|  |
| --- |
| 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, 032212020 |  |  | П.А. Темников |
|  |  | 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.

Таблица 1 – Задание по варианту

|  |  |
| --- | --- |
| Вариант | Задание |
| 8 | Фиксация проезда автомобилей: дата, номер автомобиля (строка в характерном формате) |

**Ход работы**

**Статический анализ кода:** Для выполнения 5 лабораторной работы мне потребовалось установить расширение PVS-Studio для Visual Studio. Суть его работы заключается в анализе исходного кода, также после анализа программа выведет все твои недостатки и опишет проблему. PVS-Studio - это инструмент статического анализа исходного кода.

Для того, чтобы заработала бесплатная версия PVS-Studio, в каждом cpp-файле прописываем данные комментарии в начале:

// This is a personal academic project. Dear PVS-Studio, please check it.

Запускаем PVS-Studio:

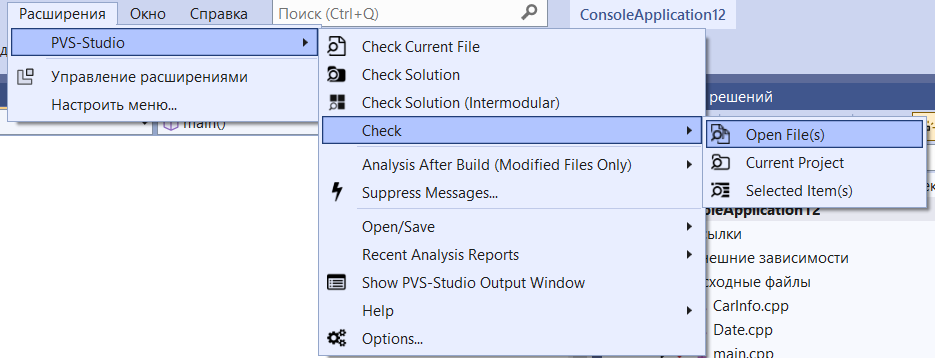


Рисунок 1 – Запуск PVS-Studio

После анализа моего кода, расширение вывело следующее:



Рисунок 2 – Анализ программы

Выходит, что код верный, исправлять ничего не требуется.

**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:

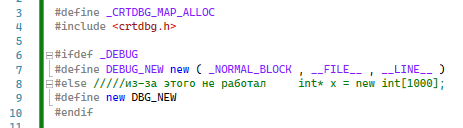


Рисунок 3 – Применение CRT

Также в начале программы прописываем:

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

А перед завершением программы прописываем:

“\_CrtDumpMemoryLeaks();”

Для проверки используем строку:

int\* x = new int[1000];

После запуска программы получим данные об утечки памяти:

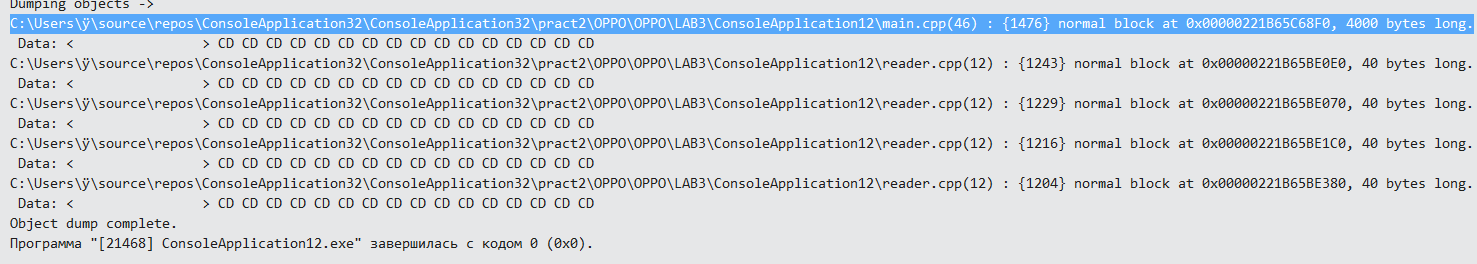


Рисунок 4 – Утечка памяти

**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.