МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 43

ОТЧЁТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ассистент | |  |  | | | |  | | Д. А. Кочин |
| должность, уч. степень, звание | |  | подпись, дата | | | |  | | инициалы, фамилия |
| ОТЧЁТ О ЛАБОРАТОРНОЙ РАБОТЕ №3 | | | | | | | | | | |
| Алгоритмическая система Поста | | | | | | | | | | |
| по дисциплине: ТЕОРИЯ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ | | | | | | | | | | |
|  | | | | | | | | | | |
| РАБОТУ ВЫПОЛНИЛА | | | | | | | | | | |
| СТУДЕНТКА ГР. | 4931 | | |  | 08.12.2022 |  | | Е.Ю. Ильченко | | |
|  |  | | |  | подпись, дата |  | | инициалы, фамилия | | |
|  |  | | |  |  |  | |  | | |

Санкт-Петербург 2022

1. **Цель работы:**

Цель настоящей работы – ознакомиться с формальной системой Поста и программно реализовать её для вычисления заданной арифметической функции.

1. **Постановка задачи:**

|  |  |
| --- | --- |
| **Вариант 39** | 3x - y |

В данной лабораторной работе требуется:

• 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.

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

import re  
  
  
def read\_tape(file\_name):  
 with open(file\_name) as file:  
 return file.readline().replace(" ","")  
  
  
def read\_data(file\_name):  
 with open(file\_name, 'r') as file:  
 data = file.read().rstrip()  
 match = re.findall(r"\{(.+)\}", data) # отбираем все внутри скобок  
 data = match[0]  
 match = re.findall(r"[^\,\ ]", data) # находим все кроме ( ) и (,)  
 return match  
  
  
def read\_rule(file\_name):  
 with open(file\_name, 'r') as file:  
 rules = [  
 line.rstrip().replace(',', '').replace('{', '').replace('}', '').replace('R', '').replace(' ', '').split(  
 '->') for line  
 in file.readlines()]  
 rules = list(rules)  
 rules[0][0] = rules[0][0].replace('=', '', 1)  
 return rules  
  
  
def init\_reg\_exp(rule):  
 reg\_exp = rule  
 reg\_exp = reg\_exp.replace('\*', '\\\*')  
 reg\_exp = reg\_exp.replace('/', '\\/')  
 reg\_exp = reg\_exp.replace('+', '\\+')  
 reg\_exp = reg\_exp.replace('-', '\\-')  
 return reg\_exp  
  
  
def replace\_tape\_byrule(tape, rule, variable\_rule, substr):  
 variable\_dictionary = {} # словарь переменных (х: 111)  
  
 old\_subtape = substr[0] # заменяемая строка после применения правила (левой части)  
  
 # сопоставление переменных  
 for i in range(len(variable\_rule)):  
 variable\_dictionary[variable\_rule[i]] = substr[i + 1] # заносим значение переменной в словарь  
 new\_subtape = rule[1] # правая часть правила для замены после применения регулярного выражения  
 temp\_new\_substring = new\_subtape  
 # находим переменую в результирующей части правила и заменяем ее на значение переменной из словаря  
 for var in new\_subtape:  
 if var in variable\_dictionary:  
 temp\_new\_substring = temp\_new\_substring.replace(var, variable\_dictionary[var])  
 tape = tape.replace(old\_subtape, temp\_new\_substring) # по правилу меняем ленту  
  
 return tape  
  
  
def replace\_var(reg\_exp, exp\_variables, variable\_alphabet):  
 variable\_rule = [] # переменные в правиле  
 for var in reg\_exp:  
 if var in variable\_alphabet: # символ находится в исходном алфавите(А)  
 if var not in variable\_rule: # символ не добавлен в группу переменных  
 variable\_rule.append(var)  
 reg\_exp = reg\_exp.replace(var, exp\_variables, 1) # заменяем символ из алфавита на регулярное выражение  
 else: # символ находится в группе переменных  
 reg\_exp = reg\_exp.replace(var, '\\' + str(  
 variable\_rule.index(var) + 1)) # замена содержимого соответствующей группой  
 return reg\_exp, variable\_rule  
  
