ГУАП

КАФЕДРА №12

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

Преподаватель		И. Г. Бартасевич
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ

по курсу: МДК 02.01

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. №	С326		Э. С. Тигранян
		подпись, дата	инициалы, фамилия

Санкт-Петербург 2025

Лабораторная работа 8.
Оценка программных средств с помощью метрик.

Цель работы: Изучение основ метрической теории программ Холстеда, расчет количественных характеристик для индивидуального модуля. Определение метрик Чепина.

Порядок выполнения работы

1. Выбранный код программы:

```
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
    setlocale(LC_ALL, "ru");
    cout << "Введите Кординаты точки" << endl;
    float x, y;
    cout << "Ввод x: " << endl;
    cin >> x;
    cout << "Ввод y: " << endl;
    cin >> y;

    if (y >= 0 && (
        (abs(x) + y <= 3 && y < 3) ||
        (abs(x) + y - 3 <= 2 && y >= 3) ||
        (abs(x) + y - 5 <= 1 && y >= 5)
        )) {
        cout << "Точка принадлежит фигуре";
    }
    else {
        cout << "Точка не принадлежит фигуре";
    }
}
```

Оператор	Номер	Число вхождений
cout	1	5
cin	2	2
endl	3	3
>=	4	3
&&	5	4
+	6	3
<=	7	3
||	8	2
;	9	10
{}	10	3
()	11	9
<<	12	5
“”	13	5
-	14	2
if … else	15	1
>>	16	2

using namespace	17	1
#include	18	2
setlocale	19	1
abs	20	3
main	21	1
	21	70

Операнд	Номер	Число вхождений
"Введите Кординаты точки"	1	1
"Ввод x: "	2	1
"Ввод y: "	3	1
X	4	5
Y	5	8
"Точка принадлежит фигуре"	6	1
"Точка не принадлежит фигуре"	7	1
3	8	4
2	9	1
5	10	2
1	11	1
0	12	1
LC_ALL	13	1
"ru"	14	1
std	15	1
<iostream>	16	1
<cmath>	17	1
	17	32

Характеристика	Значение
n_1	21
n_2	17
n	31
N_1	70
N_2	32
n_2*	7
N	65

2. Метрики Холстеда по формулам 1-11.

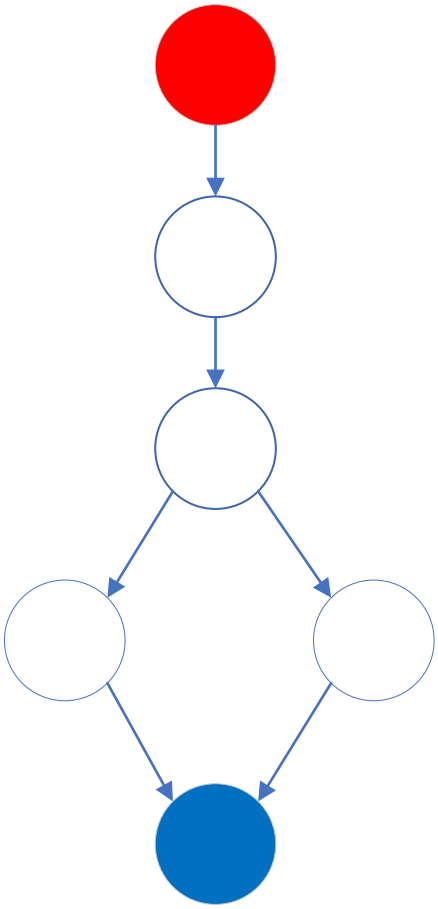
Характеристика	Формула	Значение
N’	$= n_1 \cdot \log_2(n_1) + n_2 \cdot \log_2(n_2)$	161,7255342
V	$= N \cdot \log_2(n)$	322,0227602
V*	$= (2+ n_2^*) \cdot \log_2(2+ n_2^*)$	28,52932501

V**	= (2+ (n_2*)·$\log_2$(n_2*))·$\log_2$(2+ n_2*)	68,63358189
A	= n_2*/(n_2* +2)·$\log_2$(n_2*/2)	1,405720495
L	= V*/V	0,088594126
I	= 2·n2 /(n1 ·N2) ·N·log 2 (n).	16,29281822
E	= V/L	3634,809377
T’	= E/S	201,9338543
Ã	= L·L·V	2,527530617
B_o	= V/(V*·V*·V*/(A·A))	0,002427811

3. Оценика качества реализации алгоритма на основании метрик Холстеда.

- Низкий уровень программы (L ≈ 0.088): избыточность условий в if
- Высокий объём (V ≈ 322).
- *V = 15.51** – значительно меньше фактического объёма (V), что указывает на избыточность кода.
- Умеренная ошибкоопасность (Ã 2,527).
- Большие усилия (E ≈ 3634) Код требует значительных трудозатрат на понимание и поддержку.
- Bo ≈ 0.002 – низкое значение, что подтверждает неоптимальность кода.
- T' ≈ 201 – высокое значение, указывающее на длительное время разработки.
- I ≈ 16 – низкое значение, что может указывать на недостаточную структурированность или сложность восприятия кода.

4. Постройка графа потока управления программы из п. 1 и подсчёт Цикломатической сложности.



