МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Брянский государственный технический университет

Утверждаю
Ректор университета
О.Н. Федонин
«14» 07 2017г.

МЕТРОЛОГИЯ И КАЧЕСТВО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

ОЦЕНКА ХАРАКТЕРИСТИК ПРОГРАММ НА ОСНОВЕ ЛЕКСИЧЕСКОГО АНАЛИЗА. МЕТРИКИ ДЖИЛБА

Методические указания
к выполнению лабораторной работы №2
для студентов очной формы обучения
по направлениям подготовки
09.03.01 «Информатика и вычислительная техника»
09.03.04 «Программная инженерия»
02.03.03«Математическое обеспечение и администрирование информационных систем»

Брянск 2017

УДК 004.01

Метрология и качество программного обеспечения. Оценка характеристик программ на основе лексического анализа. Метрики Джилба. [Электронный ресурс]: методические указания к выполнению лабораторной работы №2 для студентов очной формы обучения по направлению подготовки 09.03.01 «Информатика и вычислительная техника»; 09.03.04 «Программная инженерия»; 02.03.03 «Математическое обеспечение и администрирование информационных систем» – Брянск, 2017. - 26с.

Разработал
А.А. Азарченков,
канд. техн. наук, доц.

Рекомендовано кафедрой «Информатика и программное обеспечение» БГТУ (протокол № 7 от 01.06.2017г.)

Методические издания публикуются в авторской редакции

1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Достаточно практичными являются метрики программного обеспечения, предложенные Томасом Джилбом и также основанные на результатах анализа текстов программных продуктов.

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.

Абсолютная логическая сложность, по мнению автора метрик, должна задаваться числом необычных выходов из операторов, в которых происходит принятие решений. Джилб предположил, что логическая сложность должна являться значимым, если не определяющим, фактором для оценки стоимости программы на начальных этапах ее проектирования.

Логическую сложность программы Джилб определяет как насыщенность программы условными операторами типа IF-THEN-ELSE и операторами цикла (при этом следует учитывать, что фактическая запись условий и циклов в разных языках программирования может быть представлена в разной форме при сохранении указанного смысла операторов). При этом вводятся следующие характеристики программного средства:

CL – абсолютная сложность программы, характеризуемая количеством операторов условий;

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

Кроме указанных показателей Джилб предложил применять следующие характеристики программы:

- количество операторов цикла L_{loop};
- количество операторов условия L_{IF};
- число модулей или подсистем L_{mod};
- отношение числа связей между модулями к числу модулей

$$f = \frac{N_{SV}^4}{L_{mod}}$$

- отношение числа ненормальных выходов из множества операторов к общему числу операторов

$$f^* = \frac{N_{SV}^*}{L}$$

Среди всех показателей качества программ Джилб указывает надежность программы, которую он характеризует как возможность того, что данная программа проработает определенный период времени без логических сбоев. В качестве практической оценки программной надежности автор метрик предлагает рассчитывать, как единицу минус отношение числа логических сбоев к общему числу запусков.

Отношение количества правильных данных ко всей совокупности данных приводится Джилбом в качестве меры точности (свободы от ошибок), поскольку он считает, что точность нужна как средство обеспечения надежности программы. Прецизионность определяется как мера того, насколько часто появляются ошибки, вызванные одинаковыми причинами. Джилб оценивает этот показатель дробью, в числителе которой указывается число фактических ошибок на входе, а в знаменателе – общее число наблюдаемых ошибок, причинами которых явились эти ошибки на входе. Так, например, если одна ошибка вызывает в течение определенного периода времени появление 50 сообщений об ошибках, то прецизионность равна 0,02.

2. ПРИМЕР ОЦЕНКИ ХАРАКТЕРИСТИК

2.1. Задача «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.

2.1.1. Реализация программы

Текст программы для реализации возможного алгоритма решения поставленной задачи, разработанный с использованием языка программирования C#, приведен на рис. 1.

Номера строк	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
32	клавишу Y: ");
33	rep = char.Parse(Console.ReadLine());
34	if (rep = = 'Y' || rep = = 'y') goto REPEAT;
35	}
36	}

Рис. 2.1. Текст для программы

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, тем более что такой оператор не влияет на параметры оценки ее качества с использованием метрик Джилба.

Программа, разработанная для реализации заданного алгоритма, не имеет циклов и модулей. Следовательно, из перечисленного набора характеристик можно определить лишь следующие: L, L_{IF}, CL и cl, причем для такого случая $L_{IF} = CL$.

