



МИНОБРНАУКИ РОССИИ

*Федеральное государственное бюджетное образовательное учреждение
высшего образования*

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий (ИТ)

Кафедра Математического обеспечения и стандартизации информационных
технологий (МОСИТ)

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 4
по дисциплине
«Тестирование и верификация программного обеспечения»**

Тема: «Анализаторы кода»

Выполнил студенты группы ИКБО-20-23

Комисарик М. А.,

Принял

Чернов Е. А.

Практическая работа
выполнена

«__» 2025 г.

(подпись студента)

«Зачтено»

«__» 2025 г.

*(подпись
руководителя)*

Москва 2025

СОДЕРЖАНИЕ

1 ВЫПОЛНЕНИЕ ЗАДАНИЯ.....	3
ЗАКЛЮЧЕНИЕ	15

ВЫПОЛНЕНИЕ ЗАДАНИЯ

1 Проекты

1.1 Проект 1

Репозиторий представляет из себя сборник учебных заданий на языке Python по соответствующему предмету. Для анализа был взят файл class/ex2.py, который выводит в окно 3D рисованный шар с простым освещением.

Ссылка на репозиторий: <https://github.com/Krezerenko/Python>

1.2 Проект 2

Репозиторий представляет из себя 4 домашних задания по предмету конфигурационное управление. Выбранный проект – 4 задание. Проект представляет из себя 2 исполняемых файла: ассемблер и интерпретатор. Для ассемблера пишется исходный код на учебном языке, на выход он выдает байт код файл для интерпретатора. Интерпретатор выполняет написанный код в виртуальной памяти и выводит результат в выходной файл.

Ссылка на репозиторий: https://github.com/Krezerenko/Config_Homework

2 Статический анализ кода

2.1 Проект 1

Для выполнения статического анализа были выбраны статические анализаторы SonarQube, pylint и flake8.

Добавленные ошибки:

Первая ошибка: взятие корня из отрицательного числа (Рисунок 1).

```
if circle_distance >= 0: # ошибка 1
    return 0, 0, 0
z = -math.sqrt(circle_distance) + posZScaled
```

Рисунок 1 – Ошибка 1

Вторая ошибка: замена конвертации в tuple конвертацией в str (Рисунок 2).

```
    . . .
    return str(multiplier * c for c in color) # ошибка 2
```

Рисунок 2 – Ошибка 2

Третья ошибка: возможное деление на ноль (Рисунок 3).

```
def int_base(x): # simple (0%) 2 usages - Krezon
    # if x <= 0.01: return 0 # ошибка 3
    return math.exp(-1 / x)
```

Рисунок 3 – Ошибка 3

Четвертая ошибка: ошибка в названии переменной (Рисунок 4).

```
relative_ls_power = is_lit * 1 / ls_distance_squared * lightPower # ошибка 4
```

Рисунок 4 – Ошибка 4

Пятая ошибка: пропущенное двоеточие после объявления функции (Рисунок 5).

```
def sh(x, y) # ошибка 5 2 usages - Krezon *
    # Ваш код здесь:
    off_x = x - posXScaled
    off_y = y - posYScaled
```

Рисунок 5 – Ошибка 5

2.1.1 SonarQube

На рисунке 6 представлен отчет SonarQube перед добавлением ошибок.

Анализатор не выявил никаких проблем в коде.

