**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Информатика»**

Тема: **Моделирование работы Машины Тьюринга**

|  |  |  |
| --- | --- | --- |
| Студент гр. 8381 |  | Киреев К.А. |
| Преподаватель |  | Размочаева Н.В. |

Санкт-Петербург

2018

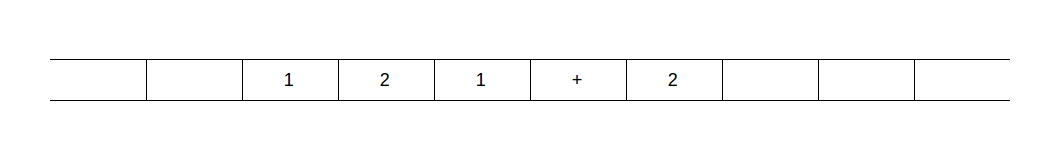
**Цель работы**

Изучить основные принципы работы машины Тьюринга. Изучить действие поразрядных операций.

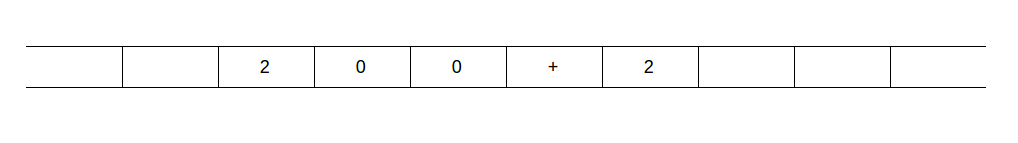
**Задание**

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. Для примера выше лента будет выглядеть так:

Ваша программа должна вывести полученную ленту после завершения работы.

Алфавит:

* 0
* 1
* 2
* +
* -
* " " (пробел)

Соглашения:

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

2. Число обязательно начинается с единицы или двойки.

3. Числа и знак операции между ними идут непрерывно.

4. Гарантируется, что в результате операции вычитания не может получиться отрицательного числа.

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

q1 - начальное состояние, которое необходимо, чтобы найти первую цифру первого числа.

**Выполнение работы**

1. Вначале проводится считывание списка memory. Начальное состояние state устанавливается в значение q1.

2. Далее пока состояние не будет равно q12, для каждого if вызывается функция q1 – q11

3. В каждой функции в зависимости от условия может меняться значение элемента списка, положение автомата или осуществляться переход в новое состояние.

4. В конце выводится обновленный список.

**Выводы**

В ходе лабораторной работы были изучены основы использования поразрядных операций. Приобретены знания работы и применения машины Тьюринга

**Приложение А. Исходный код**

def q1(memory, index):

if (memory[index] == '+'):

return memory, index + 1, 'q2'

if (memory[index] == '-'):

return memory, index + 1, 'q3'

else:

return memory, index + 1, 'q1'

def q2(memory, index):

if memory[index] == '0':

return memory, index, 'q12'

if memory[index] == '1':

return memory, index - 1, 'q4'

if memory[index] == '2':

return memory, index - 1, 'q5'

def q3(memory, index):

if memory[index] == '0':

return memory, index, 'q12'

if memory[index] == '1':

return memory, index - 1, 'q6'

if memory[index] == '2':

return memory, index - 1, 'q7'

def q4(memory, index):

list1 = ['0', '1']

if memory[index] in list1:

memory[index] = str(int(memory[index]) + 1)

return memory, index, 'q12'

if memory[index] == '2':

memory[index] = '0'

return memory, index - 1, 'q4'

if (memory[index] == '+') | (memory[index] == '-'):

return memory, index - 1, 'q4'

if memory[index] == ' ':

memory[index] = '1'

return memory, index, 'q12'

def q5(memory, index):

list1 = ['1', '2']

if memory[index] == '0':

memory[index] = '2'

return memory, index, 'q12'

if memory[index] in list1:

memory[index] = str(int(memory[index]) - 1)

return memory, index - 1, 'q4'

if (memory[index] == '+') | (memory[index] == '-'):

return memory, index - 1, 'q5'

def q6(memory, index):

list1 = ['+', '-']

if memory[index] == '0':

memory[index] = '2'

return memory, index - 1, 'q6'

if memory[index] == '1':

memory[index] = '0'

return memory, index - 1, 'q8'

