|  |
| --- |
|  |
| **РОСЖЕЛДОР**  **федеральное государственное бюджетное образовательное учреждение высшего образования**  **«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПУТЕЙ СООБЩЕНИЯ» (СГУПС)**  Кафедра «Информационные технологии транспорта»  Почти оптимальное кодирование  Лабораторная работа  по дисциплине «Теория информации и кодирования»   |  |  |  |  | | --- | --- | --- | --- | | Руководитель: |  | Выполнил: | | | преподаватель |  | студент гр. | БПИ-311 | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_В. С. Выплавень  *подпись* |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. А. Давиденко  *подпись* | | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *дата проверки* |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *дата сдачи на проверку* | | |  |  |  | |   Краткая рецензия:   |  | | --- | |  | |  | |  | |  | |  |  |  |  |  | | --- | --- | --- | |  |  |  | | *запись о допуске к защите* |  |  | |  |  |  | | *оценка по результатам защиты* |  | *подписи преподавателей и дата защиты* |   Новосибирск  2023 |

**РОСЖЕЛДОР**

**федеральное государственное бюджетное образовательное учреждение высшего образования**

**«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПУТЕЙ СООБЩЕНИЯ» (СГУПС)**

Кафедра «Информационные технологии транспорта»

Задание на выполнение лабораторной работы

по дисциплине «Теория информации и кодирования»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| студенту | Давиденко Владиславу Александровичу  в дательном падеже (например, Усовой Эльвире Анатольевне) |  | Группа | БПИ-311 |
| Вариант | 46 |  |  |  |

Тема: **Расширенный код Хэмминга (8, 4)**

Цель работы: Закрепление навыка разработки графического приложения для решения практических задач передачи информации по каналу связи с использованием метода «Расширенный код Хэмминга (8, 4)».

Задания:

1) Ознакомиться с теоретическим материалом почти оптимального кодирования. Изучить методы побуквенного кодирования, способы оценки оптимальности кода и его избыточности.

2) Составить алгоритм программы, спроектировать графическое приложение и создать тестовые примеры.

3) Разработать графическое приложение.

4) Провести численный эксперимент на компьютере, для каждого тестового примера оценить избыточность кода или сообщения.

5) Проанализировать полученные данные. Сделать выводы.

Сроки сдачи на проверку: 4 неделя текущего семестра.

Сроки защиты: 4 неделя текущего семестра.

Работу оформить в соответствии со стандартом организации СТО СГУПС 1.01 БИ.01-2019 «Система менеджмента качества. Письменная отчетная работа. Требования к оформлению».

График выполнения

| Название документа и раздела | Ориентировочно | | График  выпол-  нения (недели) |
| --- | --- | --- | --- |
| колич. страниц записки | трудо-емкость  в часах |
| Аналитический обзор, изучение теоретического материала | 1 | 2 | 7 |
| Формализация проблемы, построение математической модели, разработка алгоритма программы | 2 | 2 | 7 |
| Реализация | 20 | 10 | 8 |
| Разработка тестового примера, тестирование | 7 | 2 | 9 |
| Написание пояснительной записки и разработка руководства пользователя | - | 2 | 10 |
| *Общая трудоемкость* | *20* | *18* | - |

Основная литература и интернет-источники:

1. Матвеев, Б. В. Основы корректирующего кодирования: теория и лабораторный практикум : учебное пособие / Б. В. Матвеев. — 2-е изд., стер. — Санкт-Петербург : Лань, 2021. — 192 с. — ISBN 978-5-8114-1631-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/168880 (дата обращения: 03.09.2021). — Режим доступа: для авториз. пользователей.
2. Котенко, В. В. Теория информации : учебное пособие / Котенко В. В. - Ростов н/Д : Изд-во ЮФУ, 2018. - 239 с. - ISBN 978-5-9275-2370-2. - Текст : электронный // ЭБС "Консультант студента" : [сайт]. - URL : https://www.studentlibrary.ru/book/ISBN9785927523702.html (дата обращения: 03.09.2021). - Режим доступа : по подписке.
3. Сайт СЭОР СГУПС – Система электронных образовательных ресурсов сибирского государственного университета путей сообщения (СГУПС) Учебно-методический комплекс дисциплины «Теория информации и кодирования». / Э. А. Усова. – Новосибирск : [сайт]. – URL: http://moodle3.stu.ru/course/view.php?id=1612 (дата обращения: 01.09.2021). – Режим доступа: для авториз. пользователей. – Текст. Изображение : электронные.