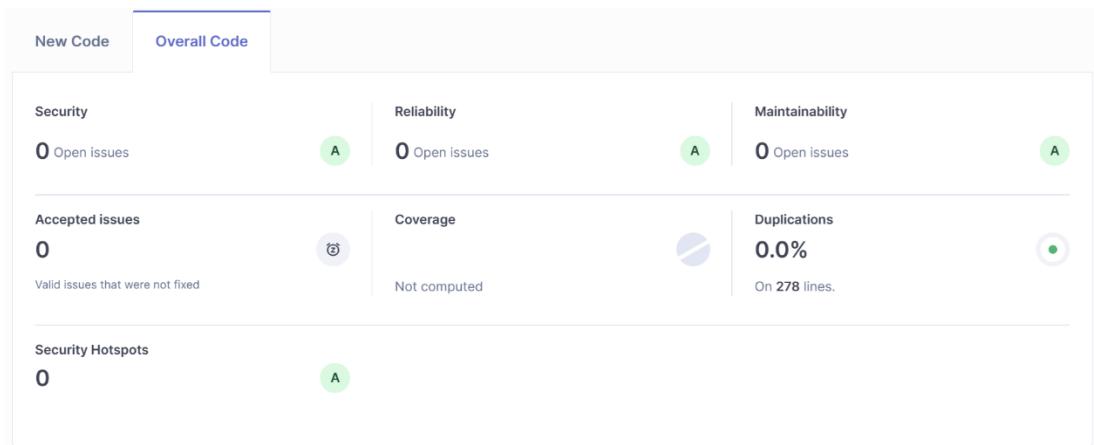


Рисунок 6 – Отчет SonarQube перед добавлением ошибок

На рисунке 7 представлен отчет SonarQube после добавления ошибок. Как видно из логов (Рисунок 8), анализатор просто не смог начать работу из-за ошибок синтаксиса.

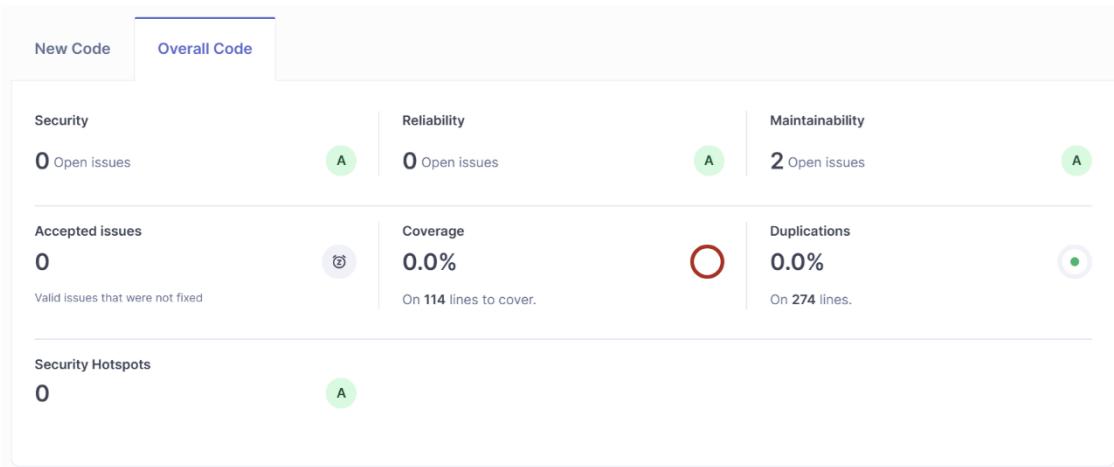


Рисунок 7 – Отчет SonarQube после добавления ошибок, часть 1

```

ERROR: Unable to parse file: ex2.py
ERROR: Parse error at line 79 column 24:

    73:
    74: lightPower = 0.2
    75: lsXScaled = lightSourceX / windowWidth
    76: lsYScaled = lightSourceY / windowHeight
    77: lsZScaled = lightSourceZ / windowWidthZ
    78:
→   def sh(x, y)
    80:
    81:     off_x = x - posXscaled
    82:     off_y = y - posYscaled
    83:     circle_distance = radSquare - off_x**2 - off_y**2
    84:     color = ballColor
    85:

INFO: 1/1 source file has been analyzed
INFO: Finished step rules execution in 329ms
  
