Новосибирский государственный технический университет

Кафедра ТПИ

Теория информации и криптография

Лабораторная работа №2

**Основные методы побуквенного кодирования**

Выполнили: Преподаватели:

Яшков Иван Авдеенко Татьяна Владимировна

Гавриленко Андрей Кутузова Ирина Александровна

Группа:

ПМИ-13

Бригада 1

Новосибирск, 2023

1. **Цель работы:**

Освоить основные алгоритмы побуквенного кодирования.

1. **Задание:**

1. Реализовать приложение для кодирования с помощью заданного в варианте алгоритма – **Алгоритм Хаффмана**:

• вероятности появления символов алфавита должны храниться в одном файле, а последовательность, подлежащая кодированию, – в другом;

• закодированный текст должен сохраняться в файл;

• приложение должно:

а) выводить полученные кодовые слова для всех символов алфавита;

б) вычислять среднюю длину кодового слова;

в) вычислять избыточность;

г) проверять неравенство Крафта.

2. Реализовать приложение для декодирования с помощью заданного в варианте алгоритма - **Алгоритм Хаффмана**:

• вероятности появления символов алфавита должны храниться в одном файле, а закодированная последовательность – в другом;

• раскодированная последовательность должна сохраняться в файл.

3. Рассмотреть 3 распределения вероятностей символов алфавита: равномерное, P1(A) и P2(A). Для каждого распределения получить кодовые слова, вычислить среднюю длину кодового слова, избыточность и проверить неравенство Крафта.

4. С помощью реализованных приложений исследовать зависимость получаемых кодовых слов от распределения вероятностей символов алфавита:

• смоделировать последовательность, символы которой подчиняются равномерному распределению. Закодировать эту последовательность при равномерном, P1(A) и P2(A) распределениях и вычислить длину каждой из трёх закодированных последовательностей;

• смоделировать последовательность, символы которой подчиняются распределению

P1(A). Закодировать эту последовательность при равномерном, P1(A) и P2(A) распределениях и вычислить длину каждой из трёх закодированных последовательностей;

• смоделировать последовательность, символы которой подчиняются распределению P2(A). Закодировать эту последовательность при равномерном, P1(A) и P2(A) распределениях и вычислить длину каждой из трёх закодированных последовательностей;

• сделать выводы о связи распределения символов последовательности, вероятностях символов, используемых при кодировании, и длины получившегося кода.

1. **Описание метода решения заданий:**

При выполнении лабораторной работы было реализовано приложение для кодирования и декодирования с помощью алгоритма Хаффмана. Алфавит символов для кодирования и декодирования может задавать сам пользователь, размерность алфавита может быть любая, сами символы могут быть любыми. Пользователю доступны два режима работы: ввод с клавиатуры и ввод из файлов. Не важно, какой режим работы выбрал пользователь, в любом случае ему необходимо задать алфавит в следующем виде:

<Символ 1>[Пробел]<Вероятность символа 1>[Пробел]<Символ 2>[Пробел]<Вероятность символа 2>[Пробел]...[Пробел]<Символ n>[Пробел]<Вероятность символа n>

А также задать строку для кодирования, используя символы алфавита, или для декодирования, используя кодовые слова алфавита. Подробнее про интерфейс приложения будет сказано далее.

**Реализация алгоритма Хаффмана**

Алгоритм Хаффмана был реализован следующим способом:

Создается массив с вероятностями размерности 2N - 1, где N - количество элементов в алфавите. Первые N элементов в этом массиве - изначальные вероятности, заданные в алфавите (в индексах это от 0 до N - 1). В течение алгоритма вероятности будут складываться, полученные суммы будут записываться в следующие индексы (N, N + 1 и т. д.).

Далее необходимо создать бинарное дерево для работы алгоритма. Потребуется массив размерности 2N - 1, в котором будут храниться индексы складываемых вероятностей. По сути эта структура и будет являться деревом. Первые N элементов - индексы изначальных вероятностей (0, 1, … N - 1), последующие элементы - элементы вида "k:n", где k и n - индексы минимальных вероятностей из массива вероятностей. После объединения в массив вероятностей дописывается сумма вероятностей по индексам k и n. Эта сумма попадает в следующий, еще не занятый, индекс. Чтобы программа выдавала правильный результат, необходимо, чтобы после нахождения некоторых минимальных вероятностей, в следующей итерации эти же вероятности не считались как минимальные. Для этого существует массив типа bool размерности 2N - 1, в котором каждой использованной вероятности ставится в соответствие значение true. Таким образом составляется бинарное дерево.

Теперь для получения кодовых слов необходимо сделать обход бинарного дерева. Это происходит рекурсивно: сначала берется последний элемент массива бинарного дерева. Сумма его индексов равна единице, это "корень" бинарного дерева. Из записи "k:n" отдельно берутся элементы k и n, они являются левой и правой ветвями некоторого узла дерева.

Левая ветка:

Если k < N, значит k - "лист" дерева, конечный элемент данной ветки, следовательно с этой итерации обход завершен, к кодовому слову элемента с индексом k добавляется последний "0". Иначе k не является "листом", а является узлом, добавляем "0" к последовательности и продолжаем обход в этом направлении.

Правая ветка:

Если n < N, значит n - "лист" дерева, конечный элемент данной ветки, следовательно с этой итерации обход завершен, к кодовому слову элемента с индексом n добавляется последний "1". Иначе n не является "листом", а является узлом, добавляем "1" к последовательности и продолжаем обход в этом направлении.

Таким образом получаются кодовые слова для каждого символа алфавита.

**Кодировка и декодировка**

Зная кодовые слова легко выполнять процессы кодировки и декодировки. Для кодировки необходимо для посимвольно обрабатывать сообщение, если тот или иной символ существует в алфавите, то записывать соответствующее кодовое слово в выходную строку, иначе сообщать об ошибке.

Для декодировки необходимо поэлементно добавлять к некоторой строке значения из сообщения, если существует такое кодовое слово, то соответствующий символ алфавита записывается в выходную строку. Если в течение обработки декодируемого сообщения длина некоторой строки превысила максимальную длину кодового слова, то значит сообщение для декодирования записано неверно. Также если после декодирования осталось некоторая необработанная строка, значит сообщение также записано неверно.

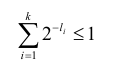
**Вычисление характеристик**

После того как были получены все кодовые слова, программа уже может однозначно кодировать и декодировать валидные сообщения. После этого можно вычислить некоторые характеристики:

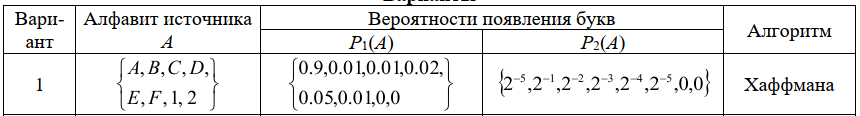
a) Среднюю длину кодовых слов. Для ее вычисления вычисляется сумма произведений длины кодового слова, соответствующего каждому символу, и вероятности появления этого символа в последовательности.

б) Избыточность. Разность единицы и отношения энтропии текущего ансамбля и ансамбля с равномерными вероятностями появления каждого символа.

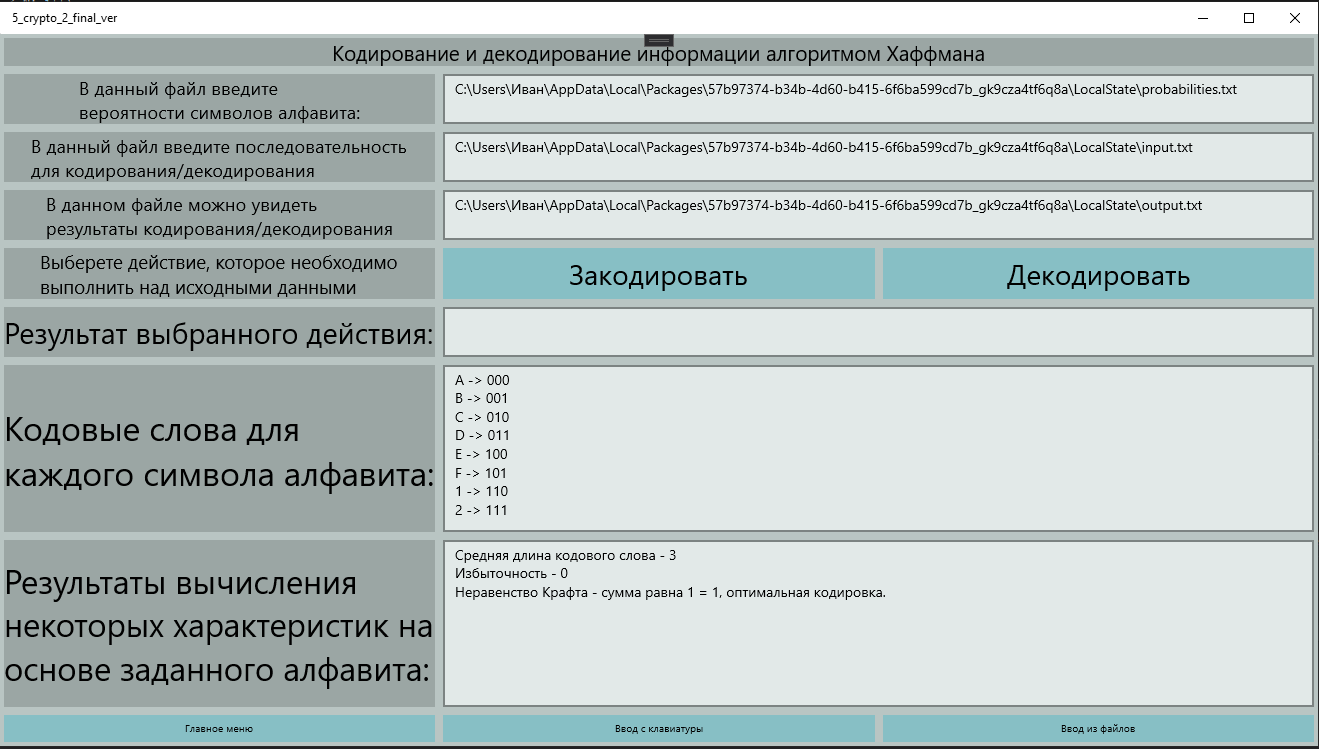
в) Проверять неравенство Крафта.

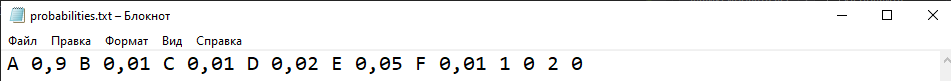
, где k – это количество символов алфавита, а li – длина кодового слова, соответствующего i-у символу. Если левая часть равна 1, то код считается оптимальным.