2.1.2. Словарь программы

При подсчете общего количества операторов L программы будем руководствоваться правилами, определенными при подсчете операторов в метриках Холстеда.

В табл. 1 приведены операторы и операции, используемые в программе. При подсчете числа операторов следует иметь в виду, что в строке 28 (см. рис. 1) скобки используются как символы строковых констант, а потому операторами не являются.

Таблица 1.

Словарь операторов и операций программы

№ п/п	Операторы, операции	Номера строк	Количество повторений
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.WriteLine()	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**

2.1.3. Оценка характеристик программы

Согласно теории Джилба необходимо определить количество операторов условия, используемых в исходном тексте программы для реализации алгоритма решения поставленной задачи.

В языке программирования C# к такой категории инструкций могут относиться условные операторы ветвления if операторы циклов $for \ldots while$ и $do \ldots while$, которые в своем составе в обязательном порядке содержат логическое выражение, являющееся условием выполнения указанных операторов. В зависимости от соблюдения условия программа может существенно изменить ход выполнения, поэтому значимость условных операторов достаточно велика. Именно поэтому автор для них определил отдельную метрику.

В исходном тексте разработанной программы используются условные операторы if, которые располагаются в строках с номерами 17, 18, 23 и 33 (рис. 1). Исходя из полученных данных (табл. 1) получаем следующие результаты оценки характеристик программы на основе метрик Джилба:

- для представленного варианта решения число операторов условий L_{if} равно 4;
- 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.

2.2. Задача «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 С оказывается пустым. Для разработанной программы на основе лексического анализа исходного текста определить значения метрик Джилба.

2.2.1. 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 на рис. 2.

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.Length; i++)
18	{
19	a[i] = g.Next(-100, 300);
20	}
21	}
22	
23	public static void Print(int[] a, string str, string str1)
24	{
25	Console.WriteLine(str1);
26	for (int i = 0; i < a.Length; 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.Length];
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.Parse(Console.ReadLine());
58    } while (r = = 'Y' || r = = 'y');
59    }
60  }
```

Рис. 2.2. Текст программы копирования элементов массива

В 6-й, 42-й, 44-й и 50-й строках (рис. 2) 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.

2.2.2. Словарь программы

В табл. 2.1 приведены операторы и операции, используемые в программе.

Таблица 2.1

Словарь 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

№ п/п	Операторы, операции	Номера строк	Количество повторений
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
14	Console.ReadLine()	42, 44, 57	3
15	for()	8, 17, 26	3
16	d0{}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**

2.2.3. Оценка характеристик программы

Определим значения характеристик $L_{IF}, L_{loop}, L_{mod}, f, L, CL, cl$. Значение характеристики L_{IF} определяется количеством используемых в программе операторов if. В представленном решении их два. Значение характеристики L_{loop} определяется количеством используемых в программе циклов. В исходном тексте данной программы содержится четыре цикла: три оператора for и один $do…whil$. Значение характеристики L_{mod} определяется количеством используемых программных модулей в решении. В представленном решении используется четыре программных модуля, каждый из которых определяется следующими строками:

- $public\ static\ voidMain()$;
- $public\ static\ int[]\ Copy(int[]$ a);
- $public\ static\ void\ Zapoln(ref\ int[]\ a, Random\ g)$;
- $public\ static\ void\ Print(int[]\ a, string\ str, string\ strl)$.

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

- $CL = 6$ – абсолютная сложность программы;
- $cl = CL/L$ = 6/263 =0,0228.

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

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

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.

2.3. Задача «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.

2.3.1. Реализация программы

Текст программы для реализации возможного алгоритма решения поставленной задачи представлен на языке C#. Решение представлено в виде одномодульной программы (рис. 3).

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; j++)
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.WriteLine())
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.WriteLine())
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-1; j++)
55    {
56    if (c[i][j] > c[i][j + 1])
```

```
57      {
58      buf = c[i][j];
59      c[i][j] = c[i][j + 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  }
```

Рис. 2.3. Текст программы для сложения элементов матриц

2.3.2. Словарь программы

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

Таблица 3.

