МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА - Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИИТ)
Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ

по дисциплине «Тестирование и верификация программного обеспечения»

Практическая работа № 4
АНАЛИЗАТОРЫ КОДА

Студент группы *ИКБО-50-23, Враженко Д.О.* _____
(подпись)

Преподаватель *Ильичев Г.П.*

_____
(подпись)

Отчет представлен «___»_____2025 г.

Москва 2025 г.

1. ЦЕЛЬ И ЗАДАЧИ

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. КРАТКОЕ ОПИСАНИЕ ПРОЕКТОВ

2.1. Проект на Python (social_network.py)

Функциональность

Проект представляет собой упрощённую модель социальной сети, реализующую базовые функции для управления пользователями и их взаимодействиями. Основные возможности включают:

- регистрация новых пользователей с уникальной электронной почтой;
- добавление взаимных друзей между пользователями;
- публикация текстовых статусов с временными метками и счётчиком лайков;
- поиск пользователей по имени пользователя.

Листинг 1 — social_network.py

```
from datetime import datetime


class User:
    def __init__(self, username, email, password):
        self.username = username
        self.email = email
        self.password = password
        self.friends = []
        self.statuses = []
        self.registered_at = datetime.now()

    def get_profile_info(self) -> dict:
        return {
            'username': self.username,
            'email': self.email,
            'friend_count': len(self.friends),
            'status_count': len(self.statuses),
            'registered_at': self.registered_at
        }


class SocialNetwork:
    def __init__(self):
        self.users = {}

    def register_user(self, user) -> None:
        if user.email in self.users:
            raise ValueError(f"Пользователь с электронной почтой"
                             f" {user.email} уже существует")
        self.users[user.email] = user
```

```
    def add_friend(self, user_email, friend_email) -> None:
        if user_email not in self.users:
            raise ValueError(f"Пользователь {user_email} не найден")
        if friend_email not in self.users:
            raise ValueError(f"Пользователь {friend_email} не найден")
        if user_email == friend_email:
            raise ValueError("Нельзя добавить самого себя в друзья")

        user = self.users[user_email]
        friend = self.users[friend_email]

        if friend_email not in user.friends:
            user.friends.append(friend_email)
        if user_email not in friend.friends:
            friend.friends.append(user_email)

    def post_status(self, user_email, text) -> None:
        if user_email not in self.users:
            raise ValueError(f"Пользователь {user_email} не найден")
        if not text.strip():
            raise ValueError("Текст статуса не может быть пустым")

        status = {
            "text": text,
            "timestamp": datetime.now(),
            "likes": 0
        }
        self.users[user_email].statuses.append(status)

    def find_users_by_username(self, username) -> list:
        return [user for user in self.users.values()
                if username.lower() in user.username.lower()]


if __name__ == "__main__":
    network = SocialNetwork()
    user1 = User("ivan_petrov", "ivan@example.com", "password123")
    user2 = User("maria_ivanova", "maria@example.com", "qwerty")
    network.register_user(user1)
    network.register_user(user2)
    network.add_friend("ivan@example.com", "maria@example.com")
    network.post_status("ivan@example.com", "Мой первый статус!")
    found_users = network.find_users_by_username("ivan")
    print(f"Найдено пользователей: {len(found_users)}")
```

2.2. Проект на C++ (matrix_diagonal_sort.cpp)

Функциональность

Программа выполняет обработку квадратной матрицы по заданному алгоритму:

- ввод размера матрицы (целое число от 2 до 5);

- выбор способа заполнения матрицы (ручной ввод или случайная генерация);
- валидация всех входных данных на корректность;
- сортировка по убыванию элементов главной диагонали и элементов ниже неё;
- умножение на -1 всех элементов выше главной диагонали;
- вывод исходной и обработанной матрицы на экран.