```

Рисунок 8 – Отчет SonarQube после добавления ошибок, часть 2

2.1.2 pylint

На рисунке 9 представлена часть отчета pylint перед добавлением ошибок.

```
PS C:\Users\krez0\Education\Sem5\Testing\Pract4\Results> cat .\proj1 pylint1.txt
***** Module ex2 ****
ex2.py:94:0: C0301: Line too long (112/100) (line-too-long)
ex2.py:103:0: C0301: Line too long (116/100) (line-too-long)
ex2.py:113:0: C0325: Unnecessary parens after '-' keyword (superfluous-parens)
ex2.py:143:0: C0304: Final newline missing (missing-final-newline)
ex2.py:1:0: C0114: Missing module docstring (missing-module-docstring)
ex2.py:5:0: C0103: Constant name "time" doesn't conform to UPPER_CASE naming style (invalid-name)
ex2.py:7:0: C0103: Constant name "zoom" doesn't conform to UPPER_CASE naming style (invalid-name)
ex2.py:8:0: C0103: Constant name "windowWidth" doesn't conform to UPPER_CASE naming style (invalid-name)
ex2.py:9:0: C0103: Constant name "windowHeight" doesn't conform to UPPER_CASE naming style (invalid-name)
ex2.py:10:0: C0103: Constant name "windowWidthZ" doesn't conform to UPPER_CASE naming style (invalid-name)
ex2.py:11:0: C0103: Constant name "tickRate" doesn't conform to UPPER_CASE naming style (invalid-name)
ex2.py:12:0: C0116: Missing function or method docstring (missing-function-docstring)
ex2.py:21:0: C0116: Missing function or method docstring (missing-function-docstring)
ex2.py:22:4: W0603: Using the global statement (global-statement)
ex2.py:30:0: C0116: Missing function or method docstring (missing-function-docstring)
ex2.py:31:4: W0603: Using the global statement (global-statement)
ex2.py:37:0: C0116: Missing function or method docstring (missing-function-docstring)
ex2.py:38:4: W0603: Using the global statement (global-statement)
ex2.py:40:8: E0602: Undefined variable 'lightSourceZ' (undefined-variable)
ex2.py:47:0: C0116: Missing function or method docstring (missing-function-docstring)
ex2.py:59:0: C0103: Constant name "rad" doesn't conform to UPPER_CASE naming style (invalid-name)
ex2.py:60:0: C0103: Constant name "posX" doesn't conform to UPPER_CASE naming style (invalid-name)
ex2.py:61:0: C0103: Constant name "posY" doesn't conform to UPPER_CASE naming style (invalid-name)
ex2.py:62:0: C0103: Constant name "posZ" doesn't conform to UPPER_CASE naming style (invalid-name)
```

Рисунок 9 – Отчет pylint перед добавлением ошибок

На рисунке 10 представлен вывод pylint после добавления ошибок. Как видно из лога, анализатор не смог начать работу из-за ошибок синтаксиса.

ხევბაზარი: ეს არის უმცირესი და უძლიერი განვითარების სამსახური, რომელიც მოწოდება საქართველოს მთელი ტერიტორიის უმცირესი მიმდინარეობის განვითარების მიზანით. ეს არის ერთ-ერთი უმცირესი მიმდინარეობის განვითარების მიზანით.

Рисунок 10 – Вывод `pylint` после добавления ошибок

2.1.3 flake8

На рисунке 11 представлен отчет flake8 перед добавлением ошибок.

```
PS C:\Users\krezo\Education\Sem5\Testing\Pract4\Results> cat .\proj1_flake1.txt
.\Proj1\ex2.py:12:1: E302 expected 2 blank lines, found 0
.\Proj1\ex2.py:13:80: E501 line too long (82 > 79 characters)
.\Proj1\ex2.py:21:1: E302 expected 2 blank lines, found 1
.\Proj1\ex2.py:25:80: E501 line too long (82 > 79 characters)
.\Proj1\ex2.py:30:1: E302 expected 2 blank lines, found 1
.\Proj1\ex2.py:37:1: E302 expected 2 blank lines, found 1
.\Proj1\ex2.py:47:1: E302 expected 2 blank lines, found 1
.\Proj1\ex2.py:58:1: E305 expected 2 blank lines after class or function definition, found 1
.\Proj1\ex2.py:79:1: E302 expected 2 blank lines, found 1
.\Proj1\ex2.py:94:80: E501 line too long (112 > 79 characters)
.\Proj1\ex2.py:95:80: E501 line too long (91 > 79 characters)
.\Proj1\ex2.py:102:80: E501 line too long (80 > 79 characters)
.\Proj1\ex2.py:103:80: E501 line too long (116 > 79 characters)
.\Proj1\ex2.py:112:80: E501 line too long (88 > 79 characters)
.\Proj1\ex2.py:115:80: E501 line too long (97 > 79 characters)
.\Proj1\ex2.py:120:1: E303 too many blank lines (3)
.\Proj1\ex2.py:121:17: E701 multiple statements on one line (colon)
.\Proj1\ex2.py:124:1: E305 expected 2 blank lines after class or function definition, found 1
.\Proj1\ex2.py:127:1: E302 expected 2 blank lines, found 1
.\Proj1\ex2.py:128:80: E501 line too long (90 > 79 characters)
.\Proj1\ex2.py:131:1: E302 expected 2 blank lines, found 1
.\Proj1\ex2.py:134:1: E302 expected 2 blank lines, found 1
.\Proj1\ex2.py:137:1: E302 expected 2 blank lines, found 1
.\Proj1\ex2.py:140:1: E302 expected 2 blank lines, found 1
.\Proj1\ex2.py:143:1: E305 expected 2 blank lines after class or function definition, found 1
.\Proj1\ex2.py:143:7: W292 no newline at end of file
```

