|  | **МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**  ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  «МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ  (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)» (МАИ) |
| --- | --- |

**Отчет**

о выполнении лабораторной работы

по дисциплине «Теория автоматов»

на тему **«Исследование начальных языков описания цифровых автоматов. Часть I»**

Кафедра С-15

Выполнил студент

группы МСО-302Б-21

Дорофеев М. К.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил преподаватель

Пахомов М.М.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**СОДЕРЖАНИЕ**

[**1. Цель работы 3**](#_2it5wvril8dg)

[**2. Ход работы 4**](#_r896afuz1mb9)

[**3. Завершение 6**](#_hrj03kk8izn)

[**4. ГСА и ЛСА 7**](#_u44vhd80cpq7)

[**4.1 Таблица переходов 8**](#_x80kukqdyzu4)

[**5. Алгоритм моделирования ГСА и структуры данных 9**](#_bnchmbkw1m5c)

[**5.1 Режимы работы программы. 9**](#_mms1loqb7qxr)

[**5.2 Программный код программы моделирования конечного автомата. 9**](#_qqim68ukzaw)

[**6. Заключение 17**](#_xspfk5vdytzi)

# 

# 

# 

# 

# 

# 

# Цель работы

Цель работы – получение навыков описания и исследования цифровых автоматов, заданных с помощью начальных языков. Создание модели конечного автомата, который задан графической схемой, а также создание нескольких режимов работы для этого автомата.

# 

# 

# 

# 

# 

# 

# 

# 

# Ход работы

Для запуска программы необходимо нажать на ярлык “FSM” на рабочем столе (рисунок 1).



Рисунок 1 - Ярлык программы.

Откроется окно терминала Windows с сообщением о выборе режима работы (рисунок 2).

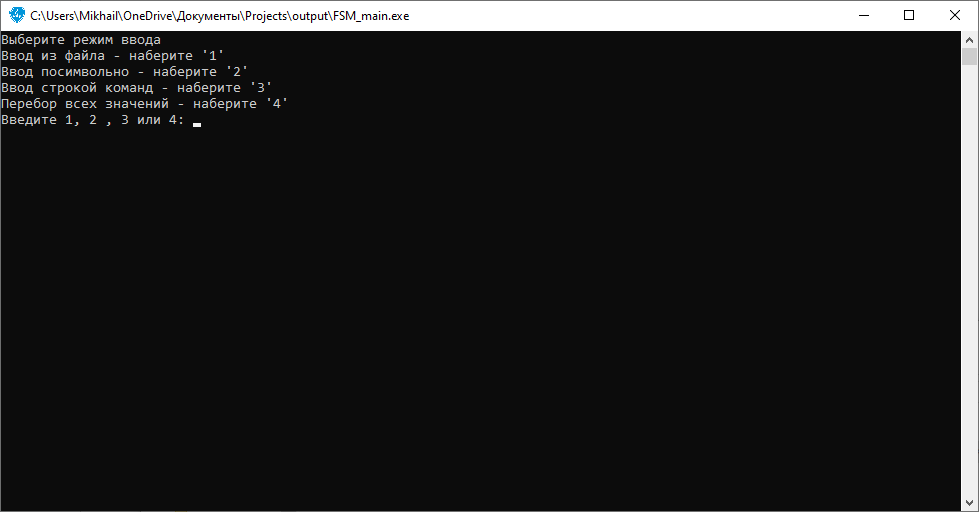


Рисунок 2 - Запуск программы и открытие окна терминала

При выборе “Ввод из файла” убедитесь, что файл предварительно загружен в директории “C:\text.txt”.

Выполнение программы при сценарии “Ввод посимвольно” начинается с ввода пользователем символа ‘2’. Далее выводятся шаги отображающие текущее состояние автомата и/или пройденные состояния (рисунок 3). Пользователю предлагается ввести одно из двух значений.

