

## СОДЕРЖАНИЕ

|   |    |
|---|----|
| 1.ЛАБОРАТОРНАЯ РАБОТА № 1. «ФОРМАЛЬНЫЕ ЯЗЫКИ И ГРАММАТИКИ» .....                      | 4  |
| 1.1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ .....   | 4  |
| 1.1.1. Основные понятия и определения .....   | 4  |
| 1.1.2. Классификация грамматик и языков по Хомскому .....                             | 5  |
| 1.1.3. Примеры грамматик и языков .....   | 6  |
| 1.1.4. Разбор цепочек .....   | 7  |
| 1.1.5. Преобразования грамматик .....   | 9  |
| 1.2. ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ 1 .....   | 10 |
| 1.2.1. Порядок выполнения работы .....  | 10 |
| 1.2.2. Задачи к лабораторной работе .....   | 10 |
| 1.2.3. Варианты заданий .....   | 13 |
| 1.2.4. Содержание отчета .....  | 14 |
| 1.2.5. Контрольные вопросы .....  | 14 |
| 2. ЛАБОРАТОРНЫЕ РАБОТЫ №2, 3. «ЛЕКСИЧЕСКИЙ АНАЛИЗАТОР. ТАБЛИЦЫ ИДЕНТИФИКАТОРОВ» ..... | 15 |
| 2.1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ .....   | 15 |
| 2.1.1. Общая схема компиляции .....   | 15 |
| 2.1.2. Задачи фаз компиляции .....  | 15 |
| 2.2. ПОСТРОЕНИЕ ЛЕКСИЧЕСКОГО АНАЛИЗАТОРА .....  | 17 |
| 2.2.1. Построение регулярной грамматики .....   | 17 |
| 2.2.2. Конечный автомат .....   | 18 |
| 2.3. ЗАДАНИЕ К ЛАБОРАТОРНОЙ РАБОТЕ 2 .....  | 25 |
| 2.3.1. Порядок выполнения работы .....  | 25 |
| 2.3.2. Варианты описаний языков .....   | 25 |
| 2.3.3. Содержание отчета .....  | 26 |
| 2.3.4. Контрольные вопросы .....  | 26 |
| 2.4. ЗАДАНИЕ К ЛАБОРАТОРНОЙ РАБОТЕ 3 .....  | 26 |
| 2.4.1. Порядок выполнения работы .....  | 26 |
| 2.4.2. Варианты описаний языков .....   | 26 |
| 2.4.3. Содержание отчета .....  | 30 |
| 2.4.4. Варианты заданий .....   | 30 |
| БИБЛИОГРАФИЧЕСКИЙ СПИСОК .....  | 31 |
| Приложение .....  | 32 |

# 1. Лабораторная работа № 1. ФОРМАЛЬНЫЕ ЯЗЫКИ И ГРАММАТИКИ

## 1.1. Теоретические сведения

### 1.1.1. Основные понятия и определения

**Определение:** *алфавит* - это конечное множество символов.

Предполагается, что термин "символ" имеет достаточно ясный интуитивный смысл и не нуждается в дальнейшем уточнении [1,2].

**Определение:** *цепочкой символов в алфавите V* называется любая конечная последовательность символов этого алфавита.

**Определение:** цепочка, которая не содержит ни одного символа, называется *пустой цепочкой*. Для ее обозначения будем использовать символ  $\varepsilon$ .

**Определение:** если  $\alpha$  и  $\beta$  - цепочки, то цепочка  $\alpha\beta$  называется *конкатенацией* (или *сцеплением*) цепочек  $\alpha$  и  $\beta$ .

Например, если  $\alpha = ab$  и  $\beta = cd$ , то  $\alpha\beta = abcd$ .

Для любой цепочки  $\alpha$  всегда  $\alpha\varepsilon = \varepsilon\alpha = \alpha$ .

**Определение:**  $n$ -ой степенью цепочки  $\alpha$  (будем обозначать  $\alpha^n$ ) называется конкатенация  $n$  цепочек  $\alpha$ .

$$\alpha^0 = \varepsilon; \alpha^n = \alpha\alpha^{n-1} = \alpha^{n-1}\alpha.$$

**Определение:** *язык* в алфавите  $V$  - это подмножество цепочек конечной длины в этом алфавите.

**Определение:** обозначим через  $V^*$  множество, содержащее все цепочки в алфавите  $V$ , включая пустую цепочку  $\varepsilon$ .

Например, если  $V = \{0,1\}$ , то  $V^* = \{\varepsilon, 0, 1, 00, 11, 01, 10, 000, 001, 011, \dots\}$ .

**Определение:** обозначим через  $V^+$  множество, содержащее все цепочки в алфавите  $V$ , исключая пустую цепочку  $\varepsilon$ .

Следовательно,  $V^* = V^+ \cup \{\varepsilon\}$ .

Известно несколько различных способов описания языков. В дальнейшем изложении будет использоваться способ описания посредством порождающих грамматик [1].

**Определение:** *декартовым произведением*  $A \times B$  множеств  $A$  и  $B$  называется множество  $\{(a,b) \mid a \in A, b \in B\}$ .

**Определение:** *порождающая грамматика G* - это четверка  $(VT, VN, P, S)$ , где

$VT$  - алфавит *терминальных символов (терминалов)*,

$VN$  - алфавит *нетерминальных символов (нетерминалов)*, не пересекающийся с  $VT$ ,

$P$  - конечное подмножество множества  $(VT \cup VN)^+ \times (VT \cup VN)^*$ ; элемент  $(\alpha, \beta)$  множества  $P$  называется *правилом вывода* и записывается в виде  $\alpha \rightarrow \beta$ ,

$S$  - *начальный символ (цель)* грамматики,  $S \in VN$ .

Для записи правил вывода с одинаковыми левыми частями

$$\alpha \rightarrow \beta_1 \quad \alpha \rightarrow \beta_2 \quad \dots \quad \alpha \rightarrow \beta_n$$

будем пользоваться сокращенной записью

$$\alpha \rightarrow \beta_1 \mid \beta_2 \mid \dots \mid \beta_n.$$

Каждое  $\beta_i, i = 1, 2, \dots, n$ , будем называть *альтернативой* правила вывода из цепочки  $\alpha$ .

**Пример грамматики:**  $G1 = (\{0,1\}, \{A,S\}, P, S)$ , где  $P$  состоит из правил

$$S \rightarrow 01A$$

$$A \rightarrow 00A1$$

$$A \rightarrow \varepsilon$$

**Определение:** цепочка  $\beta \in (VT \cup VN)^*$  непосредственно выводима из цепочки  $\alpha \in (VT \cup VN)^+$  в грамматике  $G = (VT, VN, P, S)$  (обозначим  $\alpha \rightarrow \beta$ ), если  $\alpha = \xi_1 \gamma \xi_2$ ,  $\beta = \xi_1 \delta \xi_2$ , где  $\xi_1, \xi_2, \delta \in (VT \cup VN)^*$ ,  $\gamma \in (VT \cup VN)^+$  и правило вывода  $\gamma \rightarrow \delta$  содержится в  $P$ .

Например, цепочка 00A11 непосредственно выводима из 0A1 в грамматике  $G_1$ .

**Определение:** цепочка  $\beta \in (VT \cup VN)^*$  выводима из цепочки  $\alpha \in (VT \cup VN)^+$  в грамматике  $G = (VT, VN, P, S)$  (обозначим  $\alpha \Rightarrow \beta$ ), если существуют цепочки  $\gamma_0, \gamma_1, \dots, \gamma_n$  ( $n \geq 0$ ), такие, что  $\alpha = \gamma_0 \rightarrow \gamma_1 \rightarrow \dots \rightarrow \gamma_n = \beta$ .

**Определение:** последовательность  $\gamma_0, \gamma_1, \dots, \gamma_n$  называется *выводом длины  $n$* .

Например,  $S \Rightarrow 010000A11$  в грамматике  $G_1$  (см. пример выше), т.к. существует вывод  $S \rightarrow 01A \rightarrow 0100A1 \rightarrow 010000A11$ . Длина вывода равна 3.

**Определение:** языком, порождаемым грамматикой  $G = (VT, VN, P, S)$ , называется множество  $L(G) = \{\alpha \in VT^* \mid S \Rightarrow \alpha\}$ .

Другими словами,  $L(G)$  - это все цепочки в алфавите  $VT$ , которые выводимы из  $S$  с помощью  $P$ .

Например,  $L(G_1) = \{0^n 1^n \mid n > 0\}$ .

**Определение:** цепочка  $\alpha \in (VT \cup VN)^*$ , для которой  $S \Rightarrow \alpha$ , называется *сентенциальной формой* в грамматике  $G = (VT, VN, P, S)$ .

Таким образом, язык, порождаемый грамматикой, можно определить как множество терминальных сентенциальных форм.

**Определение:** грамматики  $G_1$  и  $G_2$  называются *эквивалентными*, если  $L(G_1) = L(G_2)$ .

Например,

$$\begin{array}{ll} G_1 = (\{0,1\}, \{A,S\}, P_1, S) & \text{и} & G_2 = (\{0,1\}, \{S\}, P_2, S) \\ P_1: S \rightarrow 0A1 & & P_2: S \rightarrow 0S1 \mid 01 \\ & & 0A \rightarrow 00A1 \\ & & A \rightarrow \varepsilon \end{array}$$

эквивалентны, т.к. обе порождают язык  $L(G_1) = L(G_2) = \{0^n 1^n \mid n > 0\}$ .

### 1.1.2. Классификация грамматик и языков по Хомскому

#### ТИП 0:

Грамматика  $G = (VT, VN, P, S)$  называется *грамматикой типа 0*, если на правила вывода не накладывается никаких ограничений (кроме тех, которые указаны в определении грамматики) [1,2].

#### ТИП 1:

Грамматика  $G = (VT, VN, P, S)$  называется *неукорачивающей грамматикой*, если каждое правило из  $P$  имеет вид  $\alpha \rightarrow \beta$ , где  $\alpha \in (VT \cup VN)^+$ ,  $\beta \in (VT \cup VN)^+$  и  $|\alpha| \leq |\beta|$ .

