ОГЛАВЛЕНИЕ

1. МЕТОДОЛОГИЯ TDD

TDD (Test Driven Development) - это методология разработки программного обеспечения, которая основана на написании тестов перед написанием кода. Основная идея TDD заключается в постоянном повторении коротких циклов разработки: написание теста, написание минимального количества кода для прохождения теста, рефакторинг кода.

Процесс разработки в TDD состоит из следующих шагов:

1. Написание теста: разработчик пишет тесты для функциональности, которую он планирует реализовать. Тесты должны быть корректными и проверять только одну функциональность.

2. Запуск теста: после написания теста он запускается и ожидается его провал.

3. Написание кода: разработчик пишет минимальное количество кода, необходимого для прохождения теста. При этом необходимо следовать принципу YAGNI (You Ain't Gonna Need It), то есть писать только необходимый функционал.

4. Запуск теста: после написания кода тесты запускаются снова. Если тест проходит успешно, то это означает, что новый код не нарушил работу предыдущего функционала.

5. Рефакторинг: на этом этапе происходит улучшение кода без изменения его функциональности. Разработчик удаляет дублирующийся код,

улучшает его читаемость и поддерживаемость. После каждого изменения кода тесты должны запускаться для проверки работоспособности.

Таким образом, методология TDD позволяет разрабатывать программное обеспечение с высоким качеством и минимальным количеством ошибок. Она также способствует более гибкому и быстрому процессу разработки, так как позволяет быстро выявлять и исправлять ошибки. Кроме того, наличие тестов позволяет легче вносить изменения в код и добавлять новый функционал, не боясь нарушить работу уже существующего.

2. ПРОВЕРКА МЕТОДОВ

Тестовый случай 1 представлен в таблице 1.

Таблица 1 – тестовый случай 1

Описание тестируемой функции	Базовые функции программы
Проект	Калькулятор метода «Отжиг»
Компонент системы	Основная форма
Номер версии	1.1.0.2
Окружение	ПК с ОС MacOS M1 процессор 8 ОЗУ 512 гигабайт памяти
Описание возникновения ошибки	• Запустить приложение • Ввод варианта, ввод числа • Проверить корректный запуск • Проверка данных
Ожидаемый результат	Программа выдаст кратчайший путь по графу
Фактический результат	Программа вывела оптимальный путь

Проверка тестового случая 1 представлена на рисунке 1.

```
Начальный Маршрут: { 5 2 4 6 7 0 9 8 3 1 5 } S = 270

Итерация 1

T = 100
Замена = 4 <> 2
Вероятность P = 74.8264 <> 14
Новый Маршрут: { 5 6 4 2 7 0 9 8 3 1 5 } S = 299
Маршрут принят т.к. P = 74.8264 > 14.


Итерация 2

T = 50
Замена = 8 <> 2
Вероятность P = 100 <> 46
Новый Маршрут: { 5 8 4 2 7 0 9 6 3 1 5 } S = 299
Маршрут принят т.к. P = 100 > 46.


Итерация 3

T = 25
Замена = 2 <> 8
Вероятность P = 100 <> 23
Новый Маршрут: { 5 6 4 2 7 0 9 8 3 1 5 } S = 299
Маршрут принят т.к. P = 100 > 23.


Итерация 4

T = 12.5
Замена = 7 <> 4
Вероятность P = 332.012 <> 86
Новый Маршрут: { 5 6 4 9 7 0 2 8 3 1 5 } S = 284
Маршрут принят т.к. dS = -15 < 0.


Итерация 5

T = 6.25
Замена = 3 <> 4
Вероятность P = 2878.92 <> 20
Новый Маршрут: { 5 6 9 4 7 0 2 8 3 1 5 } S = 263
Маршрут принят т.к. dS = -21 < 0.


Итерация 6

T = 3.125
Замена = 8 <> 8
Вероятность P = 100 <> 77
Новый Маршрут: { 5 6 9 4 7 0 2 8 3 1 5 } S = 263
Маршрут принят т.к. P = 100 > 77.
```

Рисунок 1 – проверка тестового случая 1.

Тестовый случай 2 представлен в таблице 2.

