**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |  |
| --- | --- | --- |
| СОГЛАСОВАНО  профессор департамента программной инженерии, канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г. |  | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** | **RU.17701729.5071102 -12 01-1** | | **ПРОГРАММА ДЛЯ ШИФРОВАНИЯ И ДЕШИФРОВАНИЯ**  **С ПОМОЩЬЮ РЕШЕТКИ КАРДАНО**  **Текст программы**  **ЛИСТ УТВЕРЖДЕНИЯ**  **RU.17701729.5071102 -12 01-1-ЛУ** | | |
|  |  | |
| Исполнитель:  студент группы БПИ177  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Данилов А.А. /  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г. | |
|  | | |
|  | |  |

**2018**

|  |  |  |  |
| --- | --- | --- | --- |
| **УТВЕРЖДЕНО**  **RU.17701729.5071102 -12 01-1-ЛУ** | |  |  |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** | **RU.17701729.5071102 -12 01-1** | | **ПРОГРАММА ДЛЯ ШИФРОВАНИЯ И ДЕШИФРОВАНИЯ**  **С ПОМОЩЬЮ РЕШЕТКИ КАРДАНО**  **Текст программы**  **RU.17701729.5071102 -12 01-1**    **Листов 48** | | | | |
|  |  | | | |
|  | | | |
|  | | | | |

**2018**

# **Аннотация**

В данном документе приведет текст «Программы шифрования и дешифрования с помощью решетки Кардано». Программа разработана на языке C#.   
Среда разработки - Microsoft Visual Studio Professional 2017.

Функциональным назначением программы является генерация и сохранение алгеброгеометрических кодов, а также кодирование и декодирование некоторых сообщений с их помощью.

Настоящий документ разработан в соответствии с требованиями:

1) ГОСТ 19.101-77 Виды программ и программных документов [1];

2) ГОСТ 19.102-77 Стадии разработки [2];

3) ГОСТ 19.103-77 Обозначения программ и программных документов [3];

4) ГОСТ 19.104-78 Основные надписи [4];

5) ГОСТ 19.105-78 Общие требования к программным документам [5];

6) ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом[6];

7) ГОСТ 19.401-78 Текст программы. Требования к содержанию и оформлению [7].

Изменения к данному документу оформляются согласно ГОСТ 19.603-78 [8], ГОСТ 19.604-78 [9].

Перед прочтением данного документа рекомендуется ознакомиться с терминологией, приведенной в Приложении 1 и 2.

СОДЕРЖАНИЕ

Аннотация

[1 Текст программы.....................................................................................................................4](#_Toc514100496)

[**1.1** **Код библиотеки Language\_Library** 4](#_Toc514100497)

[**1.1.1** **Код класса Language** 4](#_Toc514100498)

[**1.2** **Код библиотеки Pttopx** 8](#_Toc514100499)

[**1.2.1** **Код класса convert** 8](#_Toc514100500)

[**1.3** **Код окон WPF** 9](#_Toc514100501)

[**1.3.1** **Код окна MainWindow** 9](#_Toc514100502)

[**1.3.2** **Код окна MainMenu** 11](#_Toc514100503)

[**1.3.3** **Код окна AboutProgramm** 13](#_Toc514100504)

[**1.3.4** **Код окна Client** 13](#_Toc514100505)

[**1.3.5** **Код окна EnteringField** 20](#_Toc514100506)

[**1.3.6** **Код окна GrillEditor** 22](#_Toc514100507)

[**1.3.7** **Код окна Help** 26](#_Toc514100508)

[**1.3.8** **Код окна InformationCoder** 27](#_Toc514100509)

[**1.3.9** **Код окна InformationDecoder** 31](#_Toc514100510)

[**1.3.10** **Код окна Server** 36](#_Toc514100511)

[**1.3.11** **Код окна ServerAndClientSettings** 42](#_Toc514100512)

[**1.3.12** **Код окна ServerOrClient** 43](#_Toc514100513)

[ПРИЛОЖЕНИЕ ........................................................................................................................45](#_Toc514100514)

[ПРИЛОЖЕНИЕ 2......................................................................................................................45](#_Toc514100515)

[ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ................................................................................47](#_Toc514100516)

# **1 Текст программы**

Программа состоит из двух библиотеки классов и двенадцати окон Windows Presentation Foundation. Далее будет приведен код, содержащийся в каждой из частей программы.

## **Код библиотеки Language\_Library**

### **Код класса Language**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Language\_Library

{

public class Language

{

public bool Rus

{

protected set;

get;

}

public Language(bool Rus) => this.Rus = Rus;

public string MainMenu

{

get => Rus ? "Главное меню" : "Main menu";

}

public string GrillEditor

{

get => Rus ? "Редактор решетки" : "Grill editor";

}

public string InformationCoder

{

get => Rus ? "Зашифровать информацию" : "Information coder";

}

public string InformationDecoder

{

get => Rus ? "Расшифровать информацию" : "Information decoder";

}

public string Chat

{

get => Rus ? "Чат" : "Chat";

}

public string AboutProgramm

{

get => Rus ? "О программе" : "About programm";

}

public string Help

{

get => Rus ? "Помощь" : "Help";

}

public string SizeOfGrill

{

get => Rus ? "Введите размер решетки: (Ширина) х (Высота)" : " Enter size of grill: (Width) x (Height)";

}

public string Continue

{

get => Rus ? "Продолжить" : "Continue";

}

public string ErrorMessage

{

get => Rus ? "Ошибка ввода" : "Input error";

}

public string Error

{

get => Rus ? "Ошибка" : "Error";

}

public string Cancle

{

get => Rus ? "Отмена" : "Cancle";

}

public string Hide

{

get => Rus ? "Скрыть все" : "Hide all";

}

public string Show

{

get => Rus ? "Показать все" : "Show all";

}

public string Save

{

get => Rus ? "Сохранить" : "Save";

}

public string Open

{

get => Rus ? "Открыть" : "Open";

}

public string KardanoGrild

{

get => Rus ? "Решетка Кардано | \*.krd" : "Cardano grill | \*.krd";

}

public string Warning

{

get => Rus ? "Предупреждение" : "Warning";

}

public string WarningMessage

{

get => Rus ?

"Размер открываемой решетки не совпадает с размером решетки для шифрования. Лишние элементы будут проигнорированы" :

"The size of the opened grid does not match the size of the encryption grid. Extra elements will be ignored";

}

public string ShowGrill

{

get => Rus ? "Показать" : "Show";

}

public string HideGrill

{

get => Rus ? "Скрыть" : "Hide";

}

public string Get

{

get => Rus ? "Получить" : "Get";

}

public string Enter

{

get => Rus ? "Ввести" : "Enter";

}

public string Chose

{

get => Rus ? "Выберите систему" : "Chose system";

}

public string serverclient

{

get => Rus ? "Сервер Клиент" : "Server Client";

}

public string NetWarning

{

get => Rus ?

"Чтобы полноценно пользоваться чатом, вы должны обладать базовыми знаниями сетевого программирования" :

"If you want to use a chat, you should know a basic knowlage of network programming";

}

public string Server

{

get => Rus ? "Сервер" : "Server";

}

public string Client

{

get => Rus ? "Клиент" : "Client";

}

public string LogIn

{

get => Rus ? "Войти" : "Log in";

}

public string Send

{

get => Rus ? "Отправить" : "Send";

}

public string Connected

{

get => Rus ? "Подключен" : "Connected";

}

public string Start

{

get => Rus ? "Запустить" : "Start";

}

public string Server\_enter

{

get => Rus ? " вошел в чат" : " connected to chat";

}

public string Server\_leave

{

get => Rus ? "покинул чат" : "left chat";

}

public string Server\_start

{

get => Rus ? "Сервер запущен. Ожидание подключений..." : "Server is on. Waiting for connectionss...";

}

public string aboutprogramm

{

get => Rus ? "Программа является курсовой работой студента образовательной программы \"Программная инженерия\" Научно-исследовательского университета \"Высшая Школа Экономики\" Даниловым Алексеем Андреевичем. Руководитель - профессор департамета программной инженерии Шилов Валерий Владимирович" :

"The program is the course work of the student of the educational program \"Software Engineering\" of the Research University \"Higher School of Economics\" Danilov Alexey Andreevich. The head - the professor of the department of software engineering Shilov Valery Vladimirovich";

