1. Оценка характеристик
   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. 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. 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 𝑢𝑠𝑖𝑛𝑔 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. 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. 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. 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. 3:

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

используются в программе однократно.

Список входных и выходных параметров рассматриваемой программы приведен в табл. 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. 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 |  |  |

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