Грамматика  $G = (VT, VN, P, S)$  называется *контекстно-зависимой (КЗ)*, если каждое правило из  $P$  имеет вид  $\alpha \rightarrow \beta$ , где  $\alpha = \xi_1 A \xi_2$ ;  $\beta = \xi_1 \gamma \xi_2$ ;  $A \in VN$ ;  $\gamma \in (VT \cup VN)^+$ ;  $\xi_1, \xi_2 \in (VT \cup VN)^*$ .

*Грамматику типа 1* можно определить как неукорачивающую либо как контекстно-зависимую.

Выбор определения не влияет на множество языков, порождаемых грамматиками этого класса, поскольку доказано, что множество языков, порождаемых неукорачивающими грамматиками, совпадает с множеством языков, порождаемых КЗ-грамматиками.

#### ТИП 2:

Грамматика  $G = (VT, VN, P, S)$  называется *контекстно-свободной (КС)*, если каждое правило из  $P$  имеет вид  $A \rightarrow \beta$ , где  $A \in VN$ ,  $\beta \in (VT \cup VN)^+$ .

Грамматика  $G = (VT, VN, P, S)$  называется *укорачивающей контекстно-свободной (УКС)*, если каждое правило из  $P$  имеет вид  $A \rightarrow \beta$ , где  $A \in VN$ ,  $\beta \in (VT \cup VN)^*$ .

Грамматику типа 2 можно определить как контекстно-свободную либо как укорачивающую контекстно-свободную.

Возможность выбора обусловлена тем, что для каждой УКС-грамматики существует почти эквивалентная КС-грамматика.

### ТИП 3:

Грамматика  $G = (VT, VN, P, S)$  называется *праволинейной*, если каждое правило из  $P$  имеет вид  $A \rightarrow tB$  либо  $A \rightarrow t$ , где  $A \in VN$ ,  $B \in VN$ ,  $t \in VT$ .

Грамматика  $G = (VT, VN, P, S)$  называется *леволинейной*, если каждое правило из  $P$  имеет вид  $A \rightarrow Bt$  либо  $A \rightarrow t$ , где  $A \in VN$ ,  $B \in VN$ ,  $t \in VT$ .

Грамматику типа 3 (регулярную,  $P$ -грамматику) можно определить как праволинейную либо как леволинейную [1].

Выбор определения не влияет на множество языков, порождаемых грамматиками этого класса, поскольку доказано, что множество языков, порождаемых праволинейными грамматиками, совпадает с множеством языков, порождаемых леволинейными грамматиками.

### Соотношения между типами грамматик:

- (1) любая регулярная грамматика является КС-грамматикой;
- (2) любая регулярная грамматика является УКС-грамматикой;
- (3) любая КС-грамматика является КЗ-грамматикой;
- (4) любая КС-грамматика является неукорачивающей грамматикой;
- (5) любая КЗ-грамматика является грамматикой типа 0.
- (6) любая неукорачивающая грамматика является грамматикой типа 0.

**Замечание:** УКС-грамматика, содержащая правила вида  $A \rightarrow \varepsilon$ , не является КЗ-грамматикой и не является неукорачивающей грамматикой.

**Определение:** язык  $L(G)$  является *языком типа  $k$* , если его можно описать грамматикой типа  $k$ .

### Соотношения между типами языков:

- (1) каждый регулярный язык является КС-языком, но существуют КС-языки, которые не являются регулярными (например,  $L = \{a^n b^n \mid n > 0\}$ ).
- (2) каждый КС-язык является КЗ-языком, но существуют КЗ-языки, которые не являются КС-языками (например,  $L = \{a^n b^n c^n \mid n > 0\}$ ).
- (3) каждый КЗ-язык является языком типа 0.

**Замечание:** УКС-язык, содержащий пустую цепочку, не является КЗ-языком.

## 1.1.3. Примеры грамматик и языков

**Соглашение:** если при описании грамматики указаны только правила вывода  $P$ , то будем считать, что большие латинские буквы обозначают нетерминальные символы,  $S$  - цель грамматики, все остальные символы - терминальные.

1) Язык типа 0:  $L(G) = \{a^2 b^{n^2-1} \mid n \geq 1\}$

G:  $S \rightarrow aaCFD$   
 $F \rightarrow AFB \mid AB$   
 $AB \rightarrow bBA$   
 $Ab \rightarrow bA$   
 $AD \rightarrow D$   
 $Cb \rightarrow bC$   
 $CB \rightarrow C$   
 $bCD \rightarrow \varepsilon$

2) Язык типа 1:  $L(G) = \{ a^n b^n c^n, n \geq 1 \}$

G:  $S \rightarrow aSBC \mid aBc$   
 $CB \rightarrow BC$   
 $bB \rightarrow bb$   
 $bC \rightarrow bc$   
 $cC \rightarrow cc$

3) Язык типа 2:  $L(G) = \{ (ac)^n (cb)^n \mid n > 0 \}$

G:  $S \rightarrow aQb \mid accb$   
 $Q \rightarrow cSc$

4) Язык типа 3:  $L(G) = \{ \omega \perp \mid \omega \in \{a,b\}^+ \}$ , где нет двух рядом стоящих a}

G:  $S \rightarrow A\perp \mid B\perp$   
 $A \rightarrow a \mid Ba$   
 $B \rightarrow b \mid Bb \mid Ab$

#### 1.1.4. Разбор цепочек

Цепочка принадлежит языку, порождаемому грамматикой, только в том случае, если существует ее вывод из цели этой грамматики. Процесс построения такого вывода (а, следовательно, и определения принадлежности цепочки языку) называется *разбором* [2].

С практической точки зрения наибольший интерес представляет разбор по **контекстно-свободным (КС и УКС) грамматикам**. Их порождающей мощности достаточно для описания большей части синтаксической структуры языков программирования, для различных подклассов КС-грамматик имеются хорошо разработанные практически приемлемые способы решения задачи разбора.

Рассмотрим основные понятия и определения, связанные с разбором по КС-грамматике.

**Определение:** вывод цепочки  $\beta \in (VT)^*$  из  $S \in VN$  в КС-грамматике  $G = (VT, VN, P, S)$ , называется *левым (левосторонним)*, если в этом выводе каждая очередная сентенциальная форма получается из предыдущей заменой самого левого нетерминала.

**Определение:** вывод цепочки  $\beta \in (VT)^*$  из  $S \in VN$  в КС-грамматике  $G = (VT, VN, P, S)$ , называется *правым (правосторонним)*, если в этом выводе каждая очередная сентенциальная форма получается из предыдущей заменой самого правого нетерминала.

В грамматике для одной и той же цепочки может быть несколько выводов, эквивалентных в том смысле, что в них в одних и тех же местах применяются одни и те же правила вывода, но в различном порядке.

Например, для цепочки  $a+b+a$  в грамматике

$G = (\{a,b,+ \}, \{S,T\}, \{S \rightarrow T \mid T+S; T \rightarrow a \mid b\}, S)$

можно построить выводы:

(1)  $S \rightarrow T+S \rightarrow T+T+S \rightarrow T+T+T \rightarrow a+T+T \rightarrow a+b+T \rightarrow a+b+a$

(2)  $S \rightarrow T+S \rightarrow a+S \rightarrow a+T+S \rightarrow a+b+S \rightarrow a+b+T \rightarrow a+b+a$

(3)  $S \rightarrow T+S \rightarrow T+T+S \rightarrow T+T+T \rightarrow T+T+a \rightarrow T+b+a \rightarrow a+b+a$

Здесь (2) - левосторонний вывод, (3) - правосторонний, а (1) не является ни левосторонним, ни правосторонним, но все эти выводы являются эквивалентными в указанном выше смысле.

Для КС-грамматик можно ввести удобное графическое представление вывода, называемое деревом вывода, причем для всех эквивалентных выводов дерева вывода совпадают.

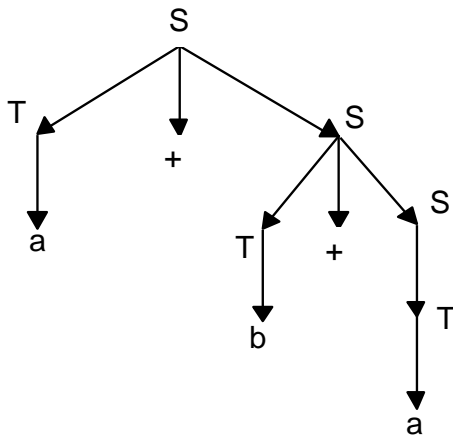
**Определение:** дерево называется *деревом вывода* (или *деревом разбора*) в КС-грамматике  $G = \{VT, VN, P, S\}$ , если выполнены следующие условия:

(1) каждая вершина дерева помечена символом из множества  $(VN \cup VT \cup \varepsilon)$ , при этом корень дерева помечен символом  $S$ ; листья - символами из  $(VT \cup \varepsilon)$ ;

(2) если вершина дерева помечена символом  $A \in VN$ , а ее непосредственные потомки - символами  $a_1, a_2, \dots, a_n$ , где каждое  $a_i \in (VT \cup \varepsilon)$ , то  $A \rightarrow a_1 a_2 \dots a_n$  - правило вывода в этой грамматике;

(3) если вершина дерева помечена символом  $A \in VN$ , а ее единственный непосредственный потомок помечен символом  $\varepsilon$ , то  $A \rightarrow \varepsilon$  - правило вывода в этой грамматике.

**Пример** дерева вывода для цепочки  $a+b+a$  в грамматике  $G$ :



**Определение:** КС-грамматика  $G$  называется *неоднозначной*, если существует хотя бы одна цепочка  $\alpha \in L(G)$ , для которой может быть построено два или более различных деревьев вывода.

Это утверждение эквивалентно тому, что цепочка  $\alpha$  имеет два или более разных левосторонних (или правосторонних) выводов.

