МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

Высшего профессионального образования

«ИЖЕВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ИМЕНИ М. Т. КАЛАШНИКОВА»

Кафедра «Программное обеспечение»

Отчет по Лабораторной работе №1

По дисциплине: «Защита информации»

На тему: «Решётка Кардано»

Студент гр. Б08-191-2 Р.О. Огорельцев

Принял: А.В. Старыгин

Ижевск 2017

Задание

Разработать программу осуществляющую шифрование и расшифрование Решёткой Кардано.

Требования:

1. Наглядность процесса шифрования\расшифрования
2. Возможность зашифровать данные с одним ключом, а расшифровать с другим
3. Возможность изменить шифрованные данные перед расшифрованием

Листинг

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.Input;

using System.Windows.Media;

namespace CardanoCode

{

public partial class MainWindow

{

private Grid \_grid;

private Grid \_markupGrid;

private Button \_btnDecrypt;

private TextBox \_tbDecrypt;

private char[,] \_cardano;

public MainWindow()

{

InitializeComponent();

}

private async void ButtonEncrypt(object sender, RoutedEventArgs e)

{

var dimension = GetDimensionForGrid(Text.Text.Length);

if (\_grid != null)

Panel.Children.Remove(\_grid);

\_grid = CreateGrid(dimension);

Panel.Children.Add(\_grid);

if (\_markupGrid != null)

Panel.Children.Remove(\_markupGrid);

\_markupGrid = CreateGrid(dimension, true);

Panel.Children.Add(\_markupGrid);

if (\_btnDecrypt != null)

Panel.Children.Remove(\_btnDecrypt);

if (\_tbDecrypt != null)

Panel.Children.Remove(\_tbDecrypt);

var markup = CreateMarkup(dimension);

var info = GetInfoFromMarkup(markup, dimension);

var code = CreateCode(info);

WriteCode(code, Code);

FillGridMarkup(\_markupGrid, markup, dimension, code);

\_cardano = await FillGridCode(Text.Text, \_grid, markup, dimension, code, 100);

CreateDecryptSection(markup, \_cardano);

}

private void CreateDecryptSection(int [,] markup, char[,] cardano)

{

var button = new Button { Content = "decrypt", DataContext = markup, CommandParameter = cardano };

button.Click += ButtonDecryptOnClick;

\_btnDecrypt = button;

Panel.Children.Add(\_btnDecrypt);

\_tbDecrypt = new TextBox();

Panel.Children.Add(\_tbDecrypt);

}

private void ButtonDecryptOnClick(object sender, RoutedEventArgs routedEventArgs)

{

var numbers = \_tbDecrypt.Text.Split(new[] {' '}, StringSplitOptions.RemoveEmptyEntries).Select(int.Parse).ToList();

var btn = sender as Button;

if (btn != null)

{

var markup = btn.DataContext as int[,];

var cardano = btn.CommandParameter as char[,];

var result = new StringBuilder();

for (var i = 0; i < Math.Sqrt(markup.Length); i++)

{

for (var j = 0; j < Math.Sqrt(markup.Length); j++)

{

if (numbers.Contains(markup[i, j]))

{

result.Append(cardano[i, j]);

}

}

}

MessageBox.Show(result.ToString());

}

}

private void WriteCode(List<int> code, TextBlock textBlock)

{

var strCode = code.Aggregate("", (current, i) => current + (i + " "));

textBlock.Text = strCode;

}

private int GetDimensionForGrid(int lenght)

{

for (int i = 4, j = 4; ; i += 2, j += 8)

{

if (j-4 > lenght)

{

return i;

}

}

}

private Grid CreateGrid(int dimension, bool colapsed = false)

{

var grid = new Grid { ShowGridLines = true, Visibility = colapsed ? Visibility.Collapsed : Visibility.Visible};

grid.MouseEnter += GridOnMouseEnter;

grid.MouseLeave += GridOnMouseLeave;

for (var i = 0; i < dimension; i++)

{

var gridCol = new ColumnDefinition();

grid.ColumnDefinitions.Add(gridCol);

for (var j = 0; j < dimension; j++)

{

var gridRow = new RowDefinition();

grid.RowDefinitions.Add(gridRow);

}

}

return grid;

}

private void GridOnMouseLeave(object sender, MouseEventArgs mouseEventArgs)

{

\_grid.Visibility = Visibility.Visible;

\_markupGrid.Visibility = Visibility.Collapsed;

}

private void GridOnMouseEnter(object sender, MouseEventArgs mouseEventArgs)

{

\_grid.Visibility = Visibility.Collapsed;

\_markupGrid.Visibility = Visibility.Visible;

}

private int[,] CreateMarkup(int dimension)

{

var markup = new int[dimension, dimension];

FillPartOfMarkup(markup, dimension, (i, dimansion) => i < dimension / 2, true,

(j, dimansion) => j < dimension / 2, true);

