ГОСУДАРСТВЕННОЕ УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«ИНСТИТУТ БИЗНЕСА БЕЛОРУССКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА»

КАФЕДРА ЦИФРОВЫХ СИСТЕМ И ТЕХНОЛОГИЙ

ОТЧЕТ
о выполнении Лабораторной работы №19
«Разработка тестовых сценариев для проверки функционирования приложения»
по дисциплине «Проектирование информационных систем»

Севрюк Александры Петровны
студентки 4 курса, группа 852
специальность «Управление информационными ресурсами»

Минск, 2021

Оглавление

1. Цель работы

Целью данной лабораторной работы является изучение методологии построения тестовых сценариев, а также приобретение практических навыков разработки тестовых сценариев.

2. Файл с разработанными тест кейсами

Были сформированы Smoke-тесты для разрабатываемой системы.

№	Тест-кейс	Алгоритм выполнения	Ожидаемый результат	Результат неудачи
1	Аутентификация в системе (личном кабинете)	1. Открыть систему 2. Заполнить форму с логином и паролем 3. Нажать кнопку «войти»	Открывается личный кабинет	Отказано в аутентификации, ничего не происходит
2	Открытие тестирования	1. Войти в раздел с тестированием 2. Выбрать нужное тестирование 3. Нажать на кнопку «приступить к тестированию»	Открывается тестирование, на экране появляется первое задание тестирования	Тестирование не открывается, ничего не происходит
3	Выполнение задания	1. Ознакомиться с заданием 2. Изучить варианты ответов 3. Выбрать и кликнуть на нужный вариант ответа 4. Нажать кнопку «перейти к следующему заданию»	Выбранный ответ помечается галочкой, выполняется переход к следующему заданию, данное задание отмечается выполненным	Нет возможности выбрать ответ, не осуществляется переход к следующему заданию, данное задание не отмечается выполненным
4	Завершение выполнения тестирования	1. Нажать кнопку «Завершить тестирование» 2. Подтвердить желание завершить тестирование нажатием кнопки «да, я	Появляется окно с текстом «Тестирование завершено», временем, затраченным на	Ничего не происходит, тестирование не закрывается, не появляется окно с оповещением

		уверен, что хочу завершить тестирование»	тестирование и результатом	о завершении тестирования и результатами
5	Просмотр результатов тестирования	1. Войти в раздел с результатами тестирования 2. Выбрать нужное тестирование 3. Нажать на кнопку «просмотреть результаты»	Открывается страница с результатами выбранного тестирования	Страница с результатами тестирования не открывается, ничего не происходит, информация отображается некорректно

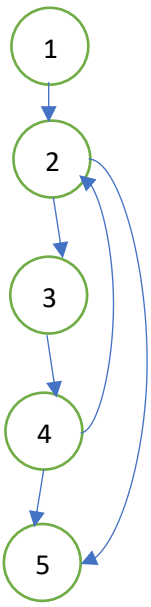
3. Контрольные задания

Задание 1. (Справка для выполнения задания [1] стр. 445-454)

Дан программный код

```
main()
{
int zeich = 'x';
while(zeich != '#')
{
printf(" Продолжить ввод ");
zeich = getchar();
}
printf("Ввод завершен ");
}
```

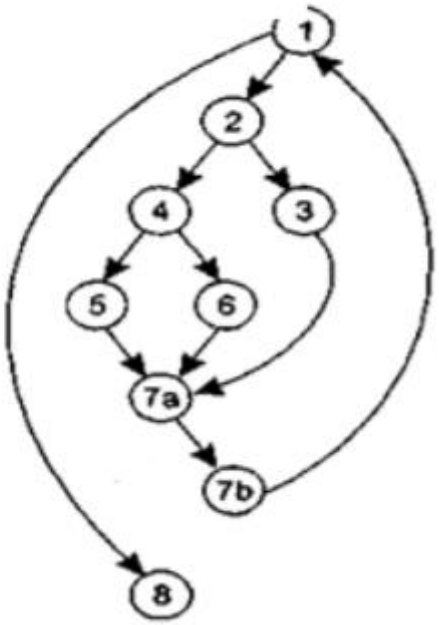
Построить потоковый граф. Определить цикломатическую сложность.



Количество регионов = 3 Цикломатическая сложность = 3

Задание 2. (Справка для выполнения задания [1] стр. 445-454)

Определить количество тестовых вариантов для программы, описанной потоковым графом, изображенным на рисунке.



