



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 Информатика и вычислительная техника

ОТЧЕТ

По лабораторной работе №2

Название: Тестирование и повышение качества программ

Дисциплина: Технология разработки программных систем

Студент

ИУ-426

(Группа)

(Подпись, дата)

С.В. Астахов

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

(И.О. Фамилия)

Москва, 2021

Вариант 3

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

Структурный контроль

Задача: Программа должна формировать массив чисел от 3 до 25, а затем сортировать элементы массива по возрастанию и исключать повторяющиеся элементы (файл исходного кода v3.doc).

Исходный код:

```
#include <iostream>
#include <time.h>
#define N 10
using namespace std;

int main()
{
    int i, j, k, L, b, m[N];

    L = N;
    srand(time(0));
    for (i = 0; i < L; i++) { m[i] = rand() % 25 + 3; cout << m[i] << '
'; }
    cout << endl;
    k = 1;
    do {
        i = 0;
        do {
            if (m[i] == m[i + 1])
            {
                for (j = i; j < L - 1; j++) m[j] = m[j + 1]; L--;
i--;
            }
            else if (m[i] > m[i + 1]) {
                b = m[i]; m[i] = m[i + 1]; m[i + 1] = b;
            }
            i++;
        } while (i < L - 1);
        k++;
    } while (k > L - 1);
    for (i = 0; i < L; i++) cout << m[i] << ' ';
    cout << endl;
}
```

Ответим на вопросы и заполним таблицу:

1 Обращение к данным

1.1 Все ли переменные инициализированы? ДА

1.2 Не превышены ли максимальные (или реальные) размеры массивов и строк? НЕТ

1.3 Не перепутаны ли строки со столбцами при работе с матрицами? НЕ ПЕРЕПУТАНЫ / ВОПРОС НЕ АКТУАЛЕН

1.4 Присутствуют ли переменные со сходными именами? ДА, i и j , однако их имена являются «стандартными»

1.5 Используются ли файлы? НЕТ

1.6 Использованы ли нетипизированные переменные, открытые массивы, динамическая память? НЕТ

2 Вычисления

2.1 Правильно ли записаны выражения? НЕТ, $m[i] = \text{rand}() \% 25 + 3$ выходит за границы $[3..25]$, указанные в задании

2.2 Корректно ли производятся вычисления неарифметических переменных? ДА / ВОПРОС НЕ АКТУАЛЕН

2.3 Корректно ли выполнены вычисления с переменными различных типов? ДА / ВОПРОС НЕ АКТУАЛЕН

2.4 Возможны ли переполнение разрядной сетки или ситуация машинного нуля? НЕТ

2.5 Соответствуют ли вычисления с заданным требованиям точности? ДА / ВОПРОС НЕ АКТУАЛЕН

2.6 Присутствуют ли сравнения переменных различных типов? НЕТ

3 Передача управления

3.1 Будут ли корректно завершены циклы? НЕТ, цикл `while (k > L — 1)` будет завершён после первой итерации

3.2 Будет ли завершена программа? ДА

3.3 Существуют ли циклы, которые не будут выполняться из-за нарушения условий входа? ФОРМАЛЬНО НЕТ, но цикл `while (k > L — 1)` будет завершён после первой итерации

3.4 Существуют ли поисковые циклы? НЕТ

4 Интерфейс

4.1 Соответствуют ли списки параметров и аргументов по порядку, типу, единицам измерения? ДА / ВОПРОС НЕ АКТУАЛЕН

4.2 Не изменяет ли подпрограмма аргументов, которые не должны изменяться? НЕ ИЗМЕНЯЕТ / ВОПРОС НЕ АКТУАЛЕН

4.3 Не происходит ли нарушения области действия глобальных и локальных переменных с одинаковыми именами? НЕ ПРОИСХОДИТ / ВОПРОС НЕ АКТУАЛЕН

4.4 Соответствует ли выводимая информация требованиям задачи?
ФОРМАТ — ДА, значения — нет, в связи с ошибками в пунктах 2.1, 3.1, 3.3

Таблица 1 -результаты структурного контроля

№ вопроса	Строки, подлежащие проверке	Результат проверки	Вывод
1.1	13, 15, 15, 17, 23, 26	<pre>L=N; for (i = 0 <...>) m[i] = rand() % 25 + 3 k = 1; for (j = i <...>) b = m[i];</pre>	Все переменные инициализированы до того как их значения будут считаны.
1.2	15, 23, 29, 32	<pre>for (i = 0; i < L; i++) for (j = i; j < L - 1; j++) while (i < L — 1); for (i = 0; i < L; i++)</pre>	Переменные, индексирующие массив не превышают реальный размер массива
2.1	15	<pre>m[i] = rand() % 25 + 3</pre>	Результат — значение от 3 до 27, задание же требует результат в диапазоне от 3 до 25
3.1, 3.3	31	<pre>while (k > L - 1)</pre>	Так как L=10, а k=2 к концу первой итерации — цикл завершится и сортировка не будет доведена до конца

Корректный вариант программы:

```
#include <iostream>
#include <time.h>
#define N 10
using namespace std;

int main()
{
    int i, j, k, L, b, m[N];

    L = N;
    srand(time(0));
    for (i = 0; i < L; i++) { m[i] = rand() % 23 + 3; cout << m[i] << '
'; }
    cout << endl;
    k = 1;
    do {
        i = 0;
        do {
            if (m[i] == m[i + 1])
            {
                for (j = i; j < L - 1; j++) m[j] = m[j + 1]; L--;
i--;
            }
            else if (m[i] > m[i + 1]) {
                b = m[i]; m[i] = m[i + 1]; m[i + 1] = b;
            }
            i++;
        } while (i < L - 1);
        k++;
    } while (k < L - 1);
    for (i = 0; i < L; i++) cout << m[i] << ' ';
    cout << endl;
}
```

Вывод:

Структурный контроль применим на ранних этапах разработки для обнаружения типовых ошибок в структуре программы, так же он позволяет контролировать технологичность программы. Кроме того, специалисты могут обнаружить и какую-либо неявную ошибку, которую будет трудно отыскать с помощью более формализованных стратегий тестирования.

Основные недостатки структурного контроля — необходимость задействовать специалистов (расход времени ценных сотрудников), большая сложность в случае объемных программ, трудность/невозможность автоматизации.

Стратегия «белого ящика»