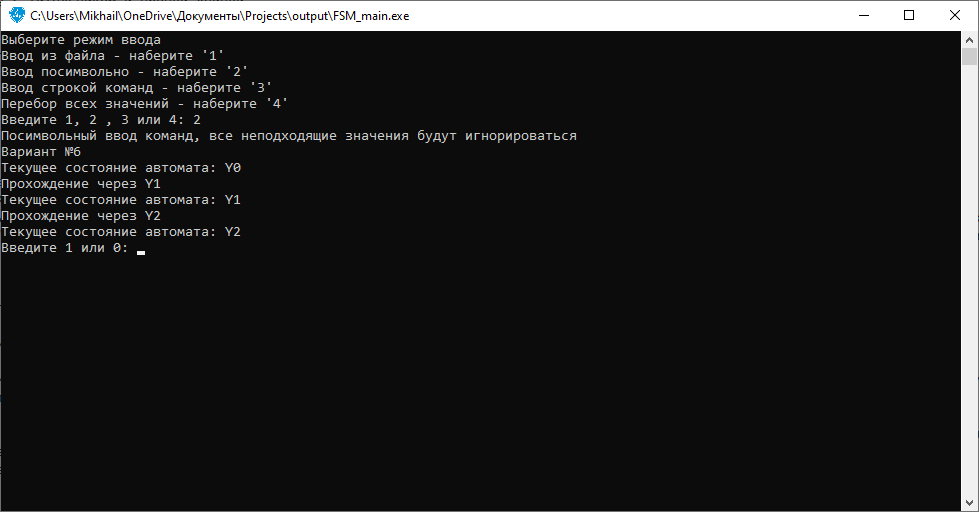


Рисунок 3 - Посимвольный режим работы программы

По окончании работы алгоритма (достижении конечного состояния). Будет выведена информация о достижении конечного состояния и все состояния автомата за период работы программы (Рисунок 4).

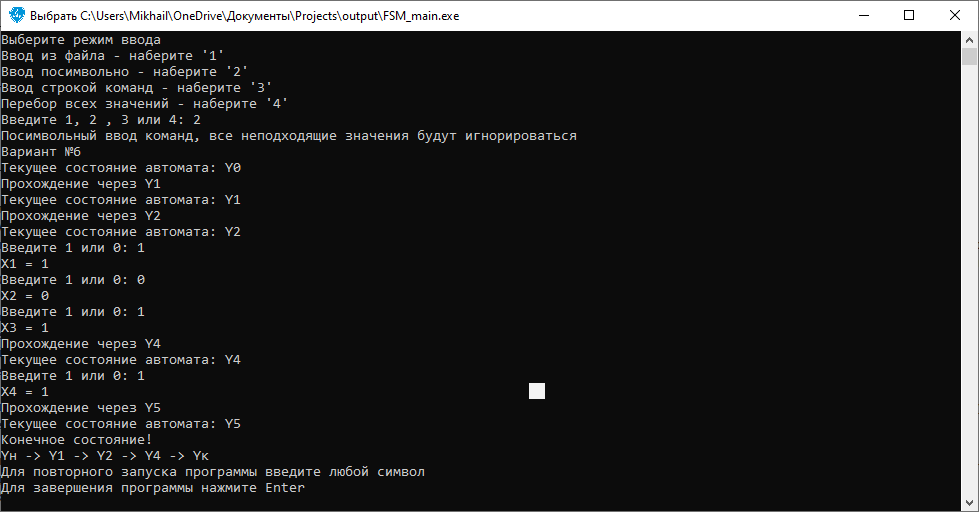


Рисунок 4 - Достижение конечного (Yк) состояния.

# Завершение

Работу программы "FSM\_main" необходимо завершать нажатием кнопки «Закрыть» в правом верхнем углу рабочего окна терминала или нажатием Enter по завершению программы.

Дополнительно можно продолжить работу программы введя любой символ с клавиатуры и нажать Enter (Рисунок 5).

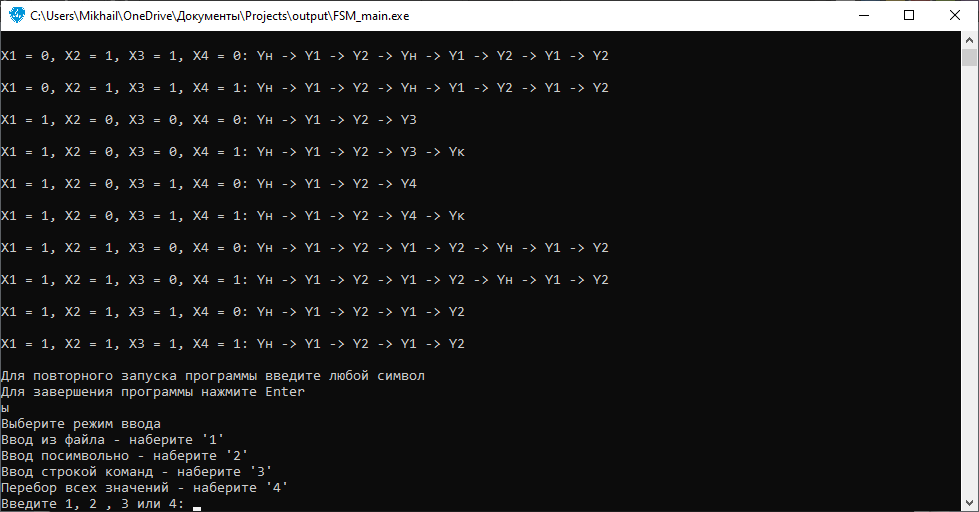


Рисунок 5 - Продолжение работы программы

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# ГСА и ЛСА

Графическая схема алгоритма изображена на рисунке 6.

