|  |
| --- |
| Министерство науки и высшего образования РФ Федеральное государственное автономное образовательное учреждение высшего профессионального образования **«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»** |
| Институт космических и информационных технологий |
| Кафедра вычислительной техники |

**Практическая работа №5**

**«**Инструментальные средства анализа кода программ**»**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Преподаватель |  |  |  |  |  | Л.С. Артемьев |
|  |  |  |  |  | подпись, дата |  |
| Студент |  | КИ22-07Б, 032214032 |  |  | М.А. Дондокова |
|  |  | номер групп, зачетной книжки |  | подпись, дата |  |

Красноярск 2023

**Задание**

1. Исследование программы полученной в ходе выполнения работы 3 с помощью инструментов статического и динамического анализа кода.
2. Оценка полученных решений в соответствии с критериями качества ПО.
3. Улучшение программы – устранение обнаруженных недочётов.

Таблица 1 – Задание по варианту

|  |  |
| --- | --- |
| № | Задание |
| 17 | Меню: название (строка), цена (дробное), время приготовления |

**Ход работы**

Для выполнения 5 лабораторной работы потребовалось установить приложение «Cppcheck». Cppcheck – это [инструмент статического анализа](http://en.wikipedia.org/wiki/Static_analysis_tool) кода на C / C ++. Он обеспечивает [уникальный анализ кода](http://cppcheck.net/#unique) для обнаружения ошибок и фокусируется на обнаружении неопределенного поведения и опасных конструкций кода. Цель состоит в том, чтобы иметь как можно меньше ложных срабатываний. Cppcheck разработан для анализа кода на C / C ++, даже если он имеет нестандартный синтаксис.

