
КАФЕДРА

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ
РУКОВОДИТЕЛЬ

должность, уч. степень, звание

подпись, дата

инициалы, фамилия

Отчет о лабораторной работе №2

Изучение принципов функционирования машины Тьюринга

По дисциплине: Теория вычислительных процессов

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №

подпись, дата

инициалы, фамилия

Санкт-Петербург 2024

Цель работы:

Изучить принципы функционирования машины Тьюринга и научиться реализовывать арифметические функции с помощью команд для машины Тьюринга в программе Algo2000. Также требуется разработать программу на языке высокого уровня, которая будет имитировать работу машины Тьюринга, демонстрируя выполнение команд и контролируя возможные ошибки.

Основные сведения из теории:

Содержательно Машина Тьюринга (МТ) как абстрактный автомат, реализующий алгоритм вычисления некоторой вычислимой функции, состоит из трех компонентов:

1. Управляющее устройство (УУ), которое может находиться в одном из состояний, образующих конечное множество $Q = \{q_0, q_1, \dots, q_n, q_f\}$ — внутренний алфавит машины Тьюринга;
2. Бесконечная лента, разбитая на ячейки, в каждой из которых может быть записан один из символов конечного алфавита $A = \{a_1, a_2, \dots, a_m, \lambda\}$ — внешний алфавит машины Тьюринга;
3. Устройство обращения к ленте — считывающая и записывающая головка, которая в текущий момент времени считывает или записывает значение одной (текущей) ячейки ленты;

Постановка задачи:

Необходимо написать программу для машины Тьюринга, реализующую вычисление арифметической функции согласно выданному варианту задания. Должна быть составлена совокупность команд P . Для выполнения данного задания следует использовать приложение Algo2000.

Аргументы задаются набором "1". Пример $2*3$, будет выглядеть следующим образом $11*111$.

Работа машины Тьюринга должна начинаться со стандартной начальной конфигурации и заканчиваться стандартной конечной конфигурацией.

Формат команды применяемый в Algo2000:

adq,

где:

a — символ, который будет записан в обозреваемую ячейку,

d — направление сдвига $\{<, >\}$,

q — состояние, в которое перейдет МТ.

Во второй части лабораторной работы требуется создать программу на языке высокого уровня имитирующую работу машины Тьюринга.

Требования к программе:

- входная лента машины Тьюринга должна считываться из файла;
- программа для машины Тьюринга должна считываться из файла;
- алфавит должен считываться из файла;
- результат работы программы должен выводиться в файл;
- результат должен содержать следующие элементы:
 - состояние ленты перед выполнением каждой команды;
 - указание положения головки на ленте;
 - выполненную команду;

- пример:
 $11 * 111 =$ — состояние ленты перед выполнением команды;
 \wedge — положение головки на ленте;
 $q01 \rightarrow q1_ >$ — выполняемая команда;
 $_ 1 * 111 =$ — состояние ленты перед выполнением команды;
 $_ \wedge$ — положение головки на ленте;

- в программе должен быть реализован контроль возможных ошибок машины Тьюринга (не задан переход, отсутствует символ в алфавите и др.).

Требования к входным данным:

- Совокупность команд:

- Формат записи команд определяется согласно форме (1) (см. выше);
- Каждая команда на отдельной строке.

- Алфавит:

- Символы внешнего алфавита, перечислены в файле через пробел.

Выполнение задачи:

Вариант:

38.

x1x2-y

Алго2000:

Пример: 2*3-2

Условие задачи: x1x2-y

Внешний алфавит: 1 a * - =

A \ Q	Q0	Q1	Q2	Q3				Q7	Q8	Q9	Q10	Q11	Q12	Q13
1	→ Q1	1 → Q1	a → Q3	1 → Q3	1 ← Q4	1 ← Q5		1 → Q7	→ Q9	1 → Q9	→ Q4	1 → Q11		
a		→ Q7	a → Q2			a → Q2	→ Q7	a → Q7						
*		* → Q2	→ Q0											
-			→ Q2	→ Q3		→ Q5		→ Q7	→ Q8					
=			→ Q3	→ Q5	→ Q5			→ Q7	→ Q11	→ Q9				
Пробел		→ Q6	→ Q1	1 ← Q4		→ Q8		1 ← Q4		→ Q10		→ Q12	→ Q0	

Таблица машины Тьюринга. Сюда вписывается алгоритм. Шаг: 164 min: 24 max: 27 Изменен