}

public string Delete

{

get => Rus ? "Отчистить" : "Clear";

}

public string GrillEditorHelp

{

get => Rus ? "В редакторе решетки можно создавать свои решетки прямоугольного размера. Для того, чтобы открыть редактор нужно нажать соответствующую кнопку в главном меню, затем в окне ввести размер решетки и нажать кнопку продолжить. Для того чтобы создать решетку для шифрования нужно выбрать клетки которые будут открытыми, затем нажать кнопку сохранить, выбрать папку и сохранить под заданным именем" :

"In grill editor you can create your own rectangle grills. if you want to open editor, you should press \"Grill editor\" button, after that you should enter the size of the grill and press continue. To create coding grill, you should choose cell, which you want to open, after press Save and choose directory and name to save";

}

public string InformationCoderHelp

{

get => Rus ? "Чтобы зашифровать информацию, нужно выбрать соответствующую кнопку в главном меню, нажать ее и ввести размер решетки для шифрования, затем нажать кнопку \"Открыть\" и выбрать решетку сделанную в редакторе решетки, после в пустые места ввести текст, который нужно зашифровать, нажать кнопку \"Скрыть\" ввести текст, оставив заполненные клетки без изменения, после этого, нажать \"Получить\", скопировать текст и отправить другому пользователю для расшифрования" :

"To code information, you should press \"Information Coder\" in main menu and enter the size of the grill, after that you should open the grid and choose grill, which you made in grill editor, after that in empty cells you should enter coded text and press \"Hide\" and complete all empty cells, then press \"Get\", copy text in text box and send it to anoter user to decode";

}

public string InformationDecoderHelp

{

get => Rus ? "Чтобы расшифровать информацию, нужно выбрать соответствующую кнопку в главном меню, нажать ее и ввести размер решетки для шифрования, затем нажать кнопку \"Открыть\" и выбрать решетку сделанную в редакторе решетки, затем ввести зашифрованный текст и нажать кнопку \"Ввести\", если решетка, с помощью которой зашифровали информацию, совпадает с открытой решеткой, то мы получим правильно расшифрованное сообщение" :

"To decode information, you should press \"Information Decoder\" in main menu and enter the size of the grill, after that you should open the grid and choose grill, which you made in grill editor, after that you sould enter coded text and press \"Enter\". If Grill is right, you will get right message";

}

public string ChatHelp

{

get => Rus ? "Для того чтобы запустить сервер, вам нужно открыть порт и узнать локальный ip адрес, и сообщить глобальный ip и выбранный порт всем клиентам. Для того чтобы подключиться к серверу вам нужно ввести данные сервера(ip port) и нажать кнопку продолжить, затем в левом верхнем углу ввести свой ник и нажать кнопку \"войти\", если подключение пройдет успешно, то в чате будет написано \"Подключен\", после того как кто-то отправил сообщение вы должны скопировать его из чата от кавычек и до кавычек не включая их, затем нужно шифровать и дешифровать информацию (см Зашивровать информацию и Расшифровать информацию)"

: "To start a server, you should open port and get your local ip and give informaion about your global ip and opened port to all clients. To connect to the server, you should enter server settings(ip port) and press \"Continue\", after that in top left text box enter your nickname and press \"Log in\", if you get the message, you should copy it from \" to \" and then decode and code information (watch Information Coder, Information Decoder)";

}

public string KardanoGridInfo

{

get => Rus ? "Решётка Кардано сделана из листа картона или пергамента, или же из тонкого металла. Чтобы обозначить линии письма, бумагу разлиновывают, и между этими линиями вырезают прямоугольные области через интервалы произвольной длины. Шифрующий помещает решётку на лист бумаги и пишет сообщение в прямоугольных отверстиях, в которых помещается отдельный символ, слог или целое слово. Исходное сообщение оказывается разделённым на большое число маленьких фрагментов. Затем решётка убирается и пустые места на бумаге заполняются посторонним текстом так, чтобы скрываемый текст стал частью криптотекста. Такое заполнение требует известного литературного таланта. У получателя сообщения должна быть такая же решётка. Копии решётки вырезаются из первичного шаблона, однако для взаимно-однозначного соответствия можно было бы сделать множество других шаблонов. " :

"A Cardan grille is made from a sheet of fairly rigid paper or parchment, or from thin metal. The paper is ruled to represent lines of handwriting and rectangular areas are cut out at arbitrary intervals between these lines. An encipherer places the grille on a sheet of paper and writes his message in the rectangular apertures, some of which might allow a single letter, a syllable, or a whole word. Then, removing the grille, the fragments are filled out to create a note or letter that disguises the true message. Cardano suggested drafting the text three times in order to smooth any irregularities that might indicate the hidden words. The recipient of the message must possess an identical grille. Copies of grilles are cut from an original template, but many different patterns could be made for one-to-one correspondence.";

}

public string CardanoGrill

{

get => Rus ? "Решетка Кардано" : "Cardano grill";

}

}

}

## **Код библиотеки Pttopx**

### **Код класса convert**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Pttopx

{

static public class convert

{

static public int doit(int px)

{

if (px < 11) return 6;

else if (px < 12) return 3;

else if (px < 14) return 5;

else if (px < 15) return 5;

else if (px < 17) return 6;

else if (px < 18) return 7;

else if (px < 19) return 8;

else if (px < 21) return 9;

else if (px < 23) return 10;

else if (px < 24) return 11;

else if (px < 25) return 12;

else if (px < 26) return 13;

else if (px < 24) return 14;

else if (px < 31) return 16;

else if (px < 34) return 18;

else if (px < 37) return 20;

else if (px < 38) return 22;

else if (px < 39) return 23;

else if (px < 40) return 24;

else if (px < 42) return 25;

else if (px < 44) return 26;

else return 28;

}

}

}

## **Код окон WPF**

### **Код окна MainWindow**

using System.Windows;

using System.Windows.Input;

using Language\_Library;

using System;

using System.Diagnostics;

namespace Kardano\_Grill

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

/// <summary>

/// Страрт программы

/// </summary>

public MainWindow()

{

try

{

string[] args =

Environment.GetCommandLineArgs();

if (args[1] == "2")

{

new MainMenu(new Language(true)).Show();

Close();

}

if (args[1] == "3")

{

new MainMenu(new Language(false)).Show();

Close();

}

InitializeComponent();

}

catch

{

InitializeComponent();

}

}

/// <summary>

/// При наведении на Русский флаг делает его более прозрачным и меняет язык Lable+Title

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void RUS\_MouseEnter(object sender, MouseEventArgs e)

{

Title = "Выберите язык";

Select\_Language.Content = "Выберите язык";

RUS.Opacity = 0.5;

}

/// <summary>

/// При наведении на Британский флаг делает его более прозрачным и меняет язык Lable+Title

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void GB\_MouseEnter(object sender, MouseEventArgs e)

{

Title = "Select language";

Select\_Language.Content = "Select language";

GB.Opacity = 0.5;

}

/// <summary>

/// Когда курсор выходит из изображения Русского флага, возвращает прозрачность на максимум

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void RUS\_MouseLeave(object sender, MouseEventArgs e) => RUS.Opacity = 1;

/// <summary>

/// Когда курсор выходит из изображения Британского флага, возвращает прозрачность на максимум

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void GB\_MouseLeave(object sender, MouseEventArgs e) => GB.Opacity = 1;

/// <summary>

/// При нажатии на Русский флаг создает объект класса Language из библиотеки LanguageLibrary и запускает главное меню

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void GB\_MouseDown(object sender, MouseButtonEventArgs e)

{

MainMenu menu = new MainMenu(new Language(false));

Visibility = Visibility.Hidden;

menu.Show();

Close();

}

/// <summary>

/// При нажатии на Британский флаг создает объект класса Language из библиотеки LanguageLibrary и запускает главное меню

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void RUS\_MouseDown(object sender, MouseButtonEventArgs e)

{

MainMenu menu = new MainMenu(new Language(true));

Visibility = Visibility.Hidden;

menu.ShowDialog();

Close();

}

}

}

### **Код окна MainMenu**

using System.Windows;

using System.Windows.Input;

using Language\_Library;

namespace Kardano\_Grill

{

/// <summary>

/// Логика взаимодействия для MainMenu.xaml

/// </summary>

public partial class MainMenu : Window

{

Language classlng;

