**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

|  |  |  |
| --- | --- | --- |
| Студент гр. 2381 |  | Рыжиков И.С. |
| Преподаватель |  | Шевская Н.В. |

Санкт-Петербург

2022

## Цель работы

Изучить принципы работы машины Тьюринга. Написать собственный автомат для выполнения задания.

## Задание

Вариант 2

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

**Напишите программу, которая заменяет в исходной строке символ, идущий после последних двух встретившихся символов 'a', на предшествующий им символ(гарантируется, что это не пробел). Наличие в строке двух подряд идущих символов 'a' гарантируется.**

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:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | a | c | a | a | b | c | b | a | b | a | a | b | 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.**

В отчете предоставьте таблицу состояний. Отдельно кратко опишите каждое состояние, например:

q1 - начальное состояние, которое необходимо, чтобы найти первый встретившийся символ ‘a’.

Пример:

| **Test** | **Input** | **Result** |
| --- | --- | --- |
| #1 | abcaabc | abcaacc |
| #2 | aabbaa | aabbaab |

## Выполнение работы

### Описание состояний

* q0 — конечное состояние;
* q1 — поиск начала слова;
* q2 — переход в конец слова;
* q3 — найти один символ 'a';
* q4 — найти последние два подряд идущих символа 'a';
* q5 — определить символ, на которой требуется заменить;
* q6 и q7, q9 и q10, q12 и q13 — перейти на два шага вперед, сохраняя информацию о заменяющем символе;
* q8, q11, q14 — заменить символ на 'a', 'b' или 'c' соответственно.

Таблица 1. Состояния

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Состояние | 'a' | 'b' | 'c' | ' ' (пробел) |
| q1 | 'a', R, 'q2' | 'b', R, 'q2' | 'c', R, 'q2' | ' ', R, 'q1' |
| q2 | 'a', R, 'q2' | 'b', R, 'q2' | 'c', R, 'q2' | ' ', L, 'q3' |
| q3 | 'a', L, 'q4' | 'b', L, 'q3' | 'c', L, 'q3' |  |
| q4 | 'a', L, 'q5' | 'b', L, 'q3' | 'c', L, 'q3' |  |
| q5 | 'a', R, 'q6' | 'b', R, 'q9' | 'c', R, 'q12' |  |
| q6 | 'a', R, 'q7' |  |  |  |
| q7 | 'a', R, 'q8' |  |  |  |
| q8 | 'a', N, 'q0' | 'a', N, 'q0' | 'a', N, 'q0' | 'a', N, 'q0' |
| q9 | 'a', R, 'q10' |  |  |  |
| q10 | 'a', R, 'q11' |  |  |  |
| q11 | 'b', N, 'q0' | 'b', N, 'q0' | 'b', N, 'q0' | 'b', N, 'q0' |
| q12 | 'a', R, 'q13' |  |  |  |
| q13 | 'a', R, 'q14' |  |  |  |
| q14 | 'c', N, 'q0' | 'c', N, 'q0' | 'c', N, 'q0' | 'c', N, 'q0' |

### Принцип работы программы

Сохраняем введенную из консоли строку в переменную tape в виде списка символов. Передаем её в функцию turing, которая в соответствии с таблицей состояний выполняет программу. Для чего в ней локально создается переменная state, хранящее текущее состояние машины, i — индекс текущего символа в списке, и move — шаг. Запускаем внутри функции цикл до тех пор, пока состояние машины не будет равно конечному. Записываем новый символ на ленту, делаем шаг и обновляем текущее состояние системы. После выполнения функции turing, которая имитирует работу машины Тьюринга, выводим получившуюся строку на экран.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | ' aaccaab ' | ' aaccaac ' | OK |
|  | ' abcaabc ' | ' abcaacc ' | OK |
|  | ' aabbaa ' | ' aabbaab ' | OK |
|  | ' acbaa ' | ' acbaab ' | OK |

## Выводы

Была изучена машина Тьюринга. Создана функция, которая заменяет в исходной строке символ, идущий после последних двух встретившихся символов 'a', на предшествующий им символ. Разработана программа на языке Python, имитирующая работу машины Тьюринга.

# Приложение А Исходный код программы

Название файла: main.py

*def* turing(tape):  
 table = {  
 *# Find a word* 'q1': {  
 ' ': [' ', 1, 'q1'],  
 'a': ['a', 1, 'q2'],  
 'b': ['b', 1, 'q2'],  
 'c': ['c', 1, 'q2']  
 },  
 *# Go to the end of word* 'q2': {  
 ' ': [' ', -1, 'q3'],  
 'a': ['a', 1, 'q2'],  
 'b': ['b', 1, 'q2'],  
 'c': ['c', 1, 'q2']  
 },  
 *# Find one 'a'* 'q3': {  
 'a': ['a', -1, 'q4'],  
 'b': ['b', -1, 'q3'],  
 'c': ['c', -1, 'q3']  
 },  
 *# Find the last couple of 'a' in a row* 'q4': {  
 'a': ['a', -1, 'q5'],  
 'b': ['b', -1, 'q3'],  
 'c': ['c', -1, 'q3']  
 },  
 *# Branching: Define a replacement character* 'q5': {  
 'a': ['a', 1, 'q6'],  
 'b': ['b', 1, 'q9'],  
 'c': ['c', 1, 'q12']  
 },  
 *# Branch of 'a':  
 # Go back* 'q6': {  
 'a': ['a', 1, 'q7']  
 },  
 'q7': {  
 'a': ['a', 1, 'q8']  
 },  
 *# Replace with 'a'* 'q8': {  
 'a': ['a', 0, 'q0'],  
 'b': ['a', 0, 'q0'],  
 'c': ['a', 0, 'q0'],  
 ' ': ['a', 0, 'q0']  
 },  
 *# Branch of 'b':  
 # Go back* 'q9': {  
 'a': ['a', 1, 'q10']  
 },  
 'q10': {  
 'a': ['a', 1, 'q11']  
 },  
 *# Replace with 'b'* 'q11': {  
 'a': ['b', 0, 'q0'],  
 'b': ['b', 0, 'q0'],  
 'c': ['b', 0, 'q0'],  
 ' ': ['b', 0, 'q0']  
 },  
 *# Branch of 'c':  
 # Go back* 'q12': {  
 'a': ['a', 1, 'q13']  
 },  
 'q13': {  
 'a': ['a', 1, 'q14']  
 },  
 *# Replace with 'c'* 'q14': {  
 'a': ['c', 0, 'q0'],  
 'b': ['c', 0, 'q0'],  
 'c': ['c', 0, 'q0'],  
 ' ': ['c', 0, 'q0']  
 },  
 }  
 state = 'q1'  
 i = 0  
 *while* state != 'q0':  
 tape[i], move, state = table[state][tape[i]]  
 i += move  
  
  
*if* \_\_name\_\_ == '\_\_main\_\_':  
 tape = *list*(*input*())  
 turing(tape)  
 *print*(''.join(tape))