|  |
| --- |
| 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, 032210304 |  |  | Н.М. Егоров |
|  |  | номер групп, зачетной книжки |  | подпись, дата |  |

Красноярск 2023

**Техническое задание**

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

|  |  |
| --- | --- |
| Вариант | Объект, свойства |
| 4 | Двумерные точки: координаты (два дробных числа), цвет (строка из списка "red", "green", "blue") |

Таблица 1 - Задание по варианту

**Ход работы**

**Статический анализ кода:** Для выполнения 5 лабораторной работы устанавливаем расширение PVS-Studio для Microsoft Visual Studio. Суть его работы заключается в анализе исходного кода и поиска мест, которые похожи на известные шаблоны кода с ошибкой. Шаблонов очень много, при этом сложность их выявления крайне разнится. Более того, некоторые диагностики для выявления опечаток прибегают к эмпирическим алгоритмам.

Запускаем PVS-Studio:

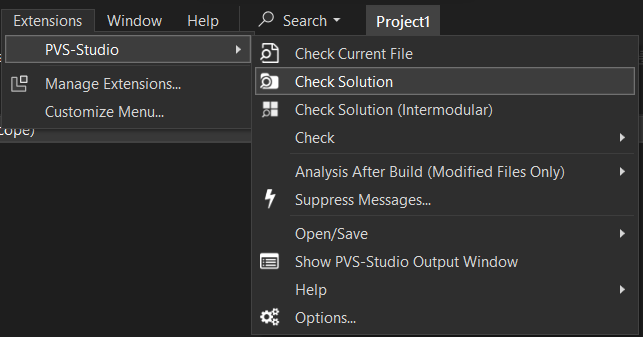


Рисунок 1 – Запуск PVS-Studio

Результат выполнения анализа:

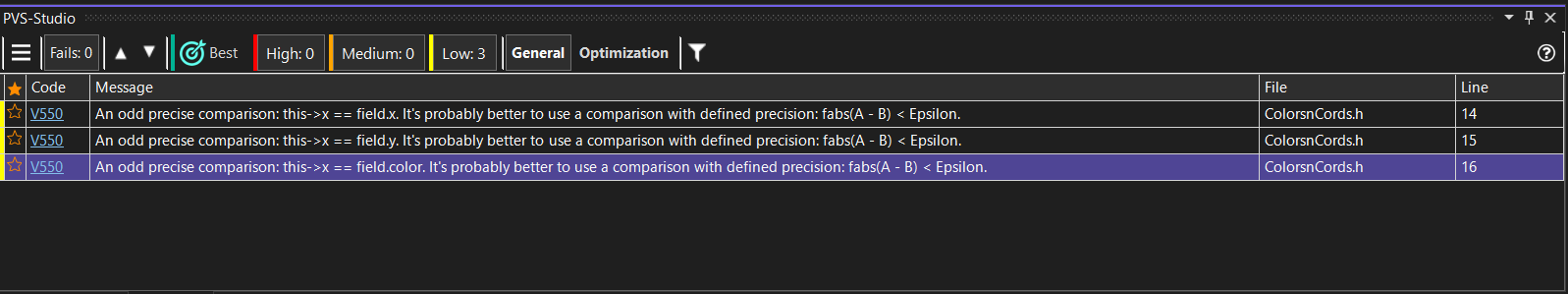


Рисунок 2 – Анализ программы

Расширение нашло 3 незначительных ошибки одинакового типа.

Их устранение может негативно повлиять на работу класса. Так как определенный для этой структуры оператор «==» служит для выявления объектов с одинаковыми полями, а отклонение на некоторое позволит считать идентичными объекты с по факту разными координатами.

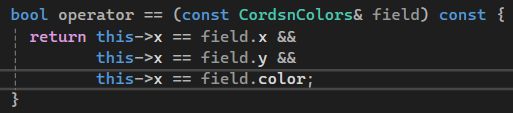


Рисунок 3 – Место с ошибкой

**Динамический анализ кода:** Для применения динамического анализатора возьмем пример утечки памяти. Для этого я использовал библиотеку CRT. Для её подключения в начале основного файла «main.cpp» прописываем данные команды:

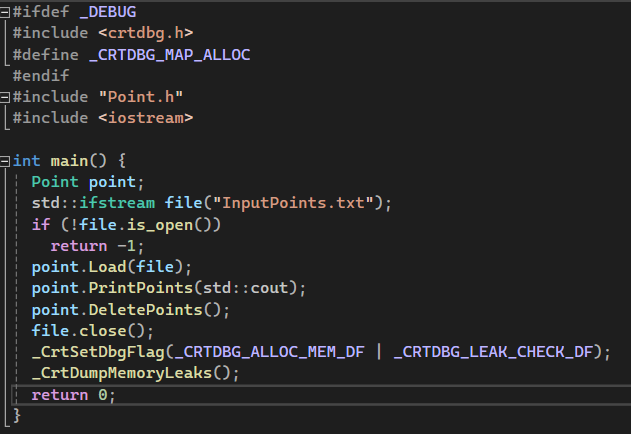


Рисунок 3 – Применение CRT

Также в начале программы прописываем:

“\_CrtSetDbgFlag(\_CRTDBG\_ALLOC\_MEM\_DF | \_CRTDBG\_LEAK\_CHECK\_DF);”

А перед завершением программы прописываем:

“\_CrtDumpMemoryLeaks();”

После запуска программы получим данные об утечки памяти:



Рисунок 4 – Утечка памяти

Так как функция вызывается до того как сработает «return» в теле функции «main», созданный ранее объект класса «Point» все еще находится в ее зоне видимости, поэтому память, выделенная полю «std::vector<CordsnColors>data\_» еще не освобождена. Поскольку в программе не используется явное динамическое выделение памяти с помощью «new» и т.п, можно сделать вывод, что программа не имеет утечек памяти.