Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра прикладной математики и кибернетики

# Лабораторная работа № 3
по дисциплине «Теория Информации»

Выполнил:
студент группы ИП-712
Алексеев Степан
Владимирович
ФИО студента

Работу проверил:
доцент кафедры ПМИК Мачикина Е.П.
ФИО преподавателя

Новосибирск 2021 г.

# Оглавление



## ЗАДАНИЕ

*Теория информации*

Практическая работа №3

Вычисление энтропии Шеннона

Цель работы: Экспериментальное изучение свойств энтропии Шеннона для текстов на естественном языке.

Язык программирования: C, C++, C#, Python

Результат: программа, тестовые примеры, отчет.

Задание:

1. Для выполнения работы используйте один из файлов с исходным кодом программы, который был написан для практических работ 1, 2 или 3. Определите алфавит исходного кода, текст комментариев и ввода/вывода на экран игнорировать.

2. Составить программу, определяющую несколько оценок энтропии файла с исходным кодом. Оценки энтропии необходимо вычислить по формуле Шеннона двумя способами, т.е. используя частоты отдельных символов и используя частоты пар символов. По желанию можно продолжить процесс вычисления оценок с использованием частот троек, четверок символов и т.д.

3. После тестирования программы необходимо заполнить таблицу для отчета и проанализировать полученные результаты. Сравнить полученные результаты с результатами работы 1 и 2.

| Язык программирования | Максимально возможное значение энтропии | Оценка энтропии (одиночные символы) | Оценка энтропии (частоты пар символов) |
|---|---|---|---|
| | | | |

4. Оформить отчет, загрузить отчет и файл с исходным кодом в электронную среду. Отчет обязательно должен содержать заполненную таблицу и анализ полученных результатов. По желанию в отчет можно включить описание программной реализации. В отчет не нужно включать содержимое этого файла.

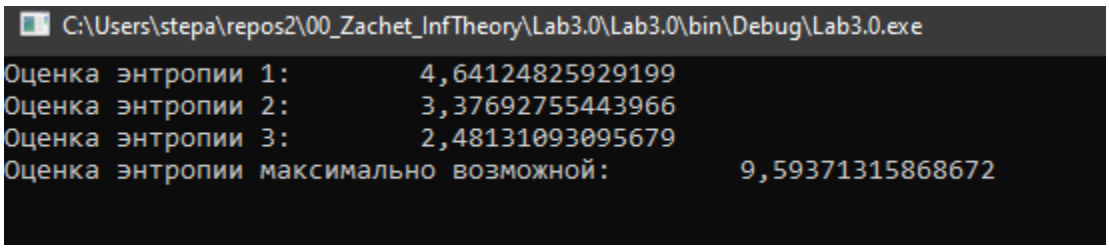
## Решение

| Язык программирования | Максимально возможное значение энтропии | Оценка энтропии (одиночные символы) | Оценка энтропии (частоты пар символов) |
|---|---|---|---|
| C# | 9,593713158686 | 4,6412482592919 | 3,3769275544396 |

## Анализ

В языке программирования определённость больше, чем в естетсвенном языке, т.к. структура программы строго определена, её нельзя(сложно) поменять с сохранением того же смысла, что и изначально. Поэтому максимально возможное значение энтропии выше.

## Скриншоты




## Листинг кода

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Lab3._0
{
    class Program
    {
        static Dictionary<string, double> dicti1 = new Dictionary<string, double>();
        static Dictionary<string, double> dicti2 = new Dictionary<string, double>();
        static Dictionary<string, double> dicti3 = new Dictionary<string, double>();
        static Dictionary<string, double> dicti20 = new Dictionary<string, double>();
        static Dictionary<string, double> dictiMax = new Dictionary<string, double>();
        static int numberOfChars = 0;
        static int numberOfLettersInABlock = 1;
        static void Main(string[] args)
        {

countProbabilitiesBasedOnRealFrequencyInFile("C:/Users/stepa/repos2/00_Zachet_InfTheory/Lab3.0/Program.txt", dicti1, numberOfLettersInABlock);
            Console.WriteLine("Оценка энтропии 1:        " + ShennonFormulaForEnthropy(dicti1, numberOfLettersInABlock));

            numberOfLettersInABlock = 2;

countProbabilitiesBasedOnRealFrequencyInFile("C:/Users/stepa/repos2/00_Zachet_InfTheory/Lab3.0/Program.txt", dicti2, numberOfLettersInABlock);
            Console.WriteLine("Оценка энтропии 2:        " + ShennonFormulaForEnthropy(dicti2, numberOfLettersInABlock));

            numberOfLettersInABlock = 3;

countProbabilitiesBasedOnRealFrequencyInFile("C:/Users/stepa/repos2/00_Zachet_InfTheory/Lab3.0/Program.txt", dicti3, numberOfLettersInABlock);
            Console.WriteLine("Оценка энтропии 3:        " + ShennonFormulaForEnthropy(dicti3, numberOfLettersInABlock));

            numberOfLettersInABlock = 1;
            foreach (var item in dicti1)
            {
                dictiMax.Add(item.Key, (double)1 / (double)dicti1.Count);
            }

countProbabilitiesBasedOnRealFrequencyInFile("C:/Users/stepa/repos2/00_Zachet_InfTheory/Lab3.0/Program.txt", dictiMax, numberOfLettersInABlock);
            Console.WriteLine("Оценка энтропии максимально возможной:        " + ShennonFormulaForEnthropy(dictiMax, numberOfLettersInABlock));

            Console.ReadLine();
        }
        static double ShennonFormulaForEnthropy(Dictionary<string, double> dict, int numberOfLettersInABlock)
        {//Количество информации, которое мы получаем, достигает максимального значения, если события равновероятны... Здесь, видимо,
            //сравниваются значения, полученные применением формулы Хартли...
            //Формула Шеннона позволяет высчитать среднее кол-во информации, передаваемое любым сообщением(блоком символов).
            double sum = 0;
            foreach (var item in dict)
            {
                sum += item.Value * Math.Log(1 / item.Value, 2);
            }
            return sum / numberOfLettersInABlock;
        }
        static void countProbabilitiesBasedOnRealFrequencyInFile(string path, Dictionary<string, double> dict, int numberOfLettersInABlock)
```

```
        {
            string str;
            using (StreamReader sr = File.OpenText(path))
            {
                str = sr.ReadToEnd();
            }
            numberOfChars = str.Length;
            char[] str_chars = str.ToCharArray();
            for (int i = 0; i < numberOfChars - numberOfLettersInABlock; i++)
            {
                string block = str_chars[i].ToString();
                for (int j = 1; j < numberOfLettersInABlock; j++)
                {
                    block += str_chars[i + j].ToString();
                }
                if (dict.ContainsKey(block))
                {
                    dict[block] += ((double)1 / ((double)numberOfChars));//   /
(double)numberOfLettersInABlock));
                }
                else
                    dict.Add(block, ((double)1 / ((double)numberOfChars)));// /
(double)numberOfLettersInABlock))) ;
            }
        }
    }
}
```