Таблица 2 – тестовый случай 2

Описание тестируемой функции	Базовые функции программы
Проект	Калькулятор метода «Отжиг»
Компонент системы	Основная форма
Номер версии	1.1.0.2
Окружение	ПК с ОС MacOS M1 процессор 8 ОЗУ 512 гигабайт памяти
Описание возникновения ошибки	• Запустить приложение • Ввод варианта, ввод числа • Проверить корректный запуск • Проверка данных
Ожидаемый результат	Программа выдаст ошибку о неправильном вводе
Фактический результат	Программа вывела ошибку

Проверка тестового случая 2 представлена на рисунке 2.

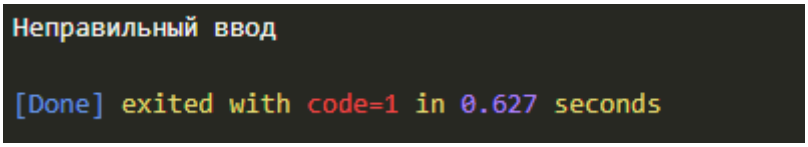

Рисунок 2 – Проверка второго тестового случая.

Тестовый случай 3 представлен в таблице 3.

Таблица 3 – тестовый случай 3

Описание тестируемой функции	Базовые функции программы
Проект	Калькулятор метода «Отжиг»
Компонент системы	Основная форма
Номер версии	1.1.0.2

Окружение	ПК с ОС MacOS M1 процессор 8 ОЗУ 512 гигабайт памяти
Описание возникновения ошибки	• Запустить приложение • Ввод варианта, ввод целочисленного числа • Проверить корректный запуск • Проверка данных
Ожидаемый результат	Программа выдаст ошибку о неправильном вводе в функции
Фактический результат	Программа вывела ошибку

Проверка тестового случая 3 представлена на рисунках 3.

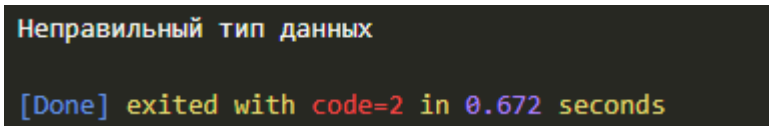

Рисунок 3 – Нажатие на кнопку.

Проект был разработан по методологии TDD. Последовательность коммитов представлена на рисунке 4.

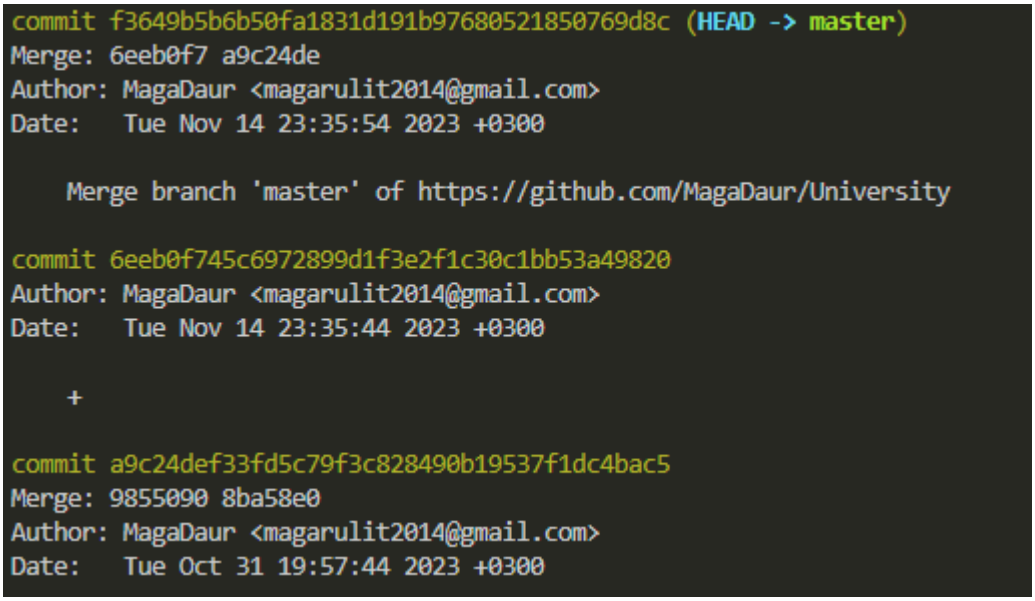

Рисунок 4 – Последовательность коммитов.