/// <summary>

/// Конструктор главного меню

/// </summary>

/// <param name="language">выбранный язык</param>

public MainMenu(Language language)

{

classlng = language;

InitializeComponent();

Title = language.MainMenu;

TypeOfProgramm.Content = language.MainMenu;

GrillEditorButton.Content = language.GrillEditor;

InformationCoderButton.Content = language.InformationCoder;

InformationDecoderButton.Content = language.InformationDecoder;

ChatButton.Content = language.Chat;

AboutProgrammButton.Content = language.AboutProgramm;

HelpButton.Content = language.Help;

}

// обработка нажатия на кнопки

private void GrillEditorButton\_Click(object sender, RoutedEventArgs e) => new EnteringField(classlng, 'E').Show();

private void InformationCoderButton\_Click(object sender, RoutedEventArgs e) => new EnteringField(classlng, 'C').Show();

private void InformationDecoderButton\_Click(object sender, RoutedEventArgs e) => new EnteringField(classlng, 'D').Show();

private void ChatButton\_Click(object sender, RoutedEventArgs e) => new EnteringField(classlng, 'S').Show();

// все что связано с закрытием программы

private void Close\_MouseEnter(object sender, MouseEventArgs e) => Close.Opacity = 0.5;

private void Close\_MouseLeave(object sender, MouseEventArgs e) => Close.Opacity = 1;

private void Close\_MouseDown(object sender, MouseButtonEventArgs e) => Close();

// все что связвно со скрытием программы

private void Hide\_MouseEnter(object sender, MouseEventArgs e) => Hide.Opacity = 0.5;

private void Hide\_MouseLeave(object sender, MouseEventArgs e) => Hide.Opacity = 1;

private void Hide\_MouseDown(object sender, MouseButtonEventArgs e) => WindowState = WindowState.Minimized;

//все что связвно с наведением на кнопки

private void GrillEditorButton\_MouseEnter(object sender, MouseEventArgs e)

{

DemostrationE.Visibility = Visibility.Visible;

DemostrationC.Visibility = Visibility.Hidden;

DemostrationD.Visibility = Visibility.Hidden;

DemostrationCH.Visibility = Visibility.Hidden;

}

private void InformationCoderButton\_MouseEnter(object sender, MouseEventArgs e)

{

DemostrationE.Visibility = Visibility.Hidden;

DemostrationC.Visibility = Visibility.Visible;

DemostrationD.Visibility = Visibility.Hidden;

DemostrationCH.Visibility = Visibility.Hidden;

}

private void InformationDecoderButton\_MouseEnter(object sender, MouseEventArgs e)

{

DemostrationE.Visibility = Visibility.Hidden;

DemostrationC.Visibility = Visibility.Hidden;

DemostrationD.Visibility = Visibility.Visible;

DemostrationCH.Visibility = Visibility.Hidden;

}

private void ChatButton\_MouseEnter(object sender, MouseEventArgs e)

{

DemostrationE.Visibility = Visibility.Hidden;

DemostrationC.Visibility = Visibility.Hidden;

DemostrationD.Visibility = Visibility.Hidden;

DemostrationCH.Visibility = Visibility.Visible;

}

/// <summary>

/// нажатие на кнопку о программе

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

new AboutProgramm(classlng).Show();

}

/// <summary>

/// нажатие на кнопку помощь

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

new Help(classlng).Show();

}

}

}

### **Код окна AboutProgramm**

using System.Windows;

using Language\_Library;

namespace Kardano\_Grill

{

/// <summary>

/// Логика взаимодействия для AboutProgramm.xaml

/// </summary>

public partial class AboutProgramm : Window

{

Language language;

public AboutProgramm( Language l)

{

language = l;

InitializeComponent();

Title = l.AboutProgramm;

Name.Content = language.CardanoGrill;

info.Text = language.aboutprogramm;

cdi.Text = language.KardanoGridInfo;

}

}

}

### **Код окна Client**

using System;

using System.Text;

using System.Threading.Tasks;

using System.Net;

using System.Net.Sockets;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;

using System.Threading;

using Language\_Library;

using Pttopx;

using Microsoft.Win32;

namespace Kardano\_Grill

{

/// <summary>

/// Логика взаимодействия для Client.xaml

/// </summary>

public partial class Client : Window

{

string[] CodedGrid;

uint H, W;

private SynchronizationContext \_context;

static string userName = "User";

IPEndPoint SERVERADRESS;

static TcpClient client = new TcpClient();

static NetworkStream stream;

Language lng;

Button[,] BBoxes;

TextBox[,] TBoxes;

/// <summary>

/// конструктор клиента

/// </summary>

/// <param name="h">высота решетки</param>

/// <param name="w">ширина решетки</param>

/// <param name="language">выбранный язык</param>

/// <param name="iPEnd">ip:port</param>

public Client(uint h,uint w,Language language,IPEndPoint iPEnd)

{

InitializeComponent();

DELETEALL.Content = language.Delete;

Title = language.Client;

code.Content = language.Get;

code.Click += Getcode;

getText.Content = language.Enter;

GText.KeyDown += EnterEv;

getText.Click += GetTextButtonClick;

ShowHide.Content = language.ShowGrill;

OpenGrid.Content = language.Open;

login.MaxLength = 20;

OpenGrid.Click += Open;

ShowHide.Click += SH;

\_context = SynchronizationContext.Current;

ipadress.Content = "IP: " + iPEnd.ToString();

SERVERADRESS = iPEnd;

Connection.Content = language.LogIn;

Send.Content = language.Send;

lng = language;

ShowHide.IsEnabled = false;

Chat.IsReadOnly = true;

Send.IsEnabled = false;

Message.IsEnabled = false;

Chat.IsEnabled = false;

H = h;

W = w;

BBoxes = new Button[H, W];

TBoxes = new TextBox[H, W];

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

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

{

TBoxes[i, j] = new TextBox

{

Height = 20,

Width = 20,

HorizontalAlignment = HorizontalAlignment.Left,

VerticalAlignment = VerticalAlignment.Top,

TextAlignment = TextAlignment.Center,

VerticalContentAlignment = VerticalAlignment.Center,

MaxLength = 1,

FontSize = convert.doit(20),

Margin = new Thickness(520 + j \* 20, 40 + i \* 20, 0, 0),

};

TBoxes[i, j].TextChanged += ku;

TBoxes[i, j].KeyUp += del;

myGrid.Children.Add(TBoxes[i, j]);

BBoxes[i, j] = new Button

{

Visibility=Visibility.Hidden,

Height = 20,

Width = 20,

HorizontalAlignment = HorizontalAlignment.Left,

VerticalAlignment = VerticalAlignment.Top,

Margin = new Thickness(520 + j \* 20, 40 + i \* 20, 0, 0),

Background = Brushes.Tan

};

myGrid.Children.Add(BBoxes[i, j]);

}

WindowState = WindowState.Maximized;

}

private void GetTextButtonClick(object sender, RoutedEventArgs e)

{

foreach (TextBox tex in TBoxes)

tex.Text = "";

try

{

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

{

if (GText.Text[i] != ' ')

TBoxes[i / W, i % W].Text = GText.Text[i].ToString();

}

}

catch (IndexOutOfRangeException)

{ }

GText.Text = "";

}

/// <summary>

/// метод для получения сообщений от сервера

/// </summary>

private void ReciveMessage()

{

while (true)

{

byte[] data = new byte[H\*W]; // буфер для получаемых данных

StringBuilder builder = new StringBuilder();

int bytes = 0;

do

{

bytes = stream.Read(data, 0, data.Length);

builder.Append(Encoding.Unicode.GetString(data, 0, bytes));

}

while (stream.DataAvailable);

string message = builder.ToString();

\_context.Post(delegate (object state)

{

Chat.AppendText(message+"\n");

}, null);

}

}

/// <summary>

/// метод для подключения к серверу

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Connect(object sender, RoutedEventArgs e)

{

try

{

client.Connect(SERVERADRESS); //подключение клиента

stream = client.GetStream(); // получаем поток

userName = login.Text;

string message = userName;

byte[] data = Encoding.Unicode.GetBytes(message);

stream.Write(data, 0, data.Length);