Рисунок 11 – Отчет flake8 перед добавлением ошибок

На рисунке 10 представлен вывод flake8 после добавления ошибок. Как видно из лога, анализатор не смог начать работу из-за ошибок синтаксиса.

```
PS C:\Users\krezo\Education\Sem5\Testing\Pract4\Results> cat .\proj1_flake2.txt  
.\\ex2.py:79:16: E999 SyntaxError: expected ':'
```

Рисунок 12 – Вывод flake8 после добавления ошибок

2.1.4 Сравнение

Все анализаторы не могут проводить анализ кода при наличии в нем синтаксических ошибок.

SonarQube, по всей видимости, показывает только серьезные предупреждения, возможные ошибки и баги, в то время как flake8 и pylint выводят все несоответствия стандартам написания кода, даже если они только стилистические. Наиболее педантичным оказался pylint.

2.2 Проект 2

Для выполнения статического анализа были выбраны статические анализаторы cppcheck, компилятор clang++ и clang-tidy.

Добавленные ошибки:

Первая ошибка: деление на ноль (Рисунок 13).

```
for (int i = 0; i < tokens.size(); ++i)  
{  
    i / (i - i); // error 1  
    char name[] = "A";  
    name[0] = 'A' + i;  
    command->SetAttribute(name, value: tokens[i].c_str());  
}
```

Рисунок 13 – Ошибка 1

Вторая ошибка: пропущенная точка с запятой (Рисунок 14).

```
        out.push_back(word);  
        out.erase( Where: out.begin()) // error 2  
//      out.erase(out.begin());  
,
```

Рисунок 14 – Ошибка 2

Третья ошибка: пропущенный break в switch выражении (Рисунок 15).

```
switch (commandNum)
{
    case 26:
        BitToInt( & command, sizes: {5, 19, 26}, byteSize: 7, remainder: 6, & operands);
    //      break; // error 3
    case 4:
        BitToInt( & command, sizes: {5, 14, 26, 26}, byteSize: 9, remainder: 1, & operands);
        break;
    case 27:
```

Рисунок 15 – Ошибка 3

Четвертая ошибка: неправильное преобразование, static_cast на указателях (Рисунок 16).

```
unsigned char* m_memory;
// #define mem_i(addr) *reinterpret_cast<unsigned int*>(&m_memory[addr])
#define mem_i(addr) *static_cast<unsigned int*>(&m_memory[addr]) // error 4
```

Рисунок 16 – Ошибка 4

Пятая ошибка: передача изменяемого параметра как копию, а не по ссылке (Рисунок 17).

```
// static void ConvertCommand(std::bitset<MaxCommandSize> &command, std::vector<unsigned int>& operands);
static void ConvertCommand(std::bitset<MaxCommandSize> &command, std::vector<unsigned int> operands); // error 5
```

Рисунок 17 – Ошибка 5

2.2.1 cppcheck

До внесения ошибок отчет анализатора пуст.

На рисунке 18 представлен отчет cppcheck после добавления ошибок.

```
PS C:\Users\krezo\Education\Sem5\Testing\Pract4\Results> cat .\proj2_cppcheck2.txt
src\Assembler.cpp:76:11: error: Division by zero. [zerodiv]
    i / (i - i); // error 1
    ^
src\Interpreter.cpp:40:29: error: Out of bounds access in expression 'operands[0]' because 'operands' is empty. [containerOutOfBounds]
    switch (operands[0])
    ^
```

Рисунок 18 – Отчет cppcheck после добавления ошибок

На отчете указаны ошибки деления на 0, а также доступ к запрещенной памяти.

2.2.2 clang++

До внесения ошибок отчет анализатора пуст.