Логическая схема алгоритма изображена на рисунке 7.



Рисунок 6 - ГСА



Рисунок 7 - ЛСА

## 

## 

## 4.1 Таблица переходов

Таблица является таблицей истинности, определяющей логику следующего состояния. Она определяет следующее состояние S как функцию входов и текущего состояния.

Таблица 1

| Текущее состояние S | Входное значение | Следующее состояние |
| --- | --- | --- |
| Yн | X | Y1 |
| Y1 | X | Y2 |
| Y2 | 0XX | Yн |
| Y2 | 101 | Y4 |
| Y2 | 100 | Y3 |
| Y2 | 11X | Y1 |
| Y3 / Y4 | 1 | Yк |
| Y3 / Y4 | 0 | Y3 |
| Yк | X | Yк |

# 

# 

# 

# 

# 

# 

# Алгоритм моделирования ГСА и структуры данных

Алгоритм моделирования ГСА представлен в виде программы,

написанной на языке Python, которая, в свою очередь, представлена в виде блок-схемы и текста программного кода.

Программа предусматривает набор функций, которые обеспечивают

выбор режима работы, хранят состояния вершин автомата и выполняют определённый алгоритм обхода этих вершин в зависимости от значений входных данных.

## 5.1 Режимы работы программы.

Программа обеспечивает работу автомата в четырёх режимах:

1. Ввод значений из файла и вывод результата каждого шага моделирования;
2. Последовательный ввод значений логических условий Х и вывод результата каждого шага моделирования;
3. Ввод значений всех логических условий Х и вывод результата моделирования;
4. Полный перебор всех значений логических условий Х и вывод результата моделирования.

Каждому режиму соответствует определённая функция, обеспечивающая выполнение алгоритма выбранного режима.

## 5.2 Программный код программы моделирования конечного автомата.

def security\_input():

x = input("Введите 1 или 0: ")

if x in (1, 0, '1', '0'):

return x

else:

print("Введено неверное значение...")

security\_input()

def len\_content\_and\_index\_checker(content = '', index=0):

if index >= len(content):

return True

else:

return False

def op\_line\_debugger(string\_line=''):

new\_line = ''

for i in string\_line:

if i in ('0', '1'):

new\_line += i

return new\_line

def fsm\_string(state = 0): #fsm == Finite State Machine

def GDG(x): #Graphical Display of the Graph state history

for i in range(len(x)):

if (i != len(x)-1):

print(f"{x[i]} ->", end=" ")

else:

print(f"{x[i]}")

content = ""

index = 0

while True:

print("Введите строку операндов")

content = input()

if (len(content) > 0):

break

else:

print("Попробуйте ввести ещё раз...")

continue

content = op\_line\_debugger(content)

print("Вариант №6")

Y0, Y1, Y2, Y3, Y4, Y5 = (0, 1, 2, 3, 4, 5)

list\_of\_states = list()

while True:

print(f"Текущее состояние автомата: Y{state}")

if (index >= len(content)):

GDG(list\_of\_states)

print("Конец работы автомата, операции кончились...")

break

if (state == Y0):

list\_of\_states.append('Yн')

print("Прохождение через Y1")

state = Y1

elif (state == Y1):

list\_of\_states.append('Y1')

print("Прохождение через Y2")

state = Y2

elif (state == Y2):

list\_of\_states.append('Y2')

if ( content[index] == '1'):

print("X1 = 1")

index += 1

if (len\_content\_and\_index\_checker(content, index)):

continue

if (content[index] == '1'):

print("X2 = 1")

index += 1

state = Y1

else:

print("X2 = 0")

if (len\_content\_and\_index\_checker(content, index)):

continue

if (content[index] == '1'):

print("X3 = 1")

index += 1

print("Прохождение через Y4")

state = Y4

else:

print("X3 = 0")

index += 1

print("Прохождение через Y3")

state = Y3

else:

print("X1 = 0")

index += 1

state = Y0

elif (state == Y3):

