Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра электронных вычислительных машин

Отчет

По лабораторной работе №1

|  |  |  |
| --- | --- | --- |
| Выполниили:  Ст. гр. 050501  Тускенис Д. С.  Горохович С. Г. |  | Проверил:  Кобяк И. П. |

Минск 2014

1. **ПОСТАНОВКА ЗАДАЧИ**

Последовательность «сигнал + шум» состоит из трёх слов, состоящих из 5 букв. Каждая из букв закодирована тремя битами, кодировка неизвестна. С каждым из слов была произведена операция сложения по модулю 2 с одной и той же шумовой последовательностью. Известны все возможные слова, которые могут быть зашифрованы в последовательности (см. Библиотеку), а также все возможные буквы, из которых составлены слова (см. Алфавит).

Используя произвольный язык программирования выполнить декодирование заданного варианта последовательности «сигнал + шум». При решении использовать метод перебора шумов для заданной реализации случайного процесса.

Библиотека:

АРМИЯ, МИЦАР, МАРИЯ, ТАРТУ, РАЦИЯ, МАРТА, МАРАТ,

ТИАРА, МИРТА, УТЯТА, ЦИТРА, МУМИЯ, ТРАТА, ТАТРА,

АРТУР, АРАМА, ТИМУР

Алфавит: М И Ц А Р Я Т У

Сообщение «сигнал+шум» (вариант 1):

000 101 101  001 101 010 110 010 011 001 001 000 000 100 001

1. **АЛГОРИТМ РЕШЕНИЯ**
2. Генерируются все возможные варианты кодирования алфавита. Все последующие пункты следует выполнять для каждого варианта кодирования.
3. Для каждой пары входных слов находится сумма по модулю два друг с другом.
4. Производится перебор всех пар слов из библиотеки и вычисляется их сумма по модулю два. При наличии такой же суммы в массиве, полученном в пункте 2, добавить ее в промежуточный массив, запомнив при этом порядковые номера слов в словаре.
5. Если после перебора всех пар слов из библиотеки промежуточный массив пункта 3 имеет количество элементов, равное количеству элементов массива из пункта 2, то в этом массиве содержаться порядковые номера слов, которые являются решениями. Для получения результата нужно перебрать все пары порядковых номеров и удалить повторяющиеся.
6. **ЛИСТИНГ ПРОГРАММЫ**

using System;

using System.Diagnostics;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

namespace Lab1

{

class Program

{

static List<String> letters = new List<String> { "М","И","Ц","А","Р","Я","Т","У" };

static List<String> library = new List<String> { "АРМИЯ", "МИРТА", "МИЦАР",

"УТЯТА", "МАРИЯ", "ЦИТРА",

"ТАРТУ", "МУМИЯ", "РАЦИЯ",

"ТРАТА", "МАРТА", "ТАТРА",

"МАРАТ", "АРТУР", "ТИАРА",

"АРАМА", "ТИМУР" };

static String signal = "000101101001101010110010011001001000000100001";

static void Main(string[] args)

{

SetupConsoleAppearance();

List<String> alphabets = GenerateAlphabets(letters);

var stopWatch = Stopwatch.StartNew();

foreach (String alphabet in alphabets)

{

List<String> xors = new List<string>();

List<String> words = WordsFromSignal(signal, alphabet);

for (int i = 1; i <= words.Count; i++)

{

xors.Add(XOR(words[i - 1], words[i % words.Count], alphabet));

}

List<Point> equals = new List<Point>();

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

{

for (int j = i + 1; j < library.Count; j++)

{

String xor = XOR(library[i], library[j], alphabet);

if (xors.Contains(xor))

{

equals.Add(new Point(i, j));

}

}

}

if (equals.Count == 3)

{

List<String> decodedWords = new List<string>();

foreach (Point p in equals)

{

if (!decodedWords.Contains(library[p.X]))

{

decodedWords.Add(library[p.X]);

}

if (!decodedWords.Contains(library[p.Y]))

{

decodedWords.Add(library[p.Y]);

}

}

foreach (String word in decodedWords)

{

Console.Write(word + " ");

}

Console.WriteLine("| " + AlphabetString(alphabet) + "| " +

BitStringFromWord(XOR(decodedWords[0], words[0], alphabet), alphabet));

}

}

stopWatch.Stop();

float secondsElapsed = (float)stopWatch.Elapsed.Milliseconds / 1000 +

stopWatch.Elapsed.Seconds;

Console.WriteLine();

Console.WriteLine("Decrypted in " + secondsElapsed + " seconds.");

Console.WriteLine("Press any key...");

Console.ReadKey();

}

static List<String> GenerateAlphabets(List<String> letters)

{

if (letters.Count == 2)

{

return new List<string>(new string[]

{ letters[0] + letters[1], letters[1] + letters[0] });

}

else

{

List<String> sets = new List<string>();

foreach (String l in letters)

{

List<String> newLetters = new List<string>(letters);

newLetters.Remove(l);

List<String> res = GenerateAlphabets(newLetters);

foreach (String st in res)

{

sets.Add(l + st);

}

}

return sets;

}

}

static String XOR(String s1, String s2, String alphabet)

{

String result = "";

if (s1.Length == s2.Length)

{

for (int i = 0; i < s1.Length; i++)

{

int v1 = alphabet.IndexOf(s1[i]);

int v2 = alphabet.IndexOf(s2[i]);

result += alphabet[v1 ^ v2];

}

}

return result;

}

static List<String> WordsFromSignal(String signal, String alphabet)

{

List<String> words = new List<string>();

for (int i = 0; i < 3; i++)

{

String encodedWord = signal.Substring(i \* 15, 15);

String word = "";

for (int j = 0; j < 5; j++)

{

word += alphabet[Convert.ToInt16(encodedWord.Substring(j \* 3, 3), 2)];

}

words.Add(new String(word.ToCharArray().Reverse().ToArray()));

}

return words;

}

static String AlphabetString(String alphabet)

{

String result = "";

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

{

result += letters[i] + ":";

String letterBits = Convert.ToString(letters.IndexOf(alphabet[i].ToString()), 2);

while (letterBits.Length < 3)

{

letterBits = letterBits.Insert(0, "0");

}

result += letterBits + " ";

}

return result;

}

static String BitStringFromWord(String word, String alphabet)

{

String result = "";

for (int i = 0; i < word.Length; i++)

{

String letterBits = Convert.ToString(alphabet.IndexOf(word[i]), 2);

while (letterBits.Length < 3)

{

letterBits = letterBits.Insert(0, "0");

}

result += letterBits + " ";

}

return result;

}

static void SetupConsoleAppearance()

{

Console.SetBufferSize(100, 300);

Console.SetWindowSize(100, 25);

}

}

}

1. **РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ**

