**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №3**

**по дисциплине «Информатика»**

Тема: Машина Тьюринга

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3343 |  | Лобова Е. И. |
| Преподаватель |  | Иванов Д. В. |

Санкт-Петербург

2023

## Цель работы

Целью работы является реализация машины Тьюринга на Python для моделирования работы вычислительного устройства.

## Задание

​**Вариант 3**

На вход программе подается строка неизвестной длины. Каждый элемент является значением в ячейке памяти ленты Машины Тьюринга.

На ленте находится последовательность латинских букв из алфавита {a, b, c}.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | a | c | c | a | b | c | b | a | b | a | a | c | a | b |  |  |  |

Напишите программу, которая заменяет в исходной строке символ, предшествующий первому встретившемуся символу ‘c' на символ, следующий за первым встретившимся символом ‘a'. Если первый встретившийся символ ‘a' в конце строки, то используйте его в качестве заменяющего.

Указатель на текущее состояние Машины Тьюринга изначально находится слева от строки с символами (но не на первом ее символе). По обе стороны от строки находятся пробелы.

Для примера выше лента будет выглядеть так:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | c | c | c | a | b | c | b | a | b | a | a | c | a | b |  |  |  |

Алфавит:

* a
* b
* c
* " " (пробел)

Соглашения:

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

2. Гарантируется, что длинна строки не менее 5 символов и не более 15.

3. В середине строки не могут встретиться пробелы.

4. При удалении или вставке символов направление сдвигов подстрок не принципиально (т. е. результат работы алгоритма может быть сдвинут по ленте в любую ее сторону на любое число символов).

5. Курсор по окончании работы алгоритма может находиться на любом символе.

Ваша программа должна вывести полученную ленту после завершения работы.

## Выполнение работы

Таблица 1 - Таблица состояний

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 'a' | 'b' | 'c' | ' ' |
| q1 | 'a', R, 'q2' | 'b', R, 'q1' | 'c', R, 'q1' | ' ', R, 'q1' |
| q2 | 'a', L, 'q3' | 'b', L, 'q4' | 'c', L, 'q5' | ' ', L, 'q3' |
| q3 | 'a', L, 'q3' | 'b', L, 'q3' | 'c', L, 'q3' | ' ', R, 'q9' |
| q4 | 'a', L, 'q4' | 'b', L, 'q4' | 'c', L, 'q4' | ' ', R, 'q10' |
| q5 | 'a', L, 'q5' | 'b', L, 'q5' | 'c', L, 'q5' | ' ', R, 'q11' |
| q6 | 'a', N, 'qT' | 'a', N, 'qT' | 'a', N, 'qT' | 'a', N, 'qT' |
| q7 | 'b', N, 'qT' | 'b', N, 'qT' | 'b', N, 'qT' | 'b', N, 'qT' |
| q8 | 'c', N, 'qT' | 'c', N, 'qT' | 'c', N, 'qT' | 'c', N, 'qT' |
| q9 | 'a', R, 'q9' | 'b', R, 'q9' | 'c', L, 'q6' |  |
| q10 | 'a', R, 'q10' | 'b', R, 'q10' | 'c', L, 'q7' |  |
| q11 | 'a', R, 'q11' | 'b', R, 'q11' | 'c', L, 'q8' |  |

Описание состояний:

* q1 - начальное состояние, которое необходимо, чтобы найти первый встретившийся символ ‘a’.
* q2 – состояние, которое определяет символ, следующий за первым встретившимся символом ‘a’.
* q3 – состояние для ‘a’, которое возвращает в начало строки.
* q4 - состояние для ‘b’, которое возвращает в начало строки.
* q5 - состояние для ‘c’, которое возвращает в начало строки.
* q6 – состояние, которое заменяет символ, предшествующий первому встретившемуся символу ‘c' на ‘a’.
* q7 - состояние, которое заменяет символ, предшествующий первому встретившемуся символу ‘c' на ‘b’.
* q8 - состояние, которое заменяет символ, предшествующий первому встретившемуся символу ‘c' на ‘c’.
* q9 – состояние, которое необходимо, чтобы найти первый встретившийся символ ‘с’ и впоследствии заменить предшествующий ему на ‘a’.
* q10 - состояние, которое необходимо, чтобы найти первый встретившийся символ ‘с’ и впоследствии заменить предшествующий ему на ‘b’.
* q11 - состояние, которое необходимо, чтобы найти первый встретившийся символ ‘с’ и впоследствии заменить предшествующий ему на ‘c’.

Принцип работы Машины Тьюринга в коде:

* memory – введенная строка;
* table – таблица состояний, заданная словарем;
* state – текущее состояние, изначально q1;
* index – индекс текущей ячейки, изначально 0;
* С помощью цикла while и таблицы состояний строка изменяется согласно условию.

Разработанный программный код см. в приложении А.

## Тестирование

Результаты тестирования представлены в табл. 2.

Таблица 2 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | abcabc | abcabc | Программа работает корректно для случая, когда  ‘c’ после ‘a’ и между ними один символ. |
|  | cbbaa | acbbaa | Программа работает корректно для случая, когда перед ‘c’пробел. |
|  | bbbbccca | bbbaccca | Программа работает корректно, для случая, когда после ‘a’пробел. |

## Выводы

Была реализована машины Тьюринга на Python для моделирования работы вычислительного устройства.

С помощью словаря была создана таблица состояний, а с помощью цикла while сымитирована работа машины Тьюринга.

# Приложение А Исходный код программы

Название файла: main.py

L, R, N = -1, 1, 0

table = {'q1':{'a':['a', R, 'q2'], 'b': ['b', R, 'q1'], 'c': ['c', R, 'q1'], ' ':[' ', R, 'q1']},

'q2':{'a':['a', L, 'q3'], 'b': ['b', L, 'q4'], 'c': ['c', L, 'q5'], ' ':[' ', L, 'q3']},

'q3':{'a':['a', L, 'q3'], 'b': ['b', L, 'q3'], 'c': ['c', L, 'q3'], ' ': [' ', R, 'q9']},

'q4':{'a':['a', L, 'q4'], 'b': ['b', L, 'q4'], 'c': ['c', L, 'q4'], ' ': [' ', R, 'q10']},

'q5':{'a':['a', L, 'q5'], 'b': ['b', L, 'q5'], 'c': ['c', L, 'q5'], ' ': [' ', R, 'q11']},

'q6':{'a':['a', N, 'qT'], 'b': ['a', N, 'qT'], 'c': ['a', N, 'qT'], ' ': ['a', N, 'qT']},

'q7':{'a':['b', N, 'qT'], 'b': ['b', N, 'qT'], 'c': ['b', N, 'qT'], ' ': ['b', N, 'qT']},

'q8':{'a':['c', N, 'qT'], 'b': ['c', N, 'qT'], 'c': ['c', N, 'qT'], ' ': ['c', N, 'qT']},

'q9':{'a':['a', R, 'q9'], 'b': ['b', R, 'q9'], 'c': ['c', L, 'q6']},

'q10':{'a':['a', R, 'q10'], 'b': ['b', R, 'q10'], 'c': ['c', L, 'q7']},

'q11':{'a':['a', R, 'q11'], 'b': ['b', R, 'q11'], 'c': ['c', L, 'q8']}}

memory = list(input())

state = 'q1'

index = 0

while state != 'qT':

current\_symbol = memory[index]

future = table[state][current\_symbol]

memory[index] = future[0]

index += future[1]

state = future[2]

print(\*memory, sep='')