list\_of\_states.append('Y3')

if (len\_content\_and\_index\_checker(content, index)):

continue

if (content[index] == '1'):

print("X4 = 1")

index += 1

print("Прохождение через Y5")

state = Y5

else:

print("X4 = 0")

index += 1

state = Y3

elif (state == Y4):

list\_of\_states.append('Y4')

if (len\_content\_and\_index\_checker(content, index)):

continue

if (content[index] == '1'):

print("X4 = 1")

index += 1

print("Прохождение через Y5")

state = Y5

else:

print("X4 = 0")

index += 1

print("Прохождение через Y3")

state = Y3

else: #state == Y5

list\_of\_states.append('Yк')

print("Конечное состояние!")

GDG(list\_of\_states)

break

def fsm\_file(state = 0): #fsm == Finite State Machine

def GDG(x): #Graphical Display of the Graph state history

for i in range(len(x)):

if (i != len(x)-1):

print(f"{x[i]} ->", end=" ")

else:

print(f"{x[i]}")

content = ""

index = 0

while True:

try:

file = open(r'C:\text.txt', 'r')

content = "".join([i.strip() for i in file.read()])

content = op\_line\_debugger(content)

file.close()

break

except FileNotFoundError:

print("Не нашёлся файл с названием text.txt")

break

except:

print("Ошибка во время открытия файла или чего нибудь ещё...")

break

finally:

file.close()

break

print("Вариант №6")

Y0, Y1, Y2, Y3, Y4, Y5 = (0, 1, 2, 3, 4, 5)

list\_of\_states = list()

while True:

print(f"Текущее состояние автомата: Y{state}")

if (index >= len(content)):

GDG(list\_of\_states)

print("Конец работы автомата, операции кончились...")

break

if (state == Y0):

list\_of\_states.append('Yн')

print("Прохождение через Y1")

state = Y1

elif (state == Y1):

list\_of\_states.append('Y1')

print("Прохождение через Y2")

state = Y2

elif (state == Y2):

list\_of\_states.append('Y2')

if ( content[index] == '1'):

print("X1 = 1")

index += 1

if (len\_content\_and\_index\_checker(content, index)):

continue

if (content[index] == '1'):

print("X2 = 1")

index += 1

state = Y1

else:

print("X2 = 0")

if (len\_content\_and\_index\_checker(content, index)):

continue

if (content[index] == '1'):

print("X3 = 1")

index += 1

print("Прохождение через Y4")

state = Y4

else:

print("X3 = 0")

index += 1

print("Прохождение через Y3")

state = Y3

else:

print("X1 = 0")

index += 1

state = Y0

elif (state == Y3):

list\_of\_states.append('Y3')

if (len\_content\_and\_index\_checker(content, index)):

continue

if (content[index] == '1'):

print("X4 = 1")

index += 1

print("Прохождение через Y5")

state = Y5

else:

print("X4 = 0")

index += 1

state = Y3

elif (state == Y4):

list\_of\_states.append('Y4')

print("Прохождение через X4")

if (len\_content\_and\_index\_checker(content, index)):

continue

if (content[index] == '1'):

index += 1

print("Прохождение через Y5")

state = Y5

else:

index += 1

print("Прохождение через Y3")

state = Y3

else: #state == Y5\

list\_of\_states.append('Yк')

print("Конечное состояние!")

GDG(list\_of\_states)

break

def main\_fsm(state = 0): #fsm == Finite State Machine

def GDG(x): #Graphical Display of the Graph state history

for i in range(len(x)):

if (i != len(x)-1):

print(f"{x[i]} ->", end=" ")

else:

print(f"{x[i]}")

Y0, Y1, Y2, Y3, Y4, Y5 = (0, 1, 2, 3, 4, 5)

list\_of\_states = list()

print("Вариант №6")

while True:

print(f"Текущее состояние автомата: Y{state}")

if (state == Y0):

list\_of\_states.append('Yн')

print("Прохождение через Y1")

state = Y1

elif (state == Y1):

list\_of\_states.append('Y1')

print("Прохождение через Y2")

state = Y2

elif (state == Y2):

list\_of\_states.append('Y2')

if ( security\_input() == '1'):

print("X1 = 1")

if (security\_input() == '1'):

print("X2 = 1")

state = Y1

else:

print("X2 = 0")

if (security\_input() == '1'):

print("X3 = 1")

print("Прохождение через Y4")

state = Y4

else:

print("X3 = 0")

print("Прохождение через Y3")

state = Y3

else:

print("X1 = 0")

print(security\_input())

state = Y0

elif (state == Y3):

list\_of\_states.append('Y3')

if (security\_input() == '1'):

print("X4 = 1")

print("Прохождение через Y5")

state = Y5

else:

print("X4 = 0")

state = Y3

elif (state == Y4):

list\_of\_states.append('Y4')

if (security\_input() == '1'):

print("X4 = 1")

print("Прохождение через Y5")

state = Y5

else:

print("X4 = 0")

print("Прохождение через Y3")

state = Y3

else: #state == Y5

list\_of\_states.append('Yк')

print("Конечное состояние!")