Задание выдано «20» сентября 2023 г.

|  |  |  |
| --- | --- | --- |
| Руководитель |  | */ В. С. Выплавень* |

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc148700963)

[1  Постановка задачи 6](#_Toc148700964)

[2  Матричный способ Хэмминга 7](#_Toc148700965)

[3  Реализация программного обеспечения 11](#_Toc148700966)

[ЗАКЛЮЧЕНИЕ 15](#_Toc148700967)

[СПИСОК ИСПОЛЬЗОВАННЫХ ПЕРВОИСТОЧНИКОВ 16](#_Toc148700968)

[Приложение А 17](#_Toc148700969)

[Приложение Б 18](#_Toc148700970)

[Приложение В 22](#_Toc148700971)

[Приложение Г 24](#_Toc148700972)

ВВЕДЕНИЕ

Цель лабораторной работы – разработка графического приложения кодирования методом «Расширенный код Хэмминга (8, 4)», оценки его статистических характеристик с помощью границы Варшамова-Гильберта.

Для достижения поставленной цели необходимо решить следующие задачи:

– постановка задачи;

– изучение теоретического материала;

– разработка комплекта тестов;

– выбор инструмента разработки и проектирование графического интерфейса;

– программирование и отладка программы;

– тестирование и анализ полученных результатов.

В данной расчетно-графической работе используется среда разработки Microsoft Visual Studio 2022, языка программирования — C#.

1. Постановка задачи

Разработать графическое приложения кодирования методом «Расширенный код Хэмминга (8, 4)», оценить его статистические характеристики с помощью границы Хэмминга.

Требования к разрабатываемому программному обеспечению:

1) Разработать графическое приложение для передачи информации по зашумленному каналу связи. Проверку правильности передачи сообщения выполнить с помощью расширенного линейного блокового кода (матричный способ) Хэмминга.

2) Предусмотреть очистку полей ввода и вывода.

3) Предусмотреть обработку корректности ввода данных.

4) Оценить оптимальность и избыточность кода.

5) Реализовать вывод промежуточных результатов, с помощью которых выполняется кодирование символов сообщения.

6) Реализовать возможность ввода искаженного сообщения.

1. Матричный способ Хэмминга

Назначение помехоустойчивого кодирования – защита информации от помех и ошибок при передаче и хранении информации. Помехоустойчивое кодирование необходимо для устранения ошибок, которые возникают в процессе передачи, хранения информации. При передачи информации по каналу связи возникают помехи, ошибки и небольшая часть информации теряется.

Без использования помехоустойчивого кодирования было бы невозможно передавать большие объемы информации (файлы), т.к. в любой системе передачи и хранении информации неизбежно возникают ошибки.

Рассмотрим пример CD диска. Там информация хранится прямо на поверхности диска, в углублениях, из-за того, что все дорожки на поверхности, часто диск хватаем пальцами, елозим по столу и из-за этого без помехоустойчивого кодирования, информацию извлечь не получится.

Использование кодирования позволяет извлекать информацию без потерь даже с поврежденного CD/DVD диска, когда какая либо область становится недоступной для считывания.

В зависимости от того, используется в системе обнаружение или исправление ошибок с помощью помехоустойчивого кода, различают следующие варианты:

* запрос повторной передачи (Automatic Repeat reQuest, ARQ): с помощью помехоустойчивого кода выполняется только обнаружение ошибок, при их наличии производится запрос на повторную передачу пакета данных;
* прямое исправление ошибок (Forward Error Correction, FEC): производится декодирование помехоустойчивого кода, т. е. исправление ошибок с его помощью.

Возможен также гибридный вариант, чтобы лишний раз не гонять информацию по каналу связи, например получили пакет информации, попробовали его исправить, и если не смогли исправить, тогда отправляется запрос на повторную передачу.

