**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**](https://e.moevm.info/mod/quiz/view.php?id=2027)

|  |  |  |
| --- | --- | --- |
| Студент гр. 3342 |  | Русанов А.И. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2023

## Цель работы

Целью работы реализация машины Тьюринга на языке программирования Python.

## 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}.

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.

Алфавит:

a

b

c

" " (пробел)

Соглашения:

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

2. Гарантируется, что длина 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

На основе условия задачи была составлена таблица состояний №1.

Таблица 1: Таблица состояний №1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | «a» | «b» | «c» | « » |
| q0 | «a», R, q1 | «c», R, q0 | «c», R, q0 | « », R, q0 |
| q1 | «a», L, qa- | «b», L, qb- | «c», L, qc- | « », L, qa- |
| qa- | «a», L, qa- | «b», L, qa- | «c», L, qa- | « », R, qa+ |
| qa+ | «a», R, qa+ | «b», R, qa+ | «c», L, qaw |  |
| qaw | «a», N, qend | «a», N, qend | «a», N, qend | «a», N, qend |
| qb- | «a», L, qb- | «b», L, qb- | «c», L, qb- | « », R, qb+ |
| qb+ | «a», R, qb+ | «b», R, qb+ | «c», L, qbw |  |
| qbw | «b», N, qend | «b», N, qend | «b», N, qend | «b», N, qend |
| qc- | «a», L, qc- | «b», L, qc- | «c», L, qc- | « », R, qc+ |
| qc+ | «a», R, qc+ | «b», R, qc+ | «c», L, qcw |  |
| qcw | «c», N, qend | «c», N, qend | «c», N, qend | «c», N, qend |

* «q0» - начальное состояние, поиск первого символа «а»
* «q1» - нахождение символа, стоящего перед «а»
* «qa-», «qb-», «qc-» - в зависимости от найденного символа, возвращение к началу ленты
* «qa+», «qb+», «qc+» - поиск первого символа «с»
* «qaw», «qbw», «qcw» - перезаписывание символа, стоящего перед «с», на символ, найденный после символа «а»
* «qend» - конечное положение

## Тестирование

Результаты тестирования представлены в табл. 2.

Таблица 2 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | abcabc | abcabc | Функция работает корректно |
|  | cbbaa | acbbaa | Функция работает корректно |
|  | cbbab | bcbbab | Функция работает корректно |

## Выводы

Была смоделирована работа машины Тьюринга для решения поставленной задачи.

# Приложение А Исходный код программы

Название файла: main.py

table = {'q0': {'a': ['a', 1, 'q1'], 'b': ['b', 1, 'q0'], 'c': ['c', 1, 'q0'], ' ': [' ', 1, 'q0']},

'q1': {'a': ['a', -1, 'qa-'], 'b': ['b', -1, 'qb-'], 'c': ['c', -1, 'qc-'], ' ': [' ', -1, 'qa-']},

'qa-': {'a': ['a', -1, 'qa-'], 'b': ['b', -1, 'qa-'], 'c': ['c', -1, 'qa-'], ' ': [' ', 1, 'qa+']},

'qa+': {'a': ['a', 1, 'qa+'], 'b': ['b', 1, 'qa+'], 'c': ['c', -1, 'qaw'], ' ': []},

'qaw': {'a': ['a', 0, 'qend'], 'b': ['a', 0, 'qend'], 'c': ['a', 0, 'qend'], ' ': ['a', 0, 'qend']},

'qb-': {'a': ['a', -1, 'qb-'], 'b': ['b', -1, 'qb-'], 'c': ['c', -1, 'qb-'], ' ': [' ', 1, 'qb+']},

'qb+': {'a': ['a', 1, 'qb+'], 'b': ['b', 1, 'qb+'], 'c': ['c', -1, 'qbw'], ' ': []},

'qbw': {'a': ['b', 0, 'qend'], 'b': ['b', 0, 'qend'], 'c': ['b', 0, 'qend'], ' ': ['b', 0, 'qend']},

'qc-': {'a': ['a', -1, 'qc-'], 'b': ['b', -1, 'qc-'], 'c': ['c', -1, 'qc-'], ' ': [' ', 1, 'qc+']},

'qc+': {'a': ['a', 1, 'qc+'], 'b': ['b', 1, 'qc+'], 'c': ['c', -1, 'qcw'], ' ': []},

'qcw': {'a': ['c', 0, 'qend'], 'b': ['c', 0, 'qend'], 'c': ['c', 0, 'qend'], ' ': ['c', 0, 'qend']},

}

memory = list(input())

q = 'q0'

index = 0

while q != 'qend' and index < len(memory):

s, move, state = table[q][memory[index]]

memory[index] = s

index += move

q = state

print(\*memory, sep='')