МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ

по дисциплине «Тестирование и верификация программного обеспечения»

Практическая работа №4

Студенты группы *ИКБО-50-23*, *Иващенко А.В.* _____ (подпись)

Преподаватель *Ильичев Г. П.* _____ (подпись)

Отчет представлен «___»_____202__г.

Москва 2025 г.

1. Цели и задачи практической работы

Цель работы: ознакомиться с основными принципами и методами использования статических и динамических анализаторов кода для раннего выявления ошибок и потенциальных уязвимостей, что позволит повысить качество, безопасность и надёжность программного обеспечения.

Для достижения поставленной цели работы студентам необходимо выполнить ряд задач:

1. Изучить теоретические основы статического и динамического анализа кода.

2. Ознакомиться с популярными инструментами статического анализа (например, ESLint, Pylint, Checkmarx, SonarQube, FindBugs, TSLint, Cppcheck) и динамического анализа (например, Valgrind, DynamoRIO, Java VisualVM, Burp Suite, OWASP ZAP).

3. Применить выбранные анализаторы к ранее разработанным учебным проектам на разных языках программирования.

4. Провести анализ исходного кода до и после внесения целенаправленных ошибок, оценить адекватность обнаружения дефектов.

5. Сформировать детальный отчёт с критическим анализом результатов, выводами о преимуществах и ограничениях каждого подхода.

2. Теоретический материал

2.1 Статический анализ кода

Статический анализ — это метод анализа программного кода без его исполнения. Его основная задача — обнаружить потенциальные ошибки, нарушения стандартов кодирования, неэффективные или опасные конструкции и утечки ресурсов непосредственно в исходном коде.

Основные характеристики:

1. Позволяет выявить ошибки ещё до запуска программы, что снижает затраты на их исправление.

2. Результаты анализа оформляются в виде отчётов, включающих список найденных проблем, ссылки на документацию, описание потенциальных рисков и рекомендации по исправлению.

2.2 Динамический анализ кода

Динамический анализ представляет собой процесс изучения поведения программы во время её исполнения. Этот метод позволяет выявить ошибки, которые не обнаруживаются статическим анализом, например, утечки памяти, ошибки исполнения и проблемы с производительностью.

Основные характеристики:

1. Позволяет оценить, как программа работает в условиях, приближенных к боевым, выявляя проблемы, связанные с взаимодействием модулей и ресурсами системы.

2. Результаты динамического анализа оформляются в виде отчётов, включающих данные о профилировании, использовании памяти, времени выполнения и других аспектах работы программы.

3. Ход работы

3.1 Проекты для тестирования

Возьмем код на языке C++ с проекта прошлого года, реализующий работу с масками. Включение/выключение битов, вывод числа в двоичном формате, а также сортировка чисел, считывая с файлов.

Листинг 1 – Проект на языке C++

```
#include <iostream>
#include <bitset>
#include <fstream>
#include <chrono>

using namespace std;

void fivebit() {
    unsigned char x = 255;
    unsigned char maska = 1;
    x = x & (~(maska << 4));
    cout << int(x);
}

void sevenbit() {
    unsigned char x = 16;
    unsigned char maska = 1;
    x = x | (maska << 6);
    cout << int(x);
}

void listing() {
    unsigned int x = 25;
    const int n = sizeof(int)*8;
    unsigned int maska = (1u << (n - 1));
    cout << "Initial appearance of the mask: " <<
bitset<n>(maska) << endl;
    cout << "Result: ";
    for (int i = 1; i <= n; i++)
    {
        cout << ((x & maska) >> (n - i));
        maska = maska >> 1;
    }
    cout << endl;
}


void asort() {
```

```
    const int A[] = {4,1,0,3,5,7,2,6};
    unsigned char B = 0;
    unsigned char maska = 1;
    for (int i = 0; i < 8; i++)
        B = B | (maska << A[i]);

    for (int i = 0; i < 8; i++)
        if ((maska << i) & B)
            cout << i << " ";
}
void bsort() {
    const int A[] = {8,23,31,6,15,3,0,48,9,29};
    unsigned long long B = 0;
    unsigned long long maska = 1;
    for (int i = 0; i < 10; i++)
        B = B | (maska << A[i]);

    for (int i = 0; i < 64; i++)
        if ((maska << i) & B)
            cout << i << " ";
}
void csort() {
    int A[] = {8,41,32,6,13,3,9,4,20,1};
    unsigned char B[] = {0,0,0,0,0,0,0,0};
    unsigned long long maska = 1;
    int size = sizeof(A)/sizeof(A[0]);;

    for (int i = 0; i < size; i++)
        B[A[i] / 8] = B[A[i] / 8] | (maska << A[i] % 8);

    for (int i = 0; i < 8; i++)
        for (int j = 0; j < 8; j++)
            if ((maska << j) & B[i])
                cout << i * 8 + j << " ";
}

void filesort() {
    fstream f("inp.txt");
    unsigned char maska = 1;
    unsigned char* B = new unsigned char[8388608 / 8];
    for (unsigned long long i = 0; i < 8388608 / 8; i++)
        B[i] = 0;

    auto start = std::chrono::high_resolution_clock::now();
    for (unsigned long long i = 0; i < 8388608; i++)
    {
        unsigned long long num;
        f >> num;
```

```
        B[num / 8] = B[num / 8] | (maska << num % 8);
    }
    f.close();
    f.open("out.txt");
    for (unsigned long long i = 0; i < 8388608 / 8; i++)
        for (unsigned long long j = 0; j < 8; j++)
            if ((maska << j) & B[i])
                f << i * 8 + j << " ";
    f.close();
    auto stop = chrono::high_resolution_clock::now();
    auto                        duration                        =
chrono::duration_cast<chrono::milliseconds>(stop - start);
    cout  <<  "Whole  work  took  "  <<  duration.count()  <<  "
milliseconds\n";
    delete[] B;
}

int main() {
    int choice;
    while (true) {
        cout << "1. Five\n"
            << "2. Seven\n"
            << "3. Listing\n"
            << "4. Sorting (from 0 to 7)\n"
            << "5. Sorting (from 0 to 63)\n"
            << "6. Different sorting\n"
            << "7. File sorting\n"
            << "0. Exit\n";
        cout << "Choose option: ";
        cin >> choice;
        switch (choice) {
        case 1:
            fivebit();
            cout << endl;
            break;
        case 2:
            sevenbit();
            cout << endl;
            break;
        case 3:
            listing();
            break;
        case 4:
            asort();
            cout << endl;
            break;
        case 5:
            bsort();
```

```
            cout << endl;
            break;
        case 6:
            csort();
            cout << endl;
            break;
        case 7:
            filesort();
            break;
        case 0:
            return 0;
            break;
        default:
            cout << "Incorrect input!\n";
            break;
        }
    }
}
```

