**Дисциплина «Надежность программного обеспечения»**

**Практическое занятие № 2**

**Тема:** Оценка характеристик программ на основе лексического анализа с использованием метрики Джилба.

**Время выполнения практического занятия (аудиторные часы)** – 4 часа.

**Цель работы:** изучить методику оценки характеристик программ с использованием метрики Джилба.

**Оборудование и программное обеспечение:** работа выполняется на ПЭВМ типа IBM PC с использованием стандартных функций ОС.

## 1. Теоретические сведения

Достаточно практичными являются метрики программного обеспечения, предложенные Томасом Джилбом (Thomas Jilb) 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.

Логическая 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

$$f = \frac{N_{SV}^4}{L_{mod}}; \quad (2.1)$$

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

$$f^* = \frac{N_{SV}^*}{L}. \quad (2.2)$$

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 ошибки на входе. Так, например, если одна ошибка вызывает в течение определенного периода времени появление 50 сообщений об ошибках, то прецизионность равна 0,02.

## Практическая реализация оценки характеристик программы на основе лексического анализа с использованием метрики Джилба

**Задача:** 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.

**Требуется:** Для разработанной программы на основе лексического анализа исходного текста определить значения метрик Джилба.

### Реализация программы

Текст программы для реализации возможного решения поставленной задачи, разработанной с использованием языка программирования C#, приведен на Таблице 2.1.