Листинг 2 — matrix_diagonal_sort.cpp

```
#include <iostream>
#include <string>
#include <windows.h>
#include <cstdlib>
using namespace std;

int length_of_matrix() {
    string m;
    while (true) {
        cout << "Введите целое значение M (количество строк и столбцов
матрицы) в диапазоне [2, 5]\n";
        cout.width(8);
        cout << "Ввод: ";
        cin >> m;
        if (m == "2" || m == "3" || m == "4" || m == "5")
            break;
        else {
            cout << "Допущена ошибка! Вы должны вводить только целые числа из
диапазона [2, 5]! Повторите попытку ввода!\n";
            m.clear();
        }
    }
    return stoi(m);
}

string answer_to_question() {
    string answer;
    while (true) {
        cout << "Вы хотите ввести данные в матрицу самостоятельно? (1/0)\n";
        cout.width(8);
        cout << "Ввод: ";
        cin >> answer;
        if (answer == "1" || answer == "0")
            break;
        else {
            cout << "Допущена ошибка! Вы должны ответить на вопрос только
одним символом (1/0)! Повторите попытку ввода!\n";
            answer.clear();
        }
    }
    return answer;
```

```
}

int matrix_from_user(int* matrix, int M) {
    cout << "Далее Вам требуется вводить значения элементов в диапазоне [1,
100] (к примеру: A[1][0] = 5)\n";
    for (int i = 0; i < M; i++) {
        for (int j = 0; j < M; j++) {
            string X;
            bool f = false;
            while (!f) {
                cout.width(5);
                cout << "A[" << i << "][" << j << "] = ";
                cin >> X;
                int l = X.length();
                bool p = true;
                for (int k = 0; k < l; k++) {
                    if (!isdigit(X[k])) {
                        cout << "Допущена ошибка! Вы должны вводить только
целые числа из диапазона [1, 100]! Повторите попытку ввода!\n";
                        X.clear();
                        p = false;
                        break;
                    }
                }
                if (p) {
                    int x = stoi(X);
                    if (x >= 1 && x <= 100) {
                        *(matrix + i * M + j) = x;
                        f = true;
                    }
                    else {
                        cout << "Допущена ошибка! Вы должны вводить только
целые числа из диапазона [1, 100]! Повторите попытку ввода!\n";
                        X.clear();
                    }
                }
            }
        }
    }
    return *matrix;
}

int matrix_from_random(int* matrix, int M) {
    srand(time(0));
    cout << "Массив будет заполнен случайными целыми числами в диапазоне [1,
100].\n";
    for (int i = 0; i < M; i++) {
        for (int j = 0; j < M; j++)
            *(matrix + i * M + j) = rand() % 100 + 1;
    }
    return *matrix;
}

void cout_matrix(int* matrix, int M) {
    for (int i = 0; i < M; i++) {
```

```
        for (int j = 0; j < M; j++) {
            cout.width(5);
            cout << *(matrix + i * M + j);
        }
        cout << endl;
    }
}

int matrix_sort(int* matrix, int M) {
    int k1 = ((M * M) - M) / 2 + M;
    int* m1 = new int[k1];
    int k2 = (M * M) - k1;
    int* m2 = new int[k2];
    k1 = 0;
    k2 = 0;
    for (int i = 0; i < M; i++) {
        for (int j = 0; j < M; j++) {
            if (i >= j) {
                m1[k1] = *(matrix + i * M + j);
                k1++;
            }
            else {
                m2[k2] = *(matrix + i * M + j);
                k2++;
            }
        }
    }
    int swap;
    for (int i = 0; i < k1; i++) {
        for (int j = i + 1; j < k1; j++) {
            if (m1[i] < m1[j]) {
                swap = m1[j];
                m1[j] = m1[i];
                m1[i] = swap;
            }
        }
    }
    k1 = 0;
    k2 = 0;
    for (int i = 0; i < M; i++) {
        for (int j = 0; j < M; j++) {
            if (i >= j) {
                *(matrix + i * M + j) = m1[k1];
                k1++;
            }
            else {
                *(matrix + i * M + j) = -m2[k2];
                k2++;
            }
        }
    }
    delete[] m1;
    delete[] m2;
    return *matrix;
}
```

```
int main()
{
    setlocale(LC_ALL, "Russian");
    int M = length_of_matrix();
    string answer = answer_to_question();
    int* matrix = new int[M * M];
    if (answer == "1")
        *matrix = matrix_from_user(matrix, M);
    else
        *matrix = matrix_from_random(matrix, M);
    cout << "Полученная матрица:\n";
    cout_matrix(matrix, M);
    *matrix = matrix_sort(matrix, M);
    cout << "Итоговая матрица:\n";
    cout_matrix(matrix, M);
    delete[] matrix;
    return 0;
}
```

3. ЧАСТЬ 1. СТАТИЧЕСКИЙ АНАЛИЗ КОДА

3.1. Анализ проекта на Python

3.1.1. Выбор инструментов

Для статического анализа были выбраны следующие инструменты:

- Pylint – комплексный анализ стиля и архитектуры;
- Flake8 – проверка соответствия PEP8;
- mypy – статическая проверка типов.

3.1.2. Внесение целенаправленных ошибок

Листинг 3 — Внесённые ошибки

```
...
    def register_user(self, User) -> None:  # Параметр с заглавной буквы
        # Отсутствует docstring
        if User.email in self.users:
            raise ValueError(f"Пользователь с электронной почтой {User.email}
уже существует") # Много символов в строке
        self.users[User.email] = User
        return "success" # Метод должен возвращать None
...
        if friend_email not in self.users:
            raise ValueError(f"Пользователь {friend_email} не найден")
        # Удалено: if user_email == friend_email:
        #     raise ValueError("Нельзя добавить самого себя в друзья")

        user = self.users[user_email]
        friend = self.users[friend_email]

        if friend_email not in user.friends:
            user.friends.append(friend_email)
        # Удалено: if user_email not in friend.friends:
        #     friend.friends.append(user_email)

        return friend  # Должен возвращать None
...
        if user_email not in self.users:
            raise Exception("User not found")  # Вместо ValueError
        if not text:  # Было text.strip():
            raise ValueError("Текст статуса не может быть пустым")
...
        unused_variable = ""  # Неиспользуемая переменная
        if False:  # Недостижимый код
            print("Не выполнится")
        return [user for user in self.users.values()
                if username.lower() in user.username.lower()]
...
```