**Определение:** язык, порождаемый грамматикой, называется *неоднозначным*, если он не может быть порожден никакой однозначной грамматикой.

**Пример** неоднозначной грамматики:

$$G = (\{if, then, else, a, b\}, \{S\}, P, S),$$

где  $P = \{S \rightarrow if\ b\ then\ S\ else\ S\ | if\ b\ then\ S\ | a\}$ .

В этой грамматике для цепочки  $if\ b\ then\ if\ b\ then\ a\ else\ a$  можно построить два различных дерева вывода.

Однако это не означает, что язык  $L(G)$  обязательно неоднозначный. Определенная нами **неоднозначность - это свойство грамматики, а не языка**, т.е. для некоторых неоднозначных грамматик существуют эквивалентные им однозначные грамматики.

Если грамматика используется для определения языка программирования, то она должна быть однозначной.

В приведенном выше примере разные деревья вывода предполагают соответствие `else` разным `then`. Если договориться, что `else` должно соответствовать ближайшему к нему `then`, и подправить грамматику  $G$ , то неоднозначность будет устранена:

$$\begin{aligned}
 S &\rightarrow if\ b\ then\ S\ | if\ b\ then\ S'\ else\ S\ | a \\
 S' &\rightarrow if\ b\ then\ S'\ else\ S'\ | a
 \end{aligned}$$

**Пример** неоднозначного КС-языка:

$$L = \{a^i b^j c^k \mid i = j \text{ или } j = k\}.$$

Интуитивно это объясняется тем, что цепочки с  $i = j$  должны порождаться группой правил вывода, отличных от правил, порождающих цепочки с  $j = k$ . Но тогда, по крайней мере, некоторые из цепочек с  $i = j = k$  будут порождаться обеими группами правил и, следовательно, будут иметь по два разных дерева вывода. Доказательство того, что КС-язык  $L$  неоднозначный, приведен в [2].

Одна из грамматик, порождающих  $L$ , такова:

$$\begin{aligned} S &\rightarrow AB \mid DC \\ A &\rightarrow aA \mid \varepsilon \\ B &\rightarrow bBc \mid \varepsilon \\ C &\rightarrow cC \mid \varepsilon \\ D &\rightarrow aDb \mid \varepsilon \end{aligned}$$

Очевидно, что она неоднозначна.

Дерево вывода можно строить *нисходящим* либо *восходящим* способом.

При нисходящем разборе дерево вывода формируется от корня к листьям; на каждом шаге для вершины, помеченной нетерминальным символом, пытаются найти такое правило вывода, чтобы имеющиеся в нем терминальные символы “проектировались” на символы исходной цепочки.

Метод восходящего разбора заключается в том, что исходную цепочку пытаются “свернуть” к начальному символу  $S$ ; на каждом шаге ищут подцепочку, которая совпадает с правой частью какого-либо правила вывода; если такая подцепочка находится, то она заменяется нетерминалом из левой части этого правила.

Если грамматика однозначная, то при любом способе построения будет получено одно и то же дерево разбора.

### 1.1.5. Преобразования грамматик

В некоторых случаях КС-грамматика может содержать недостижимые и бесплодные символы, которые не участвуют в порождении цепочек языка и поэтому могут быть удалены из грамматики [1,2].

**Определение:** символ  $x \in (VT \cup VN)$  называется *недостижимым* в грамматике  $G = (VT, VN, P, S)$ , если он не появляется ни в одной сентенциальной форме этой грамматики.

**Алгоритм удаления недостижимых символов:**

Вход: КС-грамматика  $G = (VT, VN, P, S)$

Выход: КС-грамматика  $G' = (VT', VN', P', S)$ , не содержащая недостижимых символов, для которой  $L(G) = L(G')$ .

Метод:

1.  $V_0 = \{S\}; i = 1.$
2.  $V_i = \{x \mid x \in (VT \cup VN), \text{ в } P \text{ есть } A \rightarrow \alpha x \beta \text{ и } A \in V_{i-1}, \alpha, \beta \in (VT \cup VN)^*\} \cup V_{i-1}.$
3. Если  $V_i \neq V_{i-1}$ , то  $i = i + 1$  и переходим к шагу 2, иначе  $VN' = V_i \cap VN;$   
 $VT' = V_i \cap VT;$   $P'$  состоит из правил множества  $P$ , содержащих только символы из  $V_i;$   $G' = (VT', VN', P', S).$

**Определение:** символ  $A \in VN$  называется *бесплодным* в грамматике  $G = (VT, VN, P, S)$ , если множество  $\{\alpha \in VT^* \mid A \Rightarrow \alpha\}$  пусто.

**Алгоритм удаления бесплодных символов:**

Вход: КС-грамматика  $G = (VT, VN, P, S).$

Выход: КС-грамматика  $G' = (VT, VN', P', S)$ , не содержащая бесплодных символов, для которой  $L(G) = L(G')$ .

Метод:

Рекурсивно строим множества  $N_0, N_1, \dots$

1.  $N_0 = \emptyset, i = 1$ .

2.  $N_i = \{A \mid (A \rightarrow \alpha) \in P \text{ и } \alpha \in (N_{i-1} \cup VT)^*\} \cup N_{i-1}$ .

3. Если  $N_i \neq N_{i-1}$ , то  $i = i + 1$  и переходим к шагу 2, иначе  $VN' = N_i$ ;  $P'$  состоит из правил множества  $P$ , содержащих только символы из  $VN' \cup VT$ ;  $G' = (VT, VN', P', S)$ .

**Определение:** КС-грамматика  $G$  называется *приведенной*, если в ней нет недостижимых и бесплодных символов.

**Алгоритм приведения грамматики:**

(1) обнаруживаются и удаляются все бесплодные нетерминалы.

(2) обнаруживаются и удаляются все недостижимые символы.

Удаление символов сопровождается удалением правил вывода, содержащих эти символы.

**Замечание:** если в этом алгоритме переставить шаги (1) и (2), то не всегда результатом будет приведенная грамматика.

Для описания синтаксиса языков программирования стараются использовать однозначные приведенные КС-грамматики.

## 1.2. Задание на лабораторную работу 1

### 1.2.1. Порядок выполнения работы

1. Ознакомиться с материалами методических указаний.
2. Выполнить задания, согласно варианту.
3. Ответить на два контрольных вопроса, по заданию преподавателя.
3. Составить и защитить отчет о выполнении задания.

### 1.2.2. Задачи к лабораторной работе

1. Дана грамматика. Построить вывод заданной цепочки и дерево разбора.

a)  $S \rightarrow T \mid T+S \mid T-S$

$T \rightarrow F \mid F*T$

$F \rightarrow a \mid b$

Цепочка  $a-b*a+b$

b)  $S \rightarrow aSBC \mid abC$

$CB \rightarrow BC$

$bB \rightarrow bb$

$bC \rightarrow bc$

$cC \rightarrow cc$

Цепочка  $aaabbbccc$

2. Построить все сентенциальные формы для грамматики с правилами:

$S \rightarrow A+B \mid B+A$

$A \rightarrow a$

$B \rightarrow b$

3. Построить грамматику, порождающую язык:

a)  $L = \{ a^n b^m \mid n, m \geq 1 \}$

b)  $L = \{ \alpha\beta\gamma\epsilon \mid \alpha, \beta, \gamma - \text{любые цепочки из } a \text{ и } b \}$

c)  $L = \{ a_1 a_2 \dots a_n a_n \dots a_2 a_1 \mid a_i = 0 \text{ или } 1, n \geq 1 \}$



8. Дана грамматика с правилами:

$$\begin{aligned} \text{a) } S &\rightarrow S0 \mid S1 \mid D0 \mid D1 \\ D &\rightarrow H. \\ H &\rightarrow 0 \mid 1 \mid H0 \mid H1 \end{aligned}$$

$$\begin{aligned} \text{b) } S &\rightarrow \text{if } B \text{ then } S \mid B = E \\ E &\rightarrow B \mid B + E \\ B &\rightarrow a \mid b \end{aligned}$$

Построить восходящим и нисходящим методами дерево вывода для цепочки:

$$\text{a) } 10.1001$$

$$\text{b) } \text{if } a \text{ then } b = a+b+b$$

9. Построить регулярную грамматику, порождающую цепочки в алфавите  $\{a, b\}$ , в которых символ  $a$  не встречается два раза подряд.

10. Написать КС-грамматику для языка  $L$ , построить дерево вывода и левосторонний вывод для цепочки  $aabbbsccc$ .

$$L = \{a^{2n} b^m c^{2k} \mid m=n+k, m>1\}.$$

11. Построить грамматику, порождающую сбалансированные относительно круглых скобок цепочки в алфавите  $\{a, (, ), \perp\}$ . Сбалансированную цепочку  $\alpha$  определим рекуррентно: цепочка  $\alpha$  сбалансирована, если

a)  $\alpha$  не содержит скобок,

b)  $\alpha = (\alpha_1)$  или  $\alpha = \alpha_1 \alpha_2$ , где  $\alpha_1$  и  $\alpha_2$  сбалансированы.

12. Написать КС-грамматику, порождающую язык  $L$ , и вывод для цепочки  $aacbbbca$  в этой грамматике.

$$L = \{a^n cb^m ca^n \mid n, m>0\}.$$

13. Написать КС-грамматику, порождающую язык  $L$ , и вывод для цепочки  $110000111$  в этой грамматике.

$$L = \{1^n 0^m 1^p \mid n+p>m; n, p, m>0\}.$$

14. Дана грамматика  $G$ . Определить ее тип; язык, порождаемый этой грамматикой; тип языка.

$$\begin{aligned} G: S &\rightarrow 0A1 \\ 0A &\rightarrow 00A1 \\ A &\rightarrow \varepsilon \end{aligned}$$

15. Дан язык  $L = \{1^{3n+2} 0^n \mid n \geq 0\}$ . Определить его тип, написать грамматику, порождающую  $L$ . Построить левосторонний и правосторонний выводы, дерево разбора для цепочки  $1111111100$ .