На рисунке 19 представлена часть отчета clang++ после добавления ошибок.

```
PS C:\Users\krezo\Education\Sem5\Testing\Pract4\Results> cat .\proj2_clang2.txt
src\Assembler.cpp:107:27: error: expected ';' after expression
  107 |     out.erase(out.begin()) // error 2
      |           ^
      |
      ;
1 error generated.
src\Interpreter.cpp:43:21: error: static_cast from 'unsigned char *' to 'unsigned int *' is not allowed
  43 |             mem_i(operands[2]) = operands[1];
      |             ^~~~~~
src\Interpreter.h:16:26: note: expanded from macro 'mem_i'
  16 | #define mem_i(addr) *static_cast<unsigned int*>(&m_memory[addr]) // error 4
      |             ^~~~~~
src\Interpreter.cpp:52:21: error: static_cast from 'unsigned char *' to 'unsigned int *' is not allowed
  52 |             mem_i(operands[2] + operands[3]) = (mem_i(operands[1]) >> mem_i(operands[4]) % (sizeof(int) * 8)
) |
      |             ^~~~~~
src\Interpreter.h:16:26: note: expanded from macro 'mem_i'
  16 | #define mem_i(addr) *static_cast<unsigned int*>(&m_memory[addr]) // error 4
      |             ^~~~~~
src\Interpreter.cpp:52:57: error: static_cast from 'unsigned char *' to 'unsigned int *' is not allowed
  52 |             mem_i(operands[2] + operands[3]) = (mem_i(operands[1]) >> mem_i(operands[4]) % (sizeof(int) * 8)
) |
      |             ^~~~~~
src\Interpreter.h:16:26: note: expanded from macro 'mem_i'
```

Рисунок 19 – Часть отчета clang++ после добавления ошибок

В отчете указана ошибка отсутствия точки с запятой, а также ошибка неправильного использования static_cast.

2.2.3 clang-tidy

На рисунке 20 представлен отчет clang-tidy до добавления ошибок.

```
PS C:\Users\krezo\Education\Sem5\Testing\Pract4\Results> cat .\proj2_clang-tidy1.txt
C:\Users\krezo\Education\Sem5\Testing\Pract4\Proj2\src\Assembler.cpp:10:16: warning: use a trailing return type for this function [modernize-use-trailing-return-type]
  10 | int Assembler::Compile(const std::string &sourcePath, const std::string &binPath, const std::string &logPath)
      |   ^
      | auto
      |   ^~~~ → int
C:\Users\krezo\Education\Sem5\Testing\Pract4\Proj2\src\Assembler.cpp:10:24: warning: 3 adjacent parameters of 'Compile' of similar type ('const std::string &') are easily swapped by mistake [bugprone-easily-swappable-parameters]
  10 | int Assembler::Compile(const std::string &sourcePath, const std::string &binPath, const std::string &logPath)
      |   ^
      |   ^~~~~~
C:\Users\krezo\Education\Sem5\Testing\Pract4\Proj2\src\Assembler.cpp:10:43: note: the first parameter in the range is 'sourcePath'
  10 | int Assembler::Compile(const std::string &sourcePath, const std::string &binPath, const std::string &logPath)
      |   ^
      |   ^~~~~~
C:\Users\krezo\Education\Sem5\Testing\Pract4\Proj2\src\Assembler.cpp:10:102: note: the last parameter in the range is 'logPath'
  10 | int Assembler::Compile(const std::string &sourcePath, const std::string &binPath, const std::string &logPath)
      |   ^
      |   ^~~~~~
C:\Users\krezo\Education\Sem5\Testing\Pract4\Proj2\src\Assembler.cpp:30:27: warning: statement should be inside braces [readability-braces-around-statements]
  30 |         if (words.empty()) continue;
      |         ^
      |         {
C:\Users\krezo\Education\Sem5\Testing\Pract4\Proj2\src\Assembler.cpp:60:36: warning: 26 is a magic number; consider replacing it with a named constant [cppcoreguidelines-avoid-magic-numbers,readability-magic-numbers]
  60 |         command->SetAttribute("A", 26);
      |         ^
C:\Users\krezo\Education\Sem5\Testing\Pract4\Proj2\src\Assembler.cpp:68:36: warning: 27 is a magic number; consider replacing it with a named constant [cppcoreguidelines-avoid-magic-numbers,readability-magic-numbers]
  68 |         command->SetAttribute("A", 27);
      |         ^
C:\Users\krezo\Education\Sem5\Testing\Pract4\Proj2\src\Assembler.cpp:72:36: warning: 10 is a magic number; consider replacing
```

Рисунок 20 – Часть отчета clang-tidy до добавления ошибок

В отчете представлено множество предупреждений о качестве и стиле кода, нет ошибок.

