Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Компьютерные системы и сети**

О Т Ч Е Т

по лабораторной работе № 3

Название: Изучение структур данных и методов работы с ними

Дисциплина: Технология разработки программных систем

Студент	ИУ6-42Б	14.04.2025	Т.А. Гаджиев
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель		14.04.2025	Е.К. Пугачёв
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2025

СОДЕРЖАНИЕ

Цель работы

Приобрести навыки тестирования схем алгоритмов, исходных кодов программ и исполняемых модулей.

Ход работы

1. Структурный подход

1.1 Задание для структурного анализа

Программа должна создавать динамический односвязный список вещественных чисел и проверять на совпадение первой половины списка со второй (файл исходного кода v2.doc).

1.2 Исходный код программы для тестирования

Листинг 1 — Код для структурного анализа

```
#include <iostream>
using namespace std;

struct el {
    int value;
    el* next;
};

int main() {
    el *First, *pass, *q;
    int pov, k, g, i, n;
    int mas[100];
    First = nullptr;
    cout << "Enter numbers (1000 to end): " << endl;
    cin >> i;
    First = new el;
    First->value = i;
    First->next = nullptr;
    pass = first;
    k = 1;
    cin >> i;

    do {
        q = new el;
        q->value = i;
        q->next = nullptr;
        pass->next = q;
        pass = q;
        k++;
        cin >> i;
    } while (i == 1000);

    q = first;
    while (q != nullptr) {
        cout << q->value << " ";
        q = q->next;
    }
    cout << endl;
```

```
    g = k;
    k = k / 2;

    q = first;
    for (i = 0; i < k; i++) {
        q = q->next;
        pass = q;
    }

    for (i = 0; i < k; i++) {
        pov = 0;
        if (q->value == pass->value) {
            pov++;
            q = q->next;
            pass = pass->next;
        } else {
            q = q->next;
            pass = pass->next;
        }
    }

    if (pov == i)
        cout << "Matches" << endl;
    else
        cout << "Not matches" << endl;

    return 0;
}
```

Таблица 1 — Результаты структурного тестирования

Номер вопроса	Строки, подлежащие проверке	Результат проверки	Вывод
1.1	12, 63	Переменные int pov, k, g, i, n не инициализированы при объявлении	Ошибка! pov не инициализирован, что приведет к ошибке в if (pov == i). Переменная n вообще не используется.
1.2	33	Неправильный алгоритм выхода для ввода массива	Условие while (i == 1000) приведет к ошибке ввода массива

			(неправильное условие выхода)
1.4	15, 21	Опечатка в регистре	Несоответствие first и First, ошибка
1.6	18, 26	First = new el; (строка 18), q = new el; (строка 26) — выделение памяти	Ошибка! Нет Delete для освобождения динамической памяти
3.4	51-61	Некорректное условие проверки pov == i. Счётчик pov обнуляется при каждой итерации, а проверку надо делать с числом k.	Неправильная логика алгоритма, нет выхода при несовпадении

1.3 Выводы

структурный контроль позволяет обнаружить общие ошибки кодирования.

Достоинства:

• не требует выполнения программы;

• позволяет обнаружить общие ошибки программирования.

Недостатки:

• ошибки, на обнаружение которых направлен структурный контроль, автоматически выявляются средствами разработки;

• по списку вопросов трудно обнаружить ошибки в логике программы;

• большие программы трудно инспектировать.

2. Белый ящик

2.1 Задание для метода «белого ящика»