**Получение характеристик для трех сигнатур при неизменном алфавите**

****

Равномерное:

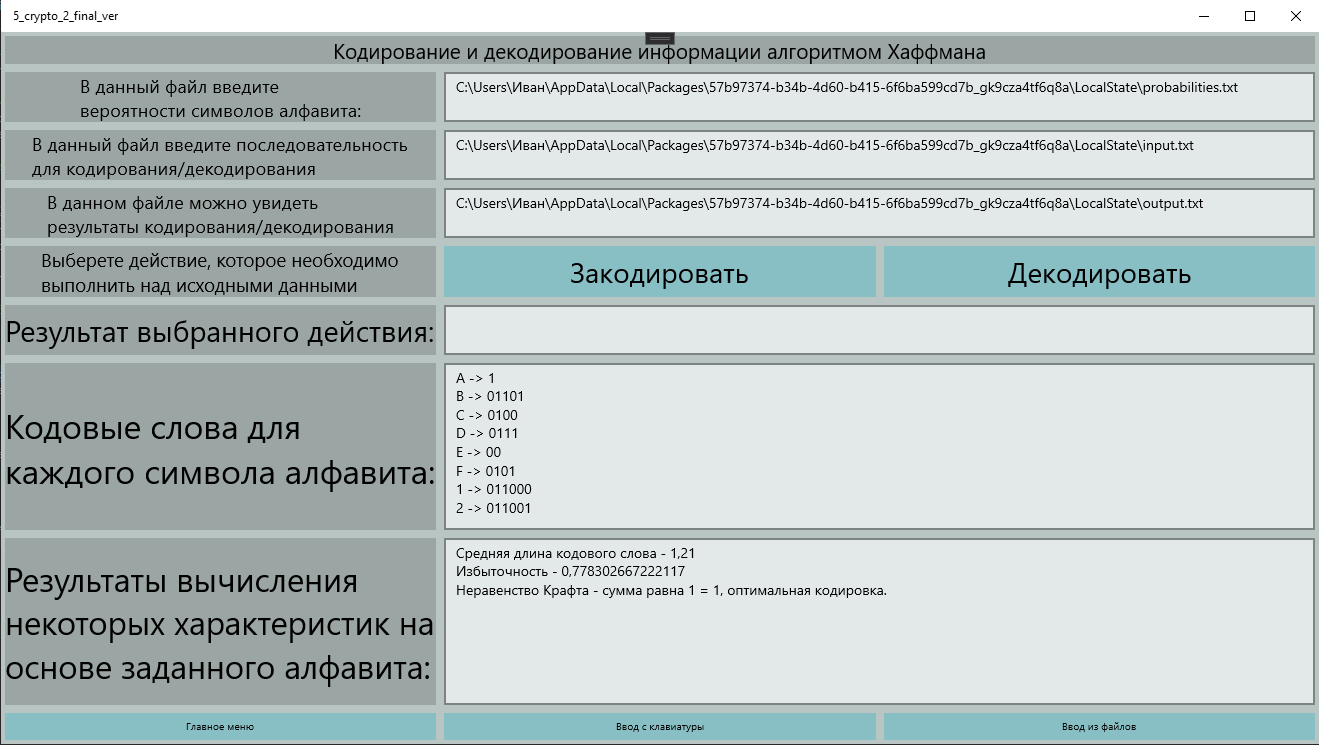


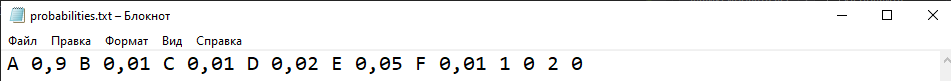


P(xi) = 0.125, i = .

Средняя длина кодового слова совпадает с длиной каждого кодового слова. При этом избыточность равна нулю, что соответствует равномерному распределению, а левая часть неравенства Крафта равна 1, что означает оптимальность кода.

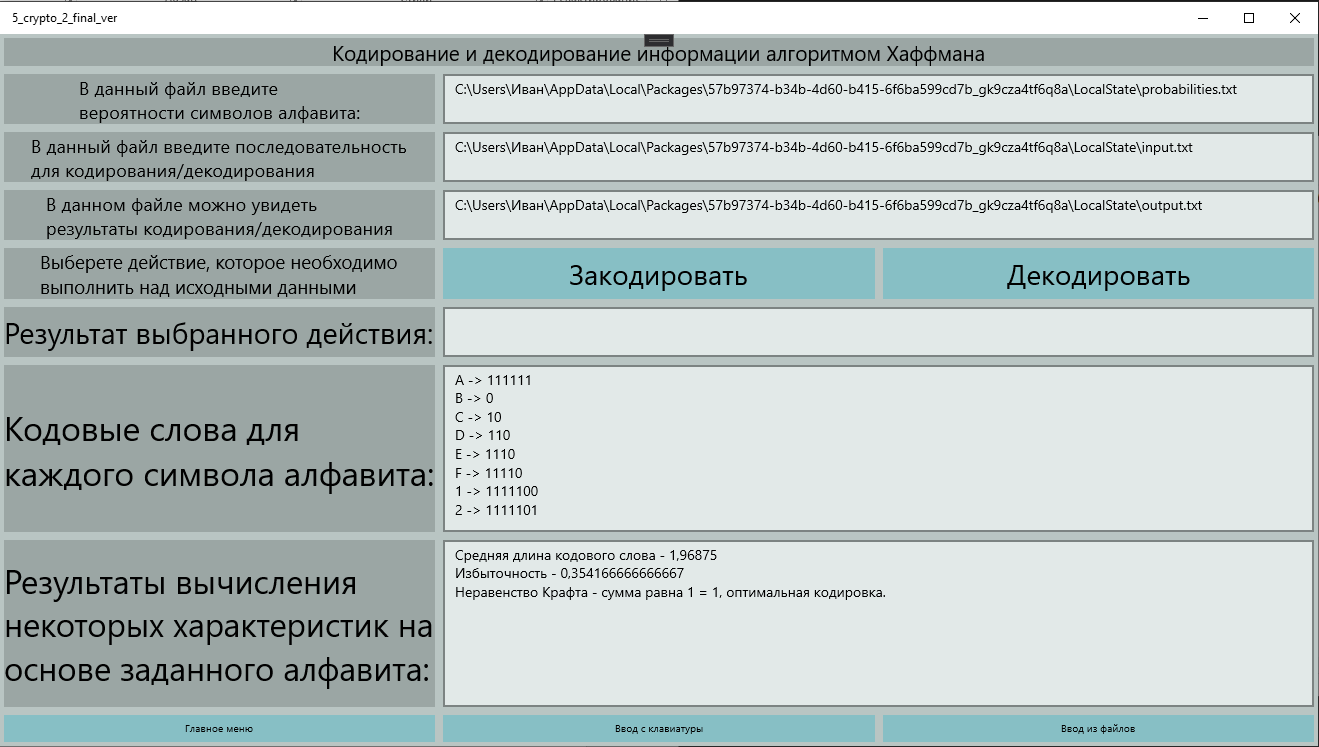
P1(A):

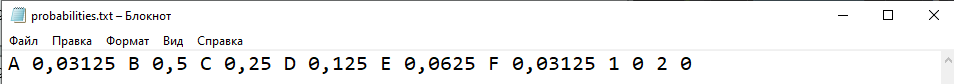




Средняя длина кодового слова 1.21, избыточность равна 0,778302667222117, неравенство Крафта выполняется, код оптимальный.

P2(A):





Средняя длина кодового слова 1,96875, избыточность равна 0,354166666666667, неравенство Крафта выполняется, код оптимальный.

**Моделирование последовательностей:**

*a) Последовательность, подчиняющаяся равномерному распределению:*

ABCDEF1221FEDCABABCDEF1221FEDCABABCDEF1221FEDCABABCDEF1221FEDCABABCDEF1221FEDCABABCDEF1221FEDCABABCDEF1221FEDCABABCDEF1221FEDCAB – 128 символов

Кодирование при равномерном распределении:

000001010011100101110111111110101100011010000001000001010011100101110111111110101100011010000001000001010011100101110111111110101100011010000001000001010011100101110111111110101100011010000001000001010011100101110111111110101100011010000001000001010011100101110111111110101100011010000001000001010011100101110111111110101100011010000001000001010011100101110111111110101100011010000001 – 384 символа

P1(A):

10110101000111000101011000011001011001011000010100011101001011011011010100011100010101100001100101100101100001010001110100101101101101010001110001010110000110010110010110000101000111010010110110110101000111000101011000011001011001011000010100011101001011011011010100011100010101100001100101100101100001010001110100101101101101010001110001010110000110010110010110000101000111010010110110110101000111000101011000011001011001011000010100011101001011011011010100011100010101100001100101100101100001010001110100101101 – 512 символов

P2(A): 11111101011011101111011111001111101111110111111001111011101101011111101111110101101110111101111100111110111111011111100111101110110101111110111111010110111011110111110011111011111101111110011110111011010111111011111101011011101111011111001111101111110111111001111011101101011111101111110101101110111101111100111110111111011111100111101110110101111110111111010110111011110111110011111011111101111110011110111011010111111011111101011011101111011111001111101111110111111001111011101101011111101111110101101110111101111100111110111111011111100111101110110101111110 – 560 символов

*б) Последовательность, подчиняющаяся распределению P1(A):*

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABCAAAAADAAAAAAAAAAAAAAEEDAAAAAEFAAAEE – 100 символов

Кодирование при равномерном распределении:

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001010000000000000000011000000000000000000000000000000000000000000100100011000000000000000100101000000000100100 – 300 символов

P1(A):

1111111111111111111111111111111111111111111111111111111111111110110101001111101111111111111111100000111111110001011110000 – 121 символ

P2(A):

1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111110101111111111111111111111111111111101111111111111111111111111111111111111111111111111111111111111111111111111111111111111110111011011111111111111111111111111111111101111011111111111111111111101110 – 485 символов

*в) Последовательность, подчиняющаяся распределению P2(A):*

BBBBBBBBBBBBDBBBBBDBBBBBBBBBBDBBBBBDBBDBBBBBBDBBBBDBBBBBBACCDCCCCCCCCCCDCCCCDCCCCCDCDCCCCCCDCEEEEDEEDEEDACCBBBBBBBBAFABBBCFFFBBB – 128 символов

Кодирование при равномерном распределении:

001001001001001001001001001001001001011001001001001001011001001001001001001001001001001011001001001001001011001001011001001001001001001011001001001001011001001001001001001000010010011010010010010010010010010010010011010010010010011010010010010010011010011010010010010010010011010100100100100011100100011100100011000010010001001001001001001001001000101000001001001010101101101001001001 – 384 символа

P1(A):

01101011010110101101011010110101101011010110101101011010110101110110101101011010110101101011101101011010110101101011010110101101011010110101101011101101011010110101101011010111011010110101110110101101011010110101101011010111011010110101101011010111011010110101101011010110101101101000100011101000100010001000100010001000100010001000111010001000100010001110100010001000100010001110100011101000100010001000100010001110100000000000111000001110000011110100010001101011010110101101011010110101101011011010110110101101011010100010101010101011010110101101 – 548 символов

P2(A):

000000000000110000001100000000000110000001100011000000011000001100000001111111010110101010101010101010101101010101011010101010101101011010101010101011010111011101110111011011101110110111011101101111111010000000001111111111011111100010111101111011110000 – 252 символа

Количество символов, необходимых для кодирования последовательности, подчиняющейся любому распределению, с помощью равномерного распределения получилось равно произведению средней длины кодового слова на количество символов в последовательности. То есть при подчинении последовательности любому распределению, количество символов в закодированной последовательности будет прямо зависеть только от количества символов.

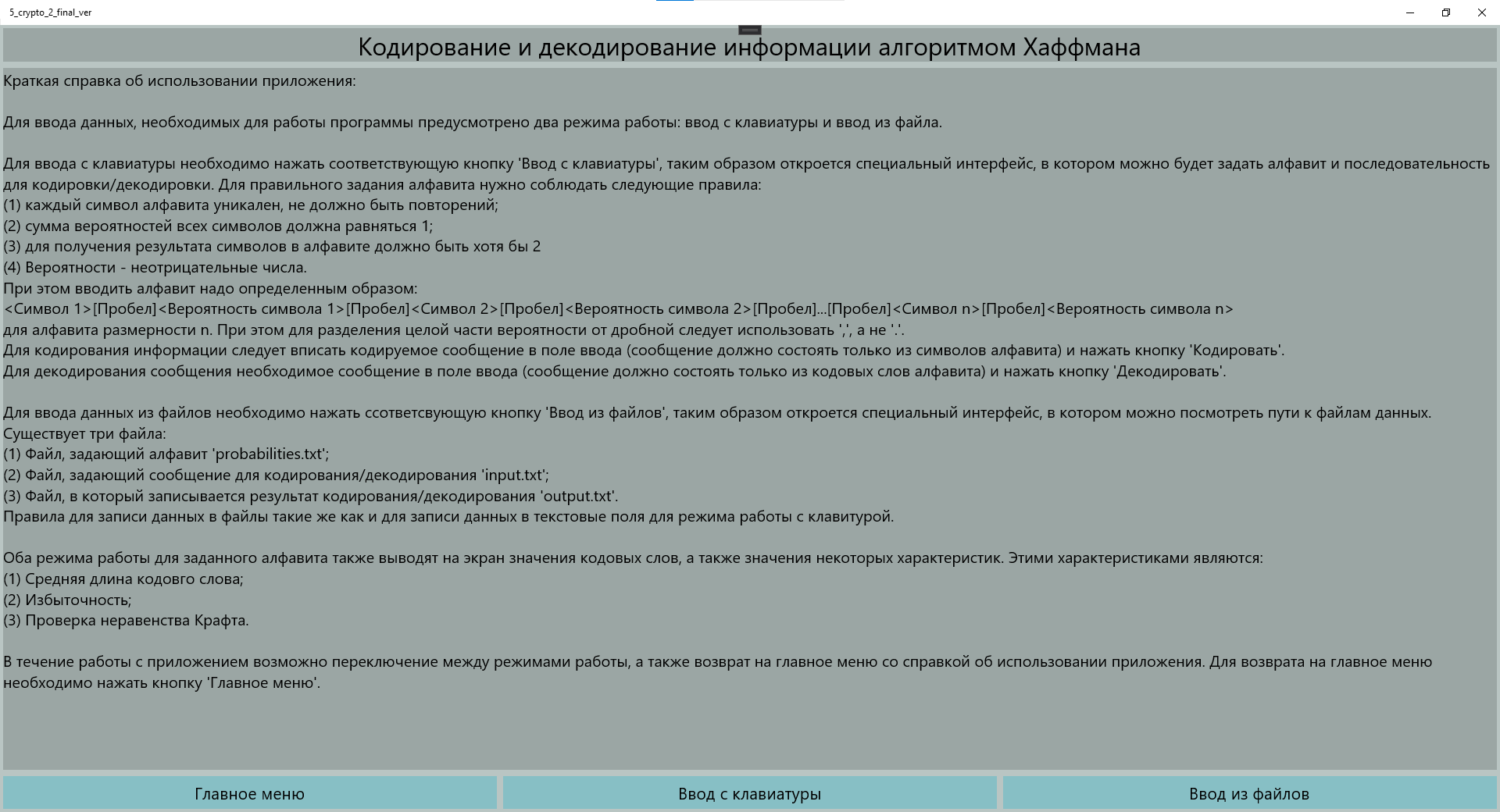
При кодировании последовательности, подчиняющейся равномерному распределению с помощью двух других распределений были получены одинаковые результаты (превышающие результаты кодирования с помощью равномерного распределения). Так как для каждого символа из одного распределения нашелся символ из другого распределения, чьи длины кодовых слов равны.

