МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБО6РОСТРОЕНИЯ»

КАФЕДРА № 43

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

Старший преподаватель		С.А. Рогачев
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2

Машина Тьюринга

по курсу: Теория Вычислительных Процессов

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4134к		Столяров Н.С.
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

Цель работы: Необходимо написать программу для машины Тьюринга, реализующую вычисление арифметической функции согласно выданному варианту задания. Должна быть составлена совокупность команд Р. Для выполнения данного задания следует использовать приложение Algo2000.

Основные сведения из теории

Маши́на Тью́ринга (сокр. МТ) — абстрактный исполнитель (абстрактная вычислительная машина). Была предложена Аланом Тьюрингом в 1936 году для определения понятия алгоритма.

Машина Тьюринга является расширением конечного автомата и, согласно тезису Чёрча — Тьюринга, способна имитировать всех исполнителей (с помощью задания правил перехода), каким-либо образом реализующих процесс пошагового вычисления, в котором каждый шаг вычисления достаточно элементарен.

Постановка задачи

Задача по варианту: Реализовать деление на 10 (x /10)

Совокупность команд для машины Тьюринга

```
q0 1 q1 _ >
q0 - q3 - >
q0 _ q3 _ >
q1 1 q1 1 >
q1 = q1 = >
q1 + q1 + >
q1 - q1 - >
q1 _ q2 1 <
q2 1 q2 1 <
q2 = q2 = <
q2 + q2 + <
q2 - q2 - <
q2 _ q0 _ >
q3 1 q4 # >
q3 + q7 + >
q3 - q3 - >
q4 1 q4 1 >
q4 = q4 = >
q4 + q4 + >
q4 _ q5 _ <
q5 1 q6 _ <
q5 _ q00 _ <
q6 1 q6 1 <
q6 = q6 = <
q6 + q6 + <
q6 # q3 # >
```

```
q7 1 q7 # >
q7 = q8 = >
q8 1 q8 1 >
q8 _ q5 1 >
```

Листинг программы на языке высокого уровня с комментариями

```
class TuringMachine:
    def __init__(self, tape, transitions, alphabet):
        # Заполнение ленты до 100 символов
        self.tape = list(tape) + ['_'] * (100 - len(tape))
        self.tape = self.tape[:100]
        self.head = 0
        self.state = 'q0'  # Начальное состояние
        self.transitions = transitions
        self.alphabet = alphabet
        self.steps = 0

    def step(self):
        current_symbol = self.tape[self.head]
        command = self.transitions.get((self.state, current_symbol))

        if (current_symbol == "_"):
            print(self.steps, current_symbol, command)

        if command is None:
            raise Exception(f"Нет перехода для состояния {self.state} и символа {current_symbol}")

        new_state, new_symbol, direction = command

        self.tape[self.head] = new_symbol
        self.state = new_state
        if new_state == 'q00':
            raise Exception(f"Завершение программы")

        if direction == '>':
            self.head += 1
        elif direction == '<':
            self.head -= 1

        self.steps += 1

        # Проверка выхода за границы ленты
        if self.head < 0 or self.head >= len(self.tape):
            raise Exception("Головка вышла за границы ленты")

    def run(self, output_file):
        with open(output_file, 'w', encoding='utf-8') as f:
            while True:
                # Запись состояния перед выполнением команды
```

```
            f.write(''.join(self.tape) + '\n')
            f.write(' ' * self.head + '^' + '\n')
            f.write(f"{self.state} -> {self.transitions.get((self.state, self.tape[self.head]), 'Нет
команды')} \n")

            try:
                self.step()
            except Exception as e:
                f.write(f"{e}\n")
                break

def read_file(filename):
    with open(filename, 'r') as f:
        return f.read().strip()

def read_transitions(filename):
    transitions = {}
    with open(filename, 'r') as f:
        for line in f:
            state, symbol, new_state, new_symbol, direction = line.strip().split()
            transitions[(state, symbol)] = (new_state, new_symbol, direction)
    return transitions

def main():
    tape = read_file('tape.txt')
    alphabet = read_file('alphabet.txt').split()
    transitions = read_transitions('transitions.txt')

    # Проверка алфавита
    for symbol in tape:
        if symbol not in alphabet:
            raise Exception(f"Символ '{symbol}' не в алфавите")

    tm = TuringMachine(tape, transitions, alphabet)
    tm.run('output.txt')

if __name__ == "__main__":
    main()
```

Пример результата выполнения

```
q6 -> ('q6', '=', '<')
______-
####+1=11_____________________________________________________
_____________
          ^
q6 -> ('q6', '1', '<')
______-
####+1=11_____________________________________________________
_____________
```

```
          ^
q6 -> ('q6', '+', '<')
______-
####+1=11_____________________________________________________________
_______________
          ^
q6 -> ('q3', '#', '>')
______-
####+1=11_____________________________________________________________
_______________
          ^
q3 -> ('q7', '+', '>')
______-
####+1=11_____________________________________________________________
_______________
          ^
q7 -> ('q7', '#', '>')
______-
####+#=11_____________________________________________________________
_______________
          ^
q7 -> ('q8', '=', '>')
______-
####+#=11_____________________________________________________________
_______________
          ^
q8 -> ('q8', '1', '>')
______-
####+#=11_____________________________________________________________
_______________
          ^
q8 -> ('q8', '1', '>')
______-
####+#=11_____________________________________________________________
_______________
          ^
q8 -> ('q5', '1', '>')
______-
####+#=111____________________________________________________________
_______________
          ^
q5 -> ('q00', '_', '<')
Завершение программы
```

Вывод

Реализована программа для выполнения задания на Машине Тьюринга на Python3