Task a = new Task(ReciveMessage);

a.Start();

Connection.IsEnabled = false;

login.IsEnabled = false;

Chat.IsEnabled = true;

Message.IsEnabled = true;

Send.IsEnabled = true;

Message.MaxLength = (int)(H \* W);

Chat.Text = lng.Connected + "\n";

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// отправка сообщения на сервер

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

try

{

string mes = Message.Text;

byte[] data = Encoding.Unicode.GetBytes(mes);

stream.Write(data, 0, data.Length);

Chat.AppendText(userName + ": \"" + mes + "\"\n");

Message.Text = "";

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// отправка сообщения на сервер

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

if (e.Key==Key.Enter)

SendMessage\_Click(sender, e);

}

/// <summary>

/// задание размера решетки относительно размера окна

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Window\_SizeChanged(object sender, SizeChangedEventArgs e)

{

if ((ActualHeight - 170) / H > (ActualWidth - 620) / W)

{

int s = (int)((ActualWidth - 620) / W);

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

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

{

BBoxes[i, j].Height = s;

BBoxes[i, j].Width = s;

BBoxes[i, j].Margin = new Thickness(520 + j \* s, 40 + i \* s, 0, 0);

TBoxes[i, j].Height = s;

TBoxes[i, j].Width = s;

TBoxes[i, j].Margin = new Thickness(520 + j \* s, 40 + i \* s, 0, 0);

TBoxes[i, j].FontSize = convert.doit(s);

}

}

else

{

int s = (int)((ActualHeight - 170) / H);

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

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

{

BBoxes[i, j].Height = s;

BBoxes[i, j].Width = s;

BBoxes[i, j].Margin = new Thickness(520 + j \* s, 40 + i \* s, 0, 0);

TBoxes[i, j].Height = s;

TBoxes[i, j].Width = s;

TBoxes[i, j].Margin = new Thickness(520 + j \* s, 40 + i \* s, 0, 0);

TBoxes[i, j].FontSize = convert.doit(s);

}

}

}

//все что связано с закрытием окна

private void Close\_MouseEnter(object sender, MouseEventArgs e) => Close.Opacity = 0.5;

private void Close\_MouseLeave(object sender, MouseEventArgs e) => Close.Opacity = 1;

private void Close\_MouseDown(object sender, MouseButtonEventArgs e) => Close();

//все что связвно со скрытием окна

private void Hide\_MouseEnter(object sender, MouseEventArgs e) => Hide.Opacity = 0.5;

private void Hide\_MouseLeave(object sender, MouseEventArgs e) => Hide.Opacity = 1;

private void Hide\_MouseDown(object sender, MouseButtonEventArgs e) => WindowState = WindowState.Minimized;

/// <summary>

/// открытие решетки

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

public void Open(object sender, EventArgs e)

{

OpenFileDialog Open = new OpenFileDialog();

Open.DefaultExt = "krd";

Open.Filter = lng.KardanoGrild;

Open.ShowDialog();

try

{

CodedGrid = System.IO.File.ReadAllLines(Open.FileName);

}

catch (ArgumentException)

{ }

if (CodedGrid != null)

{

if (CodedGrid.Length != H || CodedGrid[CodedGrid.Length - 1].Length != W)

MessageBox.Show(lng.WarningMessage, lng.Warning, MessageBoxButton.OK, MessageBoxImage.Warning);

ShowHide.IsEnabled = true;

SH(sender, e);

}

}

/// <summary>

/// показыть/скрыть решетку

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

public void SH(object sender, EventArgs e)

{

if (ShowHide.Content == lng.ShowGrill)

{

uint HG = H > (uint)CodedGrid.Length ? (uint)CodedGrid.Length : H;

uint WG = W > (uint)CodedGrid[CodedGrid.Length - 1].Length ? (uint)CodedGrid[CodedGrid.Length - 1].Length : W;

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

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

if (CodedGrid[i][j] == '0') BBoxes[i, j].Visibility = Visibility.Visible;

ShowHide.Content = lng.HideGrill;

}

else

{

foreach (Button but in BBoxes) but.Visibility = Visibility.Hidden;

ShowHide.Content = lng.ShowGrill;

}

}

/// <summary>

/// ввести текст

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void EnterEv(object sender, KeyEventArgs e)

{

if (e.Key == Key.Enter)

{

foreach (TextBox tex in TBoxes)

tex.Text = "";

try

{

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

{

if (GText.Text[i] != ' ')

TBoxes[i / W, i % W].Text = GText.Text[i].ToString();

}

}

catch (IndexOutOfRangeException)

{ }

GText.Text = "";

}

}

/// <summary>

/// очистить все ячейки

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

foreach (TextBox tex in TBoxes)

tex.Text = "";

}

/// <summary>

/// получить текст с ячеек

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Getcode(object sender, EventArgs e)

{

Message.Text = "";

GText.Text = "";

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

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

if (TBoxes[i, j].Text == "") Message.Text += " "; else Message.Text += TBoxes[i, j].Text;

}

//для удобства печати текст

private void ku(object sender, TextChangedEventArgs e)

{

if ((sender as TextBox).Text!="")

{

TraversalRequest request = new TraversalRequest(FocusNavigationDirection.Next);

((TextBox)sender).MoveFocus(request);

}

else

{

TraversalRequest request = new TraversalRequest(FocusNavigationDirection.Previous);

((TextBox)sender).MoveFocus(request);

}

}

private void del(object sender, KeyEventArgs e)

{

if (e.Key == Key.Back && (sender as TextBox).Text == "")

{

TraversalRequest request = new TraversalRequest(FocusNavigationDirection.Previous);

((TextBox)sender).MoveFocus(request);

}

if (e.Key != Key.Back && (sender as TextBox).Text != "" && e.Key != Key.Tab)

{

TraversalRequest request = new TraversalRequest(FocusNavigationDirection.Next);

((TextBox)sender).MoveFocus(request);

}

}

}

}

### **Код окна EnteringField**

using System.Windows;

using System.Windows.Input;

using Language\_Library;

namespace Kardano\_Grill

{

/// <summary>

/// Логика взаимодействия для EnteringField.xaml

/// </summary>

public partial class EnteringField : Window

{

Language L;

char TypeOfWindow;

/// <summary>

/// конструктор выбора размера решетки

/// </summary>

/// <param name="language"></param>

/// <param name="symbol"></param>

public EnteringField(Language language,char symbol)

{

L = language;

TypeOfWindow = symbol;

InitializeComponent();

title.Content = language.SizeOfGrill;

ContinueButton.Content = language.Continue;

switch (symbol)

{

case 'E':

Title = language.GrillEditor;

break;

case 'C':

Title = language.InformationCoder;

break;

case 'D':

Title = language.InformationDecoder;

break;

case 'S':

Title = language.Chat;

break;

}

Wigth.Focus();

}

/// <summary>

/// нажатие на кнопку продолжить

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void ContinueButton\_Click(object sender, RoutedEventArgs e) => CreateNewWindow();

/// <summary>

/// открытие нового окна

/// </summary>

private void CreateNewWindow()

{

bool c = true;

uint h = 0, w = 0;

if (!uint.TryParse(Wigth.Text, out w) || !uint.TryParse(Wigth\_Copy.Text, out h)) c = false;

if (h > 20 || h < 1 || w > 50 || w < 1) c = false;

if (!c)

{

MessageBox.Show(L.ErrorMessage, L.Error, MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

//Visibility = Visibility.Hidden;

if (TypeOfWindow == 'E')

new GrillEditor(h, w, L).Show();

if (TypeOfWindow == 'C')

new InformationCoder(h, w, L).Show();

if (TypeOfWindow == 'D')

new InformationDecoder(h, w, L).Show();

if (TypeOfWindow == 'S')

{

MessageBox.Show(L.NetWarning, L.Warning, MessageBoxButton.OK, MessageBoxImage.Warning);

new ServerOrClient(h, w, L).Show();

}

Close();

}

/// <summary>

/// метод для удобства запуска

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

if (e.Key == Key.Enter)

CreateNewWindow();

}

/// <summary>

/// метод для удобства запуска

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Wigth\_Copy\_KeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Enter)

CreateNewWindow();

}

}

}

### **Код окна GrillEditor**

using System;

using System.Collections.Generic;

