МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «Полоцкий государственный университет»

Факультет информационных технологий

Кафедра технологий программирования

**ЛАБОРАТОРНАЯ РАБОТА № 12**

По Дисциплине: **«**Надежность программного обеспечения**»**

На Тему: «Модульное тестирование»

База данных «Ж/Д Вокзал»

ВЫПОЛНИЛ студент группы 15-КБ

Осипенко В.В.

ПРОВЕРИЛА преподаватель

Попкова Д.В.

Полоцк 2017 г.

**Цель работы:** ознакомиться с основами модульного, объектно-ориентированного и пошагового тестирования. Научиться принципам тестирования структуры программных модулей.

**Краткие теоретические сведения**

Модульное тестирование – вид тестовой деятельности, при которой проверке подвергаются внутренние рабочие части программы, элементы или модули независимо от способа их вызова. Под модулем принято понимать программу или ограниченную часть кода с одной точкой входа и одной точкой выхода, которая выполняет одну и только одну первичную функцию.

Этот тип тестирования, как правило, осуществляется программистом, а не тестировщиком, поскольку для его проведения необходимы доскональные знания внутренней структуры и кода программы. Такое тестирование не всегда выполняется просто, например, в случае, если прикладная программа не имеет четко определенной архитектуры с компактным кодом, и для его проведения может понадобиться разработка модулей тестовых драйверов или средств тестирования.

Тестировать свои собственные продукты – это весьма трудное занятие для разработчиков, поскольку они вынуждены изменить свою точку зрения или отношение, перейдя от роли создателя в роль критика. Многие разработчики на самом деле не любят тщательно тестировать свои программные продукты, так как считают скучным и даже угнетающим наблюдать за тем, как прикладная программа справляется с возложенными на нее нагрузками. Другие не имеют ничего против подобного подхода и прекрасно выполняют такого рода работу. Такой подход, несомненно, приводит к обнаружению ошибок на более ранних этапах разработки программного продукта, а это в свою очередь способствует снижению стоимости ошибки. Некоторые исследования показывают, что плотность распределения дефектов находится в диапазоне от 49,5 до 94,6 на тысячу строк кода. Поэтому каждый разработчик должен самостоятельно тестировать свою программу с максимальной тщательностью, прежде чем передать рабочий продукт независимому испытателю.

**Краткое описание программы из варианта задания.**

Назначение этого документа состоит в детализации процедур тестирования, которые должны аттестовать функциональные возможности базы данных «Ж/Д Вокзал».

**1. Отчет обнаружения ошибок.**

Программа состоит из одной части. Файл main.cpp содержит в себе информации об выполняемых функция программы, осуществляет их вызов, а также реализует интерфейс пользователя.

В начальной версии программы было обнаружено 3 ошибки, которые в последствии были исправлены разработчиком. После исправления было пройдено 100% тестов, причем ошибок обнаружено не было. Используя полученные результаты, можно утвердить выпуск этого программного продукта. Отчет обнаружения ошибок представлен в таблице 1.

**Таблица 1** –Отчет обнаружения ошибок

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ Ошибки** | **Название модуля/функции** | **Описание ошибки** | **Важность**  **ошибки**  **(высокая, средняя, низкая)** | **Ошибка исправлена**  **Да/Нет** |
| 1 | update(List[]) | Не корректное обновления списка | Средняя | Да |
| 2 | Delete\_item() | Удаляемая строка не удаляется из файла | Средняя | Да |
| 3 | Save\_item(String) | Добавляемая строка не сохраняется в файле | Средняя | Да |

Метрический показатель сложности или цикломатическое число G потокового графа определяется по формуле: **G=R-V+2**, где **R** – количество ребер графа, **V** – количество вершин графа.

b

c

d

a

Рисунок 1 — update(List[])

Здесь:

a. Получение списка

b. Проверка на наличие следующего элемента

c. Вывод элемента

d. Отрисовка

a

b

c

d

Рисунок 2 — Save\_item(String)

Здесь:

a. Создание элемента

b. Добавление в список

c. Сохранение списка

d. Вывод списка

Для графа №1: G = R – V + 2 = 4 – 4 + 2 = 2.

Для графа№2: G = R – V + 2 = 3 – 4 + 2 = 1.

Тестовые сценарии для каждого графа представить в виде таблицы. Пример сценария приведен в таблице 2.

**Таблица 2** – Модули update(List[] list) и Save-item(String dir)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **G** | **№ сценария** | **Описание прохода** | **Контрольные примеры, позволяющие реализовать описанную ситуацию** | **Тест пройден**  **Да/Нет** |
| G=2 | 1 | a-b-c-b-d | List[] list = {}; | Да |
| 2 | a-b-d | List[] list = {el element1, el element2}; | Да |
| G=1 | 1 | a-b-c-d | Save\_item(“C:\Train”); | Да |

Схема взаимодействия всех модулей программы изображена на рисунке 1.

save

Main

delete

add

update

Read\_from\_file

Рисунок 3. - Схема взаимодействия всех модулей программы

Взаимодействие модулей приведено в таблице 3.

**Таблица 3** – Взаимодействие модулей

|  |  |  |  |
| --- | --- | --- | --- |
| **№ в последо-ватель-ности** | **Описание последовательности** | **Контрольные примеры, позволяющие реализовать описанную ситуацию** | **Тест пройден**  **Да/Нет** |
| 1 | Main -> delete | Выбрать «Удалить» в меню действий | Да |
| 2 | Main -> add | Выбрать «Добавить» в меню действий | Да |
| 3 | Main -> Read\_from\_file | Выбрать «Открыть Б/Д» в меню действий | Дa |
| 4 | delete -> save | Удалить обьект | Да |
| 5 | add -> save | Добавить обьект | Да |
| 6 | Main->update |  | Да |
| 7 | update -> Read\_from\_file |  | Да |

**Вывод:** в ходе выполнения данной лабораторной работы была изучена теоретическая часть и принципы тестирования структуры программных модулей. Проведён обзор разработанного программного кода. Составлен отчет обнаружения ошибок для базы данных «Ж/Д вокзал». А также разработаны тестовые сценарии для модульного тестирования.