Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Кемеровский государственный университет»
Новокузнецкий институт (филиал)

Факультет информатики, математики и экономики
Кафедра математики, физики и математического моделирования

Е.В. Решетникова

МЕТРОЛОГИЯ И КАЧЕСТВО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

(Метрики, основанные на лексическом анализе программ)

Методические указания к организации самостоятельной работы студентов для обучающихся по направлению подготовки 01.04.02 Прикладная математика и информатика, профиль «Математическое моделирование»

Новокузнецк

2020

УДК [378.147.88:004.42](072)
ББК 74.484(2Рос-4Кем)я73+32.973я73
Р47

Решетникова Е.В.
Р47 Метрология и качество программного обеспечения (Метрики, основанные на лексическом анализе программ): методические указания к организации аудиторной и самостоятельной работы студентов факультета информатики, математики и экономики, обучающихся по направлению подготовки 01.04.02 Прикладная математика и информатика, профиль «Математическое моделирование» / Е.В. Решетникова; Новокузнецкий ин-т (фил.) Кемеров. гос. ун-та. – Новокузнецк : НФИ КемГУ, 2020 – 80 с.

Методические указания содержат краткие теоретические сведения, примеры решения практических задач, задания для решения на практических занятиях и задания для самостоятельной работы; список основной и дополнительной литературы.

Методические указания предназначены для наиболее рациональной организации аудиторной и внеаудиторной самостоятельной работы студентов.

Рекомендовано на заседании
кафедры математики, физики и
математического моделирования
Протокол № 3 от 22 октября 2020г.
Заведующий кафедрой

 / Е.В. Решетникова

УДК [378.147.88:004.42](072)
ББК 74.484(2Рос-4Кем)я73+32.973я73
Р47

Текст представлен в авторской редакции

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Настоящие методические указания адресованы студентам, получающим квалификацию магистр по направлению подготовки: 01.04.02 Прикладная математика, профиль «Математическое моделирование» и направлены на оказание помощи студентам и преподавателю при организации работы на лабораторных занятиях и самостоятельной работы студентов по дисциплине «Метрология и качество программного обеспечения».

Непрерывное повышение сложности функций, которые необходимы запрограммировать для информационных систем, приводит к увеличению объема и трудоемкости создания таких систем. И соответственно к сложности программ, в которых еще быстрее возрастает количество как выявляемых, так и остающихся дефектов, и ошибок, что несомненно отражается на их качестве.

Разрабатываемые в настоящее время комплексы программ объемом в миллионы строк текста, принципиально не могут быть безошибочными. Поэтому проблема обнаружения и устранения ошибок обостряется по мере увеличения сложности программируемых задач.

Не выявленные ошибки могут грозить катастрофами, например, в информационных системах управления крупными, дорогими и особенно важными объектами или процессами.

В методические рекомендации включено: краткие теоретические сведения, примеры решения практических задач, задания для решения на лабораторных занятиях и задания для самостоятельной работы; список основной и дополнительной литературы.

Таким образом, данные методические материалы позволяют студенту подготовиться к лабораторным занятиям по соответствующим темам, организовать самостоятельную работу по дисциплине и успешно выполнить индивидуальные задания. Методические указания могут оказаться полезными при написании курсовых и выпускных квалификационных работ.

1 ОЦЕНКА ПРОГРАММНОГО СРЕДСТВА НА ОСНОВЕ МЕТРИК ХОЛСТЕДА

1.1 Теоретические сведения

Исходный текст программы представляет собой набор текстовых строк, которые записываются по специальным правилам, и в том числе имеет свои элементы. Любая программа определяет последовательность действий над *операндами* с помощью *операторов*.

Операнд – это некоторый объект или величина, обрабатываемая в программе, *оператор* – обозначение конкретного действия, выполняемого по отношению к операнду.

Будем считать, что словарь программы состоит только из имен операторов и операндов.

Формирование словаря операторов и операций.

Программы будем рассматривать на языках программирования C, C++ и C#. В рассматриваемых языках программирования добавление к набору символов, являющихся выражением, знака «;» превращает его в оператор. Так, набор символов «i++» является выражением, а конструкция «i++;» уже является оператором.

При разработке программ существует ряд особенностей, связанных со скобками: круглыми, фигурными и квадратными.

Одним из действий, которые могут быть записаны в исходном тексте программ, является вызов функций, записываемый в формате `Имя_функции()`. Будем различать вызов функции по появлению пары круглых скобок и следующим перед ними именем. Следует обратить внимание, что при этом нужно отличать пару круглых скобок, используемых с другими языковыми конструкциями (например, с

операторами `if` или `for`).

Учитываются также и фигурные скобки. Они, не вызывают никаких действий, но связаны с ограничением области видимости переменных.

Операции, которые связаны с квадратными скобками, это обращение к элементам массива.

Как правило, при проведении статистических исследований текстов программ к словарю операторов относят следующие элементы:

- имена арифметических и логических операций;
- присваивания;
- условные и безусловные переходы;
- разделители;
- скобки (парные);
- имена процедур и функций;
- выражения типа BEGIN...END, IF…THEN…ELSE, DO…WHILE.

Выражения типа BEGIN…END, IF…THEN…ELSE, DO…WHILE и им подобные, осуществляющие блочную группировку операторов, при этом рассматриваются как единые операторы (то же относится и к парам скобок).

Формирование словаря операндов.

Операторы и операции связаны с объектами, над которыми они выполняются. Такими объектами являются, прежде всего, константы, простые переменные, массивы и структуры, также операндами являются функции, а также классы и методы.

При формировании словарей будем учитывать элементы, используемые в разделах описаний, поскольку эти элементы также можно рассматривать как операнды или операторы.

В состав измеримых свойств алгоритма (или программы) включены

следующие *метрические характеристики*:

- n_1 – число простых (уникальных) операторов, появляющихся в данной реализации;

- n_2 – число простых (уникальных) операндов, появляющихся в данной реализации;

- N_1 – общее число всех операторов, появляющихся в данной реализации;

- N_2 – общее число всех операндов, появляющихся в данной реализации;

- f_{1j} – число появлений в программе j-го оператора, где j = 1, 2, 3, ..., n_1;

- f_{2j} – число появлений в программе j-го операнда, где j = 1, 2, 3, ..., n_2;

- n_2^* - число имен входных и выходных переменных.

В конкретной реализации текста программы можно определить:

- словарь $n = n_1 + n_2$;

- длину реализации программы $N = N_1 + N_2$;

- длину программы $\tilde{N} = n_1 \cdot \log_2 n_1 + n_2 \cdot \log_2 n_2$;

- объем программы $V = N \log_2 n = n \log_2^2 n$;

-потенциальный объем программы, соответствующий максимально компактному тексту программы, реализующей данный алгоритм $V^* = \left(n_2^* + 2\right)\log_2\left(n_2^* + 2\right)$;

- уровень реализации программы $L = \dfrac{V^*}{V}$;

- наилучшее количество модулей, которое будет обеспечивать

минимальную длину программы, $k_{опт} \approx \frac{n_2^*}{\log_2 2n_2^*}$;

- работа программирования $E = \frac{V}{L}$;

- уровень языка программирования $\lambda = LV^*$;

- число переданных ошибок в программе $B = \frac{LE}{3000}$.

Метрики длины программы и длины реализации можно использовать для выявления несовершенств программирования, которые являются следствием применения не самых удачных приемов программирования. Если расчетные значения длины программы и длины реализации отличаются более чем на 10 %, то это свидетельствует о возможном наличии в программе следующих шести ***классов несовершенств***:

1. Наличие последовательности дополняющих друг друга операторов к одному и тому же операнду (например, *A* + *C* – *A*).

2. Наличие неоднозначных операндов (например, *A* = *D* и *A* = *C*).

3. Наличие синонимичных операндов (например, *A* = *B* и *C* = *B* - более лаконичным вариантом является простое приравнивание значений переменных *A* и *C*).

4. Наличие общих подвыражений (например: (*A*+*B*)*C*+*D*(*A*+*B*) - здесь применено совсем не обязательное повторение суммирования переменных *A* и *B,* что приводит к дополнительному времени выполнения программы.

5. Ненужное присваивание (например: *C* = *A* +*B*, если переменная *C* используется в программе только один раз). При однократном выполнении каких-либо операций над переменной нецелесообразно вводить дополнительный операнд, это ведет к увеличению объема

памяти, резервируемой под переменные программы, и увеличивает размер словаря.

6. Наличие выражений, которые не представлены в свернутом виде как произведение множителей (например: $X \cdot X + 2 \cdot X \cdot Y + Y \cdot Y$ - данное преобразование можно представить как $(X + Y) \cdot (X + Y)$, т. е. свернуть выражение до квадрата суммы переменных X и Y). Такое представление окажется более лаконичным и сократит время, необходимое для выполнения программы.

Уровень реализации представляет собой метрический показатель, который характеризует степень компактности программы, экономичность использования средств алгоритмического языка. Чем ближе значение L к единице, тем более совершенна программа.

Алгоритмически сложные программы вычисления малого числа переменных будут давать значительно более низкое значение λ, чем программы вычисления большого числа переменных по элементарным выражениям. В связи с этим метрику уровня языка программирования λ для сравнения языков следует применять только для конкретной предметной области и близких типов задач. Ниже приведены данные об уровнях некоторых известных языков программирования (таблица 1).

Таблица 1. Уровни языков программирования

Язык	λ	Отклонения
Естественный язык	2,16	0,74
PL/1	1,53	0,92
Алгол	1,21	0,74
Паскаль	1,25	0,76
Бейсик	1,22	0,72
Фортран	1,14	0,81
Ассемблер	0,88	0,42

1.2 Примеры решений практических заданий

Задача 1. «Расчет значений функции».

Разработать программу для вычисления значений функции:

$$F = \begin{cases} \sin(x) + \cos^2(y), & \text{при } x < y; \\ \ln(x), & \text{при } x = y; \\ \sin^2(x) + \cos(y), & \text{при } x > y. \end{cases}$$

Значения аргументов функции ввести с клавиатуры. На экран монитора вывести значение функции. Определить значения метрик Холстеда, на основе которых дать оценку качества разработанного исходного текста программы.

Реализация программы. Текст программы для реализации возможного решения поставленной задачи, разработанной с использованием языка программирования C#, приведен в таблице 2.

Таблица 2 – Реализация программы для задачи «Расчет значения функции"

Номер строки	Строки программы
1.	using System;
2.	namespace Ex
3.	{
4.	class Program
5.	{
6.	static void Main()
7.	{
8.	double x, y, F;
9.	char check;
10.	do
11.	{
12.	Console.WriteLine("Введите значение переменной x");
13.	Console.Write("x=");
14.	x = double.Parse(Console.ReadLine());
15.	Console.WriteLine("Введите значение переменной y");
16.	Console.Write("y=");
17.	y = double.Parse(Console.ReadLine());
18.	if (x < y)
19.	F = Math.Sin(x) + Math. Cos(y)* Math.Cos(y);
20.	else
21.	if (x = = y)
22.	F = Math.Log(x);
23.	else
24.	F = Math.Sin(x) * Math. Sin(x) + Math.Cos(y);
25.	Console.WriteLine("F = "+F);

26.	Console.WriteLine("Хотите запустить программу вновь? Y/N");
27.	chek=char.Parse(Console.ReadLine());
28.	}while (check= ='Y' || check= ='y');
29.	}
30.	}
31.	}

Словарь программы

В таблице 3 приведены операторы и операции, используемые в программе.

Таблица 3 - Словарь операторов и операций программы

№ п/п	Операторы, операции	Номера строк исходной программы, где встречается оператор или операция	Количество повторений с тексте исходной программы
1	using ...;	1	1
2	namespace ...	2	1
3	class ...	4	1
4	static void...	6	1
5	double...	8	1
6	char...	9	1
7	do... while()	10-28	1
8	Console.WriteLine()	12, 15. 25.26	4
9	Console.Write()	13. 16	2
10	....Parse()	14. 17.27	3
II	Console. ReadLine()	14. 17. 27	3
12	if ()... else...	18.21	2
13	Math.Sin()	19. 24.24	3
14	Math.Cos()	19. 19. 24	3
15	Math.Log()	22	1
16	;	1.8.9. 12. 13. 14. 15. 16. 17. 19. 22. 24. 25. 26. 27. 28	16
17	,	8.8	2
18	*	19. 24	2
19	=	14. 17. 19. 22. 24. 27	
20	+	19.24. 25	3
21	<	18	1
22	= =	21.28. 28	3
23	{}	3(31). 5(30). 7(29). 11(28)	4
24	()	6. 12. 13. 14. 14. 15. 16. 17. 17. 18. 19. 19. 19.21.22. 24. 24. 24. 25. 26. 27.27. 28	23
25	| |	28	1

26	“ "	12. 13. 15. 16.25. 26.	6
27	' '	28. 28	2
28	.	12, 13, 14. 14. 15. 16. 17. 17, 19, 19, 19, 22.24.24.24. 25. 26. 27.27	19
Всего			116

Таким образом, количество строк таблицы 3 есть число уникальных операторов и операций, появляющихся в данном тексте. Если вычислить сумму значений из четвертого столбца, то получим общее число всех операторов и операций, используемых в исходном тексте программы. Отметим, что для фигурных скобок, определяющих блок, приведены два номера строки. Первый определяет левую фигурную скобку, открывающую блок, а второй – закрывающую. Отметим, что такая пара в словаре учитывается только один раз.

Проведем подробный анализ исходного текста программы в соответствии с полученной таблицей 2, начиная с ее первой позиции.

Первая строка программы `using System;` (таблица 2, п.1).

Ключевое слово `using` представляет собой команду (инструкцию), обеспечивающую доступ к именам пространства имен `System`.

Следовательно, команду `using` можно отнести к выполняемым операторам (таблица 3, п. 1). Оператор `using` встречается в программе всего один раз.

Слово `System` представляет собой имя, над которым осуществляется операция `using`. Таким образом, имя `System` заносится в таблицу словаря операндов (таблица 4, п. 1). Имя `System` встречается в программе один раз.

Следующая строчка программы `namespace Ex` (таблица 2, п. 2).

Состоит из оператора `namespace` и операнда `Ex`, которые также присутствуют в тексте программы в единственном экземпляре.

