|  |
| --- |
| 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, 032201522 |  |  | М.А. Елин |
|  |  | номер групп, зачетной книжки |  | подпись, дата |  |

Красноярск 2023

**Техническое 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.
4. Исследование программы с использованием статического анализатора

Исследование будет проводиться с использование статического анализатора PVS-Studio. После анализа нашего кода получаем результат, показанный на рисунке 1.

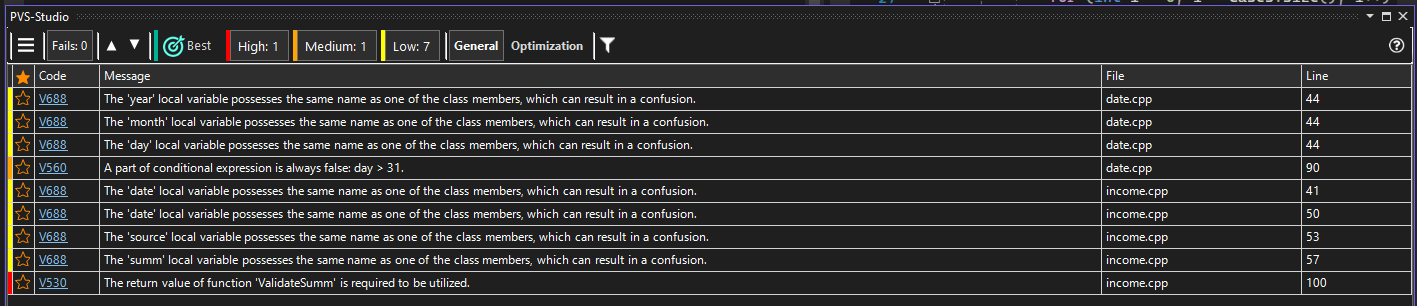


Рисунок 1 – Результат проверки в PVS-Studio

**Устранение полученных недочетов**

Для решения “High” ошибки, обнаруженной статическим анализатором PVS-Studio, я удалил лишний вызов внутри функции InvalidSumm.

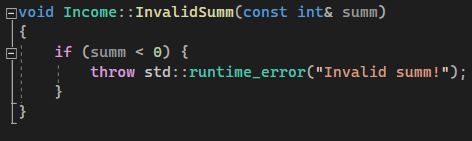


Рисунок 2 – Исправленный вариант кода.

Ошибки “Low” включают в себя только ошибки V688. Это указывает на то, что имя аргумента функции совпадает с именем члена класса, что может привести к путанице. В данном случае, это относится к методу **SetSumm**. Для устранения этой ошибки, надо изменить имя аргумента, чтобы оно отличалось от имени члена класса.

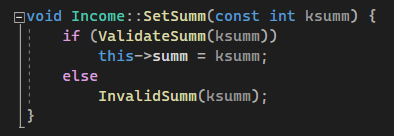


Рисунок 3 – Исправленный вариант кода для одной из “Low” ошибок.

В этом случае, ошибка V688 указывает на конфликт имен между локальной переменной summ и членом класса summ. Для разрешения этой ошибки нужно изменить имя локальной переменной, чтобы оно отличалось от имени члена класса.

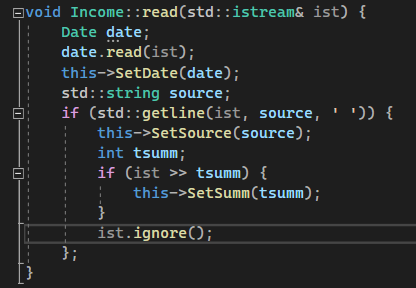
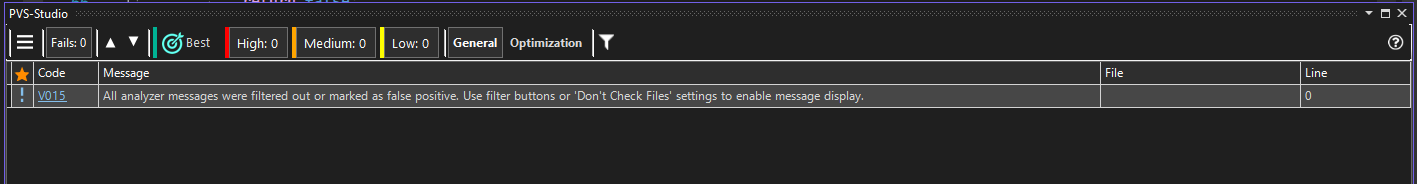


Рисунок 3 – Исправленный вариант кода для оставшихся “Low” ошибок.

Аналогично изменяем ошибки “Low” для других частей кода.

После устранения всех ошибок запускаем анализатор кода и видим:



Анализатор сообщает о том, что никаких ошибок не осталось. Соответственно мы исправили все ошибки по коду.

1. Исследование программы с использование динамического анализатора

Воспользуемся динамическим анализатором CRT для проверки на утечки памяти. Для этого добавим макросы в самое начало в основном файле, а в конце работы программы вызовем функцию \_CrtDumpMemoryLeaks().

Изменения можно посмотреть в фиксации с номером fb06e8d4.

После выполнения программы не получаем сообщений о найденных утечках, а значит программа не имеет утечек памяти.

**Заключение**

В ходе выполнения практической работы мы воспользовались статическим анализатором PVS-Studio который помог нам обнаружить ошибки, которые мы исправили в процессе выполнения работы. Также мы воспользовались динамическим анализатором, который помогает отследить утечки памяти, в нашем случае утечек памяти выявлено не было. Поставленное изначально задание было выполнено.