

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 № 1**

на тему «**Надежность программного обеспечения**»

Выполнил обучающийся гр. \_\_\_ВИ41\_\_\_

\_\_\_\_\_\_\_\_\_\_\_Ващенко А. В.\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, Имя, Отчество)

Проверил:

\_\_\_\_\_\_\_доцент, Куликова О. В.\_\_\_\_\_\_\_\_

(должность, Фамилия, Имя, Отчество)

Ростов-на-Дону

2024

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

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

Ход работы

**Задание**

* 1. Разработать программу, реализующую заданный, в соответствии с вариантом, алгоритм (рекомендуется использовать язык программирования С#);
  2. Сформировать словарь программы, охватывающий операнды, а также операторы и операции;
  3. Словари оформить в виде таких же таблиц, как в рассмотренном примере;
  4. Рассчитать метрики Холстеда, оформив результат в виде итоговой таблицы;
  5. Провести анализ полученных результатов, сформировав содержательные выводы.

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

Результат выполнения работы программы (см. Рис 1):

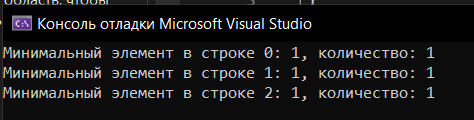


Рисунок 1 – Подсчёт минимальных элементов

Текст программы для реализации возможного решения поставленной задачи, разработанной с использованием языка программирования С#, приведен на Таблице 1.1.

*Таблица 1.1. Текст программы задачи «Подсчёт минимальных элементов»*

|  |  |
| --- | --- |
| **Номера**  **строк** | **Строки программы** |
| 1 | using System; |
| 2 | class Program |
| 3 | { |
| 4 | static void Main() |
| 5 | { |
| 6 | int[,] matrix = { |
| 7 | { 1, 2, 3 }, |
| 8 | { 4, 5, 1 }, |
| 9 | { 6, 1, 8 } |
| 10 | }; |
| 11 | for (int i = 0; i < matrix.GetLength(0); i++) |
| 12 | { |
| 13 | int min = matrix[i, 0]; |
| 14 | int count = 0; |
| 15 | for (int j = 0; j < matrix.GetLength(1); j++) |
| 16 | { |
| 17 | if (matrix[i, j] < min) |
| 18 | { |
| 19 | min = matrix[i, j]; |
| 20 | count = 1; |
| 21 | } |
| 22 | else if (matrix[i, j] == min) |
| 23 | { |
| 24 | count++; |
| 25 | } |
| 26 | } |
| 27 | Console.WriteLine($"Минимальный элемент в строке {i}: {min}, количество: {count}"); |
| 28 | } |
| 29 | } |
| 30 | } |

В таблице 1.2 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 1.2 Операторы и операции, используемые в программе*

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Операторы,  операции | Номера строк | Количество  повторений |
| 1 | using …; | 1 | 1 |
| 2 | сlass…; | 2 | 1 |
| 3 | static void ...; | 3 | 1 |
| 4 | int | 6, 11, 13, 14, 15 | 5 |
| 5 | for | 11, 15 | 2 |
| 6 | if | 17, 22 | 2 |
| 7 | else | 22 | 1 |
| 8 | ConsoIe.WriteLine() | 27 | 1 |
| 9 | matrix.GetLength() | 11, 15 | 2 |
| 10 | ; | 1, 10, 11, 13, 14, 15, 19, 20, 24, 27 | 12 |
| 11 | , | 6, 7, 8, 9, 13, 17, 19, 22 | 13 |
| 12 | = | 6, 11, 13, 14, 15, 19, 20 | 7 |
| 13 | ++ | 11, 15, 24 | 3 |
| 14 | < | 11, 15, 17 | 3 |
| 15 | = = | 22 | 1 |
| 16 | $ | 27 | 1 |
| 17 | {} | 3(30), 5(29), 6(10), 7, 8, 9, 12(28), 16(26), 18(21), 23(25), 27 | 12 |
| 18 | () | 11, 15, 17, 22 | 4 |
| 19 | [] | 6, 13, 17, 19, 22 | 5 |
| 20 | " " | 27 | 1 |
| **Всего** | | | **78** |

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.

В табице 1.3 приведены операнды рассматриваемой программы. По аналогии с таблицей 1.2 в третьем столбце находятся номера строк исходной программы, где встречается каждый операнд. В четвертом столбце указано число повторений каждого операнда в исходном тексте программы.

*Таблица 1.3. Словарь операндов программы*

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Операнды | Номера строк | Количество  повторений |
| 1 | System | 1 | 1 |
| 2 | Program | 2 | 1 |
| 3 | Main | 4 | 1 |
| 4 | matrix | 6, 11, 13, 15, 17, 19, 22 | 7 |
| 5 | Числовые значения: 0, 1, 2, 3, 4, 5, 6, 8 | 7, 8, 9, 11, 13, 14, 15, 20 | 16 |
| 6 | i | 11, 13, 17, 19, 22, 27 | 8 |
| 7 | j | 15, 17, 19, 22 | 6 |
| 8 | min | 13, 17, 19, 22, 27 | 5 |
| 9 | count | 14, 20, 24, 27 | 4 |
| 10 | "Минимальный элемент в строке: , количество: " | 27 | 1 |
|  |  | **Всего** | **50** |

В таблице 1.4 указаны входные и выходные параметры рассматриваемой программы.

*Таблица 1.4. Входные и выходные переменные программы*

|  |  |
| --- | --- |
| **Входные переменные** | **Выходные переменные** |
| matrix | "Минимальный элемент в строке: , количество: " |
| i | min |
| j | count |

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 реализации программы:

;

* уровень реализации языка:

**;**

* интеллектуальное содержание программы:

**;**

* работа по программированию:

**.**

Сведем все результаты расчетов метрик Холстеда в таблицу 1.5.

*Таблица 1.5 Значения метрик Холстеда для программы*

|  |  |  |
| --- | --- | --- |
| **Наименование**  **характеристики** | **Обозначение и формула для вычисления** | **Значение** |
| Число простых (уникальных) операторов и операций |  | 20 |
| Число простых (уникальных) операндов |  | 10 |
| Общее число всех операторов и операций |  | 78 |
| Общее число всех операндов |  | 50 |
| Число входных и выходных переменных (параметров) |  | 6 |
| Словарь программы |  | 30 |
| Длина реализации программы |  | 128 |
| Объем программы (в битах) |  | 628,48 |
| Потенциальный объем программы |  | 24 |
| Уровень реализации программы |  | 0.039 |
| Уровень реализации языка |  | 0,936 |
| Работа программирования |  | 16114,88 |

**Вывод:** Уровень исследуемой программы весьма низкий, так как потенциальный объем программы в значительной степени меньше ее реального объема.

**Задание 2 (домашняя работа)**

Организуйте в Windows приложении ввод массива декартовых координат n точек на плоскости. Вычислите массив полярных координат этих точек и организуйте вывод этого массива. Обеспечьте контроль вводимых значений.

Результат выполнения работы программы (см. Рис 2):

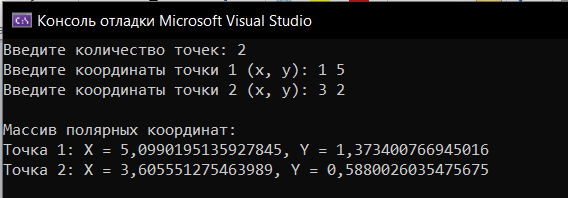


Рисунок 2 – Массив полярных координат

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

*Таблица 2.1. Текст программы задачи «Подсчёт минимальных элементов»*

|  |  |
| --- | --- |
| **Номера**  **строк** | **Строки программы** |
| 1 | using System; |
| 2 | class Program3 |
| 3 | { |
| 4 | static void Main(string[] args) |
| 5 | { |
| 6 | Console.Write("Введите количество точек: "); |
| 7 | int n = int.Parse(Console.ReadLine()); |
| 8 | double[,] cartPoints = new double[n, 2]; |
| 9 | double[,] polarPoints = new double[n, 2]; |
| 10 | for (int i = 0; i < n; i++) |
| 11 | { |
| 12 | Console.Write($"Введите координаты точки {i + 1} (x, y): "); |
| 13 | string[] input = Console.ReadLine().Split(' '); |
| 14 | cartPoints[i, 0] = double.Parse(input[0]); |
| 15 | cartPoints[i, 1] = double.Parse(input[1]); |
| 16 | } |
| 17 | for (int i = 0; i < n; i++) |
| 18 | { |
| 19 | double x = cartPoints[i, 0]; |
| 20 | double y = cartPoints[i, 1]; |
| 21 | double X = Math.Sqrt(x \* x + y \* y); |
| 22 | double Y = Math.Atan2(y, x); |
| 23 | polarPoints[i, 0] = X; |
| 24 | polarPoints[i, 1] = Y; |
| 25 | } |
| 26 | Console.WriteLine("\nМассив полярных координат:"); |
| 27 | for (int i = 0; i < n; i++) |
| 28 | { |
| 29 | Console.WriteLine($"Точка {i + 1}: X = {polarPoints[i, 0]}, Y = {polarPoints[i, 1]}"); |
| 30 | } |
| 31 | } |
| 32 | } |

*Таблица 2.2 Операторы и операции, используемые в программе*

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Операторы,  операции | Номера строк | Количество  повторений |
| 1 | using …; | 1 | 1 |
| 2 | сlass…; | 2 | 1 |
| 3 | static void ...; | 4 | 1 |
| 4 | Console.Write() | 6, 12 | 2 |
| 5 | int | 7, 10, 17, 27 | 5 |
| 6 | …Parse() | 7, 14, 15 | 3 |
| 7 | double | 8, 9, 14, 15, 19, 20, 21, 22 | 10 |
| 8 | new | 8, 9 | 2 |
| 9 | string | 13 | 1 |
| 10 | Console.ReadLine() | 7, 13 | 2 |
| 11 | Split() | 13 | 1 |
| 12 | for | 10, 17, 27 | 3 |
| 13 | Math.Sqrt | 21 | 1 |
| 14 | Math.Atan2 | 22 | 1 |
| 15 | ConsoIe.WriteLine() | 26, 29 | 2 |
| 16 | ; | 1, 6, 7, 8, 9, 10, 12, 13, 14, 15, 17, 19, 20, 21, 22, 23, 24, 26, 27, 29 | 23 |
| 17 | , | 8, 9, 14, 15, 19, 20, 22, 23, 24, 29 | 13 |
| 18 | = | 7, 8, 9, 10, 13, 14, 15, 17, 19, 20, 21, 22, 23, 24, 27 | 15 |
| 19 | + | 12, 21, 29 | 3 |
| 20 | ++ | 10, 17, 27 | 3 |
| 21 | < | 10, 17, 27 | 3 |
| 22 | \* | 21 | 2 |
| 23 | $ | 12, 29 | 2 |
| 24 | {} | 3(32), 5(31), 11(16), 12, 18(25), 28(30), 29 | 9 |
| 25 | () | 10, 17, 27 | 3 |
| 26 | [] | 4, 8, 9, 13, 14, 15, 19, 20, 23, 24, 29 | 16 |
| 27 | " " | 6, 12, 26, 29 | 4 |
| 29 | ‘ ‘ | 13 | 1 |
| **Всего** | | | **133** |

*Таблица 2.3. Словарь операндов программы*

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Операнды | Номера строк | Количество  повторений |
| 1 | System | 1 | 1 |
| 2 | Program3 | 2 | 1 |
| 3 | Main | 4 | 1 |
| 4 | n | 7, 8, 9, 10, 17, 27, | 6 |
| 5 | cartPoints | 8, 14, 15, 19, 20 | 5 |
| 6 | polarPoints | 9, 23, 24, 29 | 5 |
| 7 | Числовые значения: 0, 1, 2 | 8, 9, 10, 12, 14, 15, 17, 19, 20, 23, 24, 27, 29 | 17 |
| 8 | i | 10, 12, 14, 15, 17, 19, 20, 23, 24, 27, 29 | 19 |
| 9 | input | 13, 14, 15 | 3 |
| 10 | x | 19, 21, 22, | 4 |
| 11 | y | 20, 21, 22 | 4 |
| 12 | X | 21, 23 | 2 |
| 13 | Y | 22, 24 | 2 |
| 14 | "Введите количество точек: " | 6 | 1 |
| 15 | "Введите координаты точки (x, y): " | 12 | 1 |
| 16 | "\nМассив полярных координат:" | 26 | 1 |
| 17 | "Точка: X = , Y = " | 29 | 1 |
|  |  | **Всего** | **74** |

*Таблица 2.4. Входные и выходные переменные программы*

|  |  |
| --- | --- |
| **Входные переменные** | **Выходные переменные** |
| n | "Введите количество точек: " |
| cartPoints | "Введите координаты точки (x, y): " |
| polarPoints | "\nМассив полярных координат:" |
| input | "Точка: X = , Y = " |

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 реализации программы:

;

* уровень реализации языка:

**;**

* интеллектуальное содержание программы:

**;**

* работа по программированию:

**.**

Сведем все результаты расчетов метрик Холстеда в таблицу 2.5.

*Таблица 2.5 Значения метрик Холстеда для программы*

|  |  |  |
| --- | --- | --- |
| **Наименование**  **характеристики** | **Обозначение и формула для вычисления** | **Значение** |
| Число простых (уникальных) операторов и операций |  | 29 |
| Число простых (уникальных) операндов |  | 17 |
| Общее число всех операторов и операций |  | 133 |
| Общее число всех операндов |  | 74 |
| Число входных и выходных переменных (параметров) |  | 8 |
| Словарь программы |  | 46 |
| Длина реализации программы |  | 207 |
| Объем программы (в битах) |  | 1143,12 |
| Потенциальный объем программы |  | 32,2 |
| Уровень реализации программы |  | 0.030 |
| Уровень реализации языка |  | 0,966 |
| Работа программирования |  | 38104 |

**Вывод:** Уровень исследуемой программы весьма низкий, так как потенциальный объем программы в значительной степени меньше ее реального объема.