|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |
| Институт Информационных Технологий |
| Кафедра Вычислительной техники |

|  |  |
| --- | --- |
| **ОТЧЕТ О ВЫПОЛНЕНИИ ПРАКТИЧЕСКОЙ РАБОТЫ**  **№4**  «Преобразования недетерминированного конечного автомата (НКА) в детерминированный (ДКА) **на Python»** | |
| **по дисциплине** |  |
| **«**Теория формальных языков**»** | |
| Выполнил студент группы ИКБО-15-22 | *Оганнисян Г.А.* |
| Принял старший преподаватель | *Боронников А.С.* |

|  |  |
| --- | --- |
| Практическая работа выполнена | « » 2023 г. |
| «Зачтено» | « » 2023 г. |

Москва 2021



Рисунок 1 – Программа просит ввести данные

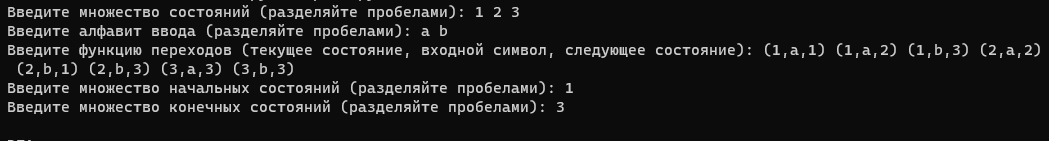


Рисунок 2 - Ввод всех нужных для расчета данных

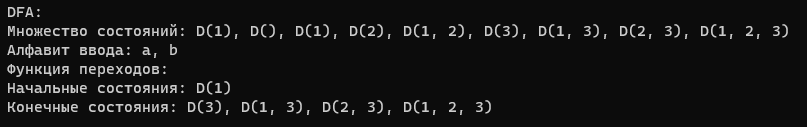


Рисунок 3 – Вывод программы

Эта программа решает задачу преобразования конечного автомата с недетерминированными переходами (NFA) в детерминированный автомат (DFA). Сначала она собирает информацию о состояниях, алфавите, переходах, начальных и конечных состояниях NFA из пользовательского ввода. Затем она использует эту информацию для построения эквивалентного DFA.

Программа использует алгоритм под названием "подмножества состояний". Она начинает с начальных состояний NFA и итеративно строит новые состояния DFA, рассматривая все возможные переходы по символам

алфавита. Это позволяет ей пошагово создавать множество состояний и определять, являются ли они конечными.

В конце программа выводит информацию о построенном DFA, включая множество состояний, алфавит, функцию переходов, начальные и конечные состояния.

Листинг кода

def powerset(s):

    if len(s) == 0:

        return [[]]

    subsets = powerset(s[:-1])

    return subsets + [item + [s[-1]] for item in subsets]

def convert\_nfa\_to\_dfa(states, alphabet, transitions, initial\_states, final\_states):

    dfa\_states = []

    dfa\_transitions = {}

    dfa\_initial\_state = ''

    dfa\_final\_states = []

*# Create the initial state for the DFA*

    dfa\_initial\_state = 'D(' + ', '.join(initial\_states) + ')'

    dfa\_states.append(dfa\_initial\_state)

    powerset\_states = powerset(states)

    for subset in powerset\_states:

        subset\_name = 'D(' + ', '.join(subset) + ')'

        dfa\_states.append(subset\_name)

        for symbol in alphabet:

            next\_state = set()

            for state in subset:

                for transition in transitions:

                    if transition[0] == state and transition[1] == symbol:

                        next\_state.add(transition[2])

            if next\_state:

                next\_state\_name = 'D(' + ', '.join(sorted(next\_state)) + ')'

                dfa\_transitions[(subset\_name, symbol)] = next\_state\_name

                if next\_state\_name not in dfa\_states:

                    dfa\_states.append(next\_state\_name)

    for subset in powerset\_states:

        for final\_state in final\_states:

            if final\_state in subset:

                dfa\_final\_states.append('D(' + ', '.join(subset) + ')')

                break

    return dfa\_states, alphabet, dfa\_transitions, dfa\_initial\_state, dfa\_final\_states

*# Ввод данных через консоль*

states = input("Введите множество состояний (разделяйте пробелами): ").split()

alphabet = input("Введите алфавит ввода (разделяйте пробелами): ").split()

transitions\_input = input("Введите функцию переходов (текущее состояние, входной символ, следующее состояние): ").split()

transitions = [tuple(transitions\_input[i:i+3]) for i in range(0, len(transitions\_input), 3)]

initial\_states = input("Введите множество начальных состояний (разделяйте пробелами): ").split()

final\_states = input("Введите множество конечных состояний (разделяйте пробелами): ").split()

*# Вызов функции для преобразования*

dfa\_states, dfa\_alphabet, dfa\_transitions, dfa\_initial\_state, dfa\_final\_states = convert\_nfa\_to\_dfa(

    states, alphabet, transitions, initial\_states, final\_states)

*# Вывод результата*

print("\nDFA:")

print("Множество состояний:", ', '.join(dfa\_states))

print("Алфавит ввода:", ', '.join(dfa\_alphabet))

print("Функция переходов:")

for key, value in dfa\_transitions.items():

    print(key[0], "=", key[1], "=", value)

print("Начальные состояния:", dfa\_initial\_state)

print("Конечные состояния:", ', '.join(dfa\_final\_states))