16. Привести пример грамматики, все правила которой имеют вид  $A \rightarrow Bt$ , либо  $A \rightarrow tB$ , либо  $A \rightarrow t$ , для которой не существует эквивалентной регулярной грамматики.

17. Написать общие алгоритмы построения по данным КС-грамматикам  $G1$  и  $G2$ , порождающим языки  $L1$  и  $L2$ , КС-грамматики для

- $L1 \cup L2$
- $L1 * L2$
- $L1^*$

**Замечание:**  $L = L1 * L2$  - это конкатенация языков  $L1$  и  $L2$ , т.е.  $L = \{ \alpha\beta \mid \alpha \in L1, \beta \in L2 \}$ ;  $L = L1^*$  - это итерация языка  $L1$ , т.е. объединение  $\{ \epsilon \} \cup L1 \cup L1*L1 \cup L1*L1*L1 \cup \dots$

18. Показать, что грамматика

$$E \rightarrow E+E \mid E*E \mid (E) \mid i$$

неоднозначна. Как описать этот же язык с помощью однозначной грамматики?

19. Показать, что наличие в КС-грамматике правил вида

a)  $A \rightarrow AA \mid \alpha$

b)  $A \rightarrow A\alpha A \mid \beta$

c)  $A \rightarrow \alpha A \mid A\beta \mid \gamma$

где  $\alpha, \beta, \gamma \in (VT \cup VN)^*$ ,  $A \in VN$ , делает ее неоднозначной. Можно ли преобразовать эти правила таким образом, чтобы полученная эквивалентная грамматика была однозначной?

20. Показать, что грамматика  $G$  неоднозначна. Какой язык она порождает? Является ли этот язык однозначным?

$$G: S \rightarrow aAc \mid aB$$

$$B \rightarrow bc$$

$$A \rightarrow b$$

21. Дана КС-грамматика  $G = \{ VT, VN, P, S \}$ . Предложить алгоритм построения множества

$$X = \{ A \in VN \mid A \Rightarrow \epsilon \}.$$

22. Для произвольной КС-грамматики  $G$  предложить алгоритм, определяющий, пуст ли язык  $L(G)$ .

23. Написать приведенную грамматику, эквивалентную данной.

a)  $S \rightarrow aABS \mid bCACd$   
 $A \rightarrow bAB \mid cSA \mid cCC$   
 $B \rightarrow bAB \mid cSB$   
 $C \rightarrow cS \mid c$

b)  $S \rightarrow aAB \mid E$   
 $A \rightarrow dDA \mid \epsilon$   
 $B \rightarrow bE \mid f$   
 $C \rightarrow cAB \mid dSD \mid a$   
 $D \rightarrow eA$   
 $E \rightarrow fA \mid g$

### 1.2.3. Варианты заданий

Выбрать из таблицы 1 задания согласно варианту. Вариант выбирается по номеру в списке подгруппы.

Таблица 1. Варианты заданий

| Вариант   | 1                 | 2                 | 3               | 4                | 5               | 6                | 7                | 8               | 9                | 10               | 11               | 12               | 13              | 14               | 15               | 16              |
|-----------|-------------------|-------------------|-----------------|------------------|-----------------|------------------|------------------|-----------------|------------------|------------------|------------------|------------------|-----------------|------------------|------------------|-----------------|
| № заданий | 1a,<br>5a,<br>23a | 1b,<br>5b,<br>23b | 2,<br>6a,<br>22 | 3a,<br>6b,<br>21 | 3b,<br>7,<br>20 | 3c,<br>8a,<br>19 | 3d,<br>8b,<br>18 | 3e,<br>9,<br>17 | 3f,<br>5a,<br>16 | 3g,<br>5b,<br>15 | 3h,<br>6a,<br>14 | 3i,<br>6b,<br>13 | 3j,<br>7,<br>14 | 3k,<br>8a,<br>13 | 4a,<br>8b,<br>12 | 4b,<br>9,<br>11 |

#### **1.2.4. Содержание отчета**

- 1) Титульный лист (Образец оформления в приложении А)
- 2) Содержание
- 3) Постановка задачи.
- 4) Решение задач
- 5) Ответы на контрольные вопросы
- 6) Выводы
- 7) Список литературы

#### **1.2.5. Контрольные вопросы**

- 1) Дайте определение усеченной итерации алфавита.
- 2) Из чего состоит правило грамматики.
- 3) Дайте определение порождающей грамматики.
- 4) Чем отличаются терминальные и нетерминальные символы?
- 5) Перечислите способы задания формальных языков.
- 6) Что лежит в основе классификации грамматик по Хомскому?
- 7) Как соотносятся типы порождающих грамматик и языков?
- 8) Дайте определение распознавателю для формальных языков.
- 9) Что понимается под конфигурацией распознавателя?
- 10) Что понимается под «рекурсивным правилом» грамматики?

## 2. Лабораторные работы № 2, 3. ЛЕКСИЧЕСКИЙ АНАЛИЗАТОР. ТАБЛИЦЫ ИДЕНТИФИКАТОРОВ

### 2.1. Теоретические сведения

#### 2.1.1. Общая схема компиляции

Транслятор – это программа, которая переводит программу на исходном языке в эквивалентную ей программу на выходном языке.

Компилятор – это транслятор, который осуществляет перевод исходной программы в эквивалентную ей результирующую программу на языке машинных кодов или на языке ассемблера.

Интерпретатор – это программа, которая воспринимает исходную программу на входном языке и выполняет ее.

Задача компилятора состоит в преобразовании ориентированного на пользователя программного обеспечения в машинно-ориентированное представление [3].

Задача компиляции рассматривается на двух этапах, каждый из которых состоит из фаз:

#### 1. Этап анализа

- 1.1. Лексический анализ.
- 1.2. Синтаксический анализ.
- 1.3. Семантический анализ.

#### 2. Этап синтеза.

- 2.1. Генерация машинно-независимого кода.
- 2.2. Оптимизация машинно-независимого кода.
- 2.3. Распределение памяти.
- 2.4. Генерация машинного кода.
- 2.5. Оптимизация машинного кода.

Общая схема компиляции представлена на рисунке 1.

#### 2.1.2. Задачи фаз компиляции

Лексический анализ. На этой фазе формируются лексемы языка.

Лексема – это структурная единица языка, которая состоит из элементарных символов языка и не содержит в своем составе других структурных единиц языка. Для языков программирования лексемами являются идентификаторы, константы, ключевые слова, знаки операций и т. д.

Задачей фазы является переход от символов входного языка к лексемам, а также проверка корректности входной строки. На вход лексического анализатора подается строка, состоящая из символов входного языка, происходит преобразование введенных символов по предусмотренным языком шаблонам в лексемы, которые кодируются и передаются на последующие фазы в виде кодов (токенов), например, ключевые слова if, while, do, then, а также знаки присваивания :=, равенства = и т.д. рассматриваются как единые сущности (лексемы) и представляются кодами (токенами), например, i, w, d, t, p, r соответственно. На дальнейших фазах компиляции работа ведется именно с токенами, а не с отдельными символами. Лексический анализатор не работает с контекстом, для него не важен порядок символов, он анализирует только принадлежность введенных символов языку, а также соответствие введенных конструкций шаблонам языка. Наряду с этим происходит удаление пробелов, комментариев и других символов, не имеющих смысловой нагрузки [4].

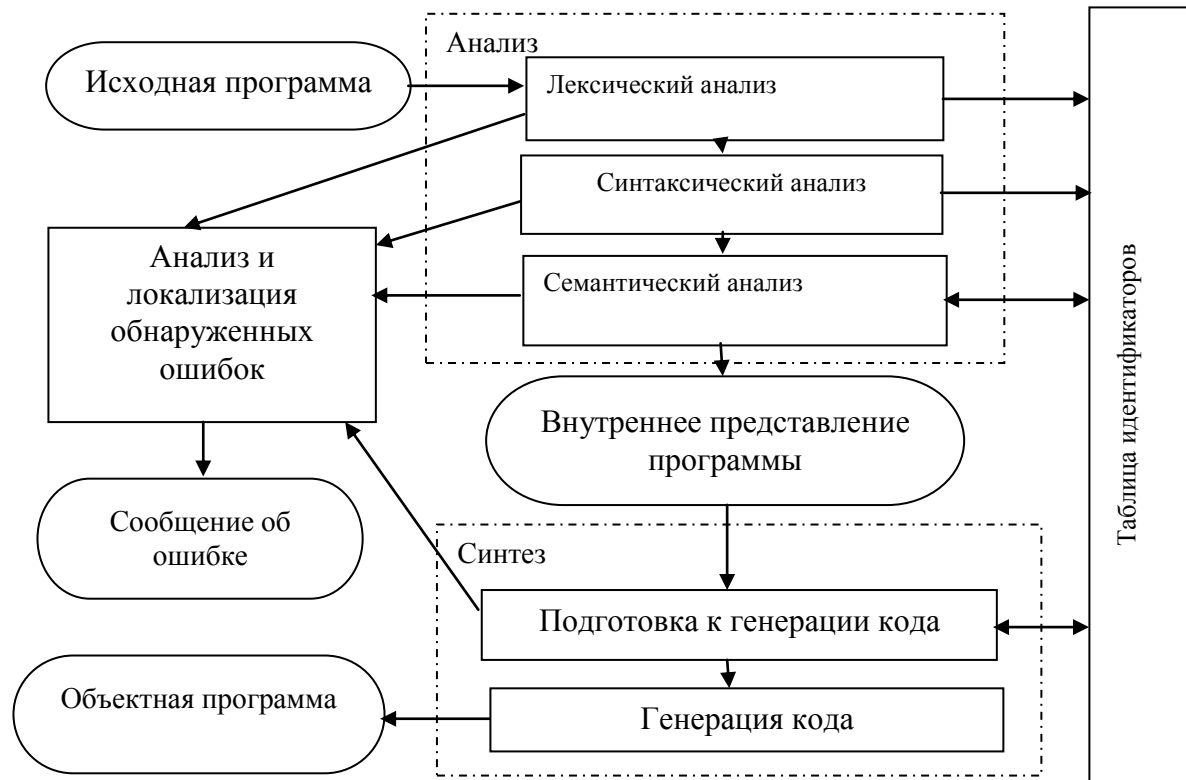


Рисунок 1 – Общая схема работы компилятора

На фазе лексического анализа начинается формирование таблицы идентификаторов.