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.Shapes;

using Pttopx;

using Language\_Library;

using Microsoft.Win32;

namespace Kardano\_Grill

{

/// <summary>

/// Логика взаимодействия для GrillEditor.xaml

/// </summary>

public partial class GrillEditor : Window

{

double BaseHeight, BaseWidth;

Language l;

Button[,] BBoxes;

TextBox[,] TBoxes;

Button cancle, HideShow, buffer, save;

uint H, W;

/// <summary>

/// конструктор редактора решетки

/// </summary>

/// <param name="h">высота решетки</param>

/// <param name="w">ширина решетки</param>

/// <param name="language">выбранный язык</param>

public GrillEditor(uint h,uint w,Language language)

{

l = language;

InitializeComponent();

Title = l.GrillEditor;

BaseWidth = 150 + w \* 20;

if ((h + 1) \* 20+40 > 140) BaseHeight = (h + 1) \* 20+40; else BaseHeight = 140;

MinHeight= BaseHeight;

MinWidth = BaseWidth;

HideShow = new Button

{

Height = 20,

Width = 100,

HorizontalAlignment = HorizontalAlignment.Right,

VerticalAlignment = VerticalAlignment.Top,

Margin = new Thickness(0, 40, 10, 0)

};

HideShow.Content = l.Show;

myGrid.Children.Add(HideShow);

HideShow.Click += HideShowall;

cancle = new Button

{

Height = 20,

Width = 100,

HorizontalAlignment = HorizontalAlignment.Right,

VerticalAlignment = VerticalAlignment.Top,

IsEnabled = false,

Margin = new Thickness(0, 10, 10, 0)

};

cancle.Content = l.Cancle;

myGrid.Children.Add(cancle);

cancle.Click += Cancle;

save = new Button

{

Height = 20,

Width = 100,

HorizontalAlignment = HorizontalAlignment.Right,

VerticalAlignment = VerticalAlignment.Bottom,

Margin= new Thickness(0,0,10,10)

};

save.Content = l.Save;

myGrid.Children.Add(save);

save.Click += Save;

H = h;

W = w;

BBoxes = new Button[H, W];

TBoxes = new TextBox[H, W];

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

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

{

TBoxes[i, j] = new TextBox

{

HorizontalAlignment = HorizontalAlignment.Left,

VerticalAlignment = VerticalAlignment.Top,

TextAlignment = TextAlignment.Center,

VerticalContentAlignment = VerticalAlignment.Center,

MaxLength = 1,

FontSize = convert.doit(20),

};

myGrid.Children.Add(TBoxes[i, j]);

BBoxes[i, j] = new Button

{

HorizontalAlignment = HorizontalAlignment.Left,

VerticalAlignment = VerticalAlignment.Top,

Background = Brushes.Tan

};

BBoxes[i, j].Click += ButtonClickEvent;

myGrid.Children.Add(BBoxes[i, j]);

}

}

/// <summary>

/// обработка события при изменении размера окна

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Window\_SizeChanged(object sender, SizeChangedEventArgs e)

{

if ((ActualHeight-60)/H>(ActualWidth - 150)/W)

{

int s = (int)((ActualWidth - 150) / W);

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

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

{

BBoxes[i, j].Height = s;

BBoxes[i, j].Width = s;

BBoxes[i, j].Margin = new Thickness(10 + j \* s, 10 + i \* s, 0, 0);

TBoxes[i, j].Height = s;

TBoxes[i, j].Width = s;

TBoxes[i, j].Margin = new Thickness(10 + j \* s, 10 + i \* s, 0, 0);

TBoxes[i, j].FontSize = convert.doit(s);

}

}

else

{

int s = (int)((ActualHeight - 60) / H);

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

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

{

BBoxes[i, j].Height = s;

BBoxes[i, j].Width = s;

BBoxes[i, j].Margin = new Thickness(10 + j \* s, 10 + i \* s, 0, 0);

TBoxes[i, j].Height = s;

TBoxes[i, j].Width = s;

TBoxes[i, j].Margin = new Thickness(10 + j \* s, 10 + i \* s, 0, 0);

TBoxes[i, j].FontSize = convert.doit(s);

}

}

}

/// <summary>

/// обработка события на нажатие показать/скрыть

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void HideShowall(object sender, EventArgs e)

{

if ((sender as Button).Content.ToString() == l.Hide)

{

foreach (Button but in BBoxes)

but.Visibility = Visibility.Hidden;

(sender as Button).Content = l.Show;

}

else

{

foreach (Button but in BBoxes)

but.Visibility = Visibility.Visible;

(sender as Button).Content = l.Hide;

}

}

/// <summary>

/// обработка нажатия на клетку решетки

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void ButtonClickEvent(object sender, EventArgs e)

{

(sender as Button).Visibility = Visibility.Hidden;

buffer = sender as Button;

cancle.IsEnabled = true;

}

/// <summary>

/// обработка нажатия на кнопку отмена

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Cancle(object sender,EventArgs e)

{

if (buffer.Visibility == Visibility.Hidden)

buffer.Visibility = Visibility.Visible;

else

buffer.Visibility = Visibility.Hidden;

}

/// <summary>

/// сохранение решетки

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Save(object sender, EventArgs e)

{

string[] file = new string[H];

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

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

if (BBoxes[i, j].Visibility == Visibility.Hidden)

file[i] += 1;

else

file[i] += 0;

SaveFileDialog SaveTXT = new SaveFileDialog();

SaveTXT.DefaultExt = "krd";

SaveTXT.Filter = l.KardanoGrild;

SaveTXT.ShowDialog();

try

{

System.IO.File.WriteAllLines(SaveTXT.FileName, file);

base.Close();

}

catch (ArgumentException)

{ }

}

}

}

### **Код окна Help**

using System.Windows;

using Language\_Library;

namespace Kardano\_Grill

{

/// <summary>

/// Логика взаимодействия для Help.xaml

/// </summary>

public partial class Help : Window

{

Language lng;

public Help(Language l)

{

lng = l;

InitializeComponent();

Title = l.Help;

GrillEditor.Content = lng.GrillEditor;

GrillEditorHelp.Text = lng.GrillEditorHelp;

InformationCoder.Content = lng.InformationCoder;

InformationCoderHelp.Text = lng.InformationCoderHelp;

InformationDecoder.Content = lng.InformationDecoder;

InformationDecoderHelp.Text = lng.InformationDecoderHelp;

Chat.Content = lng.Chat;

ChatHelp.Text = lng.ChatHelp;

}

}

}

### **Код окна InformationCoder**

using System;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using Language\_Library;

using Pttopx;

using Microsoft.Win32;

namespace Kardano\_Grill

{

/// <summary>

/// Логика взаимодействия для InformationCoder.xaml

/// </summary>

public partial class InformationCoder : Window

{

double BaseHeight, BaseWidth;

Language language;

TextBox[,] Tboxes;

Button[,] Bboxes;

uint H, W;

Button ShowHide;

TextBox GText;

string[] CodedGrid;

/// <summary>

/// конструктор для зашифрования информации

/// </summary>

/// <param name="h">вытора решетки</param>

/// <param name="w">ширина решетки</param>

/// <param name="l">выбранный язак</param>

public InformationCoder(uint h,uint w, Language l)

{

language = l;

H = h;

W = w;

InitializeComponent();

Title = language.InformationCoder;

BaseWidth = 150 + w \* 20;

if ((h + 1) \* 20 + 40 > 140) BaseHeight = (h + 2) \* 20 + 50; else BaseHeight = 150;

if (150 + w \* 20 > 250) BaseWidth = 150 + w \* 20; else BaseWidth = 250;

MinHeight = BaseHeight;

MinWidth = BaseWidth;

Button OpenGrid = new Button

{

Height = 20,

Width = 100,

Content = language.Open,

HorizontalAlignment = HorizontalAlignment.Right,

VerticalAlignment = VerticalAlignment.Top,

Margin= new Thickness(0,10,10,0),

};

ShowHide = new Button

{

Height = 20,

Width = 100,

Content = language.ShowGrill,

HorizontalAlignment = HorizontalAlignment.Right,

VerticalAlignment = VerticalAlignment.Top,

Margin = new Thickness(0, 40, 10, 0),

IsEnabled = false,

};

ShowHide.Click += SH;

myGrid.Children.Add(ShowHide);