if memory[index] == '2':

memory[index] = '1'

return memory, index, 'q12'

if memory[index] in list1:

return memory, index - 1, 'q6'

if memory[index] == ' ':

return memory, index, 'q12'

def q7(memory, index):

list1 = ['+', '-']

if memory[index] == '0':

memory[index] = '1'

return memory, index - 1, 'q6'

if memory[index] == '1':

memory[index] = '2'

return memory, index - 1, 'q6'

if memory[index] == '2':

memory[index] = '0'

return memory, index, 'q12'

if memory[index] in list1:

return memory, index - 1, 'q7'

if memory[index] == ' ':

return memory, index, 'q12'

def q8(memory, index):

if memory[index] == ' ':

return memory, index + 1, 'q9'

else:

return memory, index - 1, 'q8'

def q9(memory, index):

if memory[index] == '0':

return memory, index + 1, 'q10'

else:

return memory, index, 'q12'

def q10(memory, index):

list1 = ['1', '2', '0']

list2 = ['+', '-']

if memory[index] in list1:

return memory, index - 1, 'q11'

if memory[index] in list2:

return memory, index + 1, 'q12'

def q11(memory, index):

if memory[index] == '0':

memory[index] = ' '

return memory, index, 'q12'

memory = list(input())

state = 'q1'

i = 0

while state != 'q12':

if state == 'q1':

memory, i, state = q1(memory, i)

if state == 'q2':

memory, i, state = q2(memory, i)

if state == 'q3':

memory, i, state = q3(memory, i)

if state == 'q4':

memory, i, state = q4(memory, i)

if state == 'q5':

memory, i, state = q5(memory, i)

if state == 'q6':

memory, i, state = q6(memory, i)

if state == 'q7':

memory, i, state = q7(memory, i)

if state == 'q8':

memory, i, state = q8(memory, i)

if state == 'q9':

memory, i, state = q9(memory, i)

if state == 'q10':

memory, i, state = q10(memory, i)

if state == 'q11':

memory, i, state = q11(memory, i)

print(''.join(memory))

**Приложение Б. Таблица состояний**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | + | - | ‘ \_‘ |
| q1 | 0;R;q1 | 1;R;q1 | 2;R;q1 | +;R;q2 | -;R;q3 | \_;R;q1 |
| q2 | 0;N;q12 | 1;L;q4 | 2;L;q5 |  |  |  |
| q3 | 0;N;q12 | 1;L;q6 | 1;L;q7 |  |  |  |
| q4 | 1;N;q12 | 2;N;q12 | 0;L;q4 | +;L;q4 | -;L;q4 | 1;N;q12 |
| q5 | 2;N;q12 | 0;L;q4 | 1;L;q4 | +;L;q5 | -;L;q5 |  |
| q6 | 2;L;q6 | 0;L;q8 | 1;N;q12 | +;L;q6 | -;L;q6 | \_;N;q12 |
| q7 | 1;L;q6 | 2;L;q6 | 0;N;q12 | +;L;q7 | -;L;q7 | \_;N;q12 |
| q8 | 0;L;q8 | 1;L;q8 | 2;L;q8 | +;L;q8 | -;L;q8 | \_;R;q9 |
| q9 | 0;R;q10 | 1;N;q12 | 2;N;q12 | +;N;q12 | -;N;q12 | \_;N;q12 |
| q10 | 0;L;q11 | 1;L;q11 | 2;L;q11 | +;R;q12 | -;R;q12 |  |
| q11 | \_;N;q12 |  |  |  |  |  |

q1. Начинает искать знак: если +, то переходит в q2, если - , то переходит в q3

q2. Если 0 - конечное состояние, если 1 – q4, если 2 – q5 + движение влево

q3. Если 0 - конечное состояние, если 1 – q6, если 2 – q7 + движение влево

q4. Прибавляет 1 к числу

q5. Прибавляет 2 к числу

q6. Отнимает 1 от числа, переходит в q8, чтобы убрать незначащий 0, если надо

q7. Отнимает 2 от числа, переходит в q8, чтобы убрать незначащий 0, если надо

q8. Переход в начало числа и далее в q9

q9. Если первый символ 0, то переходит в q10

q10. Если есть цифры кроме 0, то переходит в q11

q11. Удаляет незначащий 0