Код программы представлен в листинге 1.

Листинг 1. Код программы

```
#include <algorithm>
#include <random>
#include <iostream>
#include <vector>
#include <numeric>
#include <ctime>

struct SwapIter { int from; int to; };

double rd() { return double(rand()) / RAND_MAX; }

std::vector<int> SetupPorbabilities(int k)
{
    std::vector<int> probabilities(k);
    for(int i = 0; i < k; i++)
        probabilities[i] = (rd() * 100);
    return probabilities;
}

std::vector<SwapIter> SetupSwapTable(int n, int k)
{
    std::vector<SwapIter> swap_table(k);
    for(int i = 0; i < k; i++)
        swap_table[i] = {1 + (rand() % (n - 2)), 1 + (rand() % (n - 2))};
    return swap_table;
}

std::vector<std::vector<int>> SetupEdges(int n)
{
    std::vector<std::vector<int>> l(n);
    for(int i = 0; i < n; i++)
    {
        l[i].reserve(n);
        for(int j = 0; j < n; j++)
        {
            if(j < i)
            {
                l[i][j] = l[j][i];
                continue;
            }
            else if(j == i)
            {
                l[i][j] = 0;
                continue;
            }

            l[i][j] = 50 * (rd() + 0.1);
        }
    }
```

```
    return l;
}

int f(const std::vector<int>& path, const std::vector<std::vector<int>> &l)
{
    int res = 0;
    for(int i = 1; i < path.size(); i++)
        res += l[path[i - 1]][path[i]];
    return res;
}

int main()
{
    system("clear");
    //srand(time(0));

    const int n = 10;
    const int k = 20;
    const double y = 0.5;
    double t = 100.0;

    std::vector<std::vector<int>> l = SetupEdges(n);
    std::vector<int> probabilities = SetupPorbabilities(k);
    std::vector<SwapIter> swap_table = SetupSwapTable(n, k);

    std::vector<int> v(n); std::iota(v.begin(), v.end(), 0);
    std::shuffle(v.begin(), v.end(), std::default_random_engine(time(0)));
    v.push_back(v[0]);

    std::cout << "\nДлины граней:\n";
    for(int i = 0; i < n; i++)
        for(int j = i + 1; j < n; j++)
            std::cout << i + 1 << " -> " << j + 1 << " = " << l[i][j] << "\n";

    std::cout << "\nВероятности: {";
    for(int prob : probabilities)
        std::cout << " " << prob;
    std::cout << " }\n";

    std::cout << "\nЗамены:\n";
    for(const auto& iter : swap_table)
        std::cout << iter.from + 1 << " <> " << iter.to + 1 << "\n";

    int best_s = f(v, l);

    std::cout << "\nНачальный Маршрут: {";
    for(auto i : v)
        std::cout << " " << i;
    std::cout << " } S = " << best_s << "\n";
```

```
    for(int i = 0; i < k; i++)
    {
        std::cout << "\nИтерация " << i + 1 << "\n\n";

        std::cout << "T = " << t << "\n";

        std::vector<int> u = v;
        SwapIter swap_iter = swap_table[i];

        std::cout << "Замена = " << swap_iter.from + 1 << " <> " << swap_iter.to
+ 1 << "\n";

        std::swap(u[swap_iter.from], u[swap_iter.to]);

        int s = f(u, l);
        double ds = s - best_s;
        double p = 100.0 * exp(-ds / t);

        std::cout << "Вероятность P = " << p << " <> " << probabilities[i] <<
"\n";

        std::cout << "Новый Маршрут: {";
        for(auto idx : u)
            std::cout << " " << idx;
        std::cout << " } S = " << s << "\n";

        if(ds < 0 || p > probabilities[i])
        {
            best_s = s;
            v = u;

            if(ds < 0)
                std::cout << "Маршрут принят т.к. dS = " << ds << " < 0.\n\n";
            else
                std::cout << "Маршрут принят т.к. P = " << p << " > " <<
probabilities[i] <<".\n\n";
        }
        else
            std::cout << "Маршрут не принят.\n\n";

        t *= y;
    }

    std::cout << "\nЛучший маршрут {";
    for(auto idx : v)
        std::cout << " " << idx;
    std::cout << " }, где S = " << best_s << ".\n";

    return 0;
}
```