Оператор занесен в таблицу операторов (таблица 3, п. 2), а операнд `Ex` – в таблицу операндов (таблица 4, п. 2).

Строки

- `class Program`
- `static void Main()`
- `double x, y, F;`
- `char check;`

также представляют собой сочетание операторов и операндов, которые встречаются в тексте один раз (таблица 2, п. 4, 6 и 8), где ключевые слова `class`, `static void`, `double` и `char` представляют соответственно операции, а `Program`, `Main`, `x`, `y`, `F` и `check` – имена (операнды). Все операции попадают в словарь операторов (таблица 3), а имена – в словарь операндов (таблица 4).

Следующий оператор `do … while()` (таблица 2, п. 10–28).

Представляет собой инструкцию реализации циклического алгоритма, которая используется в тексте программы один раз. Рассмотрим тело цикла (блок операторов, заключенных между ключевыми словами `do … while`).

Первой строчкой цикла является операция вызова функции вывода строк на экран монитора `Console.WriteLine()`.

Данная операция повторяется в тексте программы 4 раза (таблица 2, п. 12, 15, 25, 26). В каждом из этих случаев применения оператора вызова функции (метода) `Console.WriteLine()` входным параметром функции является строка (строковая константа). Значение строковой

константы в каждом случае применения оператора разное:

• "Введите значение переменной x";

• "Введите значение переменной y";

• "F = ";

• "Хотите запустить программу вновь? Y/N".

Все перечисленные константы являются операндами и заносятся в таблицу 4. Каждый из перечисленных операндов используется один раз.

Следующим по ходу выполнения программы выполняется оператор `Console.Write();` вызова функции вывода символов на экран (таблица 2, п. 13, 16).

Оператор используется 2 раза с разными операндами:

• "*x*=";

• "*y*=",

каждый из которых используется в программе однократно. Оператор включается в таблицу операторов, операнды – в таблицу операндов.

Следующая операция

```
x = double.Parse(Console.ReadLine());
```

Включает три оператора:

= – оператор присваивания (таблица 2, п. 14, 17, 19, 22, 24, 27) используется в программе 6 раз;

… `Parse()` – оператор вызова функции преобразования строки в заданный тип (таблица 2, п. 14, 17, 27) используется в программе 3 раза;

`Console.ReadLine()` – оператор вызова функции считывания строки с клавиатуры (таблица 2, п. 14, 17, 27) используется в программе 3 раза.

Оператор `if()…else…` (таблица 2, п. 18, 21) используется дважды в тексте программы для ветвления алгоритма.

Операции (таблица 2, п. 19, 22, 24):

```
F = Math.Sin(x) + Math.Cos(y) * Math.Cos(y);
F = Math.Log(x);
F = Math.Sin(x) * Math.Sin(x) + Math.Cos(y);
```

включают следующие ранее не рассмотренные операторы:

`Math.Sin(x)` - оператор вызова функции вычисления синуса, используется 3 раза;

`Math.Cos(y)` - оператор вызова функции вычисления косинуса, применяется 3 раза;

`Math.Log(x)` – оператор вызова функции вычисления логарифма, используется один раз.

Имена F, x и y являются операндами: F используется 5 раз, x – 8раз, y – 7 раз.

Символы «;», «,», «*» и «+», используемые в программе, обозначают следующие операции:

; – операция определения завершения оператора, используется 16 раз;

, – операция отделения элементов списка, используется 2 раза;

* – операция умножения, используется 2 раза;

+ – операция сложения (сцепления строк), используется 3 раза.

Символы «<» и «= =» используются для определения логических операций сравнения:

< – операция сравнения «меньше», используется 1 раз;

= = – операция сравнения «равно», используется 3 раза.

В позициях 23 и 24 таблицы 3. представлены символы, определяющие следующие операции:

{} – операция начала и завершения блока инструкций, используется 4 раза;

() – операция начала и завершения списка параметров или условия, используется 22 раза.

Оставшиеся четыре позиции таблицы 3 содержат символы:

|| – операция логического сложения (дизъюнкция), используется один раз;

“ ” – операция определения строковых констант, используется6 раз;

‘ ’ – операция определения символьных констант, используется 2 раза;

. – операция связывания имен, используется 19 раз.

Таблица 4. - Словарь операндов программы

№ п/п	Операнды	Номера строк	Количество повторений
1	System	1	1
2	Ex	2	1
3	Program	4	1
4	Main	6	1
5	x	8. 14. 18. 19.21.22. 24. 24	8
6	y	8. 17. 18. 19. 19.21.24	7
7	check	9. 27. 28, 28	4
8	"Введите значение переменной x"	12	1
9	"x="	13	1
10	"Введите значение переменной y"	15	1
II	"y="	16	1
12	F	8. 19, 22. 24. 25	5

13	"Хотите запустить программу вновь? Y/N"	26	1
14	'Y'	28	1
15	'y'	28	1
16	"F =*	25	1
		Всего	36

Проанализируем содержание таблица 4. Позиции 1, 2, 3 и 4 содержат имена операндов `System, Ex, Program, Main(),` которые используются в программе по одному разу. Строковые константы (п. 8, 9, 10, 11, 13 и 16 таблицы 4):

"Введите значение переменной x";

"x=";

"Введите значение переменной y";

"y=";

"Хотите запустить программу вновь? Y/N";

"F = "используются в тексте программы однократно (таблица 4).

Символьные константы 'Y' и 'y' применяются также по одному разу (таблица 4). Имена переменных x, y, check и F повторяются в программе соответственно 8, 7, 4 и 5 раз.

Для рассматриваемой программы список входных и выходных параметров (таблица 5) не обладает большим разнообразием. Входными параметрами являются значения переменных:

```
x = double.Parse(Console.ReadLine());
y = double.Parse(Console.ReadLine());
check=char.Parse(Console.ReadLine()).
```

Таблица 5. - Входные и выходные переменные программы

Входные переменные	Выходные переменные
x	"Введите значение переменной x"
y	"x="
check	"Введите значение переменной y"
	"y= "
	"F ="

	"Хотите запустить программу вновь? Y/N"
	F

Выходными значениями являются шесть констант, для которых имена совпадают со значениями, и одна переменная F:

Console.WriteLine("Введите значение переменной x");

Console.Write("x=");

Console.WriteLine("Введите значение переменной y");

Console.Write("y=");

Console.WriteLine("F = "+F) – в этом случае два выходных параметра: строковая константа "F = " и переменная F;

Console.WriteLine("Хотите запустить программу вновь? Y/N").

Оценка характеристик программы

Используя сформированные таблицы с необходимыми параметрами для расчета и применяя соотношения Холстеда, вычислим характеристики рассматриваемой программы. Сведем все результаты расчетов метрик Холстеда в таблица 6.

Таблица 6 - Значения метрик Холстеда для программы

Наименование характеристики	Обозначение и формула для вычисления	Значение
Число простых (уникальных) операторов и операций	n_1	28
Число простых (уникальных) операндов	n_2	16
Общее число всех операторов и операций	N_1	116
Общее число всех операндов	N_2	36
Число входных и выходных переменных (параметров)	n_2^*	10
Словарь программы	$n=n_1+n_2$	44
Длина реализации программы	$N = N_1 + N_2$	152
Длина программы	$\tilde{N} = n_1 \cdot \log_2 n_1 + n_2 \cdot \log_2 n_2$	198,68
Объем программы (в битах)	$V = N \log_2 n$	830
Потенциальный объем программы	$V^* = \left(n_2^* + 2\right)\log_2\left(n_2^* + 2\right)$	43
Уровень реализации программы	$L = \frac{V^*}{V}$	0,052

Уровень реализации языка	$\lambda = LV^*$	2,23
Работа программирования	$E = V / L$	15960
Число переданных ошибок в программе	$B = \frac{LE}{3000}$	0,3

Выводы:

1) Длина реализации меньше длины программы, следовательно, несовершенства в программе отсутствуют.

2) Уровень реализации исследуемой программы весьма низкий, так как потенциальный объем программы в значительной степени меньше ее реального объема.

3) Возможности языка программирования использованы на достаточном уровне.

Задача 2. «Замена строк таблицы»

Дана вещественная таблица размером $M \times N$ элементов. Размер таблицы вводится с клавиатуры. Поменять местами строки таблицы по правилу: строка с номером 0 меняется с последней, строка с номером 1 с предпоследней и т.д.

Пример проведения данной операции приведен ниже.

Таблица в исходном виде	Преобразованная таблица
1,0 2,0 3,0 4,0	9,0 10,0 11,0 12,0
5,0 6,0 7,0 8,0	5,0 6,0 7,0 8,0
9,0 10,0 11,0 12,0	1,0 2,0 3,0 4,0

Разработать программу и определить значения метрик Холстеда, на основе которых дать оценку качества разработанного исходного текста программы.

Таблица 7 – Реализация программы для задачи «Замена строк таблицы» на языке C#.

Номера строк	Строки программы
1	using System;
2	namespace Exemp

```
3   {
4   class Replace
5   {
6   static void Прочитать(double[,] a, int n, int m)
7   {
8   int i, j;
9   for (i = 0; i < n; i++)
10  for (j = 0; j < m; j++)
11  {
12  Console.Write("Элемент[{0},{1}]: ", i, j);
13  a[i, j] = double.Parse(Console.ReadLine());
14  }
15  }
16  static void Вывести(double[,] a, int n, int m, string формат)
17  {
18  int i, j;
19  for (i = 0; i < n; i++, Console.WriteLine())
20  for (j = 0; j < m; j++)
21  Console.Write(формат, a[i, j]);
22  }
23  static void Переставить (double[,] a, int ном1, int ном2, int m)
24  {
25  int j;
26  double b;
27  for (j = 0; j < m; j++)
28  {
29  b = a[ном1, j];
30  a[ном1, j] = a[ном2, j];
31  a[ном2, j] = b;
32  }
33  }
34  static void Main()
35  {
36  int n, m, mp, ном1, ном2;
37  double[,] a;
38  ConsoleKeyInfo клавиша;
39  do
40  {
41  Console.Clear();
42  Console.WriteLine("Сколько строк:");
43  n = int.Parse(Console.ReadLine());
44  Console.WriteLine ("Сколько столбцов:");
45  m = int.Parse(Console.ReadLine());
46  a = new double[n, m];
47  Прочитать(a, n, m);
48  Console. WriteLine("\nИсходная таблица");
49  Вывести(a, n, m, "{0,8:f2}");
```

```
50  mp = n/2;
51  for(ном1=0,ном2=n-1; ном1<mp; ном1++,ном2--)
52  Переставить( a, ном1, ном2 , m);
53  Console.WriteLine ("\nТаблица после перестановки строк");
54  Вывести(a, n, m, "{0,8:f2}");          ,
55  Console.WriteLine ("\nДля выхода нажмите клавишу ESC");
56  клавиша = Console.ReadKey(true);
57  } while (клавиша.Key != ConsoleKey.Escape);
58  }
59  }
60  }
```

Таблица 8 - Словарь операторов и операций программы

№ п/п	Операторы, операции	Номера строк	Количество повторений
1	using…;	1	1
2	namespace	2	1
3	class ...	4	1
4	static void...	6, 16, 23, 34	4
5	double [,]	6, 16, 23, 37	4
6	int...	6, 6, 8, 16, 16, 18, 23, 23, 23, 23, 25, 36	12
7	string...	16	1
8	ConsoleKeyInfo	38	1
9	for ()	9, 10, 19, 20, 27, 51	6
10	do{}while()	39	1
11	Console.Write()	12, 21, 42, 44	4
12	Console.WriteLine()	19, 48, 53, 55	4
13	Console.ReadLine()	13, 43, 45	3
14	....Parse()	13, 43, 45	3
15	Console.Clear()	41	1
16	Console.ReadKey()	56	1
17	new double[]	46	1
18	new int	41	1
19	Прочитать()	47	1
20	Вывести()	49, 54	2
21	Переставить()	52	1
22	.Key	57	1
23	ConsoleKey.Escape	57	1
24	;	1, 8, 9, 9, 10, 10, 12, 13, 18, 19, 19, 20, 20, 21, 25, 26, 27, 27, 29, 30, 31, 36, 37, 38, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 51, 52, 53, 54, 55, 56, 57	42
25	,	6, 6, 6, 8, 12, 12, 12, 16, 16, 16, 16, 18, 19, 21, 21,23, 23, 23, 23, 29, 30, 30, 31, 36, 36, 36, 36, 37, 46, 47, 47, 49, 49, 49, 49, 51, 51, 52, 52, 52, 54, 54, 54, 54,	44
26	:	49, 54	2
27	/	50	17

28	=	9, 10, 13, 19, 20, 27, 26, 29, 30, 31, 43, 45, 46, 50, 51, 51, 56	1
29	-	51	17
30	++	9, 10, 19, 20, 26, 27, 51	1
31	- -	51	1
32	{}	3(60), 5(59), 7(15), 11(14), 17(22), 24(33), 28(32), 35(58), 40(57)	9
33	()	6, 9, 10, 12, 13,13, 16, 19, 19, 20, 21, 23, 27, 34, 41, 42, 43, 43, 44, 45, 45, 47, 48, 49, 51, 52, 54, 55, 56, 57	30
34	[]	6, 13, 16, 23, 29, 30, 30, 31, 37, 46	10
35	“ ”	12, 42, 44, 48, 49, 54, 55	7
36	/	45	1
37	.	12, 13, 13, 19, 21, 41, 42, 43, 43, 44, 45, 48, 53, 55, 56, 57, 57	17
38	<	9, 10, 19, 20, 27, 51	6
39	!=	57	1
		Всего	**252**

Таблица 9 - Словарь операндов программы

№ п/п	Операторы, операции	Номера строк	Количество повторений
1	System	1	1
2	Exemp	2	1
3	Replace	4	1
4	Прочитать	6	1
5	a	6, 13, 16, 21, 23, 29, 30, 30, 31, 37, 46, 47, 49, 54	14
6	n	6, 9, 16, 19, 36, 43,46, 47, 49, 50, 51, 54	12
7	m	6, 10, 16, 20, 23, 27, 36, 45, 46, 47, 49, 54	12
8	i	8, 9, 9, 12, 13, 18, 19, 19, 21	9
9	j	8, 10, 10, 12, 13, 18, 20, 20, 25, 27, 27, 29, 30, 31	14
10	“Элемент[{0},{1}]:“	12	1
11	формат	16, 21	2
12	ном1	23, 29, 30, 51, 51, 51,5 2	7
13	ном2	23, 30, 31, 36	4
14	b	26, 29, 31	3
15	Main	34	1
16	mp	36, 51	2
17	клавиша	38, 56, 57	3
18	“Сколько строк: “	42	1
19	“Сколько столбцов: “	44	1
20	“\nИсходная таблица”	48	1
21	“{0,8:f2}”	49, 54	1
22	"\nТаблица после перестановки строк"	53	1
23	"\nДля выхода нажмите клавишу ESC"	55	1
24	true	56	1
		Всего	**95**

Таблица 10. - Входные и выходные переменные программы

Входные переменные	Выходные переменные
a	"Элемент[{0},{1}]:"
n	формат
m	"Сколько строк:"
клавиша	a
	"Сколько столбцов:"
	"\nИсходная таблица"
	"\nТаблица после перестановки строк"
	"\nДля выхода нажмите клавишу ESC"

Оценка характеристик программы

Таблица 11. - Значения метрик Холстеда для программы

Наименование характеристики	Обозначение и формула для вычисления	Значение
Число простых (уникальных) операторов и операций	n_1	39
Число простых (уникальных) операндов	n_2	24
Общее число всех операторов и операций	N_1	252
Общее число, всех операндов	N_2	95
Общее число входных и выходных переменных (параметров)	n_2^*	12
Словарь программы	$n = n_1 + n_2$	63
Длина реализации	$N = N_1 + N_2$	347
Длина программы	$N = n_1 \cdot \log_2 n_1 + n_2 \cdot \log_2 n_2$	316
Объем программы (в битах)	$V = (N_2 + N_1) \cdot \log_2(n_1 + n_2)$	2074
Потенциальный объем программы	$V^* = (n_2 + 2) \cdot \log_2(n_2 + 2)$	107
Уровень реализации программы	$L = V^* / V$	0,051
Уровень языка	$\lambda = L \cdot V^*$	5,48
Работа по программированию	$E = V / L$	40354

Выводы:

1) Длина реализации больше расчетной длины программы на 9%. В программе присутствуют несовершенства. Проявлением несовершенства программы являются строки:

- *b = a[ном1, j];*
- *a[ном1, j] = a[ном2,j];*

- *a[ном2, j] = b.*

Представленная последовательность присваиваний очень близка к неоднозначности операндов.

2) Уровень реализации исследуемой программы весьма низкий, так как потенциальный объем программы в значительной степени меньше ее реального объема.

3) Возможности языка программирования использованы на низком уровне.

Задача 3. «Заправка бака топливом»

Определить класс «Бак», описывающий понятие «Топливный бак». Данный класс должен иметь следующие ноля:

- ширина, длина и высота в сантиметрах;
- вид топлива.

В классе должны присутствовать операции (методы):

- полная заправка бака заданным видом топлива;
- вычисление стоимости полной заправки бака.

Бак имеет вид параллелепипеда. Стандартным считается бак, имеющий вид куба (ширина, длина и высота совпадают).

Возможные разновидности баков:

- бак с одинаковой шириной и длиной и индивидуальной высотой;
- бак с индивидуальными размерами по ширине, длине и высоте.

Стоимость одного кубического сантиметра топлива определяется полями класса «Топливо». Стоимость топлива в рамках решения задачи неизменна. Выдача стоимости топлива должна быть реализована специальной операцией этого класса.

Заполнить и вывести па экран стоимость заправки четырех баков:

- 10 x 20 x 30 см, топливо - газ;
- 10 x 10 x 20 см, топливо - керосин;
- 10 x 10 x 10 см, топливо - бензин;
- 20 x 20 x 20 см, топливо - бензин.

Разработать программу и определить значения метрик Холстеда, на основе которых дать оценку качества разработанного исходного текста программы.

Таблица 12 – Реализация программы для задачи «Заправка бака топливом» на языке C#.

Номера строк	Строки программы
1	using System;
2	namespace EX2
3	{
4	class Топливо
5	{
6	private static double ценаГаз = 0.05;
7	private static double ценаКеросин = 0.08;
8	private static double ценаБензин = 0.1;
9	public static double Цена(string вид)
10	{
11	switch (вид)
12	{
13	case “газ”: return ценаГаз;
14	case “керосин”: return ценаКеросин;
15	case “бензин”: return ценаБензин;
16	default: return 0.0;
17	}
18	}
19	}
20	class Бак
21	{
22	private double x, y, h;
23	private string видТоплива;
24	public void Заполнить(double xb, string вид)
25	{

```
26  x = xb; y = xb; h = xb; видТоплива = вид;
27  }
28  public void Заполнить(ref double xb, string вид)
29  {
30  x = xb; y = xb; h = xb; видТоплива = вид;
31  }
32  public void Заполнить(double xb, double yb, string вид)
33  {
34  x = xb; y = yb; h = xb; видТоплива = вид;
35  }
36  public void Заполнить(double xb, double yb, double hb, string вид)
37  {
38  x = xb; y = yb; h = xb; видТоплива = вид;
39  }
40  public double Оплата()
41  {
42  return x * y * h * Топливо.Цена(видТоплива);
43  }
44  }
45  class Program
46  {
47  static void Main()
48  {
49  double x = 20.0;
50  Бак b;
51  b = new Бак();
52  b.Заполнить( 10.0,20.0,30.0, “газ”);
53  Console.WriteLine(«Стоимость заправки: « + b.Оплата());
54  b.Заполнить( 10.0, 20.0,«керосин»);
55  Console.WriteLine(«Стоимость заправки: « + b.Оплата());
56  b.Заполнить( 10.0,«бензин»);
57  Console.WriteLine(«Стоимость заправки: « + b.Оплата());
58  b.Заполнить(reP x, «бензин»);
59  Console.WriteLine(«Стоимость заправки: « + b.Оплата());
60  Console.ReadKey();
61  }
62  }
63  }
```

Таблица 13. - Словарь операторов и операций программы

№ п/п	Операторы, операции	Номера строк	Количество повторений
1	using...	1	1
2	namespace	2	1
3	class...	4	1
4	double...	24, 28, 32, 36, 40	5
5	void...	9, 16, 23, 34, 47	5
6	double	6, 7, 8, 22, 24, 28, 32, 32, 36, 36, 36, 49	12
7	string ...	9, 23, 24, 28, 32, 36	6
8	Бак...	50	1
9	Console.WriteLine()	53, 55, 57, 59	4
10	Console.ReadKey()	60	1
11	new Бак()	51	1
12	Заполнить()	52, 54, 56, 58	4
13	Оплата()	53, 55, 57, 59	4
14	return	13, 14, 15, 16, 42	5
15	Топливо.Цена()	42	1
16	;	6, 7, 8, 13, 14, 15, 16, 22, 23, 26, 26, 26, 26, 30, 30, 30, 30, 34, 34, 34, 34, 38, 38, 38, 38, 42, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60	38
17	,	22, 22, 24, 28, 32, 32,36, 36, 36, 52, 52, 52, 54, 54, 56, 58	16
18	*	42, 42, 42	3
19	=	6, 7, 8, 26, 26, 26, 26, 30, 30, 30, 30, 34, 34, 34, 34, 38, 38, 38, 38, 49, 51	21
20	:	13, 14, 15, 16 ,	4
21	+	53, 55, 57, 59	4
22	{}	3(63), 5(19), 10(18), 12(17), 21(44), 25(27), 29(31), 33(35), 37(39), 41(43), 46(62), 48(61)	12
23	()	9, 24, 28, 32, 36, 40, 42, 47, 51, 52, 53, 53, 54, 55, 55, 56, 57, 57, 58, 59, 59, 60	22
24	“ ”	13, 14, 15, 52, 53, 54, 55, 56, 57, 58	10
25	switch (вид)	11	1
26	.	6, 7, 8, 16, 42, 49, 52, 52, 52, 53, 54, 54, 54, 55, 56, 56, 57, 57, 58, 59, 59, 60	21

27	case	13, 14, 15	3
28	default	16	1
		Всего	**208**

Таблица 14 - Словарь операндов программы

№ п/п	Операнды	Номера строк	Количество повторений
1	System	1	1
2	EX2	2	1
3	Топливо	4	1
4	ценаГаз	6, 13	2
5	0.05	6	1
6	ценаКеросин	7, 14	2
7	ценаБензин	8,15	2
8	0.08	7	1
9	0.1	8	1
10	Цена	9	1
11	вид	9, 11, 24, 26, 28, 30, 32, 34, 36, 38	10
12	"газ"	13, 52	2
13	"керосин"	14, 54	2
14	"бензин"	15, 56, 58	3
15	0.0	16	1
16	x	22, 26, 30, 34, 38, 42, 49,5 8	8
17	y	22, 26, 30, 34, 38, 42	6
18	h	22, 26, 30, 34, 38, 42	6
19	видТоплива	23, 26, 30, 34, 38, 42	6
20	Заполнить	24	1
21	xb	24, 26, 26, 28, 30, 30, 30, 32, 34, 34, 36, 38	12
22	yb	32, 34, 36, 38	4
23	hb	36, 38	2
24	20.0	49,52,54	3
25	10.0	52, 54, 56	3
26	b	50, 51, 52, 53, 54, 55, 56, 57, 58, 59	10
27	Main	47	1
		Всего	**93**

Таблица 15 - Входные и выходные переменные программы

Входные переменные	Выходные переменные
	"Стоимость заправки:"
	b.Оплата()

Оценка характеристик программы

Таблица 16 - Значения метрик Холстеда для программы

Наименование характеристики	Обозначение и формула для вычисления	Значение
Число простых (уникальных) операторов и операций	n_1	28
Число простых (уникальных) операндов	n_2	27
Общее число всех операторов и операций	N_1	2208
Общее число, всех операндов	N_2	93
Общее число входных и выходных переменных (параметров)	n_2^*	2
Словарь программы	$n = n_1 + n_2$	55
Длина реализации	$N = N_1 + N_2$	301
Длина программы	$N = n_1 \cdot \log_2 n_1 + n_2 \cdot \log_2 n_2$	263
Объем программы (в битах)	$V = (N_2 + N_1) \cdot \log_2(n_1 + n_2)$	1740
Потенциальный объем программы	$V^* = (n_2 + 2) \cdot \log_2(n_2 + 2)$	4
Уровень реализации программы	$L = V^* / V$	0,002
Уровень языка	$\lambda = L \cdot V^*$	0,008
Работа по программированию	$E = V / L$	757065

Выводы:

1) Длина реализации превышает расчетную длину программы больше чем на 10%. В программе присутствуют несовершенства. Проявлением несовершенства программы являются строки:

- *x = xb; y = xb; h = xb;* (таблица 12, строки 30, 34);
- *x = xb; h = xb;* (таблица 12, строка 38).

Представленная последовательность присваиваний представляет собой набор операций с синонимичными операндами, что в значительной степени снижает качественные характеристики программы.

2) Уровень реализации исследуемой программы весьма низкий, так как потенциальный объем программы в значительной степени меньше ее реального объема.

3) Возможности языка программирования использованы на низком уровне.

1.3 Задания для аудиторной работы

Задача 1. «Зеркальное число»

Ввести с клавиатуры трехзначное натуральное число. Вычислить и вывести на экран зеркальное число, полученное путем изменения цифр числа на обратное по отношению к исходной позиции (например, для числа 253 должно быть получено 352). Разработать программу и определить значения метрик Холстеда, на основе которых дать оценку качества разработанного исходного текста программы.

Таблица 17 – Реализация программы для задачи «Зеркальное число» на языке C#.

Номера строк	Строки программы
1	using System;
2	class Revers
3	{
4	public static void Main()
5	{
6	uint ch, copia, cifra, newch;
7	string str;
8	Console.Write("Введите трехзначное натуральное число: ");
9	str = Console.ReadLine();
10	ch = uint.Parse(str);
11	copia = ch;
12	newch = 0;
13	cifra = copia % 10;
14	newch = newch* 10 + cifra;
15	copia /= 10;
16	cifra = copia % 10;
17	newch = newch* 10 + cifra;
18	copia/= 10;
19	cifra=copia % 10;
20	newch = newch* 10 + cifra;
21	copia /= 10;
22	str = "Если перевернуть " + ch + " будет " + newch;
23	Console.WriteLine(slr);
24	Console.ReadLine();
25	}
26	}

Задача 2 «Вычисление суммы элементов массива»

Вычислить сумму элементов целочисленной матрицы размером $M \times N$, расположенных в треугольной области с вершинами:

•середина первой строки;

•первый элемент последней строки;

•последний элемент последней строки.

Пример области для матрицы размером 3×5:

* * 3 * *

* 5 1 7 *

1 3 2 6 2

Элементы, входящие в анализируемую область, отмечены цифрами, остальные элементы - символом *. Разработать программу и определить значения метрик Холстеда, на основе которых дать оценку качества разработанного исходного текста программы.

Таблица 18 – Реализация программы для задачи «Вычисление суммы элементов массива» на языке C#.

Номера строк	Строки программы
1.	using System;
2.	namespace Prim
3.	{
4.	class Summ
5.	{
6.	static void Read(int[,] a, int n, int m)
7.	{
8.	int i, j;
9.	for (i = 0; i < n; i++)
10.	for (j = 0; j < m; j++)
11.	{
12.	Console.Write("Элемент[{0}, {1}]: ",i,j);
13.	a[i, j] = int.Parse(Console.ReadLine());
14.	}
15.	}
16.	static void Show(int[,] a, int n, int m, string form)
17.	{
18.	int i, j;
19.	for (i = 0; i < n; i++, Console. WriteLine())
20.	for (j = 0; j < m; j++)
21.	Console.Write(form, a[i, j]);
22.	} .
23.	static int SumStr(int[,] a, int nom, int perv, int posl)
24.	{
25.	int j,s=0;

26.	for (j = perv; j <= posl; j++)
27.	s += a[nom, j];
28.	return s;
29.	}
30.	static void Main()
31.	{
32.	int n, m, s, nom, perv, posl;
33.	int[,] a;
34.	ConsoleKeyInfo klav;
35.	do
36.	{
37.	Console.Clear();
38.	Console.Write("Сколько строк:");
39.	n = int.Parse(Console.ReadLine());
40.	m = 2 * n -1;
41.	a = new int[n, m];
42.	Read(a, n, m);
43.	Console.WriteLine("\nИсходная матрица");
44.	Show(a, n, m, "{0,8:d}");
45.	for(s=0,nom=0,perv=posl=m/2; nom<n; nom++,perv--,posl++)
46.	s += SumStr(a,nom,perv,posl);
47.	Console.WriteLine("Сумма элементов=" + s);
48.	Console.WriteLineC'/Для выхода нажмите клавишу ESC");
49.	klav = ConsoIe.ReadKey(true);
50.	} while (klav.Key != ConsoleKey.Escape);
51.	}
52.	}
53.	}

1.4 Задания для самостоятельной работы

В задачах, предлагаемых для самостоятельного решения, необходимо выполнить следующее:

- разработать программу, реализующую заданный алгоритм (рекомендуется использовать языки программирования C, C++, C#);
- сформировать словарь программы, охватывающий операнды, операторы и операции;
- словари оформить в виде таблиц;
- рассчитать метрики Холстеда, оформив результат в виде итоговой таблицы;
- провести анализ полученных результатов, сформировав

содержательные выводы содержащие оценку качества разработанного текста программы:

1) оценить уровень программы, сравнив потенциальный и реальный объемы;

2) выявить наличие несовершенств на основе сравнения длины реализации и расчетной длины программы, указать строки программы, содержащие несовершенства и тип несовершенств;

3) оценить уровень использования возможностей языка реализации.

Задача 1. Написать и протестировать функцию, которая «переворачивает» строку, передаваемую ей в качестве параметра, в зеркальное состояние.

Задача 2. Дано натуральное число *N*. Вывести на экран число, которое получится после выписывания цифр числа *N* в обратном порядке. Для получения нового числа составить функцию.

Задача 3. Написать и протестировать функцию, подсчитывающую количество минимальных элементов в каждой строке целочисленной матрицы.

Задача 4. Написать и протестировать функцию для вычисления числа сочетаний по формуле

$$C_n^k = \frac{n!}{k!(n-k)!} = \frac{n(n-1)...(n-k+1)}{k!}$$

Задача 5. По заданным значениям *X*[20], *Y*[20] вычислить

$$u = \begin{cases} \sum_{i=0}^{19} x_i^2; & \text{при } \sum_{i=1}^{15} x_i \cdot y_i > 0; \\ \sum_{i=0}^{19} y_i^2; & \text{при } \sum_{i=1}^{15} x_i \cdot y_i \le 0. \end{cases}$$

Задача 6. Написать и протестировать функцию, преобразующую строку восьмеричных цифр в эквивалентное ей целое десятичное число.

Задача 7. Описать функцию *minmax(x,y),* которая присваивает

первому параметру большее, а второму - меньшее из значений *x* и *y*. Используя эту функцию, перераспределить введенные значения переменных *A, B, C* так, чтобы стало $A \le B \le C$.

Задача 8. Даны две квадратные матрицы. Напечатать ту из них, которая имеет минимальный «след», т. е. сумму элементов главной диагонали. Использовать функцию для нахождения следа матрицы и функцию печати матрицы.

Задача 9. Составить и протестировать функцию для вычисления

$$f(x,n,m)=\sum_{i=m}^{n}\frac{x^{2i}}{(2i)!}.$$

Задача 10. Написать и протестировать функцию *compress(),* которая «сжимает» строку, удаляя из нее все пробелы.

Задача 11. Написать и протестировать функцию, которая подсчитывает, сколько раз в заданной строке встретился указанный символ.

Задача 12. Написать и протестировать функцию, которая находит в массиве минимальный по модулю элемент и заменяет им все элементы с нечетными номерами.

Задача 13. Написать и протестировать функцию, которая в прямоугольной матрице находит сумму элементов *j*-й строки.

Задача 14. Написать и протестировать функцию, которая по заданному натуральному числу определяет количество цифр в нем и их сумму.

Задача 15. Написать и протестировать функцию, которая по заданной строке *Str*, содержащей буквы и цифры, формирует новую строку, состоящую только из цифр, входящих в *Str*.

Задача 16. Написать и протестировать функцию, подсчитывающую количество положительных элементов в массиве.

Задача 17. Написать и протестировать функцию, вычисляющую $y = \sqrt[3]{x}$ $(0 < |x| < 2)$ используя итерационную формулу

$$y_{i+1} = y_i + \frac{1}{3}\left(y_i - \frac{y_i^4}{x}\right).$$

Начальное приближение $y_0 = x$. Итерации прекратить при достижении условия

$$|y_{i+1} - y_i| < 2 \cdot 10^{-6}.$$

Задача 18. Составить и протестировать функцию для замены символов «:» на «.» в заданной строке, начиная с указанной позиции.

Задача 19. Выяснить, сколько простых чисел находится в интервале [*n, m*], и распечатать их. Для определения, является ли Очередное число простым, составить функцию.

Задача 20. Написать и протестировать функцию для вычисления площади треугольника, заданного координатами вершин.

Задача 21. Написать и протестировать функцию для нахождения в прямоугольной матрице номера строки, имеющей максимальную сумму элементов.

Задача 22. Написать и протестировать функцию, которая преобразует строку двоичных цифр в эквивалентное ей целое десятичное число.

Задача 23. Написать и протестировать функцию, которая в строке, передаваемой ей в качестве параметра, заменяет каждый второй элемент на заданный символ.

Задача 24. Написать и протестировать функцию для сложения и вычитания вещественных матриц. Одним из формальных параметров должен быть признак вида операции.

Задача 25. Написать и протестировать функцию, которая преобразует строку шестнадцатеричных элементов числа в

эквивалентное ей целое десятичное число.

2 ОЦЕНКА ПРОГРАММНОГО СРЕДСТВА НА ОСНОВЕ МЕТРИК ДЖИЛБА

2.1. Теоретические сведения

В качестве меры логической трудности Джилб предложил число логических «двоичных принятий решений». Наиболее ценным для практики является то, что такая оценка может быть получена вручную на основе зрительного анализа текста программы либо автоматически с помощью специально разработанных программных анализаторов, причем относительно несложных.

Логическая сложность программы определяется как насыщенность программы условными операторами и операторами цикла. Вводятся следующие характеристики программного средства:

• *CL* – абсолютная сложность программы, характеризуемая количеством операторов условий;

• *cl* – относительная сложность программы, определяющая насыщенность программы операторами условия (вычисляется как отношение абсолютной сложности *CL* к общему числу операторов *L*).

• количество операторов цикла L_{1oop};

• количество операторов условия L_{IF};

• число модулей или подсистем L_{mod};

• отношение числа связей между модулями к числу модулей $f = \frac{N_{SV}^4}{L_{\text{mod}}}$;

• отношение числа ненормальных выходов из множества операторов к общему числу операторов $f^* = \frac{N_{SV}^*}{L}$.

•надежность программы (возможность того, что данная программа проработает определенный период времени без логических сбоев), равную единице минус отношение числа логических сбоев к общему числу запусков;

•мера точности (свободы от ошибок) - отношение количества правильных данных ко всей совокупности данных;

•прецизионность (мера того, насколько часто появляются ошибки, вызванные одинаковыми причинами) – отношение числа фактических ошибок на входе к общему числу наблюдаемых, ошибок, причинами которых явились эти ошибки на входе.

2.2. Примеры решений практических заданий

Задача 1 «Вычисление значений функции»

Функция $Y=F(x)$ задана следующим образом:

$$Y = \begin{cases} 0{,}5 & \text{при } x \le 0{,}5; \\ x + 1 & \text{при } -0{,}5 < x \le 0; \\ x^2 - 1 & \text{при } 0 < x \le 1; \\ x - 1 & \text{при } x > 1. \end{cases}$$

Необходимо разработать программу для вычисления значений этой функции и на основе лексического анализа исходного текста программы оценить ее качество с использованием метрик Джилба.

Реализация программы. Текст программы для реализации возможного алгоритма решения поставленной задачи, разработанный с использованием языка программирования C#, приведен в таблице 19.

Таблица 19 – Реализация программы для задачи «Вычисление значений функции"

Номер строки	Строки программы
1	using System;
2	
3	class Operator

4	{
5	public static void Main()
6	{
7	float x,
8	y;
9	char rep;
10	string str;
11	
12	REPEAT:
13	Console.Clear();
14	Console.Write("Введите аргумент функции: ");
15	str = Console.ReadLine();
16	x = float.Parse(str);
17	if (x <= 0)
18	if (x <= -0.5)
19	y = (float)0.5;
20	else
21	y = (float)(x + 1.0);
22	else
23	if (x <= 1.0)
24	y = (float)(x * x - 1.0);
25	else
26	y = (float)(x - 1.0);
27	
28	str = "F(" + x + ")=" + y;
29	Console.WriteLine(str);
30	
31	Console.Write("Для повтора вычислений нажмите клавишу Y: ");
32	
33	rep = char.Parse(Console.ReadLine());
34	if (rep = = 'Y' || rep = = 'y') goto REPEAT;
35	}
36	}

Программа написана без использования операторов цикла. Повторение процедур выполнения операторов программы реализовано с помощью оператора `goto`, применение которого не рекомендуется при разработке профессионального программного обеспечения, поскольку затрудняет понимание программы. Однако в учебных целях применение данного оператора будем считать допустимым.

Программа не имеет циклов и модулей. Следовательно, можно определить следующие характеристики: L, L_{IF}, CL и cl, причем $CL = L_{IF}$.

При подсчете общего количества операторов L программы будем руководствоваться правилами, определенными при подсчете операторов

в метриках Холстеда.

В таблице 20 приведены операторы и операции, используемые в программе. При подсчете числа операторов следует иметь в виду, что в строке 28 (см. таблицу 19) скобки используются как символы строковых констант, а потому операторами не являются.

Таблица 20. Словарь операторов и операций программы

№п/п	Операторы, операции	Номера строк	Количество повторений
1	using...	1	
2	class...	3	
3	{}	4(35), 6(34)	
4	public static void	5	
5	float	7	
6	char	8	
7	string	10	1
8	Console.Clear()	13	1
9	Console.Write()	14, 31	2
10	=	15, 16, 19, 21, 24, 26, 28, 32	8
11	Console.ReadLine()	15, 32	2
12	....Parse	16, 32	2
13	if()... else...	17(22), 18(20), 23(25)	3
14	if()...	33	1
15	<=	17, 18, 23	3
16	+	21, 28	2
17	-	24, 26	2
18	*	24	1
19	Console.WrileLine()	29	1
20	= =	33	2
21	||	33	1
22	goto	33	1
23	;	1, 8, 9, 10, 13, 14, 15, 16, 19, 21, 24, 26, 28, 29, 31, 32, 33	17
24	,	7	1
25	.	13, 14, 15, 16,18, 19, 21, 23, 24, 26, 29, 31, 32, 32,	14
26	()	5, 13, 14, 15, 16, 17, 18, 19, 21, 21, 23, 24, 24, 26, 26, 29,31,32, 32, 33	20
		Всего	**92**

Оценка характеристик программы

В языке программирования C# к категории операторов условия могут относиться условные операторы ветвления `if` и операторы циклов `for … while` и `do … while`, которые в своем составе в обязательном порядке содержат логическое выражение, являющееся

условием выполнения указанных операторов.

В исходном тексте программы используются условные операторы `if`, которые располагаются в строках с номерами 17,18, 23 и 33 (см. таблицу 19). Исходя из полученных данных (таблица 20) получаем следующие результаты оценки характеристик программы:

• число операторов условий L_{IF} равно 4 (таблица 19, п. 13 и 14);

• абсолютная сложность $CL = 4$, так как в программе используется четыре оператора условия;

• относительная сложность программы равна:

$cl = CL/L = 4/92 = 0{,}0435$.

С точки зрения лексического анализа исходного текста программы представленное решение не является сложным, так как количество ветвлений в программе весьма невелико (4 оператора условия), что подтверждается невысокой величиной метрики относительной сложности программы.

Задача 2 «Функция копирования элементов массива»

Необходимо разработать функцию, которая копирует положительные элементы из одного одномерного целочисленного массива в другой массив. Используя этот метод, следует выполнить копирование положительных элементов двух исходных массивов *A* и *B* в массив *C*.

Размер исходных массивов *A* и *B* ввести с клавиатуры. Эти массивы заполнить случайными числами из диапазона от –100 до 300. Сформированный массив *C* вывести на экран.

Выдать сообщение на экран в случае, когда массив *C* оказывается пустым. Для разработанной программы на основе лексического анализа исходного текста определить значения метрик Джилба.

Реализация программы

Рассмотрим вариант реализации возможного алгоритма решения поставленной задачи, разработанный с использованием языка программирования C#, в котором применяются программные модули. Пример реализации такой программы приведен в таблице 21.

Таблица 21 – Реализация программы для задачи «Функция копирования элементов массива»

Номера строк	Строки программы
1	using System;
2	class Exempl
3	{
4	public static int[] Copy(int[] a)
5	{
6	int [] b = new int[a.Length];
7	int j=0;
8	for(int i=0; i<a.Length; i++)
9	{
10	if(a[i]>=0) {b[j)=a[i];j++;}
11	}
12	return b;
13	}
14	
15	public static void Zapoln(ref int[] a. Random g)
16	{
17	for (int i = 0; i < a.Lcngth; i++)
18	{
19	a[i] = g.Next(-100, 300);
20	}
21	}
22	
23	public static void Print(int[] a, string str. string strl)
24	{
25	Console.WritcLine(strl);
26	for (int i = 0: i < a.Lcngth; i++)
27	{
28	if(a[i]!=0) Console.Write(str, a[i]);
29	}
30	Console. WriteLine();
31	}
32	
33	public static void Main()
34	{
35	int[] a, b, c, p;
36	Random g = new Random();
37	char r;
38	do
39	{
40	Console.Clear();

41	Console.WriteLine("Определите размер первого массива!");
42	a = new int[int.Parse(Console.ReadLine())];
43	Console.WriteLine("Определите размер второго массива!");
44	b = new int[int.Parse<Console.ReadLine())];
45	c = new int[a.Length + b.Lcngth];
46	Exempl.Zapoln(ref a, g);
47	Exempl.Zapoln(ref b, g);
48	Exempl.Print(a," {0,5}", "Первый исходный массив!!!");
49	Exempl.Print(b," {0,5}", "Второй исходный массив!!!");
50	p = Exempl.Copy(a);
51	Array.Copy(p, 0.c, 0, a.Length);
52	p = Exempl.Copy(b);
53	Array.Copy(p, 0, c, a.Length, b.Length);
54	Exempl.Print(c," {0,5}". "Результатный массив!!!");
55	Console.WriteLine():
56	Console. WriteLine("Выполнить повтор программы? Y/N");
57	r = char.Parsc(Console.ReadLine());
58	} while (r = = 'Y' || r = = 'y');
59	}
60	}

В 6-й, 42-й, 44-й и 50-й строках (таблица 21) ключевые слова `int[…]` представляют собой операторы вызова метода конструктора. Ключевое слово `Random` в 36-й строке используется дважды: в первом случае это оператор описания типа, во втором – вызов метода-конструктора.

В таблице 22 приведены операторы и операции, используемые в программе.

Таблица 22. - Словарь операторов и операций программы

№п/п	Операторы, операции	Номера строк	Количество повторений
1	using...	1	1
2	class...	2	1
3	public static...	4, 15, 23, 33	4
4	int[]	4, 6, 15, 23, 35	5
5	new	6, 36, 42, 44, 45	5
6	int	7, 8, 15, 17, 26,	5
7	string	23, 23	2
8	char	37	1
9	Random	15, 36	2
10	. Length	17, 6. 8, 26, 45, 45, 51, 53	8
11	.Next	19	1
12	Console.WriteLine()	25, 30, 41, 43, 55, 56	6
13	Console.Write()	28,	1
22	Console.ReadLine()	42, 44, 57	3
15	for()	8, 17, 26	3

16	do{}while()	38-58	1
17	if()	10, 28	1
18	Console.Clear()	40	1
19	Exempl.Zapoln()	46, 47	2
20	.Parse()	42, 44, 57	3
21	Exempl.Print()	48, 49. 54	3
22	Exempl.Copy()	50, 52	2
23	Array.Copy()	51, 53	2
24	=	6, 7, 8, 10, 17, 19, 26, 36, 42, 44, 45, 50, 52, 57	14
25	>=	10	1
26	!=	28	1
27	= =	58, 58	2
28	<	8, 17, 26	3
29	++	8, 10, 17, 26	4
30	return	12	1
31	[]	4, 4, 6, 6, 10, 10, 10, 15, 19, 23, 28, 28, 35 ,42, 44, 45,	16
32	()	4, 8, 10, 15, 17. 19, 23, 25, 26, 28, 28, 30, 33, 36, 40, 41, 42, 42, 43, 44, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 57, 58	35
33	{}	3(60), 5(13), 9(11), 16(21), 18(20), 24(31), 27(29), 34(59), 39(58), 10, 48, 49, 54.	13
34	’	15, 19, 23, 23, 28, 35, 35, 35, 46, 47, 48, 48, 49, 49,51, 51, 51, 51, 53, 53, 53, 53, 54, 54, 54	25
35	;	1, 6, 7, 8, 8, 10, 10, 12, 17, 17, 19, 25, 26, 26, 28, 30, 35, 36, 37,40. 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58	38
36	.	6, 8, 17, 19, 26, 28, 30, 40, 41, 42, 42, 43, 44, 44, 45,45, 46, 47,48, 49, 50, 51, 51, 52, 53, 53, 53, 54, 55, 56. 57. 57	32
37	int[...]	6, 42, 44, 45	4
38	Random()	36	1
39	“ ”	41, 43, 48, 48, 49, 49, 54, 56	8
40	‘ ’	58, 58	2
		Всего	**263**

Оценка характеристик программы

Число операторов условий L_{IF} равно 2 (таблица 22, п. 17);

Значение характеристики L_{loop} определяется количеством используемых в программе циклов. В исходном тексте данной программы содержится 4 цикла: 3 оператора `for` и 1 `do… while` (таблица 22, п. 15 и 16).

Значение характеристики L_{mod} определяется количеством используемых программных модулей в решении.

В представленном решении используется четыре программных модуля, каждый из которых определяется следующими строками:

• *Public static void Main()* (таблица 21, строка 33);

• *Public static int[] Copy(int[] a)* (таблица 21, строка 4);

• *Public static void Zapoln(ref int[] a, Random g)* (таблица 21, строка 15);

• *Public static void Print(int[] a, string str, string str1)* (таблица 21, строка 23).

Общее количество операторов условия 6, из них 2 оператора `if` и четыре оператора цикла. Общее число всех используемых операторов $L = 263$ (таблица 22). Таким образом:

• $CL = 6$ – абсолютная сложность программы;

• $cl = CL / L = 6 / 263 = 0{,}0228$.

Количество связей между NSV модулями равно трем – по одной связи между основным и каждым из дополнительных модулей. Отношение числа связей к числу модулей определяется следующим образом:

$$f = \frac{N_{SV}^4}{L_{\text{mod}}} = \frac{3^4}{4} = \frac{81}{4} = 20{,}25$$

Из полученных результатов анализа текста программы следует, что исходный код имеет невысокую сложность, так как на 263 оператора текста приходится всего лишь 6 операторов условий. Общее число программных модулей решения также невелико (4 модуля), что подтверждает низкий уровень сложности программы.

Задача 3 «Дополнение массива»

В массивах *A* и *B* хранятся целые числа. Массивы заполняются

исходными данными от датчика случайных чисел в диапазоне значений от 10 до 50. Добавить в конец массива *A* все четные по значению элементы массива *B*.

Вывести на экран следующие элементы:

- исходный массив *A;*
- исходный массив *B;*
- модифицированный массив *A*.

На основе лексического анализа исходного текста программы определить значения метрик Джилба.

Реализация программы

Текст программы для реализации возможного алгоритма решения поставленной задачи представлен на языке C#. Решение представлено в виде одномодульной программы (таблица 23).

Таблица 23 – Реализация программы для задачи «Дополнение массива»

Номера строк	Строки программы
1	using System;
2	
3	namespase EX1
4	{
5	Class Program
6	{
7	static void Main()
8	{
9	int n,I,m,k,min = 10,max = 50;
10	int[] a,b;
11	ConsoleKeyInfo клавиша;
12	Random генератор = new Random();
13	
14	do
15	{
16	Console.Clear();
17	Console.Write("Сколько элементов в массиве A:");
18	n = int.Parse(Console.ReadLine());
19	Console.Write("Сколько элементов в массиве B:");
20	m = int.Parse(Console.ReadLine());
21	a = new int[n + m];
22	b = new int[m];
23	

```
24  for (i = 0; i < a.Length; i++)
25  a[i] = генератор.Next(min, max + 1);
26
27  for (i = 0; i < b.Length; i++)
28  b[i] = генератор.Next(min, max + 1);
29
30
31  Console.WriteLine("Массив A");
32  for (i = 0; i < n; i++)
33  {
34  Console.Write(" {0,4}", a[i]);
35  }
36
37  Console.WriteLine("\nМассив B");
38  for ( i = 0; i < n; i++)
39  {
40  Console.Write("{0,4}", b[i]);
41  }
42
43
44  for (k = n, i = 0; i < b.Length; i++)
45  if (b[i] % 2 = = 0)
46  a[k++] = b[i];
47
48
49  Console.WriteLine("\nМассив A");
50  for (i = 0; i < a.Length; i++)
51  {
52  Console.Write("{0,4}", a[i]);
53  }
54  Console.WriteLine("\nДля выхода нажмите клавишу ESC");
55  клавиша = Console.ReadKey(true);
56  } while (клавиша.Key!= ConsoleKey.Escape);
57
58  }
59  }
60  }
```

Таблица 24. Словарь операторов и операций программы

№п/п	Операторы, операции	Номера строк	Количество повторений
1	using ...	1	1
2	namespace...	3	1
3	class ...	5	1
4	{}	4(60), 6(59), 8(59), 15(56), 33(35), 39(41), 51(53), 34, 40, 52	10
5	static	7	1
6	int	9	1
7	int []	10	1
8	ConsoleKeylnfo	11	1
9	Random	12	1

10	new...	12, 21, 22	3
11	Random()	12	1
12	генератор.NextO	25, 28	2
13	do... while()	14(56)	1
14	Console.Clear()	16	1
15	Console. Write()	17, 19, 34, 40, 52	5
16	=	9, 9, 12, 18, 20, 21, 22, 24, 25, 27, 28, 32, 38, 44, 44, 46, 50, 55	18
17	Console.ReadLine()	18, 20,	2
18	....Parse	18, 20	2
19	int[...]	21, 22	2
20	if ()	45	1
21	for(...)	24, 27, 32, 38, 44, 50,	6
22	.Length	24, 27, 44, 50	4
23	<	24, 27, 32, 38, 44, 50	6
24	%	45	1
25	+	21, 28,	2
26	++	24, 27, 32, 38, 44, 46, 50,	7
27	!=	56	1
28	Console.WriteLine()	31, 37, 49, 54	4
29	= =	45	1
30	;	1, 7, 9, 10, 11, 12, 16, 17, 18, 19,20, 21, 22, 24,24, 25, 28, 31, 32, 32, 34, 37, 38, 38, 40, 44, 44, 46, 49, 50, 50, 52, 54, 55, 56	36
31	,	9, 10, 25, 28, 34, 40, 44, 52,	15
32	.	16, 17, 18, 19, 20, 24, 25, 27, 28, 31, 34, 37, 40, 44, 49, 50, 52, 54, 55, 56	23
33	()	7, 12, 16, 17, 18, 19, 20, 24, 25, 27, 28, 31, 32, 34, 37, 38, 40, 44, 45, 49, 50, 52, 54, 55, 56	27
34	[]	10, 21, 22, 25, 28, 34, 40, 45, 46, 46, 52	11
35	“ ”	17, 19, 31, 34, 37, 40, 49, 52, 54	9
		Всего	**209**

В 9-й и 10-й строках (таблица 23) ключевые слова *int* и *int*[...] представляют собой операторы описания переменных и массива, в 21-й и 22-й строках *int*[...] - оператор вызова метода-конструктора.

Оценка характеристик программы

В тексте программы содержится шесть операторов *for* (таблица 23, строки 24, 27, 32, 38, 44 и 50), один оператор *do ... while* (таблица 23, строки 14—56), один оператор *if* (таблица 23, строка 45).

В результате подсчета количества операторов, используемых в программе, можно определить следующие характеристики:

- L_{loop} = 7 - количество циклов;
- L_{IF}= 1 - количество условных операторов;
- L = 209 - общее количество операторов в программе;
- CL = 7+1=8 - общее количество операторов условий, из них 7 операторов циклов, в которых содержатся условия, и один условный оператор;
- $cl = CL/L$ = 8/209 = 0,038.

Программа с точки зрения метрик Джилба имеет невысокую логическую сложность, так как насыщенность текста операторами условий незначительна.

2.3 Задания для аудиторной работы

Задача 1 «Сложение элементов матриц» (вариант 1)

Создать две матрицы, размер которых вводится с клавиатуры, и заполнить их с помощью датчика случайных чисел. Затем осуществить поэлементное сложение матриц. Исходные и результирующую матрицы вывести на экран монитора, причем отсортировать строки результирующей матрицы по возрастанию. Модифицированную матрицу также вывести на экран монитора. Для разработанной программы на основе лексического анализа исходного текста определить значения метрик Джилба.

Реализация программы

Текст программы для реализации возможного алгоритма решения поставленной задачи представлен на языке C#. Решение представлено в виде одномодульной программы (таблица 25).

Таблица 25 – Реализация программы для задачи «Сложение элементов матриц» (вариант 1)

Номера строк	Строки программы
1	using System;
2	class Ex2
3	{
4	public static void Main()
5	{
6	int[] a, b, c;
7	int i, j, m, n, buf=0;
8	char rep;
9	bool bl;
10	Random g = new Random();
11	do
12	{
13	Console.Clear();
14	Console.WriteLine("Размерность обрабатываемых матриц!\n");
15	Console.WriteLine("Количество строк");
16	n = int.Parse(Console.ReadLine());
17	Console.WriteLine("Количество столбцов");
18	m = int.Parse(Console.ReadLine());
19	a=new int[n][];
20	b=new int[n][];
21	c=new int[n][];
22	for (i = 0; i < n; i++)
23	{
24	a[i] = new int[m];
25	b[i] = new int[m];
26	c[i] = new int[m];
27	}
28	for(i=0;i<a.Length;i++)
29	for(j=0;j<a[i].Length j++)
30	a[i][j]=g.Next(-20,21);
31	for (i = 0; i < a.Length; i++)
32	for (j = 0; j < a[i].Length; i++)
33	b[i][j] = g.Next(-20, 21);
34	for (i = 0; i < c.Length; i++)
35	for (j = 0; j < c[i].Length; j++)
36	c[i][j] = a[i][j] + b[i][j];
37	Console.WriteLine("\nПервая матрица”);
38	for (i = 0; i < a.Length; i++,Console.WriteLine())
39	for (j = 0; j < a[i] .Length; j++)
40	Console.Write(" {0,3}",a[i][j]);
41	Console.WriteLine("\nВторая матрица");
42	for (i = 0; i < b.Length; i++, Console.WriteLineO)
43	for (j = 0; j < b[i].Length; j++)
44	Console.Write(" {0,3}", b[i][j]);
45	Console.WriteLine("\nРезультатная матрица");
46	for (i = 0; i < c.Length; i++, Console.WriteLineO)
47	for (j = 0; j < c[i],Length; j++)
48	Console.Write(" {0,3}", c[i][j]);
49	bl = true;

```
50  while (bl)
51  {
52  bl = false;
53  for (i = 0; i < c.Length; i++)
54  for (j = 0; j < c[i].Length-l; j++)
55  {
56  if(c[i][j]>c[i][j+1])
57  {
58  buf=c[i]|j];
59  c[i][j] = c[i]U + 1];
60  c[i][j + 1] = buf;
61  bl = true;
62  }
63  }
64  }
65  Console.WriteLine("\nОтсортированная результатная матрица");
66  for (i = 0; i < c.Length; i++, Console. WriteLine())
67  for (j = 0; j < c[i],Length; j++)
68  Console.Write(" {0,3}", c[i][j]);
69  Console.WriteLine("\nПовторить расчет? Y/N");
70  rep=char.Parse(Console.ReadLine());
71  }while(rep='Y'||rep='y');
72  }
73  }
```

Задача 2 «Сложение элементов матриц» (вариант 2)

Создать две матрицы, размер которых вводится с клавиатуры, и заполнить их с помощью датчика случайных чисел. Затем осуществить поэлементное сложение матриц. Исходные и результирующую матрицы вывести на экран монитора, причем отсортировать строки результирующей матрицы по возрастанию. Модифицированную матрицу также вывести на экран монитора. Для разработанной программы на основе лексического анализа исходного текста определить значения метрик Джилба.

