МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА №  43

ОТЧЁТ

ЗАЩИЩЁН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

### ст.пр.                                   Рогачёв С.А.

должность, уч. Степень, звание   подпись, дата           инициалы, фамилия

ОТЧЁТ О ЛАБОРАТОРНОЙ РАБОТЕ №3.

Алгоритмическая система Поста.

по курсу: Теория вычислительных процессов

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. 4136                                                                                Бобрович Н. С.

                                                                         подпись, дата                      инициалы, фамилия

Санкт-Петербург 2024

1. **Цель работы:**

Изучить алгоритмическую систему Поста.

1. **Постановка задачи:**

Программа должна корректно обрабатывать ошибочные входные данные;

Контроль однозначности заданных правил проводить не требуется;

Следующие входные данные должны считываться программой из файла 

A – собственный алфавит 

X – множество переменных 

A1 – множество аксиом 

R – конечное множество продукций (или правил вывода)

Результат работы программы должен содержать: 

Файл с результатами вычисления 

Сообщение на экране о следующих событиях:

o Вычисление прошло успешно;

o В ходе вычисления произошла ошибка: 

Найден символ не входящий в алфавит; 

Найдена переменная не входящая в заданное множество переменных;

**Вариант 30: 2x**

1. **Листинг программы на языке высокого уровня с комментариями:**

#include <iostream>

#include <fstream>

#include <sstream>

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

// Функция для проверки принадлежности символа к алфавиту

bool isInAlphabet(const char c, const vector<char>& alphabet) {

return find(alphabet.begin(), alphabet.end(), c) != alphabet.end();

}

// Функция для проверки принадлежности переменной к множеству переменных

bool isInVariables(const string& var, const vector<string>& variables) {

return find(variables.begin(), variables.end(), var) != variables.end();

}

// Основная функция для эмуляции вычислений

void emulate(const vector<char>& alphabet, const vector<string>& variables, const vector<string>& rules, const string& input, ofstream& outputFile) {

// Проверка правильности входной строки

bool validInput = true;

for (char c : input) {

if (!isInAlphabet(c, alphabet)) {

cout << "Ошибка: найден символ не входящий в алфавит." << endl;

validInput = false;

break;

}

}

if (!validInput) {

return;

}

// Эмуляция вычислений

string currentString = input;

size\_t currentStringsize = input.size();

for (const auto& rule : rules) {

size\_t pos = rule.size() - 1;

string before = currentString.substr(0, pos);

string after = currentString.substr(pos, input.size());

string newString = after + after;

// Запись результата в выходной файл

outputFile << "Исходная строка: " << currentString << endl;

outputFile << "Применяемое правило: " << rule << endl;

outputFile << "Результат применения правила: " << newString << endl << endl;

currentString = newString;

cout << "Вычисление прошло успешно" << endl;

break;

}

}

int main() {

setlocale(LC\_ALL, "Russian");

// Чтение входных данных из файла

ifstream inputFile("input.txt");

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

cerr << "Не удалось открыть файл input.txt" << endl;

return 1;

}

vector<char> alphabet;

vector<string> variables;

vector<string> rules;

vector<string> axioms;

string line;

while (getline(inputFile, line)) {

if (line.empty()) continue;

if (line.find("A =") == 0) { // Собственный алфавит

size\_t startPos = line.find('{') + 1;

size\_t endPos = line.rfind('}');

string elements = line.substr(startPos, endPos - startPos);

istringstream iss(elements);

char c;

while (iss >> c) {

alphabet.push\_back(c);

}

}

else if (line.find("X =") == 0) { // Множество переменных

size\_t startPos = line.find('{') + 1;

size\_t endPos = line.rfind('}');

string elements = line.substr(startPos, endPos - startPos);

istringstream iss(elements);

string var;

while (iss >> var) {

variables.push\_back(var);

}

}

else if (line.find("A1 =") == 0) { // Множество аксиом

size\_t startPos = line.find('{') + 1;

size\_t endPos = line.rfind('}');

string elements = line.substr(startPos, endPos - startPos);

istringstream iss(elements);

string axiom;

while (iss >> axiom) {

axioms.push\_back(axiom);

}

}

else if (line.find("R =") == 0) { // Конечное множество продукций

size\_t startPos = line.find('{') + 1;

size\_t endPos = line.rfind('}');

string elements = line.substr(startPos, endPos - startPos);

istringstream iss(elements);

string rule;

while (iss >> rule) {

bool validRule = true;

// Проверим каждый символ правила

for (size\_t i = 0; i < rule.length(); ++i) {

char c = rule[i];

if (c == '-' && i + 1 < rule.length() && rule[i + 1] == '>') {

i++;

continue;

}

if (!isInAlphabet(c, alphabet) && !isInVariables(string(1, c), variables)) {

validRule = false;

break;

}

}

if (validRule) {

rules.push\_back(rule);

}

else {

cout << "Найдена переменная не входящая в заданное множество переменных." << endl;

}

}

}

}

inputFile.close();

// Открытие файла для записи результатов

ofstream outputFile("output.txt");

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

cerr << "Не удалось создать файл output.txt" << endl;

return 1;

}

// Выполнение эмуляции для каждой аксиомы

for (const auto& axiom : axioms) {

emulate(alphabet, variables, rules, axiom, outputFile);

}

outputFile.close();

return 0;

}

1. **Содержимое входного файла согласно заданию:**

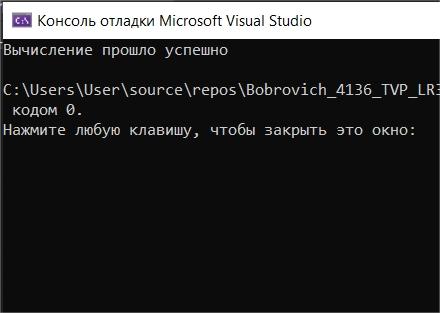
A = {1, \*};

X = {x};

A1 = {11\*111};

R = {11\*x -> xx}

1. **Пример результата выполнения:**

**Вычисление прошло успешно:**  


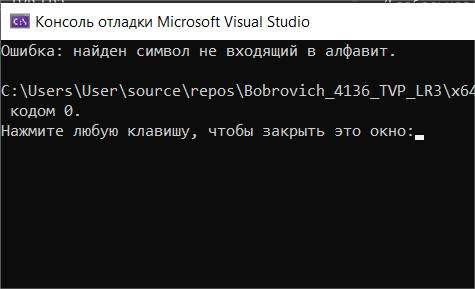
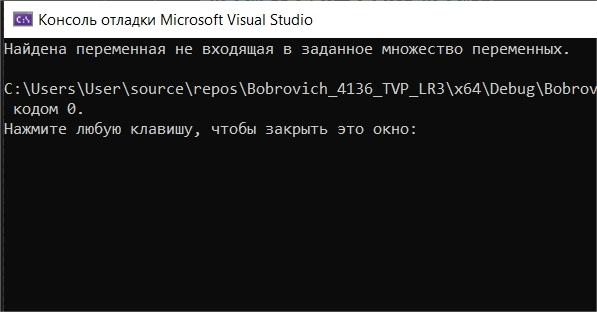
Файл вывода:

Исходная строка: 11\*111

Применяемое правило: 11\*x

Результат применения правила: 111111

**В ходе вычисления произошла ошибка:**



1. **Вывод:**

Изучил алгоритмическую систему Поста.