  
def print\_write(rule, tape, flag):  
 with open('result.txt', 'a') as file:  
 if flag == 1:  
 print('--------------------------------')  
 print('Текущая строка:\n' + tape + '\n')  
 print('Применяем правило:\n' + rule[0] + '->' + rule[1] + '\n')  
 file.write('\n--------------------------------')  
 file.write('\nТекущая строка:\n' + tape + '\n')  
 file.write('\nПрименяем правило:\n' + rule[0] + '->' + rule[1] + '\n')  
 if flag == 2:  
 print("После применения:\n" + tape)  
 file.write("\nПосле применения:\n" + tape)  
 if flag == 3:  
 print('Правило: ' + rule[0] + '->' + rule[1] + ' не применимо')  
 file.write('\nПравило: ' + rule[0] + '->' + rule[1] + ' не применимо')  
 if flag == 4:  
 file.write('\n--------------------------------')  
 file.write('\nКонечный результат: ' + tape)  
 print('--------------------------------')  
 print('Конечный результат: ', tape)  
  
  
def clear\_file():  
 f = open('result.txt', 'w')  
 f.close()  
  
  
def errors(alphabet, rules, variable\_alphabet):  
 for rule in rules:  
 for r in rule:  
 for simb in r:  
 if simb not in alphabet and simb not in variable\_alphabet:  
 exit('Неизвестный символ: ' + simb)  
  
  
def main(alphabet, variable\_alphabet, axioms, rules, tape):  
 clear\_file()  
 exp\_variables = '(['  
 for var in axioms:  
 exp\_variables += var  
 exp\_variables += ']+)'  
  
 flag = True  
 while flag:  
 flag = False  
 for rule in rules:  
 reg\_exp = init\_reg\_exp(rule[0])  
 reg\_exp, variable\_rule = replace\_var(reg\_exp, exp\_variables, variable\_alphabet)  
 # print(reg\_exp)  
 # проверяем с помощью регулярных выражений  
 substr = re.search(reg\_exp, tape)  
 if substr:  
 # нашли совпадение в ленте подходящее под правило  
 print\_write(rule, tape, 1)  
 tape = replace\_tape\_byrule(tape, rule, variable\_rule, substr) # лента после замены по правилу  
 print\_write(None, tape, 2)  
 flag = True  
 break  
 else:  
 print\_write(rule, tape, 3)  
 print\_write('', tape, 4)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 alphabet = read\_data('A.txt')  
 variable\_alphabet = read\_data('X.txt')  
 axioms = read\_data('A1.txt')  
 rules = read\_rule('R.txt')  
 tape = read\_tape('tape.txt')  
 errors(alphabet, rules, variable\_alphabet)  
 print("Собственный алфавит:", alphabet,  
 "\nМножество переменных:", variable\_alphabet,  
 "\nМножество аксиом:", axioms,  
 "\nПравила:", rules)  
 main(alphabet, variable\_alphabet, axioms, rules, tape)

1. **Пример результата выполнения:**

**Содержание файлов:**

**A.txt**

A={=,\*,1,-}

**A1.txt**

A1={1}

**R.txt**

R={1\*x-y= -> \*x-y=x  
\*y-x=x -> =}

**tape.txt**

111 \* 1111 - 1111 =

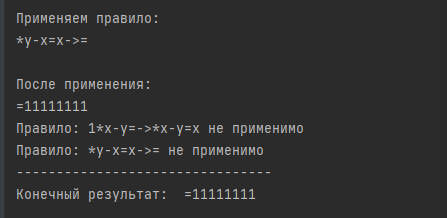
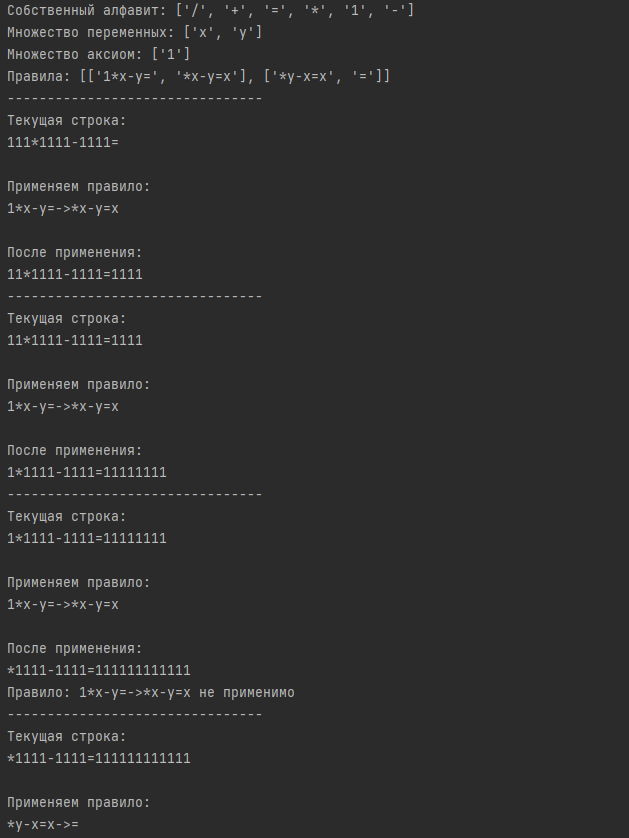
**X.txt**