Рисунок 1 – команды для машины Тьюринга для конкретного примера

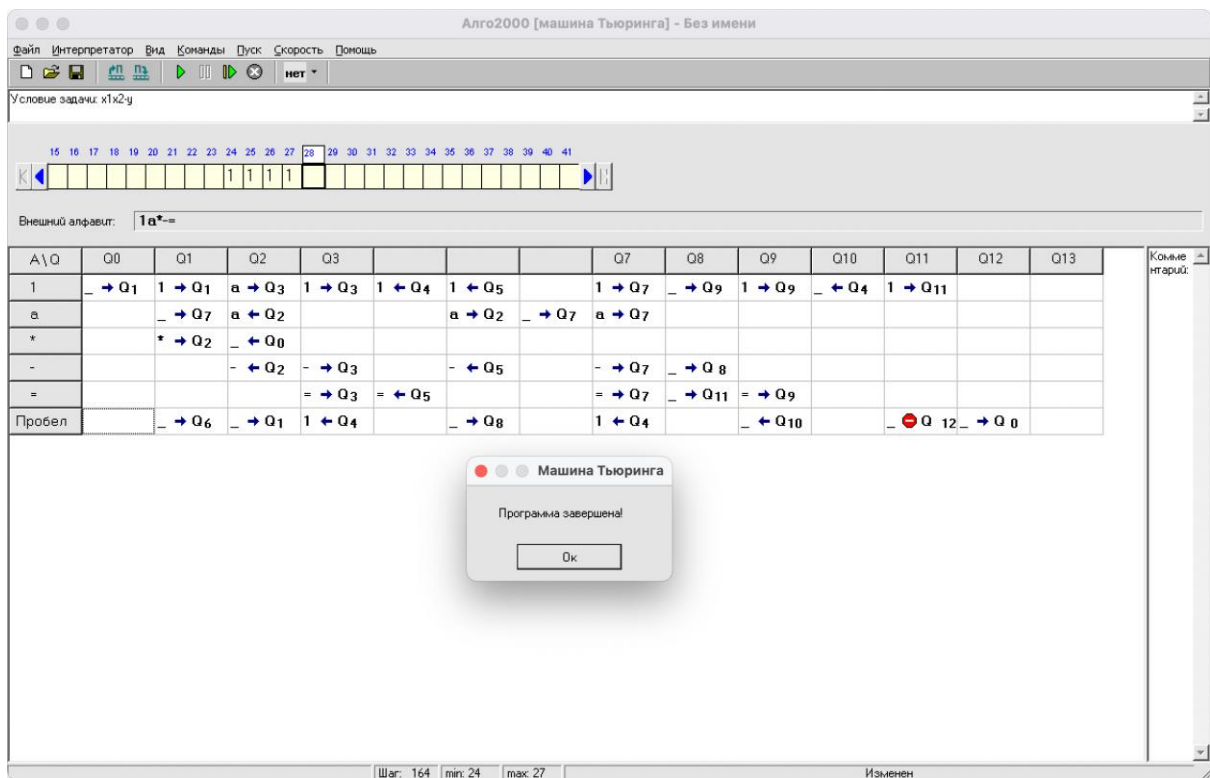