3.1.3. Сравнительная таблица

Анализатор	Без ошибок	С ошибками
Pylint	********** Module social_network_without_errors social_network_without_errors.py:1:0: C0114: Missing module docstring (missing-module-docstring) social_network_without_errors.py:4:0: C0115: Missing class docstring (missing-class-docstring) social_network_without_errors.py:13:4: C0116: Missing function or method docstring (missing-function-docstring) social_network_without_errors.py:4:0: R0903: Too few public methods (1/2) (too-few-public-methods) social_network_without_errors.py:23:0: C0115: Missing class docstring (missing-class-docstring) social_network_without_errors.py:27:4: C0116: Missing function or method docstring (missing-function-docstring) social_network_without_errors.py:33:4: C0116: Missing function or method docstring (missing-function-docstring) social_network_without_errors.py:49:4: C0116: Missing function or method docstring (missing-function-docstring) social_network_without_errors.py:62:4: C0116: Missing function or method docstring (missing-function-docstring) ---------- Your code has been rated at 8.20/10	********** Module social_network_with_errors social_network_with_errors.py:30:0: C0301: Line too long (120/100) (line-too-long) social_network_with_errors.py:1:0: C0114: Missing module docstring (missing-module-docstring) social_network_with_errors.py:4:0: C0115: Missing class docstring (missing-class-docstring) social_network_with_errors.py:13:4: C0116: Missing function or method docstring (missing-function-docstring) social_network_with_errors.py:4:0: R0903: Too few public methods (1/2) (too-few-public-methods) social_network_with_errors.py:23:0: C0115: Missing class docstring (missing-class-docstring) social_network_with_errors.py:27:4: C0116: Missing function or method docstring (missing-function-docstring) social_network_with_errors.py:27:28: C0103: Argument name "User" doesn't conform to snake_case naming style (invalid-name) social_network_with_errors.py:27:28: W0621: Redefining name 'User' from outer scope (line 4) (redefined-outer-name) social_network_with_errors.py:34:4: C0116: Missing function or method docstring (missing-function-docstring) social_network_with_errors.py:52:4: C0116: Missing function or method docstring (missing-function-docstring) social_network_with_errors.py:54:12: W0719: Raising too general exception: Exception (broad-exception-raised) social_network_with_errors.py:65:4: C0116: Missing function or method docstring (missing-function-docstring) social_network_with_errors.py:67:11: W0125: Using a conditional statement with a constant value (using-constant-test) ---------- Your code has been rated at 7.25/10
Flake8		.\social_network_with_errors.py:30:80: E501 line too long (120 > 79 characters) .\social_network_with_errors.py:30:95: E261 at least two spaces before inline comment .\social_network_with_errors.py:32:25: E261 at least two spaces before inline comment .\social_network_with_errors.py:66:9: F841 local variable 'unused_variable' is assigned to but never used
Mypy	Success: no issues found in 1 source file	social_network_with_errors.py:32: error: No return value expected [return-value]

		Found 1 error in 1 file (checked 1 source file)

3.1.4. Анализ эффективности инструментов

Pylint:

- обнаружил: стилевые ошибки, слишком общие исключения, недостижимый код;
- пропустил: логические ошибки, неправильные возвраты;
- сильные стороны: комплексный анализ архитектуры и стиля.

Flake8:

- обнаружил: нарушения PEP8, неиспользуемые переменные;
- пропустил: большинство семантических ошибок;
- сильные стороны: проверка соответствия стандартам кодирования.

Муру:

- обнаружил: ошибки типов и возвратов;
- пропустил: все стилевые и логические ошибки;
- сильные стороны: статическая проверка типов.

3.2. Анализ проекта на C++

3.2.1. Выбор инструментов

Для статического анализа были выбраны следующие инструменты:

- CppCheck - поиск ошибок управления памятью, неопределённого поведения;
- FlawFinder - поиск потенциальных уязвимостей безопасности;
- Cpplint - проверка соответствия Google C++ Style Guide.

3.2.2. Внесение целенаправленных ошибок

Листинг 4 — Внесённые ошибки

```
...
int length_of_matrix() {
    char buffer[10];
    cout << "Enter value: ";
    cin >> buffer; // Возможное переполнение буфера
    string m;
...
```

```
int matrix_from_random(int* matrix, int M) {
    int uninitialized_var; // Не инициализирована
    if (uninitialized_var > 0)
        cout << "Неинициализированная переменная";
    srand(time(0));
...
void cout_matrix(int* matrix, int M) {
    int divisor = 0;
    if (M > 2)
        divisor = M - 2; // Иногда может быть 0
    int result = 100 / divisor; // Деление на ноль при M=2
    for (int i = 0; i < M; i++) {
...
    k2 = 0;
    for (int i = 0; i <= M; i++) { // i < M
        for (int j = 0; j <= M; j++) { // j < M
            if (i >= j) {
...
    int M = length_of_matrix();
    int* extra_memory_leak = new int[100]; // Занятая память
    string answer = answer_to_question();
...
```

3.2.3. Сравнительная таблица

Анализатор	Без ошибок	С ошибками

CppCheck	cppcheck.exe :
	matrix_diagonal_sort.cpp:1:0:
	information: Include file: <iostream> not
	found. Please note: Cppcheck doe
	s not need standard library headers to
	get proper results.
	[missingIncludeSystem]
	строка:1 знак:1
	+ & $cppcheck --enable=all --inconclusive
	--suppress=missingInclude mat ...
	+
	~~
	~~
	    + CategoryInfo :
	NotSpecified:
	(matrix_diagonal...gIncludeSystem]:String
	) [], RemoteException
	    + FullyQualifiedErrorId :
	NativeCommandError
	