При кодировании последовательностей, подчиняющихся распределениям P1(A) и P2(A), наименьшее количество символов в закодированных последовательностях получилось при кодировке в соответствии с их заданным распределением.

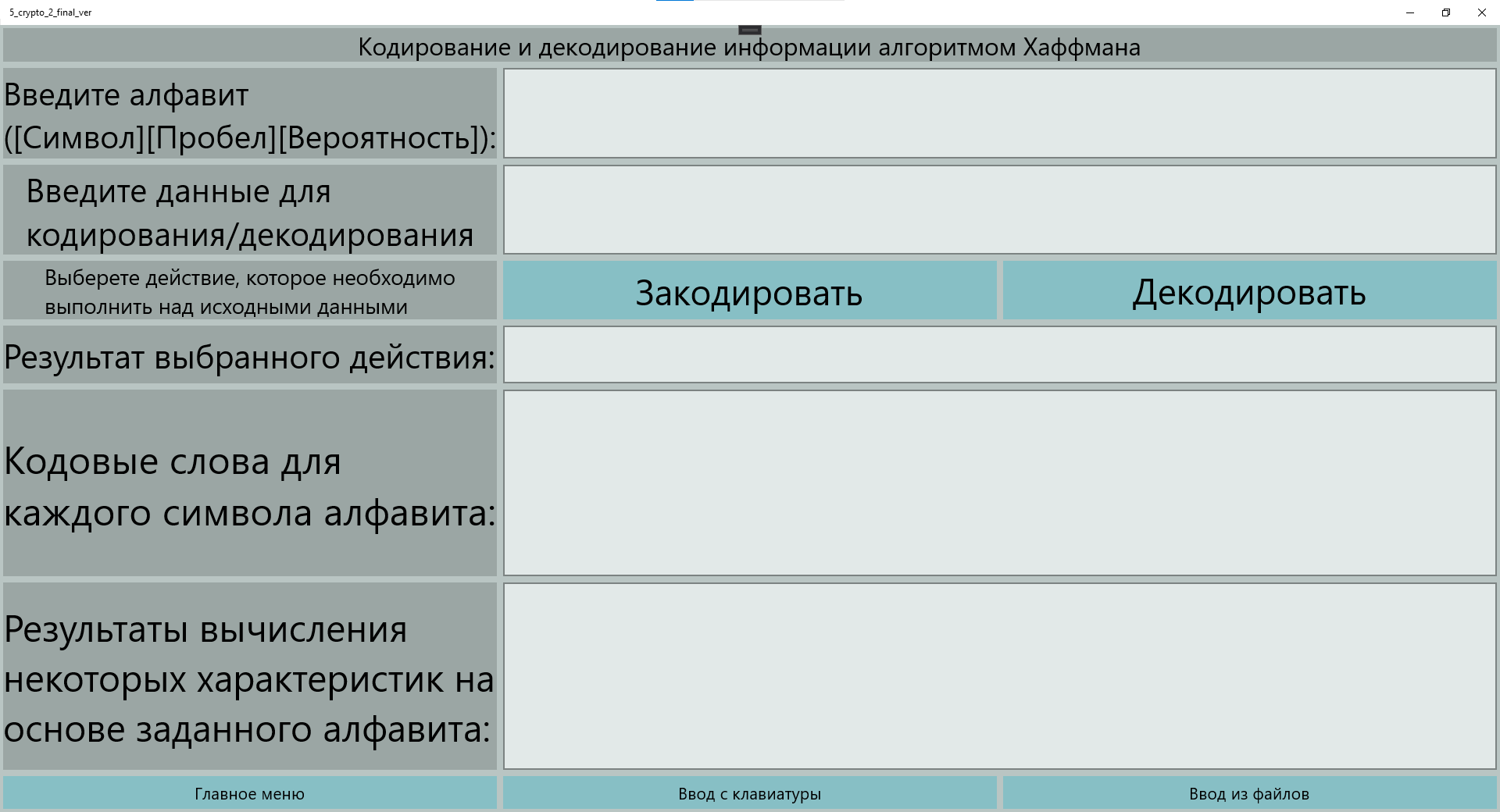
При кодировке последовательности с помощью того распределения, в соответствии с которым она была задана, результирующая длина закодированной последовательности получилась равна произведению средней длины кодового слова сигнатуры на количество символов кодируемой последовательности.

**Описание разработанного программного средства**

При запуске приложения Пользователя встречает главное меню с названием, справкой об использовании приложения и тремя кнопками "Главное меню", "Ввод с клавиатуры", "Ввод с файла":

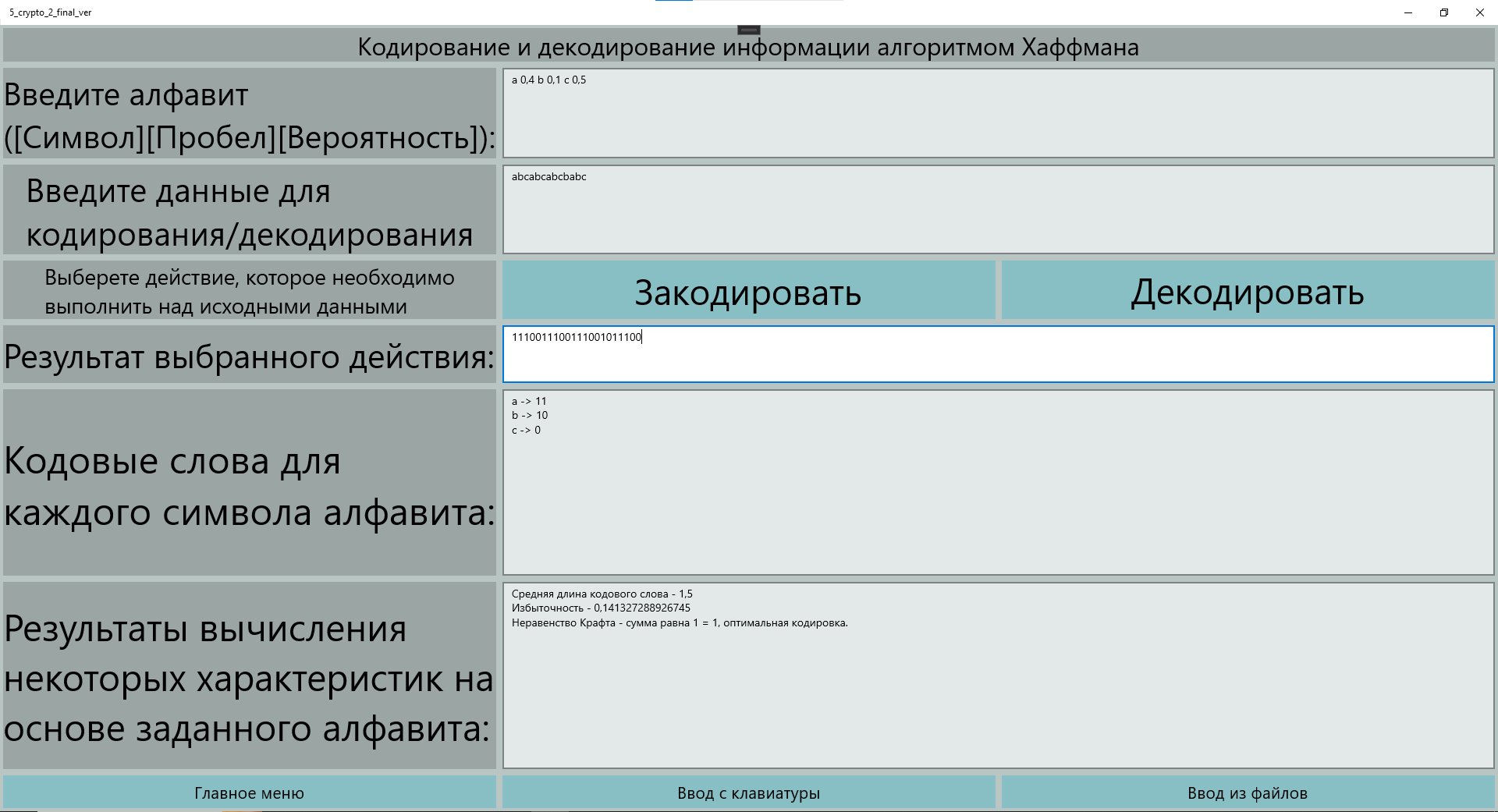


**При нажатии кнопки "Ввод с клавиатуры" Пользователь попадает в следующее меню:**

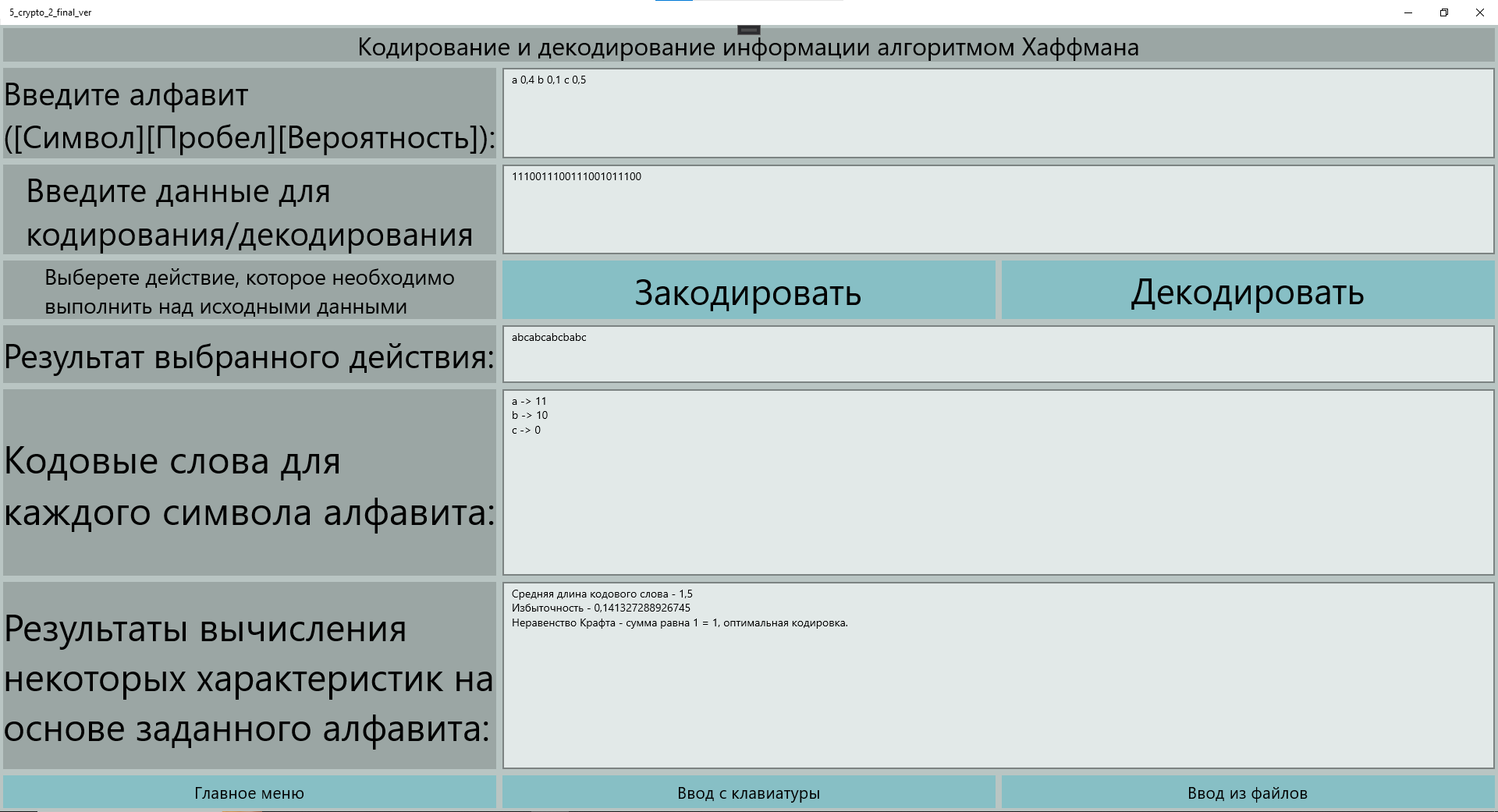


Здесь Пользователь может ввести алфавит и данные для кодирования/декодирования вручную и, при нажатии одной из кнопок, получить результат кодирования/декодирования, кодовые слова и характеристики алфавита:

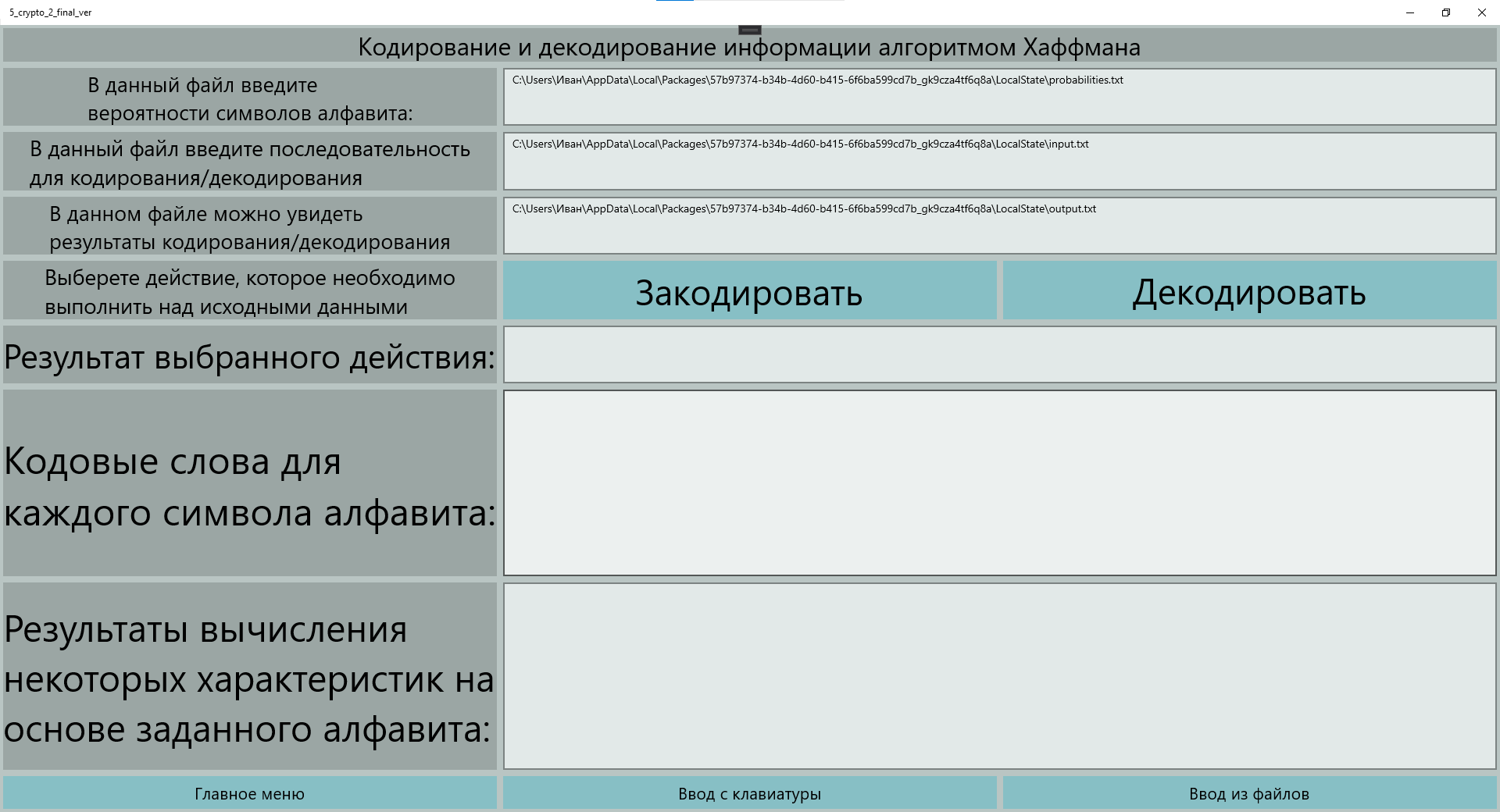
**Кодирование:**



**Декодирование:**

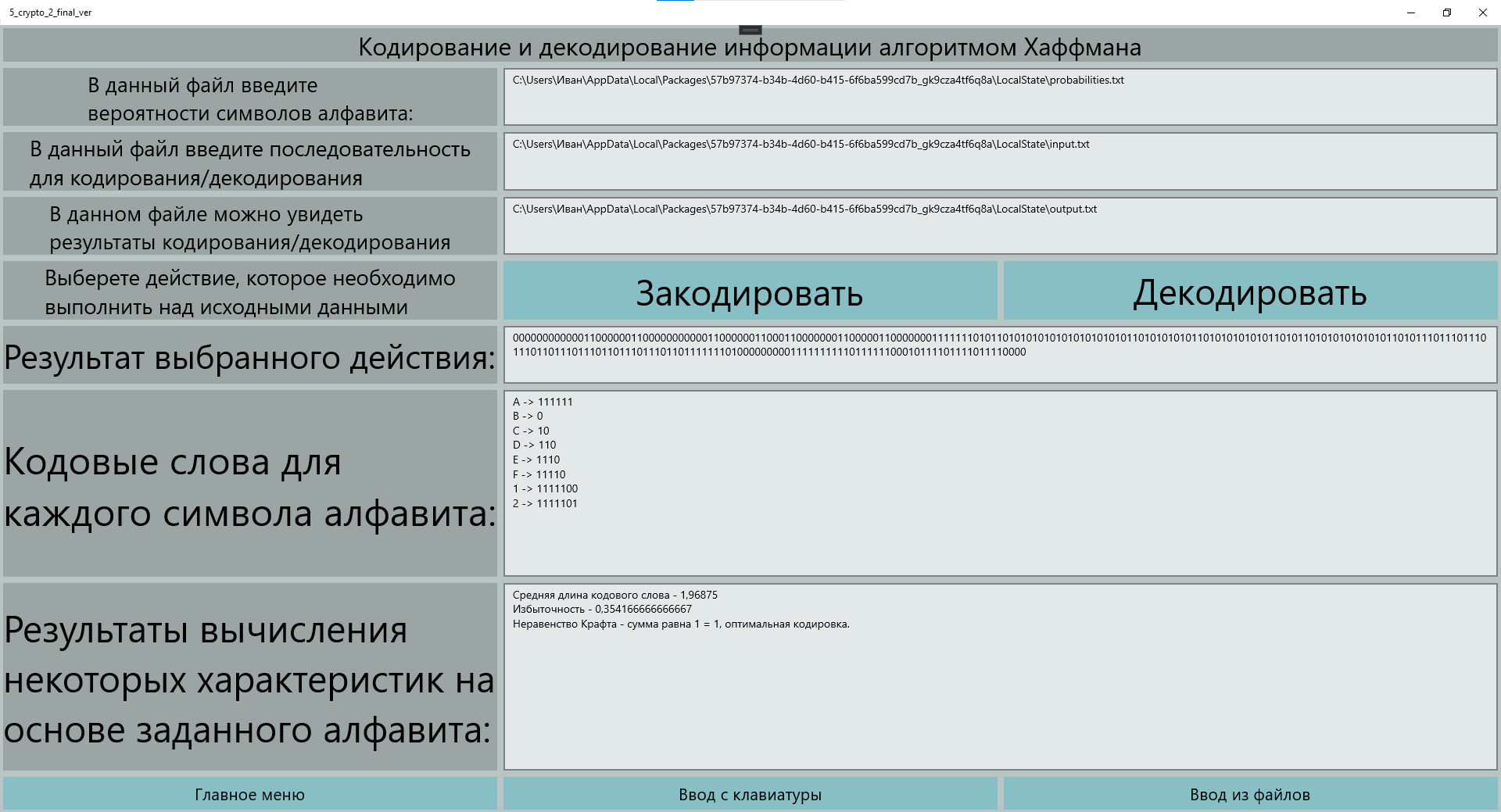


**При нажатии кнопки "Ввод из файлов" Пользователь попадает в следующее меню:**

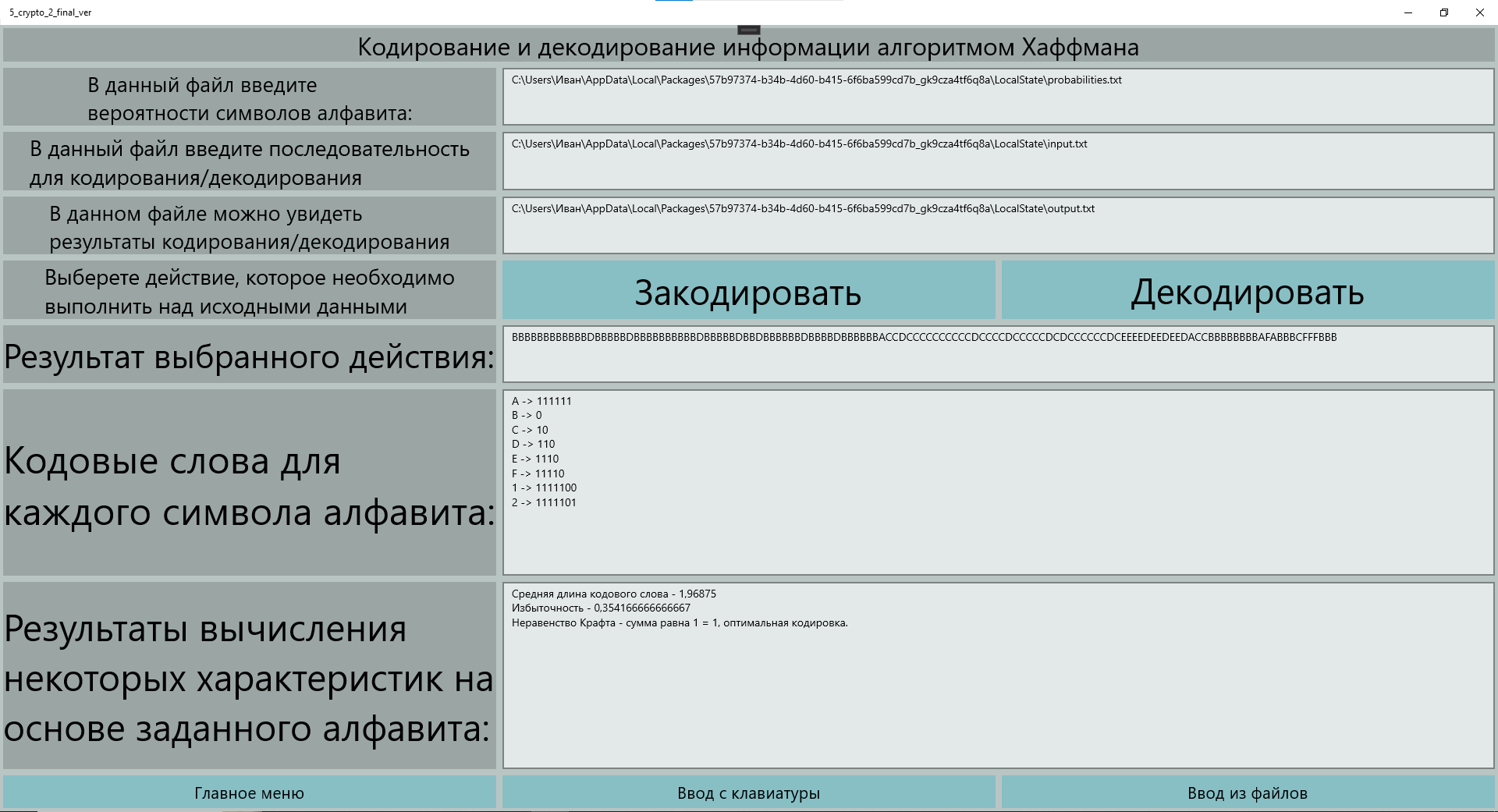
****

В первых трех полях Пользователь может увидеть пути к файлам, в первый из которых необходимо поместить информацию об алфавите, во второй - сообщение для кодирования/декодирования, а в третьем можно прочитать результат кодирования/декодирования. Также, как и в случае с режимом "Ввода с клавиатуры", в режиме "Ввода из файлов" после кодирования/декодирования на экран выводятся кодовые слова и характеристики:

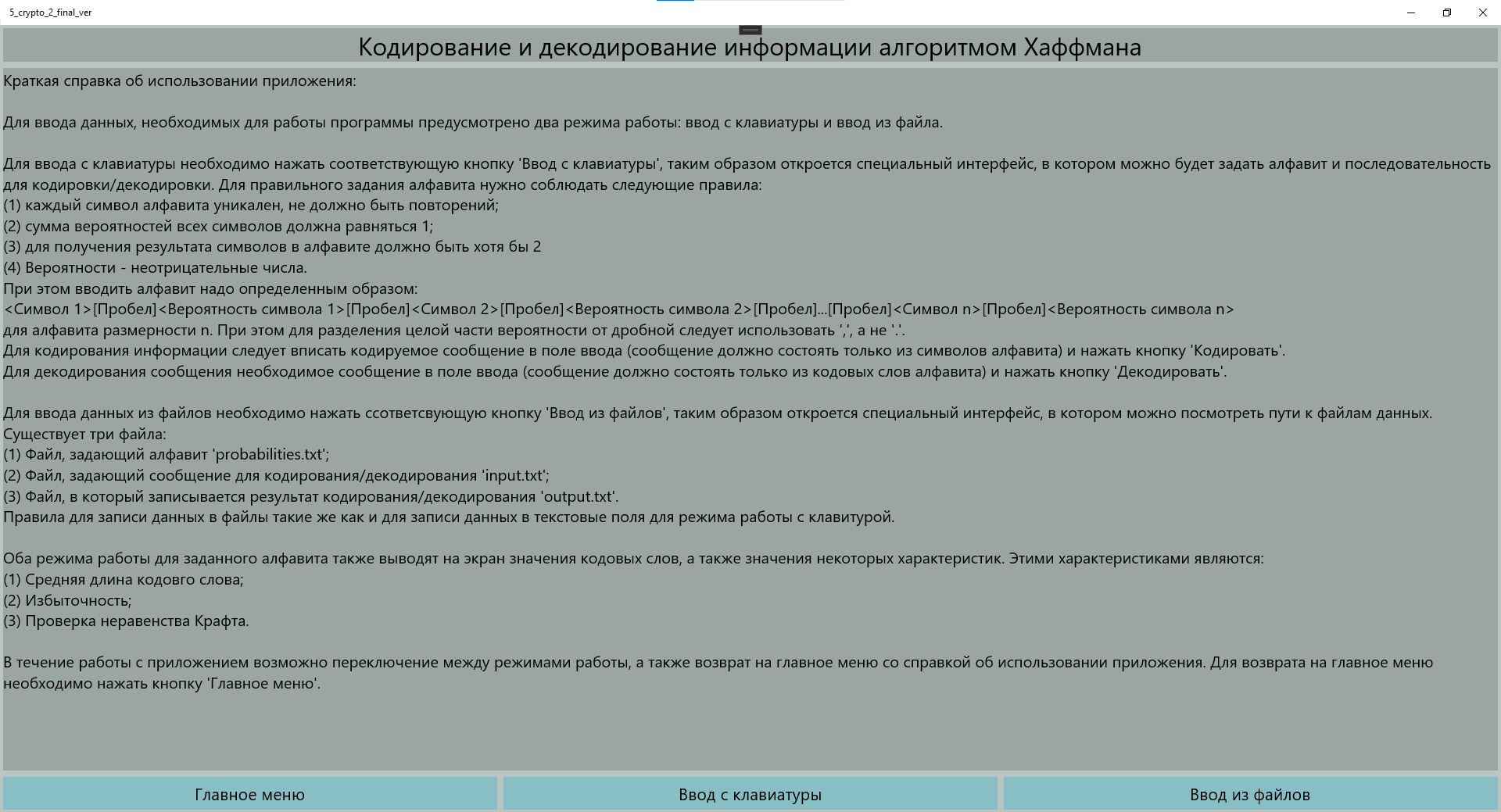
**Кодирование:**



**Декодирование:**



**При нажатии кнопки "Главное меню" Пользователь попадает в главное меню:**

****

**Программный код, написанный непосредственно студентами:**

**Функционал программы написан на языке C#.**

**Модуль MainPage.xaml.cs, содержащий класс, реализующий создание и обработку бинарного дерева, кодирование и декодирование, нахождение характеристик:**

using System;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

namespace \_5\_crypto\_2\_final\_ver

{

public sealed partial class MainPage : Page

{

public MainPage()

{

this.InitializeComponent();

frame.Content = new MainMenu();

}

private void GoToMainMenu(object sender, RoutedEventArgs e)

{

frame.Content = new MainMenu();

}

private void GoToKeyboardInput(object sender, RoutedEventArgs e)

{

frame.Content = new KeyboardInput();

}

private void GoToFileInput(object sender, RoutedEventArgs e)

{

frame.Content = new FileInput();

}

}

class StartParameters

{

public readonly int N;

public readonly double[] probabilities;

public readonly string[] names;

public readonly string[] code\_words;

private readonly string[] indices;

public readonly int max\_code\_word\_length;

public readonly double average\_length;

public readonly double redundancy;

public readonly double KraftInequality;

public StartParameters(string probs)

{

string[] parameters;

int counter = 0;

double temp = 0;

int min1, min2;

bool[] used;

string text = probs;

//Проверка валидности текста, задающего алфавит

if (text == null || !text.Contains(" ")) { throw new Exception("Неправильная запись алфавита. Каждый символ должна сопровождаться вероятностью её появления, между символами и вероятностями должны стоять пробелы"); }

//Массив элементов алфавита

parameters = text.Split(' ');

//Если элементов в алфавите нечетное число, значит пропущен какой-либо элемент

N = parameters.Length;

if (N % 2 != 0) { throw new Exception("Элементов должно быть четное число, нечетные элементы - буквы алфавита, четные элементы - соответствующие буквам вероятности. Скорее всего элементов не хватает."); }

//Необходимо убедиться, что символов в алфавите хотя бы 2

if (N < 4) { throw new Exception("Символов введено слишком мало. Введите хотя бы два символа и повторите попытку."); }

N /= 2;

names = new string[N];

code\_words = new string[N];

probabilities = new double[N + N];

probabilities[N + N - 1] = 2;

foreach (string p in parameters)

{

//Проверка символов алфавита

if (counter % 2 == 0)

{

if (Array.Exists(names, element => element == p)) { throw new Exception("Алфавит должен состоять из неповторяющихся элементов. Элемент \"" + p + "\" повторяется."); }

names[counter / 2] = p;

}

//Проверка вероятностей алфавита

else

{

try { probabilities[(counter - 1) / 2] = Convert.ToDouble(p); }

catch { throw new Exception("Вероятность \"" + p + "\" не является числом. Возможно следует поменять \".\" на \",\"."); }

if (probabilities[(counter - 1) / 2] < 0) { throw new Exception("Вероятность \"" + p + "\" меньше 0, а должна быть больше или равна 0."); }

temp += probabilities[(counter - 1) / 2];

}

counter++;

}

//Проверка равенства суммы вероятностей и единицы

if (!(temp + Math.Pow(10, -10) >= 1 && temp - Math.Pow(10, -10) <= 1)) { throw new Exception("Сумма всех вероятностей должна быть равна 1. Сейчас она равна " + temp + "."); }

indices = new string[N + N - 1];

used = new bool[N + N - 1];

for (int i = 0; i < N; i++) { indices[i] = Convert.ToString(i); }

//Создание бинарного дерева из имеющегося алфавита

for (int i = N; i < 2 \* N - 1; i++)

{

min1 = N + N - 1;

min2 = N + N - 1;

//Поиск двух самых малых вероятностей

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

{

if (probabilities[j] < probabilities[min1] && j != min1 && !used[j])

{

min2 = min1;

min1 = j;

}

else if (probabilities[j] < probabilities[min2] && j != min1 && !used[j]) { min2 = j; }

}

//Объединение самых малых вероятностей в новую вероятность

probabilities[i] = probabilities[min1] + probabilities[min2];

indices[i] = Convert.ToString(min1) + ":" + Convert.ToString(min2);

//Найденные вероятности больше не учитываются при поиске следующих самых малых вероятностей

used[min1] = true;

used[min2] = true;

}

//Обход бинарного дерева с целью получения кодовых слов

FindCodeWords(N + N - 2, "");

//Подсчет характеристик

max\_code\_word\_length = 0;

KraftInequality = 0;

temp = 0;

//Получение средней длины и суммы в правой части неравенства Крафта

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

{

temp += code\_words[i].Length \* probabilities[i];

KraftInequality += Math.Pow(2, -code\_words[i].Length);

if (code\_words[i].Length > max\_code\_word\_length) { max\_code\_word\_length = code\_words[i].Length; }

}

average\_length = temp;

//Поиск избыточности

redundancy = 0;

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

{

if (probabilities[i] == 0) { continue; }

redundancy += probabilities[i] \* Math.Log10(probabilities[i]) / Math.Log10(2);

}

redundancy /= Math.Log10(N) / Math.Log10(2);

redundancy += 1;

if (redundancy < Math.Pow(10, -10)) { redundancy = 0; }

}

private void FindCodeWords(int i, string current\_code\_word)

{

string[] halfs;

halfs = indices[i].Split(':');

//Если левый элемент является "листом" дерева, то к последовательности добавляется еще один "0" и таким образом получается кодовое слово, по индексу совпадающее с индексом "листа"

if (Convert.ToInt32(halfs[0]) < N) { code\_words[Convert.ToInt32(halfs[0])] = current\_code\_word + "0"; }

//Иначе к последовательности добавляется "0"

else { FindCodeWords(Convert.ToInt32(halfs[0]), current\_code\_word + "0"); }

//Если правый элемент является "листом" дерева, то к последовательности добавляется еще одна "1" и таким образом получается кодовое слово, по индексу совпадающее с индексом "листа"

if (Convert.ToInt32(halfs[1]) < N) { code\_words[Convert.ToInt32(halfs[1])] = current\_code\_word + "1"; }

//Иначе к последовательности добавляется "1"

else { FindCodeWords(Convert.ToInt32(halfs[1]), current\_code\_word + "1"); }

}

public string CodeMessage(string input)

{

//Кодирование сообщения

string text, output = "";

bool isElementConsists;

//Проверка на пустоту строки с сообщением для кодирования

text = input;

if (text == null) { throw new Exception("Вы не ввели строку для кодирования."); }

//Для каждого символа сообщения проверяется, существует ли он в заданном алфавите, если да, то в выходное сообщение записывается соответсвующее кодовое слово, иначе вызывается исключение

foreach (char letter in text)

{

isElementConsists = false;

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

{

if (names[i] == Convert.ToString(letter))

{

isElementConsists = true;

output += code\_words[i];

break;

}

}

if (!isElementConsists) { throw new Exception("Элемент " + Convert.ToString(letter) + " не существует в алфавите."); }

}

return output;

}

public string DecodeMessage(string input)

{

//Декодирование сообщения

string text, code\_word = "", output = "";

//Проверка на пустоту строки с сообщением для кодирования

text = input;

if (text == null) { throw new Exception("Вы не ввели строку для декодирования."); }

foreach (char letter in text)

{

code\_word += Convert.ToString(letter);

//Поэлементно добавляем к некоторой строке значения из сообщения, если существует такое кодовое слово, то соответствующий символ алфавита записывается в выходную строку

