

ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА АНАЛИЗА КОДА ПРОГРАММ

1. Исследование программы с использованием статического анализатора Cppcheck

Используя статический анализатор, удалось выяснить, что в программе нет никаких критических ошибок. Результаты работы анализатора представлены на рисунке 1.

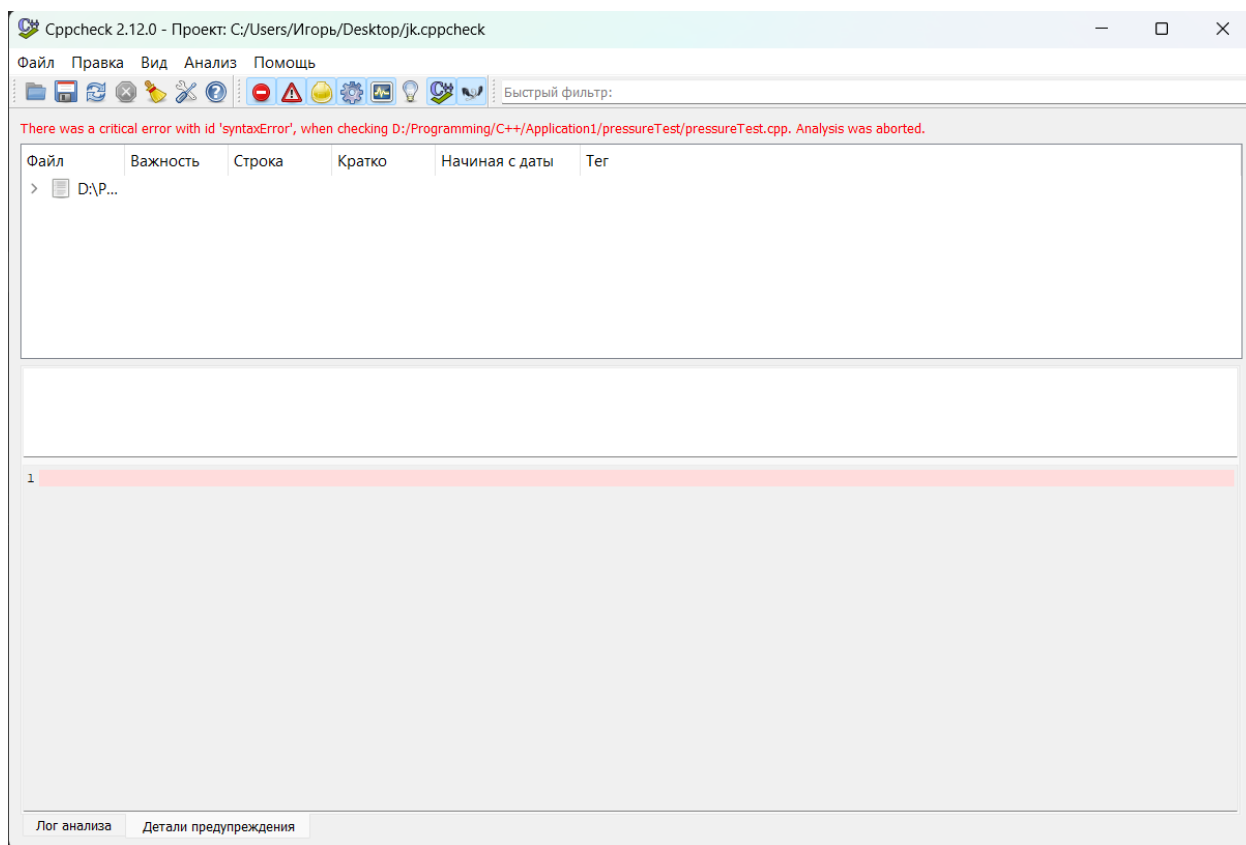


Рисунок 1 – результат работы анализатора Cppcheck.

Как видно из рисунка, ошибок действительно нет, за исключением синтаксической ошибки. Однако в коде программы подобной ошибки нет, поэтому было принято решение проверить программу повторно с использованием подобного анализатора.

