**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФГБОУ ВПО  
**«БРЯНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**Кафедра** «Информатика и программное обеспечение»

**ЛАБОРАТОРНЫЕ РАБОТЫ ПО ДИСЦИПЛИНЕ**

**«Метрология и качество программного обеспечения»**

Вариант 10

Выполнил студент гр. 15-ИВТ2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Шалом В.Ю.

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2018

Руководитель

\_\_\_\_\_\_\_\_к.т.н., доц. Азарченков А.А.

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2018

Брянск 2018

1. Лабораторная работа №1
   1. Задача «Сжатие текста»

Разработать программу для сжатия строк путем удаления из нее пробелов. Ввод строки производится с клавиатуры. Результат вывести на экран.

*Определить значения метрик Холстеда, на основе которых дать оценку качества разработанного исходного текста программы.*

* + 1. Реализация программы

Текст программы для реализации возможного решения поставленной задачи, разработанной с использованием языка программирования C#, приведен в табл.ица 1.

Таблица 1

Исходный текст программы

|  |  |
| --- | --- |
| Номера строк | Строки программы |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | using System;  namespace lab01  {  class Program  {  static void Main()  {  string someStr;  Console.Write("Исхд. строка: ");  someStr = Console.ReadLine();  someStr = someStr.Replace(" ", string.Empty);  Console.WriteLine("Форм. строка: " + someStr);  }  }  } |

* + 1. Словарь программы

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

Таблица 2

Словарь операторов и операций программы

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Операторы, операции | Номера строк | Количество повторений |
| 1 | using … | 1 | 1 |
| 2 | namespace … | 2 | 1 |
| 3 | class … | 4 | 1 |
| 4 | static void … | 6 | 1 |
| 5 | string … | 8 | 1 |
| 6 | Console.Write() | 9 | 1 |
| 7 | Console.ReadLine() | 10 | 1 |
| 8 | someStr.Replace() | 11 | 1 |
| 9 | Console.WriteLine() | 12 | 1 |
| 10 | + | 12 | 1 |
| 11 | {} | 3(23), 5(22), 7(13), | 3 |
| 12 | “” | 9, 12, 18 | 3 |
| 13 | () | 6, 9, 10, 11, 12 | 5 |
| 14 | ; | 1, 8, 9, 10, 11, 12 | 6 |
| 15 | , | 18 | 1 |
| 16 | . | 9, 10, 12, 16, 18 | 5 |
| 17 | = | 10, 11 | 2 |
| **Всего** | | | 35 |

Словарь всех операторов в исходном тексте этой реализации программы сведен в столбце 2. В третьем столбце этой таблицы приведены номера строк исходной программы, где встречаются операторы. В последнем столбце приводится количество повторений (число вхождений) операторов в тексте программы. Таким образом, количество строк этой таблицы есть число уникальных операторов программы. Сложив значения четвертого столбца, получим общее число вхождений всех операторов. Проведем подробный анализ исходного текста программы в соответствии с полученной таблицей, начиная с первой позиции (первая строка программы (𝑢𝑠𝑖𝑛𝑔 𝑆𝑦𝑠𝑡𝑒𝑚;).

Ключевое слово 𝑢𝑠𝑖𝑛𝑔 представляет собой команду (инструкцию), обеспечивающую доступ к именам пространства имен 𝑆𝑦𝑠𝑡𝑒𝑚, следовательно, команду можно отнести к выполняемым операторам. Оператор 𝑢𝑠𝑖𝑛𝑔 встречается в программе всего один раз. Слово 𝑆𝑦𝑠𝑡𝑒𝑚 представляет собой имя, над которым осуществляется операция 𝑢𝑠𝑖𝑛𝑔. Таким образом, имя 𝑆𝑦𝑠𝑡𝑒𝑚 заносится в таблицу словаря операндов (табл.ица 3). Имя 𝑆𝑦𝑠𝑡𝑒𝑚 встречается в программе один раз.

Следующая строчка программы namespace lab01 состоит из оператора namespace и операнда lab01, которые присутствуют в тексте программы в единственном экземпляре. Оператор занесен в таблицу операторов, а операнда в таблицу операндов.

Как и предыдущая пара оператор–операнда, следующие строки

* class Program;
* static void Main().

также представляют собой сочетания, которые один раз в тексте программы, где ключевые слова class и static void представляю представляют соответственно операции, а Program и Main() – операнды. Все операции попадают в словарь операторов, а имена – в словарь операндов.

Следующая пара оператор–операнда string someStr, где string обозначает оператор типа данных, возвращаемого оперируемой переменной (операндой), а someStr, собственно, саму переменную. Оператор string встречается в тексте программы 1 раз, операнда someStr – 5 раз.

Следующей операцией является вызов функции вывода строк без переноса на следующую строку Console.Write(). Операция используется всего 1 раз. Входным параметром функции является строка со значением "Исхд. строка: ".

Следующая операция someStr = Console.ReadLine(); включает в себя операнду, над которой производятся действия и два оператора:

* = – оператор присваивания, используется в программе 2 раза;
* Console.ReadLine() – оператор вызова функции считывания строки с клавиатуры используется в программе 1 раз.

Операция someStr = someStr.Replace(" ", string.Empty); состоит из операнды someStr, оператора присваивания = и оператора вызова функции замены символов строковой константы someStr.Replace(), который используется в программе 1 раз.

Далее идет операция вызова функции вывода строк на экран монитора с переносом на следующую строку – Console.WriteLine(). Данная операция встречается в тексте программы 1 раз. Входными параметрами функции являются строка, значение которой "Форм. строка: " и переменная someStr.

Имена “ ” и string.Empty являются операндами, где “ ” – строка, все вхождения которой будут заменены на пустое поле string.Empty при вызове метода someStr.Replace(), т.е. удалены.

Символы “;”, “,”, “+”, “.”, используемые в программе, обозначают следующие операции:

* “;” – операция определения завершения оператора, используется 6 раз;
* “,” – операция отделения элементов списка, используется 1 раз;
* “.” – операция связывания имен, используется 5 раз;
* “+” – операция сложения (сцепления строк), используется 1 раз.

В позициях 11 и 13 таблицы операторов представлены символы, определяющие следующие операции:

* {} – операция начала и завершения блока инструкций, используется 3 раза;
* () – операция начала и завершения списка параметров или условия, используется 5 раз.

Оставшийся в табл.ица 2 символ в позиции 12: “” – представляет собой операцию определения строковых констант, используется 1 раз.

Таблица 3

Словарь операндов программы

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Операнды | Номера строк | Количество повторений |
| 1 | System | 1 | 1 |
| 2 | lab01 | 2 | 1 |
| 3 | Program | 3 | 1 |
| 4 | Main | 6 | 1 |
| 5 | someStr | 8, 10, 11, 11, 12 | 5 |
| 6 | “Исхд. Строка: ” | 9 | 1 |
| 7 | “Форм. Строка: ” | 12 | 1 |
| 8 | “ ” | 11 | 1 |
| **Всего** | | | 12 |

Проанализируем содержание табл.ицы 3. Позиции 1, 2, 3 и 4 содержат имена операндов 𝑆𝑦𝑠𝑡𝑒𝑚, lab01, 𝑃𝑟𝑜𝑔𝑟𝑎𝑚, 𝑀𝑎𝑖𝑛(), которые используются в программе по одному разу. Операнда someStr представляет собой строковую константу, содержащую в себе произвольный набор символов, введенный с клавиатуры, используется 5 раз. Строковые константы в позициях 6, 7 и 8 в табл.ица 3:

* “Исхд. Строка: ”;
* “Форм. Строка: ”;
* “ ”;

используются в программе однократно.

Список входных и выходных параметров рассматриваемой программы приведен в табл.ице 4. Единственным входным параметром программы является значение переменной someStr = Console.ReadLine();

Таблица 4

Входные и выходные переменные программы

|  |  |
| --- | --- |
| Входные переменные | Выходные переменные |
| someStr | “Исхд. строка: ” |
|  | “Форм. строка: ” |
|  | someStr |

Выходными значениями являются три строковые константы:

* Console.Write("Исхд. строка: ");
* Console.WriteLine("Форм. строка: " + someStr) – в этом случае два выходных параметра: строковая константа "Форм. Строка: " и переменная someStr;
  + 1. Оценка характеристик программы

Используя сформированные таблицы с необходимыми параметрами для расчета и применяя соотношения Холстеда, вычислим характеристики рассматриваемой программы:

* словарь программы:

;

* длина реализации:

;

* длина программы:
* объем программы в битах:

;

* потенциальный объем программы:
* уровень программы:
* уровень языка:

;

* интеллектуальное содержание программы:

;

* работа по программированию:

Сведем все результаты расчетов метрик Холстеда в табл.ице 5.

Таблица 5

Значения метрик Холстеда для программы

|  |  |  |
| --- | --- | --- |
| Наименование характеристики | Обозначение и формула для вычисления | Значение |
| Число простых (уникальных) операторов и операций |  | 17 |
| Число простых (уникальных) операндов |  | 8 |
| Общее число всех операторов и операций |  | 35 |
| Общее число всех операндов |  | 12 |
| Число входных и выходных переменных (параметров) |  | 4 |
| Словарь программы |  | 25 |
| Длина реализации программы |  | 47 |
| Объем программы (в битах) |  |  |
| Потенциальный объем программы |  |  |
| Уровень реализации программы |  |  |
| Уровень реализации языка |  |  |
| Работа программирования |  |  |

* 1. Вывод

Реализация исследуемой программы достаточно посредственная, т.к. реальный объем программы в 14 раз превышает потенциальный. Возможна дальнейшая оптимизация.

1. Лабораторная работа №2
   1. Задача «Вычисление суммы первых 𝑁 членов ряда»

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

*Необходимо разработать программу для вычисления суммы первых N членов ряда и на основе лексического анализа исходного текста программы, оценить её качество с использованием метрик Джилба.*

* + 1. Реализация программы

Текст программы для реализации возможного решения поставленной задачи, разработанной с использованием языка программирования C#, приведен в табл.ице 6.

Таблица 6

Исходный текст программы

|  |  |
| --- | --- |
| Номера строк | Строки программы |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | using System;  namespace lab02  {  class Program  {  static void Main()  {  bool isRepeat;  do  {  int[] row = { 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29 };  Console.WriteLine("Введите число элементов: ");  int numb = Convert.ToInt32(Console.ReadLine()), summ = 0;  for (int i = 0; i < numb; i++)  {  summ += row[i];  }  Console.WriteLine("Сумма элементов ряда: " + summ);  Console.WriteLine("Повторить?(Y/N)");  char input = Convert.ToChar(Console.ReadLine());  if (input == 'Y' || input == 'y')  isRepeat = true;  else  isRepeat = false;  } while ( isRepeat );  }  }  } |

Программа, разработанная для реализации заданного алгоритма, имеет циклы и один оператор ветвления. Следовательно, из перечисленного набора характеристик можно определить следующие: 𝐿, , , 𝐶𝐿 и 𝑐𝑙.

* + 1. Словарь программы

При подсчете общего количества операторов 𝐿 программы будем руководствоваться правилами, определенными при подсчете операторов в метриках Холстеда. В табл.ице 7 приведены операторы и операции, используемые в программе.

Таблица 7

Словарь операторов и операций программы

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Операторы, операции | Номера строк | Количество повторений |
| 1 | using … | 1 | 1 |
| 2 | namespace … | 2 | 1 |
| 3 | class … | 4 | 1 |
| 4 | static void … | 6 | 1 |
| 5 | bool … | 8 | 1 |
| 6 | do … while() | 9 – 25 | 1 |
| 7 | int[] … | 11 | 1 |
| 8 | Console.WriteLine() | 12, 18, 19 | 3 |
| 9 | int … | 13, 14 | 1 |
| 10 | Convert.ToInt32() | 13 | 1 |
| 11 | Console.ReadLine() | 13, 20 | 2 |
| 12 | for() | 14 | 1 |
| 13 | char … | 20 | 1 |
| 14 | Convert.ToChar() | 20 | 1 |
| 15 | if() … else … | 21(24) | 1 |
| 16 | [] | 11, 16 | 2 |
| 17 | += | 16 | 1 |
| 18 | < | 14 | 1 |
| 19 | == | 21, 21 | 2 |
| 20 | ++ | 14 | 1 |
| 21 | + | 18 | 1 |
| 22 | {} | 3(28), 5(27), 7(26), 10(25), 11(11), 15(17) | 6 |
| 23 | “” | 12, 18, 19 | 3 |
| 24 | () | 6, 12, 13, 13, 14, 18, 19, 20, 21, 25 | 10 |
| 25 | ; | 1, 8, 11, 12, 13, 14, 14, 16, 18, 19, 20, 22, 24, 25 | 14 |
| 26 | , | 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 13 | 15 |
| 27 | . | 12, 13, 13, 18, 19, 20, 20 | 7 |
| 28 | = | 11, 13, 13, 14, 20, 22, 24 | 7 |
| 29 | || | 21 | 1 |
| **Всего** | | | 89 |

* + 1. 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 С# к такой категории инструкций могут относиться условные операторы ветвления 𝑖𝑓 операторы циклов 𝑓𝑜𝑟 ... 𝑤ℎ𝑖𝑙𝑒 и 𝑑𝑜 ... 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 условный оператор ветвления if который располагается в строке 21, операторы циклов do ... while и for, в строках 9(25) и 14(17) соответственно.

* количество операторов цикла равно 2;
* количество операторов условия равно 1;
* абсолютная сложность ;
* относительная сложность программы равна:
  1. Вывод

Основываясь на результатах лексического анализа исходного текста программы можно сделать вывод о невысокой сложности представленного решения, т.к. количество ветвлений в программе достаточно небольшое (2 оператора цикла и 1 оператор условия), что подтверждается низким числом метрики относительной сложности программы, которое равно 0.0337.

1. Лабораторная работа №3
   1. Задача «Подсчет суммы элементов матрицы из заданного диапазона»

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

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

* + 1. Реализация программы

Текст программы для реализации возможного решения поставленной задачи, разработанной с использованием языка программирования C#, приведен в табл.ице 8.

Таблица 8

Исходный текст программы

|  |  |
| --- | --- |
| Номера строк | Строки программы |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59 | using System;  namespace lab03  {  class Program  {  static void Main()  {  bool isRestart;  do  {  Random rand = new Random();  int N, M, summ = 0;  int i, j;  Console.Write("Введите кол-во строк [3-10]\nВВОД: ");  N = Convert.ToInt32(Console.ReadLine());  Console.Write("\nВведите кол-во столбцов [3-10]\nВВОД: ");  M = Convert.ToInt32(Console.ReadLine());  int[,] matrix = new int[N, M];  for (i = 0; i < N; i++)  {  for (j = 0; j < M; j++)  {  matrix[i, j] = rand.Next(11) - 5;  }  }  Console.WriteLine("\nМатрица: ");  for (i = 0; i < N; i++)  {  for (j = 0; j < M; j++)  {  Console.Write(String.Format("[{0, 2}]", matrix[i, j]));  if (i == j)  {  if (j < M - 1)  {  summ = summ + matrix[i, j] + matrix[i, j + 1];  }  else if (j == M - 1)  {  summ += matrix[i, j];  }  }  }  Console.WriteLine();  }  Console.WriteLine("\nСумма элементов\nСУММА: " + summ);  char input = Convert.ToChar(Console.ReadLine());  if (input == 'Y' || input == 'y')  isRestart = true;  else  isRestart = false;  } while (isRestart);  }  }  } |

* + 1. 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. Все множество переменных, составляющих список ввода-вывода, разбивается на четыре функциональные группы:

* 𝑃 – вводимые переменные для расчетов и для обеспечения вывода;
* 𝑀 – модифицируемые, создаваемые внутри программы переменные;
* 𝐶 – переменные, участвующие в управлении работой программного модуля (управляющие переменные);
* 𝑇 – не используемые в программе переменные.

Проанализируем исходный текст программы, чтобы определить ее характеристики для расчета метрики Чепина (табл.ица 9).

Таблица 9

Результат анализа объявленных переменных

|  |  |  |
| --- | --- | --- |
| № п/п | Наименования переменных | Номера строк |
| P (для расчетов и обеспечения вывода) | | |
| 1 | N | 12 |
| 2 | M | 12 |
| 3 | matrix | 20 |
| M (модифицируемые или создаваемые) | | |
| 1 | summ | 12 |
| 2 | i | 13 |
| 3 | j | 13 |
| C (управляемые переменные) | | |
| 1 | isRestart | 8 |
| 2 | input | 51 |
| T (не используемые в программе) | | |
|  | Отсутствуют |  |

Переменные N, M, matrix используются в качестве исходных данных.

Переменные summ, i и j. в процессе выполнения программы создаются и модифицируются.

Переменные isRestart и input используются для управления выполнением программы.

Таким образом, исходя из результатов анализа исходного текста программы, получаем следующие значения характеристик:

* – количество переменных для расчетов;
* – количество модифицируемых переменных;
* – количество переменных, используемых в управлении программой;
* – количество неиспользуемых переменных (такие переменные в программе отсутствуют).

Расчет метрики Чепина:

* 1. Вывод

Основываясь на результатах анализа, полученных с использованием метрики Чепина можно сделать вывод об низком уровне сложности данного решения, так как в исходном тексте программы было использовано относительно небольшое количество переменных, что не затрудняет чтение и понимание исходного кода.