

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. 3343

Пивоев Н.М.

Преподаватель

Иванов Д.В.

Санкт-Петербург

2023

Цель работы

Ознакомление с устройством Машины Тьюринга и создание программы на языке Python на основе этого механизма.

Задание

Вариант 4

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}, которая начинается с символа 'a'.

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

Напишите программу, которая оборачивает исходную строку. Результат работы алгоритма - исходная последовательность символов в обратном порядке.

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:

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

Алфавит (можно расширять при необходимости):

- 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 13.

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.

6. Нельзя использовать дополнительную ленту, в которую записывается результат.

Ваша программа должна вывести полученную ленту после завершения работы.

В отчет включите таблицу состояний. Отдельно кратко опишите каждое состояние, например:

q1 - начальное состояние, которое необходимо, чтобы обнаружить конец строки.

Выполнение работы

Созданный проект включает таблицу состояний и её обработку.

	‘a’	‘b’	‘c’	‘d’	‘ ’
‘q1’	‘a’, R, ‘q2’	‘b’, R, ‘q2’	‘c’, R, ‘q2’		‘ ’, R, ‘q1’
‘q2’	‘a’, R, ‘q2’	‘b’, R, ‘q2’	‘c’, R, ‘q2’		‘ ’, N, ‘q3’
‘q3’	‘d’, N, ‘q4’	‘d’, N, ‘q6’	‘d’, N, ‘q8’		‘ ’, L, ‘q3’
‘q4’	‘a’, R, ‘q4’	‘b’, R, ‘q4’	‘c’, R, ‘q4’	‘d’, R, ‘q4’	‘ ’, R, ‘q5’
‘q5’	‘a’, R, ‘q5’	‘b’, R, ‘q5’	‘c’, R, ‘q5’		‘a’, L, ‘q10’
‘q6’	‘a’, R, ‘q6’	‘b’, R, ‘q6’	‘c’, R, ‘q6’	‘d’, R, ‘q6’	‘ ’, R, ‘q7’
‘q7’	‘a’, R, ‘q7’	‘b’, R, ‘q7’	‘c’, R, ‘q7’		‘b’, L, ‘q10’
‘q8’	‘a’, R, ‘q8’	‘b’, R, ‘q8’	‘c’, R, ‘q8’	‘d’, R, ‘q8’	‘ ’, R, ‘q9’
‘q9’	‘a’, R, ‘q9’	‘b’, R, ‘q9’	‘c’, R, ‘q9’		‘c’, L, ‘q10’
‘q10’	‘a’, L, ‘q10’	‘b’, L, ‘q10’	‘c’, L, ‘q10’		‘ ’, L, ‘q11’
‘q11’	‘d’, N, ‘q4’	‘d’, N, ‘q6’	‘d’, N, ‘q8’	‘d’, L, ‘q11’	‘ ’, R, ‘q12’
‘q12’				‘ ’, R, ‘q12’	‘ ’, N, ‘end’

Таблица состояний включает 12 различных состояний для машины Тьюринга:

‘q1’ – переход к первому символу строки

‘q2’ – переход к первому пробелу после строки

‘q3’ – переход налево до первого символа строки и замена его на ‘d’, в зависимости от заменённого символа вызов ‘q4’, ‘q6’ или ‘q8’

‘q4’ – переход к первому символу перевёрнутой строки

‘q5’ – переход к первому пробелу после перевёрнутой строки, запись в эту позицию ‘a’ и переход к концу строки

‘q6’ – переход к первому символу перевёрнутой строки

‘q7’ – переход к первому пробелу после перевёрнутой строки, запись в эту позицию ‘b’ и переход к концу строки

‘q8’ – переход к первому символу перевёрнутой строки

‘q9’ – переход к первому пробелу после перевёрнутой строки, запись в эту позицию ‘c’ и переход к концу строки

‘q10’ – переход от конца перевёрнутой строки к концу изначальной

‘q11’ – обход строки, в случае присутствия ‘a’, ‘b’ или ‘c’ вызов предыдущих состояний, иначе перемещение в начало строки

‘q12’ – удаление всех символов ‘d’, выход из алгоритма

В начале идёт описание таблицы состояний в виде словаря, где ключ – состояние, значение – ещё один словарь, в котором ключ – символ на ленте, а значение – список, включающий новый символ, направление движения и новое состояние.

На вход подаётся строка неизвестной длины, которая сохраняется в списке. Для корректности работы добавляется произвольное количество пробелов с обеих сторон. Далее идёт обработка состояний. Вся текущая информация хранится в *dictionary*. Затем обновляется значение на ленте в текущей позиции на новый символ; считывающая каретка при необходимости перемещается на соседнюю ячейку; изменяется текущее состояние. Когда происходит переход в состояние ‘end’, обработка заканчивается и выводится инвертированная строка.

Разработанный программный код см. в приложении А.

Тестирование

Результаты тестирования содержатся в таблице 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	aabbcc	ccbbaa	Код работает исправно
2.	bacbca	acbcab	Код работает исправно
3.	abacaba	abacaba	Код работает исправно

Выводы

В результате работы был изучен механизм работы машины Тьюринга. Реализованный проект на её основе успешно выполняет поставленную задачу, направленную на инвертирование строки.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
L, N, R = -1, 0, 1
state_table = {'q1':
{'a': ['a', R, 'q2'], 'b': ['b', R, 'q2'], 'c': ['c', R, 'q2'], ' ': [' ', R, 'q1']}},
{'q2': {'a': ['a', R, 'q2'], 'b': ['b', R, 'q2'], 'c': ['c', R, 'q2'], ' ': [' ', N, 'q3']}},
{'q3': {'a': ['d', N, 'q4'], 'b': ['d', N, 'q6'], 'c': ['d', N, 'q8'], ' ': [' ', L, 'q3']}},
{'q4':
{'a': ['a', R, 'q4'], 'b': ['b', R, 'q4'], 'c': ['c', R, 'q4'], 'd': ['d', R, 'q4'], ' ': [' ', R, 'q5']}},
{'q5': {'a': ['a', R, 'q5'], 'b': ['b', R, 'q5'], 'c': ['c', R, 'q5'], ' ': ['a', L, 'q10']}},
{'q6':
{'a': ['a', R, 'q6'], 'b': ['b', R, 'q6'], 'c': ['c', R, 'q6'], 'd': ['d', R, 'q6'], ' ': [' ', R, 'q7']}},
{'q7': {'a': ['a', R, 'q7'], 'b': ['b', R, 'q7'], 'c': ['c', R, 'q7'], ' ': ['b', L, 'q10']}},
{'q8':
{'a': ['a', R, 'q8'], 'b': ['b', R, 'q8'], 'c': ['c', R, 'q8'], 'd': ['d', R, 'q8'], ' ': [' ', R, 'q9']}},
{'q9': {'a': ['a', R, 'q9'], 'b': ['b', R, 'q9'], 'c': ['c', R, 'q9'], ' ': ['c', L, 'q10']}},
{'q10':
{'a': ['a', L, 'q10'], 'b': ['b', L, 'q10'], 'c': ['c', L, 'q10'], ' ': [' ', L, 'q11']}},
{'q11':
{'a': ['d', N, 'q4'], 'b': ['d', N, 'q6'], 'c': ['d', N, 'q8'], 'd': ['d', L, 'q11'], ' ': [' ', R, 'q12']}},
{'q12': {'d': [' ', R, 'q12'], ' ': [' ', N, 'end']}}

array = list(input())
state = 'q1'
index = 0
spaces = list(' '*20)
```

```
array = spaces + array + spaces

while state != 'end':
    dictionary = state_table[state][array[index]]
    array[index] = dictionary[0]
    index += dictionary[1]
    state = dictionary[2]

print(*array,                                     sep='')
```