Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

**Лабораторная работа № 1**

« Основы теории чисел и их использование в криптографии»

Выполнил:

Студент: Герман А.Е.

ФИТ 3 курса 5 группы

Преподаватель: Савельева М.Г.

Минск 2024

# **1 Описание приложения**

Данное консольное приложение, разработанное на языке программирования C#, предназначено для проведения анализа энтропии в текстовых данных на различных языках, а также для вычисления энтропии в бинарных данных. Программа предоставляет пользователю инструменты для измерения количества информации в различных контекстах и сценариях передачи данных с учетом возможных ошибок.

Основные функциональности приложения:

* Расчет энтропии алфавитов:
* Вычисление энтропии для текстов на немецком и русском языках.
* Анализ алфавитов, примеров двоичных символов и вероятностей.
* Создание документа Excel с результатами.

Расчет энтропии бинарных алфавитов:

* Расчет энтропии для бинарных представлений текстов.
* Исследование бинарных алфавитов и их вероятностей.
* Создание документа Excel с результатами.
* Анализ информации для алфавитов и ASCII:

Оценка количества информации в текстах и их бинарных представлениях.

* Расчет энтропии для немецкого и русского языков.
* Создание документа Excel с результатами.
* Ошибка при передаче данных:

Исследование энтропии с учетом ошибок при передаче бинарных данных.

* Анализ ошибок для различных уровней (0.1, 0.5, 0.999).
* Создание документа Excel с результатами.

## **1.1 Цель приложения**

Предоставить пользователям инструмент для изучения и сравнения энтропии в текстовых данных на различных языках, а также для анализа эффектов ошибок при передаче бинарных данных. Приложение может быть использовано в образовательных целях для изучения основ теории информации.

# **2 Методика выполнения расчетов**

В данной лабораторной работе была поставлена цель создания приложения, позволяющее проанализировать два языка в рамках большого текста и ФИО. На листинге 2.1 представлен класс, реализующий данную функциональность.

|  |
| --- |
| class EntropyChecker : IDocumentReader, IEntropyCheck, IPrintEntropy  {  private string alphabetName;  private List<char> alphabet;  private double alphabetEntropy = 0;  string patternRussian = @"A";  string patternGerman = @"R";  public EntropyChecker()  {  }  public EntropyChecker(List<char> alphabet, double alphabetEntropy, string alphabetName)  {  Alphabet = alphabet;  AlphabetEntropy = alphabetEntropy;  AlphabetName = alphabetName;  }  private int myVar;  public List<char> Alphabet  {  get { return alphabet; }  set { alphabet = value; }  }  public string AlphabetName  {  get { return alphabetName; }  set { alphabetName = value; }  }  public double AlphabetEntropy  {  get { return alphabetEntropy; }  set { alphabetEntropy = value; }  }  public Dictionary<char, int> alphabetListToDictionary()  {  Dictionary<char, int> dict = new Dictionary<char, int>(Alphabet.Count());  foreach (char x in alphabet)  {  dict.Add(x, 0);  }  return dict;  }  public string GetAllText(string text, StreamReader reader)  {  if (reader == null)  {  throw new Exception("Document isn't open");  }  else  {  return reader.ReadToEnd();  }  }  public Dictionary<char, double> getSymbolsChances(string text, Dictionary<char, int> counts)  {  Dictionary<char, double> chances = new Dictionary<char, double>(alphabet.Count);  for (int i = 0; i < counts.Count(); i++)  {  chances.Add(alphabet[i], (double)counts[alphabet[i]] / text.Length);  }  return chances;  }  public void getSymbolsCounts(string text, Dictionary<char, int> alphabet)  {  for (int i = 0; i < text.Length; i++)  {  for (int j = 0; j < this.alphabet.Count(); j++)  {  if (text[i] == this.alphabet[j])  {  alphabet[this.alphabet[j]]++;  }  }  }  }  public void computeTextEntropy(Dictionary<char, double> chances)  {  for (int i = 0; i < alphabet.Count; i++)  {  if (chances[alphabet[i]] != 0)  {  AlphabetEntropy += chances[alphabet[i]] \* Math.Log(chances[alphabet[i]], 2);  }  }  AlphabetEntropy = -AlphabetEntropy;  }  public double computeTextEntropyWithError(Dictionary<char, double> chances, double p)  {  double q = 1 - p;  double entropy = 0;  double conditionalEntropy = 1 - (-p \* Math.Log(p, 2) - q \* Math.Log(q, 2));  if (double.IsNaN(conditionalEntropy))  {  return 0;  }  for (int i = 0; i < alphabet.Count; i++)  {  if (chances[alphabet[i]] != 0)  {  entropy += chances[alphabet[i]] \* Math.Log(chances[alphabet[i]], 2) - conditionalEntropy;  }  }  return -entropy;  }  public StreamReader OpenDocument(string path)  {  return new StreamReader(path);  }  public void printAlphabet()  {  Console.WriteLine($"\nАлфавит {AlphabetName}:"); ;  foreach (char x in alphabet)  {  if(x == 0) break;  Console.Write(x); Console.Write(" ");  }  }  public void printExampleOfBinaryChar()  {  var random = new Random();  Console.WriteLine($"\nАлфавит {AlphabetName}:");  var lowerBound = 0;  var upperBound = 1;  int rNum = random.Next(lowerBound, upperBound);  Console.WriteLine(rNum.ToString());  char firstChar = alphabet.ToArray()[rNum];  string binary = Convert.ToString(firstChar, 2);  Console.WriteLine("\tПример бинарного числа из алфавита (передние нули опущены): ");  Console.Write($"\t{binary}");  }  public void printChances(Dictionary<char, double> chances)  {  Console.WriteLine("\n\tШансы появления символа:");  foreach (char x in Alphabet)  Console.WriteLine($"\t{x} : {chances[x]}");  }  public void printAlhabetEntropy()  {  Console.WriteLine($"\nЭнтропия алфавита для языка '{AlphabetName}' равна {AlphabetEntropy}.");  }  }  }  }  } |

Листинг 2.1 – класс, реализующий необходимый функционал

Метод **computeTextEntropy** используется для получения энтропии для текста, где вероятность ошибки всегда равна 0. Метод **getSymbolsCounts** осуществляет подсчёт количества появлений символа в тексте. Метод **computeTextEntropyWithError** выполняет подсчёт энтропии с учётом переданного параметра системной ошибки. **printChances** и **printAlphabet** выводят информацию о шансах выпадения символа

# **3 Результаты работы приложения**

Для выполнения расчетов достаточно необходимо запустить приложение, выбрать необходимый пункт задания и ввести числа. Рисунки 3.1 - 3.8 показывают необходимые расчеты и вызовы методов, требуемые в данной лабораторной работе.

## **3.1 Энтропия немецкого и русского алфавитов в ASCII**

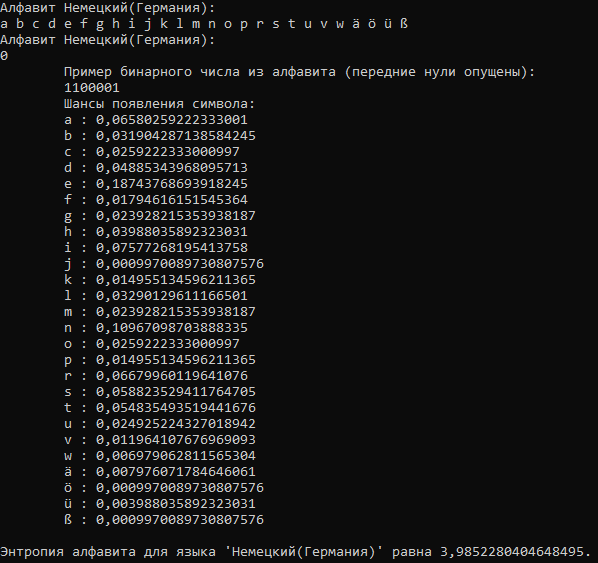


Рисунок 3.1 – Результат работы метода **computeTextEntropy** с для немецкого языка(ASCII)

В качестве тестового текста был использован текст задания к лабораторному практикуму. Используя формулу Шеннона, было найдено значение энтропии – 3,985.

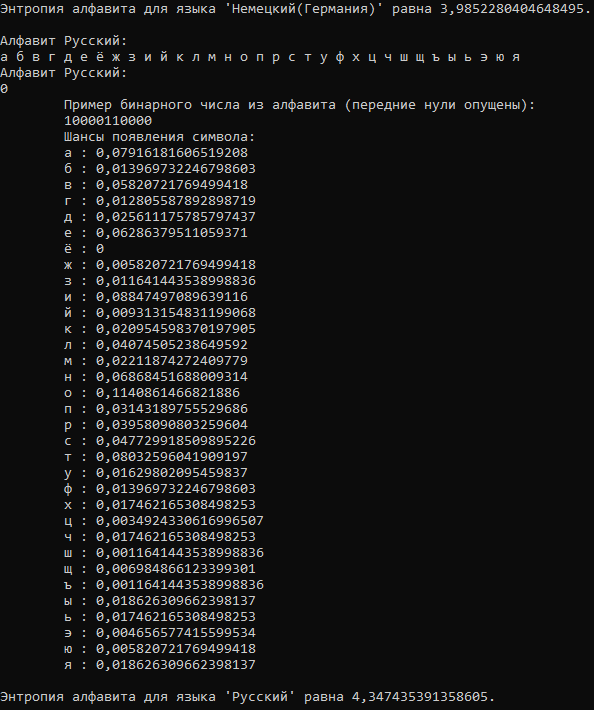


Рисунок 3.2 – Результат работы метода **computeTextEntropy** с для русского языка(ASCII)

В примере с русским языком значение энтропии по Шеннону – 4,347. Если энтропия равна 4,347, это означает, что среднее количество бит информации, необходимое для кодирования каждого символа из этого алфавита, равно 4,347. Чем выше энтропия, тем больше информации несет каждый символ, и тем сложнее предсказать следующий символ.

Энтропия 3,985 также указывает на степень неопределенности, но в этом случае энтропия меньше, чем в предыдущем примере. В целом, чем ниже энтропия, тем более предсказуемой является последовательность, и наоборот.

## **3.2 Энтропия немецкого и русского алфавитов в двоичном виде**

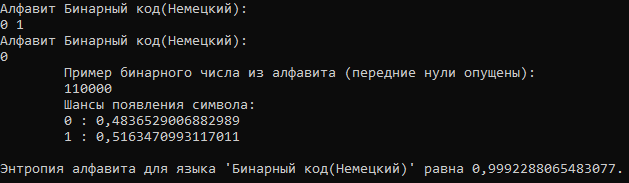


Рисунок 3.3 – Результат работы метода **computeTextEntropy** с для немецкого языка в двоичном виде

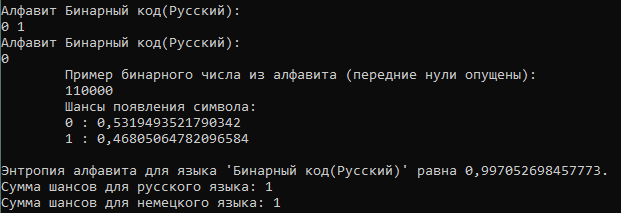


Рисунок 3.4 – Результат работы метода **computeTextEntropy** с для немецкого языка в двоичном виде

## **3.3 Количество информации с заданными значениями энтропии**

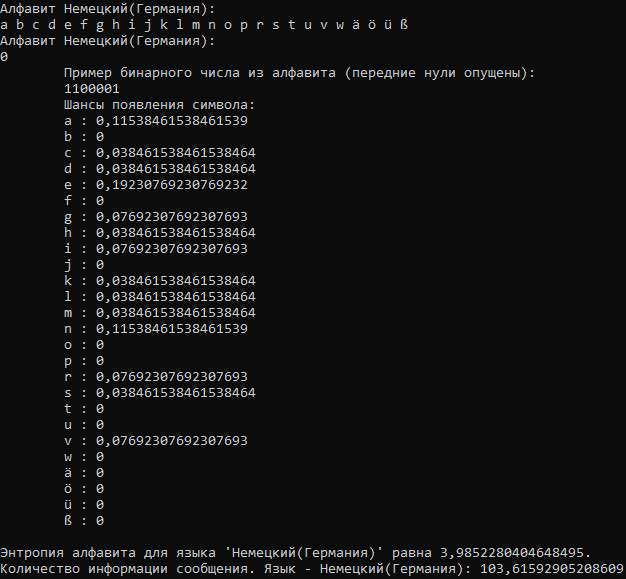


Рисунок 3.5 – Результат работы метода **computeTextEntropy** с для немецкого языка (3-й пункт)

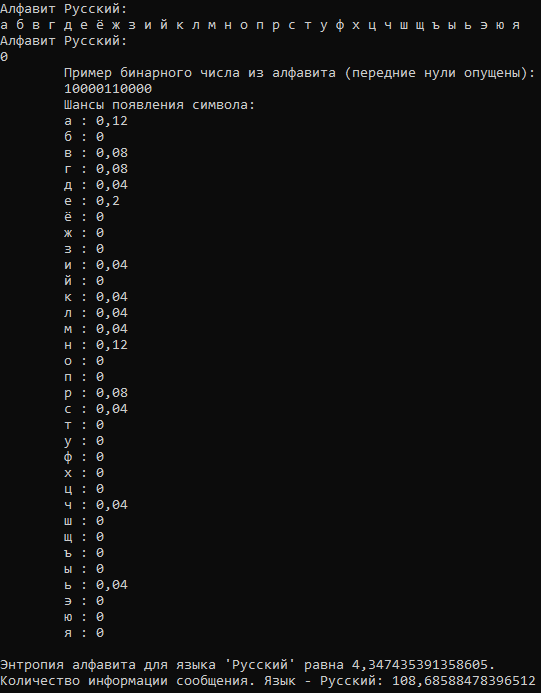


Рисунок 3.6 – Результат работы метода **computeTextEntropy** с для русского языка (3-й пункт)

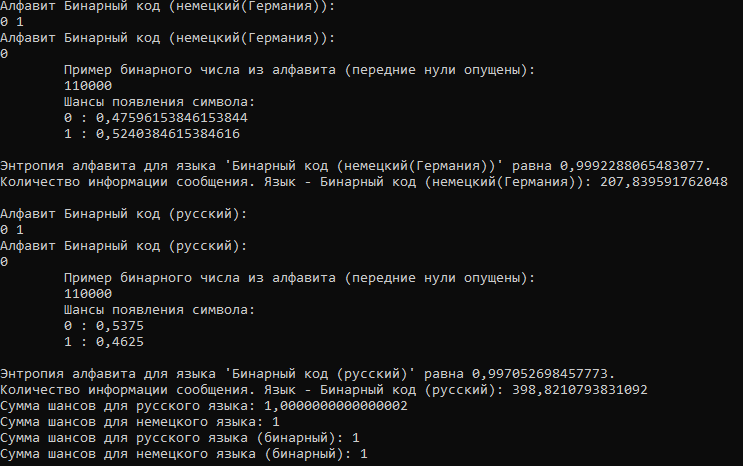


Рисунок 3.7 – Результат работы метода **computeTextEntropy** с для немецкого и русского языка в бинарном виде (3-й пункт)

В качестве значения энтропии для алфавитов были переданы значения из пунктов 1 и 2. Такие значения получаются исходя из формулы учебного пособия. То есть, путём перемножения значения энтропии алфавита и длины исходящего сообщения.

## **3.4 Энтропия при ошибке передачи**

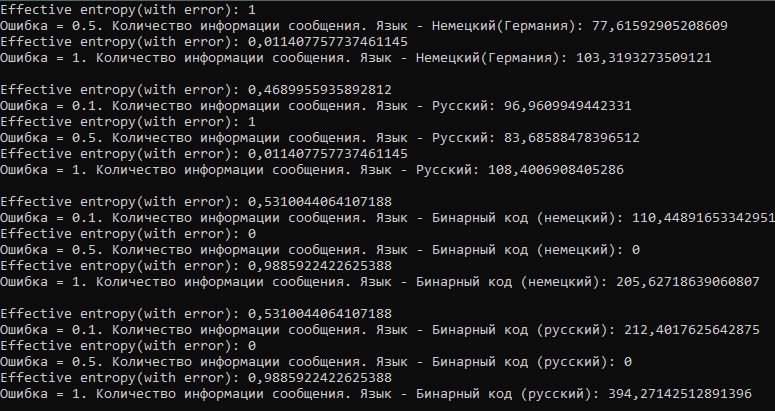


Рисунок 3.8 – Результат работы метода **computeTextEntropyWithError** для немецкого и русского языков в ASCII и бинарном виде

В данном пункте высчитываются значения в зависимости от ошибки передачи. Таким образом, эффективная энтропия считается не с помощью формулы Шеннона, а с помощью формул учебного пособия. Следовательно, длина сообщения зависит не частоты встречаемости символа, как в пунктах выше, а от вероятности ошибки, что и обуславливает такое разительное отличие в количестве информации от изначального сообщения

## **3.5 Гистограммы**

Рисунок 3.9 – Результат работы метода **addValuesFromDict** для русского языка в ASCII и бинарном виде

Рисунок 3.10 – Результат работы метода **addValuesFromDict** для русского языка в ASCII и бинарном виде

# **4 Вывод**

В ходе изучения теоретических материалов лабораторной работы и выполнения её практической части были изучены основы теории информации и анализа энтропии. Это консольное приложение, разработанное на языке программирования C#, предоставляет функциональности, направленные на измерение энтропии в различных сценариях, связанных с текстовыми данными.