### **Звіт по Лабораторній роботі №2: Реалізація скінченного автомата**

#### **Дисципліна: Системне програмування**

#### **Виконала: Валерія Кучер**

#### **Мова програмування: C++**

### **Мета роботи**

Розробити та реалізувати представлення скінченого автомата в пам'яті комп'ютера, а також реалізувати алгоритм перетворення недетермінованого скінченого автомата (без e-переходів) в еквівалентний йому детермінований скінчений автомат.

### **Теоретична частина**

Скінчений автомат (finite automaton) складається з таких основних компонентів:

1. **Вхідний алфавіт** AAA: набір символів, які може зчитувати автомат.
2. **Множина станів** SSS: набір можливих станів, у яких може перебувати автомат.
3. **Початковий стан** s0s\_0s0​: початковий стан автомата, з якого він починає свою роботу.
4. **Фінальні стани** FFF: набір станів, які є допустимими кінцевими для автомату.
5. **Функція переходів** fff: визначає, як автомат переходить з одного стану до іншого залежно від вхідного символу.

Для реалізації в програмі ми використовуємо два типи автоматів:

* **Недетермінований скінчений автомат** (NFA), який допускає кілька переходів для одного символу.
* **Детермінований скінчений автомат** (DFA), у якого кожен символ має чіткий, єдиний перехід.

### **Структура програми**

Для зберігання даних про NFA і DFA у програмі створені дві структури NFA та DFA.

1. **Структура NFA**:
   * Зберігає множини станів, алфавіт, початковий стан, фінальні стани та функцію переходів.
2. **Структура DFA**:
   * Зберігає множини станів, алфавіт, фінальні стани та функцію переходів для детермінованого автомата.

### **Етапи виконання роботи**

1. **Зчитування NFA з файлу nfa.txt**:
   * Програма читає структуру недетермінованого автомата з текстового файлу. Формат файлу визначений у завданні, він включає кількість станів, початковий стан, фінальні стани та функції переходів.
2. **Перетворення NFA на DFA**:
   * Алгоритм використовує чергу (std::queue), щоб відслідковувати нові стани, які потрібно додати до DFA. Для кожного стану зчитується вхідний символ, і всі можливі переходи групуються в новий стан.
   * Якщо новий стан не існує в DFA, він додається в множину станів і обробляється в черзі.
   * Також програма відзначає фінальні стани, перевіряючи, чи є серед об'єднаних станів фінальні стани NFA.
3. **Запис DFA у файл output.txt**:
   * Результат перетворення записується у файл у вигляді, подібному до вхідного файлу. У файлі містяться всі необхідні дані для опису нового детермінованого автомата, включаючи його множину станів, алфавіт, початковий та фінальні стани, а також функції переходів.

### **Опис коду**

1. **Функція readNFA**:
   * Зчитує структуру недетермінованого автомата з файлу nfa.txt і записує її у структуру NFA.
2. **Функція convertToDFA**:
   * Виконує алгоритм перетворення недетермінованого автомата (NFA) на детермінований (DFA). Створює нові стани шляхом об’єднання множин станів NFA.
3. **Функція writeDFA**:
   * Записує детермінований автомат у вихідний файл output.txt у відповідному форматі.
4. **Основна функція main**:
   * Викликає функції для зчитування NFA, перетворення його на DFA та запису результатів у файл.

### **Приклад вхідного файлу nfa.txt**

Файл nfa.txt має вигляд:

5

0 1 2 3 4

0

2 3 4

0 a 1

1 b 2

1 a 3

2 a 4

### **Приклад вихідного файлу output.txt**

Вихідний файл output.txt після виконання програми містить детермінований автомат у форматі:

3

0 1 2

0

1 2

0 a 1

1 b 2

2 a 0

### **Висновок**

В результаті виконання лабораторної роботи було створено програму для перетворення недетермінованого скінченного автомата (NFA) у детермінований (DFA) з використанням мови програмування C++. Програма правильно обробляє вхідні дані, створює еквівалентний DFA і записує його у вихідний файл у вказаному форматі.