FillPartOfMarkup(markup, dimension, (i, dimansion) => i < dimension / 2, true,

(j, dimansion) => j >= dimension / 2, false, dimension - 1, invert: true);

FillPartOfMarkup(markup, dimension, (i, dimansion) => i >= dimension / 2, false,

(j, dimansion) => j >= dimension / 2, false, dimension - 1, dimension - 1);

FillPartOfMarkup(markup, dimension, (i, dimansion) => i >= dimension / 2, false,

(j, dimansion) => j < dimension / 2, true, firstValueI: dimension - 1, invert: true);

return markup;

}

private void FillPartOfMarkup(int[,] markup, int dimension, Func<int, int, bool> conditionI, bool incOrDecI, Func<int, int, bool> conditionJ, bool incOrDecJ, int firstValueJ = 0, int firstValueI = 0, bool invert = false)

{

var number = 1;

for (var i = firstValueI; conditionI(i, dimension); i = incOrDecI ? ++i : --i)

{

for (var j = firstValueJ; conditionJ(j, dimension); j = incOrDecJ ? ++j : --j)

{

if (invert)

markup[j, i] = number;

else

markup[i, j] = number;

number++;

}

}

}

private List<List<int>> GetInfoFromMarkup(int[,] array, int dimension)

{

var arraysInt = new List<List<int>>();

for (var i = 0; i < dimension/2; i++)

{

for (var j = 0; j < dimension/2; j++)

{

var numbers = new List<int>();

if (IsContainsInArrayOfArray(arraysInt, array[i, j]))

continue;

numbers.Add(array[i, j]);

numbers.Add(array[i + dimension/2, j]);

numbers.Add(array[i + dimension/2, j + dimension/2]);

numbers.Add(array[i, j + dimension/2]);

arraysInt.Add(numbers);

}

}

return arraysInt;

}

private async void FillGridMarkup(Grid grid, int[,] array, int dimension, List<int> code = null, int delay = 0)

{

for (var i = 0; i < dimension; i++)

{

for (var j = 0; j < dimension; j++)

{

var txtBlock = new TextBlock

{

Text = array[i, j].ToString(),

FontSize = 25,

VerticalAlignment = VerticalAlignment.Center,

HorizontalAlignment = HorizontalAlignment.Center,

Background = code != null && code.Contains(array[i, j]) ? new SolidColorBrush(Colors.DarkSeaGreen) : new SolidColorBrush(Colors.White)

};

Grid.SetRow(txtBlock, i);

Grid.SetColumn(txtBlock, j);

await Task.Delay(delay);

grid.Children.Add(txtBlock);

}

}

}

private async Task<char[,]> FillGridCode(string text, Grid grid, int[,] array, int dimension, List<int> code, int delay = 0)

{

var result = new char[dimension, dimension];

for (var i = 0; i < dimension; i++)

{

for (var j = 0; j < dimension; j++)

{

string letter;

if (code != null && !string.IsNullOrEmpty(text) && code.Contains(array[i, j]))

{

letter = text.Substring(0, 1);

text = text.Substring(1);

}

else

{

letter = string.Empty;

}

var txtBlock = new TextBox

{

Text = !string.IsNullOrEmpty(letter) ? letter : GetRandomChar().ToString(),

FontSize = 25,

VerticalAlignment = VerticalAlignment.Center,

HorizontalAlignment = HorizontalAlignment.Center,

DataContext = new {X = i, Y = j},

};

txtBlock.SelectionChanged += TxtBlockOnSelectionChanged;

result[i, j] = !string.IsNullOrEmpty(letter) ? letter[0] : GetRandomChar();

Grid.SetRow(txtBlock, i);

Grid.SetColumn(txtBlock, j);

await Task.Delay(delay);

grid.Children.Add(txtBlock);

}

}

return result;

}

private void TxtBlockOnSelectionChanged(object sender, RoutedEventArgs routedEventArgs)

{

var tn = sender as TextBox;

var point = tn.DataContext as dynamic;

\_cardano[point.X, point.Y] = tn.Text[0];

}

private List<int> CreateCode(List<List<int>> Info)

{

return (from elem in Info

let random = new Random()

let index = random.Next(0, elem.Count)

select elem[index]).ToList();

}

private char GetRandomChar()

{

const string alphabet = "qwertyuiopasdfghjklzxcvbnmйцукенгшщзхъфывапролджэячсмитьбю ЙЦУКЕНГШЩЗХЪЭЖДЛОРПАВЫФЯЧСМИТЬБЮQWERTYUIOPLKJHGFDSAZXCVBNM";

var random = new Random();

var index = random.Next(0, alphabet.Length);

return alphabet[index];

}

private static bool IsContainsInArrayOfArray(IEnumerable<List<int>> array, int value)

{

return array.Any(elem => elem.Contains(value));

}

}

}

Контрольный пример