OpenGrid.Click += Open;

myGrid.Children.Add(OpenGrid);

Tboxes = new TextBox[H, W];

Bboxes = new Button[H, W];

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

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

{

Tboxes[i, j] = new TextBox

{

Height = 20,

Width = 20,

TextAlignment=TextAlignment.Center,

VerticalContentAlignment=VerticalAlignment.Center,

VerticalAlignment = VerticalAlignment.Top,

HorizontalAlignment = HorizontalAlignment.Left,

MaxLength = 1,

FontSize = convert.doit(20),

Margin = new Thickness(10 + j \* 20, 10 + i \* 20, 0, 0)

};

Tboxes[i, j].TextChanged += kd;

Tboxes[i, j].KeyUp += del;

myGrid.Children.Add(Tboxes[i, j]);

Bboxes[i, j] = new Button

{

Height = 20,

Width = 20,

HorizontalAlignment = HorizontalAlignment.Left,

VerticalAlignment = VerticalAlignment.Top,

Margin = new Thickness(10 + j \* 20, 10 + i \* 20, 0, 0),

Visibility=Visibility.Hidden

};

Bboxes[i, j].Background = System.Windows.Media.Brushes.Tan;

myGrid.Children.Add(Bboxes[i, j]);

}

GText = new TextBox

{

Height = 20,

Width = W > 5 ? W \* 20 : 100,

VerticalAlignment = VerticalAlignment.Bottom,

HorizontalAlignment = HorizontalAlignment.Left,

IsReadOnly = true,

FontSize = convert.doit(20),

Margin = new Thickness(10, 0, 0, 10)

};

myGrid.Children.Add(GText);

Button enter = new Button

{

Height = 20,

Width = 100,

VerticalAlignment = VerticalAlignment.Bottom,

HorizontalAlignment = HorizontalAlignment.Right,

Margin = new Thickness(0, 0, 10, 10),

Content = language.Get

};

enter.Click += GetTextButtonClick;

myGrid.Children.Add(enter);

}

/// <summary>

/// обработка нажатия на кнопку получить

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void GetTextButtonClick(object sender, EventArgs e)

{

GText.Text = "";

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

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

if (Tboxes[i, j].Text == "") GText.Text += " "; else GText.Text += Tboxes[i, j].Text;

}

/// <summary>

/// обработка события при изменении размера окна

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Window\_SizeChanged(object sender, SizeChangedEventArgs e)

{

if ((ActualHeight - 90) / H > (ActualWidth - 150) / W)

{

int s = (int)((ActualWidth - 150) / W);

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

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

{

Bboxes[i, j].Height = s;

Bboxes[i, j].Width = s;

Bboxes[i, j].Margin = new Thickness(10 + j \* s, 10 + i \* s, 0, 0);

Tboxes[i, j].Height = s;

Tboxes[i, j].Width = s;

Tboxes[i, j].Margin = new Thickness(10 + j \* s, 10 + i \* s, 0, 0);

Tboxes[i, j].FontSize = convert.doit(s);

}

GText.Width = 100 > W \* s ? 100 : W \* s;

}

else

{

int s = (int)((ActualHeight - 90) / H);

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

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

{

Bboxes[i, j].Height = s;

Bboxes[i, j].Width = s;

Bboxes[i, j].Margin = new Thickness(10 + j \* s, 10 + i \* s, 0, 0);

Tboxes[i, j].Height = s;

Tboxes[i, j].Width = s;

Tboxes[i, j].Margin = new Thickness(10 + j \* s, 10 + i \* s, 0, 0);

Tboxes[i, j].FontSize = convert.doit(s);

}

GText.Width = 100 > W \* s ? 100 : W \* s;

}

}

/// <summary>

/// открытие решетки

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

public void Open(object sender, EventArgs e)

{

OpenFileDialog Open = new OpenFileDialog();

Open.DefaultExt = "krd";

Open.Filter = language.KardanoGrild;

Open.ShowDialog();

try

{

CodedGrid = System.IO.File.ReadAllLines(Open.FileName);

}

catch (ArgumentException)

{ }

if (CodedGrid != null)

{

if (CodedGrid.Length!=H || CodedGrid[CodedGrid.Length-1].Length!=W)

MessageBox.Show(language.WarningMessage, language.Warning, MessageBoxButton.OK, MessageBoxImage.Warning);

ShowHide.IsEnabled = true;

SH(sender, e);

}

}

/// <summary>

/// нажатие на показать/скрыть

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

public void SH(object sender, EventArgs e)

{

if (ShowHide.Content == language.ShowGrill)

{

uint HG = H > (uint)CodedGrid.Length ? (uint)CodedGrid.Length : H;

uint WG = W > (uint)CodedGrid[CodedGrid.Length - 1].Length ? (uint)CodedGrid[CodedGrid.Length - 1].Length : W;

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

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

if (CodedGrid[i][j] == '0') Bboxes[i, j].Visibility = Visibility.Visible;

ShowHide.Content = language.HideGrill;

}

else

{

foreach (Button but in Bboxes) but.Visibility = Visibility.Hidden;

ShowHide.Content = language.ShowGrill;

}

}

/// <summary>

/// для удобства печати текста(1)

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void del(object sender, KeyEventArgs e)

{

if (e.Key==Key.Back && (sender as TextBox).Text=="")

{

TraversalRequest request = new TraversalRequest(FocusNavigationDirection.Previous);

((TextBox)sender).MoveFocus(request);

}

if (e.Key != Key.Back && (sender as TextBox).Text != "" && e.Key!=Key.Tab)

{

TraversalRequest request = new TraversalRequest(FocusNavigationDirection.Next);

((TextBox)sender).MoveFocus(request);

}

}

/// <summary>

/// для удобства печати (2)

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void kd(object sender, TextChangedEventArgs e)

{

if ((sender as TextBox).Text!="")

{

TraversalRequest request = new TraversalRequest(FocusNavigationDirection.Next);

((TextBox)sender).MoveFocus(request);

}

else

{

TraversalRequest request = new TraversalRequest(FocusNavigationDirection.Previous);

((TextBox)sender).MoveFocus(request);

}

}

}

}

### **Код окна InformationDecoder**

using System;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using Language\_Library;

using Pttopx;

using Microsoft.Win32;

namespace Kardano\_Grill

{

/// <summary>

/// Логика взаимодействия для InformationDecoder.xaml

/// </summary>

public partial class InformationDecoder : Window

{

double BaseHeight, BaseWidth;

Language language;

TextBox[,] Tboxes;

Button[,] Bboxes;

uint H, W;

Button ShowHide;

TextBox GText;

string[] CodedGrid;

/// <summary>

/// конструктор для дешифрования информации

/// </summary>

/// <param name="h"></param>

/// <param name="w"></param>

/// <param name="l"></param>

public InformationDecoder(uint h,uint w,Language l)

{

language = l;

H = h;

W = w;

InitializeComponent();

Title = language.InformationDecoder;

BaseWidth = 150 + w \* 20;

if ((h + 1) \* 20 + 40 > 140) BaseHeight = (h + 2) \* 20 + 50; else BaseHeight = 150;

if (150 + w \* 20 > 250) BaseWidth = 150 + w \* 20; else BaseWidth = 250;

MinHeight = BaseHeight;

MinWidth = BaseWidth;

Button OpenGrid = new Button

{

Height = 20,

Width = 100,

Content = language.Open,

HorizontalAlignment = HorizontalAlignment.Right,

VerticalAlignment = VerticalAlignment.Top,

Margin = new Thickness(0, 10, 10, 0),

};

ShowHide = new Button

{

Height = 20,

Width = 100,

Content = language.ShowGrill,

HorizontalAlignment = HorizontalAlignment.Right,

VerticalAlignment = VerticalAlignment.Top,

Margin = new Thickness(0, 40, 10, 0),

IsEnabled = false,

};

ShowHide.Click += SH;

myGrid.Children.Add(ShowHide);

OpenGrid.Click += Open;

myGrid.Children.Add(OpenGrid);

Tboxes = new TextBox[H, W];

Bboxes = new Button[H, W];