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

{

if (code\_word == code\_words[i])

{

output += names[i];

code\_word = "";

break;

}

}

//Если некоторая строка превышает максимальную длину кодового слова, значит сообщение для декодирования заисано неверно

if (code\_word.Length == max\_code\_word\_length) { throw new Exception("Текст невозможно декодировать, проверьте правильность ввода и повторите попытку."); }

}

//Если что-то осталось после попытки декодирования, значит сообщение для декодирования записано неверно

if (code\_word != "") { throw new Exception("Текст невозможно декодировать, проверьте правильность ввода и повторите попытку."); }

return output;

}

}

}

**Модуль KeyboardInput.xaml.cs, реализующий инициализацию основного класса с помощью полей для ввода в программе:**

using System;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

using Windows.UI.Popups;

namespace \_5\_crypto\_2\_final\_ver

{

public sealed partial class KeyboardInput : Page

{

public KeyboardInput()

{

this.InitializeComponent();

}

private void FirstChooseButton\_Click(object sender, RoutedEventArgs e)

{

//Кодирование информации

string output;

try

{

//Инициализация класса с использованием текстового поля с алфавитом

StartParameters sp = new StartParameters(EnterProbsTextBox.Text);

//Кодирование информации из текстового поля

output = sp.CodeMessage(EnterInputTextBox.Text);

//Вывод результата на экран

ResultTextBox.Text = output;

//Вывод кодовых слов на экран

CodeWordsTextBox.Text = "";

for (int i = 0; i < sp.N; i++)

{

CodeWordsTextBox.Text += sp.names[i] + " -> " + sp.code\_words[i] + Environment.NewLine;

}

//Вывод характеристик на экран

CharacteristicsTextBox.Text = "Средняя длина кодового слова - " + sp.average\_length + Environment.NewLine;

CharacteristicsTextBox.Text += "Избыточность - " + sp.redundancy + Environment.NewLine;

CharacteristicsTextBox.Text += "Неравенство Крафта - сумма равна " + sp.KraftInequality + " ";

if (sp.KraftInequality < 1) { CharacteristicsTextBox.Text += "< 1, условие выполняется."; }

else if (sp.KraftInequality == 1) { CharacteristicsTextBox.Text += "= 1, оптимальная кодировка."; }

else { CharacteristicsTextBox.Text += "> 1, условие не выполняется."; }

}

catch (Exception exc)

{

//Если что-то пошло не так, на экран выводится сообщение с ошибкой

MessageDialog message = new MessageDialog(exc.Message);

message.ShowAsync().AsTask();

}

}

private void SecondChooseButton\_Click(object sender, RoutedEventArgs e)

{

string output;

try

{

//Инициализация класса с использованием текстового поля с алфавитом

StartParameters sp = new StartParameters(EnterProbsTextBox.Text);

//Декодирование информации из текстового поля

output = sp.DecodeMessage(EnterInputTextBox.Text);

//Вывод результата на экран

ResultTextBox.Text = output;

//Вывод кодовых слов на экран

CodeWordsTextBox.Text = "";

for (int i = 0; i < sp.N; i++)

{

CodeWordsTextBox.Text += sp.names[i] + " -> " + sp.code\_words[i] + Environment.NewLine;

}

//Вывод характеристик на экран

CharacteristicsTextBox.Text = "Средняя длина кодового слова - " + sp.average\_length + Environment.NewLine;

CharacteristicsTextBox.Text += "Избыточность - " + sp.redundancy + Environment.NewLine;

CharacteristicsTextBox.Text += "Неравенство Крафта - сумма равна " + sp.KraftInequality + " ";

if (sp.KraftInequality < 1) { CharacteristicsTextBox.Text += "< 1, условие выполняется."; }

else if (sp.KraftInequality == 1) { CharacteristicsTextBox.Text += "= 1, оптимальная кодировка."; }

else { CharacteristicsTextBox.Text += "> 1, условие не выполняется."; }

}

catch (Exception exc)

{

//Если что-то пошло не так, на экран выводится сообщение с ошибкой

MessageDialog message = new MessageDialog(exc.Message);

message.ShowAsync().AsTask();

}

}

}

}

**Модуль FileInput.xaml.cs, реализующий инициализацию основного класса с помощью файлов:**

using System;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

using Windows.UI.Popups;

using Windows.Storage;

namespace \_5\_crypto\_2\_final\_ver

{

public sealed partial class FileInput : Page

{

public string probs\_file\_name = "probabilities.txt";

public string input\_file\_name = "input.txt";

public string output\_file\_name = "output.txt";

public FileInput()

{

this.InitializeComponent();

CheckFiles();

}

private async void CheckFiles()

{

string temp;

//Получение доступа к нужной папке

StorageFolder storageFolder = ApplicationData.Current.LocalFolder;

temp = storageFolder.Path;

//Вывод путей к файлам на экран

ProbsPathTextBox.Text = temp + @"\" + probs\_file\_name;

InputPathTextBox.Text = temp + @"\" + input\_file\_name;

OutputPathTextBox.Text = temp + @"\" + output\_file\_name;

//Если файлов не существует, создать их

await storageFolder.CreateFileAsync(probs\_file\_name, CreationCollisionOption.OpenIfExists);

await storageFolder.CreateFileAsync(input\_file\_name, CreationCollisionOption.OpenIfExists);

await storageFolder.CreateFileAsync(output\_file\_name, CreationCollisionOption.OpenIfExists);

}

private async void EncodeFileButton\_Click(object sender, RoutedEventArgs e)

{

string output, probs, input;

//Получение доступа к нужной папке

StorageFolder storageFolder = ApplicationData.Current.LocalFolder;

//Если файлов не существует, создать их

StorageFile probs\_file = await storageFolder.CreateFileAsync(probs\_file\_name, CreationCollisionOption.OpenIfExists);

StorageFile input\_file = await storageFolder.CreateFileAsync(input\_file\_name, CreationCollisionOption.OpenIfExists);

StorageFile output\_file = await storageFolder.CreateFileAsync(output\_file\_name, CreationCollisionOption.OpenIfExists);

//Получение алфавита и сообщения из файлов

probs = await FileIO.ReadTextAsync(probs\_file);

input = await FileIO.ReadTextAsync(input\_file);

try

{

//Инициализация класса с использованием текста из файла probabilities.txt

StartParameters sp = new StartParameters(probs);

//Кодирование информации из файла input.txt

output = sp.CodeMessage(input);

ResultTextBox.Text = output;

//Запись результата в файл

await FileIO.WriteTextAsync(output\_file, output);

//Вывод кодовых слов на экран

CodeWordsTextBox.Text = "";

for (int i = 0; i < sp.N; i++)

{

CodeWordsTextBox.Text += sp.names[i] + " -> " + sp.code\_words[i] + Environment.NewLine;

}

//Вывод характеристик на экран

CharacteristicsTextBox.Text = "Средняя длина кодового слова - " + sp.average\_length + Environment.NewLine;

CharacteristicsTextBox.Text += "Избыточность - " + sp.redundancy + Environment.NewLine;

CharacteristicsTextBox.Text += "Неравенство Крафта - сумма равна " + sp.KraftInequality + " ";

if (sp.KraftInequality < 1) { CharacteristicsTextBox.Text += "< 1, условие выполняется."; }

else if (sp.KraftInequality == 1) { CharacteristicsTextBox.Text += "= 1, оптимальная кодировка."; }

else { CharacteristicsTextBox.Text += "> 1, условие не выполняется."; }

}

catch (Exception exc)

{

//Если что-то пошло не так, на экран выводится сообщение с ошибкой

MessageDialog message = new MessageDialog(exc.Message);

await message.ShowAsync().AsTask();

}

}

private async void CodeFileButton\_Click(object sender, RoutedEventArgs e)

{

string output, probs, input;

//Получение доступа к нужной папке

StorageFolder storageFolder = ApplicationData.Current.LocalFolder;

//Если файлов не существует, создать их

StorageFile probs\_file = await storageFolder.CreateFileAsync(probs\_file\_name, CreationCollisionOption.OpenIfExists);

StorageFile input\_file = await storageFolder.CreateFileAsync(input\_file\_name, CreationCollisionOption.OpenIfExists);

StorageFile output\_file = await storageFolder.CreateFileAsync(output\_file\_name, CreationCollisionOption.OpenIfExists);

//Получение алфавита и сообщения из файлов

probs = await FileIO.ReadTextAsync(probs\_file);

input = await FileIO.ReadTextAsync(input\_file);

try

{

//Инициализация класса с использованием текста из файла probabilities.txt

StartParameters sp = new StartParameters(probs);

//Кодирование информации из файла input.txt

output = sp.DecodeMessage(input);

ResultTextBox.Text = output;

//Запись результата в файл

await FileIO.WriteTextAsync(output\_file, output);

//Вывод кодовых слов на экран

CodeWordsTextBox.Text = "";

for (int i = 0; i < sp.N; i++)

{

CodeWordsTextBox.Text += sp.names[i] + " -> " + sp.code\_words[i] + Environment.NewLine;

}

//Вывод характеристик на экран

CharacteristicsTextBox.Text = "Средняя длина кодового слова - " + sp.average\_length + Environment.NewLine;

CharacteristicsTextBox.Text += "Избыточность - " + sp.redundancy + Environment.NewLine;

CharacteristicsTextBox.Text += "Неравенство Крафта - сумма равна " + sp.KraftInequality + " ";

if (sp.KraftInequality < 1) { CharacteristicsTextBox.Text += "< 1, условие выполняется."; }

else if (sp.KraftInequality == 1) { CharacteristicsTextBox.Text += "= 1, оптимальная кодировка."; }

else { CharacteristicsTextBox.Text += "> 1, условие не выполняется."; }

}

catch (Exception exc)

{

//Если что-то пошло не так, на экран выводится сообщение с ошибкой

MessageDialog message = new MessageDialog(exc.Message);

await message.ShowAsync().AsTask();

}

}

}

}

**Графика программы написана на языке разметки XAML.**

**Главное меню:**

<Page

x:Class="\_5\_crypto\_2\_final\_ver.MainPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="using:\_5\_crypto\_2\_final\_ver"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d"

Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

<Grid Background="#FFB9C5C3">

<Grid.RowDefinitions>

<RowDefinition Height="0.5\*"/>

<RowDefinition Height="9\*"/>

<RowDefinition Height="0.5\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="1\*"/>

</Grid.ColumnDefinitions>

<Border

Grid.Row="0"

Grid.Column="0"

Grid.ColumnSpan="3"

Background="#FF9BA6A4"

Margin="4">

<Viewbox>

<TextBlock Text="Кодирование и декодирование информации алгоритмом Хаффмана"/>

</Viewbox>

</Border>

<Border

Grid.Row="2"

Grid.Column="0"

Background="#FF87BFC5"

Margin="4">

<Button

HorizontalAlignment="Stretch"

VerticalAlignment="Stretch"

Background="#FF87BFC5"

Name="MainMenuButton"

Click="GoToMainMenu">

<Viewbox>

<TextBlock Text="Главное меню"/>

</Viewbox>

</Button>

</Border>

<Border

Grid.Row="2"

Grid.Column="1"

Background="#FFB9C3CB"

Margin="4">

<Button

HorizontalAlignment="Stretch"

VerticalAlignment="Stretch"

Background="#FF87BFC5"

Name="KeyboardInputButton"

Click="GoToKeyboardInput">

<Viewbox>

<TextBlock Text="Ввод с клавиатуры"/>

</Viewbox>

</Button>

</Border>

<Border

Grid.Row="2"

Grid.Column="2"

Background="#FFB9C3CB"

Margin="4">

<Button

HorizontalAlignment="Stretch"

VerticalAlignment="Stretch"

Background="#FF87BFC5"

Name="FileInputButton"

Click="GoToFileInput">

<Viewbox>

<TextBlock Text="Ввод из файлов"/>

</Viewbox>

</Button>

</Border>

<Frame

x:Name="frame"

Grid.Row="1"

Grid.Column="0"

Grid.ColumnSpan="3"/>

</Grid>

</Page>

**Справка:**

<Page

x:Class="\_5\_crypto\_2\_final\_ver.MainMenu"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="using:\_5\_crypto\_2\_final\_ver"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d"

Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

<Grid Background="#FFB9C5C3">

<Border

Background="#FF9BA6A4"

Margin="4"

>

<TextBlock

FontSize="20"

TextWrapping="Wrap"

