МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра защиты информации

**

**ОТЧЁТ**

**по лабораторной работе №3**

**«**Объектно-ориентированное программирование**»**

**по дисциплине: «***Программирование***»**

Выполнил:Проверил:

Студент гр. «АБс-324», «АВТФ» *ассистент кафедры ЗИ*

*Деревянкин Д.А. Исаев Г. А.*

«\_\_» \_\_\_\_\_\_ 2024 г.«\_\_\_» \_\_\_\_\_\_ 2024 г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (подпись)

Новосибирск 2024

**Цели и задачи работы:** изучение основных принципов объектно- ориентированного программирования и основ юнит-тестирования.

**Задание к работе:** Самостоятельно решить задачи в соответствии с индивидуальным вариантом.

**Задания:**

1. Реализовать классы с базовым набором операций (private, public) на основе лабораторной работы 1 на C++. Все структуры.
2. Реализовать класс с базовым набором операций (private, public) на основе лабораторной работы 1 на Go. Хеш-таблица.
3. Реализовать покрытие тестами (не менее 90%) в проектах 1 и 2. Создать HTML-отчет о покрытии, который предоставляет визуальный анализ того, как много кода было протестировано.
4. Выполнить сериализацию и десериализацию данных бинарного и текстового форматов для проектов 1 и 2.
5. UML – диаграммы.
6. Теоретическая часть: Анализ российских стандартов оценки качества ПО. Основные виды тестирования.

# UML – Диаграммы

## **Массив:**

|  |
| --- |
| Array<T> |
| - capacity: size\_t  - size: size\_t  - data: T\* |
| + Array()  + ~Array()  + void resize()  + void add(T value)  + void addAt(size\_t index, T value)  + T get(size\_t index) const  + void remove(size\_t index)  + void replace(size\_t index, T value)  + size\_t length() const  + void print() const  + void serializeText(const string& filename) const  + void deserializeText(const string& filename)  + void serializeBinary(const string& filename) const  + void deserializeBinary(const string& filename) |

## **Списки:**

|  |
| --- |
| SinglyLinkedList<T> |
| - head: Node\*  - size: int |
| + SinglyLinkedList()  + ~SinglyLinkedList()  + void push\_front(T value)  + void push\_back(T value)  + void pop\_front()  + void pop\_back()  + void remove(T value)  + bool find(T value) const  + void print() const  + int get\_size() const  + void serializeText(const string& filename) const  + void deserializeText(const string& filename)  + void serializeBinary(const string& filename) const  + void deserializeBinary(const string& filename) |
| <<struct>> Node  - data: T  - next: Node\*  + Node(T value) |

|  |
| --- |
| DoublyLinkedList<T> |
| - head: Node\*  - tail: Node\*  - size: int |
| + DoublyLinkedList()  + ~DoublyLinkedList()  + void push\_front(T value)  + void push\_back(T value)  + void pop\_front()  + void pop\_back()  + void remove(T value)  + bool find(T value) const  + void print() const  + int get\_size() const  + void serializeText(const string& filename) const  + void deserializeText(const string& filename)  + void serializeBinary(const string& filename) const  + void deserializeBinary(const string& filename) |
| <<struct>> Node  - data: T  - next: Node\*  - prev: Node\*  + Node(T value) |

## **Стек:**

|  |
| --- |
| Stack<T> |
| - data: T\*  - top: int  - capacity: int |
| + Stack(int size)  + ~Stack()  + void push(T value)  + void pop()  + T peek()  + bool isEmpty()  + int size()  + void serializeText(const string& filename)  + void deserializeText(const string& filename)  + void serializeBinary(const string& filename)  + void deserializeBinary(const string& filename) |

## **Очередь:**

|  |
| --- |
| Queue<T> |
| - data: T\*  - front: int  - rear: int  - capacity: int  - size: int |
| + Queue(int cap)  + ~Queue()  + void push(T value)  + void pop()  + T peek()  + bool isEmpty()  + int Size()  + void serializeText(const string& filename)  + void deserializeText(const string& filename)  + void serializeBinary(const string& filename)  + void deserializeBinary(const string& filename) |

## **Хеш-таблица:**

|  |
| --- |
| HashTable<T> |
| - TABLE\_SIZE: const int = 100  - table: KeyValuePair\* [TABLE\_SIZE]  - sizetable: int |
| + HashTable()  + ~HashTable()  + void insert(const string& key, T value)  + bool get(const string& key, T& value)  + bool remove(const string& key)  + void serializeText(const string& filename)  + void deserializeText(const string& filename)  + void serializeBinary(const string& filename)  + void deserializeBinary(const string& filename)  - int hashFunction(const string& key) |
| <<struct>> KeyValuePair  - key: string  - value: T  - next: KeyValuePair\* |

## **Дерево:**

|  |
| --- |
| BinaryTree |
| - root: Node\*  - size: int |
| + BinaryTree()  + ~BinaryTree()  + void insert(int value)  + bool search(int value)  + bool isComplete()  + void print()  + void serializeText(const string& filename)  + void deserializeText(const string& filename)  + void serializeBinary(const string& filename)  + void deserializeBinary(const string& filename) |
| <<struct>> Node  - data: int  - left: Node\*  - right: Node\*  + Node(int value)  - Node\* insertRec(Node\* node, int value)  - bool searchRec(Node\* node, int value)  - int countNodes(Node\* node)  - bool isCompleteRec(Node\* node, int index, int numberNodes)  - void printTreeRec(Node\* node, int depth)  - void destroyTree(Node\* node)  - void serializeRec(Node\* node, ofstream& ofs)  - Node\* deserializeRec(ifstream& ifs)  - void serializeBinaryRec(Node\* node, ofstream& ofs)  - Node\* deserializeBinaryRec(ifstream& ifs) |