Любое помехоустойчивое кодирование добавляет избыточность, за счет чего и появляется возможность восстановить информацию при частичной потере данных в канале связи (носителе информации при хранении). В случае эффективного кодирования убирали избыточность, а в помехоустойчивом кодировании добавляется контролируемая избыточность.

Простейший пример – мажоритарный метод, он же многократная передача, в котором один символ передается многократно, а на приемной стороне принимается решение о том символе, количество которых больше.

Допустим есть 4 символа информации, А, B, С,D, и эту информацию повторяем несколько раз. В процессе передачи информации по каналу связи, где-то возникла ошибка. Есть три пакета (A1B1C1D1|A2B2C2D2|A3B3C3D3), которые должны нести одну и ту же информацию.

В каждом помехоустойчивом кодировании, существует несколько параметров. Первый параметр, скорость кода R характеризует долю информационных («полезных») данных в сообщении и определяется выражением: R=k/n=k/m+k

где n – количество символов закодированного сообщения (результата кодирования);

m – количество проверочных символов, добавляемых при кодировании;

k – количество информационных символов.

Параметры n и k часто приводят вместе с наименованием кода для его однозначной идентификации. Например, код Хэмминга (7,4) значит, что на вход кодера приходит 4 символа, на выходе 7 символов, Рида-Соломона (15, 11) и т.д.

Второй параметр, кратность обнаруживаемых ошибок – количество ошибочных символов, которые код может обнаружить.

Третий параметр, кратность исправляемых ошибок – количество ошибочных символов, которые код может исправить (обозначается буквой t).

Имеет место в таких кодах контроль четности. Самый простой метод помехоустойчивого кодирования, это добавление одного бита четности. Есть некое информационное сообщение, состоящее из 8 бит, добавим девятый бит.

Если нечетное количество единиц, добавляем 0.

1 0 1 0 0 1 0 0 | 0

Если четное количество единиц, добавляем 1.

1 1 0 1 0 1 0 0 | 1

Если принятый бит чётности не совпадает с рассчитанным битом чётности, то считается, что произошла ошибка.

1 1 0 0 0 1 0 0 | 1

Под кратностью понимается, всевозможные ошибки, которые можно обнаружить. В этом случае, кратность исправляемых ошибок 0, так как мы не можем исправить ошибки, а кратность обнаруживаемых 1.

Есть последовательность 0 и 1, и из этой последовательности составим прямоугольную матрицу размера 4 на 4. Затем для каждой строки и столбца посчитаем бит четности.

Классификация помехоустойчивых кодов:

* Код Хэмминга — наиболее известный из первых самоконтролирующихся и самокорректирующихся кодов. Позволяет устранить одну ошибку и находить двойную.
* Код Хэмминга (7,4) — 4 бита на входе кодера и 7 на выходе, следовательно 3 проверочных бита. С 1 по 4 информационные биты, с 6 по 7 проверочные (см. табл. выше). Пятый проверочный бит y5, это сумма по модулю два 1-3 информационных бит. Сумма по модулю 2 это вычисление бита чётности.

Декодирование происходит через вычисление синдрома по выражениям:

Синдром — это сложение бит по модулю два. Если синдром не нулевой, то исправление ошибки происходит по таблице декодирования:

Расстояние Хэмминга — число позиций, в которых соответствующие символы двух кодовых слов одинаковой длины различны. Если рассматривать два кодовых слова, (пример на картинке ниже, 1 0 1 1 0 0 1 и 1 0 0 1 1 0 1) видно что они отличаются друг от друга на два символа, соответственно расстояние Хэмминга равно 2.

Кратность исправляемых ошибок и обнаруживаемых, связано минимальным расстоянием Хэмминга. Любой помехоустойчивый код добавляет избыточность с целью увеличить минимальное расстояние Хэмминга. Именно минимальное расстояние Хэмминга определяет помехоустойчивость.

1. Реализация программного обеспечения

Для создания программного обеспечения было принято решения создать несколько классов.