Реализация программы

Текст программы для реализации возможного алгоритма решения поставленной задачи представлен на языке C#. Решение представлено в виде одномодульной программы (таблица 26).

Таблица 26 – Реализация программы для задачи «Сложение элементов матриц» (вариант 2)

Номера строк	Строки программы
1	using System;

```
2   class Ex2
3   {
4   public int[][] Sozd(int n, int m)
5   {
6   int [][] a=new int[n][];
7   for(int i=0; i<n; i++)
8   {
9   a[i]=new int[m]
10  }
11  return a;
12  }
13  public void Zpoln (int[][] a, Random g)
14  {
15  for(int i=0; i<a.Length; i++)
16  for(int j=0; j<a[i].Length; j++)
17  a[i][j]=g.Next(-20,21);
18
19  }
20  public void Print(int[][] a, string str)
21  {
22  Console.WriteLine(str)
23  for(int i=0; i<a.Length; i++, Console.WriteLine())
24  for(int j=0; j<a[i].Length; j++)
25  Console.Write(“{0,4}”, a[i][j]);
26  }
27  public void Sort(int[][] a)
28  {
29  for (int i = 0; i < a_Length; i++)
30  for (intj = 0; j < a[ij.Length; j++)
31  Array. Sort(a[i]);
32  }
33  public static void Main()
34  {
35  int[][] a, b, c;
36  int n,m,i j;
37  Ex2 e=new Ex2();
38  Random g=new Random();
39  char rep;
40  do
41  {
42  Console.Clear();
43  Console.WriteLine("Размерность обрабатываемых матриц!\n");
44  Console. WriteLine("Количество строк");
45  n = int.Parse(Console.ReadLine());
46  Console.WriteLine("Количество столбцов");
47  m = int.Parse(Console.ReadLine());
48  a=e.Sozd(n,m);
49  b=e.Sozd(n,m);
50  c=e.Sozd(n,m);
51  e.Zpoln(a,g);
52  e.Zpoln(b,g);
```

```
53  for (i = 0; i < c.Length; i++)
54  for (j = 0; j < c[i].Length; j++)
55  c[i][j] = a[i][j] + b[i][j];
56  e.Print(a,"\nПервая матрица");
57  e.Print(b,"\nВтораа матрица");
58  e.Print(c,"\nРезультатная матрица");
59  e.Sort(c);
60  e.Print(c,"\nОтсортированная результатная матрица");
61  Console.WriteLine("\nПовторить расчет? Y/N");
62  rep=char.Parse(Console.ReadLine());
63  } while (rep = = 'Y' || rep = = 'y');
64  }
65  }
```

2.4 Задания для самостоятельной работы

В предлагаемых задачах необходимо разработать программу, реализующую алгоритм решения по приведенному условию, а затем оценить характеристики разработанной программы на основе лексического анализа текста и применения метрик Джилба.

Задача 1. Определить число, образованное k старшими цифрами введенного с клавиатуры натурального числа. Исходное число и значение *k* вводятся с клавиатуры. Пример: для числа 456771 и *k*=2 искомое число равно 45.

Задача 2. Функция $F(x, y)$ задана следующим образом:

$$F(x,y)=\begin{cases} x-y, & \text{если } x \ge y; \\ x+y, & \text{если } x < y. \end{cases}$$

Вывести на экран в виде таблицы значения функции $F(x, y)$ для значений аргументов x = 0,5 - 0,7 с шагом 0,1 и y = 0,2 — 1,0 с шагом 0,2.

Задача 3. Вычислить значения величины *S*, которая задана следующим образом:

$$S=\sum_{k=2}^{N}\prod_{i=1}^{k-1}\sin\left(\frac{\pi i}{k}\right).$$

Натуральное число *N* вводится с клавиатуры, причем *N*> 2.

Задача 4. Вывести на экран таблицу истинности логической функции трех переменных: $F = \left(a \vee \bar{b}\right) \wedge c$.

При отображении использовать следующий прием: истинное значение отображать в виде 1, а ложное - в виде 0.

Задача 5. Сократить обыкновенную дробь, которая вводится с клавиатуры в виде числителя и знаменателя.

Задача 6. Вычислить переменную *S*, которая задана следующим образом:

$$S = \sum_{k=1}^{N} (-1)^k \cdot (2k+1)!$$

Натуральное число *N* вводится с клавиатуры. Результат вывести на экран.

Задача 7. Вывести на экран таблицу квадратов первых десяти целых положительных чисел.

Задача 8. Вывести на экран таблицу квадратов первых пяти положительных нечетных чисел.

Задача 9. Вычислить сумму заданного диапазона целых положительных чисел. При решении задачи предусмотреть проверку нижней и верхней границ указанного диапазона. Нижняя граница не должна превышать по значению верхнюю границу.

Задача 10. Вычислить сумму первых *N* членов ряда 1, 3, 5, 7, ... Количество суммируемых членов ряда *N* задается с клавиатуры.

Задача 11. Вычислить сумму первых *N* членов ряда $1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots$. Количество суммируемых элементов *N* задается с клавиатуры.

Задача 12. Вычислить сумму первых *N* членов ряда $\frac{1}{1}+\frac{3}{2}+\frac{5}{3}+\frac{7}{4}+\ldots$. Количество суммируемых элементов *N* задается с клавиатуры.

Задача 13. Вычислить сумму первых *N* членов ряда $\frac{1}{2}+\frac{3}{4}+\frac{5}{6}+\ldots.$ Количество суммируемых элементов *N* задается с клавиатуры.

Задача 14. Вычислить сумму первых *N* членов ряда $\frac{1}{1}+\frac{8}{9}+\frac{15}{17}+\frac{22}{25}+\ldots$. Количество суммируемых элементов *N* задается с клавиатуры.

Задача 15. Вычислить сумму цифр натурального числа, задаваемого с клавиатуры.

Задача 16. Вычислить сумму *N* младших (правых) цифр натурального числа, задаваемого с клавиатуры. Количество суммируемых цифр *N* задается с клавиатуры.

Задача 17. Вычислить сумму цифр натурального числа, находящихся на четных позициях (старшая цифра находится на первой позиции). Число задается с клавиатуры.

Задача 18. Вычислить значения функции *f*(*x*):

$$f(x)=\begin{cases}\sin\dfrac{\pi}{2}, & \text{если } x\le 0{,}5;\\ \sin\left(\dfrac{\pi}{2}(x-1)\right), & \text{если } x>0{,}5.\end{cases}$$

Вычисления проводить для значений аргумента *x* от -0,4 до 1,3 с шагом 0,1.

Задача 19. Вычислить значения функции $f(x)=\sqrt{x}$ по следующей итерационной формуле: $y_{i+1}=0,5\left(y_i+\frac{x}{y_i}\right)$; $y_0=x$. Итерации прекратить при выполнении условия $|y_{i+1}-y|<2\cdot10^{-5}$.

Задача 20. Вычислить значения функции $Z(x,m)=x^m(\sin(x\cdot m))^m$. Вычисления проводить без использования метода Math.Pow() для значений аргументов: *x* от -1,1 до 0,3 с шагом 0,2; *m* от 1 до 5 с шагом 1.

Задача 21. Определить *N*-ю справа цифру натурального числа. Число и номер цифры *N* вводятся с клавиатуры.

Задача 22. Вычислить значения функции:

$$f(x)=\begin{cases}\sin\left(\frac{\pi}{8}+|x|\right), & \text{если } x<0,3;\\ \sin\left(\frac{\pi}{2}\cdot x^2\right), & \text{если } x\ge 0,3.\end{cases}$$

Вычисления проводить для значений аргументов *x* от -0,5 до 1,2 с шагом 0,1.

Задача 23. Вычислить значения функции:

$$f(x)=\begin{cases}\ln\left|\frac{x}{1+y}\right|, & \text{если } x\ge y;\\ \frac{1+x}{1+y}e^{-|x+y|}, & \text{если } x<y.\end{cases}$$

Вычисления проводить для значений аргументов: *x* от 0,2 до 0,6 с шагом 0,1; *y* от 0 до 0,4 с шагом 0,05.

Задача 24. Вычислить значения функции $f(x) = \sqrt[3]{x}$ по следующей итерационной формуле $y_{i+1} = 0{,}5 \cdot \left(y_i + \frac{3x}{2y_i^2 + \frac{x}{y_i}} \right)$. Принять начальное приближение $y_0 = x$. Итерации прекратить при выполнении условия $|y_{i+1} - y| < 10^{-5}$.

Задача 25. Вычислить значения функции

$f(x) = \sin x + \sin^2(x^2) + \sin^3(x^3)$ для значений аргументах от 0 до 1,2 с шагом 0,1.

3 ОЦЕНКА ПРОГРАММНОГО СРЕДСТВА НА ОСНОВЕ МЕТРИК ЧЕПИНА

3.1 Теоретические сведения

Все множество переменных, составляющих список ввода-вывода, разбивается на четыре функциональные группы:

• *P* – вводимые переменные для расчетов и для обеспечения вывода. Примером такой переменной может служить используемая в программах лексического анализатора переменная, содержащая строку исходного текста программы – в этом случае сама переменная не модифицируется, а применяется только как контейнер для исходной информации;

• *M* – модифицируемые, или создаваемые внутри программы, переменные. К таким переменным относится большинство традиционно применяемых переменных, декларируемых в программных модулях;

• *C* – переменные, участвующие в управлении работой программного модуля (управляющие переменные). Такие переменные

предназначены для передачи управления, изменения логики вычислительных процессов и т. д.;

• T – не используемые в программе (так называемые паразитные) переменные. Такие переменные не принимают непосредственного участия в реализации процесса обработки информации, ради которого написана анализируемая программа, однако они заявлены в программном модуле. Такие переменные могут использоваться для выполнения промежуточных действий и не играют принципиальной роли в решении основной задачи.

В качестве особенности применения метрики следует назвать необходимость учета каждой переменной в каждой функциональной группе, поскольку каждая переменная может выполнять одновременно несколько функций.

Исходное выражение для определения метрики Чепина записывается в следующем виде: $Q = P + 2M + 3C + 0,5T$.

Следует отметить, что метрика сложности программы Чепина также основана на анализе исходных текстов программ, что обеспечивает единый подход к автоматизации их расчета и может быть рассчитана с использованием специально разработанного программного анализатора.

3.2. Примеры решения практических заданий

Задача 1 «Простые числа в матрице»

Дана целочисленная матрица размером $N \times M$. Вычислить и записать в одномерный массив количество простых чисел в каждом столбце матрицы. Размерность матрицы задается с клавиатуры, заполнение матрицы осуществляется посредством датчика случайных чисел. Разработать программу для решения задачи. На основе

лексического анализа исходного текста программы определить значение метрики Чепина.

Реализация программы

Текст программы для реализации возможного алгоритма решения поставленной задачи представлен на языке C# (таблица 27).

Таблица 27 - Реализация программы для задачи «Простые числа в матрице»

Номера строк	Строки программы
1	using System;
2	namespace EX2
3	{
4	class Program
5	{
6	static void Main()
7	{
8	int n,m
9	
10	int[,] a;
11	int[] b;
12	ConsoleKeyInfo клавиша;
13	int i,j,k,d;
14	bool p=false;
15	Random g;
16	do
17	{
18	Console.Clear();
19	Console.Write("Сколько строк: ");
20	n = int.Parse(Console.ReadLine());
21	Console.Write("Сколько столбцов: ");
22	m = int.Parse(Console.ReadLine());
23	g=new Random();
24	a = new int[n, m];
25	for (i = 0; i < n; i++)
26	for (j = 0; j < m; j++)
27	{
28	a[i, j] = int.Parse(g.Next(0,101));
29	}
30	
31	Console.WriteLine("\nИсходная матрица");
32	for (i = 0; i < n; i++, Console.WriteLine())
33	for (j = 0; j < m; j++)
34	Console.Write("{0,8:d}", a[i, j]);
35	b = new int[m];
36	
37	for (j = 0; j < m; j++)
38	{

39	for (i = k = 0; i < n; i++)
40	{
41	p = true;
42	for (d = 2; d < a[i,j]; d++)
43	if (a[i,j] % d == 0) p = false;
44	
45	
46	if (p) k++;
47	b[j] = k;
48	}
49	}
50	
51	
52	Console.WriteLine("\nКоличество простых");
53	for (i = 0; i < b.Length; i++)
54	Console.Write("{0,8:d}", b[i]);
55	
56	Console.WriteLine("\nДля выхода нажмите клавишу ESC");
57	клавиша = Console.ReadKey(true);
58	} while (клавиша.Key!= ConsoleKey.Escape);
59	
60	}
61	}
62	}

Оценка характеристик программы

Рассмотрим текст программы для оценки ее качества с помощью метрики Чепина, которая позволяет оценить меру трудности понимания программы на основе входных и выходных данных. Результат анализа объявленных переменных представлен в таблице 28.

Таблица 28 - Результат анализа объявленных переменных

№ п/п	Наименование переменных	Номера строк
P (для расчётов и обеспечения вывода)		
1	*m*	8
2	*n*	8
3	*a*	10
M (модифицируемые или создаваемые)		
1	*i*	13
2	*j*	13
3	*k*	13
4	*d*	13
5	*b*	11
C (управляющие переменные)		
1	*g*	15
2	*p*	14

3	*клавиша*	12
T (не используемые в программе)		
	Отсутствуют	

Переменные *m*, *n* и *a* используются в качестве исходных данных.