Количество регионов = 4

Цикломатическая сложность = 4

Независимые пути = 4

Независимые пути для потокового графа:

Путь 1: 1-8.

Путь 2: 1-2-3-7a-7b-1-8.

Путь 3: 1-2-4-5-7a-7b-1-8.

Путь 4: 1-2-4-6-7a-7b-1-8.

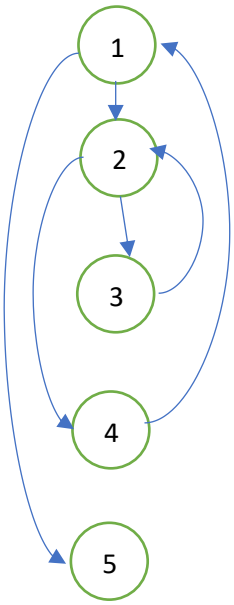
Количество тестовых вариантов для программы = количество независимых путей = 4.

Задание 3. (Справка для выполнения задания [1] стр. 445-454)

По коду программы построить потоковый граф и определить количество тестовых вариантов.

```
double s1(double x,int n)
{
double s=0;
```

```
int i,f,k1;
for(k1=0; k1<=n; k1++)
{
f=1;
for(i=1; i<=(2*k1);
i++)
{
f*=i;
}
s+=(double)pow(x,(double)2*k1)/f;
}
return s;
}
```



Независимые пути для потокового графа:

Путь 1: 1-5.

Путь 2: 1-2-4-1-5.

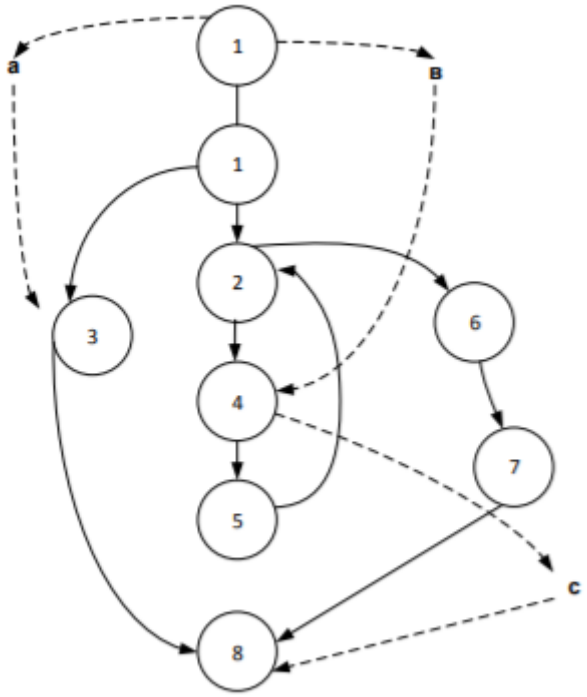
Путь 3: 1-2-3-2-4-1-5.

Количество тестовых вариантов для программы = количество независимых путей = 3.

Задание 4. (Справка для выполнения задания [1] стр. 462-467)

Дан управляющий граф программы (с указанием используемых переменных).

Построить DU цепочки и определить количество тестовых вариантов для проверки данных программы.



DU цепочки: [a,1,3], [в,1,4],[c,4,8].

Независимые пути для потокового графа:

Путь 1: 1-3-8.

Путь 2: 1-2-4-5.

Путь 3: 1-2-6-7-8.

Количество тестовых вариантов для программы = количество независимых путей = 3.

Задание 5. (Справка для выполнения задания [1] стр. 472-480)

Программа должна работать с исходными данными, удовлетворяющими следующим характеристикам:

- целое,
- положительное,
- трехзначное.

Определить классы эквивалентности и граничные условия. Построить тестовые варианты для проверки ввода данных с использованием способов разбиения по эквивалентности и анализа граничных значений. При правильном вводе выдается значение вводимого числа, при неправильном вводе информационное сообщение об ошибке.

Классы эквивалентности

V_Class1 = {100...999 mod 1= 0} - допустимый класс эквивалентности

Inv_Class1 = {x < 99} – первый недопустимый класс эквивалентности

Inv_Class2 = {x > 1000} – второй недопустимый класс эквивалентности

Inv_Class3 = {x mod 1 != 0} – третий недопустимый класс эквивалентности

Тестовые варианты:

-100; -1; 99; 100; 99,9; 999; 999,1; 1000; 1000,1.