	п>ї#include <iostream>
	^
	matrix_diagonal_sort.cpp:2:0:
	information: Include file: <string> not
	found. Please note: Cppcheck does not
	need standa
	rd library headers to get proper results.
	[missingIncludeSystem]
	#include <string>
	^
	matrix_diagonal_sort.cpp:3:0:
	information: Include file: <windows.h>
	not found. Please note: Cppcheck does not
	need sta
	ndard library headers to get proper
		cppcheck.exe :
		matrix_diagonal_sort_with_errors.cpp:1:0:
		information: Include file: <iostream> not
		found. Please note:
		Cppcheck does not need standard library
		headers to get proper results.
		[missingIncludeSystem]
		строка:1 знак:1
		+ & $cppcheck --enable=all --inconclusive
		--suppress=missingInclude mat ...
		+
		~~
		~~
		    + CategoryInfo :
		NotSpecified:
		(matrix_diagonal...gIncludeSystem]:String
		) [], RemoteException
		    + FullyQualifiedErrorId :
		NativeCommandError
		
		#include <iostream>
		^
		matrix_diagonal_sort_with_errors.cpp:2:0:
		information: Include file: <string> not
		found. Please note: Cppcheck does not
		 need standard library headers to get
		proper results. [missingIncludeSystem]
		#include <string>
		^
		matrix_diagonal_sort_with_errors.cpp:3:0:
		information: Include file: <windows.h>
		not found. Please note: Cppcheck does
		not need standard library headers to get
		proper results. [missingIncludeSystem]
		#include <windows.h>

	results. [missingIncludeSystem]
	#include <windows.h>
	^
	matrix_diagonal_sort.cpp:4:0:
	information: Include file: <cstdlib> not
	found. Please note: Cppcheck does not
	need stand
	ard library headers to get proper
	results. [missingIncludeSystem]
	#include <cstdlib>
	^
	matrix_diagonal_sort.cpp:0:0:
	information: Limiting analysis of
	branches. Use --check-level=exhaustive to
	analyze all b
	ranches. [normalCheckLevelMaxBranches]
	
	^
	nofile:0:0: information: Unmatched
	suppression: missingInclude
	[unmatchedSuppression]
	
	nofile:0:0: information: Active checkers:
	179/836 (use --checkers-report=<filename>
	to see details) [checkersReport]
		^
		matrix_diagonal_sort_with_errors.cpp:4:0:
		information: Include file: <cstdlib> not
		found. Please note: Cppcheck does no
		t need standard library headers to get
		proper results. [missingIncludeSystem]
		#include <cstdlib>
		^
		matrix_diagonal_sort_with_errors.cpp:0:0:
		information: Limiting analysis of
		branches. Use --check-level=exhaustive to
		a
		nalyze all branches.
		[normalCheckLevelMaxBranches]
		
		^
		matrix_diagonal_sort_with_errors.cpp:173:
		5: error: Memory leak: extra_memory_leak
		[memleak]
		return 0;
		^
		matrix_diagonal_sort_with_errors.cpp:98:2
		2: error: Division by zero. [zerodiv]
		int result = 100 / divisor; //
		Деление на ноль при M=2
		^
		matrix_diagonal_sort_with_errors.cpp:95:1
		9: note: Assignment 'divisor=0', assigned
		value is 0
		int divisor = 0;
		^
		matrix_diagonal_sort_with_errors.cpp:96:1
		1: note: Assuming condition is false
		if (M > 2)
		^
		matrix_diagonal_sort_with_errors.cpp:98:2
		2: note: Division by zero
		int result = 100 / divisor; //
		Деление на ноль при M=2
		^
		matrix_diagonal_sort_with_errors.cpp:160:
		10: style: Variable 'extra_memory_leak'
		can be declared as pointer to const [c
		onstVariablePointer]
		int* extra_memory_leak = new
		int[100]; // Занятая память
		^
		matrix_diagonal_sort_with_errors.cpp:83:9
		: error: Uninitialized variable:
		uninitialized_var [uninitvar]
		if (uninitialized_var > 0)
		^
		matrix_diagonal_sort_with_errors.cpp:98:1
		6: style: Variable 'result' is assigned a
		value that is never used. [unreadVar
		iable]
		int result = 100 / divisor; //
		Деление на ноль при M=2
		^
		matrix_diagonal_sort_with_errors.cpp:160:
		28: style: Variable 'extra_memory_leak'
		is assigned a value that is never used
		. [unreadVariable]
		int* extra_memory_leak = new
		int[100]; // Занятая память
		^
		matrix_diagonal_sort_with_errors.cpp:160:
		10: style: Variable 'extra_memory_leak'
		is allocated memory that is never used

		. [unusedAllocatedMemory]
		    int* extra_memory_leak = new int[100]; // Занятая память
		        ^
		nofile:0:0: information: Unmatched suppression: missingInclude [unmatchedSuppression]
		
		nofile:0:0: information: Active checkers: 179/836 (use --checkers-report=<filename> to see details) [checkersReport]

FlawFinder	Flawfinder version 2.0.19, (C) 2001-2019 David A. Wheeler.
	Number of rules (primarily dangerous function names) in C/C++ ruleset: 222
	Error: encoding error in matrix_diagonal_sort.cpp
	'charmap' codec can't decode byte 0x98 in position 5039: character maps to <undefined>
	
	Python3 requires input character data to be perfectly encoded;
	it also requires perfectly correct system encoding settings.
	Unfortunately, your data and/or system settings are not.
	Here are some options:
	1. Run: PYTHONUTF8=0 python3 flawfinder
	   if your system and and data are all properly set up for
	   a non-UTF-8 encoding.
	2. Run: PYTHONUTF8=0 LC_ALL=C.ISO-2022 python3 flawfinder
	   if your data has a specific encoding such as ISO-2022
	   (replace "ISO-2022" with the name of your encoding,
	   and optionally replace "C" with your native language).
	3. Run: PYTHONUTF8=0 LC_ALL=C.ISO-8859-1 python3 flawfinder
	   if your data has an unknown or inconsistent encoding
	   (ISO-8859-1 encoders normally allow anything).
	4. Convert all your source code to the UTF-8 encoding.
	   The system program "iconv" or Python program "cvt2utf" can
	   do this (for cvt2utf, you can use "pip install cvt2utf").
	5. Run: python2 flawfinder
	   (That is, use Python 2 instead of Python 3).
	Some of these options may not work depending on circumstance.
	In the long term, we recommend using UTF-8 for source code.
	For more information, see the documentation.
		Flawfinder version 2.0.19, (C) 2001-2019 David A. Wheeler.
		Number of rules (primarily dangerous function names) in C/C++ ruleset: 222
		matrix_diagonal_sort_with_errors.cpp:85: [3] (random) srand:
		  This function is not sufficiently random for security-related functions
		  such as key and nonce creation (CWE-327). Use a more secure technique for
		  acquiring random values.
		matrix_diagonal_sort_with_errors.cpp:8: [2] (buffer) char:
		  Statically-sized arrays can be improperly restricted, leading to potential
		  overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use
		  functions that limit length, or ensure that the size is larger than the
		  maximum possible length.
		
		ANALYSIS SUMMARY:
		
		Hits = 2
		Lines analyzed = 174 in approximately 0.01 seconds (29450 lines/second)
		Physical Source Lines of Code (SLOC) = 167
		Hits@level = [0] 0 [1] 0 [2] 1 [3] 1 [4] 0 [5] 0
		Hits@level+ = [0+] 2 [1+] 2 [2+] 2 [3+] 1 [4+] 0 [5+] 0
		Hits/KSLOC@level+ = [0+] 11.976 [1+] 11.976 [2+] 11.976 [3+] 5.98802 [4+] 0 [5+] 0
		Minimum risk level = 1
		
		Not every hit is necessarily a security vulnerability.
		You can inhibit a report by adding a comment in this form:
		// flawfinder: ignore
		Make *sure* it's a false positive!
		You can use the option --neverignore to show these.
		
		There may be other security vulnerabilities; review your code!
		See 'Secure Programming HOWTO'
		(https://dwheeler.com/secure-programs) for more information.

Cpplint	Done processing matrix_diagonal_sort.cpp Total errors found: 18	Done processing matrix_diagonal_sort_with_errors.cpp

		Total errors found: 42

3.2.4. Анализ эффективности инструментов

CppCheck:

- обнаружил 3 из 5 семантических ошибок;
- сильные стороны: анализ потока данных, обнаружение утечек памяти, деления на ноль, неинициализированных переменных;
- ложные срабатывания: предупреждения о неиспользуемых переменных.

FlawFinder:

- обнаружил 1 из 5 ошибок (переполнение буфера);
- сильные стороны: фокус на уязвимостях безопасности;
- ограничения: не обнаруживает логические ошибки и проблемы с памятью.

Cpplint:

- обнаружил 0 из 5 семантических ошибок;
- назначение: исключительно проверка стиля кодирования;
- полезность: количество стилевых ошибок увеличилось с 18 до 42 после внесения изменений.

4. ЧАСТЬ 2. ДИНАМИЧЕСКИЙ АНАЛИЗ КОДА

4.1. Анализ проекта на Python

4.1.1. Выбор инструментов

Для динамического анализа были выбраны следующие инструменты:

- pytest + coverage – для тестирования функциональности и анализа покрытия кода;
- memory_profiler – для отслеживания утечек памяти и анализа использования памяти.

4.1.2. Внесение целенаправленных ошибок

Листинг 5 — Внесённые ошибки

```
...
    def __init__(self):
        self.users = {}
        self.memory_leak_list = []  # Для утечки памяти
...
        if user.email in self.users:
            raise ValueError(f"Пользователь с электронной почтой {user.email}
уже существует")

        # Список, который никогда не очищается
        self.memory_leak_list.append({
            'user': user,
            'timestamp': datetime.now(),
            'metadata': 'x' * 1000  # Дополнительные данные
        })

        self.users[user.email] = user
...
    def add_friend(self, user_email, friend_email):
        if user_email not in self.users:
            # Неправильный тип исключения
            raise RuntimeError(f"Пользователь {user_email} не найден")

        if friend_email not in self.users:
            raise ValueError(f"Пользователь {friend_email} не найден")

        # Бесконечный цикл при специальном условии
        if user_email == "trigger@loop.com":
            counter = 0
            while True:  # Бесконечный цикл
                counter += 1
                if counter > 1000:  # Защита от реального бесконечного цикла
в тестах
                    break
```

```
...
    def recursive_status_posting(self, user_email, text, depth=0):
        # Глубокая рекурсия, приводящая к переполнению стека
        if depth > 50:
            return

        self.post_status(user_email, f"Recursive: {text} - {depth}")
        self.recursive_status_posting(user_email, text, depth + 1)
...
```