GDG(list\_of\_states)

break

def fsm\_sort\_through(state = 0): #fsm == Finite State Machine

def GDG(x): #Graphical Display of the Graph state history

for i in range(len(x)):

if (i != len(x)-1):

print(f"{x[i]} ->", end=" ")

else:

print(f"{x[i]}", '\n')

list\_of\_options = ['0000', '0001', '0010', '0011',

'0100', '0101', '0110', '0111',

'1000', '1001', '1010', '1011',

'1100', '1101', '1110', '1111']

Y0, Y1, Y2, Y3, Y4, Y5 = (0, 1, 2, 3, 4, 5)

print("Вариант №6")

for index\_of\_option in range(0, 16):

index = 0; list\_of\_states = list(); content = list\_of\_options[index\_of\_option]

state = 0

print(f"X1 = {content[0]}, X2 = {content[1]}, X3 = {content[2]}, X4 = {content[3]}", end=': ')

while True:

if (index >= len(content) and state != Y5):

GDG(list\_of\_states)

break

if (state == Y0):

list\_of\_states.append('Yн')

state = Y1

elif (state == Y1):

list\_of\_states.append('Y1')

state = Y2

elif (state == Y2):

list\_of\_states.append('Y2')

if (len\_content\_and\_index\_checker(content, index)):

continue

if ( content[index] == '1'): # -> X1

index += 1

if (len\_content\_and\_index\_checker(content, index)):

continue

if (content[index] == '1'):

index += 1

state = Y1

else:

index += 1 # -> X2

if (len\_content\_and\_index\_checker(content, index)):

continue

if (content[index] == '1'): # -> X3

index += 1

state = Y4

else:

index += 1

state = Y3

else:

index += 1

state = Y0

elif (state == Y3):

list\_of\_states.append('Y3')

if (len\_content\_and\_index\_checker(content, index)):

continue

if (content[index] == '1'):

index += 1

state = Y5

else:

index += 1

state = Y3

elif (state == Y4):

list\_of\_states.append('Y4')

if (len\_content\_and\_index\_checker(content, index)):

continue

if (content[index] == '1'):

index += 1

state = Y5

else:

index += 1

state = Y3

else: #state == Y5

list\_of\_states.append('Yк')

GDG(list\_of\_states)

break

def mode\_of\_input():

def security\_input\_2():

x = input("Введите 1, 2 , 3 или 4: ")

if x in ('1', '2', '3', '4'):

if (x == '1'):

print("Убедитесь, что название файла text.txt и что он размещён в пути C:\\text.txt")

elif(x == '2'):

print("Посимвольный ввод команд, все неподходящие значения будут игнорироваться")

elif(x == '3'):

print("Выбран ввод команд одной строкой, все неподходящие значения будут игнорироваться")

elif(x == '4'):

print("Перебор всех значений")

return int(x)

else:

print("Введено неверное значение...")

security\_input\_2()

print("Выберите режим ввода")

print("Ввод из файла - наберите \'1\'")

print("Ввод посимвольно - наберите \'2\'")

print("Ввод строкой команд - наберите \'3\'")

print("Перебор всех значений - наберите \'4\'")

dict\_of\_modes = {1: 1, 2: 2, 3: 3, 4: 4}

return dict\_of\_modes[security\_input\_2()]

def start\_fsm():

input\_mode = mode\_of\_input()

if (input\_mode == 1):

fsm\_file()

elif (input\_mode == 2):

main\_fsm()

elif (input\_mode == 4):

fsm\_sort\_through()

else:

fsm\_string()

while True:

start\_fsm()

print("Для повторного запуска программы введите любой символ")

print("Для завершения программы нажмите Enter")

x = input()

if x == '':

break

else:

continue

# Заключение

В данной лабораторной работе была разработана программа, реализующая конечный автомат, заданный ГСА, а также четыре режима работы для этого автомата. Результаты теста на корректность работы программы - положительные.