Также возьмем код на языке Python с проекта этого года, реализующий работу набор из пяти игр: Камень, Ножницы, Бумага; Орел или Решка; Висельник; Угадай число и Битва Кубиков.

Листинг 2 – Проект на языке Python

```
import random

def rock_paper_scissors():
    sp = {'1': "Scissors", '2': "Paper", '3': "Rock"}
    k1, chs = random.choice(list(sp.items()))
    chs = k1+chs
    x               =               input("Are                you
ready?\nRock..Paper..Scissors!\n").capitalize()
    if not(x in sp.values()):
        return "No..."
    k2 = next((k for k, v in sp.items() if v == x), None)
    x = k2+x
    print(x[1:]," VS ",chs[1:])
    if x == chs:
        return "Draw!"
    else:
        match(int(x[:1])):
            case 1:
                if int(chs[:1])==2:
                    return "You Win!"
                else:
                    return "You Lost!"
            case 2:
                if int(chs[:1])==3:
                    return "You Win!"
                else:
                    return "You Lost!"
            case 3:
                if int(chs[:1])==1:
                    return "You Win!"
                else:
                    return "You Lost!"
        return None


def head_or_tails():
    sp = ["Heads","Tails"]
    chs = random.choice(sp)
    x=input("Heads or Tails?\n").capitalize()
    if x==chs:
        return "Correct!"
    elif x in sp:
```

```
        return "You Lost!"
    else:
        return "...Huh? Anyway..You Lost!"

def guess_number():
    sp = random.randint(1, 200)
    print("Can you guess a number?")
    while True:
        x=int(input("Take a guess!\n"))
        if x>sp:
            print("Too much! Try lesser number.")
        elif x<sp:
            print("Not enough! Try bigger number.")
        else:
            return "Correct! Good job!"

def hangman():
    sp                                                          =
["determination","patience","bravery","justice","integrity","ki
ndness","perseverance"]
    chs = random.choice(sp)
    att = 7
    gs = set()
    print("Can you guess a word?")
    while att > 0:
        show = " ".join([letr if letr in gs else "_" for letr in
chs])
        print(show)
        x = input("Take a guess on letter: ")
        if x in chs:
            print("Correct letter!")
            gs.add(x)
            if all(ch in gs for ch in chs):
                return "You Win!\nCorrect word is " + chs
        else:
            print("Wrong letter...")
            att -= 1
    return "You couldn't guess word... Correct answer was " +
chs

def dice_battle():
    print("Ready to throw a dice?")
    hp = 1
    ehp = 1
    print("You both had 5 hp!")
    while hp>0 and ehp>0:
        input("Press Enter to throw a dice!")
        yu = random.randint(1, 6)
```

```
        en = random.randint(1, 6)
        print("You threw",yu,"and your enemy threw",en)
        if yu > en:
            print("You Won!")
            ehp-=1
        elif yu < en:
            print("You Lost..")
            hp-=1
        else:
           print("Draw!")
    if hp==0:
        return "You died...Game over!"
    else:
        return "Congratulations! Hp left... " + str(hp)
```

3.2 Статический анализ кода

3.2.1 Анализ кода на C++

Возьмем код на языке C++, введем 5 ошибок и используем 3 инструмента статического анализа кода:

FlawFinder – Уязвимости безопасности;

CppLint – Соответствие Google C++ Style Guide;

CppCheck – Ошибки управления памятью.

Рисунок 1 – CppCheck до введения ошибок

Рисунок 2 – CppCheck после введения ошибок

Обнаружена 1 ошибка, связанная с неиспользованной переменной. Также предварительно были объявлены предупреждения с использованием библиотек и ограниченным анализом.