4.1.3. Результаты динамического анализа

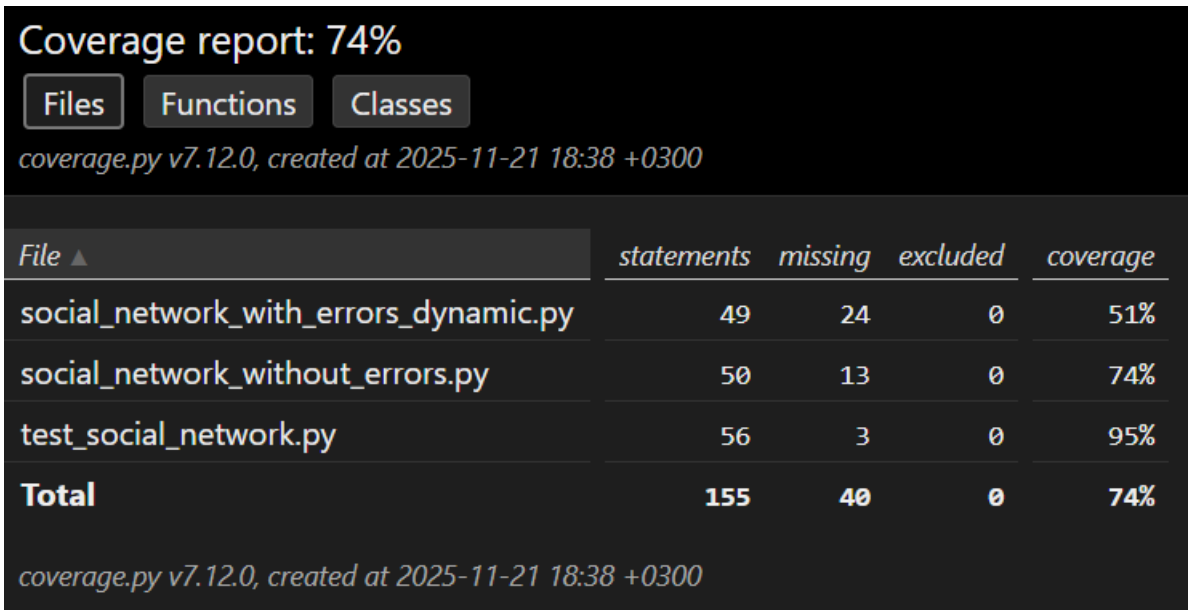

Рисунок 1 – Coverage-анализ

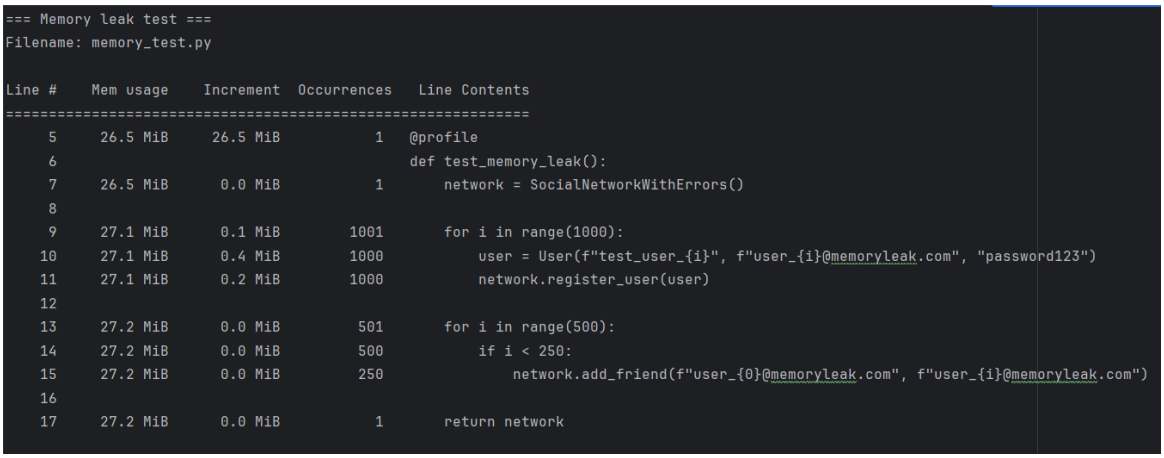

Рисунок 2 – Анализ памяти

4.1.4. Анализ эффективности инструментов

Pytest:

- обнаружил: неправильный тип исключения (RuntimeError вместо ValueError);

- пропустил: утечки памяти, бесконечные циклы, проблемы с рекурсией;
- сильные стороны: идеален для функционального тестирования и проверки исключений;
- ограничения: не отслеживает использование ресурсов и проблемы производительности.

Coverage:

- обнаружил: низкое покрытие кода в версии с ошибками (51% против 74%);
- пропустил: все семантические ошибки выполнения;
- сильные стороны: выявил проблемные участки кода, требующие дополнительного тестирования;
- ограничения: не обнаруживает логические ошибки, только показывает выполненные строки.

Memory_profiler:

- обнаружил: утечку памяти в методе register_user (рост с 26.5 MiB до 27.2 MiB);
- пропустил: все остальные типы runtime-ошибок;
- сильные стороны: эффективен для анализа использования памяти и поиска утечек;
- ограничения: требует ручного анализа результатов, неинтегрирован с тестами.

4.2. Анализ проекта на C++

4.2.1. Выбор инструментов

Для динамического анализа использовались:

- Visual Studio 2022 Diagnostic Tools – мониторинг использования памяти и процессорного времени;
- ручное тестирование – различные сценарии выполнения программы.