4. Ответы на контрольные вопросы

1. Задачи функционального тестирования.

Функциональное тестирование (functional testing) – процесс проверки программного обеспечения, сконцентрированный на анализе соответствия ПО требованиям и спецификациям. Основной задачей функционального тестирования является подтверждение того, что разрабатываемый программный продукт обладает всем функционалом, требуемым заказчиком.

2. Как строится потоковый граф? Приведите пример.

1. Граф строится отображением управляющей структуры программы. В ходе отображения закрывающие скобки условных операторов и операторов циклов (end if, end loop) рассматриваются как отдельные (фиктивные) операторы.

2. Узлы (вершины) потокового графа соответствуют линейным участкам программы, включают один или несколько операторов программы.

3. Дуги потокового графа отображают поток управления в программе (передачи управления между операторами) Дуга – это ориентированное ребро.

4. Различают операторные и предикатные узлы. Из операторного узла выходит одна дуга, а из предикатного - две дуги.

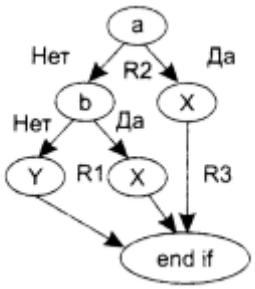
5. Предикатные узлы соответствуют простым условиям в программе. Составное условие программы отображается в несколько предикатных узлов. Составным называют условие, в котором используется одна или несколько булевых операций (OR, AND).

6. Замкнутые области, образованные дугами и узлами, называют регионами.

7. Окружающая граф среда рассматривается как дополнительный регион.

Пример:

```
if a OR b
        then x
        else y
end if;
```



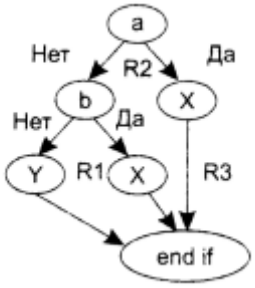
3. Как по потоковому графу определить количество тестовых вариантов. Приведите пример.

Суть метода состоит в том, чтобы представить программу в виде графа, узлами которого являются отдельные операторы или их линейные последовательности. Выполнение тестового варианта соответствует отдельному пути в таком графе (называемом потоковым, т.к. путь в графе реализует поток управления при выполнении программы). Набор тестов строится таким образом, чтобы проверить базовое множество путей (маршрутов) в программе. Это гарантирует однократное выполнение каждого оператора программы при тестировании. Способ тестирования базового пути даёт возможность получить оценку комплексной сложности программы и использовать ее для определения необходимого **количества тестовых вариантов**. Тестовые варианты разрабатываются для проверки базового множества путей в программе. Они обеспечивают однократное выполнение каждого оператора программы при тестировании. Реальные результаты каждого тестового варианта сравниваются с ожидаемыми результатами. После выполнения всех тестовых вариантов гарантируется, что все операторы программы выполнены, по меньшей мере, один раз.

Независимый путь – это любой путь, в который входит новый оператор обработки или новое условие, т.е. независимый путь должен содержать дугу, не входящую в ранее определенные пути.

Пример: три независимых пути, соответственно три тестовых варианта.

```
if a OR b
        then x
        else y
end if;
```



4. Построить тестовые варианты для тестирования ввода данных с использованием классов эквивалентности и анализа граничных значений, если в спецификациях указано, что входные параметры – целые числа в диапазоне 0-100.

Классы эквивалентности

V_Class1 = {0...100 mod 1= 0} - допустимый класс эквивалентности

Inv_Class1 = {x < 0} – первый недопустимый класс эквивалентности

Inv_Class2 = {x > 100} – второй недопустимый класс эквивалентности

Inv_Class3 = {x mod 1 != 0} – третий недопустимый класс эквивалентности

Тестовые варианты:

-1; -0,99; 0; 50; 100; 100,1; 200.

5. Зачем нужно строить тест кейсы?

Тест-кейс (test case) — набор входных данных, условий выполнения и ожидаемых результатов, разработанный с целью проверки того или иного свойства или поведения программного средства.

5. Выводы

В результате выполнения данной лабораторной работы я изучила методологии построения тестовых сценариев, а также приобрела практических навыков разработки тестовых сценариев.