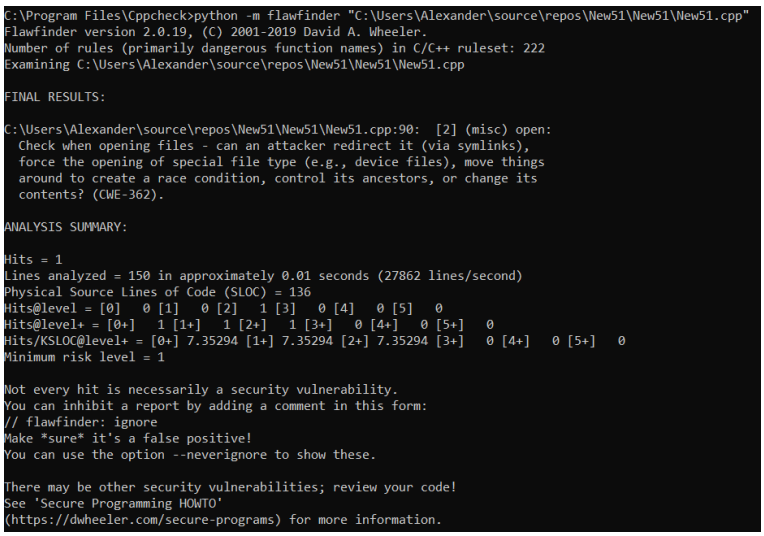

Рисунок 3 – FlawFinder до введения ошибок

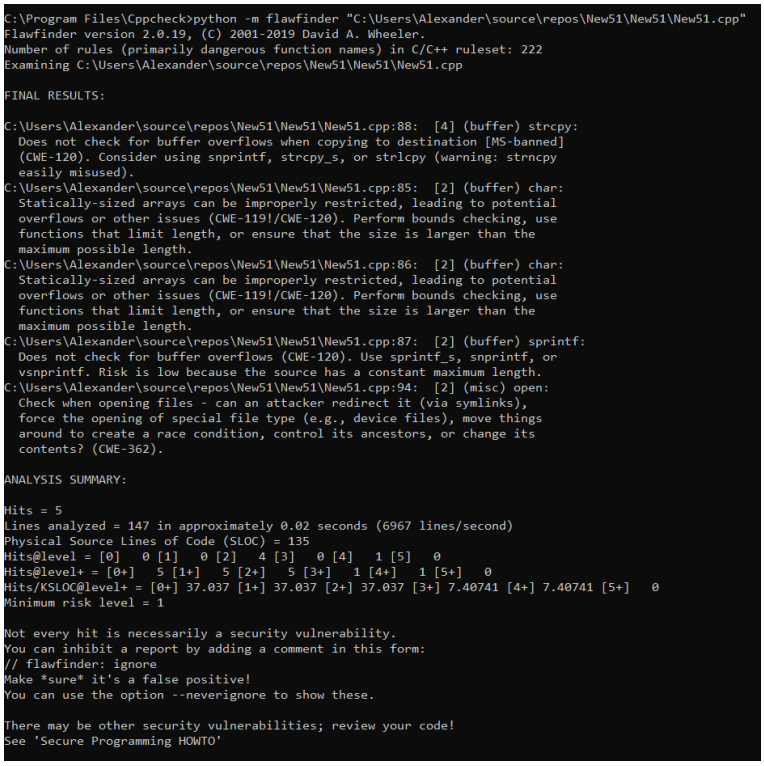

Рисунок 4 – FlawFinder после введения ошибок

Обнаружены 3 ошибки, связанных с небезопасным использованием переменных (в основном работа с длинной объектов). Предварительно было сделано предупреждение, связанное с открытием файла.

