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.

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				