**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**



ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

**Лабораторная работа №1**

по дисциплине: Теория автоматов и формальных языков

тема: «Формальные грамматики. Выводы»

Выполнил: ст. группы ПВ-233

Мовчан Антон Юрьевич

Проверили:

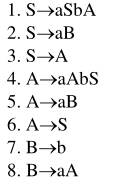
ст. пр. Рязанов Юрий Дмитриевич

Белгород 2025 г.

**Лабораторная работа №1**

Цель работы: изучить основные понятия теории формальных языков и грамматик.

**Вариант 8**

****

1. Найти терминальную цепочку α, |α| > 10, для которой существует не менее двух левых выводов в заданной КС-грамматике (см. варианты заданий). Записать различные левые выводы этой цепочки. Построить деревья вывода. Определить последовательности правил, применяемые при этих выводах.

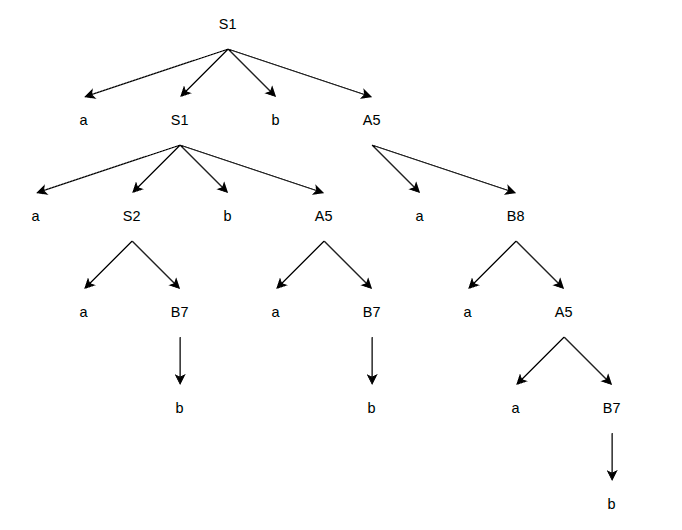
Левый вывод 1:

=> => => => => => => => => =>

Терминальная цепочка:

Последовательность правил: 1 1 2 7 5 7 5 8 5 7

Дерево вывода:



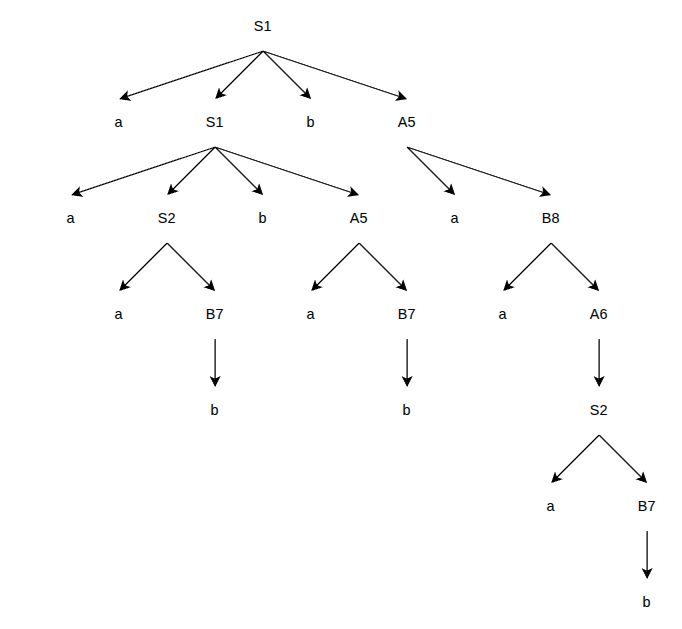
Левый вывод 2:

=> => => => => => => => => => =>

Терминальная цепочка:

Последовательность правил: 1 1 2 7 5 7 5 8 6 2 7

Дерево вывода:



2. Написать программу, которая определяет, можно ли применить заданную последовательность правил при левом выводе терминальной цепочки в заданной КС-грамматике, формирует левый вывод и линейную скобочную форму дерева вывода.

Обработать программой последовательности правил, полученные в п.1.

Примечание. Если к нетерминалу А в процессе вывода применяется правило с номером n, то в выводе и в линейной скобочной форме дерева вывода после нетерминала А должен быть символ с кодом n.