4.2.2. Внесение целенаправленных ошибок

Листинг 6 — Внесённые ошибки

```
...
int matrix_sort(int* matrix, int M) {
    int* uninitialized_read = nullptr;
    if (M == 4 && uninitialized_read != nullptr) // Чтение
неинициализированной памяти
        int value = *uninitialized_read;
    int k1 = ((M * M) - M) / 2 + M;
...
void inefficient_algorithm(int* matrix, int M) {
    // Искусственная неэффективность
    for (int i = 0; i < M * 1000; i++)
        for (int j = 0; j < M * 1000; j++)
            volatile int dummy = i * j;
}
...
int* matrix = new int[M * M];
if (M == 3) // Утечка при определённом входе
    int* conditional_leak = new int[500];
if (answer == "1")
...
    *matrix = matrix_sort(matrix, M);
    inefficient_algorithm(matrix, M); // Неэффективный алгоритм
    cout << "Итоговая матрица:\n";
...
```

4.2.3. Результаты динамического анализа

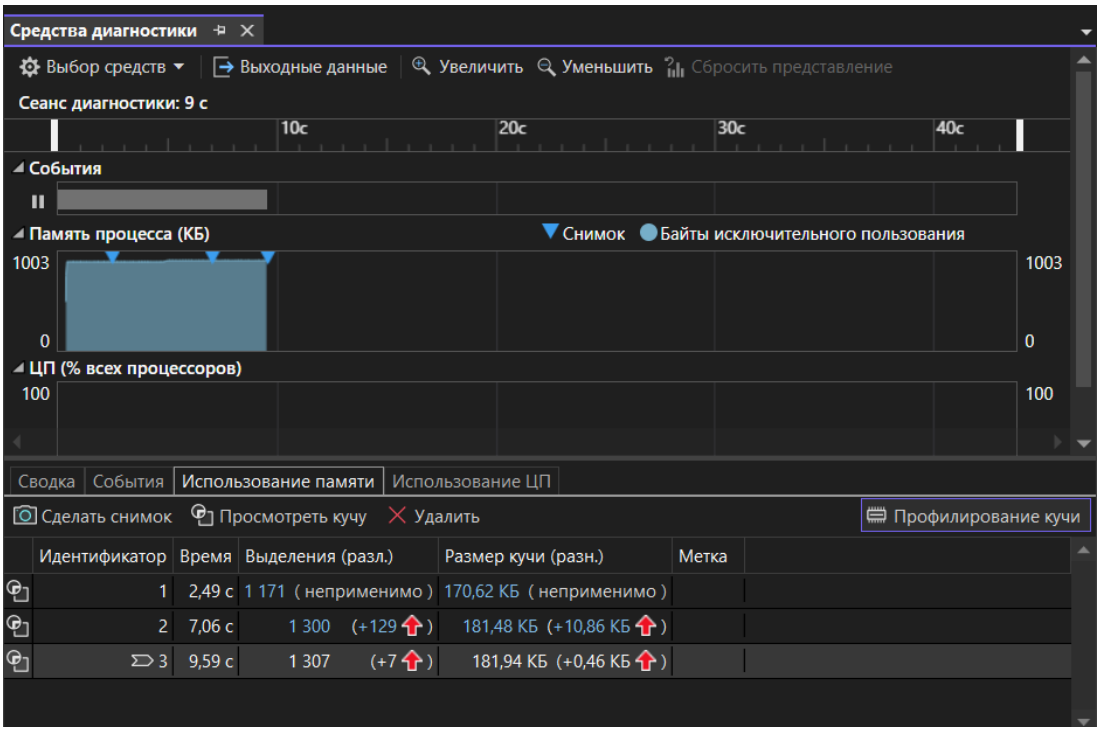

Рисунок 3 – Анализ при входных данных (2, 0)