2. Исследование программы с использованием статического анализатора PVS-Studio

Используя анализатор PVS-Studio удалось выявить несколько ошибок, разделённых различными категориями. Ошибки вида "General", найденные в программе, представлены на рисунке 2.

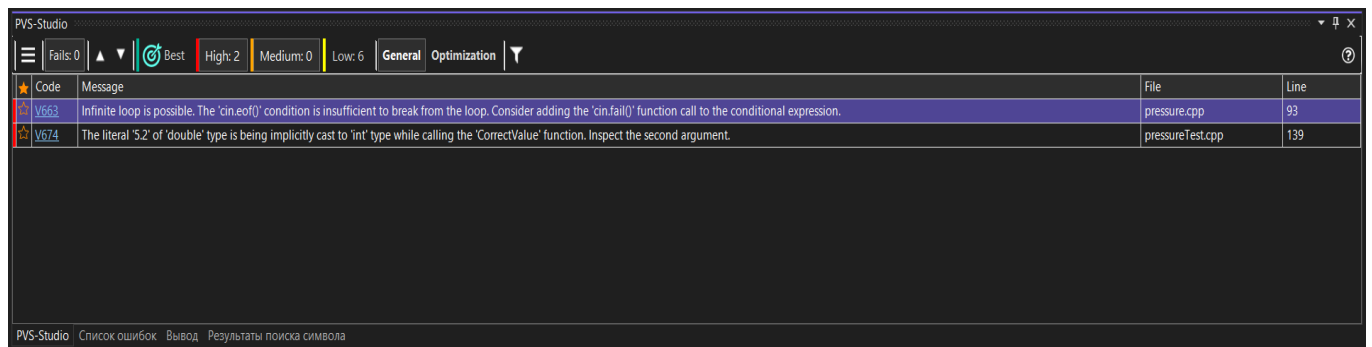


Рисунок 2 – Ошибки вида “General”, найденные в программе.

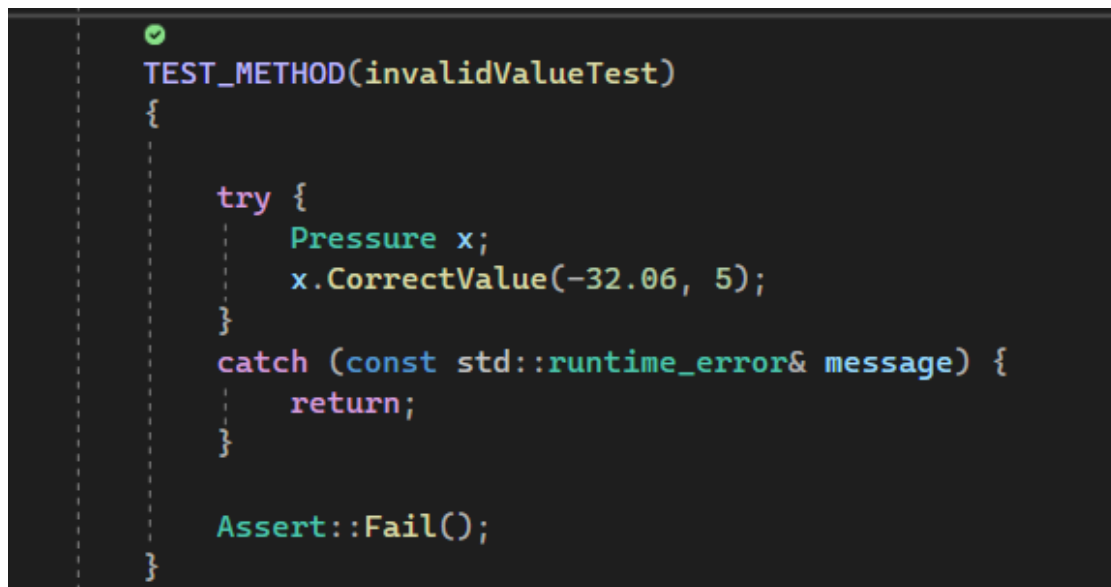
Как видно из рисунка 2, было найдено две ошибки в файлах: pressure.cpp и pressureTest.cpp. Для исправления первой ошибки “-V663 *Infinite loop is possible. The 'cin.eof()' condition is insufficient to break from the loop. Consider adding the 'cin.fail()' function call to the conditional expression. pressure.cpp 93*” было принято решение добавить в условие цикла while дополнительный параметр. Исправление первой ошибки представлено на рисунке 3.

```
//Определение функции считывания данных из файла
void Pressure::readPressure(std::istream& file, std::vector<Pressure>& pvec) {
    while (!file.eof() && !std::cin.fail()) {
        Pressure pressureData = Pressure::createFromStream(file);
        pvec.push_back(pressureData);
    }
}
```

