МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Информатика»

Тема: Моделирование работы Машины Тьюринга

Студент гр. 9381 _____ Колованов Р.А.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург

2019

Цель работы.

Изучение принципа работы машины Тьюринга и ее моделирование на языке программирования Python.

Задание.

На вход программе подается строка неизвестной длины. Каждый элемент является значением в ячейке памяти ленты Машины Тьюринга. На ленте находится троичное число, знак (плюс или минус) и троичная цифра.

		1	2	1	+	2			

Напишите программу, которая выполнит арифметическую операцию. Указатель на текущее состояние Машины Тьюринга изначально находится слева от числа (но не на первом его символе). По обе стороны от числа находятся пробелы. Результат арифметической операции запишите на месте первого числа. Для примера выше лента будет выглядеть так:

		2	0	0	+	2			

Ваша программа должна вывести полученную ленту после завершения работы.

Алфавит: 0, 1, 2, +, -, " " (пробел)

Соглашения:

1. Направление движения автомата может быть одно из R (направо), L (налево), N (неподвижно).

2. Число обязательно начинается с единицы или двойки.

3. Числа и знак операции между ними идут непрерывно.

4. Гарантируется, что в результате операции вычитания не может получиться отрицательного числа.

В отчет включите таблицу состояний. Отдельно кратко опишите каждое состояние, например: q1 - начальное состояние, которое необходимо, чтобы найти первую цифру первого числа.

Выполнение работы.

Для начала выполняется считывание “ленты”, по которой будет перемещаться машина Тьюринга. На вход подается строка, которая в дальнейшем при помощи функции *list()* преобразуется в список, где каждый элемент соответствует одному символу строки входных данных. Этим списком инициализируется переменная ***tape***. Далее создается словарь ***table***, который хранит программу для машины Тьюринга. Каждому ключу, означающему состояние машины, соответствует словарь, ключи которого – значения ячеек ленты (алфавит), а значения – действия, которая должна выполнить машина. Действия представлены в виде списка из трех элементов, где первый элемент – символ, который нужно записать в текущую ячейку, второй элемент – символ, определяющий перемещение автомата (‘L’ – переместиться влево, ‘R’ – переместиться вправо, ‘N’ – не перемещаться), а третий – состояние, в которое нужно перейти (храниться в виде строки: ‘q0’, ‘q1’ и т.д.).

Был разработан класс ***TuringMachine***, который позволяет моделировать работу машины Тьюринга. У класса существуют следующие поля:

1. ***tape*** – хранит “ленту”, по которой перемещается автомат. Представлена в виде списка символов.

2. ***table*** – хранит программу для машины Тьюринга. Представлена в виде словаря, который был описан выше.
3. ***state*** – хранит текущее состояние машины. Представлено в виде строки.
4. ***end_state*** – хранит конечное состояние машины. Представлено в виде строки.
5. ***index*** – хранит текущую позицию автомата на “ленте” ***tape***. Представлена в виде числа. Изначально равна 0.

Помимо полей были определены методы класса:

1. *read()* – возвращает символ на позиции ***index*** “ленты” ***tape***.
2. *write(value)* – записывает значение ***value*** на позицию ***index*** “ленты” ***tape***.
3. *move(direction)* – изменяет переменную **index** в зависимости от переданного значения (‘R’ – инкрементировать, ‘L’ – декрементировать, в любом другом случае – не изменять).
4. *execute()* – выполняет программу машины Тьюринга, которая хранится в переменной ***table***. Возвращает “ленту” ***tape*** после выполнения программы.

Рассмотрим подробнее метод *execute()*. Он содержит цикл *while*, который будет выполняться до тех пор, пока текущее состояние машины (***state***) не станет равно конечному состоянию машины (***end_state***). В цикле происходит следующее: для начала с “ленты” считывается символ при помощи метода *read()*. Далее в зависимости от значения прочитанного символа и текущего состояния машины выбираются действия из словаря ***table***. Как говорилось ранее, действия представлены в виде списка из трех элементов, значения которых записываются в переменные ***symbol***, ***direction***, ***state***. Далее выполняются эти три действия: записывается значение

переменной ***symbol*** в позицию ***index*** “ленты” ***tape*** при помощи метода *write()*, изменяется переменная ***index*** в зависимости от значения переменной ***direction*** при помощи метода *move()* и меняется значение текущего состояния машины на значение переменной ***state***, полученной из списка действий.

Программа машины Тьюринга для решения поставленной задачи имеет следующий вид:

	0	1	2	+	-	“ ”
q0	0; R; q0	1; R; q0	2; R; q0	+; R; q1	-; R; q2	“ ”; R; q0
q1	0; N; q10	1; L; q4	2; L; q3			
q2	0; N; q10	1; L; q6	2; L; q5			
q3	2; L; q10	0; L; q4	1; L; q4	+; L; q3	-; L; q3	
q4	1; N; q10	2; N; q10	0; L; q4	+; L; q4	-; L; q4	1; N; q10
q5	1; L; q6	2; L; q6	0; N; q10	+; L; q5	-; L; q5	
q6	2; L; q6	0; L; q7	1; N; q10	+; L; q6	-; L; q6	
q7	0; N; q10	1; N; q10	2; N; q10			“ ”; R; q8
q8	“ ”; R; q8	1; N; q10	2; N; q10		-; L; q9	
q9						0; N; q10

Краткое описание состояний (начальное состояние – q0, конечное состояние – q10):

- q0 – встает на позицию знака арифметической операции и в зависимости от знака переходит в одно из состояний q1 (ветвь сложения) или q2 (ветвь вычитания).
- q1 – смотрит на второе число (состоит из одной цифры) и в зависимости от его значения переходит в одно из состояний q3 или q4.
- q2 – смотрит на второе число (состоит из одной цифры) и в зависимости от его значения переходит в одно из состояний q5 или q6.
- q3, q4 – прибавляют соответствующее значение к последней цифре первого числа. Если случается переполнение разряда, то автомат сдвигается на разряд влево и переходит в состояние q4 (q4 – прибавление единицы).
- q5, q6 – вычитают соответствующее значение из последней цифры первого числа. Если в разряде не хватает единиц, то автомат переходит

в состояние q6 (q6 – вычитание единицы). Если после вычитания получается 0, то автомат переходит в состояние q7.

- q7 – проверяет, не является ли этот разряд старшим. Если он является старшим, то автомат переходит в состояние q8.
- q8 – убирает все нули со старших разрядов и обрабатывает случай, если первое число после выполнения операции равно нулю. Если это так, то автомат переходит в состояние q9
- q9 – записывает 0 перед арифметическим знаком.

Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	112-2	110-2	
2.	1-1	0-1	
3.	10-1	2-1	
4.	12+2	21+2	
5.	22+2	101+2	

Выводы.

Были изучены принципы работы машины Тьюринга. На языке Python была разработана программа, моделирующая работу машины Тьюринга. Для написания программы использовались условные операторы *if-elif*, операторы цикла while, классы. Был написан класс, который способен моделировать работу машины Тьюринга.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
class TuringMachine:
    def __init__(self, tp, tbl, start_state='q0', end_state='q0'):
        self.tape = tp
        self.table = tbl
        self.state = start_state
        self.end_state = end_state
        self.index = 0

    def read(self):
        return self.tape[self.index]

    def write(self, value):
        self.tape[self.index] = value

    def move(self, direction):
        if direction == 'R':
            self.index += 1
        elif direction == 'L':
            self.index -= 1

    def execute(self):
        while self.state != self.end_state:
            symbol, direction, state = self.table[self.state]
[self.read()]
            self.write(symbol)
            self.move(direction)
            self.state = state
        return self.tape

if __name__ == '__main__':
    tape = list(input())
    table = {
        'q0': {' ': [' ', 'R', 'q0'], '2': ['2', 'R', 'q0'], '1':
['1', 'R', 'q0'], '0': ['0', 'R', 'q0'],
               '+': ['+', 'R', 'q1'], '-': ['-', 'R', 'q2']},
        'q1': {'2': ['2', 'L', 'q3'], '1': ['1', 'L', 'q4'], '0':
['0', 'N', 'q10']},
        'q2': {'2': ['2', 'L', 'q5'], '1': ['1', 'L', 'q6'], '0':
['0', 'N', 'q10']},
        'q3': {'+': ['+', 'L', 'q3'], '-': ['-', 'L', 'q3'], '2':
['1', 'L', 'q4'], '1': ['0', 'L', 'q4'],
               '0': ['2', 'N', 'q10']},
        'q4': {'+': ['+', 'L', 'q4'], '-': ['-', 'L', 'q4'], '2':
['0', 'L', 'q4'], '1': ['2', 'N', 'q10'],
               '0': ['1', 'N', 'q10'], ' ': ['1', 'N', 'q10']},
        'q5': {'+': ['+', 'L', 'q5'], '-': ['-', 'L', 'q5'], '2':
['0', 'N', 'q10'], '1': ['2', 'L', 'q6'],
               '0': ['1', 'L', 'q6']},
        'q6': {'+': ['+', 'L', 'q6'], '-': ['-', 'L', 'q6'], '2':
['1', 'N', 'q10'], '1': ['0', 'L', 'q7'],
```

```
                '0': ['2', 'L', 'q6']},
          'q7': {'0': ['0', 'N', 'q10'], '1': ['1', 'N', 'q10'], '2':
['2', 'N', 'q10'], ' ': [' ', 'R', 'q8']},
          'q8': {'0': [' ', 'R', 'q8'], '1': ['1', 'N', 'q10'], '2':
['2', 'N', 'q10'], '-': ['-', 'L', 'q9']},
          'q9': {' ': ['0', 'N', 'q10']}
      }

      turing_machine = TuringMachine(tape, table, start_state='q0',
end_state='q10')
      print("".join(turing_machine.execute()))
```