X={x,y}

**result.txt**

--------------------------------  
Текущая строка:  
111\*1111-1111=  
  
Применяем правило:  
1\*x-y=->\*x-y=x  
  
После применения:  
11\*1111-1111=1111  
--------------------------------  
Текущая строка:  
11\*1111-1111=1111  
  
Применяем правило:  
1\*x-y=->\*x-y=x  
  
После применения:  
1\*1111-1111=11111111  
--------------------------------  
Текущая строка:  
1\*1111-1111=11111111  
  
Применяем правило:  
1\*x-y=->\*x-y=x  
  
После применения:  
\*1111-1111=111111111111  
Правило: 1\*x-y=->\*x-y=x не применимо  
--------------------------------  
Текущая строка:  
\*1111-1111=111111111111  
  
Применяем правило:  
\*y-x=x->=  
  
После применения:  
=11111111  
Правило: 1\*x-y=->\*x-y=x не применимо  
Правило: \*y-x=x->= не применимо  
--------------------------------  
Конечный результат: =11111111

**Результат выполнения:**



**Проверка на универсальность:**

**Вариант 1**

**Содержание файлов:**

**A.txt**

A={+,=,1}

**A1.txt**

A1={1}

**R.txt**

R={x+y= -> =xy}

**tape.txt**

1111 + 111111 =

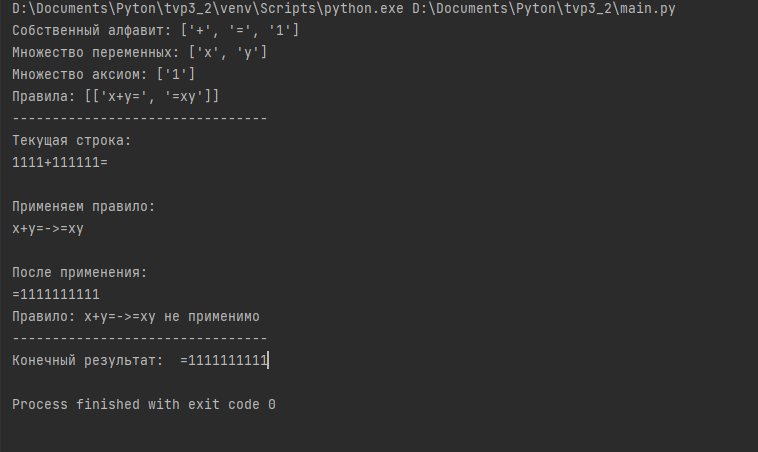
**X.txt**

X={x,y}

**result.txt**

--------------------------------  
Текущая строка:  
1111+111111=  
  
Применяем правило:  
x+y=->=xy  
  
После применения:  
=1111111111  
Правило: x+y=->=xy не применимо  
--------------------------------  
Конечный результат: =1111111111