Характеристика	Значение
e	6
n	6
p	1

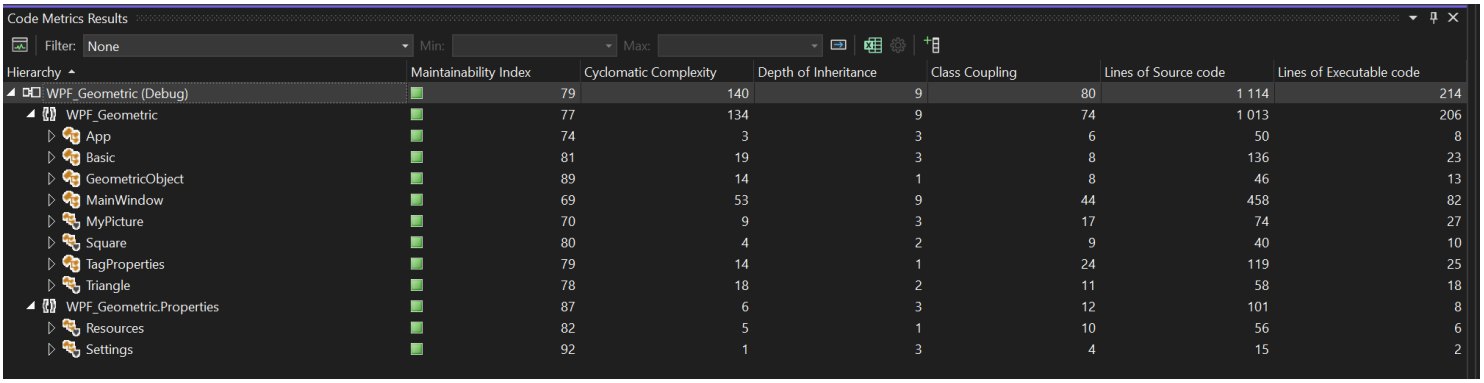
C = e – n + 2p = 2

5. Подсчёт метрик Чепина.

Группа	Количество
P	2
M	0
C	2
T	0

Q = P + 2M + 3C + 0.5T = 8

6. Определение метрик кода с помощью встроенных средств анализа Visual Studio.


Code Metrics Results

Filter: None Min: Max:

Hierarchy	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Source code	Lines of Executable code
WPF_Geometric (Debug)	79	140	9	80	1 114	214
WPF_Geometric	77	134	9	74	1 013	206
App	74	3	3	6	50	8
Basic	81	19	3	8	136	23
GeometricObject	89	14	1	8	46	13
MainWindow	69	53	9	44	458	82
MyPicture	70	9	3	17	74	27
Square	80	4	2	9	40	10
TagProperties	79	14	1	24	119	25
Triangle	78	18	2	11	58	18
WPF_Geometric.Properties	87	6	3	12	101	8
Resources	82	5	1	10	56	6
Settings	92	1	3	4	15	2

Вывод:
Метод 0.1W7_Secreotide (124) имеет высокую сложность, что значительно увеличивает риск ошибок и затрудняет тестирование.
Модули 0.1W7_Multimodeer (33) и 0.1W7_SecreotideProperties (16) также сложны.
Класс 0.1W7_Multimodeer (уровень наследования = 9) обладает сложной иерархией.
Модули 0.1W7_Generate (80), 0.1W7_Secreotide (74) и 0.1W7_DutyInjection (24) имеют большое количество зависимостей - снижающее гибкость кода.
Аномально высокие значения дат (Date of Secure/Excusable code)
у 0.1W7_Secreotide (1013/406) и 0.1W7_Multimodeer (458/52) указывают на потенциальные риски безопасности или наличие устаревшего кода.

Контрольные вопросы:
1. Где можно использовать метрики Холстеда и Чепина?
Метрики Холстеда и Чепина применяются для количественной оценки сложности программного кода.

2. Чем определяются характеристики программы?
Характеристики программы в метриках Холстеда определяются параметрами:

- Количество операторов
- Количество операндов
- Число уникальных операторов и операндов
- Ветвления и циклы

На основе параметров рассчитываются метрики:

- Словарь программы (n) – сумма уникальных операторов и операндов.
- Длина программы (N) – общее количество операторов и операндов.
- Объем (V) – оценка размера программы в битах.

3. Как оценить качество реализации алгоритма по метрикам?
Качество реализации алгоритма можно оценить:

- Уровень программы (L) – чем выше L, тем проще код для понимания
- Сложность (V) – чем ниже V, тем проще жальнейшая работа с кодом.
- Усилия программиста (E) – чем ниже E, тем меньше ресурсов нужно для написания и отладки.

4. В чем недостаток программометрии?
Основные ограничения метрического подхода:

- Игнорирование семантики – метрики не учитывают логику и смысл кода, только синтаксис.
- Читаемость – код с низкой сложностью по Холстеду может быть плохо структурирован или документирован.
- Архитектура – не оценивает модульность, связи между компонентами или паттерны проектирования.
- Контекст – не учитывает предметную область или бизнес-логику.
- Динамическое поведение – метрики анализируют статический код, а не runtime-характеристики (например, производительность).

5. В чем смысл цикломатической сложности?
Цикломатическая сложность метрика, которая:

- Показывает число независимых путей в графе управления программой.
- Оценивает тестируемость: чем выше сложность, тем больше тестов нужно для полного покрытия.
- Помогает выявить риски: код с высокой цикломатической сложностью (например, >10) сложнее поддерживать и отлаживать.