

Министерство цифрового развития, связи и массовых коммуникаций
РФ

Федеральное государственное бюджетное образовательное
учреждение высшего образования

«Санкт-Петербургский государственный университет
телекоммуникаций им. проф. М.А. Бонч-Бруевича» (СПбГУТ)

Кафедра программной инженерии и вычислительной техники

МЕТОДИЧЕСКИЕ УКАЗАНИЯ к лабораторной работе

Дисциплина: «Алгоритмические основы программной инженерии»

Курс 1-2, бакалавриат, направление 09.03.04 «Программная
инженерия»

Санкт-Петербург, 2026

1. ТИТУЛЬНАЯ ЧАСТЬ И ИНФОРМАЦИЯ

Параметр	Значение
Тема	Разработка ДКА и моделирование МТ на C++
Цель	Сформировать практические навыки проектирования конечных автоматов и программирования машин Тьюринга с использованием объектно-ориентированного подхода.
Длительность	4 академических часа (180 минут)
Форма проведения	компьютерный класс / индивидуально
ПО	g++ 9+ или MSVC 2019+, C++17, VS Code / CLion / Code::Blocks
Источник	М.А. Бирюков, Т.В. Ермакова, А.И. Попов «Алгоритмические основы программной инженерии», 2026 (главы 8-9)

Оглавление:

- Цели и образовательные результаты
- Теоретическая справка
- План работы с таймингом
- Задание 1: ДКА

5. Задание 2: Эмулятор МТ
6. Задание 3: Индивидуальные варианты
7. Требования к отчёту
8. Критерии оценки
9. Контрольные вопросы
10. Список литературы

2. ЦЕЛИ И ОБРАЗОВАТЕЛЬНЫЕ РЕЗУЛЬТАТЫ

Цель работы: Изучение математических основ теории автоматов и машин Тьюринга, а также приобретение практических навыков их алгоритмической реализации на языке C++ с применением структур данных STL.

После выполнения лабораторной работы студент **ДОЛЖЕН ЗНАТЬ:**

- формальное определение детерминированного конечного автомата (ДКА) $A = (Q, \Sigma, \delta, q_0, F)$;
- определение машины Тьюринга (МТ) и её такт работы;
- формат правила перехода МТ в виде $(q_i, a) \rightarrow (q_j, b, D)$;
- иерархию Хомского и классы распознаваемых языков (кратко).

ДОЛЖЕН УМЕТЬ:

- проектировать таблицу переходов ДКА по словесному описанию формального языка;
- реализовывать ДКА на C++17 с использованием контейнеров STL (``std::map``, ``std::set``);
- разрабатывать и реализовывать эмулятор одноленточной МТ;
- составлять программы (наборы правил) для машины Тьюринга;
- тестировать и отлаживать разработанные программы.

3. ТЕОРЕТИЧЕСКАЯ СПРАВКА

3.1. Конечный автомат

Детерминированный конечный автомат (ДКА) формально задаётся пятёркой $A = (Q, \Sigma, \delta, q_0, F)$, где:

- Q — конечное множество состояний;
- Σ — конечный входной алфавит;
- δ — функция переходов ($\delta: Q \times \Sigma \rightarrow Q$);
- $q_0 \in Q$ — начальное состояние;
- $F \subseteq Q$ — множество допускающих (конечных) состояний.

Расширенная функция переходов определяется рекурсивно: $\delta^*(q, \epsilon) = q$; $\delta^*(q, xa) = \delta(\delta^*(q, x), a)$. Автомат допускает слово w , если $\delta^*(q_0, w) \in F$. ДКА распознают класс *регулярных языков*.

3.2. Машина Тьюринга

Машина Тьюринга состоит из бесконечной ленты, разбитой на ячейки, читающей/пишущей головки и управляющего устройства. Конфигурация МТ описывается текущим состоянием, содержимым ленты и позицией головки.

Каждое правило имеет вид $(qi, a) \rightarrow (qj, b, D)$, где qi — текущее состояние, a — считанный символ, qj — новое состояние, b — записываемый символ, $D \in \{L, R, S\}$ — направление сдвига (Left, Right, Stay).

Такт работы: считать символ $\rightarrow$ найти подходящее правило $\rightarrow$ записать новый символ $\rightarrow$ сменить состояние $\rightarrow$ сдвинуть головку.

3.3. Иерархия Хомского

Согласно иерархии Хомского, ДКА распознают регулярные языки (Тип 3), которые являются наиболее ограниченными. Машины Тьюринга способны распознавать рекурсивно перечислимые языки (Тип 0) и по выразительной мощности эквивалентны любым современным алгоритмам (согласно тезису Чёрча–Тьюринга).

4. ПЛАН РАБОТЫ С ТАЙМИНГОМ

Время	Длительность	Этап	Описание
-------	--------------	------	----------

00:00–10:00	10 мин	Орг. часть, входной контроль	Проверка готовности, ответы на 3-5 вопросов
10:00–25:00	15 мин	Подготовка среды	Проверка g++ --version, создание проекта
25:00–70:00	45 мин	ЗАДАНИЕ 1	ДКА для языка $L = \{w \in \{0,1\}^* \mid w \text{ заканчивается на } \langle 01 \rangle\}$
70:00–75:00	5 мин	Перерыв	Краткий отдых
75:00–140:00	65 мин	ЗАДАНИЕ 2	Эмулятор МТ + инкремент двоичного числа
140:00–165:00	25 мин	ЗАДАНИЕ 3	Выполнение индивидуального варианта
165:00–175:00	10 мин	Оформление	Тестирование, подготовка отчёта
175:00–180:00	5 мин	Защита	Ответы на контрольные вопросы
ИТОГО:			180 минут

ВХОДНОЙ КОНТРОЛЬ (преподаватель задаёт 3-5 вопросов):

1. Что такое алфавит и слово в формальных языках?
2. Сформулируйте пятёрку конечного автомата.
3. Чем детерминированный КА отличается от недетерминированного?
4. Что такое регулярный язык?
5. Какие компоненты входят в состав машины Тьюринга?

5. ЗАДАНИЕ 1: Разработка ДКА

5.1. Постановка: реализовать ДКА для языка $L = \{w \in \{0,1\}^* \mid w \text{ заканчивается на } \langle 01 \rangle\}$.

5.2. Анализ задачи: нам потребуется три состояния. q_0 (ничего не прочитано или прервана последовательность), q_1 (прочитан символ '0', ждём '1'), q_2 (прочитана последовательность «01» — это состояние допускающее).

5.3. Таблица переходов:

Состояние	Вход 0	Вход 1	Допускающее
$\rightarrow q_0$	q_1	q_0	нет
q_1	q_1	q_2	нет

q2 (F)	q1	q0	ДА
--------	----	----	----

5.4. ШАБЛОН C++ КОДА:

```
// dfa_template.cpp - Шаблон ДКА для языка L = {w | w заканчивается на "01"}
// Компиляция: g++ -std=c++17 -Wall dfa_template.cpp -o dfa
#include <iostream>
#include <string>
#include <map>
#include <set>
#include <utility>

class DFA {
private:
    std::map<std::pair<int, char>, int> delta; // функция переходов
    int startState;
    std::set<int> acceptStates;
public:
    DFA(int start) : startState(start) {}

    void setTransition(int from, char symbol, int to) {
        // TODO: добавить переход в delta
    }

    void addAcceptState(int q) {
        // TODO: добавить q в acceptStates
    }

    bool run(const std::string& input) {
        // TODO: реализовать симуляцию автомата
        return false;
    }
};

int main() {
    DFA dfa(0);
    // TODO: задать переходы согласно таблице

    std::string w;
    std::cout << "Введите строку из {0,1}: ";
    std::cin >> w;
    std::cout << (dfa.run(w) ? "ACCEPTED" : "REJECTED") << std::endl;
    return 0;
}
```

Пример: ЭТАЛОННОЕ РЕШЕНИЕ (DFA)

```
#include <iostream>
#include <string>
#include <map>
#include <set>
#include <utility>

class DFA {
private:
    std::map<std::pair<int, char>, int> delta;
    int startState;
    std::set<int> acceptStates;
public:
    DFA(int start) : startState(start) {}

    void setTransition(int from, char symbol, int to) {
        delta[{from, symbol}] = to;
    }

    void addAcceptState(int q) { acceptStates.insert(q); }
```

```

bool run(const std::string& input){
    int current = startState;
    for (char c : input) {
        auto it = delta.find({current, c});
        if (it == delta.end()) return false; // нет перехода
        current = it->second;
    }
    return acceptStates.count(current) > 0;
}

};

int main() {
    DFA dfa(0); // q0 = 0
    // Переходы согласно таблице
    dfa.setTransition(0, '0', 1);
    dfa.setTransition(0, '1', 0);
    dfa.setTransition(1, '0', 1);
    dfa.setTransition(1, '1', 2);
    dfa.setTransition(2, '0', 1);
    dfa.setTransition(2, '1', 0);
    dfa.addAcceptState(2);

    // Тесты
    std::string tests[] = {"01", "101", "10", "0011", ""};
    for (const auto& t : tests) {
        std::cout << "\"\" << t << "\" -> "
                    << (dfa.run(t) ? "ACCEPTED" : "REJECTED") << std::endl;
    }
    return 0;
}

```

5.5. Таблица ожидаемых результатов тестов:

Вход	Ожидание	Объяснение
«01»	ACCEPTED	заканчивается на 01
«101»	ACCEPTED	заканчивается на 01
«10»	REJECTED	заканчивается на 0
«0011»	ACCEPTED	заканчивается на 01
«»	REJECTED	пустая строка



Подсказка: для отладки выводите промежуточное состояние после считывания каждого символа с помощью `std::cout` внутри цикла `for (char c : input)`.

6. ЗАДАНИЕ 2: Моделирование Машины Тьюринга

6.1. Постановка: реализовать эмулятор одноленточной МТ и написать для него программу инкремента двоичного числа (прибавления единицы).

6.2. Таблица правил инкремента:

Состояние	Символ	Новое состояние	Запись	Сдвиг
q0	1	q0	0	L
q0	0	qf	1	S
q0	□ ()	qf	1	S

Пояснение алгоритма: идём справа налево, заменяя 1→0 (перенос), пока не встретим 0 или пустоту (□) — пишем 1 и останавливаемся (qf).

⚠ Внимание: Головка перед запуском ставится на **МЛАДШИЙ** разряд (самый правый символ). Не путайте порядок!

6.3. ШАБЛОН C++ КОДА:

```
// tm_template.cpp - Шаблон эмулятора одноленточной машины Тьюринга
// Компиляция: g++ -std=c++17 -Wall tm_template.cpp -o tm
#include <iostream>
#include <string>
#include <vector>
#include <map>
#include <tuple>
#include <utility>

class TuringMachine {
public:
    static constexpr char BLANK = '_'; // символ пустой ячейки □
private:
    std::vector<char> tape;
    int head;
    std::string state;
    std::string haltState;
    std::map<std::pair<std::string, char>, std::tuple<std::string, char, char>>
rules;
public:
    TuringMachine(const std::string& start, const std::string& halt)
        : head(0), state(start), haltState(halt) {}

    void setRule(const std::string& q, char a,
                const std::string& q2, char b, char dir) {
        // TODO: добавить правило в rules
    }

    void loadTape(const std::string& input, int startHead) {
        // TODO: загрузить input в tape, установить head = startHead
    }

    bool step() {
        // TODO: один такт работы МТ:
        // 1. Если state == haltState – вернуть false
        // 2. Считать символ под головкой (BLANK если за пределами)
        // 3. Найти правило rules[{state, символ}]
        // 4. Записать новый символ, сменить состояние, сдвинуть головку
        // 5. Если правила нет – остановиться
        return false;
    }

    void run(int maxSteps = 10000) {
```

```

        // TODO: вызывать step() пока возможно или пока шаги не исчерпаны
    }

    std::string getTape() const {
        // TODO: вернуть содержимое ленты без крайних BLANK
        return "";
    }
};

int main() {
    TuringMachine tm("q0", "qf");
    // TODO: задать правила инкремента

    std::string input;
    std::cout << "Введите двоичное число: ";
    std::cin >> input;
    tm.loadTape(input, (int)input.size() - 1); // головка на младшем разряде
(справа)
    tm.run();
    std::cout << "Результат: " << tm.getTape() << std::endl;
    return 0;
}

```

6.4. Пример: ЭТАЛОННОЕ РЕШЕНИЕ (Эмулятор МТ)

```

#include <iostream>
#include <string>
#include <vector>
#include <map>
#include <tuple>

class TuringMachine {
public:
    static constexpr char BLANK = '_';
private:
    std::vector<char> tape;
    int head;
    std::string state;
    std::string haltState;
    std::map<std::pair<std::string, char>, std::tuple<std::string, char, char>>
rules;

    char readSymbol() {
        if (head < 0 || head >= (int)tape.size()) return BLANK;
        return tape[head];
    }

    void writeSymbol(char b) {
        while (head < 0) { tape.insert(tape.begin(), BLANK); head++; }
        while (head >= (int)tape.size()) tape.push_back(BLANK);
        tape[head] = b;
    }
public:
    TuringMachine(const std::string& start, const std::string& halt)
        : head(0), state(start), haltState(halt) {}

    void setRule(const std::string& q, char a,
                const std::string& q2, char b, char dir) {
        rules[{q, a}] = std::make_tuple(q2, b, dir);
    }

    void loadTape(const std::string& input, int startHead) {
        tape.assign(input.begin(), input.end());
        head = startHead;
    }

    bool step() {
        if (state == haltState) return false;
        char a = readSymbol();
        auto it = rules.find({state, a});

```

```

        if (it == rules.end()) return false;

        auto [q2, b, dir] = it->second;
        writeSymbol(b);
        state = q2;
        if (dir == 'L') head--;
        else if (dir == 'R') head++;
        return true;
    }

    void run(int maxSteps = 10000) {
        for (int i = 0; i < maxSteps; ++i) {
            if (!step()) return;
        }
    }

    std::string getTape() const {
        int l = 0, r = tape.size() - 1;
        while (l <= r && tape[l] == BLANK) l++;
        while (r >= l && tape[r] == BLANK) r--;
        if (l > r) return "";
        return std::string(tape.begin() + l, tape.begin() + r + 1);
    }
};

int main() {
    std::string tests[] = {"1011", "111", "0"};
    for (const auto& t : tests) {
        TuringMachine tm("q0", "qf");
        tm.setRule("q0", '1', "q0", '0', 'L');
        tm.setRule("q0", '0', "qf", '1', 'S');
        tm.setRule("q0", '_', "qf", '1', 'S');

        tm.loadTape(t, (int)t.size() - 1);
        tm.run();
        std::cout << t << " -> " << tm.getTape() << std::endl;
    }
    return 0;
}

```

6.5. Тесты инкремента:

Вход	Ожидание	Комментарий
«1011»	«1100»	$11 + 1 = 12$
«111»	«1000»	$7 + 1 = 8$ (переполнение разряда)
«0»	«1»	$0 + 1 = 1$

 **Подсказка:** для трассировки работы МТ полезно после каждого вызова `step()` выводить содержимое ленты и позицию головки. Это поможет понять, как машина «движется» по ленте.

7. ЗАДАНИЕ 3: Индивидуальные варианты

Каждый студент выполняет один вариант согласно номеру в списке группы (модуль 10).

Вар.	Описание задачи	Тестовые примеры
B1	ДКА для языка слов в алфавите $\{0,1\}$ с ЧЁТНЫМ количеством нулей.	$\langle \rangle \rightarrow \text{accept}$, $\langle 00 \rangle \rightarrow \text{accept}$, $\langle 010 \rangle \rightarrow \text{reject}$
B2	ДКА для слов, НАЧИНАЮЩИХСЯ с $\langle 11 \rangle$.	$\langle 11 \rangle \rightarrow \text{accept}$, $\langle 110 \rangle \rightarrow \text{accept}$, $\langle 1 \rangle \rightarrow \text{reject}$
B3	ДКА для слов, длина которых КРАТНА 3.	$\langle \rangle \rightarrow \text{accept}$, $\langle 010 \rangle \rightarrow \text{accept}$, $\langle 0101 \rangle \rightarrow \text{reject}$
B4	ДКА для слов, СОДЕРЖАЩИХ подстроку $\langle 101 \rangle$.	$\langle 101 \rangle \rightarrow \text{accept}$, $\langle 01011 \rangle \rightarrow \text{accept}$, $\langle 11 \rangle \rightarrow \text{reject}$
B5	ДКА для слов, НЕ содержащих $\langle 00 \rangle$.	$\langle \rangle \rightarrow \text{accept}$, $\langle 1010 \rangle \rightarrow \text{accept}$, $\langle 100 \rangle \rightarrow \text{reject}$
B6	МТ для УДВОЕНИЯ количества единиц ($1^n \rightarrow 1^{2n}$).	$\langle 1 \rangle \rightarrow \langle 11 \rangle$, $\langle 11 \rangle \rightarrow \langle 1111 \rangle$
B7	МТ для КОПИРОВАНИЯ слова из $\{a,b\}^*$ через '#': $w\# \rightarrow w\#w$.	$\langle ab\# \rangle \rightarrow \langle ab\#ab \rangle$, $\langle aa\# \rangle \rightarrow \langle aa\#aa \rangle$
B8	МТ для СЛОЖЕНИЯ двух унарных чисел: $1^n + 1^m \rightarrow 1^{n+m}$.	$\langle 111 + 11 \rangle \rightarrow \langle 11111 \rangle$
B9	МТ для определения ЧЁТНОСТИ двоичного числа (0/1).	$\langle 1010 \rangle \rightarrow \langle 0 \rangle$ (чёт), $\langle 11 \rangle \rightarrow \langle 1 \rangle$ (нечёт)
B10	МТ для побитовой ИНВЕРСИИ: $0 \rightarrow 1$ и $1 \rightarrow 0$.	$\langle 1010 \rangle \rightarrow \langle 0101 \rangle$, $\langle 111 \rangle \rightarrow \langle 000 \rangle$

8. ТРЕБОВАНИЯ К ОТЧЁТУ

Отчёт должен быть оформлен в формате PDF/DOCX и содержать:

1. Титульный лист (вуз, группа, ФИО, преподаватель, дата).
2. Цель работы.
3. Краткие теоретические сведения (0.5 страницы).
4. Задание 1: постановка, таблица переходов, листинг кода, скриншот вывода, анализ.
5. Задание 2: постановка, таблица правил, листинг, скриншот, трассировка одного теста.
6. Задание 3: номер варианта, постановка, описание решения, листинг, тесты.
7. Выводы (3-5 предложений).
8. Список литературы.

9. КРИТЕРИИ ОЦЕНКИ

Оценка	Критерии выполнения
«Отлично» (5)	Выполнены все 3 задания, код чистый и закомментирован, тесты пройдены, отчёт по стандарту, на защите даны полные ответы.
«Хорошо» (4)	Все 3 задания выполнены, незначительные замечания к коду или отчёту, отвечает на большинство вопросов.
«Удовл.» (3)	Задания 1 и 2 выполнены, задание 3 частично или с ошибками, отчёт имеется, базовые ответы на защите.
«Неудовл.» (2)	Не выполнены задания 1 или 2, либо отсутствует отчёт/защита.

10. КОНТРОЛЬНЫЕ ВОПРОСЫ ДЛЯ ЗАЩИТЫ

1. Дайте формальное определение ДКА.
2. Чем ДКА отличается от НКА? Эквивалентны ли они по выразительности?
3. Что такое расширенная функция переходов δ^* ?
4. Какой класс языков распознаётся конечными автоматами?
5. Приведите пример нерегулярного языка.
6. Что такое автомат Мили и автомат Мура? В чём их различие?
7. Опишите структуру машины Тьюринга.
8. Что такое конфигурация МТ?
9. Опишите такт работы МТ.
10. Какие направления движения головки существуют и что означает 'S'?
11. Что такое тезис Чёрча–Тьюринга?
12. Что такое проблема останова и почему она неразрешима?
13. Сравните выразительность ДКА, автомата с магазинной памятью и МТ.
14. Что произойдёт в Задании 2, если убрать правило $(q_0, _') \rightarrow (qf, '1', 'S')$?
15. Как можно реализовать многоленточную МТ через одноленточную?

11. СПИСОК ЛИТЕРАТУРЫ

1. Бирюков М.А., Ермакова Т.В., Попов А.И. Алгоритмические основы программной инженерии: учебное пособие. — СПб.: СПбГУТ, 2026. — Главы 8-9.
2. Хопкрофт Дж., Мотвани Р., Ульман Дж. Введение в теорию автоматов, языков и вычислений. — М.: Вильямс, 2008.
3. Ахо А., Сети Р., Ульман Дж. Компиляторы: принципы, технологии, инструменты. — М.: Вильямс, 2008.
4. Страуструп Б. Программирование. Принципы и практика с использованием С++. — 2-е изд. — М.: Вильямс, 2016.
5. `cppreference.com` — справочник по стандартной библиотеке С++ (<https://en.cppreference.com>).