Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования (ОАиП)

ОТЧЕТ

по разминке №1

Тема работы: Кодирование строки

Выполнил

студент: гр. 251003 Панкратьев Е.С.

Проверил: Фадеева Е.П.

Минск 2023

Содержание

[1 Постановка задачи 3](#_Toc134463641)

[2 Дополненная постановка задачи 4](#_Toc134463642)

[3 Метод решения 5](#_Toc134463643)

[3.1 Проверка введенных данных 5](#_Toc134463644)

[3.2 Описание способа работы с программой 5](#_Toc134463645)

[3.3 Описание алгоритма кодирования 5](#_Toc134463646)

[3.4 Описание алгоритма декодирования 6](#_Toc134463647)

[4 Описание алгоритмов решения задачи 7](#_Toc134463648)

[5 Структура данных 11](#_Toc134463649)

[6 Результаты расчетов 16](#_Toc134463650)

[Приложение А 17](#_Toc134463651)

[Приложение Б 29](#_Toc134463652)

# Постановка задачи

Дана строка некоторой длины. Необходимо написать программу кодирования и декодирования (предоставляется также ключ) строки по алгоритму кругового обхода матрицы из центра.

# Дополненная постановка задачи

Дана строка некоторой длины. Необходимо написать программу кодирования и декодирования (предоставляется также ключ) строки по алгоритму кругового обхода матрицы из центра. Количество повторений для кодирования (количество изменений строки по данному алгоритму) задавать случайным образом. Также случайным образом определять направление первого шага (вверх, вниз, вправо, влево) и направление движения (по или против часовой стрелки) для каждого отдельного кодирования.

# Метод решения

## Проверка введенных данных

Проверка введенных данных происходит с помощью цикла с пocтусловием repeat..until, чтобы при вводе некорректных данных пользователь вводил их заново. Также для проверки существования файла используется функция FileExists. Она позволяет проверить, существует ли файл с указанным именем в указанном месте. Функция возвращает значение типа Boolean: True, если файл существует, и False, если файл не существует.

## Описание способа работы с программой

Программа представляет собой меню с двумя опциями: кодирование и декодирование текста.

Если пользователь выбирает опцию кодирования, то программа запрашивает имя файла для исходного текста, который нужно закодировать. Затем программа кодирует текст, сохраняет его в новый файл в папке "Files\ Encoded" и создает ключ шифрования для этого текста, который также сохраняется в папке "Files\Encoded".

Если пользователь выбирает опцию декодирования, то программа запрашивает имя файла закодированного текста и имя файла ключа для декодирования. Если оба файла существуют в папке "Files\Encoded", то программа использует ключ для декодирования текста и сохраняет результат в новый файл в папке "Files\Decoded".

Если пользователь вводит неправильный выбор, программа сообщает о недоступности данной опции и запрашивает повторный выбор. При успешном завершении операции, программа выводит сообщение об успешном выполнении.

## Описание алгоритма кодирования

Алгоритм кодирования можно описать следующим образом:

1. Вычисляем порядок матрицы, необходимой для кодирования исходной строки. Для этого находим первое нечетное число, квадрат которого больше или равен длине исходной строки. Этот порядок используется для создания матрицы, в которой будут храниться закодированные символы.
2. Создаем матрицу заданного размера и заполняем ее буферными символами, которые будут заменены на закодированные символы в процессе кодирования.
3. Инициализируем структуру Result, которая будет содержать закодированную строку и ключ для раскодирования.
4. Случайным образом выбираем количество проходов по матрице, которые будут выполнены для кодирования строки. Для этого используется функция Random, которая генерирует случайное целое число в заданном диапазоне.
5. Далее случайным образом выбираем одну из 8 возможных комбинаций характеристик для генерации: 4 по направлению первого шага и 2 по направлению движения. Эти характеристики определяют направление движения по матрице и действие, которое необходимо выполнить в каждой клетке матрицы. Затем добавляем текущее направление движения в ключ. Следующим действием проходим по матрице, начиная с центра, и в каждой клетке выполняем действие, соответствующее текущей комбинации характеристик: либо записываем в клетку следующий символ из исходной строки и двигаемся дальше по заданному направлению, либо записываем текущий символ в клетку и меняем направление движения. И в конце считываем закодированные символы из матрицы, пропуская буферными символы, и записываем их в итоговую закодированную строку.
6. Возвращаем структуру Result, содержащую закодированную строку и ключ для раскодирования.