В первом классе «ViewModelBase» происходит наследование от класса «INotifyPropertyChanged», это требуется для динамического отслеживания изменения параметров. Метод «Set» позволит будущим созданным классам привязаться к таблице и динамически обновлять свои свойства. Метод «onPropetyChanged» проверяет, произошли ли эти самые изменения, а «Dispose» освобождает ресурсы. Код данного класса представлен на рисунке 1 и в приложении А.

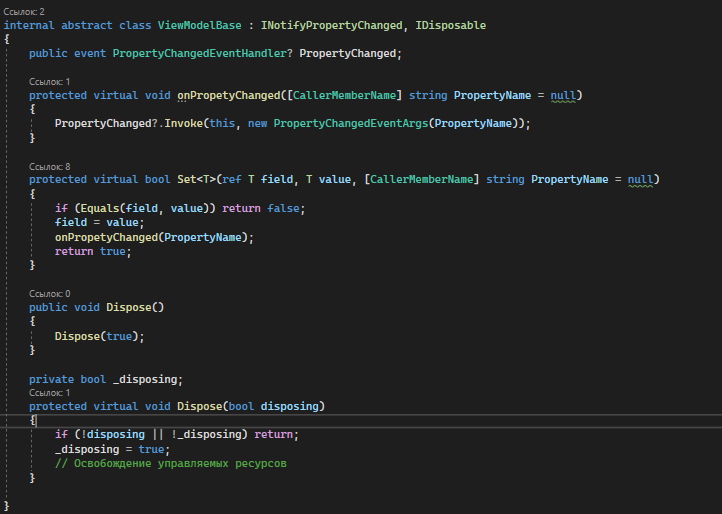


Рисунок 1 – Класс «ViewModelBase»

Второй класс «table» содержит все необходимые свойства выводимые на таблицу с реализованными через предыдущий класс «геттерами» и «сеттерами», с прописанными из варианта порождающей и проверочной матрицы и методами, которые взаимодействуют с этими данными и непосредственно производят все вычисления. Так же присутствует конструктор класса, через который все данные и инициализируются и метод «Refresh», позволяющий обновлять данные при внесении пользователем изменений. Столбцы таблицы в разметке xaml привязаны непосредственно к данному классу. Код данной сущности представлен на рисунке 2 и в приложении Б.

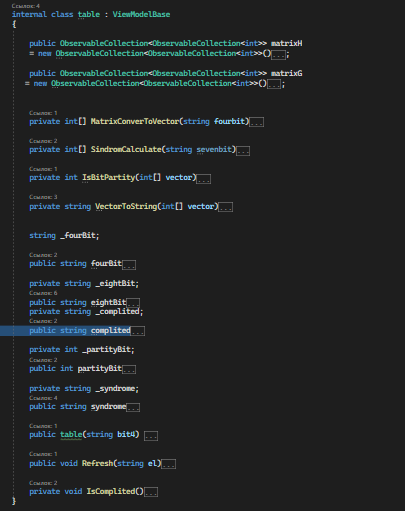


Рисунок 2 – Класс «table»

Далее в разметке xaml был разработан дизайн приложения (рисунок 3,4), а в классе «MainWindow», через обработчик события нажатия кнопки в окне или кнопки «enter» реализован вызов логики кода (рисунок 5).