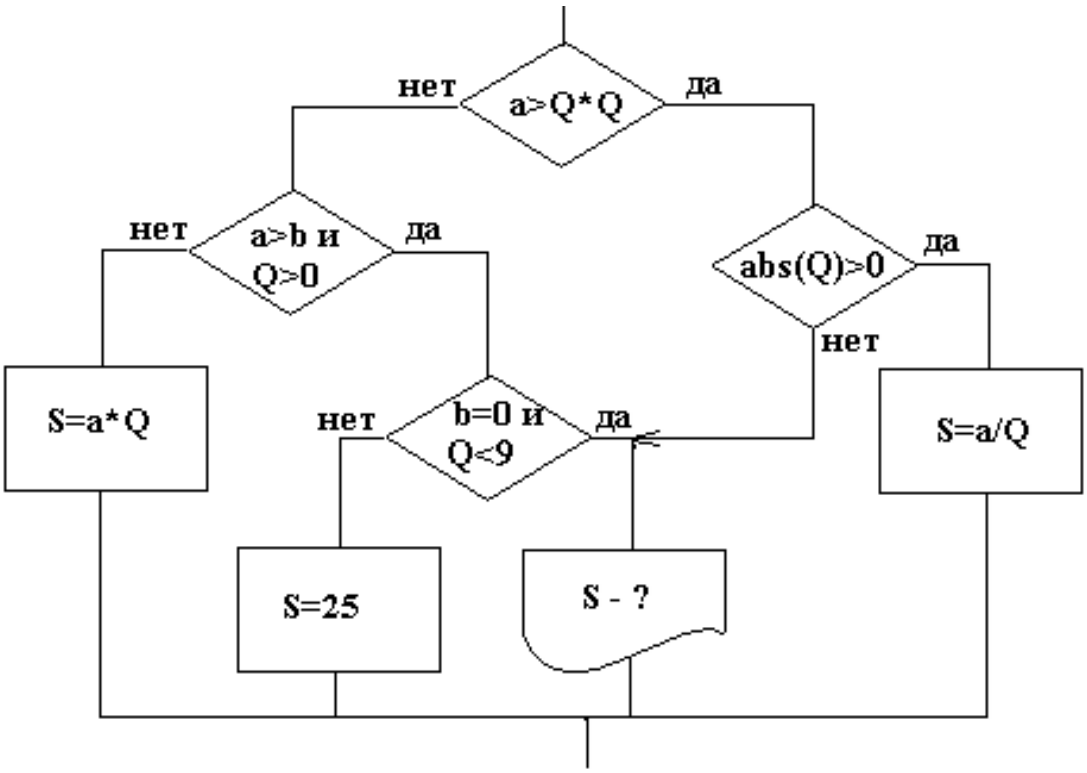

Рисунок 1 — Схема алгоритм задания для метода «белого ящика»

2.2 Метод покрытия операторов

Результаты тестирования по методу покрытия операторов представлены в таблице 2.

Таблица 2 — Результаты тестирования по методу покрытия операторов

Номер теста	Назначение теста	Значения исходных данных	Ожидаемый результат
1	Проверить оператор S=Q*a	a = 2 b = 3 Q = 4	S = 8
2	Проверить оператор S=25	a = 2 b = 1 Q = 4	S = 25
3	Проверить оператор S=a/Q	a = 6 b = 1 Q = 2	S = 3

2.3 Метод покрытия решений

Результаты тестирования по методу покрытия решений представлены в таблице 3.

Таблица 3 – Результаты тестирования по методу покрытия решений

Номер теста	Назначение теста	Значения исходных данных	Ожидаемый результат
1	да да	a = 6 b = 1 Q = 2	S = 3
2	да нет	a = 6 b = 1 Q = 0	S не присвоено значение!
3	нет да да	a = 3 b = 0 Q = 2	S не присвоено значение!
4	нет да нет	a = 3 b = 1 Q = 4	S = 25
5	нет нет	a = 3 b = 5 Q = 4	S = 12

2.4 Метод комбинаторного покрытия решений

По схеме алгоритма можно выделить 10 комбинаций условий:

1. a>Q*Q
2. a<Q*Q
3. a>b Q>0
4. a<b Q>0
5. a<b Q<0
6. a>b Q<0
7. b=0 Q<9
8. b=0 Q>9
9. abs(Q)>0
10. abs(Q)<0

Вышеперечисленные комбинации можно покрыть 6 тестами. Результаты тестирования представлены в таблице 4.

Таблица 4 — Таблица результатов тестирования для комбинаторного покрытия условий

Номер теста	Номера покрытия вариантов	Значения исходных данных	Ожидаемый результат
1	1, 9	a = 5 b = 1 Q = 1	S = 5
2	1, 10	a = 1 b = 2 Q = 0	S не присвоено значение!
3	2, 3, 7	a = 3 b = 0 Q = 2	S не присвоено значение!
4	2, 3, 8	a = 3 b = 0 Q = 10	S = 25
5	2, 4	a = 1 b = 2 Q = 3	S = 3
6	2, 5, 6	a = 1 b = 2 Q = -1	S = -1

2.5 Вывод

В ходе тестирования при помощи комбинированного метода и метода покрытия решений удалось обнаружить ошибки: при некоторых входных данных значение S инициализировалось мусорным значением, так как во время выполнения алгоритма ему не было оно присвоено.

3. Метод «Чёрного ящика»

3.1 Задание для метода «чёрного ящика»

Реализовать калькулятор, который выполняет два действия «+» и «–» с целыми числами (исполняемый модуль v2.exe).

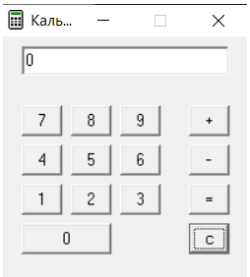

Рисунок 2 – Интерфейс программы для тестирования

3.2 Метод эквивалентного разбиения

Результаты тестирования методом эквивалентного преобразования представили в таблице 5.

Таблица 5 – Результаты тестирования

Номер теста	Назначение теста	Значение исходных данных	Ожидаемый результат	Реакция программы	Вывод
1	Проверка ввода чисел	Последовательно 5 8 5 4 8	В поле: 58548	58548	OK
2	Проверка операции	Последовательно 5 + 3	В поле: 8	8	OK
3	Проверка вычитания с отрицательным результатом	Последовательно 5 - 8	В поле: -3	-3	OK
4	Проверка сброса	Последовательно 5 + 3 C	В поле: 0	0	OK
5	Проверка многократного сложения	Последовательно 5 + 3 + 2	В поле: 8 -> 10	10	OK
6	Проверка ввода нулей	Последовательно 0 0 0 0 0	В поле: 0	0	OK
8	Проверка ввода не чисел	В поле abdsds	В поле: abdsds	0	OK

3.3 Метод граничных значений

Программа работает стабильно при любых сценариях работы.

3.4 Вывод

Одни и те же ошибки можно обнаружить разными методами «черного ящика»; тестирование — это очень трудоемкий процесс; нет гарантии, что обнаружены все ошибки даже при обеспечении полноты тестов по каждому методу.

Вывод

В ходе выполнения лабораторной работы освоили методы тестирования, а именно структурный метод, методы «белого» и «чёрного» ящиков. Приобрели навыки тестирования схем алгоритмов, исходных кодов программ и исполняемых модулей.