## Описание алгоритма декодирования

Алгоритм декодирования строки можно описать следующим образом:

1. Вычисляем порядок матрицы, необходимой для декодирования закодированной строки. Для этого находим первое нечетное число, квадрат которого больше или равен длине закодированной строки. Этот порядок используется для создания матрицы, в которой будут раскодированные символы.

2. Создаем матрицу заданного размера и заполняем ее буферными символами, которые будут заменены на раскодированные символы в процессе декодирования.

3. Инициализируем структуру Result, которая будет содержать раскоди-

рованную строку.

4. Итерируем по каждому символу ключа в обратном порядке.

5. Далее получаем характеристики направления движения по матрице и действий, необходимых для декодирования символов, используя текущий символ ключа. Заем проходим по матрице, начиная с центра, и в каждой клетке выполняем действие, соответствующее текущей комбинации характеристик: либо записываем в клетку следующий символ из закодированной строки и двигаемся дальше по заданному направлению, либо записываем текущий символ в клетку и меняем направление движения. И в конце считываем раскодированные символы из матрицы, пропуская буферными символы, и записываем их в итоговую раскодированную строку.

6. Возвращаем содержащую раскодированную строку.

# Описание алгоритмов решения задачи

Таблица – Описание алгоритмов решения задачи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  п.п. | Наименование алгоритма | Название алгоритма | Формальные  параметры | Предпола-гаемый тип реализации |
| 1 | Основной алгоритм | Запрашивает ввод. Вызов следующих подпрограмм:  Encoding и Decoding |  |  |
| 2 | WriteToMatrix  (  AMatrix,  AStr,  I, J, L  ) | Записывает символ с индексом L из строки AStr в элемент массива AMatrix с координатами (I, J) | AMatrix – получает от фактического параметра адрес. Возвращаемый параметр.  AStr – получает от фактического параметра адрес.  Возвращаемый параметр.  I, J, L – получает от фактического параметра адрес с защитой | Процедура |
| 3 | WriteToStr  (  AMatrix,  AStr,  I, J, L  ) | Записывает символ из массива AMatrix с координатами (I, J) в строку AStr с индексом L | AMatrix – получает от фактического параметра адрес.  Возвращаемый параметр.  AStr – получает от фактического параметра адрес.  Возвращаемый параметр.  I, J, L – получает от фактического параметра адрес с защитой | Процедура |

Продолжение Таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | GetDirections  (  ANum,  ADirection,  ACurrDir  ) | В зависимости от входного числа ANum происходит присвоение направления по часовой или против часовой стрелки в переменную ADirection и обновление индекса текущего направления ACurrDir | ANum – получает от фактического параметра адрес с защитой.  ADirection – получает от фактического параметра адрес.  Возвращаемый параметр.  ACurrDir  – получает от фактического параметра адрес.  Возвращаемый параметр | Процедура |
| 5 | MatrixPass  ( AMatrix,  AStr,  ADirections,  ACurrDir,  AAction ) | Выполняет обход матрицы AMatrix в соответствии направлением ADirections и текущим направлением ACurrDir. Вызывает подпрограмму AAction для каждого элемента матрицы, что позволяет записывать либо в AStr, либо в AMatrix. Матрица обходится по спирали, начиная с центра, при этом направление и частота сброса меняются в заранее определенном порядке. | AMatrix – получает от фактического параметра адрес.  Возвращаемый параметр.  AStr – получает от фактического параметра адрес.  Возвращаемый параметр.  ADirections – получает от фактического параметра адрес с защитой.  ACurrDir –  получает от фактического параметра значение.  AAction – получает от фактического параметра адрес с защитой | Процедура |