Рисунок 2 – успешное выполнение МТ

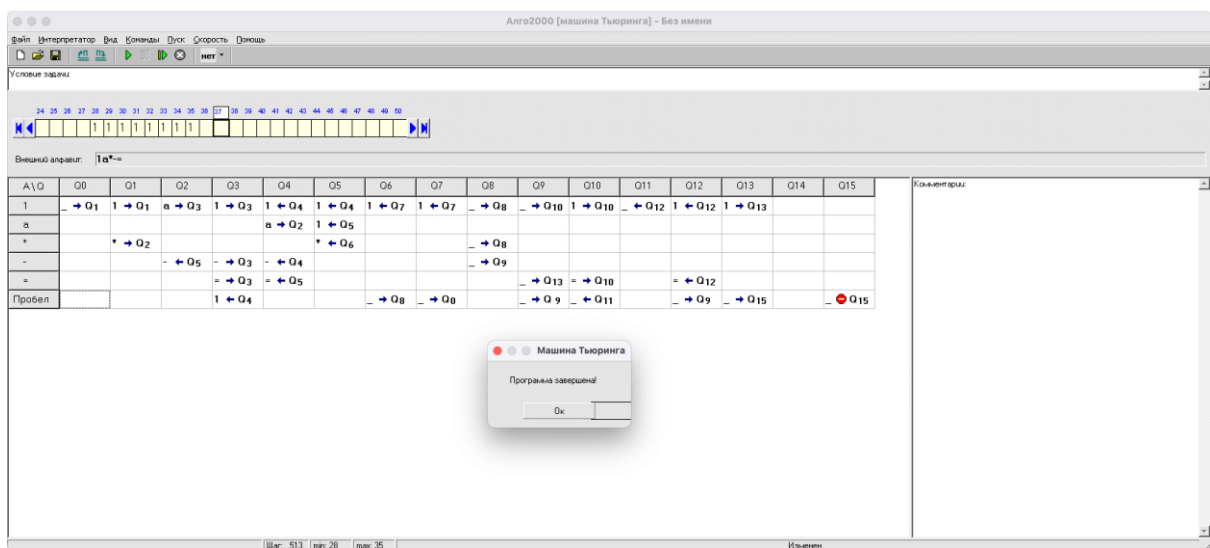
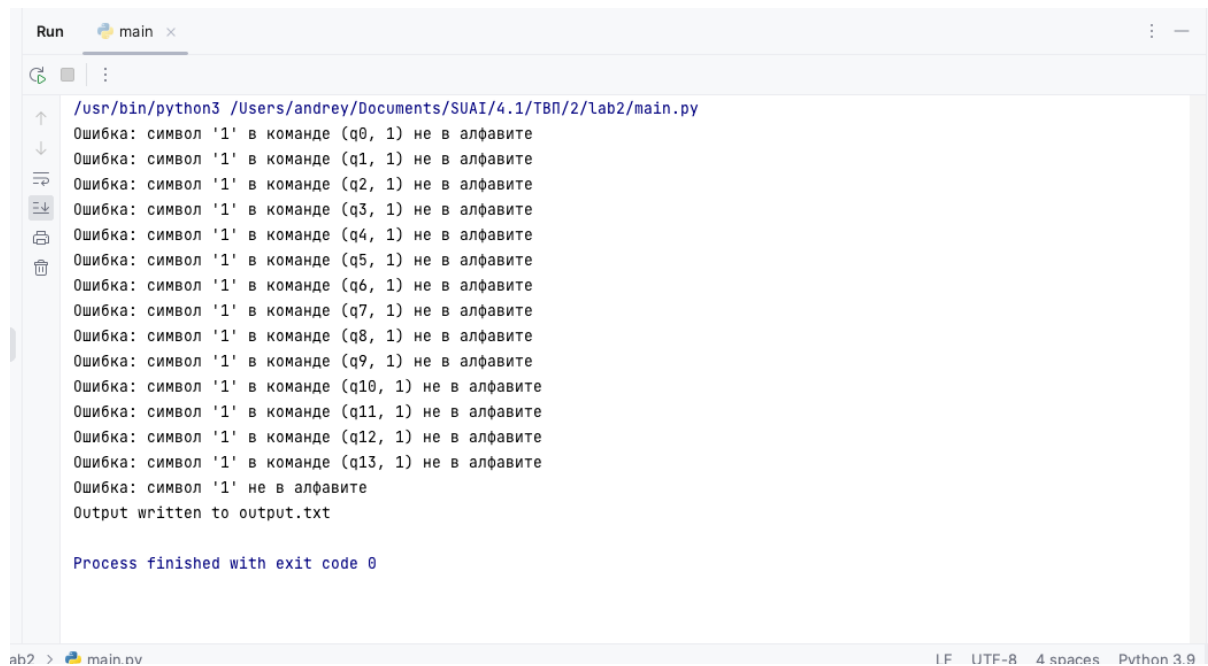