Text="Краткая справка об использовании приложения:&#10;&#10;Для ввода данных, необходимых для работы программы предусмотрено два режима работы: ввод с клавиатуры и ввод из файла.&#10;&#10;Для ввода с клавиатуры необходимо нажать соответствующую кнопку 'Ввод с клавиатуры', таким образом откроется специальный интерфейс, в котором можно будет задать алфавит и последовательность для кодировки/декодировки. Для правильного задания алфавита нужно соблюдать следующие правила:&#10;(1) каждый символ алфавита уникален, не должно быть повторений;&#10;(2) сумма вероятностей всех символов должна равняться 1;&#10;(3) для получения результата символов в алфавите должно быть хотя бы 2&#10;(4) Вероятности - неотрицательные числа.&#10;При этом вводить алфавит надо определенным образом: &#10;&lt;Символ 1>[Пробел]&lt;Вероятность символа 1>[Пробел]&lt;Символ 2>[Пробел]&lt;Вероятность символа 2>[Пробел]...[Пробел]&lt;Символ n>[Пробел]&lt;Вероятность символа n>&#10;для алфавита размерности n. При этом для разделения целой части вероятности от дробной следует использовать ',', а не '.'.&#10;Для кодирования информации следует вписать кодируемое сообщение в поле ввода (сообщение должно состоять только из символов алфавита) и нажать кнопку 'Кодировать'.&#10;Для декодирования сообщения необходимое сообщение в поле ввода (сообщение должно состоять только из кодовых слов алфавита) и нажать кнопку 'Декодировать'.&#10;&#10;Для ввода данных из файлов необходимо нажать ссответсвующую кнопку 'Ввод из файлов', таким образом откроется специальный интерфейс, в котором можно посмотреть пути к файлам данных. Существует три файла:&#10;(1) Файл, задающий алфавит 'probabilities.txt';&#10;(2) Файл, задающий сообщение для кодирования/декодирования 'input.txt';&#10;(3) Файл, в который записывается результат кодирования/декодирования 'output.txt'.&#10;Правила для записи данных в файлы такие же как и для записи данных в текстовые поля для режима работы с клавитурой.&#10;&#10;Оба режима работы для заданного алфавита также выводят на экран значения кодовых слов, а также значения некоторых характеристик. Этими характеристиками являются:&#10;(1) Средняя длина кодовго слова;&#10;(2) Избыточность;&#10;(3) Проверка неравенства Крафта.&#10;&#10;В течение работы с приложением возможно переключение между режимами работы, а также возврат на главное меню со справкой об использовании приложения. Для возврата на главное меню необходимо нажать кнопку 'Главное меню'."

/>

</Border>

</Grid>

</Page>

**Меню с вводом с клавиатуры:**

<Page

x:Class="\_5\_crypto\_2\_final\_ver.KeyboardInput"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="using:\_5\_crypto\_2\_final\_ver"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d"

Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

<Grid Background="#FFB9C5C3">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="1\*" />

<ColumnDefinition Width="1\*" />

<ColumnDefinition Width="1\*" />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="1.5\*" />

<RowDefinition Height="1.5\*" />

<RowDefinition Height="1\*" />

<RowDefinition Height="1\*" />

<RowDefinition Height="3\*" />

<RowDefinition Height="3\*" />

</Grid.RowDefinitions>

<Border

Background="#FF9BA6A4"

Grid.Column="0"

Grid.Row="0"

Margin="4"

Name="EnterProbsTextBlock">

<Viewbox>

<TextBlock>Введите алфавит<LineBreak/>([Символ][Пробел][Вероятность]):</TextBlock>

</Viewbox>

</Border>

<Border

Background="#FF9BA6A4"

Grid.Column="0"

Grid.Row="1"

Margin="4"

Name="EnterInputTextBlock">

<Viewbox>

<TextBlock>Введите данные для<LineBreak/> кодирования/декодирования</TextBlock>

</Viewbox>

</Border>

<Border

Background="#FF9BA6A4"

Grid.Column="0"

Grid.Row="2"

Margin="4"

Name="ChooseModeTextBlock">

<Viewbox>

<TextBlock>Выберете действие, которое необходимо<LineBreak/>выполнить над исходными данными</TextBlock>

</Viewbox>

</Border>

<Border

Background="#FF9BA6A4"

Grid.Column="0"

Grid.Row="3"

Margin="4"

Name="ResultTextBlock">

<Viewbox>

<TextBlock>Результат выбранного действия:</TextBlock>

</Viewbox>

</Border>

<Border

Background="#FF9BA6A4"

Grid.Column="0"

Grid.Row="4"

Margin="4"

Name="CodeWordsTextBlock">

<Viewbox>

<TextBlock>Кодовые слова для<LineBreak/>каждого символа алфавита:</TextBlock>

</Viewbox>

</Border>

<Border

Background="#FF9BA6A4"

Grid.Column="0"

Grid.Row="5"

Margin="4"

Name="CharacteristicsTextBlock">

<Viewbox>

<TextBlock>Результаты вычисления<LineBreak/>некоторых характеристик на<LineBreak/>основе заданного алфавита:</TextBlock>

</Viewbox>

</Border>

<Border

Background="#FFCFDAD8"

Grid.Column="1"

Grid.Row="0"

Grid.ColumnSpan="2"

Margin="4">

<TextBox Name="EnterProbsTextBox"/>

</Border>

<Border

Background="#FFCFDAD8"

Grid.Column="1"

Grid.Row="1"

Grid.ColumnSpan="2"

Margin="4">

<TextBox Name="EnterInputTextBox"/>

</Border>

<Border

Background="#FFCFDAD8"

Grid.Column="1"

Grid.Row="3"

Grid.ColumnSpan="2"

Margin="4">

<TextBox TextWrapping="Wrap" Name="ResultTextBox"/>

</Border>

<Border

Background="#FFCFDAD8"

Grid.Column="1"

Grid.Row="4"

Grid.ColumnSpan="2"

Margin="4">

<TextBox TextWrapping="Wrap" Name="CodeWordsTextBox"/>

</Border>

<Border

Background="#FFCFDAD8"

Grid.Column="1"

Grid.Row="5"

Grid.ColumnSpan="2"

Margin="4">

<TextBox TextWrapping="Wrap" Name="CharacteristicsTextBox"/>

</Border>

<Border

Grid.Column="1"

Grid.Row="2"

Margin="4">

<Button

Name="FirstChooseButton"

Background="#FF87BFC5"

VerticalAlignment="Stretch"

HorizontalAlignment="Stretch"

Click="FirstChooseButton\_Click">

<Viewbox>

<TextBlock Text="Закодировать"/>

</Viewbox>

</Button>

</Border>

<Border

Grid.Column="2"

Grid.Row="2"

Margin="4">

<Button

Name="SecondChooseButton"

Background="#FF87BFC5"

VerticalAlignment="Stretch"

HorizontalAlignment="Stretch"

Click="SecondChooseButton\_Click">

<Viewbox>

<TextBlock Text="Декодировать"/>

</Viewbox>

</Button>

</Border>

</Grid>

</Page>

**Меню с вводом из файлов:**

<Page

x:Class="\_5\_crypto\_2\_final\_ver.FileInput"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="using:\_5\_crypto\_2\_final\_ver"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d"

Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

<Grid Background="#FFB9C5C3">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="1\*" />

<ColumnDefinition Width="1\*" />

<ColumnDefinition Width="1\*" />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="1\*" />

<RowDefinition Height="1\*" />

<RowDefinition Height="1\*" />

<RowDefinition Height="1\*" />

<RowDefinition Height="1\*" />

<RowDefinition Height="3\*" />

<RowDefinition Height="3\*" />

</Grid.RowDefinitions>

<Border

Background="#FF9BA6A4"

Grid.Column="0"

Grid.Row="0"

Margin="4"

Name="ProbsPathTextBlock">

<Viewbox>

<TextBlock>В данный файл введите<LineBreak/>вероятности символов алфавита:</TextBlock>

</Viewbox>

</Border>

<Border

Background="#FF9BA6A4"

Grid.Column="0"

Grid.Row="1"

Margin="4"

Name="InputPathTextBlock">

<Viewbox>

<TextBlock>В данный файл введите последовательность<LineBreak/> для кодирования/декодирования</TextBlock>

</Viewbox>

</Border>

<Border

Background="#FF9BA6A4"

Grid.Column="0"

Grid.Row="2"

Margin="4"

Name="OutputPathTextBlock">

<Viewbox>

<TextBlock>В данном файле можно увидеть<LineBreak/>результаты кодирования/декодирования</TextBlock>

</Viewbox>

</Border>

<Border

Background="#FF9BA6A4"

Grid.Column="0"

Grid.Row="3"

Margin="4"

Name="ChooseModeTextBlock">

<Viewbox>

<TextBlock>Выберете действие, которое необходимо<LineBreak/>выполнить над исходными данными</TextBlock>

</Viewbox>

</Border>

<Border

Background="#FF9BA6A4"

Grid.Column="0"

Grid.Row="4"

Margin="4"

Name="ResultTextBlock">

<Viewbox>

<TextBlock>Результат выбранного действия:</TextBlock>

</Viewbox>

</Border>

<Border

Background="#FF9BA6A4"

Grid.Column="0"

Grid.Row="5"

Margin="4"

Name="CodeWordsTextBlock">

<Viewbox>

<TextBlock>Кодовые слова для<LineBreak/>каждого символа алфавита:</TextBlock>

</Viewbox>

</Border>

<Border

Background="#FF9BA6A4"

Grid.Column="0"

Grid.Row="6"

Margin="4"

Name="CharacteristicsTextBlock">

<Viewbox>

<TextBlock>Результаты вычисления<LineBreak/>некоторых характеристик на<LineBreak/>основе заданного алфавита:</TextBlock>

</Viewbox>

</Border>

<Border

Background="#FFCFDAD8"

Grid.Column="1"

Grid.Row="0"

Grid.ColumnSpan="2"

Margin="4">

<TextBox Name="ProbsPathTextBox"/>

</Border>

<Border

Background="#FFCFDAD8"

Grid.Column="1"

Grid.Row="1"

Grid.ColumnSpan="2"

Margin="4">

<TextBox Name="InputPathTextBox"/>

</Border>

<Border

Background="#FFCFDAD8"

Grid.Column="1"

Grid.Row="2"

Grid.ColumnSpan="2"

Margin="4">

<TextBox Name="OutputPathTextBox"/>

</Border>

<Border

Background="#FFCFDAD8"

Grid.Column="1"

Grid.Row="4"

Grid.ColumnSpan="2"

Margin="4">

<TextBox TextWrapping="Wrap" Name="ResultTextBox"/>

</Border>

<Border

Background="#FFCFDAD8"

Grid.Column="1"

Grid.Row="5"

Grid.ColumnSpan="2"

Margin="4">

<TextBox TextWrapping="Wrap" Name="CodeWordsTextBox"/>

</Border>

<Border

Background="#FFCFDAD8"

Grid.Column="1"

Grid.Row="6"

Grid.ColumnSpan="2"

Margin="4">

<TextBox TextWrapping="Wrap" Name="CharacteristicsTextBox"/>

</Border>

<Border

Grid.Column="1"

Grid.Row="3"

Margin="4">

<Button

Name="FirstChooseButton"

Background="#FF87BFC5"

VerticalAlignment="Stretch"

HorizontalAlignment="Stretch"

Click="EncodeFileButton\_Click">

<Viewbox>

<TextBlock Text="Закодировать"/>

</Viewbox>

</Button>

</Border>

<Border

Grid.Column="2"

Grid.Row="3"

Margin="4">

<Button

Name="SecondChooseButton"

Background="#FF87BFC5"

VerticalAlignment="Stretch"

HorizontalAlignment="Stretch"

Click="CodeFileButton\_Click">

<Viewbox>

<TextBlock Text="Декодировать"/>

</Viewbox>

</Button>

</Border>

</Grid>

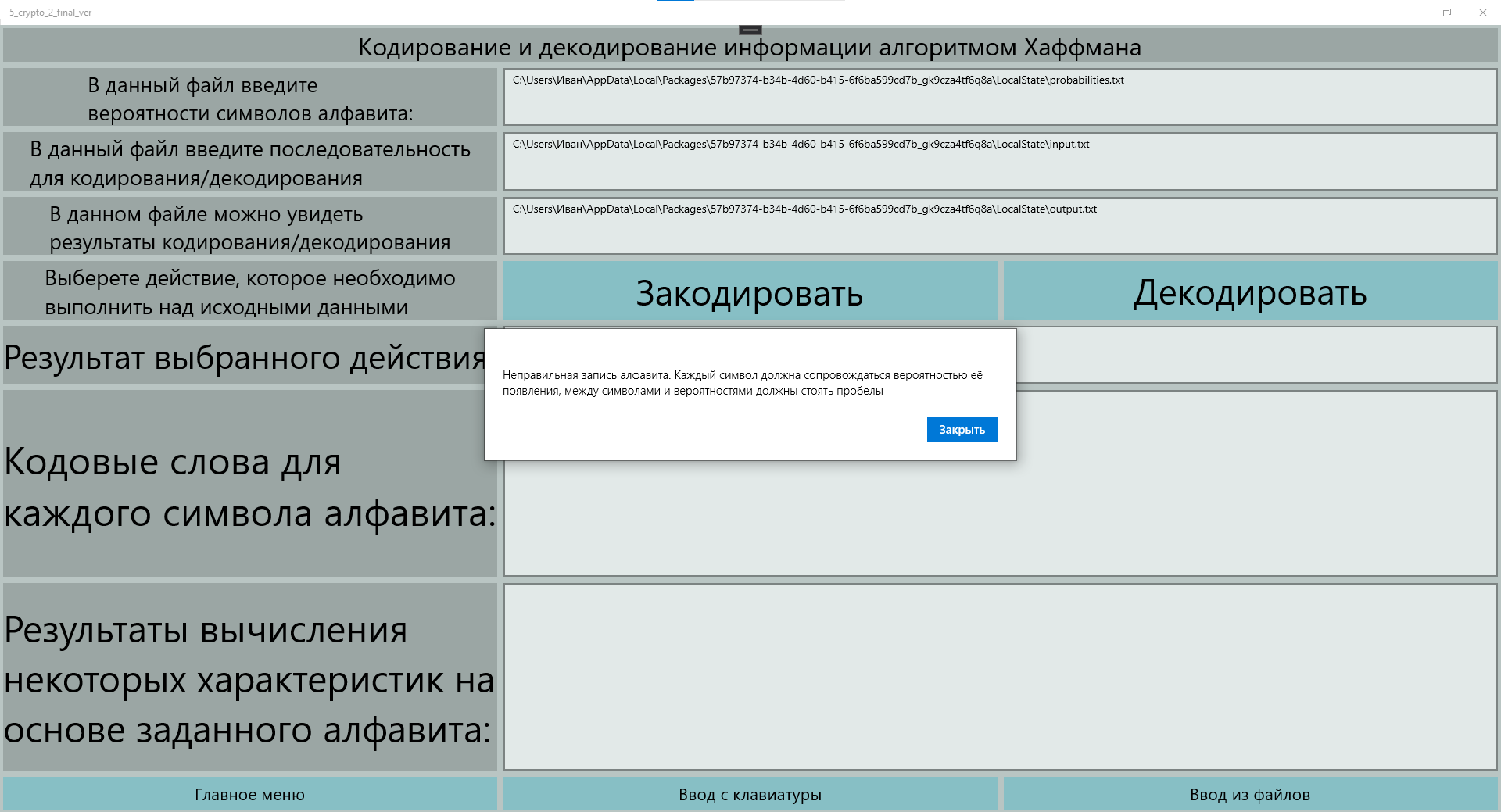
</Page>

**Тестирование программы**

**Если пользователь делает что-либо не так при работе с приложением, большое количество исключительных ситуаций сообщат ему о той или иной ошибке в вводе. Ошибки, случающиеся при вводе с клавиатуры, совпадают с ошибками, случающимися при вводе из файлов, так как по сути происходит один и тот же процесс получения данных - в виде строк. Для удобства будут рассмотрены исключительные ситуации только при работе с файлами, поскольку нет смысла в показе исключительных ситуаций при работе с клавиатурой, ведь они будут теми же самыми.**

1. Файл, задающий алфавит (probabilities.txt), пуст:

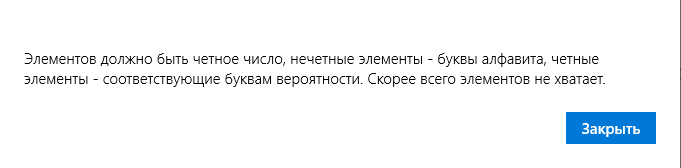




Далее для экономии места будет показываться только окно с ошибкой, все последующие ошибки также выводятся на том же месте, с таким же задним планом.

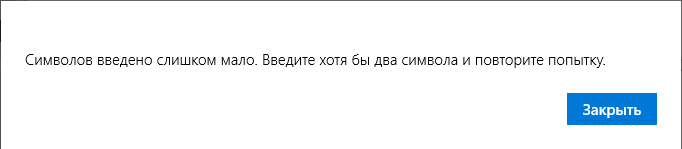
2. Нечетное число элементов в файле probabilities.txt:





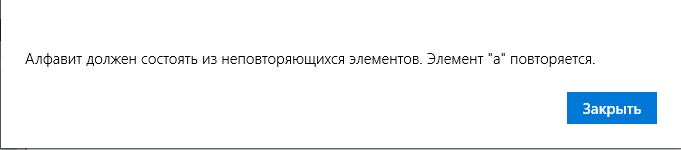
3. Слишком мало символов:





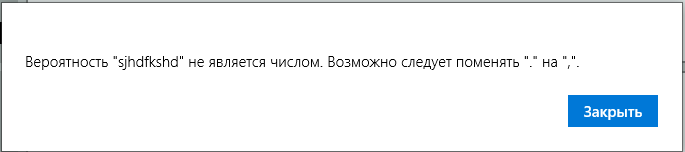
4. Символы в алфавите повторяются:



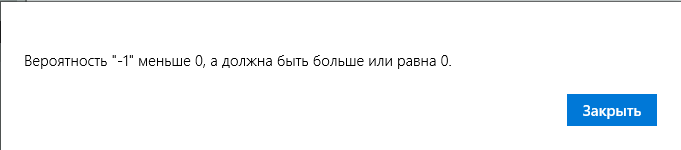


5. Одна из вероятностей не является числом:

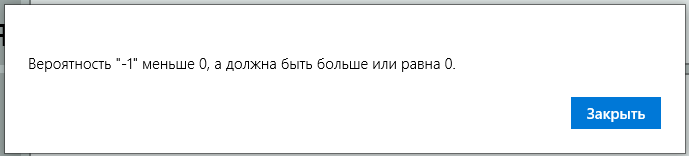




6. Одна из вероятностей меньше 0:

7. Сумма вероятностей больше 1:

8. При кодировании сообщения было обнаружено, что одного из символов нет в алфавите:

Алфавит:

Сообщение:



Результат:



9. При декодировании сообщения было обнаружено, сообщение не подлежит декодированию:

Алфавит:



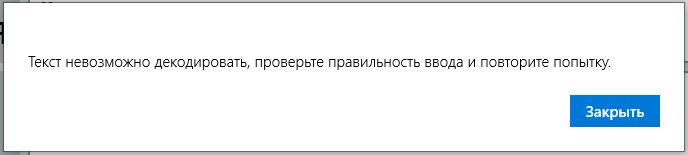
Кодовые слова:



Сообщение:



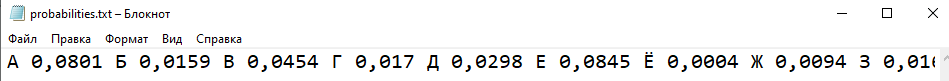
Результат:



**При отсутствии данных ошибок приложение может работать с любым алфавитом:**

****

Алфавит:



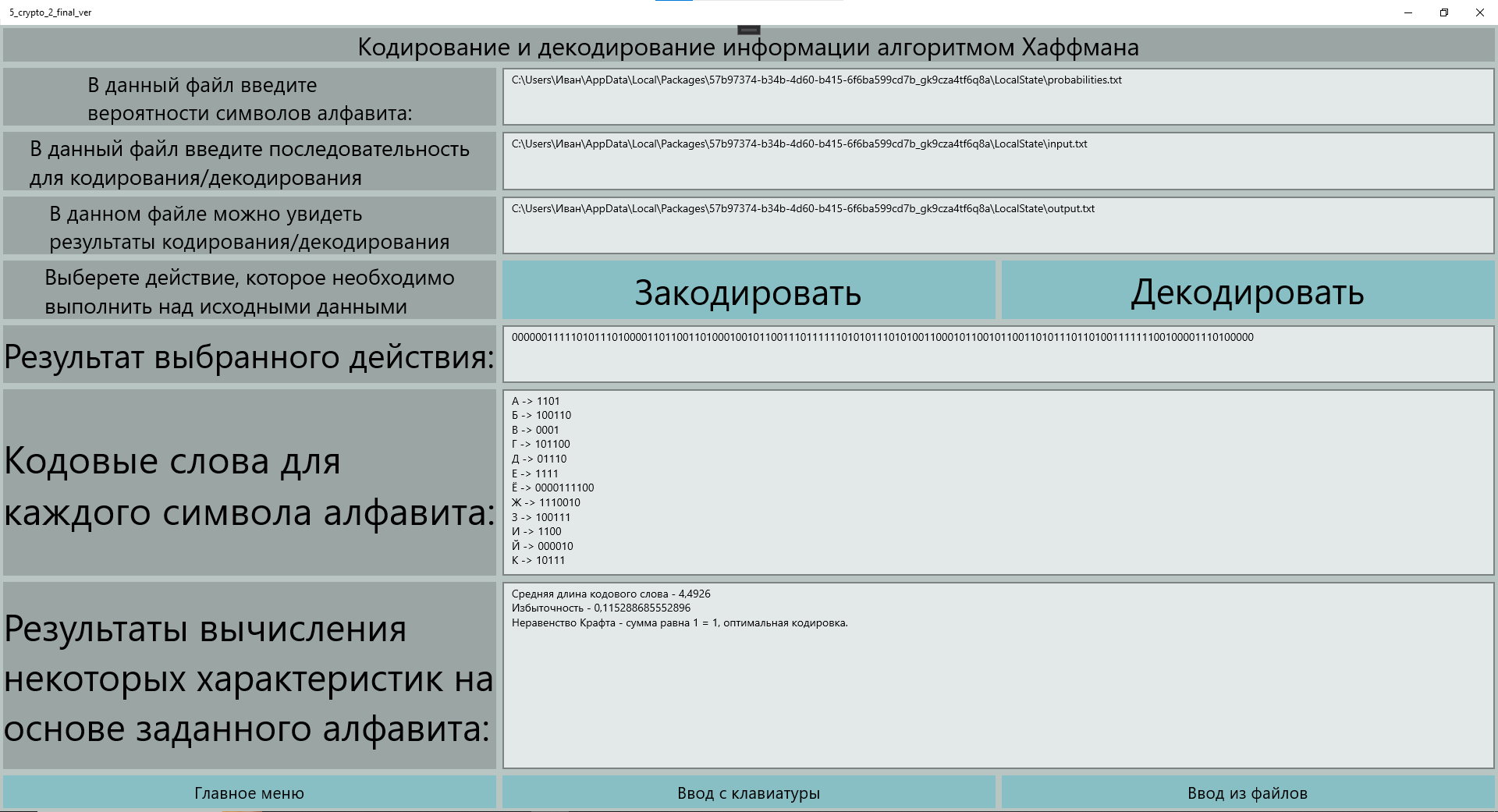
А 0,0801 Б 0,0159 В 0,0454 Г 0,017 Д 0,0298 Е 0,0845 Ё 0,0004 Ж 0,0094 З 0,0165 И 0,0735 Й 0,0121 К 0,0349 Л 0,044 М 0,0321 Н 0,0670 О 0,1097 П 0,0281 Р 0,0473 С 0,0547 Т 0,0626 У 0,0262 Ф 0,0026 Х 0,0097 Ц 0,0048 Ч 0,0144 Ш 0,0073 Щ 0,0036 Ъ 0,0004 Ы 0,0190 Ь 0,0174 Э 0,0032 Ю 0,0064 Я 0,02

**Кодировка:**

Сообщение:

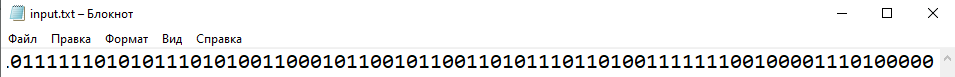


Результат:



**Декодировка:**

Сообщение:



000000111110101110100001101100110100010010110011101111110101011101010011000101100101100110101110110100111111100100001110100000

Результат:

