочка β называется выводимой из цепочки α, если выполняется одно из двух условий: - β непосредственно выводима из α (α→β); - существует γ такая, что γ выводима из α, и β непосредственно выводима из γ (α→γ, γ→β).

Пример: Грамматика для языка целых десятичных чисел со знаком:

G ({0,1,2,3,4,5,6,7,8,9,–,+}, {S, T, F}, P, S)

Правила P:

S→T | +T | –T

T→F | TF

F → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Построим несколько цепочек вывода в этой грамматике:

1. S –T –TF –TFF –FFF –4FF –47F –479
2. S T TF T8 F8 18
3. T TF T0 TF0 T50 F50 350
4. F 5

Получаем, что: S → –479, S → 18, T → 350, F → 5.

Пример:

$G = \langle \{a, b, +\}, \{S, A, B\}, \{S \to A + B, S \to B + A, A \to a, B \to b\}, S \rangle$,

$S \Rightarrow A + B$,

$A + B \Rightarrow a + B$,

$a + B \Rightarrow a + b$,

$A + B \Rightarrow A + b$,

$A + b \Rightarrow a + b$,

$S \Rightarrow B + A$,

$B + A \Rightarrow b + A$,

$b + A \Rightarrow b + a$.

Порождаемый язык $L = \{a + b, b + a\}$

7. СЕНТЕНЦИАЛЬНАЯ ФОРМА ГРАММАТИКИ.

Вывод называется ***законченным*** (или ***конечным),*** если на основе цепочки β, полученной в результате этого вывода (нельзя больше сделать ни одного шага вывода).

β называется ***сентенциальной формой*** грамматики $G = \langle T, N, P, S \rangle$,

если $S \Rightarrow^* \beta$ и $\beta \in (T \cup N)^*$.

Если $S \Rightarrow^* \beta$ и $\beta \in T^*$, то β называется ***терминальной сентенциальной*** формой грамматики $G = \langle T, N, P, S \rangle$.

Тогда цепочки –479 и 18 являются конечными сентенциальными формами грамматики целых десятичных чисел со знаком, так как существуют выводы: $S \Rightarrow^* –479$, $S \Rightarrow^* 18$

8. ЛЕВОСТОРОННИЙ И ПРАВОСТОРОННИЙ ВЫВОДЫ

Вывод называется ***левосторонним,*** если в нем на каждом шаге вывода правило грамматики применяется к крайнему левому нетерминальному символу в цепочке.

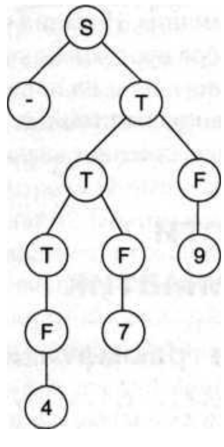
Вывод называется правосторонним, если в нем на каждом шаге вывода правило грамматики применяется всегда к крайнему правому нетерминальному символу в цепочке.

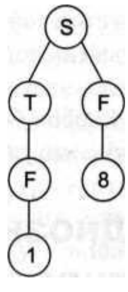
9. ДЕРЕВО ВЫВОДА

Деревом вывода грамматики G(T,N,P,S) называется дерево (граф), которое соответствует некоторой цепочке вывода и удовлетворяет следующим условиям:

- каждая вершина дерева обозначается символом грамматики $A \in (\mathbf{T} \cup \mathbf{N} \cup \{\varepsilon\})$;
- корнем дерева является вершина, обозначенная целевым символом грамматики – S;
- листьями дерева (концевыми вершинами) являются вершины, обозначенные терминальными символами грамматики или символом пустой цепочки;
- если некоторый узел дерева обозначен нетерминальным символом $A \in \mathbf{N}$, а связанные с ним узлы — символами b_1, b_2 … b_n; $n > 0$, $n \geq i > 0$: $b_i \in (\mathbf{T} \cup \mathbf{N} \cup \{\varepsilon\})$, то в грамматике G(T,N,P,S) существует правило $A \rightarrow b_1, b_2 \dots b_n$.