Исходный код:

#include *<bits/stdc++.h>*

**using** **namespace** **std**;

*/\* ---------- 1. Структура правила и инициализация грамматики ---------- \*/*

**struct** **Rule** {

string lhs; *// левые члены (непустой символ)*

string rhs; *// правые члены*

int num; *// номер правила*

};

**static** **const** vector<Rule> rules = {

{"S", "aSbA", 1},

{"S", "aB", 2},

{"S", "A", 3},

{"A", "aAbS", 4},

{"A", "aB", 5},

{"A", "S", 6},

{"B", "b", 7},

{"B", "aA", 8}

};

*/\* ---------- 2. Узел дерева вывода ---------- \*/*

**struct** **Node** {

string symbol; *// символ*

bool terminal; *// true – терминал*

int ruleNum = -1; *// номер правила, которым он был развернут*

vector<Node\*> children; *// потомки*

Node(**const** string& s, bool t = true) : symbol(s), terminal(t) {}

};

*/\* ---------- 3. Поиск правила по номеру ---------- \*/*

**const** Rule\* findRule(int num) {

**for** (**const** **auto**& r : rules)

**if** (r.num == num) **return** &r;

**return** **nullptr**;

}

*/\* ---------- 4. Функция вывода лексической скобочной формы ---------- \*/*

void printNode(Node\* n, string &out) {

**if** (n->terminal) {

out += n->symbol;

**return**;

}

out += '(' + n->symbol + to\_string(n->ruleNum);

**for** (Node\* c : n->children) printNode(c, out);

out += ')';

}

*/\* ---------- 5. Главная функция ---------- \*/*

int main() {

ios::sync\_with\_stdio(false);

cin.tie(**nullptr**);

*/\* ---------- ввод последовательности номеров правил ---------- \*/*

vector<int> seqRules;

int x;

**while** (cin >> x) seqRules.push\_back(x);

*/\* ---------- инициализация дерева и списка текущих нод ---------- \*/*

Node\* root = **new** Node("S", false);

vector<Node\*> curList = {root};

bool ok = true;

**for** (int ruleNum : seqRules) {

**const** Rule\* r = findRule(ruleNum);

**if** (!r) { ok = false; **break**; }

*/\* поиск левого самого первого нерегулярного символа \*/*

size\_t pos = 0;

**while** (pos < curList.size() && curList[pos]->terminal) ++pos;

**if** (pos == curList.size()) { ok = false; **break**; }

Node\* target = curList[pos];

**if** (target->symbol != r->lhs) { ok = false; **break**; }

*/\* развернуть правило \*/*

target->ruleNum = r->num;

target->terminal = false;

target->children.clear();

*/\* создаём потомков и обновляем список текущих нод \*/*

vector<Node\*> newNodes;

**for** (char ch : r->rhs) {

**if** (isupper(ch)) { *// нетерминал*

Node\* child = **new** Node(string(1, ch), false);

target->children.push\_back(child);

newNodes.push\_back(child);

} **else** { *// терминал*

Node\* child = **new** Node(string(1, ch), true);

target->children.push\_back(child);

newNodes.push\_back(child);

}

}

curList.erase(curList.begin() + pos, curList.begin() + pos + 1);

curList.insert(curList.begin() + pos, newNodes.begin(), newNodes.end());

}

*/\* ---------- проверка, что все стали терминалами ---------- \*/*

**for** (Node\* n : curList)

**if** (!n->terminal) ok = false;

*/\* ---------- вывод результата ---------- \*/*

**if** (!ok) {

cout << "Невозможно применить заданную последовательность правил.**\n**";

**return** 0;

}

*/\* строка левого вывода \*/*

string derivation;

**for** (Node\* n : curList) derivation += n->symbol;

cout << "Левый вывод: " << derivation << '\n';

*/\* линейная скобочная форма \*/*

string linear;

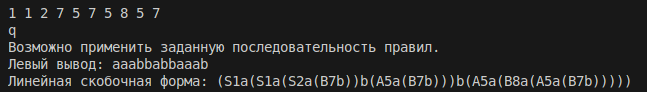
printNode(root, linear);

cout << "Линейная скобочная форма: " << linear << '\n';

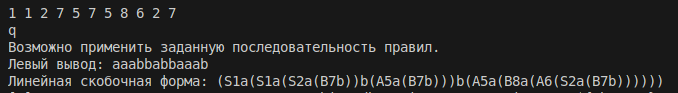
**return** 0;

}

Результат выполнения программы, для п1:



Результат выполнения программы, для п2:



3. Найти последовательность правил р, |р| > 10, которую можно применить при произвольном выводе терминальной цепочки, но нельзя применить при левом или правом выводе в заданной КС-грамматике (см. варианты заданий). Записать вывод v, в процессе которого применяется последовательность правил р. Построить дерево вывода.

Записать левый и правый выводы, эквивалентные выводу v.