Рисунок 3 – решение на МТ для функции $x1x2-y$, пример: $1111*111-1111=$

Совокупность команд для машины Тьюринга:

q0 1 -> q1 _ R
q1 1 -> q1 1 R
q1 * -> q2 * R
q2 1 -> q3 a R
q3 1 -> q3 1 R
q3 - -> q3 - R
q3 = -> q3 = R
q3 _ -> q4 1 L
q4 1 -> q4 1 L
q4 = -> q5 = L
q5 1 -> q4 1 L
q4 - -> q4 - L
q4 a -> q2 a R
q2 - -> q5 - L
q5 a -> q5 1 L
q5 * -> q6 * L
q6 1 -> q7 1 L
q7 1 -> q7 1 L
q7 _ -> q0 _ R
q6 _ -> q8 _ R
q8 * -> q8 _ R
q8 1 -> q8 _ R
q8 - -> q9 _ R
q9 1 -> q10 _ R
q10 1 -> q10 1 R
q10 = -> q10 = R
q10 _ -> q11 _ L
q11 1 -> q12 _ L
q12 1 -> q12 1 L
q12 = -> q12 = L
q12 _ -> q9 _ R
q9 = -> q13 _ R
q9 _ -> q9 _ R
q13 1 -> q13 1 R
q13 _ -> q15 _ R
q15 _ -> qz _ N
qz _ -> qz _ N

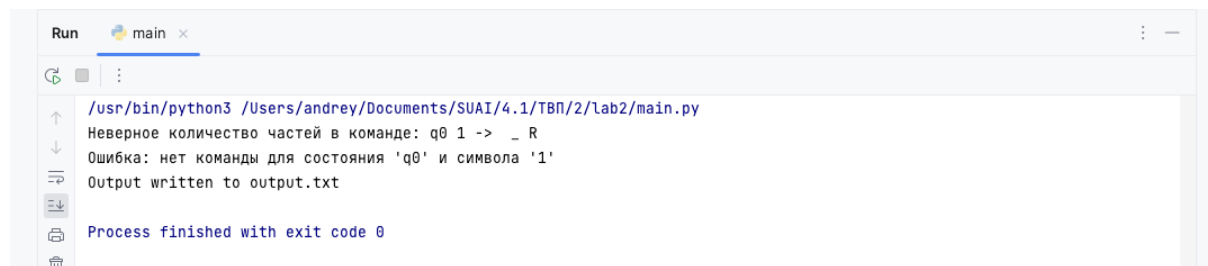
Пример результата выполнения:



```
Run main x
/usr/bin/python3 /Users/andrey/Documents/SUAI/4.1/TBП/2/lab2/main.py
Ошибка: символ '1' в команде (q0, 1) не в алфавите
Ошибка: символ '1' в команде (q1, 1) не в алфавите
Ошибка: символ '1' в команде (q2, 1) не в алфавите
Ошибка: символ '1' в команде (q3, 1) не в алфавите
Ошибка: символ '1' в команде (q4, 1) не в алфавите
Ошибка: символ '1' в команде (q5, 1) не в алфавите
Ошибка: символ '1' в команде (q6, 1) не в алфавите
Ошибка: символ '1' в команде (q7, 1) не в алфавите
Ошибка: символ '1' в команде (q8, 1) не в алфавите
Ошибка: символ '1' в команде (q9, 1) не в алфавите
Ошибка: символ '1' в команде (q10, 1) не в алфавите
Ошибка: символ '1' в команде (q11, 1) не в алфавите
Ошибка: символ '1' в команде (q12, 1) не в алфавите
Ошибка: символ '1' в команде (q13, 1) не в алфавите
Ошибка: символ '1' не в алфавите
Output written to output.txt

Process finished with exit code 0
```

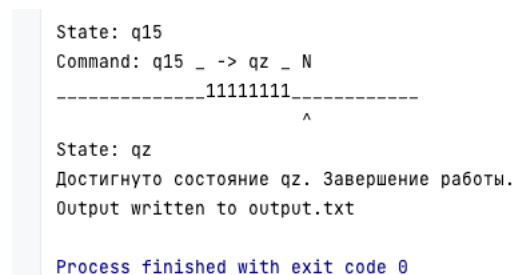
Рисунок 5 – обработка ошибки, отсутствие символа в алфавите



```
Run main x
/usr/bin/python3 /Users/andrey/Documents/SUAI/4.1/TBП/2/lab2/main.py
Неверное количество частей в команде: q0 1 -> _ R
Ошибка: нет команды для состояния 'q0' и символа '1'
Output written to output.txt

Process finished with exit code 0
```

Рисунок 6 – обработка ошибки, Отсутствие части команды



```
State: q15
Command: q15 _ -> qz _ N
-----11111111-----
                        ^

State: qz
Достигнуто состояние qz. Завершение работы.
Output written to output.txt

Process finished with exit code 0
```

Рисунок 7 – корректное завершение и ответ для ленты вида: 1111*111-1111=

Вывод:

В ходе выполнения лабораторной работы была успешно изучена работа машины Тьюринга и принципы реализации арифметических функций на основе команд для данной машины. Программа на языке высокого уровня, имитирующая работу машины Тьюринга, разработана и проверена, корректно выполняя заданные команды с контролем ошибок.