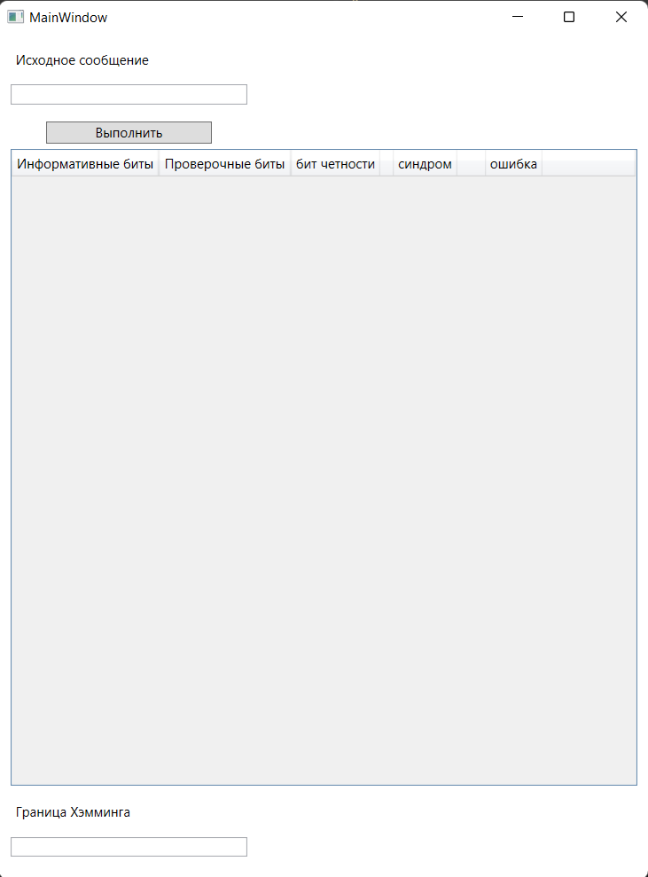


Рисунок 3 – Интерфейс приложения

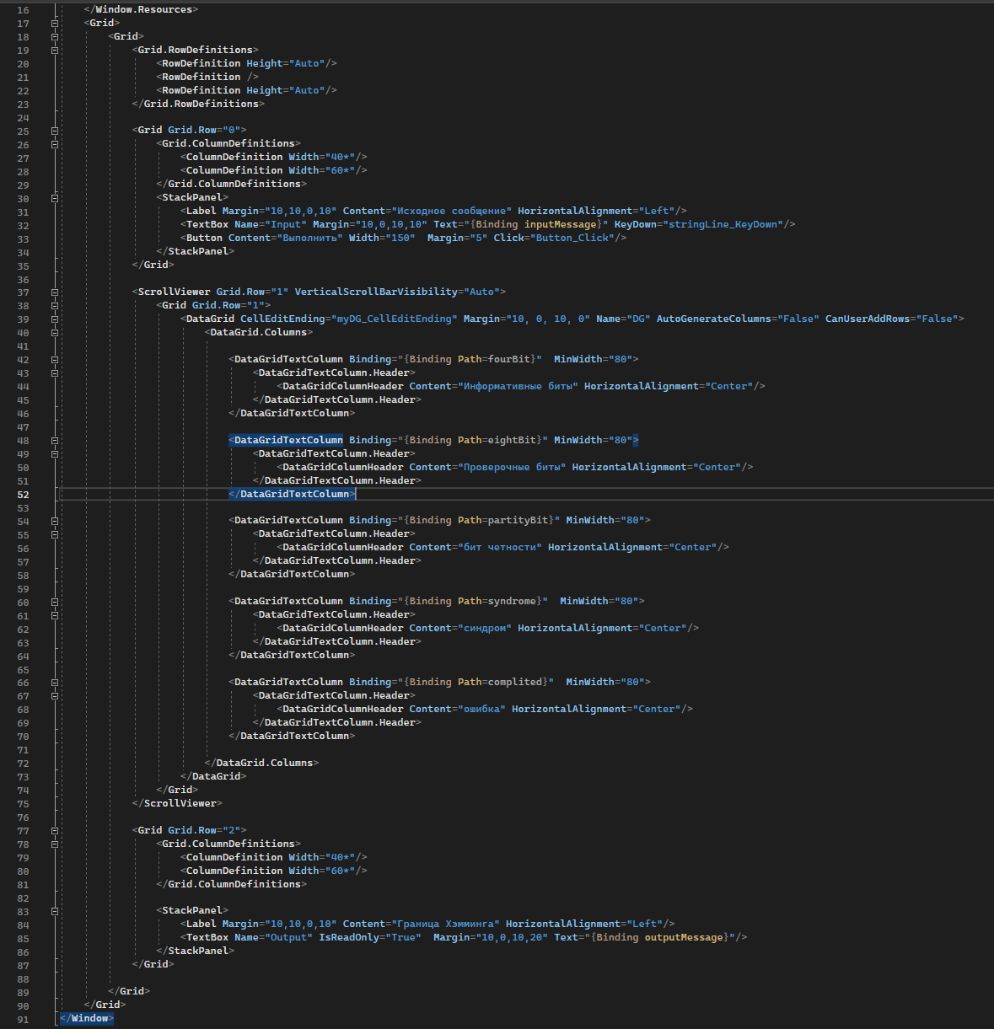


Рисунок 4 – Код разметки «xaml»