На рисунке 21 представлена часть отчета clang-tidy после добавления ошибок.

Рисунок 21 – Часть отчета clang-tidy после добавления ошибок

В отчете указано множество предупреждений, так же как и до добавления ошибок в код, а также указаны ошибки, которые указал clang++.

2.2.4 Сравнение

Все анализаторы справились с поиском ошибок на этапе компиляции. По количеству ошибок отчеты cppcheck и clang++ схожи, однако они обнаружили разные типы ошибок. clang-tidy кроме ошибок компиляции выводил также много предупреждений, оценивая качество и стиль кода.

3 Динамический анализ кода

3.1 Проект 2

Для выполнения динамического анализа были выбраны санитайзеры address и undefined, а также анализатор Valgrind.

Добавленные ошибки:

Первая ошибка: утечка памяти (Рисунок 22).

```
for (int i = 0; i < tokens.size(); ++i)
{
    char* leak = new char; // error 1
    char name[] = "A";
    name[0] = 'A' + i;
    command->SetAttribute(name, value: tokens[i].c_str());
}
```

Рисунок 22 – Ошибка 1

Вторая ошибка: пропущенный break в switch выражении (Рисунок 23).

```
switch (commandNum)
{
    case 26:
        BitToInt( & command, sizes: {5, 19, 26}, byteSize: 7, remainder: 6, & operands);
        break; // error 3
    case 4:
        BitToInt( & command, sizes: {5, 14, 26, 26}, byteSize: 9, remainder: 1, & operands);
        break;
    case 27:
```

Рисунок 23 – Ошибка 2

Третья ошибка: передача изменяемого параметра как копию, а не по ссылке (Рисунок 24).

```
// static void ConvertCommand(std::bitset<MaxCommandSize> &command, std::vector<unsigned int>& operands);
static void ConvertCommand(std::bitset<MaxCommandSize> &command, std::vector<unsigned int> operands); // error 5
static void
```

Рисунок 24 – Ошибка 3

3.1.1 Санитайзеры

На рисунке 25 представлен отчет санитайзеров до добавления ошибок.

```

PS C:\Users\krezo\Education\Sem5\Testing\Pract4\Results> cat .\proj2_sam1.txt
/mnt/c/Users/krezo/Education/Sem5/Testing/Pract4/Proj2/src/Interpreter.cpp:43:40: runtime error: store to misaligned address
0x7776467ff80f for type 'unsigned int', which requires 4 byte alignment
0x7776467ff80f: note: pointer points here
be be
^
/mnt/c/Users/krezo/Education/Sem5/Testing/Pract4/Proj2/src/Interpreter.cpp:52:57: runtime error: load of misaligned address
0x7776467ff80f for type 'unsigned int', which requires 4 byte alignment
0x7776467ff80f: note: pointer points here
be be be be 05 00 00 00 05 01 00 00 00 be be
^
/mnt/c/Users/krezo/Education/Sem5/Testing/Pract4/Proj2/src/Interpreter.cpp:52:79: runtime error: load of misaligned address
0x7776467ff815 for type 'unsigned int', which requires 4 byte alignment
0x7776467ff815: note: pointer points here
00 00 05 05 01 00 00 00 be be
^
/mnt/c/Users/krezo/Education/Sem5/Testing/Pract4/Proj2/src/Interpreter.cpp:53:30: runtime error: load of misaligned address
0x7776467ff80f for type 'unsigned int', which requires 4 byte alignment
0x7776467ff80f: note: pointer points here
be be be be 05 00 00 00 05 01 00 00 00 be be
^
/mnt/c/Users/krezo/Education/Sem5/Testing/Pract4/Proj2/src/Interpreter.cpp:53:71: runtime error: load of misaligned address
0x7776467ff815 for type 'unsigned int', which requires 4 byte alignment
0x7776467ff815: note: pointer points here
00 00 05 05 01 00 00 00 be be
^
/mnt/c/Users/krezo/Education/Sem5/Testing/Pract4/Proj2/src/Interpreter.cpp:52:54: runtime error: store to misaligned address
0x7776467ff819 for type 'unsigned int', which requires 4 byte alignment
0x7776467ff819: note: pointer points here
01 00 00 00 be be
^