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

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

{

Tboxes[i, j] = new TextBox

{

Height = 20,

Width = 20,

TextAlignment = TextAlignment.Center,

VerticalContentAlignment = VerticalAlignment.Center,

VerticalAlignment = VerticalAlignment.Top,

HorizontalAlignment = HorizontalAlignment.Left,

MaxLength = 1,

FontSize = convert.doit(20),

Margin = new Thickness(10 + j \* 20, 10 + i \* 20, 0, 0)

};

myGrid.Children.Add(Tboxes[i, j]);

Bboxes[i, j] = new Button

{

Height = 20,

Width = 20,

HorizontalAlignment = HorizontalAlignment.Left,

VerticalAlignment = VerticalAlignment.Top,

Margin = new Thickness(10 + j \* 20, 10 + i \* 20, 0, 0),

Visibility = Visibility.Hidden

};

Bboxes[i, j].Background = System.Windows.Media.Brushes.Tan;

myGrid.Children.Add(Bboxes[i, j]);

}

GText = new TextBox

{

Height = 20,

Width = W > 5 ? W \* 20 : 100,

VerticalAlignment = VerticalAlignment.Bottom,

HorizontalAlignment = HorizontalAlignment.Left,

FontSize = convert.doit(20),

MaxLength=(int)(H\*W),

Margin = new Thickness(10, 0, 0, 10)

};

myGrid.Children.Add(GText);

Button enter = new Button

{

Height = 20,

Width = 100,

VerticalAlignment = VerticalAlignment.Bottom,

HorizontalAlignment = HorizontalAlignment.Right,

Margin = new Thickness(0, 0, 10, 10),

Content = language.Enter

};

enter.Click += GetTextButtonClick;

myGrid.Children.Add(enter);

}

/// <summary>

/// нажатие на кнопку ввести

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void GetTextButtonClick(object sender, EventArgs e) => ENTERTEXT();

/// <summary>

/// метод для получения зашифрованного текста

/// </summary>

private void ENTERTEXT()

{

try

{

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

{

if (GText.Text[i] != ' ')

Tboxes[i / W, i % W].Text = GText.Text[i].ToString();

}

}

catch (IndexOutOfRangeException)

{ }

GText.Text = "";

}

/// <summary>

/// обработка события по изменению размера окна

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Window\_SizeChanged(object sender, SizeChangedEventArgs e)

{

if ((ActualHeight - 90) / H > (ActualWidth - 150) / W)

{

int s = (int)((ActualWidth - 150) / W);

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

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

{

Bboxes[i, j].Height = s;

Bboxes[i, j].Width = s;

Bboxes[i, j].Margin = new Thickness(10 + j \* s, 10 + i \* s, 0, 0);

Tboxes[i, j].Height = s;

Tboxes[i, j].Width = s;

Tboxes[i, j].Margin = new Thickness(10 + j \* s, 10 + i \* s, 0, 0);

Tboxes[i, j].FontSize = convert.doit(s);

}

GText.Width = 100 > W \* s ? 100 : W \* s;

}

else

{

int s = (int)((ActualHeight - 90) / H);

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

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

{

Bboxes[i, j].Height = s;

Bboxes[i, j].Width = s;

Bboxes[i, j].Margin = new Thickness(10 + j \* s, 10 + i \* s, 0, 0);

Tboxes[i, j].Height = s;

Tboxes[i, j].Width = s;

Tboxes[i, j].Margin = new Thickness(10 + j \* s, 10 + i \* s, 0, 0);

Tboxes[i, j].FontSize = convert.doit(s);

}

GText.Width = 100 > W \* s ? 100 : W \* s;

}

}

/// <summary>

/// обработка нажатия на кнопу открыть для открытия решетки

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Open(object sender, EventArgs e)

{

OpenFileDialog Open = new OpenFileDialog();

Open.DefaultExt = "krd";

Open.Filter = language.KardanoGrild;

Open.ShowDialog();

try

{

CodedGrid = System.IO.File.ReadAllLines(Open.FileName);

}

catch (ArgumentException)

{ }

if (CodedGrid != null)

{

if (CodedGrid.Length != H || CodedGrid[CodedGrid.Length - 1].Length != W)

MessageBox.Show(language.WarningMessage, language.Warning, MessageBoxButton.OK, MessageBoxImage.Warning);

ShowHide.IsEnabled = true;

SH(sender, e);

}

}

/// <summary>

/// показыание скрытие решетки

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void SH(object sender, EventArgs e)

{

if (ShowHide.Content == language.ShowGrill)

{

uint HG = H > (uint)CodedGrid.Length ? (uint)CodedGrid.Length : H;

uint WG = W > (uint)CodedGrid[CodedGrid.Length - 1].Length ? (uint)CodedGrid[CodedGrid.Length - 1].Length : W;

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

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

if (CodedGrid[i][j] == '0') Bboxes[i, j].Visibility = Visibility.Visible;

ShowHide.Content = language.HideGrill;

}

else

{

foreach (Button but in Bboxes) but.Visibility = Visibility.Hidden;

ShowHide.Content = language.ShowGrill;

}

}

/// <summary>

/// нажатие Enter на textbox

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void EnterEv(object sender, KeyEventArgs e)

{

if (e.Key == Key.Enter)

ENTERTEXT();

}

}

}

### **Код окна Server**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Net;

using System.Net.Sockets;

using Language\_Library;

using System.Diagnostics;

namespace Kardano\_Grill

{

/// <summary>

/// Логика взаимодействия для Server.xaml

/// </summary>

public partial class Server : Window

{

static private SynchronizationContext \_context;

static Language language;

static TextBox

Chat = new TextBox

{

Margin = new Thickness(10, 10, 10, 40),

TextWrapping = TextWrapping.Wrap,

IsReadOnly=true

},

command = new TextBox

{

Height = 23,

Margin = new Thickness(10, 0, 90, 10),

TextWrapping = TextWrapping.Wrap,

VerticalAlignment = VerticalAlignment.Bottom

};

static Button

send = new Button

{

HorizontalAlignment = HorizontalAlignment.Right,

Margin = new Thickness(0, 0, 10, 10),

VerticalAlignment = VerticalAlignment.Bottom,

Width = 75,

Height = 23

};

static IPEndPoint IP;

static int H, W;

/// <summary>

/// класс подключенных клиентов

/// </summary>

public class ClientObject

{

public string Id { get; private set; }

public NetworkStream Stream { get; private set; }

string userName;

TcpClient client;

ServerObject server; // объект сервера

public ClientObject(TcpClient tcpClient, ServerObject serverObject)

{

Id = Guid.NewGuid().ToString();

client = tcpClient;

server = serverObject;

serverObject.AddConnection(this);

}

/// <summary>

/// процесс клиента

/// </summary>

public void Process()

{

try

{

Stream = client.GetStream();

// получаем имя пользователя

string message = GetMessage();

userName = message;

message = userName + language.Server\_enter;

// посылаем сообщение о входе в чат всем подключенным пользователям

server.BroadcastMessage(message, this.Id);

\_context.Post(delegate (object state)

{

Chat.AppendText(message + "\n");

}, null);

// в бесконечном цикле получаем сообщения от клиента

while (true)

{

try

{

message = GetMessage();

message = String.Format("{0}: \"{1}\"", userName, message);

\_context.Post(delegate (object state)

{

Chat.AppendText(message + "\n");

}, null);

server.BroadcastMessage(message, this.Id);

}

catch

{

message = String.Format("{0}: {1}", userName,language.Server\_leave);

\_context.Post(delegate (object state)

{

Chat.AppendText(message + "\n");

}, null);

server.BroadcastMessage(message, this.Id);

break;

}

}

}

catch (Exception e)

{

\_context.Post(delegate (object state)

{

Chat.AppendText(e.Message + "\n");

}, null);

}

finally

{

// в случае выхода из цикла закрываем ресурсы

server.RemoveConnection(this.Id);

Close();

}

}

// чтение входящего сообщения и преобразование в строку

private string GetMessage()

{

byte[] data = new byte[H\*W]; // буфер для получаемых данных

StringBuilder builder = new StringBuilder();

int bytes = 0;

do

{

bytes = Stream.Read(data, 0, data.Length);

builder.Append(Encoding.Unicode.GetString(data, 0, bytes));

}

while (Stream.DataAvailable);

return builder.ToString();

}

// закрытие подключения

public void Close()

{

if (Stream != null)

Stream.Close();

if (client != null)

client.Close();