# Теоретическая часть

**Анализ российских стандартов оценки качества ПО**

**1.1. Общие сведения о стандартах**

В России существует несколько стандартов, регламентирующих оценку качества ПО. Основными из них являются:

* *ГОСТ Р ИСО/МЭК 25010-2011* – "Системы и программное обеспечение. Модели качества". Этот стандарт определяет характеристики качества ПО и их подкатегории, такие как функциональность, надежность, удобство использования, эффективность, поддерживаемость и переносимость.
* *ГОСТ Р 51245-99* – "Программное обеспечение. Оценка качества". Данный стандарт устанавливает общие требования к процессу оценки качества ПО, включая методы тестирования и критерии оценки.
* *ГОСТ Р ИСО/МЭК 25012-2012* – "Системы и программное обеспечение. Качество информации". Этот стандарт акцентирует внимание на качестве информации, обрабатываемой программными системами.

**1.2. Основные характеристики качества ПО**

Согласно ГОСТ Р ИСО/МЭК 25010-2011, качество ПО можно оценивать по следующим характеристикам:

* Функциональность – соответствует ли ПО заявленным требованиям и выполняет ли оно необходимые функции.
* Надежность – способность ПО выполнять заданные функции в определенных условиях на протяжении заданного времени.
* Удобство использования – легкость взаимодействия пользователя с ПО.
* Эффективность – использование ресурсов (времени, памяти) в процессе выполнения программой своих функций.
* Поддерживаемость – легкость внесения изменений и исправлений в ПО.
* Переносимость – возможность использования ПО в различных средах и на различных платформах.

**1.3. Процесс оценки качества ПО**

Процесс оценки качества ПО включает в себя несколько этапов:

1. Определение требований – формулирование требований к качеству ПО на основе стандартов и потребностей заказчика.
2. Планирование оценки – разработка плана оценки, в который входят методы тестирования и критерии оценки.
3. Проведение тестирования – выполнение тестов, направленных на проверку соответствия ПО установленным требованиям.
4. Анализ результатов – обработка данных, полученных в ходе тестирования, и формирование заключения о качестве ПО.
5. Документирование – составление отчетов о проведенной оценке и выявленных недостатках.

**2. Основные виды тестирования**

Тестирование ПО – это процесс, направленный на выявление дефектов и проверку соответствия ПО установленным требованиям. Существует множество видов тестирования, каждый из которых имеет свои цели и задачи.

**2.1. Функциональное тестирование**

Функциональное тестирование направлено на проверку функциональности ПО в соответствии с его требованиями. Оно включает в себя:

* Тестирование требований – проверка выполнения всех заявленных требований к функциональности.
* Тестирование пользовательского интерфейса – оценка удобства и интуитивности интерфейса для пользователя.

**2.2. Нефункциональное тестирование**

Нефункциональное тестирование оценивает характеристики, не связанные с функциональностью, такие как производительность, безопасность и удобство использования. Основные виды:

* Тестирование производительности – оценка времени отклика, пропускной способности и устойчивости системы под нагрузкой.
* Тестирование безопасности – проверка на наличие уязвимостей и соответствие стандартам безопасности.

**2.3. Регрессионное тестирование**

Регрессионное тестирование проводится после внесения изменений в ПО (например, исправления ошибок или добавления новых функций) с целью проверки, не повлияли ли эти изменения на существующую функциональность.

**2.4. Интеграционное тестирование**

Интеграционное тестирование проверяет взаимодействие между различными компонентами системы. Оно может включать как "черный ящик" (тестирование без знания внутренней структуры) так и "белый ящик" (тестирование с учетом внутренней структуры).

**2.5. Системное тестирование**

Системное тестирование охватывает все компоненты системы в целом и направлено на проверку соответствия системы установленным требованиям и спецификациям.

Качество программного обеспечения играет важную роль в его успешности на рынке. Российские стандарты оценки качества ПО предоставляют четкие критерии и методы, которые помогают организациям оценивать и улучшать свои продукты. Основные виды тестирования, такие как функциональное, нефункциональное, регрессионное, интеграционное и системное тестирование, обеспечивают всестороннюю проверку ПО и способствуют выявлению дефектов на различных этапах разработки.

**Источники**

1. ГОСТ Р ИСО/МЭК 25010-2011. Системы и программное обеспечение. Модели качества. - М.: Стандартинформ, 2012.
2. ГОСТ Р 51245-99. Программное обеспечение. Оценка качества. - М.: Государственный комитет Российской Федерации по стандартизации и метрологии, 1999.
3. ГОСТ Р ИСО/МЭК 25012-2012. Системы и программное обеспечение. Качество информации. - М.: Стандартинформ, 2013.
4. Бенч, М. А. Основы тестирования программного обеспечения. - М.: Издательство "Наука", 2018. - 320 с.
5. Кан, С. Управление качеством программного обеспечения. - М.: Издательство "Эксмо", 2016. - 416 с.

**Вывод**

Во время выполнения лабораторной работы №3 я познакомился с понятием Unit-тестирования, научился пользоваться библиотекой Boost test и самостоятельно реализовал свои тесты.

Ссылка на репозиторий: [нажмите](https://github.com/WarpMiner/Lab3)