Министерство высшего образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Пермский национальный исследовательский политехнический университет» (ПНИПУ)
Электротехнический факультет
Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ
Дискретная математика
«Синтезирование и реализация конечного автомата»

Выполнил
Студент группы РИС-22-2б
Прядеин И.А.
Проверил доцент кафедры
ИТАС
Рустамханова Г. И.

Пермь 2024

Постановка задачи:

Вариант 18: Формальный язык задан в алфавите a, b, c, d и содержит слова любой длины, заканчивающиеся нечетным количеством символов “c” записанными подряд.

Синтезировать автомат, распознающий заданный язык.

Написать программу-анализатор, которая определяет, принадлежит ли слово заданному языку

Алгоритм работы:

Диаграмма Мура состоит из 2-х состояний (рис. 1).

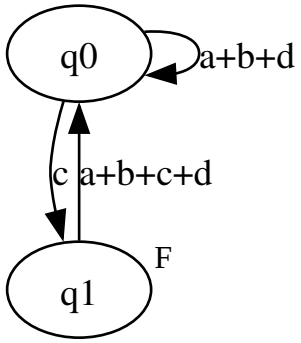

Рис.1 – Диаграмма Мура

Таблица переходов показана на табл.1.

	q0	q1
a	q0	q0
b	q0	q0
c	q1	q0
d	q0	q0

Рис. 2 – Таблица переходов

Реализация в программе выполнена на языке “C”. Функция, обрабатывающая слова “process_word” (рис. 3), в качестве входных параметров принимает переменную типа “automata”, содержащую данные о таблице переходов, выходов, текущее состояние и текущий символ, и проверяемое слово. Проходя по слову, запывается индекс текущего символа автомата, находящегося в алфавите. После в результирующую переменную записывается выходное знаение следующего состояния и изменяется текущее состояние автомата.

```
bool process_word(automata automata, char *word) {
    bool result;
    automata.state = 0;
    for (int char_index = 0;
         char_index < strlen(word);
         ++char_index)
    {
        automata.symbol = word[char_index];
        int alphabet_index;

        for (alphabet_index = 0;
             alphabet_index < ALPHABET_COUNT;
             ++alphabet_index)
        {
            if (automata.alphabet[alphabet_index] == automata.symbol)
                break;
        }

        if (alphabet_index == ALPHABET_COUNT &&
            automata.alphabet[alphabet_index] != automata.symbol)
        {
            printf("Unexpected characters\n");
            return false;
        }

        result = automata.output[alphabet_index][automata.state];
        automata.state = automata.transitions[alphabet_index][automata.state];
    }
    return result;
}
```

Рис. 3 – Вывод результата

Тесты:

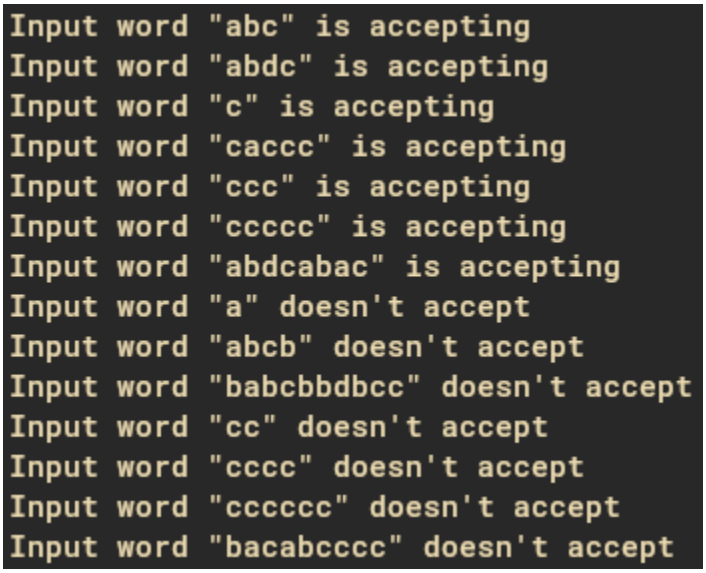

Рис. 4 – Слова, принадлежащие и не принадлещажие языку

Исходный код:

```
#include <stdio.h>
#include <string.h>
#include <stdbool.h>

#define STATE_COUNT 2
#define ALPHABET_COUNT 4

typedef struct {
   int transitions[ALPHABET_COUNT][STATE_COUNT];
   bool output[ALPHABET_COUNT][STATE_COUNT];
   char alphabet[ALPHABET_COUNT];
   int state;
   int symbol;
} automata;

bool process_word(automata automata, char *word) {
   bool result;
   automata.state = 0;
   for (int char_index = 0;
        char_index < strlen(word);
        ++char_index)
   {
      automata.symbol = word[char_index];
      int alphabet_index;
```

```
        for (alphabet_index = 0;
             alphabet_index < ALPHABET_COUNT;
             ++alphabet_index)
        {
            if (automata.alphabet[alphabet_index] == automata.symbol)
                break;
        }

        if (alphabet_index == ALPHABET_COUNT &&
            automata.alphabet[alphabet_index] != automata.symbol)
        {
            printf("Unexpected characters\n");
            return false;
        }

        result = automata.output[alphabet_index][automata.state];
        automata.state = automata.transitions[alphabet_index][automata.state];
    }
    return result;
}

int main(int argc, char *argv[]) {
    automata automata = {0};
    automata.transitions[2][0] = 1;
    automata.output[2][0] = 1;
    automata.alphabet[0] = 'a';
```

```
automata.alphabet[1] = 'b';
automata.alphabet[2] = 'c';
automata.alphabet[3] = 'd';

if (argc == 2) {
    if (process_word(automata, argv[1]))
        printf("Input word \"%s\" is accepting\n", argv[1]);
    else
        printf("Input word \"%s\" doesn't accept\n", argv[1]);
} else {
    int test_count = 14;
    char *tests[test_count];
    tests[0] = "abc";
    tests[1] = "abdc";
    tests[2] = "c";
    tests[3] = "caccc";
    tests[4] = "ccc";
    tests[5] = "ccccc";
    tests[6] = "abdcabac";
    tests[7] = "a";
    tests[8] = "abcb";
    tests[9] = "babcbbdbcc";
    tests[10] = "cc";
    tests[11] = "cccc";
    tests[12] = "cccccc";
    tests[13] = "bacabcccc";
```

```
        for (int test_index = 0; test_index < test_count; ++test_index)
            if (process_word(automata, tests[test_index]))
                printf("Input word \"%s\" is accepting\n", tests[test_index]);
            else
                printf("Input word \"%s\" doesn't accept\n", tests[test_index]);
    }

    return 0;
}
```