Рисунок 3 – Исправление ошибки V663.

Как видно из рисунка 3, в условие цикла было добавлено выражение “!std::cin.fail()”, что позволило корректно завершать цикл в случае, если чтение данных из файла будет неудачным.

Для исправления второй ошибки “V674 *The literal '5.2' of 'double' type is being implicitly cast to 'int' type while calling the 'CorrectValue' function. Inspect the second argument. pressureTest.cpp 139*” было принято решение изменить второй параметр в файле PressureTest.cpp в функции invalidValueTest(строка 133). Исправление второй ошибки представлено на рисунке 4.



```

TEST_METHOD(invalidValueTest)
{
    try {
        Pressure x;
        x.CorrectValue(-32.06, 5);
    }
    catch (const std::runtime_error& message) {
        return;
    }

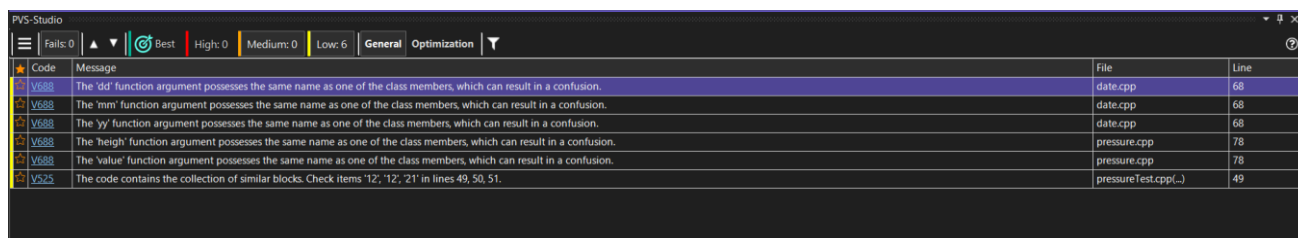
    Assert::Fail();
}

```

Рисунок 4 – Исправление ошибки V674

Как видно из рисунка 4, старое значение «5.02», имеющее тип float, было заменено на новое корректное значение «5», имеющее тип int, что помогло избавиться от ошибки.

Помимо ошибок вида “General”, были найдены также ошибки вида “Low”, которые являются незначительными. На рисунке 5 представлены найденные ошибки.



Code	Message	File	Line
V688	The 'dd' function argument possesses the same name as one of the class members, which can result in a confusion.	date.cpp	68
V688	The 'mm' function argument possesses the same name as one of the class members, which can result in a confusion.	date.cpp	68
V688	The 'yy' function argument possesses the same name as one of the class members, which can result in a confusion.	date.cpp	68
V688	The 'heigh' function argument possesses the same name as one of the class members, which can result in a confusion.	pressure.cpp	78
V688	The 'value' function argument possesses the same name as one of the class members, which can result in a confusion.	pressure.cpp	78
V525	The code contains the collection of similar blocks. Check items '12', '12', '21' in lines 49, 50, 51.	pressureTest.cpp(..)	49

Рисунок 5 – Ошибки вида “Low”, найденные в программе.