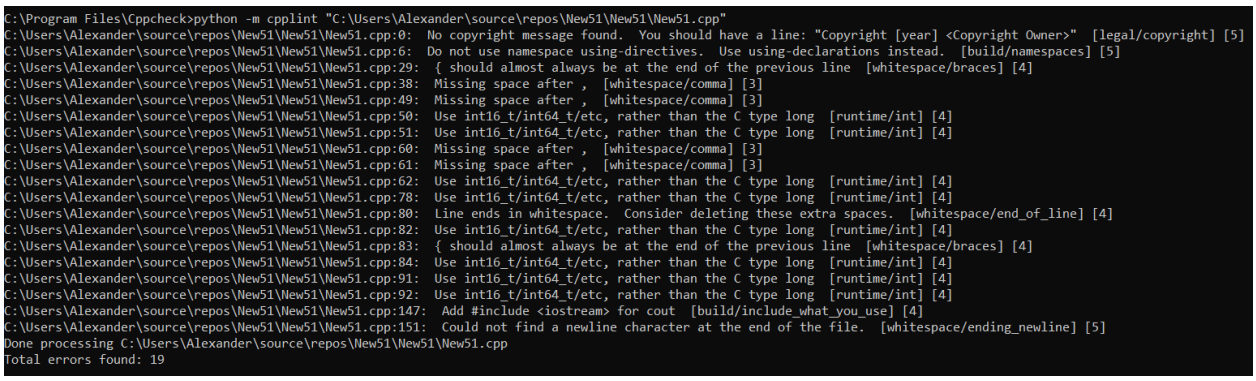

Рисунок 5 – CppLint до введения ошибок

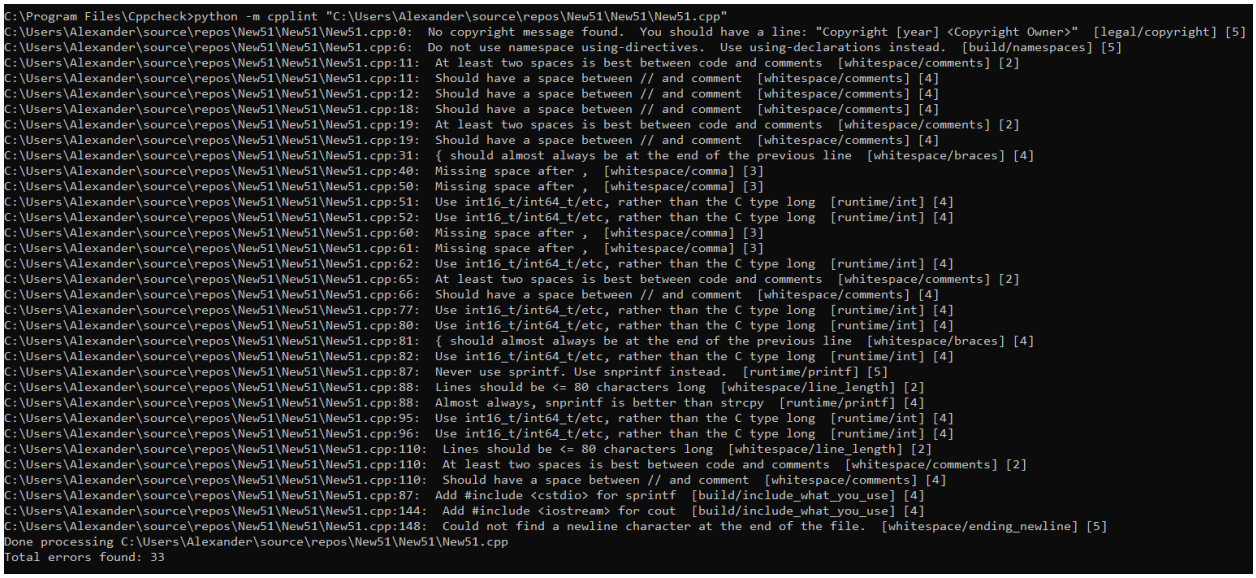

Рисунок 6 – CppLint после введения ошибок

Обнаружены дополнительно 14 ошибок в оформлении кода, в основном отсутствуют дополнительные пробелы/переходы на новую строку, тип данных у переменных. До добавления было 19 ошибок оформления.

3.2.2 Анализ кода на Python

Возьмем код на языке Python, введем 5 ошибок и используем 3 инструмента статического анализа кода:

PyLint – Стиль и архитектура;

Flake8 – Соответствие PEP8;

MyPy – Проверка типов данных.

```
C:\Users\Alexander\Downloads\РТУ МИРЭА\3 Курс\5 Семестр\ТиВПО\ПР2>python -m pylint module.py
************* Module module
module.py:8:0: C0325: Unnecessary parens after 'not' keyword (superfluous-parens)
module.py:95:0: W0311: Bad indentation. Found 11 spaces, expected 12 (bad-indentation)
module.py:100:0: C0305: Trailing newlines (trailing-newlines)
module.py:1:0: C0114: Missing module docstring (missing-module-docstring)
module.py:3:0: C0116: Missing function or method docstring (missing-function-docstring)
module.py:13:4: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
module.py:18:16: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
module.py:23:16: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
module.py:28:16: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
module.py:3:0: R0911: Too many return statements (9/6) (too-many-return-statements)
module.py:35:0: C0116: Missing function or method docstring (missing-function-docstring)
module.py:39:4: R1705: Unnecessary "elif" after "return", remove the leading "el" from "elif" (no-else-return)
module.py:46:0: C0116: Missing function or method docstring (missing-function-docstring)
module.py:58:0: C0116: Missing function or method docstring (missing-function-docstring)
module.py:78:0: C0116: Missing function or method docstring (missing-function-docstring)
module.py:96:4: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)

-------------------------------------------------------------------
Your code has been rated at 8.02/10 (previous run: 8.02/10, +0.00)
```

Рисунок 7 – PyLint до проверки

```
C:\Users\Alexander\Downloads\РТУ МИРЭА\3 Курс\5 Семестр\ТиВПО\ПР2>python -m pylint module.py
************* Module module
module.py:8:0: C0325: Unnecessary parens after 'not' keyword (superfluous-parens)
module.py:97:0: W0311: Bad indentation. Found 11 spaces, expected 12 (bad-indentation)
module.py:102:0: C0305: Trailing newlines (trailing-newlines)
module.py:1:0: C0114: Missing module docstring (missing-module-docstring)
module.py:3:0: C0116: Missing function or method docstring (missing-function-docstring)
module.py:7:8: E0602: Undefined variable 'inpoot' (undefined-variable)
module.py:13:4: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
module.py:18:16: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
module.py:23:16: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
module.py:28:16: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
module.py:3:0: R0911: Too many return statements (9/6) (too-many-return-statements)
module.py:35:0: C0116: Missing function or method docstring (missing-function-docstring)
module.py:39:4: R1705: Unnecessary "elif" after "return", remove the leading "el" from "elif" (no-else-return)
module.py:46:0: C0116: Missing function or method docstring (missing-function-docstring)
module.py:58:0: C0116: Missing function or method docstring (missing-function-docstring)
module.py:80:0: C0116: Missing function or method docstring (missing-function-docstring)
module.py:98:4: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
module.py:98:7: E1102: hp is not callable (not-callable)

-------------------------------------------------------------------
Your code has been rated at 6.79/10 (previous run: 8.02/10, -1.23)
```

Рисунок 8 – PyLint после проверки

Обнаружены 2 ошибки вызова неопределенной функции (inpoot()) и попытка вызова переменной как функции (hp()). Предварительно было выведено 16 предупреждений из-за излишних повторений и вызовов (в частности else и return).

