Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Омский государственный технический университет»

Кафедра «Информатика и вычислительная техника»

Отчёт по лабораторной работе № 2

по дисциплине

«Проектирование и тестирование программного обеспечения»

|  |  |
| --- | --- |
|  | Выполнил  Студент гр. МО-181  Грачев Д. В.    (подп., дата)  Проверил  Старший преподаватель каф. ИВТ  Звонов А.О.    (подп., дата) |

Омск 2020

**Задача 3.3 (Дешифратор II)**

PC/UVa IDs: 110304/850

Распространенный, но ненадежный метод шифровки текста состоит в перемене букв алфавита. Другими словами, каждая буква алфавита последовательно заменяется в тексте какой-то другой буквой. Чтобы шифровка была обратимой, никакие две буквы не заменяются одной и той же буквой.

Мощным методом криптоанализа является атака с известным открытым текстом (known plain text attack). При атаке с известным открытым текстом дешифровщик знает фразу или предложение, зашифрованное противником, и путем изучения зашифрованного текста выясняет метод кодировки.

Ваша задача - расшифровать несколько зашифрованных строк текста, полагая, что каждая строка использует один и тот же набор замещений и что одна из закодированных строк является шифровкой открытого текста *the quick brown fox jumps over the lazy dog*.

**Входные данные**

Входные данные начинаются со строки, содержащей одно положительное целое число, которое означает количество тестовых блоков, за которой следует пустая строка. Между двумя последовательными тестовыми блоками также находится пустая строка.

Каждый блок состоит из нескольких строк входных данных, зашифрованных способом, описанным выше. Зашифрованные строки содержат только строчные буквы и пробелы, и их длина не превышает 80 символов. Число строк входных данных не превышает 100. Номер известной вам строки не известен вам заранее.

**Выходные данные**

Для каждого тестового блока расшифруйте каждую строку и напечатайте их в стандартный вывод. Если существует несколько различных вариантов дешифровки, то подойдет любой. Если расшифровка невозможна, выведите No solution.

Выходные данные для двух последовательных блоков должны быть разделены

пустой строкой.

Ниже представлена программа:

namespace L21

{

class Program

{

const string keySent = "the quick brown fox jumps over the lazy dog";

static void Main()

{

try

{

using (StreamReader stream = new StreamReader("file.txt"))

{

int blocks = Convert.ToInt32(stream.ReadLine());

for (int i = 0; i < blocks; i++)

{

stream.ReadLine();

string input = null;

List<string> sentences = new List<string>();

while ((input = stream.ReadLine()) != null)

sentences.Add(input);

Tuple<bool, Dictionary<char, char>> tuple = new Tuple<bool, Dictionary<char, char>>(false, null);

for (int j = 0; j < sentences.Count && !tuple.Item1; j++)

{

tuple = KeySentCheck(sentences[j]);

}

if (tuple.Item1)

{

string res = "";

foreach (var sent in sentences)

res += Decrypt(sent, tuple.Item2) + "\n";

Console.WriteLine(res);

}

else Console.WriteLine("No solution");

}

}

}

catch (IOException IOe)

{

Console.WriteLine(IOe.Message);

}

catch (FormatException Fe)

{

Console.WriteLine(Fe.Message);

}

}

static Tuple<bool, Dictionary<char, char>> KeySentCheck(string str)

{

Dictionary<char, char> dict = new Dictionary<char, char>();

for (char i = 'a'; i <= 'z'; i++)

{

dict[i] = i;

}

int minL = keySent.Length < str.Length ? keySent.Length : str.Length;

for (int i = 0; i < minL; i++)

{

if (str[i] != keySent[i])

{

dict[str[i]] = keySent[i];

}

}

string res = "";

for (int i = 0; i < str.Count(); i++)

{

res += dict.Keys.Contains(str[i]) ? dict[str[i]] : str[i];

}

return new Tuple<bool, Dictionary<char, char>>(res == keySent, dict);

}

static string Decrypt (string str, Dictionary<char, char> dict)

{

string res = "";

for (int i = 0; i < str.Count(); i++)

{

res += dict.Keys.Contains(str[i]) ? dict[str[i]] : str[i];

}

return res;

}

}

}

Выбранный путь решения задачи:

1. Считывание данных

2. Создание коллекции зашифрованных фраз

3. Попытаться расшифровать каждую фразу, если одна из них расшифровывается как *the quick brown fox jumps over the lazy dog*, то, использую полученный словарь, расшифровать все фразы. Если ни одна из фраз не приводится к *the quick brown fox jumps over the lazy dog*, вывести сообщение и закончить программу.

Используемые структуры данных для решения задачи: словари, массивы, кортежи, списки, строки. классы

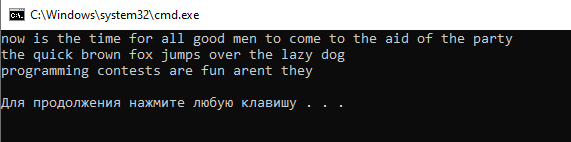
Результат выполнения программы для входных данных:

1

vtz ud xnm xugm itr pyy jttk gmv xt otgm xt xnm puk ti xnm fprxq

xnm ceuob lrtzv ita hegfd tsmr xnm ypwq ktj

frtjrpgguvj otvxmdxd prm iev prmvx xnmq



**Задача 4.3 (Задача сапожника)**

PC/UValDs: 110405/10026

У сапожника имеется N заказов от покупателей, которые он должен выполнить. Сапожник может заниматься в день только одним заказом, и заказы обычно требуют на выполнение несколько дней. Для i-го заказа целое число Ti  
(1 ≤ Ti ≤ 1000) означает число дней, необходимых сапожнику для завершения заказа.

Но за популярность нужно платить. За каждый день задержки перед тем, как он приступит к работе над i-м заказом, сапожник согласился платить штраф в размере Si (1 ≤ Si ≤ 1000) центов в день. Помогите сапожнику, написав программу, находящую последовательность работ, ведущую к минимальному штрафу.

**Входные данные**

Входные данные начинаются со строки, содержащей одно положительное целое число, которое означает количество тестовых блоков, за которой следует пустая строка. Между двумя последовательными тестовыми блоками также находится пустая строка.

Первая строка каждого блока содержит целое число, задающее число заказов N, причем 1 ≤ N ≤ 1000. i-я последующая строка содержит время завершения Ti и ежедневный штраф Si для i-го заказа.

**Выходные данные**

Для каждого тестового блока ваша программа должна вывести последовательность заказов, ведущую к минимальному штрафу. Каждый заказ представляется своей позицией во входных данных. Все целые числа должны находиться на одной строке выходных данных и каждая пара чисел должна быть разделена одним пробелом. Если возможны несколько решений, выведите первое в лексикографическом порядке.

Выходные данные для двух последовательных блоков должны быть разделены пустой строкой.

Ниже представлена программа:

Файл Program.cs

namespace L22

{

class Program

{

static void Main()

{

using (StreamReader stream = new StreamReader("file.txt"))

{

int blocks = Convert.ToInt32(stream.ReadLine());

stream.ReadLine();

for (int i = 0; i < blocks; i++)

{

int n = Convert.ToInt32(stream.ReadLine());

Order[] orders = new Order[n];

for (int j = 0; j < n; j++)

{

string input = stream.ReadLine();

orders[j] = new Order(j + 1, Convert.ToInt32(input.Split(' ')[0]), Convert.ToInt32(input.Split(' ')[1]));

}

Array.Sort(orders);

foreach (var ord in orders)

Console.Write($"{ord.n} ");

}

}

}

}

}

Файл Order.cs

namespace L22

{

class Order : IComparable<Order>

{

public int n;

public int dur;

public int pen;

public Order(int n, int dur, int pen)

{

this.n = n;

this.dur = dur;

this.pen = pen;

}

public int CompareTo(Order b)

{

return (dur \* b.pen).CompareTo(b.dur \* pen) == 0 ? n.CompareTo(b.n) : (dur \* b.pen).CompareTo(b.dur \* pen);

}

}

}

Выбранный путь решения задачи:

1. Считывание данных  
2. Формирование массива заказов  
3. Сортировка массива по соотношению длительность/штраф

4. Вывод отсортированного массива

Используемые структуры данных для решения задачи: массивы, строки, классы

Результат работы программы при следующих входных данных:

1

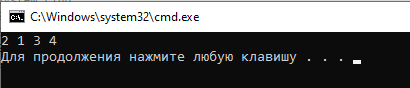
4

3 4

1 1000

2 2

5 5



Выводы:

В процессе выполнения лабораторной работы развил навыки работы со строками и словарями, вспомнил как работать с массивами.