Продолжение Таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | GenerateDecodedText  (  AText,  AKey,  Result  ) | Расшифровывает строку AText, используя ключ AKey путем прохождения по матрице символов, изменяя строку на каждом шаге в соответствии с текущим направлением и действием. Количество проходов и направлений определены в AKey. Результат записывается в Result. Вызов следующих подпрограмм:  GetDirections  и MatrixPass | AText – получает от фактического параметра адрес.  Возвращаемый параметр.  AKey – получает от фактического параметра адрес.  Возвращаемый параметр.  Result – получает от фактического параметра адрес.  Возвращаемый параметр | Функция. Вовзвращаемый параметр Reult |
| 6 | Decoding  ( ATextFileName,  AKeyFileName ) | Cчитывает зашифрованный текст из ATextFileName и ключ шифрования из AKeyFileName, вызывает подпрограмму GenerateDecodedText для расшифровки текста и записывает расшифрованный текст в новый файл | ATextFileName – получает от фактического параметра адрес с защитой. AKeyFileName – получает от фактического параметра адрес с защитой | Процедура |
| 7 | GenerateEncodedText  (  AText  Result  ) | Генерирует зашифрованную версию входной строки AText путем прохождения по матрице символов, изменяя строку на | AText – получает от фактического параметра адрес.  Возвращаемый параметр. | Функция. Возвращаемый параметр Result |

Продолжение Таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | каждом шаге в соответствии с текущим направлением и действием. Количество проходов случайно выбирается. Ключ и зашифрованная строка записываются в Result.  Вызов следующих подпрограмм:  GetDirections  и MatrixPass | Result – получает от фактического параметра адрес. Возвращаемый параметр |  |
| 8 | Encoding  (  ATextFileName  ) | Кодирует текстовый файл, генерируя зашифрованный текст и ключ шифрования. Она читает входной текстовый файл ATextFileName  , подпрограмма GenerateEncodedText генерирует зашифрованный текст и ключ. Далее записывает ся зашифрованный текст и ключ в отдельные файлы | ATextFileName – получает от фактического параметра адрес с защитой. | Процедура |

# Структура данных

Таблица 2 – Структура типов программы

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| TEncodedWithKey | Record  Key: string;  EncodedStr: string;  End; | Представляет зашифрованную строку и ключ, которые могут быть получены в результате процесса шифрования, описанного в коде. |
| TDirection | array [0..3] of char; | Представляет набор возможных направлений движения в матрице |
| TMatrix | array of array of Char; | Представляет матрицу символов, которая используется для кодирования сообщения. |
| TAction | procedure (var AMatrix: TMatrix; var AStr: string; const I, J, L: Integer); | Представляет собой процедурный тип данных, который может использоваться в качестве типа параметра функций или процедур. |

Таблица 3 – Структура данных основной программы

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| Choice | Char | Выбор пользователя |
| isCorrect | Bolean | Индикатор правильного ввода |
| TextFileName | String | Имя файла для кодирование или декодирования |
| MinMaxPoint | String | Имя файла ключа |

Таблица 4 – Структура данных алгоритма WriteToMatrix(AMatrix, AStr, I, J, L)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| AMatrix | TMatrix | Двумерный массив, хранящий зашифрованную матрицу | Формальный |

Продолжение Таблицы 4

|  |  |  |  |
| --- | --- | --- | --- |
| AStr | String | Исходный текст | Формальный |
| I, J, K | Integer | Счетчики циклов | Формальные |

Таблица 5 – Структура данных алгоритма WriteToStr(AMatrix, AStr, I, J, L)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | | Тип параметра |
| AMatrix | TMatrix | Двумерный массив, хранящий зашифрованную матрицу | | Формальный |
| AStr | String | Исходный текст | Формальный | |
| I, J, K | Integer | Счетчики циклов | Формальные | |

Таблица 6 – Структура данных алгоритма GetDirections (ANum, ADirection, ACurrDir)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| ANum | Integer | Номер функции | Формальный |
| ADirection | TDirection | Направление движения | Формальный |
| ACurrDir | Integer | Номер текущего направления | Формальный |

Таблица 7 – Структура данных алгоритма MatrixPass(AMatrix, AStr, ADirections, ACurrDir, AAction)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| AMatrix | TMatrix | Двумерный массив, хранящий зашифрованную матрицу | Формальный |
| AStr | String | Исходный текст | Формальный |
| CurrDist | Double | Лучшая точка | Формальный |
| ADirections | TDirections | Направление движения | Формальный |
| AAction | TAction | Действие, которое нужно выполнить при переборе элементов матрицы | Формальный |

Продолжение Таблицы 7

|  |  |  |  |
| --- | --- | --- | --- |
| ConsecutiveQuantity | Integer | Количество элементов в текущей последовательности | Локальный |
| ResetCounter | Integer | Счетчик последовательности до сброса | Локальный |
| Counter | Integer | Счетчик количества в последовательности | Локальный |
| I, J, L | Integer | Счетчики цикла | Локальный |

