

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет
«Московский институт электронной техники»
институт системной и программной инженерии
и информационных технологий»

**Курс «Теория алгоритмических
языков и компиляторов»**

Лабораторная работа № 3

Недетерминированные магазинные автоматы.

Выполнил:

Студент группы ПИН-41Д

Джугели Дмитрий Александрович

Москва, 2025

Задание на лабораторную работу

Написать программу, реализующую работу недетерминированного магазинного автомата.

Входные данные:

1. Текстовый файл с описанием грамматики, для которой строится магазинный автомат.
Каждая строка в файле может задавать несколько правил грамматики с одинаковой левой частью. В этом случае правила отделяются друг от друга символом ‘|’.

Для отделения левой части правила от правой используется символ ‘>’. В одной строке может быть несколько символов ‘>’. В этом случае первый из них трактуется, как символ, отделяющий правую и левую части продукции, все последующие – как терминальный символ.

Все нетерминалы задаются с помощью прописных букв латинского алфавита.

Все символы, не описанные выше, являются терминальными символами.

Правил в грамматике (а значит и во входном файле) – неограниченное количество.

Рекомендуется считать пробельные символы в файле незначащими, а для терминала «пробел» использовать какой-нибудь другой символ (например, ‘~’, либо заключать пробел в апострофы).

Пример входного файла

E > □E + T | T

T > □T * F | F

F > (E) | a | ~

2. Стока символов, которую нужно проанализировать с помощью построенного автомата и дать заключение о допустимости (или недопустимости) автоматом данной цепочки символов.

Выходные данные

1. На оценку «удовлетворительно»: заключение о допустимости (или недопустимости) автоматом цепочки символов.
2. На оценку «отлично»: значение множеств P , Z и т.д.; список команд (1) – (4) (см раздел «Построение магазинного автомата»); цепочка конфигураций магазинного автомата, полученная в процессе его работы; заключение о допустимости (или недопустимости) автоматом цепочки символов.

Код:

```
#include <string>
#include <vector>
#include <algorithm>
#include <iostream>
#include <fstream>
#include <map>
#include <set>
#include <regex>

using namespace std;

struct Link
{
    char s;
    string inp;
    string stack; // состояние магазина сейчас
    int index;
    bool term; //менять ли выбор ветки

    Link(char s, string p, string h, bool t) : s(s), inp(p), stack(h), index(-1), term(t)
}

    Link(char s, string p, string h) : s(s), inp(p), stack(h), index(-1), term(false) {}

struct Fargs
{
    char s;
    char p;
    char h;

    Fargs(char s, char p, char h) : s(s), p(p), h(h) { }

};

struct Value
{
    char s;
    string c;

    Value(char s, string c) : s(s), c(c) { }

};

struct Command
{
    Fargs f;
    vector<Value> values;

    Command(Fargs f, vector<Value> v) : f(f), values(v) { }

};

class Storage
{
private:
    ifstream file;
    set<char> P;
    set<char> H;
    char s0 = '0', h0 = '|', empty_symbol = '\0';
    vector<Command> commands;
    vector<Link> chain;
```

```

public:

    Storage(const char* filename) : file(filename)
    {
        if (!file.is_open())
            throw runtime_error("Не удалось открыть файл для чтения\n");
        string tmpStr;
        int vsize;
        const regex exp("[[:upper:]]>[[:print:]]+");
        smatch match;
        while (getline(file, tmpStr))
        {
            if (tmpStr.size() == 0)
                continue;
            if (!regex_match(tmpStr, match, exp) || tmpStr[tmpStr.size() - 1] == '|'
|| tmpStr[2] == '|')
            {
                throw runtime_error("Не удалось распознать содержимое файла\n");
            }
            else
            {
                H.insert(match[1].str()[0]);
                commands.push_back(Command(Fargs(s0, empty_symbol,
match[1].str()[0]), vector<Value>()));
                commands[commands.size() - 1].values.push_back(Value(s0, ""));
                for (int i = 0; i < match[2].str().size(); i++)
                {
                    if (match[2].str()[i] == '|')
                    {
                        if (match[2].str()[i - 1] != '|')
                            commands[commands.size() -
1].values.push_back(Value(s0, ""));
                    }
                    else
                    {
                        P.insert(match[2].str()[i]);
                        vsize = commands[commands.size() - 1].values.size();
                        commands[commands.size() - 1].values[vsize -
1].c.push_back(match[2].str()[i]);
                    }
                }

                for (int i = 0; i < commands[commands.size() - 1].values.size();
i++)
                    reverse(commands[commands.size() - 1].values[i].c.begin(),
commands[commands.size() - 1].values[i].c.end());
            }
            for (const auto &c : H)
                P.erase(c);
            for (const auto& c : P)
                commands.push_back(Command(Fargs(s0, c, c), vector<Value>({ Value(s0,
"\0") })));
            commands.push_back(Command(Fargs(s0, empty_symbol, h0), vector<Value>({ Value(s0, "\0") })));
        }

        void showInfo()
        {
            cout << "Входной алфавит:\nP = ";
            for (const auto& c : P)
                cout << c << ", ";
            cout << "\b\b}\n\n";
            cout << "Алфавит магазинных символов:\nZ = ";
            for (const auto& c : H)

```

```

        cout << c << ", ";
    for (const auto& c : P)
        cout << c << ", ";
    cout << "h0}\n\n";

    cout << "Список команд:\n";
    for (const auto& c : commands)
    {
        cout << "f(s" << c.f.s << ", ";
        if (c.f.p == empty_symbol)
            cout << "lambda";
        else
            cout << c.f.p;
        cout << ",";
        if (c.f.h == h0)
            cout << "h0";
        else
            cout << c.f.h;
        cout << ") = {";
        for (Value v : c.values)
        {
            cout << "(s" << v.s << ", ";
            if (v.c[0] == empty_symbol)
                cout << "lambda";
            else
                cout << v.c;
            cout << "); ";
        }
        cout << "\b\b}\n";
    }
    cout << endl;
}

void showChain()
{
    cout << "\nЦепочка конфигураций: \n";
    for (const auto& link : chain)
        cout << "(s" << link.s << ", " << ((link.inp.size() == 0) ? "lambda" :
link.inp) << ", h0" << link.stack << ") |- ";
        cout << "(s0, lambda, lambda)" << endl;
}

bool push_link()
{
    int ch_size = chain.size();
    int mag_size, j, i;
    for (i = 0; i < commands.size(); i++) {
        mag_size = chain[ch_size - 1].stack.size();
        if (chain[chain.size() - 1].inp.size() != 0 && chain[chain.size() - 1].stack.size() != 0 && chain[ch_size - 1].s == commands[i].f.s && (chain[ch_size - 1].inp[0] == commands[i].f.p || empty_symbol == commands[i].f.p) && chain[ch_size - 1].stack[mag_size - 1] == commands[i].f.h)
        {
            for (j = 0; j < commands[i].values.size(); j++)
            {
                if (commands[i].f.p == empty_symbol)
                {
                    chain.push_back(Link(commands[i].values[j].s,
chain[ch_size - 1].inp, string(chain[ch_size - 1].stack)));
                }
                else
                {
                    chain.push_back(Link(commands[i].values[j].s,
chain[ch_size - 1].inp, string(chain[ch_size - 1].stack)));
                }
            }
        }
    }
}

```

```

        reverse(chain[ch_size].inp.begin(),
chain[ch_size].inp.end());                                chain[ch_size].inp.pop_back();
reverse(chain[ch_size].inp.begin(),
chain[ch_size].inp.end());                                }
}

chain[ch_size].stack.pop_back();
chain[ch_size].stack += commands[i].values[j].c;

if (chain[ch_size].inp.size() < chain[ch_size].stack.size())
{
    chain.pop_back();
    chain.pop_back();
    return false;
}
else
{
    if (chain[chain.size() - 1].inp.size() == 0 &&
chain[chain.size() - 1].stack.size() == 0 || push_link())
        return true;
}
}

if (i == commands.size())
{
    chain.pop_back();
    return false;
}
}

bool check_line(const string& str)
{
    if (commands[0].values.size() == 1)
        chain.push_back(Link(s0, str, string(""), false));
    else
        chain.push_back(Link(s0, str, string(""), true));

    chain[0].stack.push_back(commands[0].f.h);

    bool res = push_link();
    if (res)
    {
        cout << "Валидная строка\n";
        showChain();
    }
    else {
        cout << "Невалидная строка\n";
    }
    chain.clear();
    return res;
}

~Storage()
{
    file.close();
}
};

int main()
{
    setlocale(LC_ALL, "Russian");
    string str;
    try {

```

```
Storage strg("C:\\\\Users\\\\ant_daddy\\\\OneDrive\\\\Рабочий  
стол\\\\ТАЯК\\\\ТАЯК\\\\Laba3\\\\test2.txt");  
strg.showInfo();  
  
while (true)  
{  
    cout << "Введите строку: \\n";  
    getline(cin, str);  
    strg.check_line(str);  
    cout << endl;  
}  
}  
catch (const exception& err) {  
    cerr << err.what() << endl;  
}  
return 0;  
}
```

Файл Правка Вид Git Проект Сборка Отладка Тест Анализ Средства Расширения Окно Справка Поиск (Ctrl+Q)

Выбрать C:\Users\ant_daddy\OneDrive\Рабочий стол\ТАЯК\Лаб3\Лаб3\Debug

Входной алфавит:

```
SouP = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, x, y}
```

Алфавит магазинных символов:

```
Z = {C, D, E, S, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, x, y, h0}
```

Список команд:

```
f(s0, lambda, E) = {(s0, C); (s0, SC)}
f(s0, lambda, C) = {(s0, a); (s0, b); (s0, x); (s0, y)}
f(s0, lambda, S) = {(s0, C); (s0, D); (s0, SC); (s0, SD)}
f(s0, lambda, D) = {(s0, 0); (s0, 1); (s0, 2); (s0, 3); (s0, 4); (s0, 5); (s0, 6); (s0, 7); (s0, 8); (s0, 9)}
f(s0, 0, 0) = {(s0, lambda)}
f(s0, 1, 1) = {(s0, lambda)}
f(s0, 2, 2) = {(s0, lambda)}
f(s0, 3, 3) = {(s0, lambda)}
f(s0, 4, 4) = {(s0, lambda)}
f(s0, 5, 5) = {(s0, lambda)}
f(s0, 6, 6) = {(s0, lambda)}
f(s0, 7, 7) = {(s0, lambda)}
f(s0, 8, 8) = {(s0, lambda)}
f(s0, 9, 9) = {(s0, lambda)}
f(s0, a, a) = {(s0, lambda)}
f(s0, b, b) = {(s0, lambda)}
f(s0, x, x) = {(s0, lambda)}
f(s0, y, y) = {(s0, lambda)}
f(s0, lambda, h0) = {(s0, lambda)}
```

Введите строку:

```
a
```

Валидная строка

Цепочка конфигураций:

```
(s0, a, h0E) |- (s0, a, h0C) |- (s0, a, h0a) |- (s0, lambda, h0) |- (s0, lambda, lambda)
```

Введите строку:

```
b
```

Валидная строка

Цепочка конфигураций:

```
(s0, b, h0E) |- (s0, b, h0C) |- (s0, b, h0b) |- (s0, lambda, h0) |- (s0, lambda, lambda)
```

** \Введите строку:

** (ab

*** Валидная строка

PS (

SouВведите строку:

```
b
```

Валидная строка

Цепочка конфигураций:

```
(s0, b, h0E) |- (s0, b, h0C) |- (s0, b, h0b) |- (s0, lambda, h0) |- (s0, lambda, lambda)
```

Введите строку:

```
ab
```

Валидная строка

Цепочка конфигураций:

```
(s0, ab, h0E) |- (s0, ab, h0SC) |- (s0, ab, h0Sa) |- (s0, b, h0C) |- (s0, b, h0b) |- (s0, lambda, h0) |- (s0, lambda, lambda)
```

Введите строку:

```
a1
```

Валидная строка

Цепочка конфигураций:

```
(s0, a1, h0E) |- (s0, a1, h0SC) |- (s0, a1, h0Sa) |- (s0, 1, h0S) |- (s0, 1, h0D) |- (s0, 1, h01) |- (s0, lambda, h0) |- (s0, lambda, lambda)
```

Введите строку:

```
a1b0
```

Валидная строка

Цепочка конфигураций:

```
(s0, a1b0, h0E) |- (s0, a1b0, h0SC) |- (s0, a1b0, h0Sa) |- (s0, 1b0, h0S) |- (s0, 1b0, h0SD) |- (s0, 1b0, h0S1) |- (s0, b0, h0S) |- (s0, b0, h0SC) |- (s0, b0, h0Sb) |- (s0, 0, h0S) |- (s0, 0, h0D) |- (s0, lambda, h0) |- (s0, lambda, lambda)
```

Введите строку:

```
axb1y4
```

Валидная строка

Цепочка конфигураций:

```
(s0, axb1y4, h0E) |- (s0, axb1y4, h0SC) |- (s0, axb1y4, h0Sa) |- (s0, xb1y4, h0S) |- (s0, xb1y4, h0SC) |- (s0, xb1y4, h0Sx) |- (s0, b1y4, h0S) |- (s0, b1y4, h0SC) |- (s0, b1y4, h0Sb) |- (s0, 1y4, h0S) |- (s0, 1y4, h0SD) |- (s0, 1y4, h0S1) |- (s0, y4, h0S) |- (s0, y4, h0SC) |- (s0, y4, h0Sy) |- (s0, 4, h0S) |- (s0, 4, h0D) |- (s0, 4, h04) |- (s0, lambda, h0) |- (s0, lambda, lambda)
```

** \

** (

PS (

Тестовые данные:

```
C:\Users\ant_daddy\OneDrive\Рабочий стол\ТАЯ... — □ ×  
14fq, b=15fq  
15fq, f=0f  
15qq, c=15qq  
15qq, d=2q  
15qq, f=0f  
2q, a=5q  
2q, d=0f  
5q, f=0f  
q0, a=134qqq  
a  
NOT VALID  
b  
NOT VALID  
e  
NOT VALID  
ab  
VALID  
ae  
VALID  
ad  
NOT VALID  
abe  
NOT VALID  
aeb  
VALID  
abed  
NOT VALID  
abec  
NOT VALID
```