**Синтаксический анализ.** Работает с контекстом. Определяет общую структуру конструкций языка. Например, может проверить соответствие скобок, корректность структур `if –then-else`, `while- do`, и т.д. Результатом синтаксического анализа является представление программы в виде синтаксического дерева, листья которого соответствуют токенам входной строки. На этом этапе продолжает формироваться таблица идентификаторов.

**Семантический анализ.** Выполняет проверку соответствия типов данных в исходной программе, отлавливает бесконечные рекурсии, деление на ноль и другие недопустимые действия.

На этапе анализа ведущей фазой является синтаксический анализ. Обычно разработанная программа – это программа синтаксического анализатора, по необходимости вызывающая модули лексического и семантического анализа.

На этапе синтеза не все фазы являются обязательными. Например, можно не строить промежуточный (машинно-независимый) код, можно обойтись без фазы оптимизации на уровне машинно-независимого кода, на уровне машинно-зависимого кода или на обоих. Но есть ряд причин, по которым эти фазы присутствуют в процессе компиляции. Рассмотрим эти причины.

**Генерация машинно-независимого (промежуточного) кода.** Включение этой фазы в процесс компиляции позволяет:

- обеспечить переносимость компилятора на другие машины, поскольку позволяет обособить компилятор от целевого языка и конкретной машины;
- облегчить процесс построения целевого кода, т.к. строить машинно-зависимый код гораздо удобнее по промежуточному коду;
- обеспечить дополнительный уровень оптимизации синтезируемого кода.

В качестве промежуточного представления могут использоваться следующие средства: синтаксическое дерево разбора, полученное в результате синтаксического анализа; ПОЛИЗ (польская инверсная запись), представление кода в виде триад и т.д. [3,4].

Оптимизация. Потребность в оптимизации может быть различной. Если требуется очень эффективный код, то компилятор должен обеспечить значительную оптимизацию. В то же время, во многих средах скорость работы ПО не является критическим параметром, следовательно, достаточно лишь незначительной оптимизации. Поскольку многие типы оптимизации трудоемки и требуют значительных затрат, то выбор уровня оптимизации является серьезной задачей. В некоторых компиляторах пользователь сам может выбрать уровень оптимизации.

Распределение памяти. На этой фазе каждая константа и переменная, используемые в программе получают зарезервированное место в памяти для хранения своего значения. Данная область памяти может иметь один из следующих типов [5]:

1) Статическая память, если время жизни переменной равно времени жизни программы. Не может быть освобождена до завершения выполнения программы.

2) Динамическая память, если время жизни переменной равно времени жизни определенного блока, функции или процедуры. Может быть освобождена после выполнения данного фрагмента, до завершения выполнения программы.

3) Глобальная матрица, если время жизни переменной в момент компиляции неизвестно, а память должна освобождаться и выделяться в процессе выполнения по необходимости.

От эффективности распределение памяти во многом зависит эффективность работы компилятора.

## 2.2. Построение лексического анализатора

На вход лексического анализатора подается строка символов, анализатор должен просканировать строку, проверить ее корректность, удалить пробелы и другие не несущие смысловой нагрузки символы, распознать лексемы, заменить последовательности символов кодами лексем. На выходе должен выдаваться результат в виде последовательности токенов, найденных во входной строке лексем [3,4]. Удобно обозначать лексемы сокращениями, обозначающими тип лексемы и цифрой, означающей порядковый номер лексемы этого типа, например: `if` можно обозначить `i1`, если этот тип лексемы встречается первый раз.

Лексический анализатор должен находить и, по возможности, диагностировать ошибки во входной строке. Лексический анализатор не может распознать все возможные ошибки, например, не сможет проследить соответствие скобок, наличие бесконечной рекурсии и т.д. Поскольку лексический анализатор не работает с контекстом, то он может отследить только следующие типы ошибок:

- 1). Наличие символа, не принадлежащего рассматриваемой грамматике.
- 2). Несоответствие лексем входной строки ни одному из шаблонов языка.

Программа лексического анализатора должна обеспечить возможность определения места возникновения ошибки, вывести сообщение об ошибке и продолжить разбор.

### 2.2.1. Построение регулярной грамматики

Синтаксис лексем, как правило, описывается в рамках автоматной грамматики, или грамматики типа 3 в соответствии с классификацией Хомского [2]. Такие грамматики могут быть представлены регулярными выражениями.

Основные приемы записи регулярных выражений:

- 1). Операция «или» обозначается вертикальной чертой: **a|b** и означает: a или b;
- 2). Итерация, означает нуль или более повторений, обозначается **{}**: **{a}** -- нуль или любое количество символов a;
- 3). Конкатенация, т.е. одна строка приписывается к другой: **ab**.

Например, цепочка из символов a, за которой следует строка, состоящая из b или c и b или d.

Шаблон такой строки может быть записан с помощью следующего регулярного выражения.

$\{a\}(b|c)(b|d)$ .

Этому шаблону соответствует множество строк, например:  $aaaaacd$ ,  $abb$ ,  $abd$ ,  $cd$  и т.д.

Поскольку грамматика может быть представлена регулярными выражениями, то лексический анализатор (сканер) может быть организован в виде модели конечного автомата.

Конечный автомат для каждой входной цепочки языка дает ответ на вопрос о том, принадлежит ли цепочка языку, заданному автоматом.

### 2.2.2. Конечный автомат

Конечным автоматом называется пятерка следующего вида:

$M(Q, V, \delta, q_0, F)$ , где

1) конечное множество  $Q$  внутренних состояний, каждое из которых, за исключением одного (обозначим его -  $F$ ), соответствует нетерминальному символу грамматики;

2) конечный входной алфавит  $V$ , каждый символ которого соответствует терминальному символу грамматики;

3)  $\delta$  - функция переходов, отображающая  $\delta(a, q)=R$ ,  $a \in V$ ,  $q \in Q$ ,  $R \subseteq Q$ ;

4) начальное состояние  $q_0$ ;

5) множество  $F$  конечных состояний автомата.

Попадание автомата в одно из состояний множества  $F$  означает, что процесс распознавания входной строки закончен. Можно сказать, что каждому из последних состояний соответствует свой выходной символ (свойство модели автомата Мура [3]), при появлении которого на выходе автомата запускается соответствующая семантическая подпрограмма сканера, формирующая код распознанной лексемы.

Конечный автомат должен быть детерминированным, т. е. в каждом его состоянии для любого входного символа функция переходов содержит не более одного состояния. Иными словами, из каждого состояния по каждому входному символу должно быть не более одного перехода.

Графом конечного автомата называется направленный граф, вершины которого помечены состояниями автомата, а дуги -- символами, встречающимися во входной цепочке.

Для детерминированного автомата в графе из каждого состояния для каждого символа языка выходит не более одной дуги, помеченной этим символом.

#### ПРИМЕР 1.

Пусть даны следующие шаблоны входных цепочек:

1). Строки, начинающиеся с символа  $b$ , затем любое количество символов  $a$  и заканчивающиеся на символы  $cd$ .

2). Строки, начинающиеся на символ  $*$ , затем любое количество  $b$ , заканчивается на  $*$ .

3). Строки, начинающиеся сочетанием символов  $ab$ , затем четное количество  $ab$  и заканчиваются «.».

Записать эти шаблоны можно с помощью регулярных выражений:

1).  $b\{a\}cd$

2).  $\{*\{b\}*\}$

3).  $ab\{abab\}$ .

Соответствующий граф автомата представлен на рис. 2.

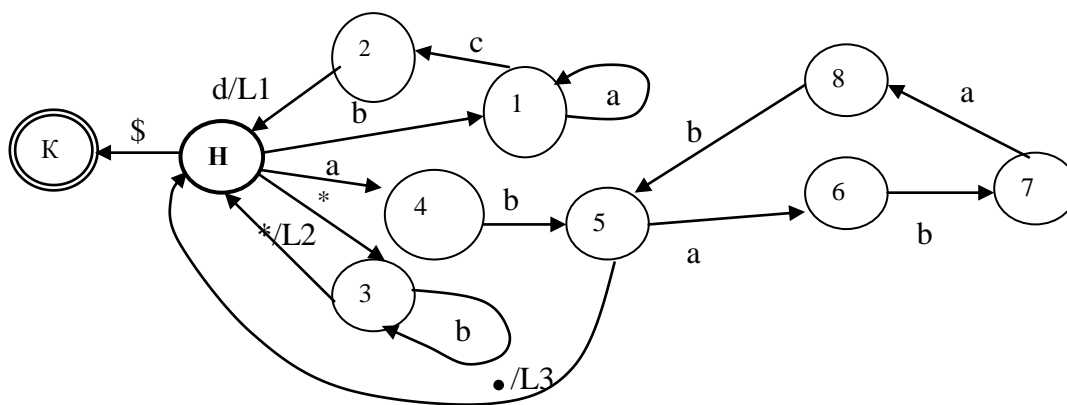


Рисунок 2 – Граф автомата

Знак \$ означает конец строки. Отметки на дугах, входящих в состояние H, соответствуют типам лексем. Конечное состояние K автомата, связано с допуском распознанных лексем. Обозначается конечное состояние двойным кружком. По его достижении выдается список найденных во входной цепочке лексем, с указанием их типа.

