Федеральное государственное автономное образовательное учреждение

высшего образования «Санкт-Петербургский государственный университет аэрокосмического приборостроения» (ГУАП)

ОТЧЕТ ЗАЩИЩЕН

Ст. преподаватель Рогачев С.А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Должность Ф.И.О дата, подпись

**Отчет**

**о лабораторной работе №3**

**«Алгоритмическая система Поста»**

по дисциплине «Теория вычислительных процессов»

ОТЧЕТ ВЫПОЛНИЛ:

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ф.И.О. группа дата, подпись

Санкт-Петербург

2024

**Цель работы**

Изучить и построить формальную систему Поста, реализующую вычисление заданной арифметической функции.

**Постановка задачи**

* Построить формальную систему Поста FSp, реализующую вычисление заданной арифметической функции.
* Написать программу на языке высокого уровня имитирующую (эмулирующую) вычисления на основе выводимости в формальной системе Поста.
* Программа должна работать на любых входных данных из заданного множества.
* Программа должна удовлетворять предъявляемым требованиям.

Вариант:



**Листинг программы на языке высокого уровня с комментариями**

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

#include <regex>

#include <Windows.h>

#include <algorithm>

using namespace std;

// Получение индекса переменной в строке

int get\_index(const string& found\_variables, char ch) {

return found\_variables.find(ch);

}

// Проверка корректности входных данных

bool is\_input\_correct(const string& function, const string& alphabet, const string& variables, const string& axioms, const vector<string>& left\_rules, const vector<string>& right\_rules) {

// Проверка символов в исходной строке и аксиомах

auto check\_symbols = [&](const string& str) {

for (char character : str) {

if (alphabet.find(character) == string::npos) {

cout << "Ошибка: символ '" << character << "' не входит в алфавит!" << endl;

return false;

}

}

return true;

};

if (!check\_symbols(function) || !check\_symbols(axioms)) return false;

// Проверка символов в правилах

for (const string& rule : left\_rules) {

for (char character : rule) {

if (variables.find(character) == string::npos && alphabet.find(character) == string::npos) {

cout << "Ошибка: символ '" << character << "' не входит в алфавит или множество переменных!" << endl;

return false;

}

}

}

return true;

}

// Чтение и разбор входных данных из файла

bool read\_input(const string& filename, string& function, string& alphabet, string& variables, string& axioms, vector<string>& left\_rules, vector<string>& right\_rules) {

ifstream fi(filename);

if (!fi.is\_open()) {

cout << "Ошибка открытия файла!" << endl;

return false;

}

// Считывание строк

getline(fi, function);

getline(fi, alphabet);

getline(fi, variables);

getline(fi, axioms);

// Очистка лишних символов

auto clean\_input = [](string& str, const string& prefix, const string& suffix) {

str.erase(0, prefix.size());

str.erase(str.size() - suffix.size());

str.erase(remove(str.begin(), str.end(), ','), str.end());

};

clean\_input(alphabet, "A={", "};");

clean\_input(variables, "X={", "};");

clean\_input(axioms, "A1={", "};");

// Считывание правил

string line;

while (getline(fi, line) && line != "}") {

size\_t pos = line.find("->");

if (pos != string::npos) {

left\_rules.push\_back(line.substr(0, pos));

right\_rules.push\_back(line.substr(pos + 2));

}

}

// Очистка оставшихся символов

if (!left\_rules.empty()) left\_rules[0].erase(0, 3);

if (!right\_rules.empty()) right\_rules.back().pop\_back();

fi.close();

return true;

}

// Функция для применения правил

void apply\_rules(string function, const string& variables, const vector<string>& left\_rules, const vector<string>& right\_rules) {

ofstream out("output.txt");

if (!out.is\_open()) {

cout << "Ошибка открытия файла для записи!" << endl;

return;

}

size\_t rule\_counter = 0;

int return\_flag = 0;

// Переменная для хранения найденных переменных

string found\_variables;

// Пока не будет пройдена вся таблица правил

while (return\_flag < left\_rules.size()) {

string reg\_ex;

found\_variables.clear();