Дерево вывода цепочек –479 и 18:





1). Цепочка $\beta \in (N \cup T)^*$ ***выводима*** из цепочки $\alpha \in (N \cup T)^+$ в грамматике $G = \langle T, N, P, S \rangle$, если существуют цепочки, такие что $\alpha \Rightarrow \gamma_0 \Rightarrow \gamma_1 \Rightarrow \gamma_2 \Rightarrow ... \Rightarrow \gamma_n \Rightarrow \beta$. Записывается как $\alpha \Rightarrow^* \beta$.

Тогда последовательность $\alpha, \gamma_0, \gamma_1, \gamma_2, \gamma_n, \beta$ называется ***выводом*** длины n.

Пример: $G = \langle \{0,1\}, \{S, A\}, \{S \to 0A1, 0A \to 00A1, A \to \lambda\}, S \rangle$
Вывод $S \Rightarrow 0A1 \Rightarrow 00A1 \Rightarrow 000A1$
Длина вывода цепочки $000A1$ из стартового символа S равна 2.
(Запись второго правила рекурсивна).

2). Запись $\alpha \Rightarrow^* \beta$ предполагает $n \geq 0$ шагов вывода β из α.
В том случае, если $\alpha \Rightarrow \beta$, то число шагов вывода $n = 0$.
Запись $\alpha \Rightarrow^+ \beta$ предполагает $n > 0$ шагов вывода β из α.

3). Для записи правил вывода с одинаковыми левыми частями

$$\alpha \to \beta_1 \quad \alpha \to \beta_2 \quad ... \quad \alpha \to \beta_n$$

будем пользоваться сокращенной записью

$$\alpha \to \quad \beta_1 \mid \beta_2 \mid ... \mid \beta_n.$$

Каждое β_i, i = 1, 2, ... ,n – ***альтернатива*** правила вывода из цепочки α.

4). $L(G)$ – язык, ***порождаемый*** грамматикой G.
Язык $L(G)$ содержит все терминальные цепочки, выводимые из S: $L(G) = \{\alpha \in T^* \mid S \Rightarrow {}^*\alpha\}$.

5). $L(G)$ – множество терминальных сентенциальных форм грамматики G.

Пример:
$G = \langle \{a, b, +\}, \{S, A, B\}, \{S \to A + B, S \to B + A, A \to a, B \to b\}, S \rangle$,
$L(G) = \{a + b, b + a\}$

6). $G_2 = G_1 \Leftrightarrow L(G_2) = L(G_1)$ – грамматики ***эквивалентны***, если они порождают один язык.

Пример:

Язык L в алфавите $V = \{1, 0\}$, состоящий из пустой строки и строк, каждая из которых содержит строку, состоящую из нулей и такого же количества единиц, можно также описать с помощью формальной системы определения множеств как $L = \{0^n 1^n \mid n \geq 0\}$.

10. КЛАССИФИКАЦИЯ ЯЗЫКОВ И ГРАММАТИК ПО ХОМСКОМУ

Хомский Ноам: 1928, США, лингвист, профессор Массачусетского технологического института, автор классификации формальных языков (иерархия Хомского), ввел понятие порождающей грамматики (1950).

1) **Иерархия Хомского:**
   $G_0 \supset G_I \supset G_{II} \supset G_{III}$,
   где $G_0, G_I, G_{II}, G_{III}$ множества грамматик типа 0, 1, 2 и 3.

2) **Грамматики типа 0:**
   $G_0 = \langle T, N, P, S \rangle$ – **неограниченные** грамматики, у которых нет никаких ограничений для правил.

   Правила имеют вид: $\alpha \to \beta$, где $\alpha \in V^+$, $\beta \in V^*$.

3) **Грамматики типа 1:**
   $G_I = \langle T, N, P, S \rangle$ – **контекстно-зависимые** (КЗ) грамматики (неукорачивающие грамматики).
   Правила имеют вид:
   $\alpha \to \beta$, где $\alpha \in V^+$, $\beta \in V^+$ и $|\alpha| \le |\beta|$.

   Контекстно-зависимая грамматика: один и тот же нетерминальный символ может быть заменен на ту или иную цепочку символов в зависимости от контекста (цепочки) в которой они встречаются.

   Примеры: слово «коса» имеет разный смысл во фразах «нашла коса на камень» и «коса расплелась», то же самое слово имеет различные значения в зависимости от контекста, в котором мы его находим.
   Пример на С++: **vec < a > b** без контекста неясно, это — сравнение двух переменных или специкация шаблона.

4) **Грамматики типа 2:**
   $G_{II} = \langle T, N, P, S \rangle$ – **контекстно-свободные** (КС) грамматики.

   Правила имеют вид: $A \to \alpha$, где $A \in N$, $\alpha \in V^*$, где А — нетерминал, β — цепочка нетерминалов и терминалов.

   Пример: в языках программирования символ “присваивание” раскрывается однозначно и не зависит от того, что окружает присваивание.

5) **Грамматики типа 3:**

$G_{III} = \langle T, N, P, S \rangle$ – **регулярные грамматики**.

Регулярные грамматики бывают праволинейными и леволинейными.

Правила праволинейной грамматики имеют вид:

$A \to \alpha$ или $A \to \alpha B$, где $A, B \in N$, $\alpha \in T^*$.

Правила леволинейной грамматики имеют вид:

$A \to \alpha$ или $A \to B\alpha$, где $A, B \in N$, $\alpha \in T^*$.

6) **Соотношения грамматик в иерархии Хомского:**

a) любая регулярная грамматика является контекстно-свободной грамматикой;
b) любая контекстно-свободная грамматика является контекстно-зависимой грамматикой;
c) любая контекстно-зависимая грамматика является грамматикой типа 0.

7) **Формальные языки классифицируются по типу порождающих их грамматик.**

8) **Соотношения между типами формальных языков:**

a) каждый регулярный язык является контекстно-свободным языком, но существуют контекстно-свободные языки, которые не являются регулярными;
b) каждый контекстно-свободный язык является контекстно-зависимым, но существуют контекстно-зависимые, которые не являются контекстно-свободными.
c) каждый контекстно-зависимый язык является языком типа 0.

Пример1: $G_1 = \langle T_1, N_1, P_1, S \rangle$, где $T_1 = \{a, 0, 1, ..., 9\}$, $N_1 = \{S, D\}$,

Правила: P_1: $S \to a \mid aD$,

$D \to 0 \mid 1 \mid 2 \mid 3 \mid ...9 \mid 0D \mid 1D \mid 2D \mid 3D \mid ... \mid 9D$.

$L(G_1) = \{a, a0, a1, ..., a9, a01, a02, ..a09..., a1, ...\}$ – регулярный правосторонний язык (тип 3).

Пример2: $G_2 = \langle T_2, N_2, P_2, S \rangle$, где $T_2 = \{a, 0, 1, ..., 9\}$, $N_2 = \{S, D, F\}$,

Правила: P_2: $S \to a \mid aD$,

$F \to 0 \mid 1 \mid 2 \mid 3 \mid ...9 \mid 0$,

$D \to F \mid DF$.

$L(G_2) = \{a, a0, a1, ..., a9, a01, a02, ..a09..., a1, ...\}$ – язык, порожденный контекстно-свободной грамматикой (тип 2).

$L(G_2) = L(G_1) \Leftrightarrow G_2 = G_1$, следовательно G_2 и G_1 эквивалентные грамматики.

Пример3:

$G_3 = \langle T_3, N_3, P_3, S \rangle$, $T_3 = \{0, 1\}$, $N_3 = \{S, A\}$,

где правила P_3: $S \to 0A1$, $0A \to 00A11$, $A \to \lambda$.

$L(G_3) = \{01, 0011, 000111, 00001111, ...\} = \{0^n 1^n \mid n > 0\}$ -грамматика язык типа 0.

Пример4:

$G_4 = \langle T_4, N_4, P_4, S \rangle$, $T_4 = \{a, c, b\}$, $N_4 = \{S, Q\}$,

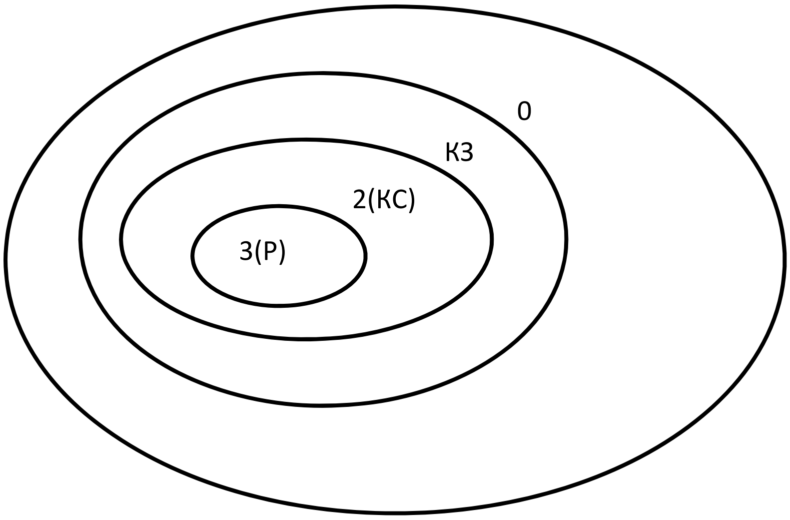
где правила P_4: $S \to aQb \mid accb$, $Q \to cSc$.

$L(G_4) = \{accb, acaccbcb, acacaccbcbcb, ...\} = \{(ac)^n (cb)^n \mid n > 0\}$ - грамматика и язык типа 2 (контекстно-независимая).

Пример5: $G_5 = \langle T_5, N_5, P_5, S \rangle$, $T_5 = \{+, -, *, /, (,), a, b\}$, $N_5 = \{S\}$, где P_5 $: S \to S + S \mid S - S \mid S * S \mid S / S \mid (S) \mid a \mid b$.

$L(G_5) = \{a * b, a + b, a - b, b / a, (a + b) * (b - a), ...\}$ - грамматика и язык типа 2 (контекстно-независимая).

Соотношение типов грамматик и языков:



Тип 3 (Р) – регулярная грамматика;
Тип 2 (КС) – контекстно-свободная грамматика;
Тип 3 (КЗ) – контекстно-зависимая грамматика;
Тип 0 – неограниченная грамматика.

Примеры различных типов формальных языков и грамматик по классификации Хомского.
Дана грамматика $G = \langle T, N, P, S \rangle$.

а) ***Язык типа 0*** $L(G)=\{a^2 b^{n^2-1} \mid n \geq 1\}$ определяется грамматикой с правилами вывода:

1) $S \to aaCFD$;	2) $AD \to D$;
3) $F \to AFB \mid AB$;	4) $Cb \to bC$;
5) $AB \to bBA$;	6) $CB \to C$;
7) $Ab \to bA$;	8) $bCD \to \varepsilon$.

б) ***Контекстно-зависимый язык*** $L(G)=\{a^n b^n c^n \mid n \geq 1\}$ определяется грамматикой с правилами вывода:

1) $S \to aSBC \mid abc$;	2) $bC \to bc$;
3) $CB \to BC$;	4) $cC \to cc$;
5) $BB \to bb$.	

в) ***Контекстно-свободный язык*** $L(G)=\{(ab)^n(cb)^n \mid n>0\}$ определяется грамматикой с правилами вывода:

1) $S \to aQb \mid accb$;
2) $Q \to cSc$.

г) ***Регулярный язык*** $L(G)=\{\omega \mid \omega\in\{a,\ b\}^+\}$, где нет двух рядом стоящих а определяется грамматикой с правилами вывода:

1) $S \to A \mid B$;
2) $A \to a \mid Ba$;
3) $B \to b \mid Bb \mid Ab$.

Все реальные языки программирования являются контекстно-зависимыми (тип 1), но большинство может быть разобрано как контекстно-свободные (тип 2) с последующей обработкой в виде проверок типов и т.п.

Фронтенд компилятора делят на три стадии:

- лексический анализ (в рамках регулярной грамматики текст делится на строковые литералы, числа, идентификаторы, операторы, разделители, убираются пробелы и комментарии);
- синтаксический анализ (в рамках контекстно-свободной грамматики);
- семантический анализ (проверка соответствия контекстно-зависимым правилам, таким как "переменная должна быть объявлена заранее").

11. СПОСОБЫ ЗАДАНИЯ СХЕМ ГРАММАТИК.

Схема грамматики содержит правила вывода, определяющие структуру цепочек порождаемого языка. Для задания правил используются различные формы описания:

- символическая;
- форма Бэкуса-Наура;
- итерационная форма;
- синтаксические диаграммы.

11.1. Формы Бэкуса - Наура

Цепочки языка могут содержать метасимволы, имеющие особое назначение. Метаязык, предложенный Бэкусом и Науром (БНФ) использует следующие обозначения:

- символ «::=» отделяет левую часть правила от правой (читается: «определяется как»);
- нетерминалы обозначаются произвольной символьной строкой, заключенной в угловые скобки «<» и «>»;
- терминалы – это символы, используемые в описываемом языке;
- правило может определять порождение нескольких альтернативных цепочек, которые отделяются друг от друга символом вертикальной черты «|» (читается: «или»).

Для удобства и компактности описаний, в расширенных БНФ (РБНФ) вводятся следующие дополнительные конструкции (***метасимволы***):

- квадратные скобки «[» и «]» означают, что заключенная в них синтаксическая конструкция может отсутствовать;
- фигурные скобки «{» и «}» означают повторение заключенной в них синтаксической конструкции ноль или более раз;
- сочетание фигурных скобок и косой черты «{/» и «/}» используется для обозначения повторения один и более раз;
- круглые скобки «(» и «)» используются для ограничения альтернативных конструкций;
- кавычки используются в тех случаях, когда один из метасимволов нужно включить в цепочку обычным образом.

Пример1. Правила, определяющие понятие «идентификатор» некоторого языка программирования:

<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*
<цифра> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<идентификатор> ::= <буква> { (<буква> | <цифра>) }

***Пример2*:** форма Бэкуса-Наура, описывающая целое число

<целое> := <целое без знака>
<целое> := +<целое без знака>
<целое> := –<целое без знака>
<целое без знака> := <цифра>
<целое без знака> := <целое без знака><цифра>
<цифра> := 0
<цифра> := 1
<цифра> := 2
<цифра> := 3
<цифра> := 4
<цифра> := 5
<цифра> := 6
<цифра> := 7
<цифра> := 8
<цифра> := 9

Пример3: сокращенная форма Бэкуса-Наура

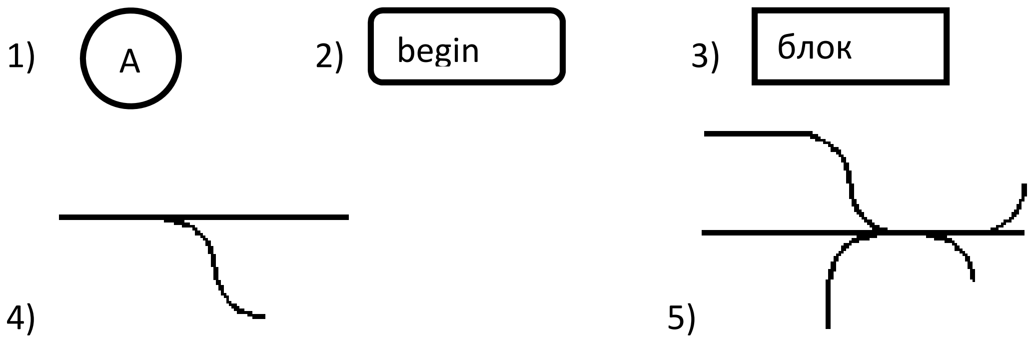
<целое> := <целое без знака>|+<целое без знака>
|–<целое без знака>
<целое без знака> := <цифра>|<целое без знака><цифра>
<цифра> := 0|1|2|3|4|5|6|7|8|9

11.2. Диаграммы Вирта

В метаязыке диаграмм Вирта используются графические примитивы, представленные на рисунке 1.4.1.

При построении диаграмм учитывают следующие правила:

- каждый графический элемент, соответствующий терминалу или нетерминалу, имеет по одному входу и выходу, которые обычно изображаются на противоположных сторонах;
- каждому правилу соответствует своя графическая диаграмма, на которой терминалы и нетерминалы соединяются посредством дуг;
- альтернативы в правилах задаются ветвлением дуг, а итерации - их слиянием;
- должна быть одна входная дуга (располагается обычно слева или сверху), задающая начало правила и помеченная именем определяемого нетерминала, и одна выходная, задающая его конец (обычно располагается справа и снизу);
- стрелки на дугах диаграмм обычно не ставятся, а направления связей отслеживаются движением от начальной дуги в соответствии с плавными изгибами промежуточных дуг и ветвлений.



Графические примитивы диаграмм Вирта

1) – терминальный символ, принадлежащий алфавиту языка;

2) – постоянная группа терминальных символов, определяющая название лексемы, ключевое слово и т.д.;

3) – нетерминальный символ, определяющий название правила;

4) – входная дуга с именем правила, определяющая его название;

5) – соединительные линии, обеспечивающие связь между терминальными и нетерминальными символами в правилах.

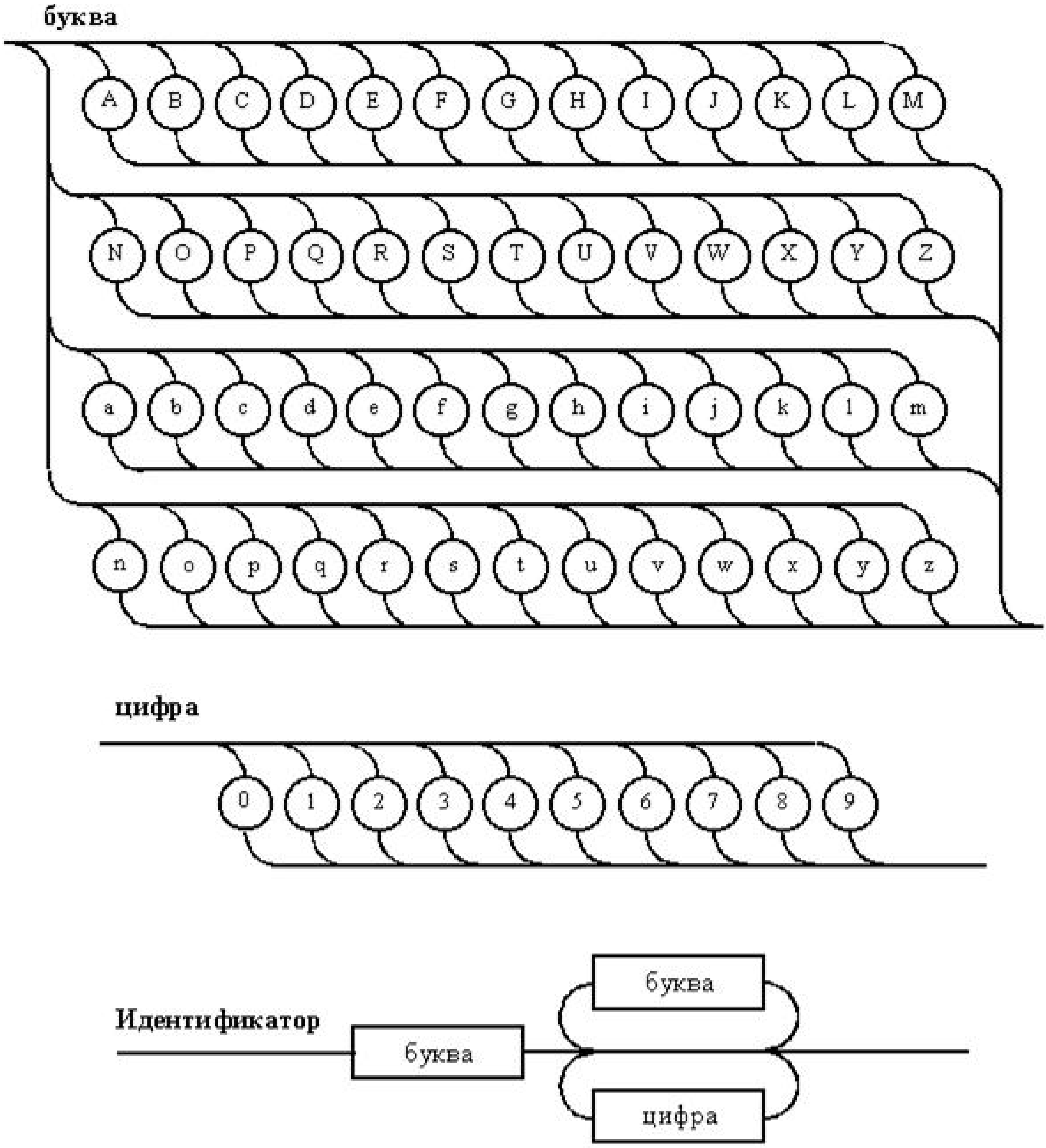
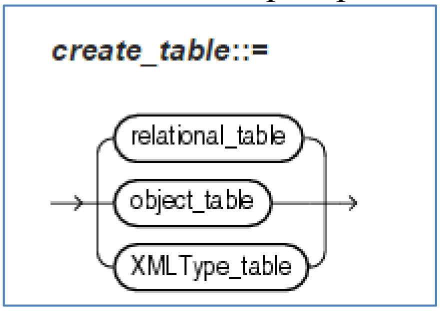
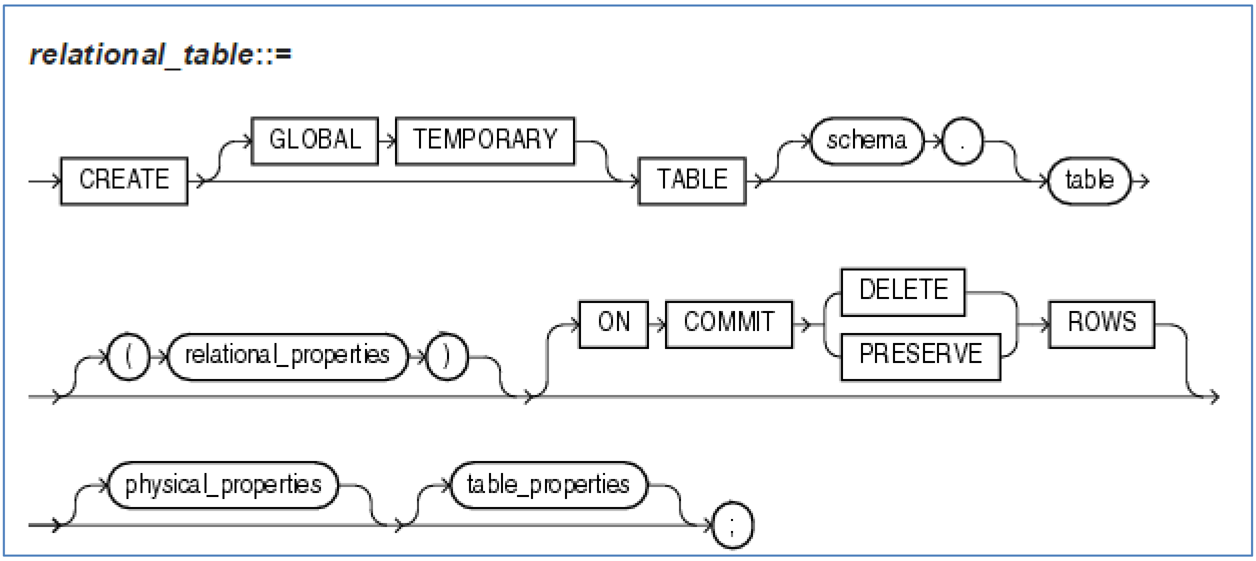
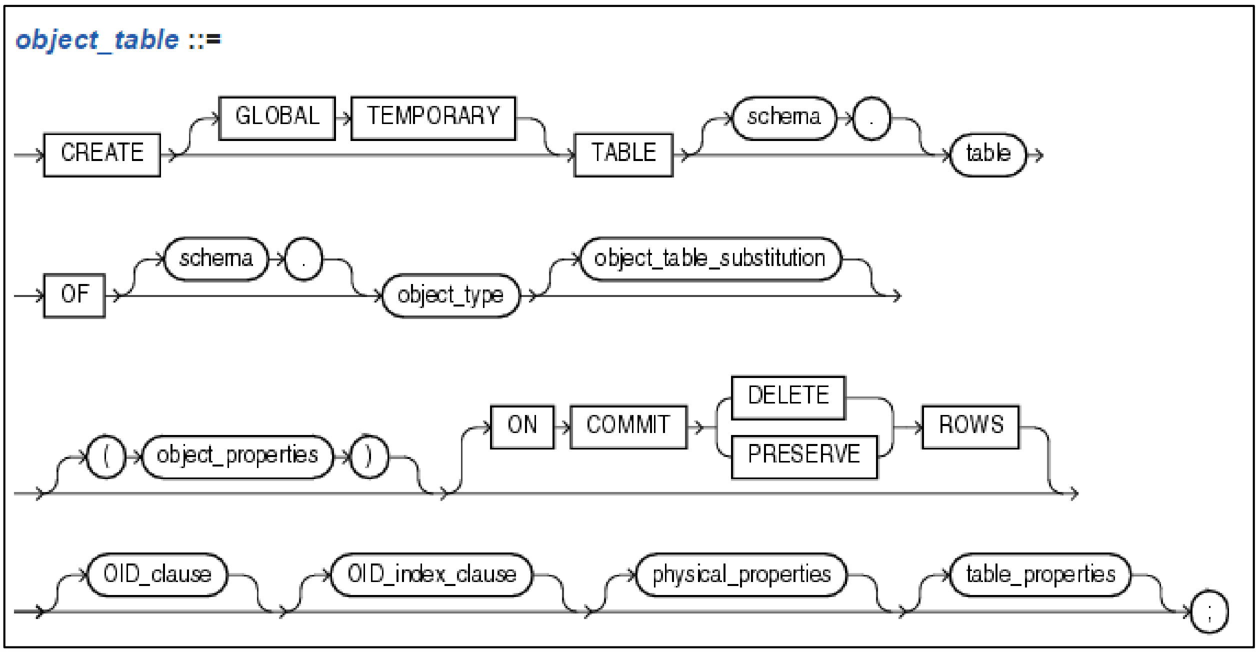