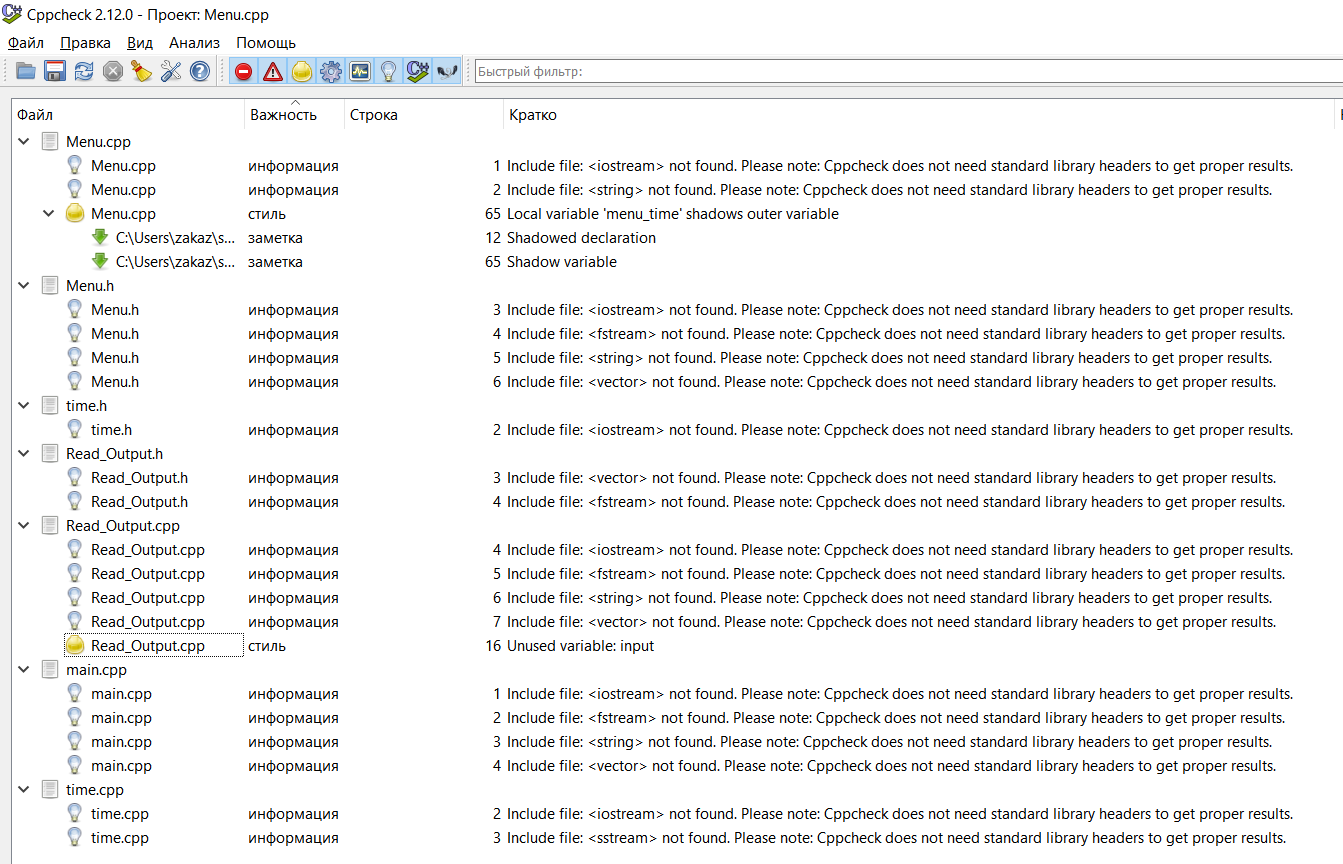


Рисунок 1 – Анализ программы

**Примечания:**

* во вкладке «Файл» содержатся все открытые пользователем файлы;
* вкладка «Важность» информирует пользователя о степени ошибочности:

1. Информация – информационная важность показывает на то, что библиотеки, содержащиеся в файлах не нужны программе для чтения;
2. Стиль – показывает, как может быть объявлен параметр.

* вкладка «Строка» показывает строку в файле, где была обнаружена ошибка;
* вкладка «Кратко» содержит общую информацию об ошибочности конкретной строки в конкретном файле.

**Устранение полученных недочетов**

Ошибка «shadow variable» (теневая переменная) возникает, когда в блоке кода определяется переменная с тем же именем, что и переменная, объявленная во внешнем блоке кода. Таким образом, внутренняя переменная «затеняет» внешнюю переменную, и при обращении к переменной внутри блока кода будет использоваться внутренняя переменная, а не внешняя. Это может привести к непредсказуемому поведению программы. Чтобы избежать этого нужно переименовать переменные так, чтобы они не конфликтовали с членами класса:

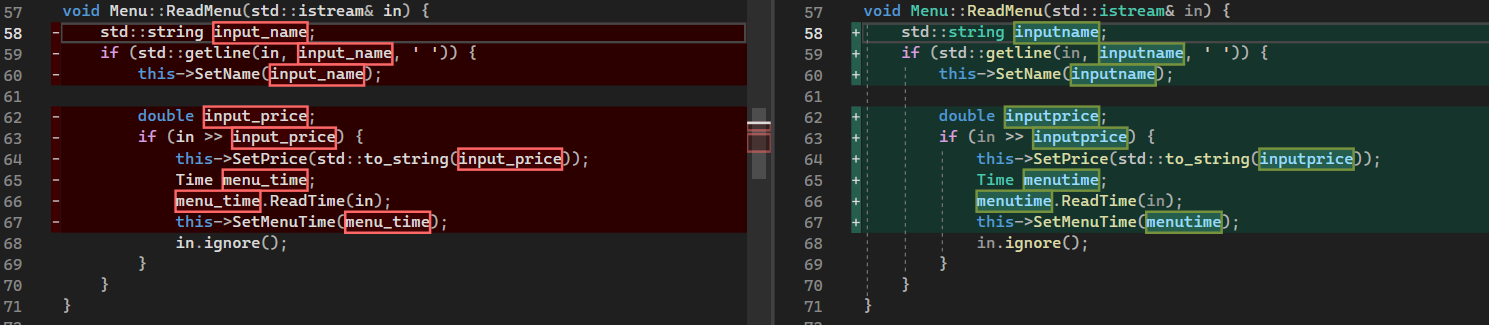


Рисунок 2 – Устранение ошибок в «Menu.cpp»

Ошибка «unused variable» (неиспользуемая переменная) возникает, когда в коде определена переменная, но она не используется ни в каких операциях. Это может произойти, если переменная была объявлена по ошибке или если она была использована ранее, но потом код был изменен, и переменная стала не нужна. Необходимо удалить неиспользуемую переменную input:

Изображение выглядит как Мультимедийное программное обеспечение, программное обеспечение, Графическое программное обеспечение, снимок экрана

Автоматически созданное описание

Рисунок 3 – Устранение ошибки в «Read\_Output.cpp»

Также во вкладке важность критерий информация во всех файлах содержит одну и ту же проблему – «missingIncludeSystem». Cppcheck не нуждается в заголовках стандартных библиотек для получения правильного результата. Для решения этих проблем, в каждом файле нужны удалить библиотеки.

После повторного анализа программы можно увидеть, что остались только информационные критерии, а все остальные ошибки исправлены:

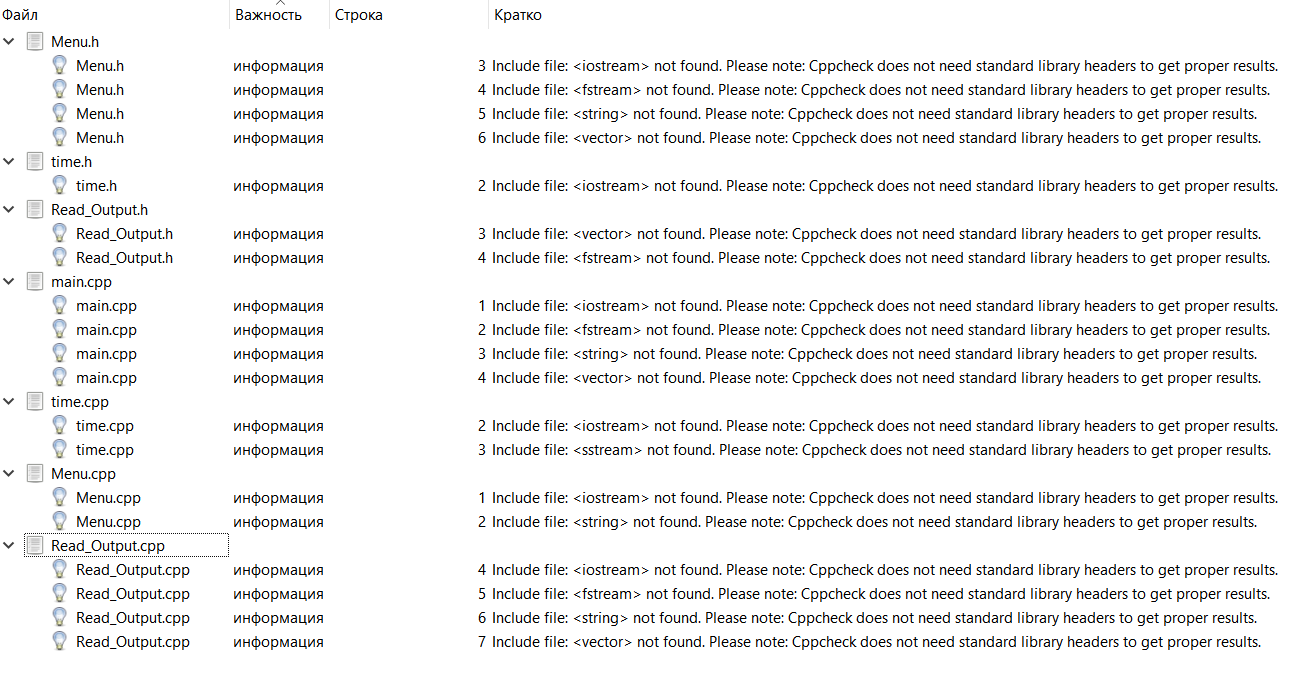


Рисунок 4 – Анализ программы

Для динамического анализа используем библиотеку CRT. Для её подключения нужно внести изменения в «main.cpp»:

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 5 – Изменение программы в «main.cpp»

После запуска программы получим данные об утечки памяти:

Изображение выглядит как текст, снимок экрана, Шрифт, черно-белый

Автоматически созданное описание

Рисунок 5 – Утечка памяти

**Заключение**

В ходе выполнения лабораторной работы были получены новые и закреплены старые навыки анализа ошибок и их устранения, также получены знания в оценке полученных решений в соответствие с критериями качества ПО. Поставленная задача решена полностью. Её итогом стала программа, не имеющая ошибок.