**Результат выполнения:**



**Вариант 2**

**Содержание файлов:**

**A.txt**

A={+,/,=,1}

**A1.txt**

A1={1}

**R.txt**

R={x/x= -> /x=1  
x+/y= -> x+=  
x+= -> =x}

**tape.txt**

1111 + 111111/11 =

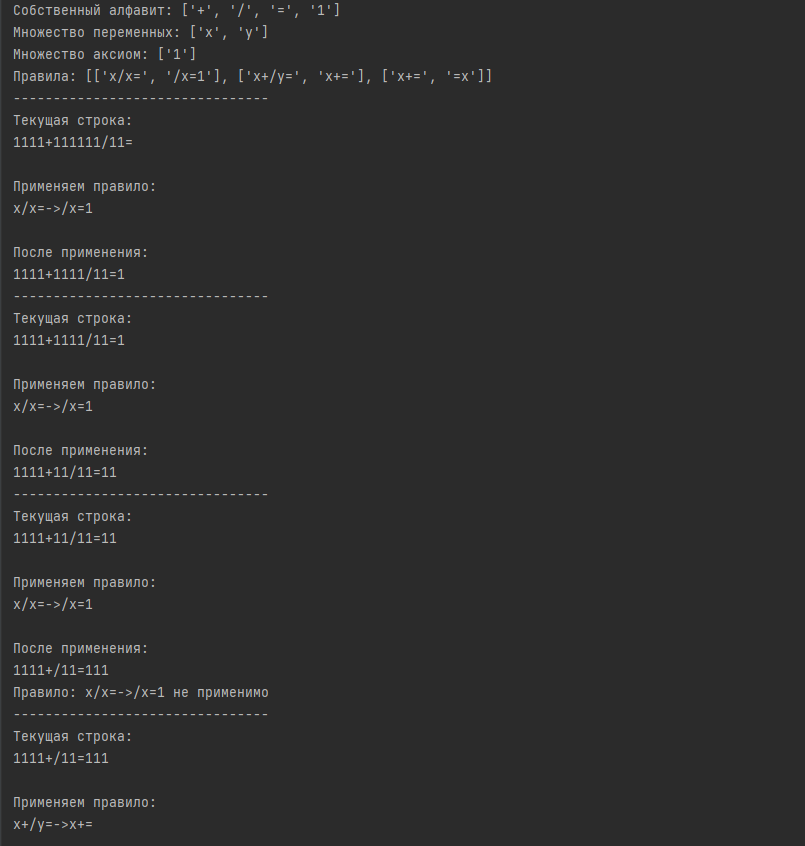
**X.txt**

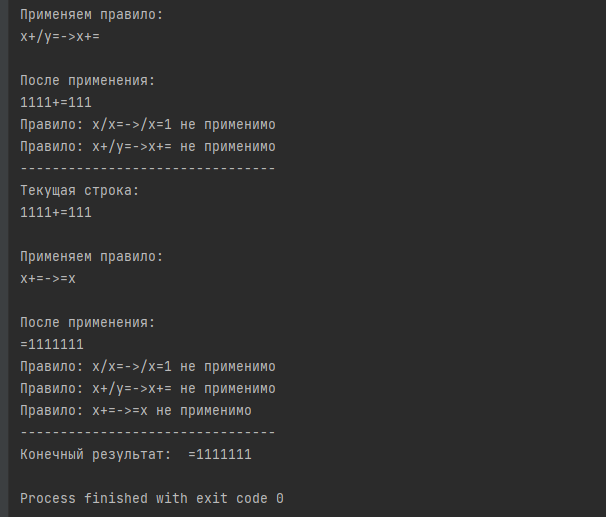
X={x,y}

**result.txt**

--------------------------------  
Текущая строка:  
1111+111111/11=  
  
Применяем правило:  
x/x=->/x=1  
  
После применения:  
1111+1111/11=1  
--------------------------------  
Текущая строка:  
1111+1111/11=1  
  
Применяем правило:  
x/x=->/x=1  
  
После применения:  
1111+11/11=11  
--------------------------------  
Текущая строка:  
1111+11/11=11  
  
Применяем правило:  
x/x=->/x=1  
  
После применения:  
1111+/11=111  
Правило: x/x=->/x=1 не применимо  
--------------------------------  
Текущая строка:  
1111+/11=111  
  
Применяем правило:  
x+/y=->x+=  
  
После применения:  
1111+=111  
Правило: x/x=->/x=1 не применимо  
Правило: x+/y=->x+= не применимо  
--------------------------------  
Текущая строка:  
1111+=111  
  
Применяем правило:  
x+=->=x  
  
После применения:  
=1111111  
Правило: x/x=->/x=1 не применимо  
Правило: x+/y=->x+= не применимо  
Правило: x+=->=x не применимо  
--------------------------------  
Конечный результат: =1111111

**Результат выполнения:**





1. **Вывод:**

Ознакомились с формальной системой Поста и программно реализовали её для вычисления заданной арифметической функции.