Задание: провести тестирование алгоритма по приведенной схеме

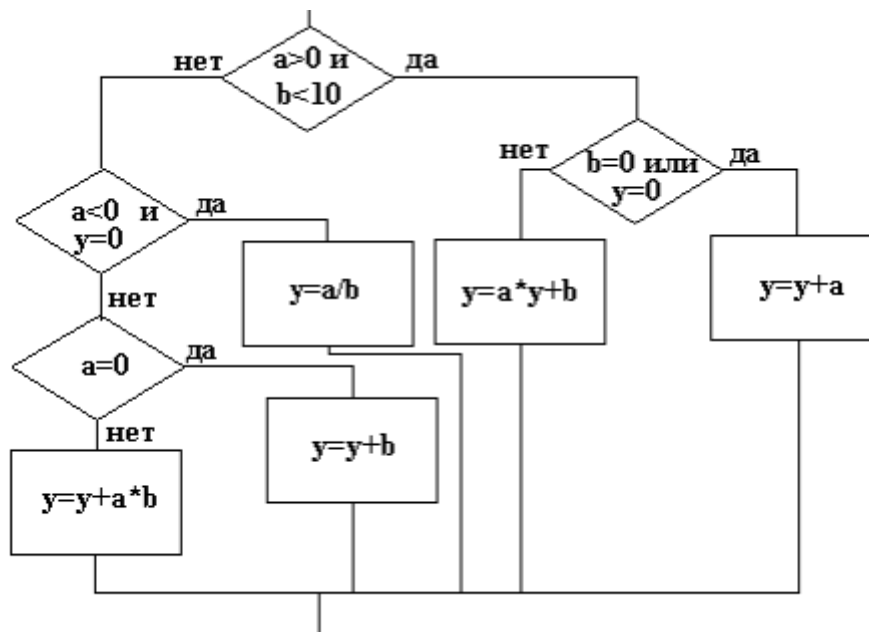


Таблица 2 — покрытие операторов

Номер теста	Назначение теста	Значения исходных данных	Ожидаемый результат
1	Покрытие $y = y + a * b$	$a = 1$ $b = 11$ $y = 2$	$y = y + a * b = 2 + 1 * 11 = 13$
2	Покрытие $y = y + b$	$a = 0$ $b = 12$ $y = 2$	$y = y + b = 2 + 12 = 14$
3	Покрытие $y = a/b$	$a = -2$ $b = 0$ $y = 0$	$y = a/b = -2/0$ → ошибка
4	Покрытие $y = a * y + b$	$a = 2$ $b = 2$ $y = 2$	$y = a * y + b = 2 * 2 + 2 = 6$
5	Покрытие $y = y + a$	$a = 2$ $b = 2$ $y = 0$	$y = y + a = 0 + 2 = 2$

Так как в данной программе операторы присутствуют во всех ветвях, таблица методы покрытия решений будет аналогична Таблице 2

Таблица 3 — покрытие решений

Номер теста	Назначение теста	Значения исходных данных	Ожидаемый результат
1	Путь нет-нет-нет	$a = 1$ $b = 11$ $y = 2$	$y = y + a * b = 2 + 1 * 11 = 13$
2	Путь нет-нет-да	$a = 0$ $b = 12$ $y = 2$	$y = y + b = 2 + 12 = 14$
3	Путь нет-да	$a = -2$ $b = 0$ $y = 0$	$y = a / b = -2 / 0$ → ошибка
4	Путь да-нет	$a = 2$ $b = 2$ $y = 2$	$y = a * y + b = 2 * 2 + 2 = 6$
5	Путь да-да	$a = 2$ $b = 2$ $y = 0$	$y = y + a = 0 + 2 = 2$

По схеме алгоритма можно выделить следующие комбинации условий:

1. $a > 0, b < 10$
2. $a > 0, b \geq 10$
3. $a \leq 0, b < 10$
4. $a \leq 0, b \geq 10$
5. $a = 0$
6. $a \neq 0$
7. $b = 0, y = 0$
8. $b \neq 0, y = 0$
9. $b = 0, y \neq 0$
10. $b \neq 0, y \neq 0$
11. $a < 0, y = 0$
12. $a \geq 0, y = 0$
13. $a < 0, y \neq 0$
14. $a \geq 0, y \neq 0$

Таблица 4 — комбинаторное покрытие условий

Номер теста	Назначение теста	Значения исходных данных	Ожидаемый результат
1	Проверка комбинаций 1, 7	$a = 1$ $b = 0$ $y = 0$	$y = y + a = 0 + 1 = 1$
2	Проверка комбинаций 1, 8	$a = 1$ $b = 1$ $y = 0$	$y = y + a = 0 + 1 = 1$
3	Проверка комбинаций 1, 9	$a = 1$ $b = 0$ $y = 1$	$y = y + a = 1 + 1 = 2$
4	Проверка комбинаций 1, 10	$a = 1$ $b = 1$ $y = 1$	$y = a * y + b = 1 * 1 + 1 = 2$
5	Проверка комбинаций 3, 5, 12	$a = 0$ $b = 5$ $y = 0$	$y = y + b = 0 + 5 = 5$
6	Проверка комбинаций 2, 6, 14	$a = 1$ $b = 11$ $y = 1$	$y = y + a * b = 1 + 1 * 11 = 12$
7	Проверка комбинаций 4, 6, 13	$a = -5$ $b = 11$ $y = 1$	$y = y + a * b = 1 - 5 * 11 = -54$
8	Проверка комбинаций 3, 11	$a = -5$ $b = 0$ $y = 0$	$y = a / b = -5 / 0$ → ошибка

Вывод:

Стратегия «белого ящика» особенно полезна в случае, когда программа содержит большое число операторов ветвления и позволяет проверить правильность построения внутренней логики программы (например, отсутствие «мертвых ветвей»).

Однако, метод белого ящика не обнаруживает:

- Пропущенных маршрутов
- Ошибок, появление которых зависит от обрабатываемых данных
- Не дает гарантии, что программа соответствует описанию

Стратегия «черного ящика»

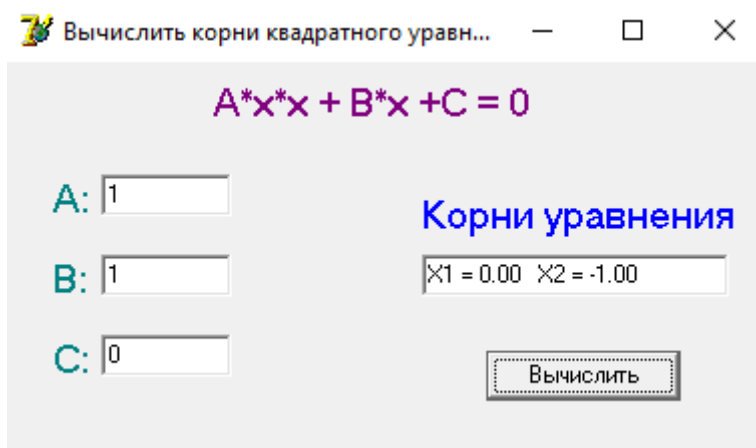


Рисунок 2 — интерфейс тестируемой программы

Описание программы:

Программа должна рассчитывать корни квадратного уравнения на основе вводимых коэффициентов (уравнение может вырождаться). В случае некорректных исходных данных необходимо выдать сообщение об ошибке.

Корни квадратного уравнения ищутся в соответствии с выражением:

$$x_{1,2} = \sqrt{(b^2 - 4*a*c)} / (2*a)$$

Исходя из этого выражения можно выделить следующие классы эквивалентности:

Неправильные классы:

1. коэффициент a содержит нечисловые символы
2. коэффициент b содержит нечисловые символы
3. коэффициент c содержит нечисловые символы

Правильные классы:

4. Коэффициенты дают дискриминант >0 (два корня) положительны
5. Коэффициенты дают дискриминант >0 (два корня) и отрицательны
6. Коэффициенты дают дискриминант >0 (два корня) и вещественны
7. Коэффициенты дают дискриминант >0 (два корня) и целые

Граничные значения:

1. есть только один корень (Коэффициенты дают дискриминант $=0$)
2. нет корней (Коэффициенты дают дискриминант <0)
3. Коэффициент $a=0$
4. Коэффициент $b=0$
5. Коэффициент $c=0$
6. Большой коэффициент a
7. Большой коэффициент b

Таблица 5 — метод эквивалентного разбиения

Номер теста	Назначение теста	Значения исходных данных	Ожидаемый результат	Реакция программы	Вывод
1	Проверка класса эк. 1	$a = ef$ $b = 1$ $c = 1$	$X1 = -1.00$ или Ошибка	Вычисления не производятся	Программа некорректно обрабатывает данный класс эк.
2	Проверка класса эк. 2	$a = 1$ $b = eb$ $c = -1$	$X1 = 1.00$ или Ошибка	Вычисления не производятся	Программа некорректно обрабатывает данный класс эк.
3	Проверка класса эк. 3	$a = 1$ $b = 1$ $c = egg$	$X1 = 0.00$ $X2 = -1.00$ или Ошибка	$X1 = 0.00$ $X2 = -1.00$	Программа корректно обрабатывает данный класс эк.
4	Проверка классов эк. 4,5,7	$a = 1$ $b = 2$ $c = -1$	$X1 = 0.41$ $X2 = -2.41$	$X1 = 0.41$ $X2 = -2.41$	Программа корректно обрабатывает данные класс эк.
5	Проверка класса эк. 6	$a = 1$ $b = 2.1$ $c = 1$	(вычисление корней)	Вычисления не производятся	Программа некорректно обрабатывает данный класс эк.

Таблица 6 — метод граничных условий

Номер теста	Назначение теста	Значения исходных данных	Ожидаемый результат	Реакция программы	Вывод
1	Проверка граничного усл. 1	$a = 1$ $b = 2$ $c = 1$	Один корень $X = -1.00$	Один корень $X = -1.00$	Программа работает корректно при данных граничных условиях
2	Проверка граничного усл. 2	$a = 1$ $b = 1$ $c = 2$	Нет решений!	Нет решений!	Программа работает корректно при данных граничных условиях
3	Проверка граничного усл. 3	$a = 0$ $b = 1$ $c = -1$	Один корень $X = 1.00$	Ошибка	Программа работает некорректно при данных граничных условиях
4	Проверка граничного усл. 4, 5	$a = 1$ $b = 0$ $c = 0$	Один корень $X = 0.00$	Один корень $X = 0.00$	Программа работает корректно при данных граничных

					условиях
6	Проверка граничного усл. 6	a = 1000 b = 1 c = 0	X1 = 0.00 X2 = -0.0001	X1 = 0.00 X2 = -0.00	Определение корректности программы требует уточнения требований точности
7	Проверка граничного усл. 7	a = 1 b = 999999999 99 c = 0	X1 = 0.00 X2 = -99999999999	(Вычисления не производятся)	Программа работает некорректно при данных граничных условиях

Анализ причинно-следственных связей

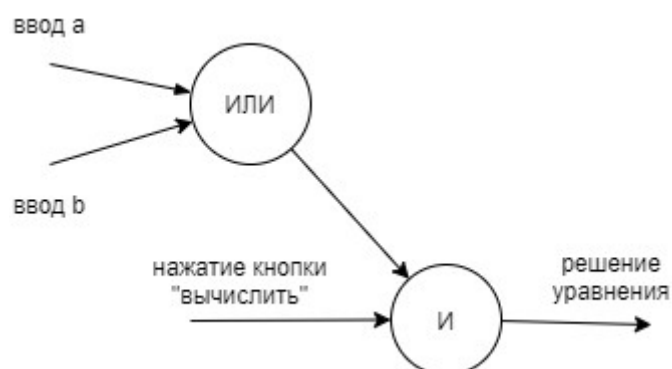


Рисунок 3 — логическая схема программы

Таблица 7 - анализ причинно-следственных связей

Номер теста	Назначение теста	Значения исходных данных			Ожидаемый результат	Реакция программы	Вывод
		Ввод a	Ввод b	Нажат ие кнопки			
1	Проверка логической структуры программы	0	0	0	Отсутствие реакции	Отсутствие реакции	Программа работает корректно
2		0	0	1	Ошибка	Отсутствие реакции	Программа работает некорректно
3		0	1	0	X=0 или ошибка или запрет расчета	Отсутствие реакции	Программа работает некорректно
4		0	1	1	Вычисление корня или ошибка или запрет расчета	Отсутствие реакции	Программа работает некорректно
5		1	0	0	X=0 или ошибка или	Отсутствие реакции	Программа работает

					запрет расчета		некорректно
6		1	0	1	Вычисление корней или ошибка или запрет расчета	Отсутствие реакции	Программа работает некорректно
7		1	1	0	Отсутствие реакции	Отсутствие реакции	Программа работает корректно
8		1	1	1	Вычисление корней	Вычисление корней	Программа работает корректно

Вывод:

Стратегия «черного ящика» позволяет проверить правильность работы программы, абстрагируясь от ее внутренней логики в условиях, максимально близким к условиям реальной эксплуатации.

Такая стратегия является наиболее «наглядной», а также позволяет не изменять тесты при изменении внутренней структуры программы в ходе ее модификации.

Недостатком такого подхода является влияние человеческого фактора при выделении граничных условий и классов эквивалентности, риск протестировать не все возможные условия работы программы.

Общий вывод по лабораторной работе:

В ходе данной лабораторной работы были изучены основные стратегии и методы тестирования программ, а также их достоинства и недостатки, оптимальные условия применения.