Лабораторная работа №3
Объектно-ориентированное программирование

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.

Методика выполнения работы:

1. Разработать алгоритмы решения задачи по индивидуальному заданию.
2. Написать и отладить программы решения задачи (C++, Go или Rust).
3. Протестировать работу программ на различных исходных данных.
4. По запросу преподавателя быть готовым модифицировать/добавить алгоритмы/блоки кода в контексте ООП.
5. Ответить на теоретические вопросы к лабораторной работе на выбор преподавателя (не менее двух вопросов).

Перечень вопросов к защите лабораторной работы 3.

- Классы. Назначение секций класса. Конструкторы, списки инициализации.
- Деструкторы и время жизни объектов класса. Константность методов.
- Классы. Порядок конструирования объектов класса. Шаблоны классов
- Хэш функции. Понятие. Свойства хэш функций. Требования к хэш функциям. Понятие коллизии.
- Элементарные способы построения хэш-функций. Классификация хэш-функций. Современные криптографические хэш-функции.
- Применение хэш-функций в криптографии.
- Основные принципы создания объектной модели (абстрагирование, инкапсуляция, модульность, иерархия, типизация, параллелизм, сохраняемость).
- Экземпляры класса. Уровни доступа к членам класса. Классификация методов объекта.
- Средства UML для создания абстракций. Вид нотации класса, взаимодействия классов. Представление иерархических отношений.
- Понятие интерфейса. Интерфейс и абстрактный класс: сравнение.
- Наследование.
- Полиморфизм. Понятие виртуальных методов. Sizeof классов при наследовании (обоснование).
- Понятие таблицы виртуальны функций.
- Сериализация и десериализация данных. Область применения. Форматы сериализации данных.
- Понятие тестирования. Основные цели тестирования.
- Стандарты тестирования, основные показатели.
- Поддерживаемость программы на основе ISO 25010 и ее составные элементы.
- Понятие и характеристика ISTQB.
- Основные виды тестирования.
- Основные принципы тестирования.

- Уровни тестирования, понятие, примеры.
- Как Вы считаете: почему не рекомендуют, чтобы тестированием занимались программисты?
- Юнит тестирование в C++: Boost Test Library и Google test. Реализация.
- Юнит тестирования на примере другого ЯП (Go или Rust). Основные фреймворки.
- Порядок создания html файла при юнит-тестировании в C++.
- Порядок создания html файла при юнит-тестировании на примере другого ЯП (Go или Rust)

Реализация абстрактных структур данных.

1. Реализовать классы[1] с базовым набором операций (private, public) на основе лабораторных работ 1 и 2 на языке программирования C++ и одном из множества языков $X \in \{\text{Rust}, \text{Go}\}$:
    a. Массив
    b. Список (односвязный, двусвязный).
    c. Очередь
    d. Стек
    e. Хеш таблицы
    f. Деревья (вариант 1 Бинарное дерево поиска, вариант 2 Full Binary Tree, вариант 3 Complete Binary Tree)

Использование линтеров обязательно.

2. Реализовать покрытие тестами (не менее 85%) в проекте. Benchmark использование обязательно.
    a. C++: 1) Boost Test Library, 2) Google test, 3) CxxTest || Catch2 || Qtest || etc. Создать HTML-отчеты о покрытии, которые предоставляют визуальный анализ того, как много кода было протестировано.
    b. Go: testing, testify. Создать HTML-отчет о покрытии, который предоставляет визуальный анализ того, как много кода было протестировано.
    c. Rust: Unit-Test Rust
3. Выполнить сериализацию и десериализацию данных бинарного и текстового форматов для проектов 1 и 2 соответственно[2].
4. В отчете представить UML диаграммы классов.
5. Теоретическая часть (не более 15 страниц).
    a. Четный вариант. Проанализировать мировые практики к оценке качества ПО (например, ISO 25010).
    b. Нечетный вариант. Анализ российских стандартов оценки качества ПО.
    c. Для всех вариантов. Основные виды тестирования.

Отчет содержит список использованных источников.

[1] Обязательно использовать многофайловый проект.

[2] В случае трудоемкости процессов сериализации/десериализации на ЯП C++ допускается использование только бинарного формата

Литература

1. Майерс, С. Искусство тестирования программ / С. Майерс [и др.]. – М. : Диалектика, 2020. – 272 с.

2. Кент, Б. Экстремальное программирование: разработка через тестирование / Б. Кент. – СПб: Питер, 2023. – 224 с.

3. Назина, О. Что такое тестирование. Курс молодого бойца / О. Назина. – БХВ, 2022. – 592 с.

https://habr.com/ru/articles/597859/