// Построение регулярного выражения на основе текущего правила

for (char ch : left\_rules[rule\_counter]) {

if (variables.find(ch) != string::npos) {

if (get\_index(found\_variables, ch) == string::npos) {

reg\_ex += "(1\*)"; // Найдено новое вхождение переменной

found\_variables += ch;

}

else {

reg\_ex += "(\\" + to\_string(get\_index(found\_variables, ch) + 1) + ')';

}

}

else if (ch == '+' || ch == '=') {

reg\_ex += "\\" + string(1, ch); // Обработка символов + и =

}

else {

reg\_ex += ch;

}

}

regex re(reg\_ex);

smatch match;

if (regex\_search(function, match, re)) {

out << "Исходная строка: " << function << endl;

out << "Применяемое правило: " << left\_rules[rule\_counter] << " -> " << right\_rules[rule\_counter] << endl;

return\_flag = 0;

string tmp;

vector<string> values(match.size() - 1);

for (size\_t i = 1; i < match.size(); ++i) {

values[i - 1] = match[i];

}

// Применение правой части правила

for (char ch : right\_rules[rule\_counter]) {

if (variables.find(ch) != string::npos) {

tmp += values[get\_index(found\_variables, ch)];

}

else {

tmp += ch;

}

}

// Замена строки на новую

function = match.prefix().str() + tmp + match.suffix().str();

out << "Результат применения правила: " << function << endl << endl;

}

else {

return\_flag++;

}

rule\_counter = (rule\_counter + 1) % left\_rules.size();

}

cout << "Вычисления завершены" << endl;

out.close();

}

int main() {

SetConsoleOutputCP(1251);

string function, alphabet, variables, axioms;

vector<string> left\_rules, right\_rules;

if (read\_input("input.txt", function, alphabet, variables, axioms, left\_rules, right\_rules) &&

is\_input\_correct(function, alphabet, variables, axioms, left\_rules, right\_rules)) {

apply\_rules(function, variables, left\_rules, right\_rules);

}

return 0;

}

**Содержимое входного файла согласно заданию**

111+11=

A={1,+,=};

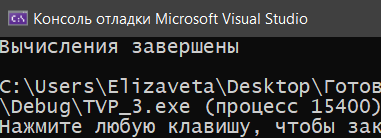
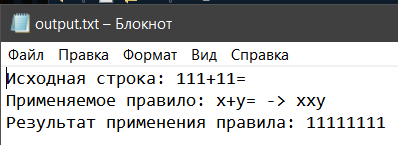
X={x,y};

A1={1};

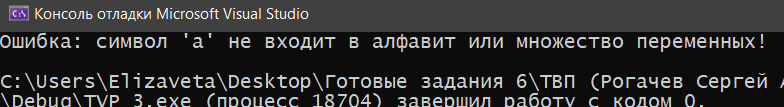
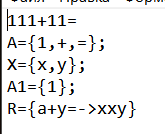
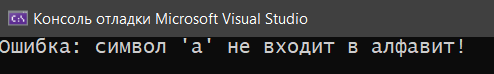
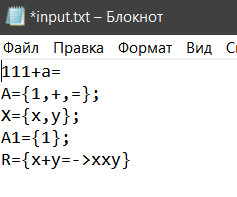
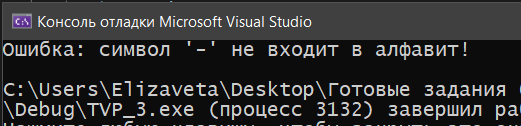
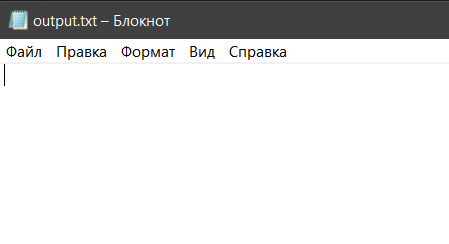
R={x+y=->xxy}

**Примеры результатов выполнения**

**Примеры c корректными входными данными**



**Примеры c некорректными входными данными**



**Вывод**

Изучена и построена формальная система Поста, реализующая вычисление заданной арифметической функции.