

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

**ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА АНАЛИЗА
КОДА ПРОГРАММ**

Вариант №14

Преподаватель

подпись, дата

Л. С. Артемьев

инициалы, фамилия

Студент

КИ22-08Б, 032210145
номер группы, зачетной книжкой

подпись, дата

К. В. Рожков

инициалы, фамилия

Красноярск 2023

1 Проведение анализа с помощью статических анализаторов

Статический анализ кода был проведён с помощью двух статических анализаторов: Cppcheck и PVS-Studio.

1.1 Статический анализ с помощью Cppcheck

Для анализа кода с помощью Cppcheck необходимо передать утилите при запуске в качестве параметра путь к директории с файлами исходного кода. Результат анализа приведён на рисунке 1.



```
kirill@kirill-IdeaPad-3:~/2_course/OPP0/1$ cppcheck ./src
Checking src/fish.cpp ...
1/4 files checked 15% done
Checking src/main.cpp ...
2/4 files checked 47% done
Checking src/sea.cpp ...
3/4 files checked 79% done
Checking src/utils.cpp ...
4/4 files checked 100% done
kirill@kirill-IdeaPad-3:~/2_course/OPP0/1$
```

Рисунок 1 – Результат анализа с помощью Cppcheck

Как видно из результатов выше, Cppcheck не выдал никаких предупреждений.

1.2 Статический анализ с помощью PVS-Studio

Процесс проведения статического анализа кода проекта с помощью PVS-Studio приведён на рисунке 2.

```

kirill@kirill-IdeaPad-3:~/2_course/OPP0/1/src/build$ pvs-studio-analyzer analyze -o ./project.log
Using tracing file: strace_out
[ 25%] Analyzing: ../fish.cpp
[ 50%] Analyzing: ../main.cpp
[ 75%] Analyzing: ../sea.cpp
[100%] Analyzing: ../utils.cpp
Analysis finished in 0:00:02.42
The results are saved to /home/kirill/2_course/OPP0/1/src/build/./project.log
kirill@kirill-IdeaPad-3:~/2_course/OPP0/1/src/build$ plog-converter -t fullhtml ./project.log -o ./report_dir/
Analyzer log conversion tool.
Copyright (c) 2023 PVS-Studio LLC


PVS-Studio is a static code analyzer and SAST (static application security
testing) tool that is available for C and C++ desktop and embedded development,
C# and Java under Windows, Linux and macOS.

Total messages: 4
Filtered messages: 0
kirill@kirill-IdeaPad-3:~/2_course/OPP0/1/src/build$

```

Рисунок 2 – Проведение статического анализа с помощью PVS-Studio

В результате выполнения приведённых выше команд был получен следующий отчёт в формате html (рисунок 3).

 <h2>PVS-Studio Analysis Results</h2>	
Date:	Wed Dec 20 19:26:23 2023
PVS-Studio Version:	7.28.78193.374
Command Line:	plog-converter -t fullhtml ./project.log -o ./report_dir/
Total Warnings (GA):	3

Group	Location	Level	Code	Message
General Analysis	main.cpp:35	High	V563	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.
General Analysis	main.cpp:35	Medium	V1024	The 'ist' stream is checked for EOF before reading from it, but is not checked after reading. Potential use of invalid data.
General Analysis	utils.hpp:4	High	V1043	A global object variable 'RUS_ALPH' is declared in the header. Multiple copies of it will be created in all translation units that include this header file.

Рисунок 3 – Отчёт об анализе с помощью PVS-Studio

Продублируем предупреждения в виде таблицы (таблица 1).

Таблица 1 – Предупреждения, выданные анализатором PVS-Studio

№	Group	Location	Level	Code	Message
1	General Analysis	main.cpp:25	High	V663	Infinity loop is possible. The 'cin.eof()' condition is insufficient to break from the loop. Consider adding the 'cin.fail()' function call to the condition.
2	General Analysis	main.cpp:35	Medium	V1024	The 'ist' stream is checked for EOF before reading from it, but is not checked after reading. Potential use of invalid data.
3	General Analysis	utils.hpp:4	High	V1043	A global object variable 'RUS_ALPH' is declared in the header. Multiple copies of it will be created in all translation units that include this header file.

Анализатор обнаружил потенциальную возможность появления бесконечного цикла, возможность использования некорректных данных после неудачного чтения данных из потока, а также неправильное объявление константы, что может привести к созданию её множественных копий.

Ниже приведён исправленный фрагмент программы для 1 и 2 предупреждений:

```
while (!ist.eof() && !ist.fail()){
    <...>
}
```

В условие выхода из цикла добавлена проверка состояния флагов ошибок потока.

Ниже приведён исправленный фрагмент программы для 3 предупреждения:

```
inline const std::string RUS_ALPH = "абвгдеёжзийклмнопрстуфхцчщъыьэю-  
яАБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧЩЪЫЬЭЮЯ";
```

Константа объявляется с ключевым словом `inline`, при этом инициализация константы произойдёт один раз.

После исправления исходного кода программы был проведён повторный анализ, результаты которого приведены на рисунке 4.

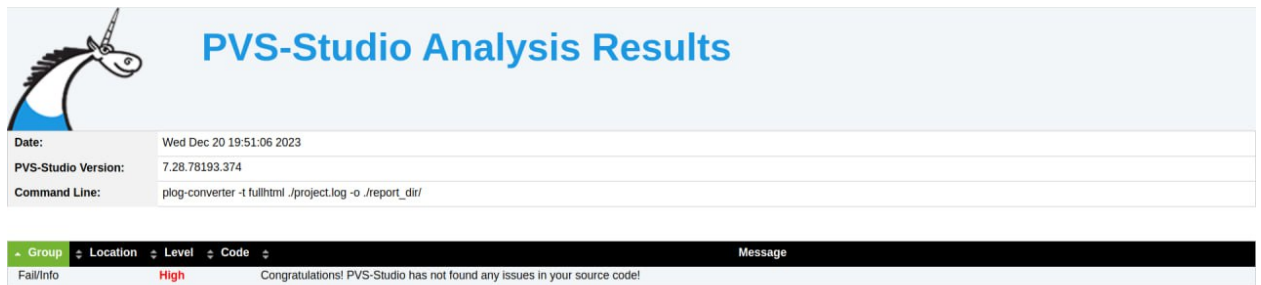


Рисунок 4 – Результат повторного анализа с помощью PVS-Studio

Как видно из результатов повторного анализа, больше предупреждений выдано не было.

2 Проведение анализа с помощью динамического анализатора

Динамический анализ кода был проведён с помощью анализатора Valgrind. Для запуска анализа необходимо передать анализатору путь к исполняемому файлу скомпилированной ранее программы.

Результат анализа приведён на рисунке 5.

```
kirill@kirill-IdeaPad-3:~/2_course/OPP0/1$ valgrind --leak-check=full ./src/a.out
==5282== Memcheck, a memory error detector
==5282== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==5282== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==5282== Command: ./src/a.out
==5282==
--file opening error--
==5282==
==5282== HEAP SUMMARY:
==5282==   in use at exit: 0 bytes in 0 blocks
==5282==   total heap usage: 14 allocs, 14 frees, 83,971 bytes allocated
==5282==
==5282== All heap blocks were freed -- no leaks are possible
==5282==
==5282== For lists of detected and suppressed errors, rerun with: -s
==5282== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
kirill@kirill-IdeaPad-3:~/2_course/OPP0/1$
```

Рисунок 5 – Результат анализа с помощью Valgrind

По результатам динамического анализа утечек памяти выявлено не было. Вся выделенная память была освобождена.