Таблица переходов и выходов конечного автомата

|   | a | b | c | d     | *     | .     | \$       |
|---|---|---|---|-------|-------|-------|----------|
| H | 4 | 1 |   |       | 3     |       | <b>K</b> |
| 1 | 1 |   | 2 |       |       |       |          |
| 2 |   |   |   | H/ L1 |       |       |          |
| 3 |   | 3 |   |       | H/ L2 |       |          |
| 4 |   | 5 |   |       |       |       |          |
| 5 | 6 |   |   |       |       | H/ L3 |          |
| 6 |   | 7 |   |       |       |       |          |
| 7 | 8 |   |   |       |       |       |          |
| 8 |   | 5 |   |       |       |       |          |

Ячейки, незаполненные значениями, соответствуют ошибке. Переход в конечное состояние соответствует выводу о допустимости входной цепочки символов.

Для определения допустимости входной цепочки символов, достаточно найти путь на графе автомата, дуги которого помечены символами, входящими в цепочку. Например, на вход подается цепочка baaaacdababab\$.

Соответствующий цепочке путь: H→1→1→1→1→1→2→ (обнаружена лексема L1) H→4→5→6→7→8→5→H(обнаружена лексема L2), следующий символ конец строки, следовательно, разбор окончен, его результатом является сообщение о корректности входной строки и обнаруженные лексемы.

Для нашего примера результат следующий:

baaaacd – лексема L1,

ababab – лексема L2,

«входная строка принадлежит языку».

Для обработки ошибок можно ввести дополнительное внутреннее состояние автомата - E("ошибка") - и заменить в таблице переходов пустые ячейки символами состояния E, по этому символу запускается семантическая подпрограмма обработки ошибочной ситуации.

**ПРИМЕР 2.**

При работе с языками программирования шаблоны лексем определяются синтаксисом языка.

ОЦ  $\rightarrow$  WHILE <выражение> DO<оператор>END.

<выражение>  $\rightarrow$  n<n|n<=n|n=n|n<>n|n>n|n>=n

<оператор>  $\rightarrow$  n:=<оператор>|n+n|n-n|n\*n|n/n|n

n  $\rightarrow$  <id>|<d>

<id>  $\rightarrow$  <b>|<id> <b>

<d>  $\rightarrow$  <z>|<d> <z>

<b>  $\rightarrow$  a|...|z

<z>  $\rightarrow$  0|...|9

ws  $\rightarrow$  ('|tab)<sup>+</sup>

Цель – построение автомата распознавателя. Строить графы будем по частям, затем объединим в один, поэтому нумерация состояний автомата будет приведена с учетом дальнейшего объединения.

1) Строим граф переходов для ключевых слов.

Граф автомата представлен на рисунке 3.

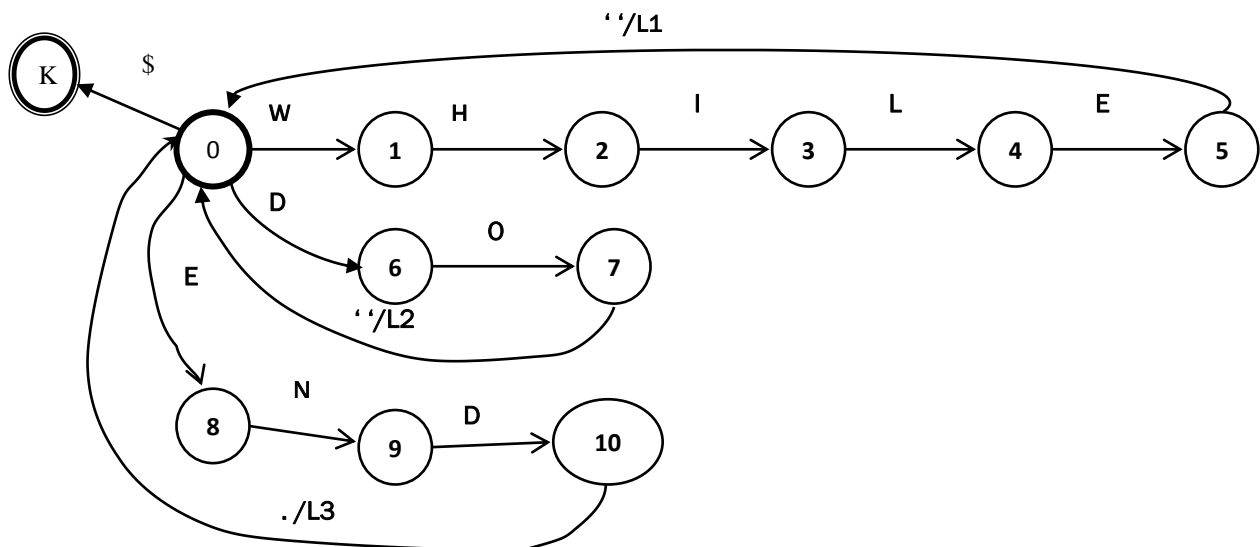


Рисунок 3 – Граф автомата-распознавателя ключевых слов

2) Строим граф для идентификаторов. Граф автомата для идентификаторов, представлен на рисунке 4.

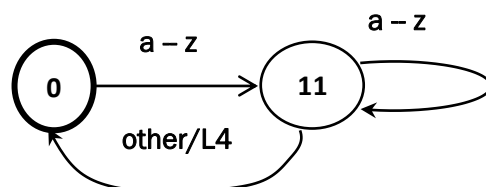


Рисунок 4 – Граф автомата-распознавателя идентификаторов

На рисунке 4, множество other = {0|1|...|9|' |+|-|\*|/|:|<|>|=|tab|.}

3) Строим граф для чисел. Граф представлен на рисунке 5. На рисунке 5, множество  $other1 = \{a|b|\dots|z\}^* \{+|-|*|/|:|<|>|=|tab|\}$ .

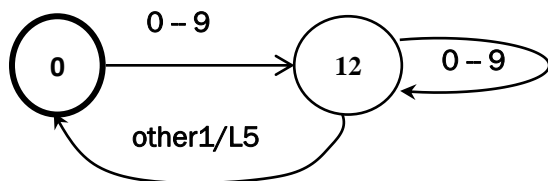


Рисунок 5 – Граф автомата-распознавателя чисел

4) Граф для арифметических операций представлен на рисунке 6.

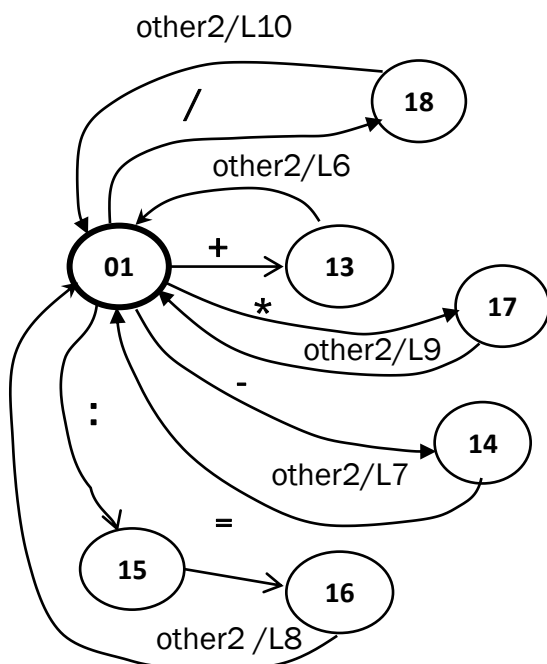


Рисунок 6 – Граф автомата-распознавателя арифметических выражений

На рисунке 6 множество  $other2 = \{0|1|\dots|9|a|b|\dots|z\}^* \{.$

5) Граф для операций отношения

Граф распознавателя отношений представлен на рисунке 7. Множество  $other2 = \{0|1|\dots|9|a|b|\dots|z\}^* \{.$

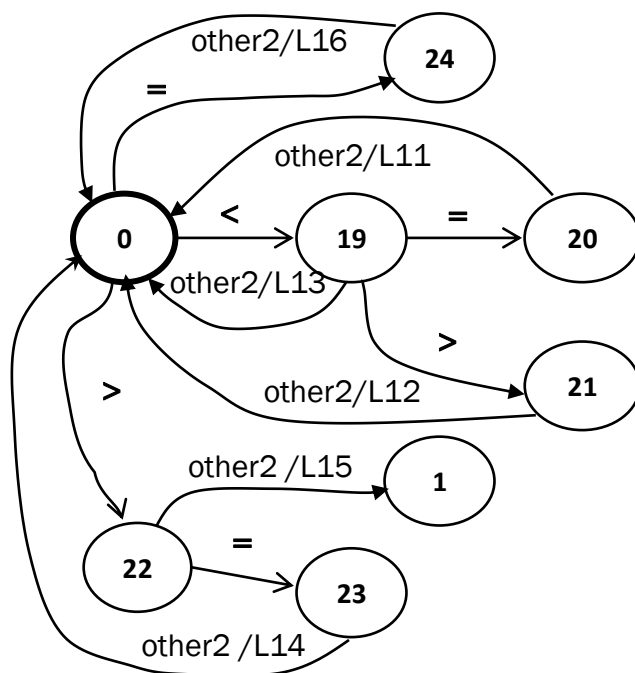


Рисунок 7 – Граф автомата-распознавателя операций отношения

б) Граф для обработки пробелов и табуляций представлен на рисунке 8.

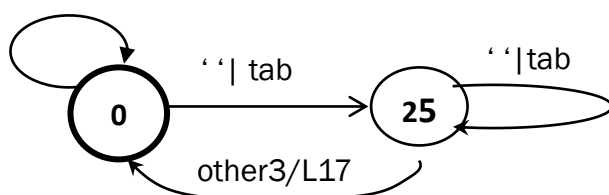


Рисунок 8 – Граф автомата-распознавателя пробелов и табуляций

Множество  $other3 = \{a|b|...|z|0|1|...|9|+|-|*|/|:|<|>|=|. \}$

При построении лексического анализатора языка программирования, все автоматы, распознающие отдельные объекты языка объединяются в один автомат.

Для нашего примера объединенный граф автомата ввиду его громоздкости приводить не будем, приведём таблицу переходов автомата (рис. 9).

Построенный автомат будет распознавать строки следующего вида:  
 WHILE a>5 DO a:=8+b END.\$.