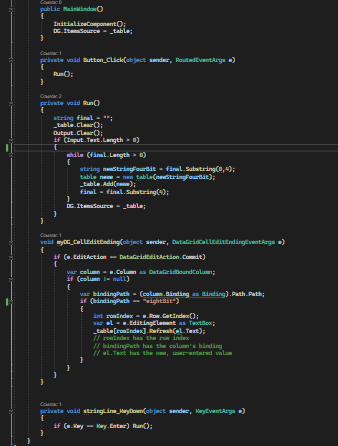


Рисунок 5 – Код класса «MainWindow»

В интерфейсе преобладают элементы button, textbox, label и DataGrid. Логика в «MainWindow» вынесена в отдельный метод.

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы было разработано графическое приложения кодирования методом Хэмминга (8,4), оценки его статистических характеристик с помощью границы Хэмминга.

Проведённый расчёт продемонстрировал адекватность разработанного программного продукта, так как расчетные данные совпали с тестовыми значениями и созданное программное обеспечение удовлетворяют всем требованиям, указанным в постановке задачи.

В результате выполнения лабораторной работы было установлено, что данный метод представляет собой простую и эффективную систему кодирования, основанную на расширении кода каждой буквы в алфавите. Этот метод может быть применен в различных областях, таких как компьютерная наука, криптография.

При использовании данного метода кодирования, можно достичь значительной точности при зашумленном канале, что особенно полезно при передаче информации по сети или сохранении на диске. Кроме того, этот метод обладает простой структурой и легко реализуется в программном обеспечении.

СПИСОК ИСПОЛЬЗОВАННЫХ ПЕРВОИСТОЧНИКОВ

1 Сайт СЭОР СГУПС – Система электронных образовательных ресурсов сибирского государственного университета путей сообщения (СГУПС) Учебно-методический комплекс дисциплины «Теория информации и кодирования». / Э. А. Усова. – Новосибирск : [сайт]. – URL: http://moodle3.stu.ru/course/view.php?id=1612 (дата обращения: 03.06.2021). – Режим доступа: для авториз. пользователей. – Текст. Изображение : электронные.

2 Матвеев, Б. В. Основы корректирующего кодирования: теория и лабораторный практикум : учебное пособие / Б. В. Матвеев. — 2-е изд., стер. — Санкт-Петербург : Лань, 2021. — 192 с. — ISBN 978-5-8114-1631-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/168880 (дата обращения: 03.06.2021). — Режим доступа: для авториз. пользователей.

3 Котенко, В. В. Теория информации : учебное пособие / Котенко В. В. - Ростов н/Д : Изд-во ЮФУ, 2018. - 239 с. - ISBN 978-5-9275-2370-2. - Текст : электронный // ЭБС "Консультант студента" : [сайт]. - URL : https://www.studentlibrary.ru/book/ISBN9785927523702.html (дата обращения: 03.06.2021). - Режим доступа : по подписке.

Приложение А

Код класса ViewModelBase

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

namespace Hemchik.ViewModels.Base

{

internal abstract class ViewModelBase : INotifyPropertyChanged, IDisposable

{

public event PropertyChangedEventHandler? PropertyChanged;

protected virtual void onPropetyChanged([CallerMemberName] string PropertyName = null)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(PropertyName));

}

protected virtual bool Set<T>(ref T field, T value, [CallerMemberName] string PropertyName = null)

{

if (Equals(field, value)) return false;

field = value;

onPropetyChanged(PropertyName);

return true;

}

public void Dispose()

{

Dispose(true);

}

private bool \_disposing;

protected virtual void Dispose(bool disposing)

{

if (!disposing || !\_disposing) return;

\_disposing = true;