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	1
4	int [][]	6	1
5	int	7	1
6	new	10, 20, 21, 22, 25, 26, 27	5
7	char	8	1
8	Random	10	1
9	bool	9	1
10	. Length	29, 32, 33, 35, 36, 39, 40, 43, 44, 47, 48, 54, 55, 67, 68	15
11	.Next	31, 34	2
12	Console.WriteLine()	14, 18, 38, 42, 43, 46, 47, 66, 67, 70	10
13	Console.Write()	41, 45, 49, 69	4
14	Console.ReadLine()	17, 19, 71	3
15	for()	23, 29, 30, 32, 33, 35, 36, 39, 40, 43, 44, 47, 48, 54, 55, 67, 68	17

№ п/п	Операторы, операции	Номера строк	Количество повторений
16	d0{}while()	11–72	1
17	if()	57	1
18	while() {}	51–65	1
19	Console.Clear()	13	1
20	.Parse()	17, 19, 71	3
21	=	7, 10, 17, 19, 20, 21, 22, 23, 25, 26, 27, 29, 30, 31, 32, 33, 34, 35, 36, 37, 39, 40, 43, 44, 47, 48, 50, 53, 54, 55, 59, 60, 61, 62, 67, 68, 71	37
22	>	57	1
23	==	72, 72	2
24	<	23, 29, 30, 32, 33, 35, 36, 39, 40, 43, 44, 47, 48, 54, 55, 67, 68	17
25	++	23, 29, 30, 32, 33, 35, 36, 39, 40, 43, 44, 47, 48, 54, 55, 67, 68	17
26	[]	6, 6, 20, 20, 21, 21, 22, 22, 25, 25, 26, 26, 27, 27, 30, 31, 31, 33, 34, 34, 36, 37, 37, 37, 37, 37, 37, 40, 41, 41, 44, 45, 45, 48, 49, 49, 55, 57, 57, 57, 57, 59, 59, 60, 60, 60, 60, 61, 61, 68, 69, 69	52
27	()	4, 10, 13, 14, 16, 17, 17, 18, 19, 19, 23, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 39, 40, 41, 42, 43, 43, 44, 45, 46, 47, 47, 48, 49, 51, 54, 55, 57, 66, 67, 67, 68, 69, 70, 71, 71, 72	48
28	{}	3(74), 5(73), 12(72), 24(28), 52(65), 56(64), 58(63)	7
29	,	6, 6, 7, 7, 7, 7, 31, 34, 39, 41, 41, 43, 45, 45, 47, 49, 49, 67, 69, 69	20
30	;	1, 6, 7, 8, 9, 10, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23, 23, 25, 26, 27, 29, 29, 30, 30, 31, 32, 32, 33, 33, 34, 35, 35, 36, 36, 37, 38, 39, 39, 40, 40, 41, 42, 43, 43, 44, 44, 45, 46, 47, 47, 48, 48, 49, 50, 53, 54, 54, 55, 55, 59, 60, 61, 62, 66, 67, 67, 68, 68, 69, 70, 71, 72	72
31	.	13, 14, 16, 17, 18, 19, 29, 30, 31, 32, 33, 34, 35, 36, 38, 39, 39, 40, 41, 42, 43, 43, 44, 45, 46, 47, 47, 48, 49, 54, 55, 66, 67, 67, 68, 69, 70, 71, 71	39

№ п/п	Операторы, операции	Номера строк	Количество повторений
32	int[…]	20, 21, 22, 25, 26, 27	6
33	Random()	10	1
34	“ ”	14, 16, 18, 38, 41, 42, 45, 46, 49, 66, 69, 70	12
35	‘ ’	72, 72	2
		Всего	**404**

2.3.3. Оценка характеристик программы

Определим значения характеристик $L_{IF}, L_{loop}, L_{mod}, f, L, CL, cl$. Значение характеристики L_{IF} определяется количеством используемых в программе операторов if В представленном решении он один. Значение характеристики L_{loop} определяется количеством используемых в программе циклов. В исходном тексте данной программы содержится девятнадцать циклов: семнадцать операторов for, один оператор $while()$ и один $do…while$. Таким образом, общее количество операторов условия 20, из них один оператор if девятнадцать операторов цикла. Общее число всех используемых операторов $L = 404$.

Тогда сложность программы определится следующим образом:

- $CL = 20$ – абсолютная сложность программы;
- $cl = CL / L = 20 / 404 = 0{,}049$ – относительная сложность программы.

Значения полученных характеристик свидетельствуют, что сложность программы можно отнести к среднему уровню, так как в исходном ее коде достаточно большое количество циклов (19), следовательно, присутствует и большое количество операторов условий, которые усложняют программу.

2.4. Задача «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.

2.4.1. Реализация программы

Текст программы для реализации возможного алгоритма решения поставленной задачи представлен на языке C#. Решение представлено в виде одномодульной программы (рис. 4)

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 (int j = 0; j < a[i].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,"\nВторая 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  }
```

Рис. 2.4. Текст программы для сложения элементов матриц

2.4.2. Словарь программы

В табл. 4 приведены операторы и операции, используемые в программе.

Таблица 4.

Словарь операторов и операций программы

№ п/п	Операторы, операции	Номера строк	Количество повторений
1	using…	1	1
2	class…	2	1
3	public …	4, 13, 20, 27, 33	5
4	int [][]	4, 6, 13, 20, 27, 35	6
5	int	4, 4, 7, 15, 16, 32, 24, 29, 30, 36	10
6	new	6, 9, 37, 38	5
7	char	39	1
8	Random	38	1
9	. Length	15, 16, 23, 24, 29, 30, 54, 55	8
10	.Next	17	1
11	Console.WriteLine()	22, 23, 43, 45, 47	5
12	Console.Write()	25	1
13	Console.ReadLine()	46, 48, 63	3
14	for()	7, 15, 16, 23, 24, 29, 30, 54, 55	9
15	d0{}while()	40–64	1
16	Console.Clear()	42	1
17	.Parse()	46, 48, 63	3
18	=	6, 7, 9, 15, 16, 17, 23, 24, 29, 30, 37, 38, 46, 48, 49, 50, 51, 54, 55, 63	20
19	= =	64, 64	2
20	<	7, 15, 16, 23, 24, 29, 30, 54, 55	9
21	++	7, 15, 16, 23, 24, 29, 30, 54, 55	9
22	[]	4, 4, 6, 6, 6, 6, 9, 9, 13, 13, 16, 17, 17, 20, 20, 24, 25, 25, 27, 27, 30, 31, 35, 35, 55, 56, 56, 56, 56, 56, 56	31
23	()	4, 7, 13, 15, 16, 17, 20, 22, 23, 23, 24, 25, 27, 29, 30, 31, 33, 37, 38, 42, 43, 44, 45, 46, 46, 47, 48, 48, 49, 50, 51, 52, 53, 54, 55, 57, 58, 59, 60, 61, 62, 63, 63, 64	44
24	{}	3(66), 5(12), 8(10), 14(19), 21(26), 28(32), 34(65), 41(64)	8
25	,	4, 13, 17, 20, 23, 25, 35, 35, 36, 36, 36, 52, 53, 57, 58, 59, 61	17
26	;	1, 6, 7, 7, 9, 11, 17, 22, 23, 23, 24, 24, 25, 29, 29, 30, 30, 31, 35, 36, 37, 38, 39, 42, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 54, 55, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64	47
27	Array.Sort()	31	1

№ п/п	Операторы, операции	Номера строк	Количество повторений
28	e.Sozd()	49, 50, 51	3
29	e.Zpoln()	52, 53	2
30	e.Print()	57, 58, 59, 61	4
31	e.Sort()	60	1
32	.	15, 16, 17, 22, 23, 23, 24, 25, 29, 30, 31, 42, 43, 45, 46, 46, 47, 48, 48, 49, 50, 51, 52, 53, 54, 55, 57, 58, 59, 60, 62, 63, 63	33
33	int[…]	6, 9	2
34	Random()	38	1
35	“ ”	25, 43, 45, 47, 57, 58, 59, 61, 62	9
36	‘ ’	72, 72	2
		Всего	**308**

2.4.3. Оценка характеристик программы

Определим значения характеристик $L_{IF}, L_{loop}, L_{mod}, f, L, CL, cl$. Значение характеристики L_{IF} определяется количеством используемых в программе операторов if. В представленном решении их нет. Значение характеристики L_{loop} определяется количеством используемых в программе циклов. В исходном тексте данной программы содержится четыре цикла: девять операторов for и один $do…while$.

Таким образом, общее количество операторов условия 10, из них 10 операторов цикла. Общее число всех используемых операторов $L = 308$. Таким образом:

- $CL = 10$ – абсолютная сложность программы;
- $cl = CL / L = 10 / 308 = 0{,}032$ – относительная сложность программы.

Количество связей между N_{sv} модулями равно четырем - по одной связи между основным и каждым из дополнительных модулей.

Отношение числа связей к числу модулей определяется так:

$$f = \frac{N_{SV}^4}{L_{mod}} = \frac{4^4}{5} = 51{,}2$$

Для многомодульных решений увеличение количества программных модулей приводит к логическому усложнению: для 4 модулей $f = 20{,}25$, а для 5 модулей $f = 51{,}2$. При переходе от одномодульной схемы решения к многомодульной логическая

сложность исходного кода в значительной мере снижается. Для одномодульного решения значение характеристики абсолютной сложности $CL = 20$, для многомодульного варианта той же задачи $CL = 10$.

3. ЗАДАЧИ ДЛЯ САМОСТОЯТЕЛЬНОГО РЕШЕНИЯ

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 к вводятся с клавиатуры. Пример: для числа 456771 и $k = 2$ искомое число равно 45.

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

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

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:

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

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

Задача 4. Вывести на экран таблицу истинности логической функции трех переменных:

$$F = (a \vee \bar{b}) \wedge c.$$

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 $1, 3, 5, 7, \ldots$.Количество суммируемых членов ряда N задается с клавиатуры.

Задача 11. Вычислить сумму первых N членов ряда

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots.$$

Количество суммируемых элементов N задается с клавиатуры.

Задача 12. Вычислить сумму первых N членов ряда

$$1 + \frac{3}{2} + \frac{5}{3} + \frac{7}{4} + \cdots.$$

Количество суммируемых элементов N задается с клавиатуры.

Задача 13. Вычислить сумму первых N членов ряда

$$\frac{1}{2} + \frac{3}{4} + \frac{5}{6} + \cdots.$$

Количество суммируемых элементов N задается с клавиатуры.

Задача 14. Вычислить сумму первых N членов ряда

$$1 + \frac{8}{9} + \frac{15}{17} + \frac{22}{25} + \cdots.$$

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 $f(x)$:

$$f(x) = \begin{cases} \sin(\frac{\pi}{2} \text{ при } x \le 0{,}5; \\ \sin\left((x-1)\cdot\frac{\pi}{2}\right) \text{ при } x \ge 0{,}5. \end{cases}$$

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:

$$y_{i+1} = 0{,}5 \cdot \left(y_i + \frac{x}{y_i}\right);\ y_0 = x.$$

Итерации прекратить при выполнении условия

$$|y_{i+1} - y_i| < 2 \cdot 10^{-5}$$

Задача 20. Вычислить значения функции $Z(x, m)$:

$$Z(x, m) = x^m (\sin(x \cdot m))^m$$

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 -ю справа цифру натурального числа. Число и номер цифры N вводятся с клавиатуры.

Задача 22. Вычислил» значения функции $f(x)$:

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

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

Задача 23. Вычислить значения функции $f(x)$:

$$f(x) = \begin{cases} ln\left|\frac{x}{1+y}\right| \text{ при } x \geq y; \\ \frac{1+x}{1+y} \cdot e^{-|x+y|} \text{ при } x < y. \end{cases}$$

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

$$|y_{i+1} - y_i| < 10^{-5}$$

Задача 25. Вычислить значения функции

$$f(x) = sinx + sin^2(x^2) + sin^3(x^3)$$

для значений аргумента x от 0 до 1,2 с шагом 0,1.

СПИСОК РЕКОМЕНДУЕМО ЛИТЕРАТУРЫ

1. Черников Б.В. Управление качеством программного обесепчения:/ Б.В. Черников. – М.: ИД «ФОРУМ»: ИНФРА-М, 2012. – 240 с.:

Черников Б.В. Оценка качества программногго обеспечения: Практикум: учебное пособие / Б.В. Черненко. Б.Е. Поклонов/ Под ред. Б.В. Чернякова. – М.: ИД «Форум»: ИНФРА-М, 2012. – 400 с.

Метрология и качество программного обеспечения. Оценка характеристик программ на основе лексического анализа. Метрики Джилба: методические указания к выполнению лабораторной работы №2 для студентов очной формы обучения по направлению подготовки 09.03.01 «Информатика и вычислительная техника»; 09.03.04 «Программная инженерия»; 02.03.03 «Математическое обеспечение и администрирование информационных систем»

АЗАРЧЕНКОВ АНДРЕЙ АНАТОЛЬЕВИЧ

Научный редактор Д.А. Коростелев
Компьютерный набор А.А. Азарченков
Иллюстрации А.А. Азарченков

Подписано в печать 07.07.2017г. Формат 60x84 1/16 Бумага офсетная. Офсетная печать. Усл.печ.л. 1,6 Уч.-изд.л. 1,6 Тираж 1 экз.

Брянский государственный технический университет
Кафедра «Информатика и программное обеспечение», тел. 56-09-84
241035, Брянск, бульвар 50 лет Октября, 7 БГТУ