*Таблица 2.1 Текст программы 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 | ifl(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 strl) |
| 24 | { |
| 25 | Console. WriteLine(strl); |
| 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}", "Результирующий 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  }
```

В 6-й, 42-й, 44-й и 50-й строках (см. табл. 2.1) 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.

### Словарь программы

В таблице 2.2 приведены операторы и операции, используемые в программе (столбец 2). Номера строк исходной программы, где встречается каждый оператор или операция, указаны в третьем столбце. В четвертом столбце указано число повторений каждого оператора или операции в исходном тексте программы.

*Таблица 2.2 Операторы и операции, используемые в программе*

| № 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 |
| 14 | Console.ReadLine() | 42, 44, 57 | 3 |
| 15 | for() | 8, 17, 26 | 3 |
| 16 | do{}while() | 38-58 | 1 |
| 17 | if() | 10,28 | 2 |
| 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** | **264** |

количество операторов цикла $L_{loop}$;

- количество операторов условия $L_{IF}$;
- число модулей или подсистем $L_{mod}$;

Определим значения характеристик $L_{IF}$, $L_{loop}$, $L_{mod}$, $f$, $L$, $CL$ и $cl$. Значение характеристики $L_{IF}$ определяется количеством используемых в программе операторов *if*. В представленном решении их два (см. табл. 2.2, п. 17). Значение характеристики $L_{loop}$ определяется количеством используемых в программе циклов. В исходном тексте данной программы содержится четыре цикла: три оператора *for* и один *do... while* (см. табл. 2.2, 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()* (см. табл. 2.1, строка 33);
- *public static int[] Copy(int[] a)* (см. табл. 2.1, строка 4);
- *public static void Zapoln(ref int[] a, Random g)* (см. табл. 2.1, строка 15);
- *public static void Print(int[] a, string str, string strl)* (см. табл. 2.1, строка 23).

Общее количество операторов условия 6, из них 2 оператора *if* и четыре оператора цикла. Общее число всех используемых операторов $L = 264$ (см. табл. 2.2). Таким образом:

- $CL = 6$ - абсолютная сложность программы;
- $cl = CL/L = 6/264 = 0{,}0227$.

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 (см. форм 2.1):

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

**Вывод:** Из полученных результатов анализа текста программы следует, что исходный код имеет невысокую сложность, так как на 264 оператора текста приходится всего лишь 6 операторов условий. Общее число программных модулей решения также невелико (4 модуля), что подтверждает низкий уровень сложности программы.

## 2. Задание

При оценке характеристик программ на основе лексического анализа с использованием метрики Джилба необходимо выполнить следующее:

1. разработать программу, реализующую заданный, в соответствии с вариантом, алгоритм (рекомендуется использовать язык программирования C#);
2. оценить характеристики разработанной программы на основе лексического анализа текста и применения метрик Джилба.

Практическая работа выполняется в соответствии с вариантом (номер варианта соответствует номеру студента в журнале группы).

### Варианты заданий

**Вариант 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.

**Вариант 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.

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

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

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

**Вариант 4.** Написать и протестировать функцию, преобразующую строку восьмеричных цифр в эквивалентное ей целое десятичное число.

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

**Вариант 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.

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

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

**Вариант 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.

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

**Вариант 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.

**Вариант 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.

**Вариант 13**. Вычислить сумму первых *N* членов ряда

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

Количество суммируемых элементов *N* задается с клавиатуры.

**Вариант 14**. Вычислить сумму первых *N* членов ряда

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

Количество суммируемых элементов *N* задается с клавиатуры.

**Вариант 15.** Вычислить сумму цифр натурального числа, задаваемого с клавиатуры.

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

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

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

$$f(x)=\begin{cases} sin\left(\frac{\pi}{2}\right), \text{если } x \le 0.5; \\ sin\left((x-1)\cdot\frac{\pi}{2}\right), \text{если } x > 0.5. \end{cases}$$

Вычисления проводить для значений аргумента $\boldsymbol{x}$ от -0,4 до 1,3 с шагом 0,1.

**Вариант 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 $|y_{i+1}-y_i|<2\cdot10^{-5}$.

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

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

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.

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

**Вариант 22.** Вычислить значения функции $f(x)$:

$$f(x)=\begin{cases} sin\left(\frac{\pi}{8}+|x|\right), \text{если } x < 0.3; \\ sin\left(x^2\cdot\frac{\pi}{2}\right), \text{если } x \ge 0.3. \end{cases}$$

Вычисления проводить для значений аргумента $\boldsymbol{x}$ от -0,5 до 1,2 с шагом 0,1.

**Вариант 23.** Вычислить значения функции $f(x)$:

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

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.

**Вариант 24**. Вычислить значения функции $f(x) = \sqrt[3]{x}$ по следующей итерационной формуле:

$$y_{i+1} = 0.5 \cdot \left( y_i + \frac{3x}{2y_i^2 + \frac{x}{y_i}} \right);\ y_0 = x.$$

Итерации прекратить при выполнении условия $|y_{i+1} - y_i| < 10^{-5}$

**Вариант 25**. Вычислить значения функции

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

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

## 3. Требования к отчету

Отчёт выполняется каждым студентом индивидуально. Работа должна быть оформлена в электронном виде в формате .doc и распечатана на листах формата А4. На титульном листе указываются: наименование учебного учреждения, наименование дисциплины, название и номер работы, вариант, выполнил: фамилия, имя, отчество, группа, проверил: преподаватель ФИО (образец титульного листа представлен в приложении 1).

Отчет должен содержать:

— титульный лист;
— цель работы;
— краткие теоретические сведения, ответы на контрольные вопросы;
— задания в соответствии с вариантом;
— выводы.

## 4. Контрольные вопросы

1. Что в метрике Джилба является мерой логической трудности, какое практическое значение эта мера имеет?
2. Какие характеристики программного средства предусмотрены метрикой Джилба?
3. Как автор метрики определяет надежность программы?
4. Что такое прецизионность в метриках Джилба?

## 5. Литература

1. Керниган Б., Ритчи Д. Язык программирования С: Пер. с англ. М.: Вильямс, 2009.

2. Майерс Г. Надежность программного обеспечения. М.: Мир, 1980.

3. Холстед М.Х. Начала науки о программах. М.: Финансы и статистика, 1981.

4. Черников Б.В. Управление качеством программного обеспечения: учебник. М.: ИД «ФОРУМ»: ИНФРА-М, 2012.

**Приложение 1**

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)**

Факультет Информатика и вычислительная техника

Кафедра Кибербезопасность информационных систем

**Практическое занятие № ____**

на тему «_____»

Выполнил обучающийся гр._____

_____

(Фамилия, Имя, Отчество)

Проверил:

_____

(должность, Фамилия, Имя, Отчество)

Ростов-на-Дону

2018