Как видно из рисунка, в трёх файлах (date.cpp, pressure.cpp, pressureTest.cpp), были найдены ошибки всего двух видов – V688 и V525. Для решения ошибки V688, которая имеет вид: “V688 The 'dd' function argument possesses the same name as one of the class members, which can result in a confusion. date.cpp 68”, было принято решение о простом переименовании параметров функции Correct(). Исправление ошибки представлено на рисунке 6.

```

//Определение функции проверки даты на корректность
void DateStruct::Correct(int day, int month, int year) {
    int maxDaysInMonth = 31;
    switch (mm) {
        case 2:
            maxDaysInMonth = isLeapYear(yy) ? 29 : 28;
            break;
        case 4:
        case 6:
        case 9:
        case 11:
            maxDaysInMonth = 30;
            break;
    }

    if (day < 1 || !(day <= maxDaysInMonth)) throw std::runtime_error("День введён неверно! Проверьте корректность входных данных");
    if (month < 1 || month > 12) throw std::runtime_error("Месяц введён неверно! Проверьте корректность входных данных");
    if (year < 0 || year > 99) throw std::runtime_error("Год введён неверно! Проверьте корректность входных данных");
}

```

Рисунок 6 – Исправление ошибки V688.

Как видно из рисунка, старые параметры функции Correct() – dd, mm, yy, были заменены на более осмысленные day, month, year, что помогло решить ошибку. Аналогичное решение, представленное на рисунке 7, было принято в файле pressure.cpp.

```

//Определение функции проверки на корректность вводимых значений
void Pressure::CorrectValue(float h, int v) {
    if (h <= 0) throw std::runtime_error("Неверно введена высота! Проверьте корректность данных!");
    if (v <= 0) throw std::runtime_error("Неверно введено значение давления! Проверьте корректность данных!");
}

```

Рисунок 7 – Исправление ошибки V688.

Подобно решению выше, параметры функции CorrectValue() – height, value, были заменены на h, v, что помогло решить проблему.

Для исправления ошибки V525, которая имеет вид – “V525. Code contains collection of similar blocks. Check items 12, 12, 21 in lines 49, 50, 51”, было принято решение об изменении параметра в функции setMM() (строка 50). Исправление ошибки V688 представлено на рисунке 8.

```

TEST_METHOD(correctDatePrintTest) {
    DateStruct date;

    date.setDD(12);
    date.setMM(11);
    date.setYY(21);

    std::string d = "Дата: 12.11.21";

    std::stringstream x;
    date.printDate(x);
    Assert::AreEqual(x.str(), d);
}

```

Рисунок 8 – Исправление ошибки V525.

Как видно из рисунка, параметр «12» в функции setMM() был заменён на параметр «11», что позволило избавиться от повторяющихся блоков и исправить ошибку V525.

Помимо всех вышеперечисленных ошибок, были найдены также ошибки вида “High”. На рисунке 9 представлены найденные ошибки.

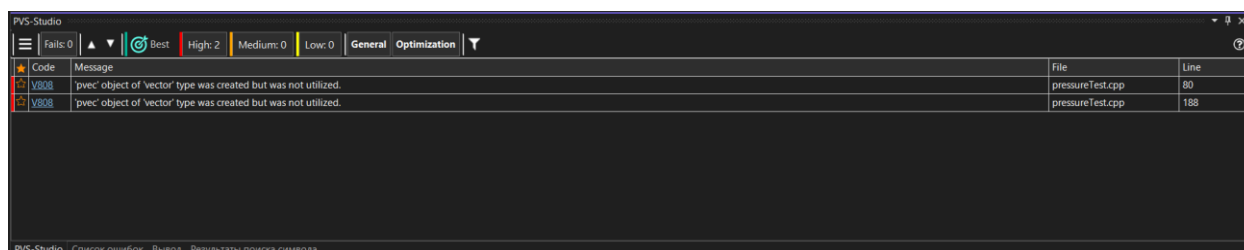


Рисунок 9 – Ошибки вида “High”, найденные в программе.

Как видно из рисунка 9, была найдено две ошибки одного типа – V808, которые имеют вид: “V808. ‘pvec’ object of ‘vector’ type was declared but was not utilized.”. На рисунке 10 представлено исправление этой ошибки.