Переменные *i*, *j*, *k*, *d* и *b* в процессе выполнения программы создаются и модифицируются.

Переменные *g*, *p* и *клавиша* используются для управления выполнением программы.

Таким образом, исходя из результатов анализа исходного текста программы, получаем следующие значения характеристик: $P = 3$, $M = 5$, $C = 3$, $T = 0$.

Метрика Чепина: $Q = P + 2M + 3C + 0{,}5T = 3 + 2 \cdot 5 + 3 \cdot 3 + 0{,}5 \cdot 0 = 22$.

Выводы. На основе полученных значений метрики Чепина уровень сложности данного решения можно считать сравнительно низким, как так в исходном тексте программы используется незначительное количество переменных, что не затрудняет понимание программы.

Задача 2 «Сортировка строк матрицы»

Дана вещественная матрица размером *N*M* элементов. Размер матрицы вводится с клавиатуры (*M* не больше 10).

Необходимо отсортировать строки матрицы по возрастанию их поэлементных сумм. Матрица заполняется случайными числами в диапазоне от 1 до 5.

Разработать программу для решения задачи. На основе лексического анализа исходного текста программы определить значение метрики Чепина.

Реализация программы

Текст программы для реализации возможного алгоритма решения поставленной задачи представлен на языке программирования C# (Таблица 29).

Таблица 29 - Реализация программы для задачи «Сортировка строк матрицы»

Номера строк	Строки программы
1	using System;
2	class MyArray
3	{
4	public static void sum(double[,] a, int n, int m, double[] s)
5	{
6	int ij;
7	for(i=0; i<n; i++)
8	for(s[i]=0 j=0; j<m; j++)
9	s[i] += a[ij];
10	}
11	public static void change(double[,] a, int m, int rowl, int row2)
12	*{ '*
13	double buf;
14	int j; //Номер столбца
15	for(j=0; j<m; j++)
16	{
17	buf = a[rowl j];
18	a[rowl j] = a[row2j];
19	a[row2j] = buf;
20	}
21	}
22	public static void sort(double[,] a, int n, int m, doublef] s)
23	{
24	int i;
25	bool flag;
26	double buf;
27	do
28	{
29	n—;
30	for(flag=false,i=0; i<n; i++)
31	if(s[i] > s[i+l])
32	{
33	change(a,m,i,i+l);
34	buf=s[ij;
35	s[i] = s[i+l];
36	s[i+l] = buf;
37	flag = true;
38	}

39	} while(flag);
40	}
41	}
42	class MyMetod
43	{
44	public static void MainQ
45	{
46	double[,] a;
47	double[] s;
48	double min = 1.0,max = 5.0;
49	int n,m;
50	char rep;
51	string sinp;
52	do
53	{
54	Console.Write("Строк: ");
55	sinp = Console.ReadLineQ;
56	n = int.Parse(sinp);
57	Console.Write("Столбцов: ");
58	sinp = Console.ReadLineQ;
59	m = int.Parse(sinp);
60	a = new double[n,m];
61	s = new double[n];
62	IOArray.rand(a,n,m,min,max);
63	IOArray.print(a,n,m,"{0,8:f2}");
64	Console.WriteLineQ;
65	MyArray.sum(a,n,m,s);
66	My Array .sort(a,n,m,s);
67	IOArray.print(a,n,m,"{0,8:f2}");
68	Console.\Угйе("\nДля повтора нажмите клавишу Y: ");
69	rep = char.Parse(Console.ReadLineQ);
70	Console.WriteLineQ;
71	}whilc(rep = = 'Y' || rep = = 'y');
72	}
73	}
74	class lOArray
75	{
76	public static void print(double[,] a, int n, int m, string fmt)
77	{
78	int ij;
79	for(i=0; i<n; i++, Console.WriteLineQ)
80	for(j=0;j<m; j++)
81	Console. Write(fmt,a[iJ]);
82	}
83	
84	public static void rand(double[,] a, int n, int m, double min, double max)
85	{
86	Random ran = new RandomQ;
87	int i j;

88	for(i=0; i<n; i++)
89	for(j=0; j<m; j++)
90	a[i j] = min + (max-min)*ran.NextDoubleQ;
91	}
92	}

Оценка характеристик программы

Определим основные характеристики для расчета метрики Чепина. На основе анализа исходного текста составим таблицу 30.

Таблица 30 - Результат анализа объявленных переменных

№ n/n	Наименования переменных	Номера строк
P (для расчетов и обеспечения вывода)		
1	sinp	51
2	n	56
3	m	59
4	min	48
5	max	48
M (модифицируемые или создаваемые)		
1	a	46
2	s	47
3	i	6
4	j	6
5	buf	13
6	row1	11
7	row2	11
C (управляющие переменные)		
1	ran	75
2	flag	25
3	rep	50
4	fmt	65
T (не используемые в программе)		
Отсутствуют		

Исходя из полученных на основе анализа данных имеем: $P = 5$, $M=1$, $C = 2$, $T = 0$. $Q = 5 + 14 + 12 + 0 = 31$.

Выводы. Задача 1 имеет одномодульное решение, в то время как задача 2 имеет многомодульное построение решения, что повышает уровень ее сложности. Этот фактор оказывает влияние и на уровень сложности решения. Использование многомодульного формирования алгоритма решения в значительной степени усложняет понимание программы из-за дополнительного ввода переменных для расчета,

модифицируемых переменных й переменных управления вычислительным процессом.

Задача 3 «Заправка топливных баков»

Определить класс «Бак», описывающий понятие «Топливный бак». При этом должны быть использованы следующие поля:

- ширина, длина и высота в сантиметрах;
- вид топлива.

Следует учитывать операции (методы): .

- полная заправка бака заданным видом топлива;
- вычисление стоимости полной заправки бака.

Бак в общем случае имеет вид параллелепипеда. Стандартным считается бак, имеющий вид куба (ширина, длина и высота совпадают).

Возможные разновидности баков:

- бак с одинаковой шириной и длиной, но с индивидуальной высотой;
- бак с индивидуальными размерами по ширине, длине и высоте.

Стоимость одного кубического сантиметра топлива определяется полями класса «Топливо». Стоимость топлива в рамках решения задачи неизменна. Выдача стоимости топлива реализуется специальной операцией этого класса.

Заполнить и вывести на экран стоимость заправки четырех баков:

- 10x20x30 см: топливо–газ;
- 10x10x20 см: топливо–керосин;
- 10x10x10 см: топливо–бензин;
- 20x20x20 см: топливо–бензин

Реализация программы

Текст программы для реализации возможного алгоритма решения

поставленной задачи представлен на языке программирования C# (Таблица 31).

Таблица 31 – Реализация программы для задачи «Заправка топливных баков»

Номера строк	Строки программы
1	using System;
2	namespace EX2_1
3	{
4	class Топливо
5	{
6	private static double ценаГаз = 0.05
7	private static double ценаГаз = 0.05
8	private static double ценаБензии = 0.1;;
9	public static double L[eHa(string вид)
10	{
11	switch (вид)
12	{
13	case "газ": return ценаГаз;
14	case "керосин": return ценаКеросин;
15	case "бензин": return ценаБензии;
16	default: return 0.0
17	}
18	}
19	}
20	class Бак
21	{
22	private double x, y, h;
23	private string видТоплива;
24	public void Заполнить(double xb, string вид)
25	{
26	x = xb; y = xb; h = xb; видТоплива = вид;
27	{
28	public void Заполнить(ref double xb, string вид)
29	{
30	x = xb; y = xb; h = xb; видТоплива = вид;
31	}
32	public void Заполнить(double xb, double yb, string вид)
33	}
34	x = xb; y = yb; h = xb; видТоплива = вид;
35	}
36	public void Заполнить(double xb. double vb. double hb. string вид)
37	{
38	x = xb; y = yb; h = hb; видТоплива = вид;
39	}
40	public double Оплата()

41	{
42	return x * y * h * Топливо.Цена(видТоплива);
43	}
44	}
45	class Program
46	{
47	static void Main(string[] args)
48	{
49	double x = 20.0;
50	Бак b;
51	b = new Бак();
52	b.Заполнить(10.0,20.0,30.0, "газ");
53	Console.WriteLine("Стоимость заправки: " + b.Оплата());
54	b.Заполнить(10.0, 20.0,"керосин");
55	Console.WriteLine("Стоимость заправки: " + b.Оплата());
56	b.Заполнить(10.0,"бензин");
57	Console.WriteLine("Стоимость заправки:" + b.Оплата());
58	b.Заполнить(ref x, "бензин");
59	Console.WriteLine("Стоимость заправки:" + b.Оплата());
60	Console.ReadK.ey();
61	}
62	}
63	}

Оценка характеристик программы

Необходимо отметить, что в исходном тексте программы используются одноименные программные модули и имена переменных входных параметров этих модулей (таблица 31, строки 24, 28, 32 и 36). Несмотря на совпадение имен, эти переменные являются различными элементами программы, так как принадлежат различным программным модулям.

На основе анализа исходного текста составим таблицу 32. Для одноименных переменных введем дополнительный столбец.

Таблица 32. – Результат анализа объявленных переменных

№ п/п	Наименования переменных	Номера строк	Количество переменных
P (для расчетов и для обеспечения вывода)			
1	xb	24,28, 32, 36	4
2	yb	32, 36,	2
3	hb	36	1
M (модифицируемые или создаваемые)			
1	ценаГаз	6	1

2	ценаКеросин	7	1
3	ценаБензии	8	1
4	x	22, 50	2
5	y	22	1
6	h	22	1
7	видТоплива	23	1
8	b	51	1
C (управляющие переменные)			
1	вид	9	1
T (не используемые в программе)			
Отсутствуют			

Исходя из полученных на основе анализа данных имеем: $P = 7$, $M = 9$, $C = 1$, $T = 0$. $Q = 7 + 2 \cdot 9 + 3 \cdot 1 + 0{,}5 \cdot 0 = 28$.

Выводы. Уровень сложности задачи является достаточно высоким, так как реализация задачи имеет многомодульную схему, и, как следствие, используется большое количество переменных для расчета ($P = 7$) и модифицируемых переменных ($M = 9$).

3.3 Задания для аудиторной работы

Задача 1. «Формирование вещественной матрицы»

Определить класс с методами заполнения и вывода вещественных массивов. Используя эти методы, сформировать вещественную матрицу размером $N \times M$ элементов. Размер матрицы вводится с клавиатуры (N не больше 10). Отсортировать каждую строку матрицы по возрастанию элементов строки. Матрица заполняется случайными числами в диапазоне от 1 до 5. Разработать программу для решения задачи. На основе лексического анализа исходного текста программы определить значение метрики Чепина.

Реализация программы

Текст программы для реализации возможного алгоритма решения поставленной задачи представлен на языке программирования *C#* (таблица 33).

Таблица 33 – Реализация программы для задачи «Формирование

вещественной матрицы»

Номера строк	Строки программы
1	using System;
2	class MyMetod
3	{
4	public static void Main()
5	{
6	double[][] a;
7	double min = 1.0, max = 5.0;
8	int n=0, m=0, i;
9	char rep;
10	string sinp;
11	do
12	{
13	try
14	{
15	Console.Write("Строк: ");
16	sinp = Console.ReadLine();
17	n = int.Parse(sinp);
18	Console.Write("Столбцов: ");
19	sinp = Console.ReadLineO;
20	m = int.Parse(sinp);
21	a = new double[n][];
22	for(i=0; i<n; i++)
23	a[i] = new double[m]
24	IOArray.rand(a,min,max);
25	IOArray.print(a,"{0,8:f2}");
26	Console.WriteLine();
27	for(i=0; i<=n; i++)
28	Array.Sort(a[i]);
29	IOArray.print(a,"{0,8:f2}");
30	}
31	catch (IndexOutOfRangeException) {Console.WriteLine("Переполнение массива!!!");}
32	Console.\Мгйе("\пДля повтора нажмите клавишу Y: ");
33	rep = char.Parse(Console.ReadLine());
34	Console.WriteLine();
35	}while(rep = = 'У || rep = = 'y');
36	}
37	}
38	
39	class IOArray
40	{
41	public static void rand(int[][] a, int min, int max)
42	{
43	Random ran = new Random();
44	intij;
45	for(i=0; i<a.Length; i++)

46	for(j=0; j<a[i]. Length; j++)
47	a[i][j] = ran.Next(min,max+l);
48	}
49	public static void print(double[][] a, string fmt)
50	{
51	int ij;
52	for(i=0; i<a.Length; i++, Console.WriteLine())
53	for(j=0; j<a[i].Length; j++)
54	Console.Write(fmt,a[i][j]);
55	}
56	}

Задача 2. «Расчет платежей за электроэнергию»

Предметная область данной задачи - коммунальные платежи за электроснабжение. Определить класс, описывающий операцию «Платеж за электроэнергию» с использованием следующих полей:

- фамилия плательщика;
- потребление электроэнергии за оплачиваемый месяц;
- нормативное среднемесячное потребление;
- тариф (стоимость одного киловатт-часа).

При решении задачи следует учитывать методы:

- вычисление суммы оплаты;
- формирование сводной информации по одному платежу (вид платежа, фамилия, сумма, потребление);

Платеж может выполняться по показаниям счетчика или по нормативно установленному среднемесячному потреблению. В процессе эксплуатации программы тариф и нормативно установленное среднемесячное потребление не изменяются.

Для создания конкретного платежа необходимо предусмотреть соответствующий конструктор. Все платежи сохраняются в архиве. Запросы по ведению архива выполняется статическими методами класса «Запрос»: занесение платежей в архив.

Данные платежа вводятся с клавиатуры. Ввод отрицательного

показания счетчика означает оплату по нормативно установленному среднемесячному потреблению. Вывод сводной информации проводится из архива. Архив моделируется массивом объектов. В основном классе «Платежи» следует сформировать архив платежей. По данным архива выдать сводную информацию о платежах.

Реализация программы

Текст программы для реализации возможного алгоритма решения поставленной задачи представлен на языке программирования C# (таблица 34).

Таблица 34 – Реализация программы для задачи «Расчет платежей за электроэнергию»

Номера строк	Строки программы
1	using System;
2	namespace EXI
3	{
4	class Платежи
5	{
6	static void Main()
7	{
8	ConsoleKeylnfo rep;
9	Электро[] плэ;
10	int кпэ;
11	do
12	{
13	Console.Clear();
14	Console.Write("Количество платежей за электроэнергию: ");
15	кпэ = int.Parse(Console.ReadLine());
16	плэ = new Электро[кпэ];
17	Запрос.Заполнить(плэ);
18	Запрос.Вывести(плэ);
19	Console.WriteLine("Для выхода нажмите ESC");
20	rep = Console.ReadKey(true);
21	}while(rep.Key!= ConsoleKey.Escape);
22	}
23	}
24	class Запрос
25	{
26	public static void Заполнить(Электро[] платеж)
27	{

28	string фам;
29	int счПред=0, счТек=0;
30	Console.ClearO;
31	for (int i = 0; i < платеж-Length; i++)
32	{
33	Console.Write("Платеж "+ i + Фамилия -> ");
34	фам = Console.ReadLineO;
35	Console.\Мп1е("Платеж" + i +Текущее значение счетчика ->");
36	счТек = int.Parse(Console.ReadLine());
37	if (счТек >0)
38	{
39	Console.Write("Платсж" + i +Предыдущее значение счетчика ->");
40	счПред = int.Parse(Console.ReadLine());
41	}
42	if(счТек <= 0)
43	платеж[i] = new Электро(фам);
44	else
45	платеж[i]=new Электро(фам, счТек, счПред);
46	}
47	}
48	public static void Вывести(Электро[] платеж)
49	{
50	for (int i = 0; i < платеж.Length; i++)
51	Console. АУп1е1лпе(платеж[i].Инфо());
52	}
53	}
54	class Электро
55	{
56	
57	private static double тариф = 1.84;
58	private static int потреблениеСреднее = 300;
59	private string фамилия;
60	private int потребление;
61	public Электро(string фамилия)
62	{
63	this.фамилия - фамилия;
64	потребление = потреблениеСреднее;
65	}
66	public Электро(string фамилия, int текущее, int предыдущее)
67	{
68	this.фамилия = фамилия;
69	потребление = текущее - предыдущее;
70	}
71	public double Сумма() { return потребление * тариф;}
72	public string Инфо()
73	{
74	return string.Format(" {0,-20} {1,-20} {2,10:12} {3,10:d6}",
75	"Электроэнергия",фамилия,Сумма(),потребление);
76	}

77	}
78	}

3.4. Задания для самостоятельного решения

В задачах, предлагаемых для самостоятельного решения, необходимо выполнить следующее:

• разработать программу, реализующую заданный алгоритм (рекомендуется использовать язык программирования C#);

• сформировать таблицы по образцу, приведенному в рассмотренных задачах;

• на основе лексического анализа исходного текста программы определить значение метрики Чепина;

• провести анализ полученных результатов, сформировав содержательные выводы.

Задача 1. В целочисленной матрице *A* размером *N*×*M* заменить элементы главной диагонали на номера столбцов (числа *N* и *M* задаются с клавиатуры в диапазоне от 3 до 10). Первоначальное заполнение матрицы осуществить при помощи датчика случайных чисел в диапазоне от –10 до 10. Исходную и видоизмененную матрицы вывести на экран.

Задача 2. В целочисленной матрице *A* размером *N*×*M* заменить элементы главной диагонали на значения элементов одномерного массива *B* (числа *N и M* задаются с клавиатуры в диапазоне от 3 до 10).

Первоначальное заполнение массивов *A* и *B* осуществить при помощи датчика случайных чисел в диапазоне от –10 до 10. Исходную и видоизмененную матрицы, а также массив *B* вывести на экран.

Задача 3. В целочисленной матрице *A* размером *N*×*M* заменить элементы столбца, расположенного по центру, на значения элементов одномерного массива *B* (числа *N и M* задаются с клавиатуры в

диапазоне от 3 до 10). Первоначальное заполнение массивов *A* и *B* осуществить при помощи датчика случайных чисел в диапазоне от –5 до 15. Исходную и видоизмененную матрицы, а также массив B вывести на экран.

Зада*ча 4.* В целочисленной матрице *A* размером *N*×*M* заменить элементы нечетных строк на значения элементов одномерного массива *B* (числа *N* и *M* задаются с клавиатуры в диапазоне от 3 до 10). Первоначальное заполнение массивов *A* и *B* осуществить при помощи датчика случайных чисел в диапазоне от –5 до 15. Исходную и видоизмененную матрицы, а также массив *B* вывести на экран.

Задача 5. В целочисленной матрице *A* размером *N*×*M* заменить элементы нечетных строк на значения элементов одномерного массива *B* (числа *N* и *M* задаются с клавиатуры в диапазоне от 3 до 10). Первоначальное заполнение массивов *A* и *B* осуществить при помощи датчика случайных чисел в диапазоне от –20 до 20. Исходную и видоизмененную матрицы, а также массив *B* вывести на экран.

Задача 6. В целочисленной матрице *A* размером *N*×*M* заменить элементы главной диагонали и расположенные выше нее на значение 2 (числа *N* и *M* задаются с клавиатуры в диапазоне от 3 до 10). Первоначальное заполнение массива *A* осуществить при помощи датчика случайных чисел в диапазоне от –5 до 15. Исходную и видоизмененную матрицы вывести на экран.

Задача 7. В целочисленной матрице *A* размером *N*×*M* заменить элементы главной диагонали и расположенные ниже нее на значение 3 (числа *N* и *M* задаются с клавиатуры в диапазоне от 3 до 10). Первоначальное заполнение массива *A* осуществить при помощи датчика случайных чисел в диапазоне от –5 до 15. Исходную и

видоизмененную матрицы вывести на экран.

Задача 8. В целочисленной матрице *A* размером *N*×*M* заменить элементы побочной диагонали и расположенные выше нее на значение 4 (числа *N* и *M* задаются с клавиатуры в диапазоне от 3 до 10). Первоначальное заполнение массива *A* осуществить при помощи датчика случайных чисел в диапазоне от –5 до 15. Исходную и видоизмененную матрицы вывести на экран.

Задача 9. В целочисленной матрице *A* размером *N*×*M* заменить элементы побочной диагонали и расположенные ниже нее на значение 4 (числа *N* и *M* задаются с клавиатуры в диапазоне от 3 до 10). Первоначальное заполнение массива *A* осуществить при помощи датчика случайных чисел в диапазоне от –5 до 15. Исходную и видоизмененную матрицы вывести на экран.

Задача 10. В целочисленной матрице *A* размером *N*×*M* подсчитать сумму элементов главной диагонали и расположенных выше (числа *N и M* задаются с клавиатуры в диапазоне от 3 до 10). Заполнение массива *A* осуществить при помощи датчика случайных чисел в диапазоне от –5 до 5. Исходную матрицу и сумму элементов заданной области матрицы вывести на экран.

Задача 11. В целочисленной матрице *A* размером *N*×*M* подсчитать сумму элементов главной диагонали и расположенных ниже нее (числа *N* и *M* задаются с клавиатуры в диапазоне от 3 до 10). Заполнение массива *A* осуществить при помощи датчика случайных чисел в диапазоне от -3 до 3. Исходную матрицу и сумму элементов заданной области матрицы вывести на экран.

Задача 12. В целочисленной матрице *A* размером *N*×*M* подсчитать сумму элементов побочной диагонали и расположенных выше нее

(числа *N* и *M* задаются с клавиатуры в диапазоне от 3 до 10). Заполнение массива *A* осуществить при помощи датчика случайных чисел в диапазоне от 1 до 10. Исходную матрицу и сумму элементов заданной области матрицы вывести на экран.

Задача 13. В целочисленной матрице *A* размером *N*×*M* подсчитать сумму элементов побочной диагонали и расположенных ниже нее (числа *N* и *M* задаются с клавиатуры в диапазоне от 3 до 10). Заполнение массива *A* осуществить при помощи датчика случайных чисел в диапазоне от 2 до 12. Исходную матрицу и сумму элементов заданной области матрицы вывести на экран.

Задача 14. Сформировать вещественную матрицу *A* размером *N*×*M* (числа *N* и *M* задаются с клавиатуры в диапазоне от 3 до 10). Суммы элементов строк вывести на экран с указанием номера строки. Первоначальное заполнение матрицы осуществить при помощи датчика случайных чисел в диапазоне от -2,2 до 10,2, матрицу вывести на экран.

Задача 15. Сформировать вещественную матрицу *A* размером *N*×*M* (числа *N* и *M* задаются с клавиатуры в диапазоне от 3 до 10). Суммы элементов столбцов вывести на экран с указанием номера столбца. Первоначальное заполнение матрицы осуществить при помощи датчика случайных чисел в диапазоне от -5,2 до 5,2, матрицу вывести на экран.

Задача 16. Сформировать вещественную матрицу *A* размером *N*×*M* (числа *N* и *M* задаются с клавиатуры в диапазоне от 3 до 10). Суммы элементов строк занести в одномерный массив B. Первоначальное заполнение матрицы осуществить при помощи датчика случайных чисел в диапазоне от -3,3 до 3,3. Массивы *A* и *B* вывести на экран.

Задача 17. Сформировать вещественную матрицу *A* размером *N*×*M* (числа *N* и *M* задаются с клавиатуры в диапазоне от 3 до 10). Суммы элементов столбцов занести в одномерный массив *B*. Первоначальное заполнение матрицы осуществить при помощи датчика случайных чисел в диапазоне от -5,5 до 5,5. Массивы *A* и *B* вывести на экран.

Задач*а 18.* Сформировать вещественную матрицу *A* размером *N*×*M* (числа *N* и *M* задаются с клавиатуры в диапазоне от 3 до 10). Суммы элементов нечетных строк занести в одномерный массив *B*. Первоначальное заполнение матрицы осуществить при помощи датчика случайных чисел в диапазоне от -10,1 до 3,1. Массивы *A* и *B* вывести на экран.

Задача 19. Сформировать вещественную матрицу *A* размером *N*×*M* (числа *N* и *M* задаются с клавиатуры в диапазоне от 3 до 10). Суммы элементов нечетных столбцов занести в одномерный массив *B*. Первоначальное заполнение матрицы осуществить при помощи датчика случайных чисел в диапазоне от -2,5 до 2,5. Массивы *A* и *B* вывести на экран.

Задача 20. Создать двухмерный массив размером *N*×*M* и заполнить случайным образом нулями и единицами. Проверить, есть ли строгое чередование нулей и единиц в каждой строке массива. Исходный массив и номера строк, для которых выполняется условие чередования, вывести на экран. Значения *N* и *M* вводятся с клавиатуры.

Задача 21. Создать двухмерный массив размером *N*×*M* и заполнить случайным образом нулями и единицами. Проверить, есть ли строгое чередование нулей и единиц в каждом столбце массива. Исходный массив и номера столбцов, для которых выполняется условие

чередования, вывести на экран. Значения *N* и *M* вводятся с клавиатуры.

Задача 22. Сформировать целочисленный двухмерный массив размером *N* строк. Строки имеют разное количество элементов. Правило формирования длины строки и значений элементов строки показано ниже. Полученный массив построчно вывести на экран. Значение *N* вводится с клавиатуры.

1	2					
1	2	3	4			
…	…	…	…	…	…	
1	2	3	4	…	2*N*-1	2*N*

Задача 23. Сформировать целочисленный двухмерный массив размером *N* строк. Строки имеют разное количество элементов. Правило формирования длины строки и значений элементов строки показано ниже. Полученный массив построчно вывести на экран. Значение *N* вводится с клавиатуры.

1	2	3	4	5	6	⋯	2*N*-1	2*N*
1	2	3	4	⋯	2*N*-3	2*N*-2		
…	…	…	…	…				
1	2	3	4					
1	2							

Задача 24. Сформировать целочисленный двухмерный массив размером *N* строк. Строки имеют разное количество элементов. Правило формирования длины строки и значений элементов строки показано на рисунке. Полученный массив построчно вывести на экран. Значение *N* вводится с клавиатуры.

1				
2	1			
3	2	1		
…	…	…	1	
N	*N*-1	*N*-2	…	1

Задача 25. Сформировать целочисленный двухмерный массив

размером N строк. Строки имеют разное количество элементов. Правило формирования длины строки и значений элементов строки показано на рисунке. Полученный массив построчно вывести на экран. Значение N вводится с клавиатуры.

N	N-1	N-2	…	1
N-1	N-2	…	1	
…	…	…		
2	1			
1				

4 РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

4.1 Основная литература

1. Черников, Б. В. Управление качеством программного обеспечения: Учебник / Б.В. Черников. - Москва : ИД ФОРУМ: ИНФРА-М, 2012. - 240 с.: ил.; . - (Высшее образование). ISBN 978-5-8199-0499-2. - Текст : электронный. - URL: https://znanium.com/catalog/product/256901. – Режим доступа: по подписке.

2. Черников, Б. В. Оценка качества программного обеспечения: Практикум: Учебное пособие / Б.В. Черников, Б.Е. Поклонов; Под ред. Б.В. Черникова - Москва : ИД ФОРУМ: НИЦ Инфра-М, 2012. - 400 с.: ил.; . - (Высшее образование). ISBN 978-5-8199-0516-6. - Текст : электронный. - URL: https://znanium.com/catalog/product/31526. – Режим доступа: по подписке.

4.2 Дополнительная литература

1. Кайгородцев, Г. И. Введение в курс метрической теории и метрологии программ/КайгородцевГ.И. - Новосибирск : НГТУ, 2016. - 192 с.: ISBN 978-5-7782-1648-8. - Текст : электронный. - URL: https://znanium.com/catalog/product/549419. – Режим доступа: по

подписке.

2. Ананьева, Т. Н. Стандартизация, сертификация и управление качеством программного обеспечения : учебное пособие / Т.Н. Ананьева, Н.Г. Новикова, Г.Н. Исаев. — Москва : ИНФРА-М, 2020. — 232 с. — - ISBN 978-5-16-014887-8. - Текст : электронный. - URL: https://znanium.com/catalog/product/1062373. – Режим доступа: по подписке.