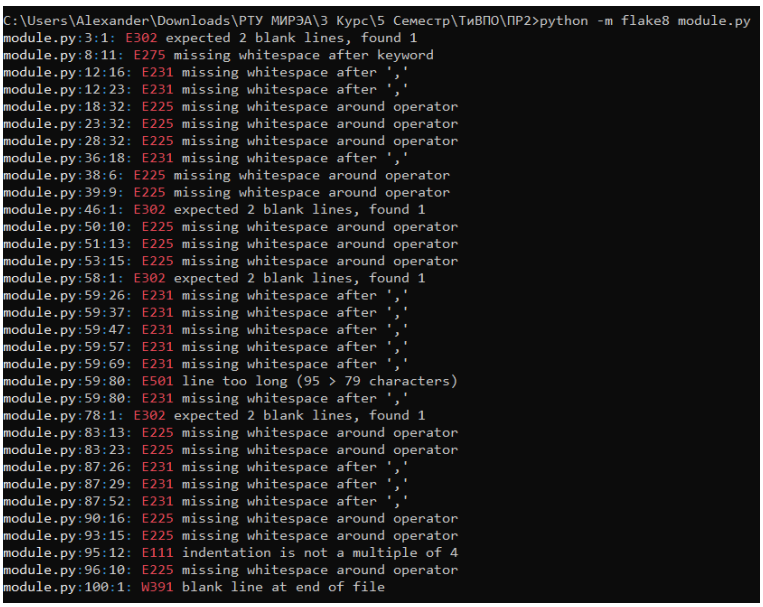

Рисунок 9 – Flake8 до проверки

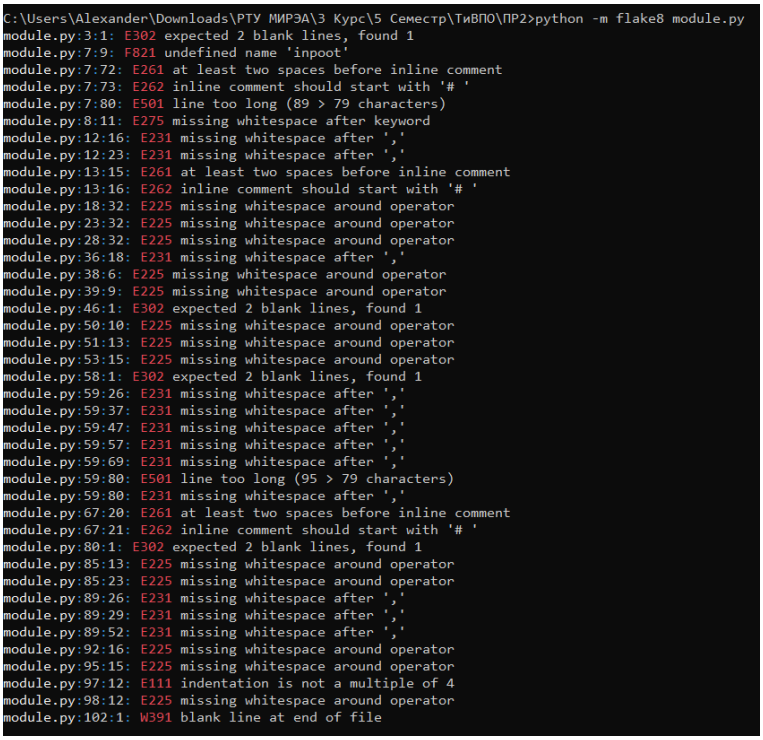

Рисунок 10 – Flake8 после проверки

Обнаружено 8 ошибок, связанных с оформлением кода согласно стилю PEP8 (лишние выводы на новую строку, ошибка inpoot). Предварительно было обнаружено 33 ошибки в оформлении кода, завязанных также на лишних выводах на новую строку, их отсутствия, а также отсутствия пробелов.

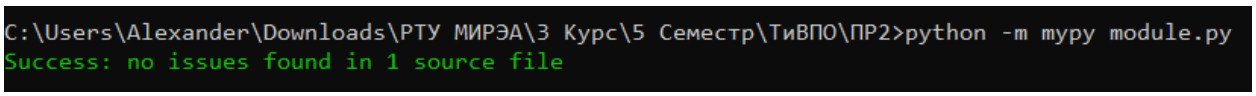

Рисунок 11 – MyPy до проверки

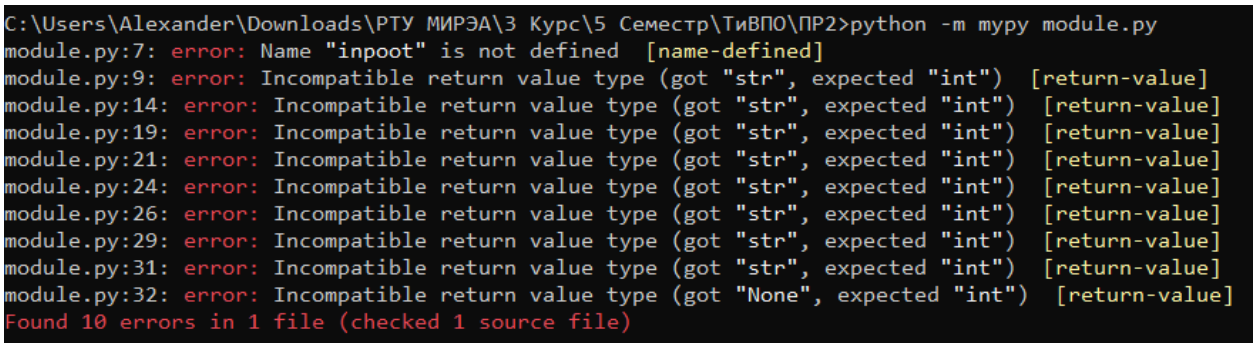

Рисунок 12 – MyPy после проверки

Обнаружено 10 ошибок, связанных с некорректным типом данных на выходе функции. Предварительно ошибок не было обнаружено.

