МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Информатика» Тема:

Моделирование работы машины Тьюринга

Студент гр. 0382		Шангичев В. А.
Преподаватель		Шевская Н. В.

Санкт-Петербург

2020

Цель работы.

Смоделировать работу машины Тьюринга, используя язык 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. 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. Ваша программа должна вывести полученную ленту после завершения работы.

Основные теоретические положения.

Составляющие машины Тьюринга:

1. Неограниченная лента Она является бесконечной в обе стороны и разделена на ячейки.
2. Автомат – управляемая программа, головка-сканер для считывания и записи данных.

Каждая машина связывает два конечных ряда данных: алфавит входящих символов A = {a0, a1, ..., am} и алфавит состояний Q = {q0, q1, ..., qp}. Состояние qp называют пассивным. Считается, что устройство заканчивает свою работу, когда попадает именно на него. Состояние q0 называют начальным - машина начинает свои вычисления, находясь на старте в нем. Входное слово располагается на ленте по одной букве подряд в каждой позиции. С обеих сторон от него располагаются только пустые ячейки.

Конструкции языка Python.

`class <class name>():` - создает класс

метод `__init__()` – метод, который будет вызван объектом класса в момент его создания

`def <method name>():` - объявляет метод класса

`self` – ссылка на объект класса

`<object>.<method name>()` – вызывает метод класса, передавая в аргумент `self <object>`.

`Array.copy()` – возвращает копию списка.

`{key_1: value_1, key_2: value_2, …}`-создает словарь.

Выполнение работы.

Решение задачи будет разделено на несколько этапов:

1. Создание класса `T_machine`, моделирующего работу машины Тьюринга.
2. Создание таблицы программы из словарей.
3. Ввод данных, передача данных и программы объекту класса `T_machine`.
4. Вызов метода `start()` и вывод данных.

Для решения первой подзадачи необходимо определить, какие атрибуты и методы должен содержать в себе класс `T_machine`.

Атрибуты:

- `memory` – здесь будет храниться лента.
- `table` – в данном атрибуте будет сохраняться программа.
- `state` – это поле хранит текущее состояние.
- `index` – индекс текущего элемента на ленте.

Методы:

- `__init ()` – метод, который будет вызван однажды, при создании объекта класса. Принимает следующие аргументы: `self, string, program, init_state` со значением по умолчанию `'q0'`. В этом

методе задаются начальные значения всех атрибутов используя вышеуказанные аргументы. Атрибуту `index` присваивается значение `0`.

- `start()` – метод моделирующий выполнение программы на машине Тьюринга. В данном методе используется цикл `while`, который выполняется до тех пор, пока значение атрибута `state` не равно названию конечного состояния (qt). В теле цикла в переменную `st` созраняется словарь с инструкциями для конкретного случая. Для этого сначала выделяем весь словарь состояния, используя ключ, значением которого является название текущего состояния. Снова обращаемся к словарю по ключу, значением которого теперь является значение текущего элемента на ленте. Затем последовательно выполняются три действия: запись, перемещение, переход из состояния в состояние. После завершения работы цикла в переменную `answer` сохраняется результат работы метода `get_answer()`, который и возвращается в качестве результата.
- `get_answer()` – используется для преобразования массива строк в единую строку. В начале объявляется пустая строковая переменная `answer`, которой впоследствии прибавляются все элементы списка строк `memory`. После этого переменная `answer` возвращается в качестве результата.

Решение второй подзадачи.

Формат представления таблицы следующий:

```
table = {
    <state 1>: {
        <char 1>: {'write': <some char>, 'direct': <0, 1, -1>, 'state': <next state>},
        ...
```

```
        <char n>: {...},
    },
    ...
    <state n>: ...
}
```

В программе сначала создается словарь для каждого состояния, затем все полученные словари объединяются в один – `program`. Ниже описана Таблица 1 – таблица состояний:

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

Краткое описание состояний:

- q0 – начальное состояние, которое необходимо, чтобы определить знак арифметической операции.
- q1 – определяет цифру, с которой складывается число.
- q2 – прибавляет к текущей цифре числа 1.
- q3 – прибавляет к текущей цифре числа 2.
- q4 – определяет цифру, которая вычитается из числа.
- q5 – вычитает из текущей цифры числа 1.
- q6 – вычитает из текущей цифры числа 2.

- q7 – стирает незначащие нули.
- q8 – если результатом арифметической операции является нуль, записывает его.
- q9 – переводит указатель на первую цифру числа.

Выполнение третьей подзадачи.

Данные вводятся с помощью функции `input()`, после чего преобразовываются в список строк. Копия списка и программа передаются классу `T_machine`.

Выполнение четвертой подзадачи.

Вызывается метод объекта `machine start()`, результат работы которого сохраняется в переменной `answer` и выводится на экран.

Тестирование.

Здесь результаты тестирования, которые помещаются на одну страницу. Результаты тестирования представлены в табл. 1.

Таблица 2 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1	111+1	112+1	С задачей программа справилась успешно
2	222+2	1001+2	С задачей программа справилась успешно
3	212+1	220+1	С задачей программа справилась успешно

Выводы.

С помощью языка Python была смоделирована работа машины Тьюринга. Для построения модели использовалось объектно-ориентированное программирование – был создан класс `T_machine`, позволяющий

моделировать выполнение любой корректно написанной программы на машине Тьюринга. Была написана простейшая программа для сложения и вычитания троичного числа и троичной цифры.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: `main.py`

Расположение: `src/main.py`

```
class T_machine():
      def_init_(self, string, program, init_state='q0'):
            self.memory = string
            self.table = program
            self.state = init_state
            self.index = 0

      def start(self):
            while self.state != 'qt':
                  st=self.table[self.state][self.memory[self.index]]
                  self.memory[self.index] = st['write']
                  self.index += st['direct']
                  self.state = st['state']
            answer = self.get_answer()
            return answer

      def get_answer(self):
            answer = ''
            for i in range(len(self.memory)):
                  answer += self.memory[i]
            return answer

q_0 = {
      '0': {'write': '0', 'direct': 1, 'state': 'q0'},
      '1': {'write': '1', 'direct': 1, 'state': 'q0'},
      '2': {'write': '2', 'direct': 1, 'state': 'q0'},
      '+': {'write': '+', 'direct': 1, 'state': 'q1'},
      '-': {'write': '-', 'direct': 1, 'state': 'q4'},
      ' ': {'write': ' ', 'direct': 1, 'state': 'q0'},
}

q_1 = {
      '0': {'write': '0', 'direct': 0, 'state': 'qt'},
```

```
      '1': {'write': '1', 'direct': -1, 'state': 'q2'},
      '2': {'write': '2', 'direct': -1, 'state': 'q3'},
      '+': {},
      '-': {},
      ' ': {},
}

q_2 = {
      '0': {'write': '1', 'direct': 0, 'state': 'qt'},
      '1': {'write': '2', 'direct': 0, 'state': 'qt'},
      '2': {'write': '0', 'direct': -1, 'state': 'q2'},
      '+': {'write': '+', 'direct': -1, 'state': 'q2'},
      '-': {},
      ' ': {'write': '1', 'direct': 0, 'state': 'qt'},
}

q_3 = {
      '0': {'write': '2', 'direct': 0, 'state': 'qt'},
      '1': {'write': '0', 'direct': -1, 'state': 'q2'},
      '2': {'write': '1', 'direct': -1, 'state': 'q2'},
      '+': {'write': '+', 'direct': -1, 'state': 'q3'},
      '-': {},
      ' ': {},
}

q_4 = {
      '0': {'write': '0', 'direct': 0, 'state': 'qt'},
      '1': {'write': '1', 'direct': -1, 'state': 'q5'},
      '2': {'write': '2', 'direct': -1, 'state': 'q6'},
      '+': {},
      '-': {},
      ' ': {},
}

q_5 = {
      '0': {'write': '2', 'direct': -1, 'state': 'q5'},
      '1': {'write': '0', 'direct': 0, 'state': 'q9'},
      '2': {'write': '1', 'direct': 0, 'state': 'qt'},
      '+': {},
```

```
      '-': {'write': '-', 'direct': -1, 'state': 'q5'},
      ' ': {},
}

q_6 = {
      '0': {'write': '1', 'direct': -1, 'state': 'q5'},
      '1': {'write': '2', 'direct': -1, 'state': 'q5'},
      '2': {'write': '0', 'direct': 0, 'state': 'q9'},
      '+': {},
      '-': {'write': '-', 'direct': -1, 'state': 'q6'},
      ' ': {},
}

q_7 = {
      '0': {'write': ' ', 'direct': 1, 'state': 'q7'},
      '1': {'write': '1', 'direct': 0, 'state': 'qt'},
      '2': {'write': '2', 'direct': 0, 'state': 'qt'},
      '+': {'write': '+', 'direct': -1, 'state': 'q8'},
      '-': {'write': '-', 'direct': -1, 'state': 'q8'},
      ' ': {},
}

q_8 = {
      '0': {'write': '0', 'direct': 0, 'state': 'qt'},
      '1': {'write': '1', 'direct': 0, 'state': 'qt'},
      '2': {'write': '2', 'direct': 0, 'state': 'qt'},
      '+': {'write': '+', 'direct': 0, 'state': 'qt'},
      '-': {'write': '-', 'direct': 0, 'state': 'qt'},
      ' ': {'write': '0', 'direct': 0, 'state': 'qt'},
}

q_9 = {
      '0': {'write': '0', 'direct': -1, 'state': 'q9'},
      '1': {'write': '1', 'direct': -1, 'state': 'q9'},
      '2': {'write': '2', 'direct': -1, 'state': 'q9'},
      '+': {},
      '-': {},
      ' ': {'write': ' ', 'direct': 1, 'state': 'q7'},
}
```

```
program = {
      'q0': q_0,
      'q1': q_1,
      'q2': q_2,
      'q3': q_3,
      'q4': q_4,
      'q5': q_5,
      'q6': q_6,
      'q7': q_7,
      'q8': q_8,
      'q9': q_9,
}

request = list(input())
machine = T_machine(request.copy(), program)
answer = machine.start()
print(answer)
```