}

}

/// <summary>

/// класс клиента

/// </summary>

public class ServerObject

{

static TcpListener tcpListener; // сервер для прослушивания

public List<ClientObject> clients = new List<ClientObject>(); // все подключения

public void AddConnection(ClientObject clientObject)

{

clients.Add(clientObject);

}

public void RemoveConnection(string id)

{

// получаем по id закрытое подключение

ClientObject client = clients.FirstOrDefault(c => c.Id == id);

// и удаляем его из списка подключений

if (client != null)

clients.Remove(client);

}

// прослушивание входящих подключений

public void Listen()

{

try

{

tcpListener = new TcpListener(IP);

try

{

tcpListener.Start();

}

catch (SocketException ex)

{

MessageBox.Show(ex.Message, language.Error, MessageBoxButton.OK, MessageBoxImage.Error);

Disconnect();

}

\_context.Post(delegate (object state)

{

Chat.AppendText(language.Server\_start + "\n");

}, null);

while (true)

{

TcpClient tcpClient = tcpListener.AcceptTcpClient();

ClientObject clientObject = new ClientObject(tcpClient, this);

Thread clientThread = new Thread(new ThreadStart(clientObject.Process));

clientThread.Start();

}

}

catch (Exception ex)

{

Disconnect();

}

}

// трансляция сообщения подключенным клиентам

public void BroadcastMessage(string message, string id)

{

byte[] data = Encoding.Unicode.GetBytes(message);

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

{

if (clients[i].Id != id) // если id клиента не равно id отправляющего

{

clients[i].Stream.Write(data, 0, data.Length); //передача данных

}

}

}

// отключение всех клиентов

public void Disconnect()

{

tcpListener.Stop(); //остановка сервера

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

{

clients[i].Close(); //отключение клиента

}

}

}

static ServerObject server; // сервер

static Thread listenThread;

/// <summary>

/// конструктор окна

/// </summary>

/// <param name="h"></param>

/// <param name="w"></param>

/// <param name="lng"></param>

/// <param name="ipadress"></param>

public Server(uint h, uint w, Language lng, IPEndPoint ipadress)

{

InitializeComponent();

Title = lng.Server;

IP = ipadress;

H = (int)h;

W = (int)w;

language = lng;

send.Content = language.Send;

Mygrid.Children.Add(Chat);

Mygrid.Children.Add(send);

Mygrid.Children.Add(command);

Chat.Visibility = Visibility.Hidden;

send.Visibility = Visibility.Hidden;

command.Visibility = Visibility.Hidden;

send.Click += sendcommand;

command.KeyDown += KeyDown;

\_context = SynchronizationContext.Current;

}

private void KeyDown(object sender,KeyEventArgs e)

{

if (e.Key == Key.Enter)

sendcommand(sender, e);

}

private void Window\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

{

if (server!=null)

server.Disconnect();

if (language.Rus == true)

{

Process.Start("Kardano Grill.exe", "2");

}

else

{

Process.Start("Kardano Grill.exe", "3");

}

Thread.Sleep(1000);

Environment.Exit(0);

}

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

{

try

{

server = new ServerObject();

listenThread = new Thread(new ThreadStart(server.Listen));

listenThread.Start(); //старт потока

Start.Visibility = Visibility.Hidden;

Chat.Visibility = Visibility.Visible;

send.Visibility = Visibility.Visible;

command.Visibility = Visibility.Visible;

command.MaxLength = H \* W;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, language.Error, MessageBoxButton.OK, MessageBoxImage.Error);

server.Disconnect();

}

}

/// <summary>

/// отправить сообщение от сервера

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void sendcommand(object sender, RoutedEventArgs e)

{

string com = command.Text;

Chat.AppendText("[SERVER]: \""+com + "\"\n");

command.Text = "";

server.BroadcastMessage("[SERVER]: \"" + com + "\"",null);

}

}

}

### **Код окна ServerAndClientSettings**

using System;

using System.Windows;

using System.Net;

using System.Windows.Input;

namespace Kardano\_Grill

{

/// <summary>

/// Логика взаимодействия для ServerAndClientSetttings.xaml

/// </summary>

public partial class ServerAndClientSetttings : Window

{

char T;

uint H, W;

Language\_Library.Language Lang;

/// <summary>

/// выбор ip и порта

/// </summary>

/// <param name="h"></param>

/// <param name="w"></param>

/// <param name="l"></param>

/// <param name="Type\_of\_window"></param>

public ServerAndClientSetttings(uint h, uint w, Language\_Library.Language l, char Type\_of\_window)

{

InitializeComponent();

if (Type\_of\_window == 'C') Title = l.Client; else Title = l.Server;

Continue.Content = l.Continue;

H = h;

W = w;

Lang = l;

T = Type\_of\_window;

}

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

{

if (e.Key == Key.Enter)

Continue\_Click(sender, e);

}

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

{

if (e.Key == Key.Enter)

Continue\_Click(sender, e);

}

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

{

try

{

IPEndPoint iPEndPoint = new IPEndPoint(IPAddress.Parse(IP.Text), int.Parse(Port.Text));

if (T == 'C')

{

new Client(H, W, Lang, iPEndPoint).Show();

Close();

}

else

{

new Server(H, W, Lang, iPEndPoint).Show();

Close();

}

}

catch(Exception ex)

{

MessageBox.Show(Lang.ErrorMessage, Lang.Error, MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

}

### **Код окна ServerOrClient**

using System;

using System.Collections.Generic;

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.Shapes;

using Language\_Library;

namespace Kardano\_Grill

{

/// <summary>

/// Логика взаимодействия для ServerOrClient.xaml

/// </summary>

public partial class ServerOrClient : Window

{

Language lng;

uint h, w;

// для нажатия сервер / клиент (начало)

private void Server\_MouseDown(object sender, MouseButtonEventArgs e)

{

new ServerAndClientSetttings(h, w, lng, 'S').Show();

Close();

}

private void Server\_MouseEnter(object sender, MouseEventArgs e) => Server.Opacity = 0.5;

private void Server\_MouseLeave(object sender, MouseEventArgs e) => Server.Opacity = 1;

private void Client\_MouseDown(object sender, MouseButtonEventArgs e)

{

new ServerAndClientSetttings(h,w,lng,'C').Show();

Close();

}

private void Client\_MouseEnter(object sender, MouseEventArgs e) => Client.Opacity = 0.5;

private void Client\_MouseLeave(object sender, MouseEventArgs e) => Client.Opacity = 1;

// конец

/// <summary>

/// метод для запуска выбора сервер/клиент

/// </summary>

/// <param name="H">ширана решетки</param>

/// <param name="W">длина решетки</param>

/// <param name="L">выбранный язык</param>

public ServerOrClient(uint H, uint W, Language L)

{

InitializeComponent();

h = H;

w = W;

lng = L;

Chose.Content = lng.Chose;

serverclient.Content = lng.serverclient;

Title = lng.Chose;

}

}

}

# **ПРИЛОЖЕНИЕ 1**

**ТЕРМИНОЛОГИЯ**

|  |
| --- |
|  |

Таблица

|  |  |
| --- | --- |
| **Термин** | **Определение** |
| **Решетка Кардано** | Инструмент шифрования и дешифрирования, представляющий собой специальную прямоугольную (в частном случае — квадратную) таблицу-карточку, часть ячеек которой вырезана. |
| **Шифрование** | Обратимое преобразование информации в целях скрытия от неавторизованных лиц, с предоставлением, в это же время, авторизованным пользователям доступа к ней. Шифрование служит для обеспечения конфиденциальности передаваемой информации. Важной особенностью любого алгоритма шифрования является использование ключа, который утверждает выбор конкретного преобразования из совокупности возможных для данного алгоритма. |

**ПРИЛОЖЕНИЕ 2**

**ОПИСАНИЕ ФОРМАТА .krd**

Файл формата .krd содержит информацию о решетки, созданной в программе Cardano Grill, в текстовом виде.

Открыть файл формата .krd можно программой Cardano Grill.

Программа Cardano Grill при этом осуществляет корректное открытие только файлов формата .krd, созданных в этой программе.

# **ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц в докум.) | № документа | Входящий № сопроводительного докум. и дата | Подп. | Дата |
| Изм. | Измененных | Замененных | Новых | Аннулированх |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |