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.

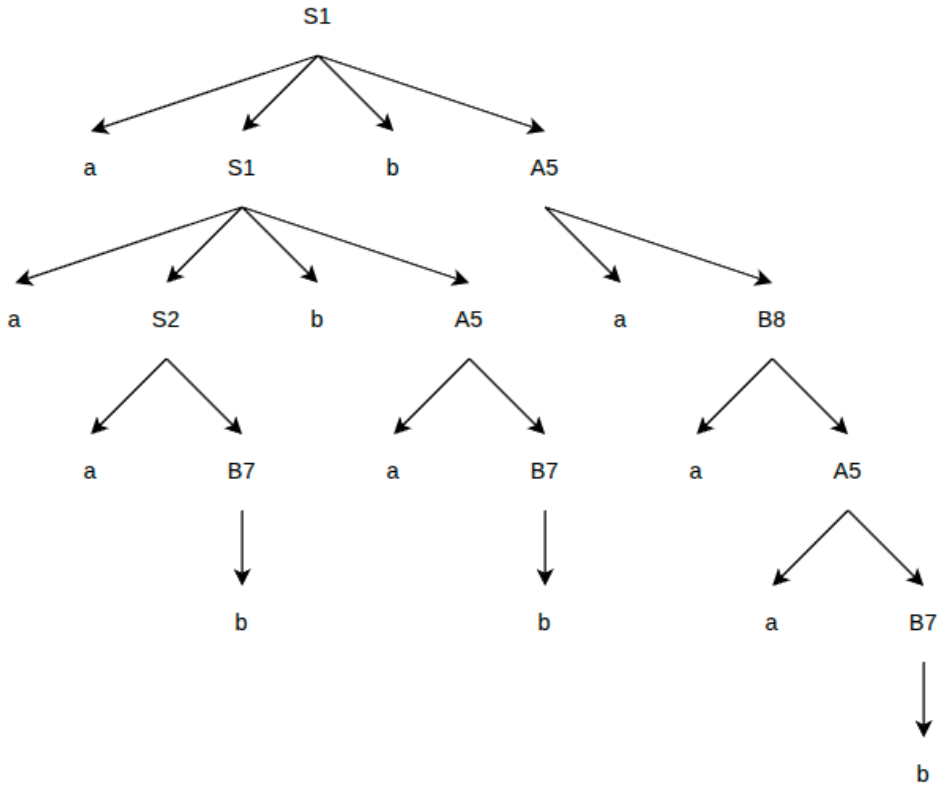
Вариант 8

1. S→aSbA
2. S→aB
3. S→A
4. A→aAbS
5. A→aB
6. A→S
7. B→b
8. B→aA

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.

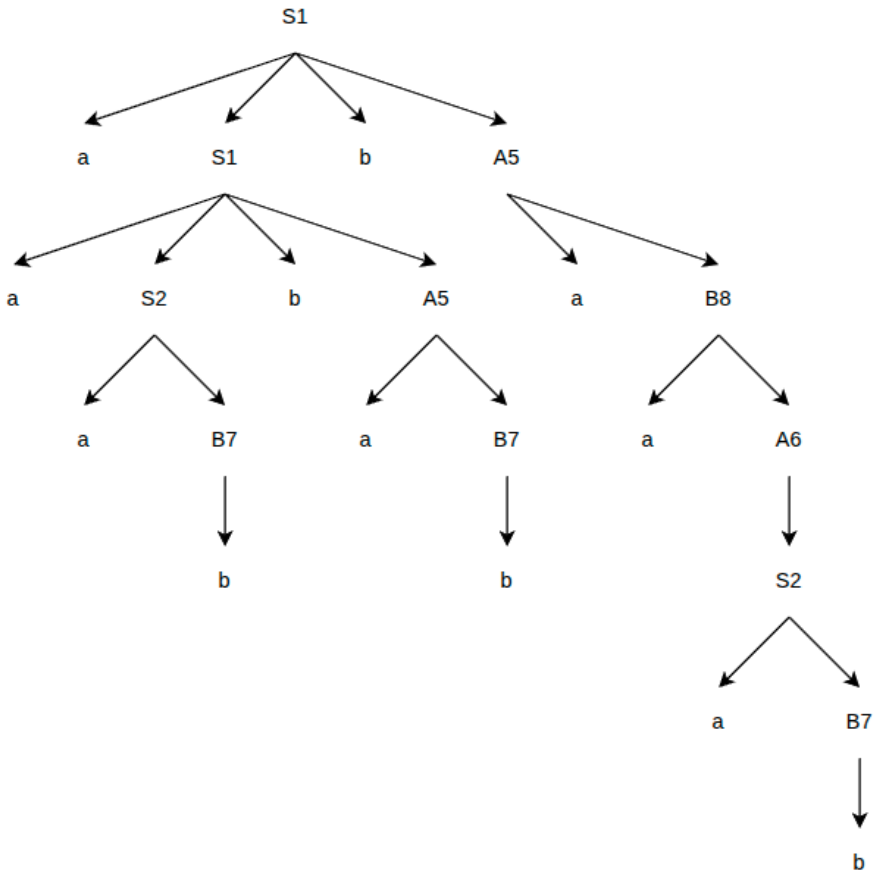
Левый вывод 1:
$S_1 => aS_1bA => aaS_2bAbA => aaaB_7bAbA => aaabbA_5bA => aaabbaB_7bA => aaabbabbA_5 => aaabbabbaB_8 => aaabbabbaaA_5 => aaabbabbaaaB_7 => aaabbabbaaab$

Терминальная цепочка: *aaabbabbaaab*
Последовательность правил: 1 1 2 7 5 7 5 8 5 7
Дерево вывода:



Левый вывод 2:
$S_1 => a\,S_1\,bA => aa\,S_2\,bAbA => aaa\,B_7\,bAbA => aaabb\,A_5\,bA => aaabba\,B_7\,bA => aaabbabb\,A_5 =>$
$aaabbabba\,B_8 => aaabbabbaa\,A_6 => aabbabbaa\,S_2 => aaabbabbaaa\,B_7 => aaabbabbaaab$

Терминальная цепочка: *aaabbabbaaab*
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 A в процессе вывода применяется правило с номером n, то в выводе и в линейной скобочной форме дерева вывода после нетерминала A 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, int r = -1) : symbol(s), terminal(t), ruleNum(r) {}
};

/* ---------- 3. Поиск правила по номеру ---------- */
const Rule *findRule(int num)
{
    for (const auto &r : rules)
        if (r.num == num)
            return &r;
    return nullptr;
}

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 += ')';
}

string getCurrentState(vector<Node *> &curList, int appliedRule)
{
    bool added = false;
    string derivation;
    for (Node *n : curList)
    {
        derivation += n->symbol;
        if (!added && !n->terminal && (appliedRule != -1))
        {

            derivation += to_string(appliedRule);
            added = true;
        }
    }
    return derivation;
}

/* ---------- 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 */
    /* строка левого вывода */
    string deriv = getCurrentState(curList, r->num);
    cout << deriv << " => ";

    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, r->num);
            target->children.push_back(child);
            newNodes.push_back(child);
        }
        else
        { // терминал
            Node *child = new Node(string(1, ch), true, r->num);
            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;
}

/* линейная скобочная форма */
string deriv = getCurrentState(curList, -1);
```

```
    cout << deriv << '\n';
    string linear;
    printNode(root, linear);
    cout << linear << '\n';


    return 0;
}
```

Результат выполнения программы, для п1:

```
1 1 2 7 5 7 5 8 5 7
q
S1 => aS1bA => aaS2bAbA => aaaB7bAbA => aaabbA5bA => aaabbaB7bA => aaabbabbA5 => aaabbabbaB8 => aaabbabbaaA5 => aaabbabbaaaB7 => aaabbabbaaab
(S1a(S1a(S2a(B7b))b(A5a(B7b)))b(A5a(B8a(A5a(B7b)))))
```

Результат выполнения программы, для п2:

```
1 1 2 7 5 7 5 8 6 2 7
q
S1 => aS1bA => aaS2bAbA => aaaB7bAbA => aaabbA5bA => aaabbaB7bA => aaabbabbA5 => aaabbabbaB8 => aaabbabbaaA6 => aaabbabbaaS2 => aaabbabbaaaB7 => aaabbabbaaab
(S1a(S1a(S2a(B7b))b(A5a(B7b)))b(A5a(B8a(A6(S2a(B7b))))))
```

3. Найти последовательность правил p, |p| > 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 p. Построить дерево вывода.

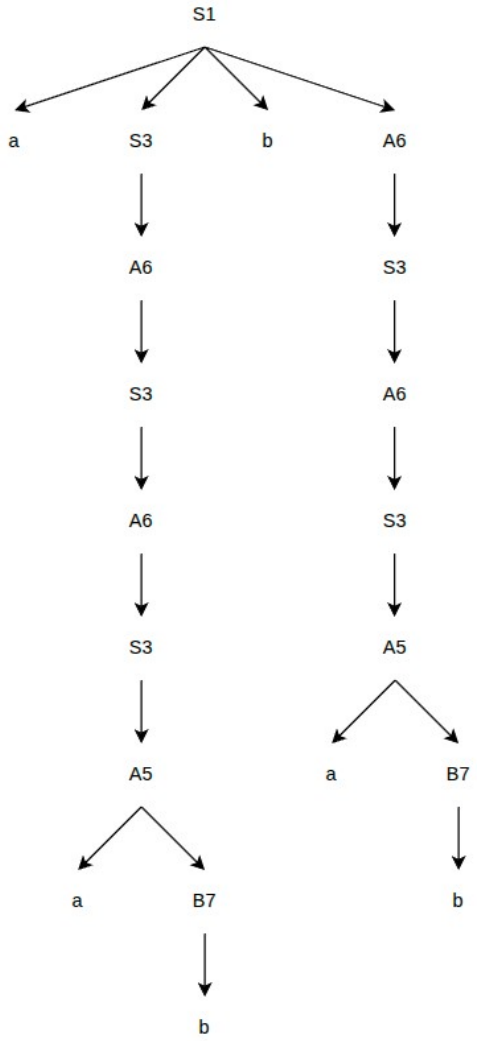
Записать левый и правый выводы, эквивалентные выводу v.

Произвольный: вывод:

$S_1 => a\,S_3\,bA => aAb\,A_6 => a\,A_6\,bS => aSb\,S_3 => a\,S_3\,bA => aAb\,a_6 => a\,A_6\,bS => aSb\,S_3 => a\,S_3\,bA => aAb\,A_5 => a\,A_5\,baB => aaBba\,B_7 => aa\,B_7\,bab => aabbab$

Последовательность правил p: 1 3 6 6 3 3 6 6 3 3 5 5 7 7
Терминальная цепочка: *aabbab*

Дерево вывода:



Левый вывод:

$S_1 => a\,S_3\,bA => a\,A_6\,bA => a\,S_3\,bA => a\,A_6\,bA => a\,S_3\,bA => a\,A_5\,bA => aa\,B_7\,bA => aabb\,A_6 =>$
$aabb\,S_3 => aabb\,A_6 => aabb\,S_3 => aabb\,A_5 => aabba\,B_7 => aabbab$

Последовательность правил: 1 3 6 3 6 3 5 7 6 3 6 3 5 7
Терминальная цепочка: *aabbab*

Правый вывод:

$S_1 => aSb\,A_6 => aSb\,S_3 => aSb\,A_6 => aSb\,S_3 => aSb\,A_5 => aSba\,B_7 => a\,S_3\,bab => a\,A_6\,bab => a\,S_3\,bab$
$=> a\,A_6\,bab => a\,S_3\,bab => a\,A_5\,bab => aa\,B_7\,bab => aabbab$

Последовательность правил: 1 6 3 6 3 5 7 3 6 3 6 3 5 7
Терминальная цепочка: *aabbab*

4. Написать программу, которая определяет, можно ли применить заданную последовательность правил p при выводе терминальной цепочки в заданной КС-грамматике и формирует линейную скобочную форму дерева вывода. Если последовательность правил p можно применить при выводе v терминальной цепочки, то программа должна вывести последовательность правил, применяемую при левом выводе, эквивалентном выводу v.

Обработать программой последовательность правил, найденную в п.3.

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

Исходный код:

```
#include <bits/stdc++.h>
using namespace std;

map<int, pair<char, string>> rules = {
    {1, {'S', "aSbA"}},
    {2, {'S', "aB"}},
    {3, {'S', "A"}},
    {4, {'A', "aAbS"}},
    {5, {'A', "aB"}},
    {6, {'A', "S"}},
    {7, {'B', "b"}},
    {8, {'B', "aA"}}
};

bool isNonterm(char c) { return c >= 'A' && c <= 'Z'; }

void updateLsf(std::string &lsf, char lhs, std::string &rhs, int rule)
{
    for (int j = 0; j < (int)lsf.size(); j++)
    {
        if (lsf[j] == lhs && (j + 1 == lsf.size() || !isdigit(lsf[j + 1])))
        {
            // формируем раскрытие: N<rule>(rhs)
            string expanded;
            for (char c : rhs)
            {
                expanded += c; // терминал
            }
            lsf = lsf.substr(0, j) + string(1, lhs) + to_string(rule) + "(" + expanded + ")" + lsf.substr(j + 1);
            break;
        }
    }
}
// применяет правило к текущей цепочке и одновременно обновляет ЛСФ
string apply_rule_once(string cur, int rule, string &lsf) {
    char lhs = rules[rule].first;
    string rhs = rules[rule].second;

    // ищем первый подходящий нетерминал и строим новую цепочку
    int found = -1;
    for (int i = 0; i < (int)cur.size(); i++) {
        if (cur[i] == lhs) {
            found = i;
            break;
        }
    }

    if (found== -1) {
        cout << "Невозможно применить правило" << endl;
        return "";
    }

    string step = cur.substr(0, found+1) + to_string(rule) + cur.substr(found+1);
```

```
    cout << step << " => ";
    cur = cur.substr(0, found) + rhs + cur.substr(found + 1);


    // теперь аналогично заменить в LSF
    // находим первый тот же нетерминал в LSF
    updateLsf(lsf, lhs, rhs, rule);

    return cur;
}

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    // читаем последовательность правил
    string line;
    getline(cin, line);
    vector<int> rules;
    stringstream ss(line);
    int r;
    while (ss >> r) rules.push_back(r);

    // начальное состояние
    string cur = "S";
    string lsf = "S";

    // пошаговое применение правил
    for (int rule : rules) {
        cur = apply_rule_once(cur, rule, lsf);
        if (cur == "") {
            return 0;
        }
    }
    // финальная цепочка
    cout << cur << "\n";

    // Линейная скобочная форма (строилась сразу!)
    cout << "\nЛинейная скобочная форма дерева вывода:\n" << lsf << "\n";

    cout << "\nПоследовательность правил при левом выводе:\n";
    for (char c : lsf) if (isdigit(c)) cout << c << " ";
    cout << "\n";
    for (char c : lsf) if (c >= 'a' && c <= 'z') cout << c << " ";
    cout << "\n";

    return 0;
}
```

Результат выполнения программы:

```
1 3 6 6 3 3 6 6 3 3 5 5 7 7
S1 => aS3bA => aA6bA => aSbA6 => aS3bS => aAbS3 => aA6bA => aSbA6 => aS3bS => aAbS3 => aA5bA => aaBbA5 => aaB7baB => aabbaB7 => aabbab

Линейная скобочная форма дерева вывода:
S1(aS3(A6(S3(A6(S3(A5(aB7(b)))))))bA6(S3(A6(S3(A5(aB7(b)))))))

Последовательность правил при левом выводе:
1 3 6 3 6 3 5 7 6 3 6 3 5 7
a a b b a b
```

Вывод: в ходе выполнения л.р. я изучил основные понятия теории формальных языков и грамматик.