Диаграмма Вирта понятия «идентификатор»

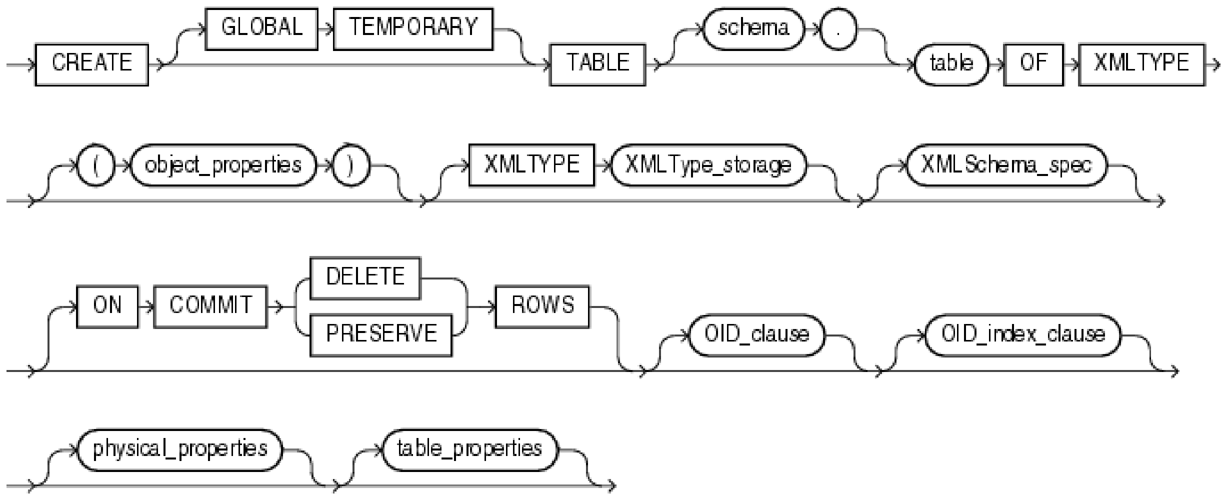
Пример: диаграмма задающая язык SQL СУБД Oracle. В примере описывается оператор CREATE TABLE – создание таблицы в базе данных.







XMLType_table ::=



relational_properties::=