```

Рисунок 25 – Отчет санитайзеров до добавления ошибок

В отчете указано, что программа нарушает правила записи значений, сдвигая их не на целое число их размера при записи. В случае выполнения интерпретатора с виртуальной памятью и другими правилами записи это допустимо и не является ошибкой.

На рисунке 26 представлен отчет санитайзеров после добавления ошибок.

```

PS C:\Users\krezo\Education\Sem5\Testing\Pract4\Results> cat .\proj2_san2.txt
/usr/include/c++/13/bits/stl_vector.h:1129:34: runtime error: reference binding to null pointer of type 'value_type'
/mnt/c/Users/krezo/Education/Sem5/Testing/Pract4/Proj2/src/Interpreter.cpp:40:13: runtime error: load of null pointer of type 'value_type'
AddressSanitizer:DEADLYSIGNAL
=====
=7871==ERROR: AddressSanitizer: SEGV on unknown address 0x000000000000 (pc 0x6369da0e7dfa bp 0x000000000000 sp 0x7fff0bbdc410 T0)
=7871==The signal is caused by a READ memory access.
=7871==Hint: address points to the zero page.
#0 0x6369da0e7dfa in Interpreter::Execute(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > const&, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > const&, unsigned int, unsigned int) (/mnt/c/Users/krezo/Education/Sem5/Testing/Pract4/Proj2/build/Release/int+0x49dfa) (BuildId: e8137731cb41ca932f7ee321cc8e96d036fa92e0)
#1 0x6369da0d91a0 in main (/mnt/c/Users/krezo/Education/Sem5/Testing/Pract4/Proj2/build/Release/int+0x3b1a0) (BuildId: e8137731cb41ca932f7ee321cc8e96d036fa92e0)
#2 0xe8edb22a1c9 in __libc_start_main ..//sysdeps/nptl/libc_start_call_main.h:58
#3 0x7e8edb22a28a in __libc_start_main_impl ..//csu/libc-start.c:360
#4 0x6369da0d94c4 in _start (/mnt/c/Users/krezo/Education/Sem5/Testing/Pract4/Proj2/build/Release/int+0x3b4c4) (BuildId: e8137731cb41ca932f7ee321cc8e96d036fa92e0)

AddressSanitizer can not provide additional info.
SUMMARY: AddressSanitizer: SEGV (/mnt/c/Users/krezo/Education/Sem5/Testing/Pract4/Proj2/build/Release/int+0x49dfa) (BuildId: e8137731cb41ca932f7ee321cc8e96d036fa92e0) in Interpreter::Execute(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > const&, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > const&, unsigned int, unsigned int)
=7871==ABORTING

```

Рисунок 26 – Отчет санитайзеров после добавления ошибок

В отчете указана ошибка доступа к неинициализированному вектору (связано с ошибкой 3).