// Освобождение управляемых ресурсов

}

}

}

Приложение Б

Код класса table

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Controls;

namespace Hemchik.ViewModels

{

internal class table : ViewModelBase

{

public ObservableCollection<ObservableCollection<int>> matrixH

= new ObservableCollection<ObservableCollection<int>>()

{

new ObservableCollection<int> { 1, 0, 0, 0 },

new ObservableCollection<int> { 1, 1, 0, 1 },

new ObservableCollection<int> { 1, 0, 1, 1 },

new ObservableCollection<int> { 1, 1, 1, 0 },

new ObservableCollection<int> { 1, 1, 1, 1 },

new ObservableCollection<int> { 1, 1, 0, 0 },

new ObservableCollection<int> { 1, 0, 1, 0 },

new ObservableCollection<int> { 1, 0, 0, 1 }

};

public ObservableCollection<ObservableCollection<int>> matrixG

= new ObservableCollection<ObservableCollection<int>>()

{

new ObservableCollection<int> { 1, 0, 0, 0, 1, 0, 1 },

new ObservableCollection<int> { 0, 1, 0, 0, 0, 1, 1 },

new ObservableCollection<int> { 0, 0, 1, 0, 1, 1, 0 },

new ObservableCollection<int> { 0, 0, 0, 1, 1, 1, 1 }

};

private int[] MatrixConverToVector(string fourbit)

{

int[] vector = new int[7]

{0,0,0,0,0,0,0 };

for (int i=0; i<4; i++)

{

if (fourbit[i] == '1')

{

for (int j = 0; j < 7; j++)

{

vector[j] = vector[j] ^ matrixG[i][j];

}

}

}

return vector;

}

private int[] SindromCalculate(string sevenbit)

{

int[] sindrom = new int[4]

{0,0,0,0};

for (int i = 0; i < 8; i++)

{

if (eightBit[i] == '1')

{

for (int j = 0; j < 4; j++)

{

sindrom[j] = sindrom[j] ^ matrixH[i][j];

}

}

}

return sindrom;

}

private int IsBitPartity(int[] vector)

{

int partity = 1;

int summ = 0;

foreach (var item in vector)

{

summ += item;

}

if (summ % 2 == 0) partity = 0;

return partity;

}

private string VectorToString(int[] vector)

{

string final="";

foreach (var item in vector)

{

final += item.ToString();

}

return final;

}

string \_fourBit;

public string fourBit

{

get => \_fourBit;

set { Set(ref \_fourBit, value); }

}

private string \_eightBit;

public string eightBit

{

get => \_eightBit;

set { Set(ref \_eightBit, value);

}

}

private string \_complited;

public string complited

{

get => \_complited;

set

{

Set(ref \_complited, value);

}

}

private int \_partityBit;

public int partityBit

{

get => \_partityBit;

set { Set(ref \_partityBit, value); }

}

private string \_syndrome;

public string syndrome

{

get => \_syndrome;

set { Set(ref \_syndrome, value); }

}

public table(string bit4)

{

fourBit = bit4;

int[] vector = MatrixConverToVector(fourBit);

eightBit = VectorToString(vector);

partityBit = IsBitPartity(vector);

eightBit = partityBit.ToString() + eightBit;

syndrome = VectorToString(SindromCalculate(eightBit));

IsComplited();

}

public void Refresh(string el)

{

Set(ref \_eightBit, el);

syndrome = VectorToString(SindromCalculate(eightBit));

IsComplited();

}

private void IsComplited()

{

if (syndrome == "0000")

{ complited = "ошибок нет"; }

else

{

int[] sindrom = new int[4]

{0,0,0,0};

for (int i = 0; i < 4; i++)

{

sindrom[i] = int.Parse(syndrome[i].ToString());

}

bool flag = false;

for (int i = 0; i < 8; i++)

{

flag = true;

for (int j = 0; j < 4; j++)

{

if (sindrom[j] != matrixH[i][j]) flag = false;

}

if (flag == true)

{

complited = "ошибка в " + (i + 1).ToString() + " бите";

return;

}

}

}

}

}

}

Приложение В

Код разметки xaml

<Window x:Class="Hemchik.MainWindow"