Результат работы распознавателя:

WHILE лексема L1; ' ' лексема L17; а лексема L4; > лексема L15; 5 лексема L5; ' ' лексема L17; DO лексема L2; ' ' лексема L17; а лексема L4; := лексема L8; 8 лексема L5; + лексема L6; b лексема L4; ' ' лексема L17; END лексема L3; . лексема L18.

Таблица идентификаторов, заполняемая на этом этапе для нашего примера, будет иметь вид:

| Лексема | Код лексемы |
|---------|-------------|
| L1      | W           |
| L17     | s           |
| L4      | i           |
| L15     | r           |
| L5      | d           |
| L2      | D           |
| L8      | p           |
| L6      | +           |
| L3      | E           |
| L18     | .           |

На этап синтаксического анализа будет передана строка, в которой удалены пробелы, комментарии и другие не несущие смысловой нагрузки символы, а символы языка заменены кодами лексем (токенами), для нашего примера: **WirdDipd+iE..** Синтаксический анализатор не различает лексем по значению, для него важен только тип лексем, поэтому в переданной строке все идентификаторы представлены одним типом лексем *i* (identification), все отношения – одним типом *r* (relation).

На последующих этапах таблица идентификаторов будет дополнена полями, содержащими данные о типе переменных, ссылки на ячейки памяти, в которых будут храниться значения переменных и т. д.

|    | W    | H    | I    | L    | E    | D    | O    | N    | a...z | 0...9 | +    | -    | *    | /    | :    | =    | <    | >    | .    | " "  | tab  | \$   |    |
|----|------|------|------|------|------|------|------|------|-------|-------|------|------|------|------|------|------|------|------|------|------|------|------|----|
| 0  | 1    |      |      |      | 8    | 6    |      |      | 11    | 12    | 13   | 14   | 17   | 18   | 15   | 24   | 19   | 22   |      | 25   | 25   | K    |    |
| 1  |      | 2    |      |      |      |      |      |      |       |       |      |      |      |      |      |      |      |      |      |      |      |      |    |
| 2  |      |      | 3    |      |      |      |      |      |       |       |      |      |      |      |      |      |      |      |      |      |      |      |    |
| 3  |      |      |      | 4    |      |      |      |      |       |       |      |      |      |      |      |      |      |      |      |      |      |      |    |
| 4  |      |      |      |      | 5    |      |      |      |       |       |      |      |      |      |      |      |      |      |      |      |      |      |    |
| 5  |      |      |      |      |      |      |      |      |       |       |      |      |      |      |      |      |      |      |      |      | 0/L1 |      |    |
| 6  |      |      |      |      |      |      | 7    |      |       |       |      |      |      |      |      |      |      |      |      |      |      |      |    |
| 7  |      |      |      |      |      |      |      |      |       |       |      |      |      |      |      |      |      |      |      |      | 0/L2 |      |    |
| 8  |      |      |      |      |      |      |      | 9    |       |       |      |      |      |      |      |      |      |      |      |      |      |      |    |
| 9  |      |      |      |      | 10   |      |      |      |       |       |      |      |      |      |      |      |      |      |      |      |      |      |    |
| 10 |      |      |      |      |      |      |      |      |       |       |      |      |      |      |      |      |      |      |      |      | 0/L3 |      |    |
| 11 |      |      |      |      |      |      |      |      | 11    | 0/L4  | 0/L4 | 0/L4 | 0/L4 | 0/L4 | 0/L4 | 0/L4 | 0/L4 | 0/L4 | 0/L4 | 0/L4 | 0/L4 | 0/L4 |    |
| 12 |      |      |      |      |      |      |      |      | 0/L5  | 12    | 0/L5 | 0/L5 | 0/L5 | 0/L5 | 0/L5 | 0/L5 | 0/L5 | 0/L5 | 0/L5 | 0/L5 | 0/L5 | 0/L5 |    |
| 13 |      |      |      |      |      |      |      |      | 11/L  | 11/L  |      |      |      |      |      |      |      |      |      |      |      | 11/L |    |
| 14 |      |      |      |      |      |      |      |      | 11/L  | 11/L  |      |      |      |      |      |      |      |      |      |      |      | 11/L |    |
| 15 |      |      |      |      |      |      |      |      |       |       |      |      |      |      |      | 16   |      |      |      |      |      |      |    |
| 16 |      |      |      |      |      |      |      |      | 11/L  | 11/L  |      |      |      |      |      |      |      |      |      |      |      | 11/L |    |
| 17 |      |      |      |      |      |      |      |      | 11/L  | 11/L  |      |      |      |      |      |      |      |      |      |      |      | 11/L |    |
| 18 |      |      |      |      |      |      |      |      | 11/L  | 11/L  |      |      |      |      |      |      |      |      |      |      |      | 11/L |    |
| 19 |      |      |      |      |      |      |      |      | 11/L  | 11/L  |      |      |      |      |      | 20   |      | 21   |      |      |      | 11/L |    |
| 20 |      |      |      |      |      |      |      |      | 11/L  | 11/L  |      |      |      |      |      |      |      |      |      |      |      | 11/L |    |
| 21 |      |      |      |      |      |      |      |      | 11/L  | 11/L  |      |      |      |      |      |      |      |      |      |      |      | 11/L |    |
| 22 |      |      |      |      |      |      |      |      | 11/L  | 11/L  |      |      |      |      |      | 23   |      |      |      |      |      | 11/L |    |
| 23 |      |      |      |      |      |      |      |      | 11/L  | 11/L  |      |      |      |      |      |      |      |      |      |      |      | 11/L |    |
| 24 |      |      |      |      |      |      |      |      | 11/L  | 11/L  |      |      |      |      |      |      |      |      |      |      |      | 11/L |    |
| 25 | 0/L1 | 0/L1 | 0/L1 | 0/L1 | 0/L1 | 0/L1 | 0/L1 | 0/L1 | 0/L1  | 0/L1  | 0/L1 | 0/L1 | 0/L1 | 0/L1 | 0/L1 | 0/L1 | 0/L1 | 0/L1 | 0/L1 | 0/L1 | 0/L1 | 25   | 25 |
|    | 7    | 7    | 7    | 7    | 7    | 7    | 7    | 7    | 7     | 7     | 7    | 7    | 7    | 7    | 7    | 7    | 7    | 7    | 7    | 7    | 7    | 7    |    |

Рисунок 9 – Таблица переходов автомата-распознавателя

## 2.3. Задание к лабораторной работе 2

### 2.3.1. Порядок выполнения работы

1. Ознакомиться с материалами методических указаний.
2. Построить регулярную грамматику для заданного языка (в соответствии с вариантом). При распознавании лексемы выбирается самое короткое слово входной цепочки.
3. Построить конечный автомат для полученной грамматики (в отчете представить граф и таблицу переходов автомата).
4. Ответить на два контрольных вопроса, по заданию преподавателя.
5. Подготовить и защитить отчет по выполнению лабораторной работы.

### 2.3.2. Варианты описаний языков

Варианты заданий выбираются по последнему номеру зачетной книжки (если это цифра 0, выбирается 10 вариант).

- 1) Символ "c", цепочка символов "a" произвольной длины, после которой следует символ "b";  
символ "b", цепочка символов "a" произвольной длины, после которой следует символ "c";  
символ "a", цепочка символов "b" произвольной длины, после которой следуют "a" или "c".
- 2) Цепочка пар символов "ab" произвольной длины, после которой следует "b";  
цепочка пар символов "ba" произвольной длины, после которой следует "cbc";  
символ "c".
- 3) Цепочка из символов abc, произвольное количество символов c, цепочка из символов "abc";  
Символ a, произвольная цепочка чередующихся символов "b", "c", заканчивающаяся cba.
- 4) Три подряд пришедших символа "a" в произвольной цепочке из "ba", после которых следует "b";  
три подряд пришедших символа "b", после которых следует "a";  
три подряд пришедших символа "b", после которых следует "c".
- 5) Произвольное число символов "a" между двумя символами "b";  
символ a, произвольное число символов "b", символ "c";  
три подряд пришедших символа "c".
- 6) Произвольная цепочка из 0 и 1 между \* и \*;  
последовательность двух пар 01;  
символ \*, цепочка из чередующихся 0 и 1, символ ".".
- 7) Произвольная цепочка из 0 и 1, после которой следует ".";  
цепочка четной длины из 01 между двумя символами ".";  
два символа "\*".
- 8) Цепочка четной длины из 0 между двумя 1;  
цепочка нечетной длины из 1 между двумя 0;  
две 1 подряд.
- 9) 1 между двумя цепочками из 0, четной длины каждая, в конце ".";  
0 между двумя цепочками из 1, четной длины каждая, в конце ".".
- 10) Две 1, за которыми следует два 0;  
цепочка пар чередующихся символов 0 и 1 нечетной длины, за которой следует ".".

### 2.3.3. Содержание отчета

- 1) Титульный лист
- 2) Постановка задачи.
- 3) Регулярная грамматика.
- 4) Граф автоматной модели грамматики и соответствующая таблица переходов и выходов.
- 5) Описание тестовых примеров.
- 6) Ответы на контрольные вопросы.
- 7) Выводы.
- 8) Список литературы

Отчет оформляется в текстовом редакторе и предоставляется преподавателю в электронном виде.

### 2.3.4. Контрольные вопросы

1. Дайте определение автомата-распознавателя.
2. Чем недетерминированный КА отличается от детерминированного КА?
3. Как соотносятся регулярные грамматики и регулярные выражения?
4. Дайте определение минимального ДКА.
5. Перечислите основные операции над регулярными языками.
6. Как формально определить принадлежность языка к типу регулярных языков?
7. Что понимается под замыканием регулярных языков относительно операции объединения?
8. Перечислите операции над множеством состояний НКА для его преобразования в ДКА.
9. Для чего используется  $\epsilon$ -замыкание?
10. Перечислите свойства регулярных языков.

## 2.4. Задание к лабораторной работе 3

### 2.4.1. Порядок выполнения работы

1. Ознакомиться с материалами методических указаний.
2. Построить конечный автомат для заданной грамматики (в отчете представить граф и таблицу переходов автомата).
3. Разработать и отладить программу лексического анализатора - препроцессор на основе построенной автоматной модели.
4. Сформировать таблицу идентификаторов.
5. Протестировать разработанную программу.
6. Подготовить и защитить отчет по выполнению лабораторной работы.

### 2.4.2. Варианты описаний языков

Грамматика языка **Lan1**:

$\langle P \rangle \rightarrow \langle B \rangle$

$\langle B \rangle \rightarrow \langle O \rangle | \langle O \rangle ; \langle B \rangle$

$\langle O \rangle \rightarrow \langle V \rangle := \langle E \rangle$

$\langle O \rangle \rightarrow \text{If } \langle E \rangle \text{ Then } \langle O \rangle \text{ Endif}$

$\langle E \rangle \rightarrow \langle F \rangle | \langle E \rangle \# \langle F \rangle$

$\langle F \rangle \rightarrow \langle V \rangle | \langle F \rangle \& \langle V \rangle$

<V>→<I>|<C>  
 <C>→<H>  
 <H>→<D>|<H><D>  
 <D>→0|1|2|3|4|5|6|7|8|9  
 <I>→<W>|<I><W>|<I><D>  
 <W>→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

### Грамматика языка Lan2:

<P>→{<B>}  
 <B>→<O>|<O>;< B >  
 <O>→<V>:=<E>  
 <O>→while <V> {<B>}  
 <E>→<F>|<E>#<F>|!<F>  
 <F>→< V >|<F>&< V >  
 < V >→<I>|<C>  
 <C>→<Z>  
 <Z>→<H>  
 <H>→<D>|<H><D>  
 <D>→0|1|2|3|4|5|6|7|8|9  
 <I>→<W>|<I><W>|<I><D>  
 <W>→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

### Грамматика языка Lan3:

<P>→var <O> <B>  
 <O>→<O>|<O>;<O>  
 <O>→<I>  
 <B>→begin <S> end  
 <S>→<O>|<S>;<O>|  
 <O>→<V>:=<E>  
 <O>→for <V>:=<E> to <Z> do <B>  
 <E>→<F>|<E>#<F>|!<F>  
 <F>→< V >|<F>&< V >  
 < V >→<I>|<C>  
 <C>→<Z>  
 <Z>→<H>  
 <H>→<D>|<H><D>  
 <D>→0|1|2|3|4|5|6|7|8|9  
 <I>→<W>|<I><W>  
 <W>→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

### Грамматика языка Lan4:

<P>→<B>  
 <B>→<O>|<O>;< B >  
 <O>→<V>:=<E>  
 <O>→if <V> ? <O> : <O>  
 <E>→<F>|<E>#<F>  
 <F>→<V>|<F>&<V>  
 <V>→<I>|<C>|(<E>)  
 <C>→<Z>

<Z>→<H>  
 <H>→<D>|<H><D>  
 <D>→0|1|2|3|4|5|6|7|8|9  
 <I> →<W>|<I><W>|<I><D>  
 <W>→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

Грамматика языка **Lan5**:

<P>→<B>  
 <B>→<O>|<O>;< B >  
 <O>→<V>:=<E>  
 <O>→ if <V> ? <O> : <O>  
 <E>→<F>|<E>#<F>  
 <F>→<V>|<F>&<V>  
 <V>→<I>|<C>|(<E>)  
 <C>→<Z>  
 <Z>→<H>|(<H>|<-H>  
 <H>→<D>|<H><D>  
 <D>→0|1|2|3|4|5|6|7|8|9  
 <I> →<W>|<I><W>  
 <W>→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

Грамматика языка **Lan6**:

<P>→var <O> <блок>  
 <O>→<S>|<S>;<O>  
 <S>→ byte <I>| word <I>  
 <B>→<R>|<R>;< B >  
 <R>→<V>:=<E>  
 <R>→If <V> THEN <R> Endif  
 <E>→<F>|<E>#<F>  
 <F>→<V>|<F>&<V>  
 <V>→<I>|<C>| ! (<E>)  
 <C>→<Z>  
 <Z>→<H>  
 <H>→<D>|<H><D>  
 <D>→0|1|2|3|4|5|6|7|8|9  
 <I> →<W>|<I><W>|<I><D>  
 <W>→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

Грамматика языка **Lan7**:

<P>→<B>.  
 <B>→<O>|<O>;< B >  
 <O>→<V>:=<E>  
 <O>→If <V> THEN <O> End  
 <E>→<F>|<E>+<F>  
 <F>→<V>|<F>-<V>|<F>\*<V>  
 <V>→<I>|<C>  
 <C>→<Z>  
 <Z>→<H>  
 <H>→<D>|<H><D>

$\langle D \rangle \rightarrow 0|1|2|3|4|5|6|7|8|9$   
 $\langle I \rangle \rightarrow \langle W \rangle | \langle I \rangle \langle W \rangle$   
 $\langle W \rangle \rightarrow A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z$

Грамматика языка **Lan8**:

$\langle P \rangle \rightarrow \{ \langle B \rangle \}$   
 $\langle B \rangle \rightarrow \langle O \rangle | \langle O \rangle ; \langle B \rangle$   
 $\langle O \rangle \rightarrow \langle V \rangle := \langle E \rangle$   
 $\langle O \rangle \rightarrow \text{while } \langle V \rangle \text{ do } \langle B \rangle \text{ loop}$   
 $\langle E \rangle \rightarrow \langle F \rangle | \langle E \rangle * \langle F \rangle | ! \langle F \rangle$   
 $\langle F \rangle \rightarrow \langle V \rangle | \langle F \rangle + \langle V \rangle$   
 $\langle V \rangle \rightarrow \langle I \rangle | \langle C \rangle$   
 $\langle C \rangle \rightarrow \langle Z \rangle$   
 $\langle Z \rangle \rightarrow \langle H \rangle$   
 $\langle H \rangle \rightarrow \langle D \rangle | \langle H \rangle \langle D \rangle$   
 $\langle D \rangle \rightarrow 0|1|2|3|4|5|6|7|8|9$   
 $\langle I \rangle \rightarrow \langle W \rangle | \langle I \rangle \langle W \rangle$   
 $\langle W \rangle \rightarrow a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z$

Грамматика языка **Lan9**:

$\langle P \rangle \rightarrow \text{var } \langle O \rangle \langle B \rangle$   
 $\langle O \rangle \rightarrow \langle O \rangle | \langle O \rangle ; \langle O \rangle$   
 $\langle O \rangle \rightarrow \langle I \rangle$   
 $\langle B \rangle \rightarrow \text{begin } \langle S \rangle \text{ end}$   
 $\langle S \rangle \rightarrow \langle O \rangle | \langle S \rangle ; \langle O \rangle |$   
 $\langle O \rangle \rightarrow \langle V \rangle := \langle E \rangle$   
 $\langle O \rangle \rightarrow \text{for } \langle V \rangle := \langle E \rangle \text{ to } \langle Z \rangle \text{ do } \langle B \rangle$   
 $\langle E \rangle \rightarrow \langle F \rangle | \langle E \rangle \# \langle F \rangle | ! \langle F \rangle$   
 $\langle F \rangle \rightarrow \langle V \rangle | \langle F \rangle \& \langle V \rangle | \langle F \rangle \oplus \langle V \rangle$   
  
 $\langle V \rangle \rightarrow \langle I \rangle | \langle C \rangle$   
 $\langle C \rangle \rightarrow \langle Z \rangle$   
 $\langle Z \rangle \rightarrow \langle H \rangle$   
 $\langle H \rangle \rightarrow \langle D \rangle | \langle H \rangle \langle D \rangle$   
 $\langle D \rangle \rightarrow 0|1|2|3|4|5|6|7|8|9$   
 $\langle I \rangle \rightarrow \langle W \rangle | \langle I \rangle \langle W \rangle | \langle I \rangle \langle D \rangle$   
 $\langle W \rangle \rightarrow A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z$

Грамматика языка **Lan10**:

$\langle P \rangle \rightarrow \langle B \rangle .$   
 $\langle B \rangle \rightarrow \langle O \rangle | \langle O \rangle ; \langle B \rangle$   
 $\langle O \rangle \rightarrow \langle V \rangle := \langle E \rangle$   
 $\langle O \rangle \rightarrow \text{if } \langle V \rangle \text{ then } \langle O \rangle : \langle O \rangle \text{ end}$   
 $\langle E \rangle \rightarrow \langle F \rangle | \langle E \rangle \oplus \langle F \rangle$   
 $\langle F \rangle \rightarrow \langle V \rangle | \langle F \rangle \& \langle V \rangle$   
 $\langle V \rangle \rightarrow \langle I \rangle | \langle C \rangle | (\langle E \rangle)$   
 $\langle C \rangle \rightarrow \langle Z \rangle$   
 $\langle Z \rangle \rightarrow \langle H \rangle$   
 $\langle H \rangle \rightarrow \langle D \rangle | \langle H \rangle \langle D \rangle$

<D>→0|1|2|3|4|5|6|7|8|9

<I> →<W>|<I><W>

<W>→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

### 2.4.3. Содержание отчета

- 1) Титульный лист.
- 2) Постановка задачи.
- 3) Граф автоматной модели грамматики и соответствующая таблица переходов и выходов.
- 5) Таблица идентификаторов.
- 6) Текст программы.
- 7) Описание тестовых примеров.
- 8) Выводы.
- 9) Список литературы.

Отчет оформляется в текстовом редакторе и предоставляется преподавателю в электронном виде.

### 2.4.4. Варианты заданий

Задания выбираются по номеру в списке подгруппы согласно таблице 1.

Таблица 1. Варианты заданий

|          |     |    |    |    |    |    |    |    |    |    |
|----------|-----|----|----|----|----|----|----|----|----|----|
| Вариант  | 1   | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 |
| Задание* | L10 | L9 | L8 | L7 | L6 | L5 | L4 | L3 | L2 | L1 |

\*L1 - Грамматика языка Lan1 и т.д.