*Федеральное государственное бюджетное образовательное учреждение* *высшего образования*

|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | ***«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)

**Отчет**

**по лабораторной работе № 3**

**Дисциплина:** Технологии разработки программных систем

**Название лабораторной работы:**

Тестирование программного обеспечения

**Вариант 5**

Студент гр. ИУ6-42  **\_\_\_\_\_\_\_\_\_\_\_\_\_ А.С. Бурлаков**

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О. Фамилия)

Москва, 2018

**ВВЕДЕНИЕ**

## 1. Цель работы

## - Знакомство с существующими стратегиями тестирования, приобретение навыков выбора стратегии и разработки тестов для отдельных задач, сравнение и оценка различных методов тестирования и их возможностей.

## ОСНОВНАЯ ЧАСТЬ

1. **Структурный контроль**

Для своего варианта задания выполните **структурный контроль**, используя перечень вопросов в Приложении. В процессе выполнения заполните таблицу. Сделайте общий вывод о роли структурного контроля в процессе создания программы. Сформулируйте его достоинства и недостатки.

Вариант 4. Программа должна генерировать массив чисел и сортировать данные методом вставки.

# program v4;

# const

# N = 10;

# type

# tm = array [1..N] of integer;

# var

# m: tm;

# i, j, k, q, b: integer;

# begin

# { TODO -oUser -cConsole Main : Insert code here }

# randomize;

# for i := 1 to N do begin

# m[i] := random(100);

# write(' ', m[i])

# end;

# writeln;

# k := 3;

# for i := k to N do begin

# for j := 1 to i - 1 do begin

# if (m[i] <= m[j]) then begin

# b := m[i];

# for q := i to j do

# m[q] := m[q - 1];

# m[j] := b;

# end;

# end;

# k := k + 1;

# end;

# for i := 1 to N do begin

# write(' ', m[i])

# end;

# end.

|  |  |  |  |
| --- | --- | --- | --- |
| Номер вопроса | Строки, которые надо проверить | Результат проверки | Вывод |
| 1.1 | 17,21,25 | k:=3;  b:=m[i];  m[i]:=random(100); | В программе нет лишних объявлений |
| 1.2 | 7,10,16,22 | tm = array [1..N] of integer;m: tm;for i := 1 to N do begin for i := k to N do beginfor j := 1 to i - 1 do begin | Массив m имеет фиксированный размер N=10, в программе размер не превышается. |
| 1.3 | - | - | Матрицы не используются |
| 1.4 | 9-11 | varm: tm;i, j, k, q, b: integer; | В программе не присутствуют переменные со сходными/одинаковыми именами |
| 1.5 | - | - | Файлы не используются |
| 1.6 | - | - | Нетипизированные переменные, открытые массивы, динамическая память не используются |
| 2.1 | Вся программа | - | Порядок следования операторов верен |
| 2.2 | Вся программа | - | Вычислений неарифметических параметров не производится |
| 2.3 | - | - | Вычисления с переменными различных типов не используются |
| 2.4 | - | - | Переполнение разрядной сетки или ситуация машинного нуля невозможны |
| 2.5 | Вся программа | - | В программе отсутствуют вещественные переменные |
| 2.6 | Вся программа | В программе только один тип переменных | Сравнения переменных различных типов нет |
| 3.1 | 26 | **for** q := i **to** j **do** | Неправильный порядок следования операторов.  Правильный:  **for** q := j down**to** i **do** |
| 3.2 | Вся программа | - | Программа будет завершена |
| 3.3 | 21, 22 | k := 3;  **for** i := k **to** N **do begin** | Неправильное входное условия для цикла, правильно: k := 2; |
| 3.4 | Вся программа | - | Поисковые циклы не используются |

**Вывод**: Структурный контроль позволяет избавиться от ошибок, которые могут привести к неправильной работе программы или ошибке выполнения. Он позволяет учесть большое количество ошибок, которые описаны в вопросах и в таблице.

Достоинства:

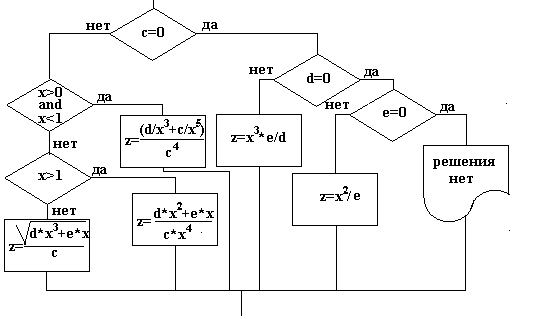
* Четкие вопросы по программе (Тестирование);
* Простота проверки;
* После удаления ошибок программа практически всегда будет работать безотказно.

Недостатки:

* На проверку тратится много времени;
* Вопросы для проверки не учитывают ошибок, связанных с работой алгоритма.

**Тестирование по принципу «белого ящика».**

Вариант 8.



Используя комбинаторное покрытие условий, можно выделить следующие комбинации:

1. c = 0
2. c != 0
3. x > 0 and x < 1
4. x <= 0 and x < 1 = x <= 0
5. x >= 1 and x > 0 = x >= 1
6. x <= 0 and x >= 1 – невозможная комбинация
7. x > 1
8. x <= 1
9. d = 0

10) d != 0

11) e = 0

12) e != 0

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Покрытие комбинаций | Значения исходных данных | Ожидаемый результат |
| 1 | 2, 5, 8 | x=1,c=1,d=1,e=1 | sqrt(2) |
| 2 | 2, 5, 7 | x=2,c=1,d=1,e=1 | 3/8 |
| 3 | 2, 3 | x=0.5,c=1,d=1,e=1 | 5/32 |
| 4 | 2, 4, 8 | x=0,c=1,d=1,e=1 | 0 |
| 5 | 1, 10 | x=1,c=0,d=1,e=1 | 1 |
| 6 | 1, 9, 12 | x=1,c=0,d=0,e=1 | 1 |
| 7 | 1, 9, 11 | x=1,c=0,d=0,e=0 | Решения нет |

**Вывод**: стратегия «Белого ящика» используется для устранения ошибок в логике программы. Самый лучший метод этой стратегии – комбинаторное покрытие условий, так как он позволяет проверить все возможные комбинации.

Достоинства:

* Проверяет все варианты решений
* Алгоритм рассматривается детально

Недостатки:

* Для проверки всех путей и условий уходит много времени

**Тестирование по принципу «Черного ящика»**

*Задача 7.*

Программа должна вычислять с заданной точностью значение интеграла функции  на введенном с клавиатуры интервале от a до b.

Т.е. это получается

Для eps выделим следующие классы эквивалентности входных условий:

1. Число
   1. <= 0
   2. > 0
2. Другое

Для параметров a и b выделим следующие классы эквивалентности:

1. Число
2. Другое

В методе причинно-следственных связей выделим причину «B>A» и следствие «Результат > 0», при b>a, рез >0 и при b <= a, рез <= 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Назначение теста | Значения исходных данных | Ожидаемый результат | Реакция программы | Вывод |
| 1 | Эквивалентное разбиение | A=0, b=1, eps=0.1 | 0,3 | 0,3 | Верно |
| 2 | Эквивалентное разбиение | A=’a’, b=1,eps=0,1 | Ошибка ввода | Ошибка данных! | Верно |
| 3 | Эквивалентное разбиение | A=1, b=’a’,eps=0,1 | Ошибка ввода | -0,3 | Неверно |
| 4 | Эквивалентное разбиение | A=1, b=1,eps=’a’ | Ошибка ввода | Ошибка данных! | Верно |
| 5 | Эквивалентное разбиение | A=1, b=1, eps=0 | Ошибка ввода | Floating point division by zero | Верно |
| 6 | Граничные значения | A=1, b=2,eps=1e-10 | 2,333 | 2,333 | Верно |
| 7 | Граничные значения | A=1, b=2,  eps=-1e-10 | Ошибка ввода | 2,333 | Неверно |
| 8 | Причинно-следственные связи | A=0, b=3, eps=0,1 | 9 | 9 | Верно |
| 9 | Причинно-следственные связи | A=3, b=0, eps=0,1 | -9 | -9 | Верно |
| 10 | Причинно-следственные связи | A=-3, b=3, eps=0,1 | 18 | 0 | Неверно |
| 11 | Причинно-следственные связи | A=-3, b=0, eps=0,1 | 9 | -9 | Неверно |
| 12 | Предположение об ошибке | A=’’, b=’’, eps=’’ | Ошибка ввода | Ошибка данных! | Верно |

**Вывод**

Тестирование по принципу черного ящика позволяет распознать ошибки в работе алгоритма. Однако для обнаружения большего количества ошибок в программе необходимо использовать тестирование на всех исходных данных.

**ЗАКЛЮЧЕНИЕ**

Для правильной работы приложения необходимо проводить тестирование на этапах:

* Разработки – Структурный контроль
* Написания кода – Стратегия белого ящика
* Выпуск приложения – Стратегия черного ящика