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

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

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

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

xmlns:local="clr-namespace:Hemchik"

xmlns:cmd="clr-namespace:Hemchik.Commands"

xmlns:vm="clr-namespace:Hemchik.ViewModels"

mc:Ignorable="d"

Title="MainWindow" Height="450" Width="800">

<Window.DataContext>

<vm:MainWindowViewModel/>

</Window.DataContext>

<Window.Resources>

<cmd:EnterClick x:Key="EnterClick"/>

</Window.Resources>

<Grid>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition />

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<StackPanel Grid.Row="0">

<Label Margin="0,10,0,10" Content="Исходное сообщение" HorizontalAlignment="Center"/>

<TextBox Name="Input" Margin="10,0,10,10" Text="{Binding inputMessage}" KeyDown="stringLine\_KeyDown"/>

<Button Content="Выполнить" Width="150" Margin="5" Click="Button\_Click"/>

</StackPanel>

<ScrollViewer Grid.Row="1" VerticalScrollBarVisibility="Auto">

<Grid Grid.Row="1">

<DataGrid CellEditEnding="myDG\_CellEditEnding" Margin="10, 0, 10, 0" Name="DG" AutoGenerateColumns="False" CanUserAddRows="False">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding Path=fourBit}" MinWidth="80">

<DataGridTextColumn.Header>

<DataGridColumnHeader Content="4 бита" HorizontalAlignment="Center"/>

</DataGridTextColumn.Header>

</DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Path=eightBit}" MinWidth="80">

<DataGridTextColumn.Header>

<DataGridColumnHeader Content="7 бит" HorizontalAlignment="Center"/>

</DataGridTextColumn.Header>

</DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Path=partityBit}" MinWidth="80">

<DataGridTextColumn.Header>

<DataGridColumnHeader Content="бит четности" HorizontalAlignment="Center"/>

</DataGridTextColumn.Header>

</DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Path=syndrome}" MinWidth="80">

<DataGridTextColumn.Header>

<DataGridColumnHeader Content="синдром" HorizontalAlignment="Center"/>

</DataGridTextColumn.Header>

</DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Path=complited}" MinWidth="80">

<DataGridTextColumn.Header>

<DataGridColumnHeader Content="ошибка" HorizontalAlignment="Center"/>

</DataGridTextColumn.Header>

</DataGridTextColumn>

</DataGrid.Columns>

</DataGrid>

</Grid>

</ScrollViewer>

<StackPanel Grid.Row="2">

<Label Margin="0,10,0,10" Content="Граница Варшава-Гилберта" HorizontalAlignment="Center"/>

<TextBox Name="Output" IsReadOnly="True" Margin="10,0,10,20" Text="{Binding outputMessage}"/>

</StackPanel>

</Grid>

</Grid>

</Window>

Приложение Г

Код класса MainWindow

using Hemchik.ViewModels;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace Hemchik

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

ObservableCollection<table> \_table = new();

public MainWindow()

{

InitializeComponent();

DG.ItemsSource = \_table;

}

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

{

Run();

}

private void Run()

{

string final = "";

\_table.Clear();

Output.Clear();

if (Input.Text.Length > 0)

{

while (final.Length > 0)

{

string newStringFourBit = final.Substring(0,4);

table newe = new table(newStringFourBit);

\_table.Add(newe);

final = final.Substring(4);

}

DG.ItemsSource = \_table;

}

}

void myDG\_CellEditEnding(object sender, DataGridCellEditEndingEventArgs e)

{

if (e.EditAction == DataGridEditAction.Commit)

{

var column = e.Column as DataGridBoundColumn;

if (column != null)

{

var bindingPath = (column.Binding as Binding).Path.Path;

if (bindingPath == "eightBit")

{

int rowIndex = e.Row.GetIndex();

var el = e.EditingElement as TextBox;

\_table[rowIndex].Refresh(el.Text);

// rowIndex has the row index

// bindingPath has the column's binding

// el.Text has the new, user-entered value

}

}

}

}

private void stringLine\_KeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Enter) Run();

}

}

}