3.3 Динамический анализ кода

3.3.1 Анализ кода на C++

Возьмем код на языке C++, введем несколько ошибок и используем инструмент статического анализа кода:

Встроенные в Visual Studio инструменты диагностики – Работа с памятью;

Реализуем работу с вводом данных с консоли.

В качестве ошибки для излишка памяти уберем закрытие файла в функции filesort() и удаление массива символов B[].

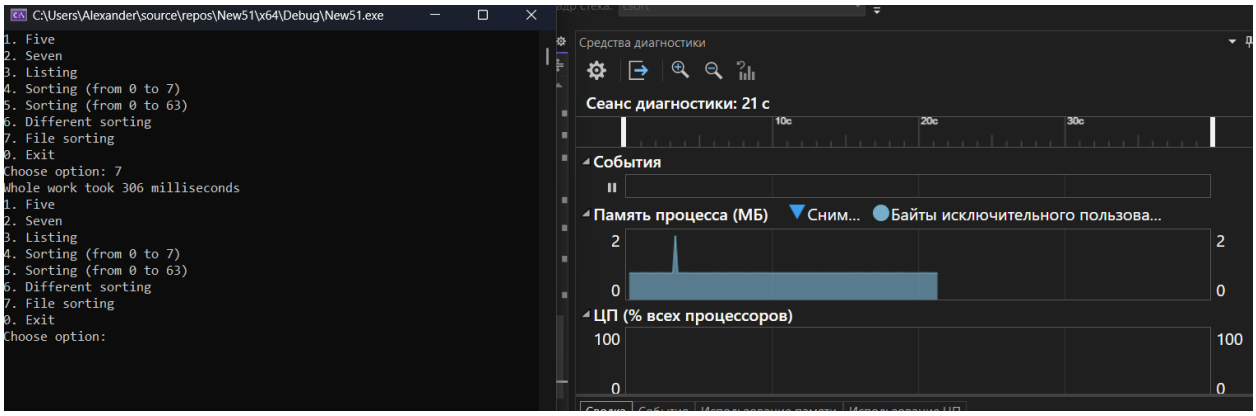

Рисунок 13 – Диагностика до введения ошибки

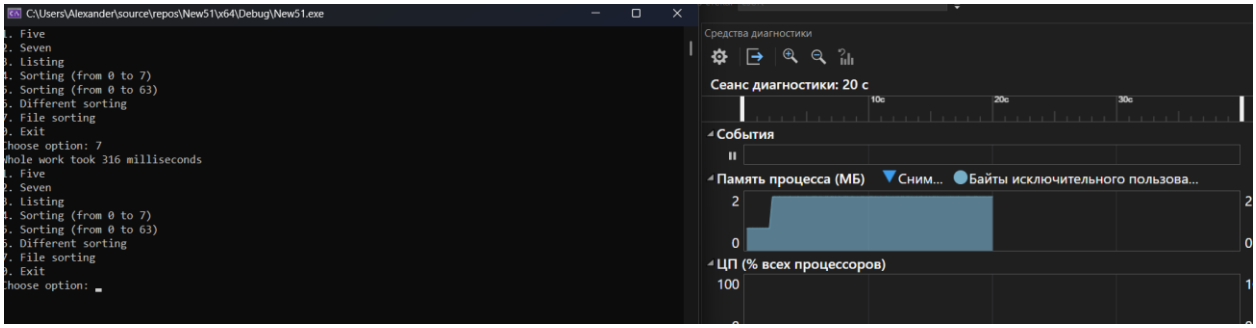

Рисунок 14 – Диагностика после введения ошибки

Как мы видим, после определенного выбора идет лишний расход памяти. По данному примеру легко определить точную причину данной ситуации.

3.3.2 Анализ кода на Python

Возьмем код на языке Python, введем несколько ошибок и используем инструмент статического анализа кода:

Pytest и Coverage – Проверка покрытия всего кода тестами;

Mprof – отслеживание расхода памяти.

Используем те же ошибки, что и при статическом анализе кода: вместо input – inpoot, в rock_paper_scissors() поменяем тип выходных данных, поставим сравнение строки и целого числа, обращение к hp как к функции.

Листинг 3 – Pytest

```
import pytest
from unittest.mock import patch
import module as a

@patch('builtins.input',return_value="Heads")
@patch("random.choice",return_value="Heads")
def test_hh_head_or_tails(mock_input, mock_choice):
    assert a.head_or_tails() == "Correct!"

@patch('builtins.input',return_value="Heads")
@patch("random.choice",return_value="Tails")
def test_ht_head_or_tails(mock_input, mock_choice):
    assert a.head_or_tails() == "You Lost!"

@patch('builtins.input',return_value="Tails")
@patch("random.choice",return_value="Heads")
def test_th_head_or_tails(mock_input, mock_choice):
    assert a.head_or_tails() == "You Lost!"

@patch('builtins.input',return_value="Tails")
@patch("random.choice",return_value="Tails")
def test_tt_head_or_tails(mock_input, mock_choice):
    assert a.head_or_tails() == "Correct!"

@patch('builtins.input',return_value="tails")
@patch("random.choice",return_value="Tails")
def   test_cc_head_or_tails(mock_input,   mock_choice):   #Check
capitalize
    assert a.head_or_tails() == "Correct!"

@patch('builtins.input',return_value="ITSPIKACHU")
def test_rr_head_or_tails(mock_input):
    assert a.head_or_tails() == "...Huh? Anyway..You Lost!"
```

```
@patch("random.randint",return_value=10)
@patch('builtins.input',return_value="10")
def  test_ft_guess_number(mock_input,  mock_randint,  capsys):
#First try
    res = a.guess_number()
    cap =  capsys.readouterr()
    assert res == "Correct! Good job!"
    assert "Can you guess a number?" in cap.out
    assert "Too much! Try lesser number." not in cap.out
    assert "Not enough! Try bigger number." not in cap.out

@patch("random.randint",return_value=10)
@patch('builtins.input',side_effect=["2","10"])
def  test_lt_guess_number(mock_input,  mock_randint,  capsys):
#Less try
    res = a.guess_number()
    cap = capsys.readouterr()
    assert res == "Correct! Good job!"
    assert "Can you guess a number?" in cap.out
    assert "Too much! Try lesser number." not in cap.out
    assert "Not enough! Try bigger number." in cap.out

@patch("random.randint",return_value=10)
@patch('builtins.input',side_effect=["12","10"])
def test_bt_guess_number(mock_input, mock_randint, capsys): #Big
try
    res = a.guess_number()
    cap = capsys.readouterr()
    assert res == "Correct! Good job!"
    assert "Can you guess a number?" in cap.out
    assert "Too much! Try lesser number." in cap.out
    assert "Not enough! Try bigger number." not in cap.out

@patch("random.randint",return_value=10)
@patch('builtins.input',side_effect=["100","1","10"])
def test_bt_guess_number(mock_input, mock_randint, capsys): #Big
try
    res = a.guess_number()
    cap = capsys.readouterr()
    assert res == "Correct! Good job!"
    assert  cap.out.splitlines()[1]  ==  "Too  much!  Try  lesser
number."
    assert  cap.out.splitlines()[2]  ==  "Not  enough!  Try  bigger
number."

#rps = rock paper scissorgshfk Ножницы короче
@patch('builtins.input',return_value="rock")
@patch('random.choice',return_value=('2','Paper'))
```

```
def test_cc_rps(mock_input,mock_choice):
    assert a.rock_paper_scissors()=="You Lost!"

@patch('builtins.input',return_value="Rock")
@patch('random.choice',return_value=('2','Paper'))
def test_l_rps(mock_input,mock_choice):
    assert a.rock_paper_scissors()=="You Lost!"

@patch('builtins.input',return_value="Scissors")
@patch('random.choice',return_value=('2','Paper'))
def test_w_rps(mock_input,mock_choice):
    assert a.rock_paper_scissors()=="You Win!"

@patch('builtins.input',return_value='Paper')
@patch('random.choice',return_value=('2','Paper'))
def test_d_rps(mock_input,mock_choice):
    assert a.rock_paper_scissors()=="Draw!"

@patch('builtins.input',return_value="Эскперимент не был
провальным")
@patch('random.choice',return_value=('2','Paper'))
def test_ri_rps(mock_input,mock_choice):
    assert a.rock_paper_scissors()=="No..."

@patch('builtins.input',return_value="qxzcdwu")
@patch('random.choice',return_value="bravery")
def test_l_hangman(mock_input,mock_choice,capsys):
    res = a.hangman()
    cap = capsys.readouterr()
    assert res == "You couldn't guess word... Correct answer was
bravery"
    assert "Can you guess a word?" in cap.out
    assert "Wrong letter..." in cap.out

@patch('builtins.input',side_effect=["a","b","v","e","r","y"])
@patch('random.choice',return_value="bravery")
def test_w_hangman(mock_choice,mock_input,capsys):
    res = a.hangman()
    cap = capsys.readouterr()
    assert res == "You Win!\nCorrect word is bravery"
    assert "Can you guess a word?" in cap.out
    assert "Wrong letter..." not in cap.out

@patch('builtins.input',side_effect=["Выкидываем
игру","5","f","c","d","q","a","b","v","e","r","y"])
@patch('random.choice',return_value="bravery")
def test_cl_hangman(mock_choice,mock_input,capsys):
    res = a.hangman()
```

```
    cap = capsys.readouterr()
    assert res == "You Win!\nCorrect word is bravery"
    assert "Can you guess a word?" in cap.out
    assert "Wrong letter..." in cap.out
    assert "Correct letter!" in cap.out
    assert "_ _ a _ _ _ _" in cap.out

@patch('builtins.input',return_value="")
@patch('random.randint',side_effect=[6,1])
def test_w_dice_battle(mock_input,mock_choice, capsys):
    res = a.dice_battle()
    cap = capsys.readouterr()
    assert res == "Congratulations! Hp left... 1"
    assert "You Won!" in cap.out
    assert "Ready to throw a dice?" in cap.out
    assert "You both had 5 hp!" in cap.out

@patch('builtins.input',return_value="")
@patch('random.randint',side_effect=[1,6])
def test_l_dice_battle(mock_input,mock_choice, capsys):
    res = a.dice_battle()
    cap = capsys.readouterr()
    assert res == "You died...Game over!"
    assert "You Lost.." in cap.out
    assert "Ready to throw a dice?" in cap.out
    assert "You both had 5 hp!" in cap.out

@patch('builtins.input',return_value="")
@patch('random.randint',side_effect=[1,1,6,1])
def test_dw_dice_battle(mock_input,mock_choice, capsys):
    res = a.dice_battle()
    cap = capsys.readouterr()
    assert res == "Congratulations! Hp left... 1"
    assert "You Won!" in cap.out
    assert "Ready to throw a dice?" in cap.out
    assert "You both had 5 hp!" in cap.out
    assert "Draw!" in cap.out
```

```
C:\Users\Alexander\Downloads\РТУ МИРЭА\3 Курс\5 Семестр\ТиВПО\ПР2>python -m coverage run tests.py

C:\Users\Alexander\Downloads\РТУ МИРЭА\3 Курс\5 Семестр\ТиВПО\ПР2>python -m coverage report
Name        Stmts   Miss  Cover
-------------------------------
module.py      83     77     7%
tests.py      135     70    48%
-------------------------------
TOTAL         218    147    33%

C:\Users\Alexander\Downloads\РТУ МИРЭА\3 Курс\5 Семестр\ТиВПО\ПР2>
```

Рисунок 15 – Pytest и Coverage до введения ошибки

```
========================================= short test summary info =========================================
FAILED tests.py::test_cc_rps - NameError: name 'inpoot' is not defined
FAILED tests.py::test_l_rps - NameError: name 'inpoot' is not defined
FAILED tests.py::test_w_rps - NameError: name 'inpoot' is not defined
FAILED tests.py::test_d_rps - NameError: name 'inpoot' is not defined
FAILED tests.py::test_ri_rps - NameError: name 'inpoot' is not defined
FAILED tests.py::test_w_dice_battle - TypeError: 'int' object is not callable
FAILED tests.py::test_l_dice_battle - TypeError: 'int' object is not callable
FAILED tests.py::test_dw_dice_battle - TypeError: 'int' object is not callable
====================================== 8 failed, 12 passed in 0.14s =======================================
```

Рисунок 16 – Pytest после введения ошибки

Тесты, как мы видим, неполностью покрывают наш модуль и не увидел все ошибки, из-за ошибок, возникающие до второстепенных. Однако несмотря на это pytest позволяет увидеть какие конкретно и в какой функции были допущены зарегистрированные ошибки.

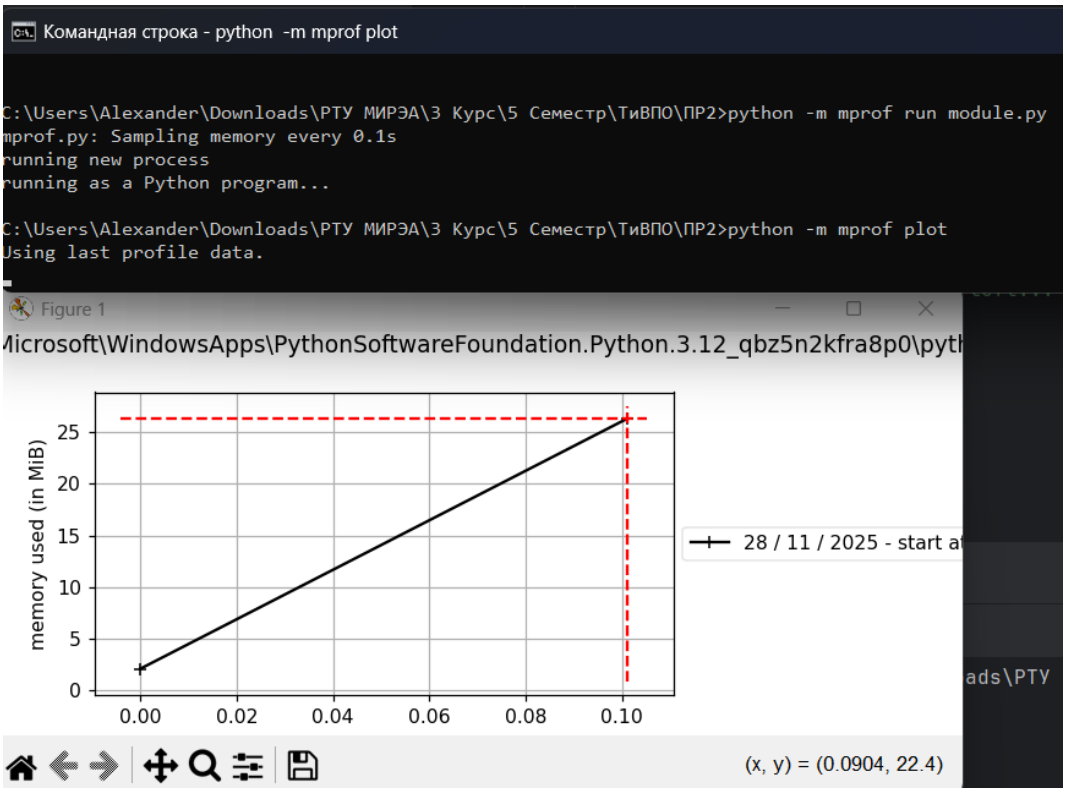

Рисунок 17 – вызов mprof

4. Заключение

В ходе практической работы были изучены и освоены методы анализа кода (Статический и Динамический) на основе уже написанных кодов прошлых лет на разных языках программирования. Используя разные библиотеки, мы провели Статический и Динамический анализ кода, сравнив результаты до и после введения ошибок. На основе полученных результатов мы выявили сильные и слабые стороны использования различных методов проверки кода.