Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий
Кафедра «Вычислительная техника»

Основы проектирования программного обеспечения

Инструментальные средства анализа кода программ

Преподаватель _____ А. П. Яблонский
подпись, дата инициалы, фамилия

Студент КИ24-06Б, 032430855 _____ А. А. Каракулов
номер группы, зачетной книжкой подпись, дата инициалы, фамилия

Красноярск 2025

Введение

В качестве объекта исследования используется программа из практической работы №3 для обработки данных о морях (название, глубина, солёность). В работе №3 программа была рефакторингована и покрыта модульными тестами GoogleTest. Цель текущей работы - оценить качество решения с применением внешних инструментов анализа и устранить выявленные недостатки.

Работа заключается в:

1. исследовании программы с использованием инструментов статического и динамического анализа кода, не встроенных в IDE по умолчанию;
2. оценке качества решения по критериям качества ПО;
3. рефакторинге и устранении выявленных дефектов;
4. оформлении UML-диаграммы зависимостей.

Исходный код программы (работа 3)

Исходный код работы №3 включает следующие основные файлы проекта:

1. seas.h - объявление структуры Sea, констант и прототипов функций;
2. Seas_table_KarakulovKI24-06B (2)_fixed.cpp - реализация функций и консольного интерфейса;
3. test.cpp - модульные тесты GoogleTest для нетривиальной логики.

Полная версия проекта и история изменений размещаются в репозитории
OPPO/Master/work_3_googletest/
https://github.com/Sosiska-v-Smekte/OPPO.git

Статический анализ кода

Для статического анализа использовались консольные инструменты компилятора GCC (предупреждения -Wall/-Wextra/-Wpedantic и встроенный статический анализатор -fanalyzer). Пример команды:

```
g++ -std=c++17 -DUNIT_TEST -Wall -Wextra -Wpedantic -fanalyzer -c Seas_table_KarakulovKI24-06B (2)_fixed.cpp
```

Результат анализа (обнаруженные дефекты/замечания):

Дефект 1. Несовпадение знаковости типов (signed/unsigned) в циклах поиска индексов:

```
/mnt/data/Seas_table_KarakulovKI24-06B (2)_fixed.cpp: In function 'int
FindDeepestSeaIndex(const std::vector<Sea>&)':
/mnt/data/Seas_table_KarakulovKI24-06B (2)_fixed.cpp:73:23: warning: comparison of integer
expressions of different signedness: 'int' and 'std::vector<Sea>::size_type' {aka 'long unsigned int'}
[-Wsign-compare]
  73 |     for (int i = 1; i < seas.size(); ++i) {
     |                     ~~^~~~~~~~~~~~~~
/mnt/data/Seas_table_KarakulovKI24-06B (2)_fixed.cpp: In function 'int
FindLeastSaltySeaIndex(const std::vector<Sea>&)':
/mnt/data/Seas_table_KarakulovKI24-06B (2)_fixed.cpp:86:23: warning: comparison of integer
expressions of different signedness: 'int' and 'std::vector<Sea>::size_type' {aka 'long unsigned int'}
[-Wsign-compare]
  86 |     for (int i = 1; i < seas.size(); ++i) {
     |                     ~~^~~~~~~~~~~~~~
```

Дефект 2. Потенциальная непереносимость кода: функции SetConsoleCP/SetConsoleOutputCP используются без дополнительной защиты. Это мешает сборке на не-Windows платформах.

Дефект 3. Предупреждения о неинициализированных полях структуры Sea (типичный случай при включённом статическом анализе компилятора/IDE).

Дефект 4. Парсер файла жёстко пропускал первую строку как заголовок. Если файл не содержит заголовка, первая запись теряется.

Динамический анализ кода

Динамический анализ выполнен с помощью модульных тестов GoogleTest и санитайзера AddressSanitizer (ASan) в составе GCC. Пример команды для запуска с ASan:

```
g++ -std=c++17 -O0 -g -DUNIT_TEST -fsanitize=address -fno-omit-frame-pointer
Seas_table_KarakulovKI24-06B (2)_fixed.cpp asan_harness.cpp -o asan_run
```

При выполнении тестов и запуске вспомогательного сценария (harness) ошибок времени выполнения, утечек памяти и обращений вне границ памяти не обнаружено.

Рисунок 1 - результат выполнения модульных тестов GoogleTest.

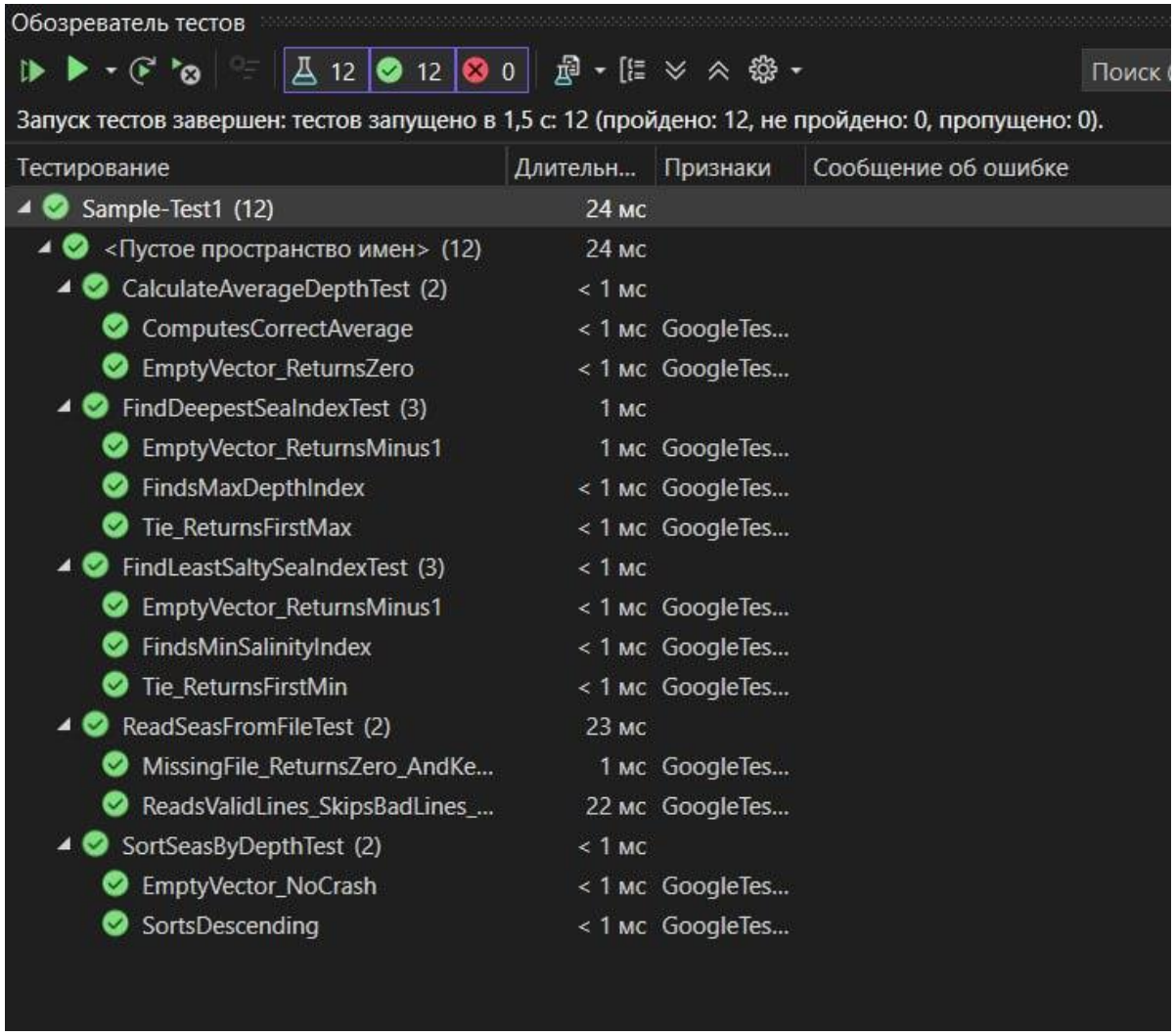

Рисунок 1 – Тесты

Оценка качества решения

Оценка выполнена по базовым критериям качества ПО.

Таблица 1

Критерий	Состояние (до исправлений)	Что улучшено
Надёжность	Ошибки парсинга файла могли приводить к пропуску строк; предупреждения о типах.	Добавлена безопасная функция ParseSeaLine, обработка заголовка, исправлены предупреждения типов.
Сопровождаемость	Часть логики смешана с I/O; дублирование логики парсинга.	Выделены вспомогательные функции парсинга, упрощены циклы; константы сделаны inline constexpr.
Переносимость	Windows-специфичный код мог мешать сборке на других ОС.	Вызовы SetConsoleCP/SetConsoleOutputCP обёрнуты в #ifdef _WIN32.
Тестируемость	Логика в целом отделена, но поля Sea могли быть неинициализированы.	Структура Sea получила значения по умолчанию, что снижает риск UB и ложных дефектов.

Улучшенный исходный код (после устранения дефектов)

https://github.com/Sosiska-v-Smekte/OPPO.git

После анализа кода выполнены следующие точечные улучшения:

1. исправлены предупреждения signed/unsigned: индексы в циклах поиска переведены на std::size_t;

2. константы в заголовочном файле заменены на inline constexpr;
3. структура Sea получила значения по умолчанию (исключение неинициализированных полей);
4. улучшен парсер ReadSeasFromFile: добавлен разбор первой строки как заголовка или данных, добавлен trim полей;
5. Windows-специфичные функции смены кодовой страницы обёрнуты в #ifdef _WIN32;
6. при вводе с клавиатуры вектор очищается, чтобы не накапливать данные при повторном вызове.

Улучшенная версия исходного кода приведена в приложении и должна быть размещена в репозитории вместе с отчётом.

UML-диаграмма зависимостей

Рисунок 2 - UML-диаграмма зависимостей компонентов программы.

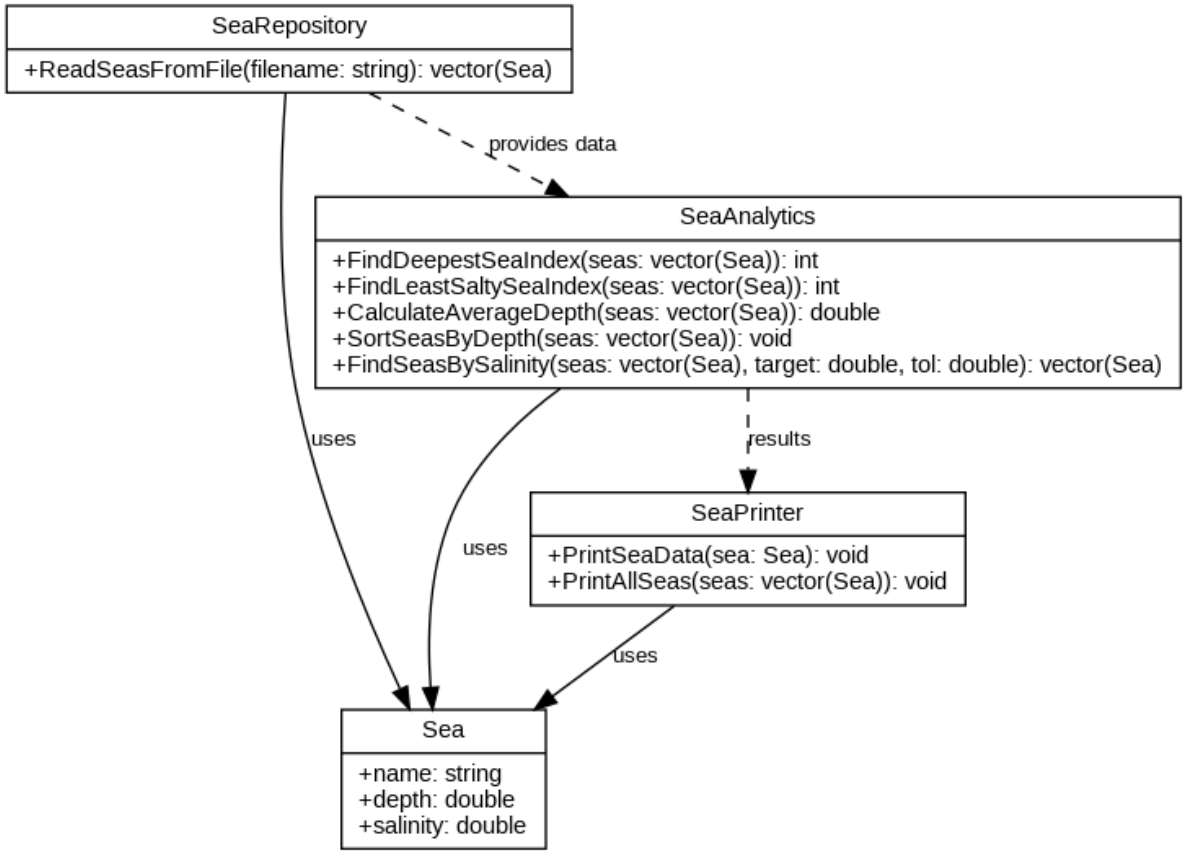

Рисунок 2 – UML-диаграмма

Заключение

В ходе работы выполнен статический и динамический анализ программы из работы №3 с использованием внешних инструментов (GCC -fanalyzer, AddressSanitizer, GoogleTest). Выявлены и устранены дефекты, влияющие на переносимость и сопровождаемость. Подготовлена UML-диаграмма зависимостей и оформлен отчёт.