3.1.2 Valgrind

На рисунке 27 представлен отчет Valgrind до добавления ошибок.

```
PS C:\Users\krezo\Education\Sem5\Testing\Pract4\Results> cat .\proj2_valgrind1_2.txt
==7781= Memcheck, a memory error detector
==7781= Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==7781= Using Valgrind-3.22.0 and LibVEX; rerun with -h for copyright info
==7781= Command: ./build/Release/int ./Test/bin.edc ./Test/out.xml 0 50
==7781= Parent PID: 5627
==7781=
==7781= Conditional jump or move depends on uninitialised value(s)
==7781=     at 0x11003C: void std::bitset<8U>::__M_copy_to_string<char, std::char_traits<char>, std::allocator<char>>(std::
==7781=     _cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>&, char, char) const (in /mnt/c/Users/krezo/Educat
==7781=     ion/Sem5/Testing/Pract4/Proj2/build/Release/int)
==7781=     by 0x10E350: Interpreter::Execute(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>
==7781=     > const&, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>> const&, unsigned int, unsigned int
==7781=     ) (in /mnt/c/Users/krezo/Education/Sem5/Testing/Pract4/Proj2/build/Release/int)
==7781=     by 0x10CDA6: main (in /mnt/c/Users/krezo/Education/Sem5/Testing/Pract4/Proj2/build/Release/int)
==7781= Uninitialised value was created by a heap allocation
==7781=     at 0x48485C3: operator new[](unsigned long) (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==7781=     by 0x10D141: Interpreter::Interpreter() (in /mnt/c/Users/krezo/Education/Sem5/Testing/Pract4/Proj2/build/Release
==7781=     /int)
==7781=     by 0x10CCF5: main (in /mnt/c/Users/krezo/Education/Sem5/Testing/Pract4/Proj2/build/Release/int)
==7781=
==7781=
==7781= HEAP SUMMARY:
==7781=     in use at exit: 0 bytes in 0 blocks
==7781=     total heap usage: 120 allocs, 120 frees, 67,209,233 bytes allocated
==7781=
==7781= All heap blocks were freed -- no leaks are possible
==7781=
==7781= For lists of detected and suppressed errors, rerun with: -s
==7781= ERROR SUMMARY: 36 errors from 1 contexts (suppressed: 0 from 0)
```

Рисунок 27 – Отчет Valgrind до добавления ошибок

На отчете указана проблема доступа к неинициализированной памяти. Ошибка исходной программы. При выводе памяти из указанного на входе промежутка виртуальной машины, неиспользованная память не обнуляется.

На рисунке 28 представлена часть отчета Valgrind после добавления ошибок.

```
PS C:\Users\krezo\Education\Sem5\Testing\Pract4\Results> cat .\proj2_valgrind2_2.txt
==7740= Memcheck, a memory error detector
==7740= Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==7740= Using Valgrind-3.22.0 and LibVEX; rerun with -h for copyright info
==7740= Command: ./build/Release/int ./Test/bin.edc ./Test/out.xml 0 50
==7740= Parent PID: 5627
==7740=
==7740= Invalid read of size 4
==7740=    at 0x10DD00: Interpreter::Execute(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>
> const&, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > const&, unsigned int, unsigned int
) (in /mnt/c/Users/krezo/Education/Sem5/Testing/Pract4/Proj2/build/Release/int)
==7740=    by 0x10CD86: main (in /mnt/c/Users/krezo/Education/Sem5/Testing/Pract4/Proj2/build/Release/int)
==7740= Address 0x0 is not stack'd, malloc'd or (recently) free'd
==7740=
==7740=
==7740= Process terminating with default action of signal 11 (SIGSEGV)
==7740= Access not within mapped region at address 0x0
==7740=    at 0x10DD00: Interpreter::Execute(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>
> const&, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > const&, unsigned int, unsigned int
) (in /mnt/c/Users/krezo/Education/Sem5/Testing/Pract4/Proj2/build/Release/int)
==7740=    by 0x10CD86: main (in /mnt/c/Users/krezo/Education/Sem5/Testing/Pract4/Proj2/build/Release/int)
==7740= If you believe this happened as a result of a stack
==7740= overflow in your program's main thread (unlikely but
==7740= possible), you can try to increase the size of the
==7740= main thread stack using the --main-stacksize= flag.
==7740= The main thread stack size used in this run was 8388608.
==7740=
==7740= HEAP SUMMARY:
==7740=     in use at exit: 67,191,256 bytes in 4 blocks
==7740=     total heap usage: 9 allocations, 5 frees, 67,191,324 bytes allocated
```

Рисунок 28 – Часть отчета Valgrind после добавления ошибок

На отчете указана ошибка утечки памяти, а также доступ к запрещенной памяти (ошибки 1 и 3).

3.1.3 Сравнение

Valgrind кроме самих ошибок также выводит статистику выполнения программы: количество использованной программой памяти. Также анализ Valgrind выявил больше проблем в коде. Кроме того, для использования Valgrind нет необходимости перекомпилировать исполняемые файлы.

ЗАКЛЮЧЕНИЕ

В ходе проделанной работы на проектах в двух разных языках были опробованы и сравнение различные статические и динамические анализаторы кода. Были получены навыки работы с статическими и динамическими анализаторами, а также навыки анализа отчетов анализаторов.