Листинг программы:

```
class TuringMachine:
    def __init__(self, tape, alphabet, program):
        self.alphabet = alphabet.strip().split() # Алфавит должен инициализироваться
первым
        self.tape = list(tape.strip()) + ['_'] * 20 # Лента с зазором для вычислений
        self.head_position = 0 # Положение головки
        self.state = 'q0' # Начальное состояние
        self.program = self.parse_program(program) # Программа инициализируется после
алфавита

    def parse_program(self, program_text):
        program = {}
        for line in program_text.strip().splitlines():
            line = line.split('#')[0].strip() # Удаляем комментарии и лишние пробелы
            if not line:
                continue # Игнорируем пустые строки
            parts = line.split()
            if len(parts) == 6:
                state, symbol, arrow, next_state, new_symbol, move = parts
                if arrow == '->': # Проверка правильного формата
                    program[(state, symbol)] = (new_symbol, move, next_state)
                else:
                    self.write_output(f'Ошибка в формате команды: {line}')
            else:
                self.write_output(f'Неверное количество частей в команде: {line}')

        # Проверка на корректность символов в программе
        for (state, symbol) in program.keys():
            if symbol not in self.alphabet:
                self.write_output(f'Ошибка: символ '{symbol}' в команде ({state}, {symbol}) не
в алфавите")

        return program

    def step(self):
        if self.head_position < 0 or self.head_position >= len(self.tape):
            self.write_output("Ошибка: головка вышла за пределы ленты")
            return False # Остановить выполнение

        state = self.state
        symbol = self.tape[self.head_position]

        # Проверка на наличие символа в алфавите
        if symbol not in self.alphabet:
            self.write_output(f'Ошибка: символ '{symbol}' не в алфавите")
            return False # Остановить выполнение

        # Проверка на состояние qz
        if state == 'qz':
```



```

self.write_output(f'Достигнуто состояние qz. Завершение работы.')
return False # Завершение выполнения

# Находим подходящую команду для текущего состояния и символа
if (state, symbol) in self.program:
    new_symbol, move, new_state = self.program[(state, symbol)]

    # Записываем текущее состояние ленты перед выполнением команды
    self.write_output(self.get_tape_state())

    # Записываем команду в файл
    self.write_output(f'Command: {state} {symbol} -> {new_state} {new_symbol}
{move}')

    # Изменяем символ на ленте
    self.tape[self.head_position] = new_symbol

    # Двигаем головку вправо или влево
    if move == 'R':
        self.head_position += 1
    elif move == 'L':
        self.head_position -= 1

    # Проверка на выход головки за пределы ленты
    if self.head_position < 0 or self.head_position >= len(self.tape):
        self.write_output("Ошибка: головка вышла за пределы ленты после
перемещения")
        return False # Остановить выполнение

    # Меняем состояние машины
    self.state = new_state

    # Записываем текущее состояние ленты после выполнения команды
    self.write_output(self.get_tape_state())

    return True # Возвращаем True для продолжения работы
else:
    self.write_output(f'Ошибка: нет команды для состояния '{state}' и символа
'{symbol}')
```

```

    return False # Остановить выполнение

def get_tape_state(self):
    tape_str = ".join(self.tape)
    head_str = ' ' * self.head_position + '^'
    return f'{tape_str}\n{head_str}\nState: {self.state}'

def write_output(self, text):
    # Печатаем в консоль
    print(text)
    # Записываем в файл
    with open('output.txt', 'a') as f:
```

```

        f.write(text + '\n')

def run(self, output_file):
    with open(output_file, 'w') as f:
        pass # Очистка файла перед записью
    while self.step():
        pass
    print(f'Output written to {output_file}')

# Чтение данных из файлов
with open('tape.txt') as tape_file:
    tape = tape_file.read()

with open('alphabet.txt') as alphabet_file:
    alphabet = alphabet_file.read()

with open('program.txt') as program_file:
    program = program_file.read()

# Создание машины Тьюринга и запуск программы
tm = TuringMachine(tape, alphabet, program)
tm.run('output.txt')

```