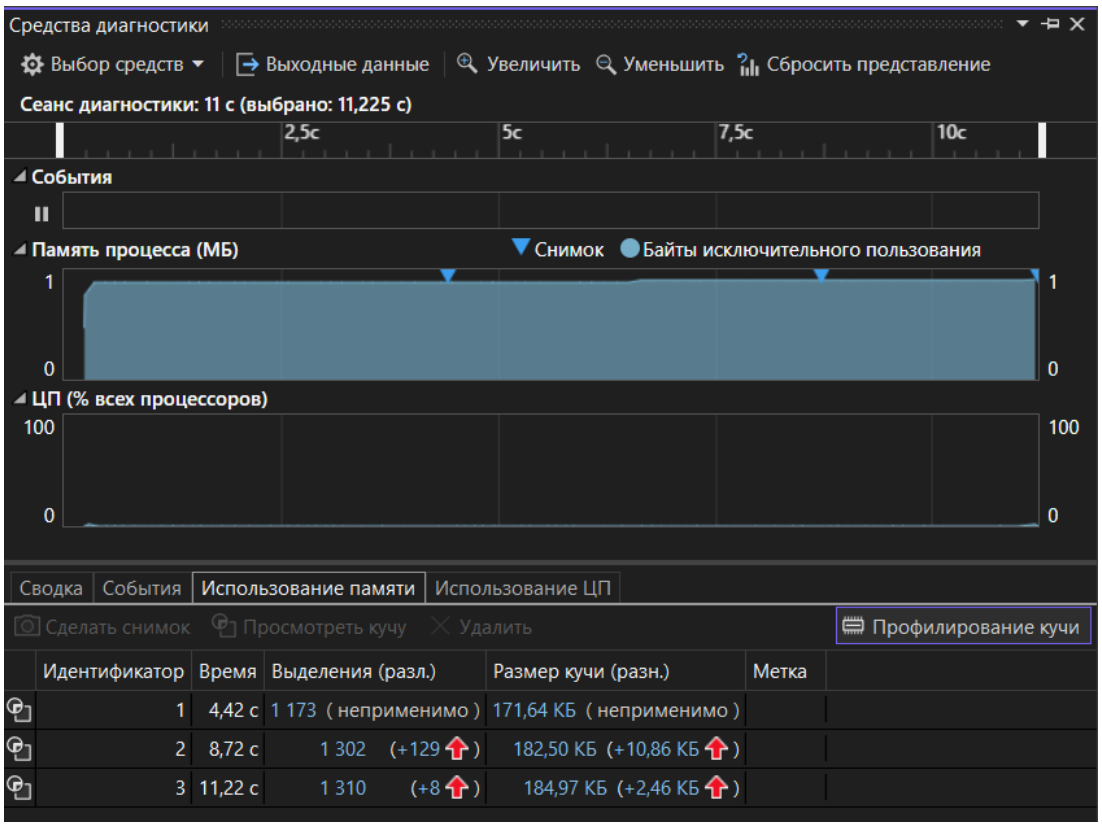

Рисунок 4 – Анализ при входных данных (3, 0)

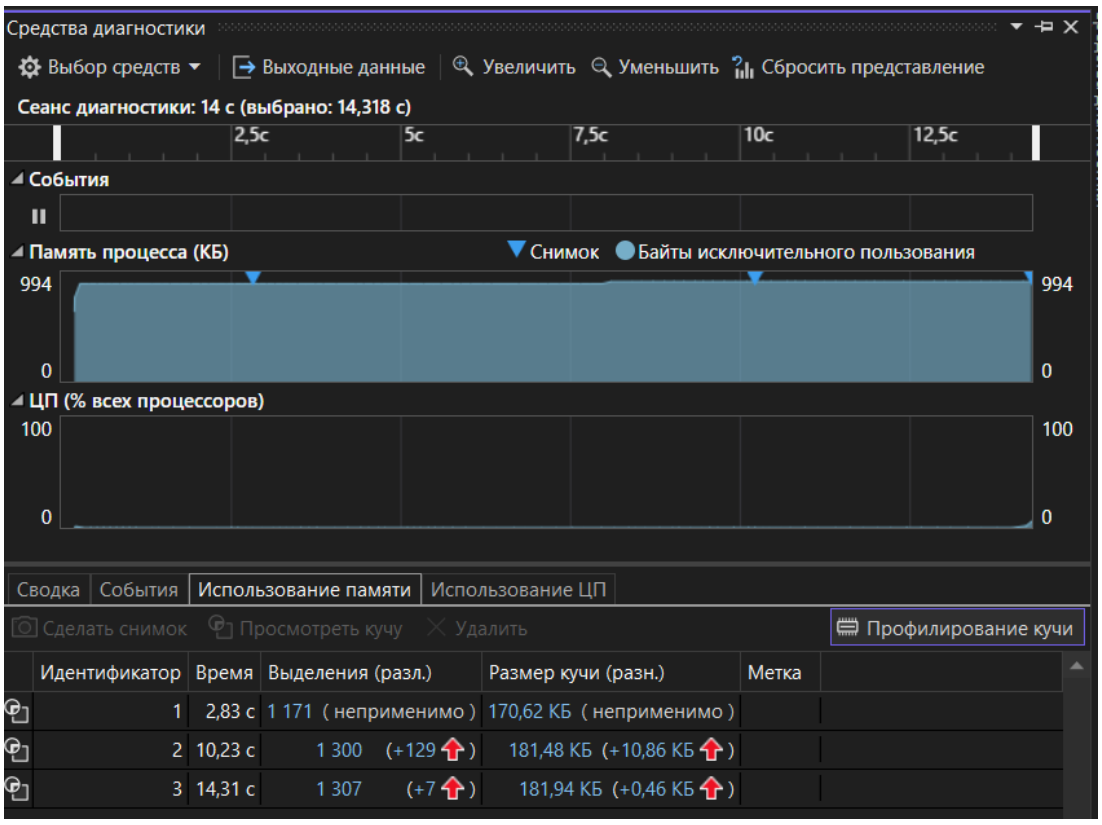

Рисунок 5 – Анализ при входных данных (4, 0)

4.2.4. Анализ эффективности инструментов

Visual Studio Diagnostic Tools

- обнаружено: Косвенные признаки утечки через рост использования памяти;
- эффективность: инструмент показал аномальный рост памяти при M=3, но не указал точное место утечки;
- ложные срабатывания: Отсутствуют;
- пропущенные ошибки: Не обнаружена точная причина утечки (conditional_leak).

5. ЗАКЛЮЧЕНИЕ

В ходе выполнения практической работы был успешно применён комплекс современных методов статического и динамического анализа кода на примере проектов социальной сети на Python и программы сортировки матрицы на C++, что позволило не только выявить существующие и потенциальные ошибки, нарушения стандартов кодирования и проблемы производительности, но и оценить эффективность различных анализаторов в обнаружении целенаправленно внесённых дефектов, продемонстрировав тем самым критическую важность интеграции обоих видов анализа в процесс разработки для обеспечения высокого качества, безопасности и надёжности программного обеспечения.