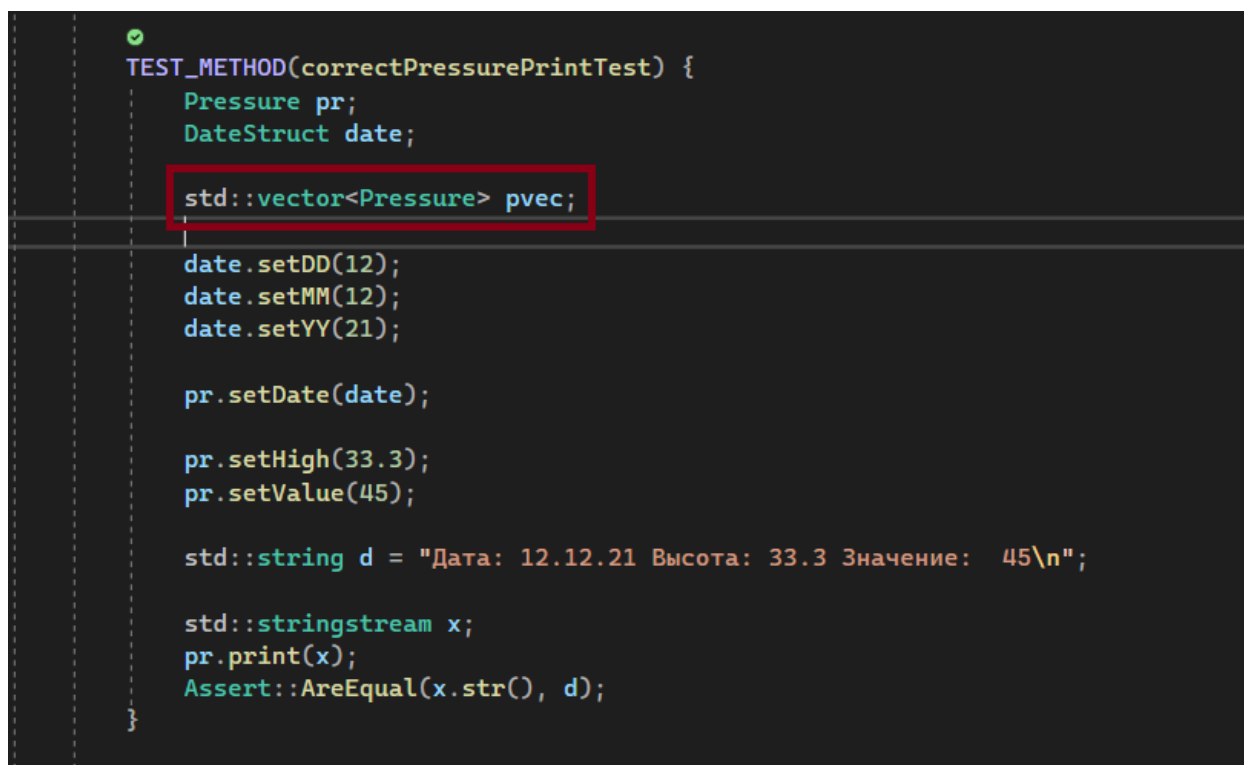


Рисунок 10 – Исправление ошибки V808.

Как видно из рисунка 10, для устранения ошибки было принято решение об удалении вектора pvec, так как после создания в функции он не использовался, что и вызвало подозрение у анализатора. Удаление неиспользуемого вектора помогло исправить ошибку V808. Подобная ошибка была исправлена и в другой функции, расположенной на строке 188. Её решение аналогично решению выше.

После выполнения всех вышеперечисленных ошибок, анализатор уведомил о том, что ошибок больше в программе нет. Уведомление представлено на рисунке 11.

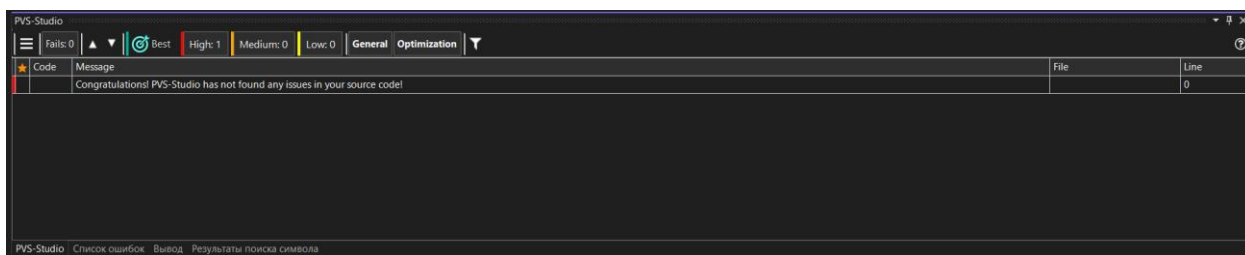


Рисунок 11 – Уведомление об отсутствии ошибок.

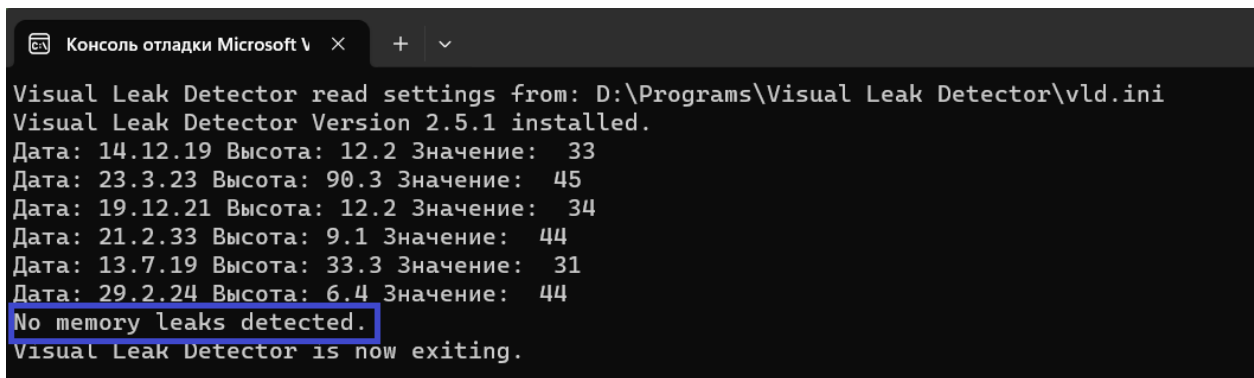
3. Исследование программы на возможные утечки памяти с использованием Visual Leak Detector

Для проверки программы на всевозможные утечки памяти было принято решение об использовании утилиты Visual Leak Detector. Результат работы утилиты по умолчанию выводится в консоль. Для обеспечения её работоспособности необходимо подключить её, используя директиву `#include`. На рисунке 12 представлено подключение утилиты в основной файл программы – `Application1.cpp`.

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <vld.h>
#include "date.h"
#include "pressure.h"
```

Рисунок 12 – Подключение утилиты.

После подключения достаточно лишь запустить программу для проверки её на возможные утечки. Уведомление о том, что утечки памяти в программе отсутствуют, изображено на рисунке 13.



```
Консоль отладки Microsoft V  X + v
Visual Leak Detector read settings from: D:\Programs\Visual Leak Detector\vld.ini
Visual Leak Detector Version 2.5.1 installed.
Дата: 14.12.19 Высота: 12.2 Значение: 33
Дата: 23.3.23 Высота: 90.3 Значение: 45
Дата: 19.12.21 Высота: 12.2 Значение: 34
Дата: 21.2.33 Высота: 9.1 Значение: 44
Дата: 13.7.19 Высота: 33.3 Значение: 31
Дата: 29.2.24 Высота: 6.4 Значение: 44
No memory leaks detected.
Visual Leak Detector is now exiting.
```

Рисунок 13 – Уведомление об отсутствии утечек памяти в программе.

Как видно из рисунка 13, в программе утечек памяти не наблюдается.

После выполнения всех вышеперечисленных действий, можно сказать, что программа была исследована на предмет всевозможных ошибок и утечек с использованием различных инструментов.