Таблица 8 – Структура данных алгоритма GenerateDecodedText(AText, AKey, Result)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| AText | String | Зашифрованный текст | Формальный |
| AKey | String | Ключ | Формальный |
| Result | String | Расшифрованная строка | Формальный |
| MatrixOrder | Integer | Параметр матрицы | Локальный |
| I, J, K, L | Integer | Счетчики цикла | Локальный |
| Matrix | Integer | Двумерный массив, хранящий зашифрованную матрицу | Локальный |
| CurrDir | Integer | Текущее направление движения | Локальный |
| Directions | TDirections | Направление обхода | Локальный |

Таблица 9 – Структура данных алгоритма Decoding(ATextFileName, AKeyFileName)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| ATextFileName | String | Имя файла, в котором находится зашифрованная строка | Формальный |

Продолжении Таблицы 10

|  |  |  |  |
| --- | --- | --- | --- |
| AKeyFileName | String | Имя файла, в котором находится ключ | Формальный |
| flOutput | TextFile | Файл, в которой необходимо положить данные | Локальный |
| flInput | TextFile | Файл, в которой необходимо взять данные | Локальный |
| flKey | TextFile | Файл, в котором хранится ключ | Локальный |
| InputString | String | Входная строка | Локальный |
| InputKey | String | Строка ключа | Локальный |

Таблица 11 – Структура данных алгоритма GenerateEncodedText(AText, Result)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| AText | String | Исходный текст | Формальный |
| Result | TEncodedWithKey | Зашифрованная строка с ключом | Формальный |
| MatrixOrder | Integer | Параметр матрицы | Локальный |
| I, J, K, L | Integer | Счетчики цикла | Локальный |
| Matrix | Integer | Двумерный массив, хранящий зашифрованную матрицу | Локальный |
| CurrDir | Integer | Текущее направление движения | Локальный |
| Directions | TDirections | Направление обхода | Локальный |

Таблица 12 – Структура данных алгоритма Encoding (ATextFileName)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| ATextFileName | String | Имя файла, в котором находится зашифрованная строка | Формальный |

Продолжении Таблицы 12

|  |  |  |  |
| --- | --- | --- | --- |
| flOutput | TextFile | Файл, в которой необходимо положить данные | Локальный |
| flInput | TextFile | Файл, в которой необходимо взять данные | Локальный |
| InputString | String | Входная строка | Локальный |
| EncodedWithKey | TEncodedWithKey | Сформированный ключ с зашифрованной строкой | Локальный |

# Результаты расчетов

В ходе работы были получены следующие результаты:

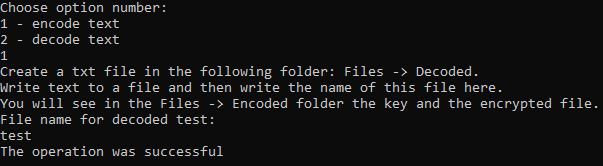


Рисунок 1 – Результаты расчетов

Приложение А

(обязательное)

Исходный код программы

program WarmUp2;

{

This program provides the user with the option to encode or

decode text using a

encryption algorithm. The user is asked to choose between

two options:

to encode text or to decode it

}

uses

System.SysUtils,

En\_Decoding in 'Unit\En\_Decoding.pas';

var

Choice: Char;

isCorrect: Boolean;

TextFileName, KeyFileName: string;

// Choice - stores the user's choice

// isCorrect - indicates whether the user's input was

// correct or not

// TextFileName, KeyFileName - store the file names entered

// by the

begin

Writeln('Choose option number:');

Writeln('1 - encode text');

Writeln('2 - decode text');

isCorrect:= True;

repeat

Readln(Choice);

case Choice of

'1':

begin

// This option is used to encode a text file

Writeln('Create a txt file in the following folder:

Files -> Decoded.');

Writeln('Write text to a file and then write the name

of this file here.');

Writeln('You will see in the Files -> Encoded folder

the key and the encrypted file.');

// This loop continues until the user enters a valid

// file name for the decoded text

repeat

Writeln('File name for decoded test:');

Readln(TextFileName);

until FileExists('..\..\..\Files\Decoded\' +

TextFileName + '.txt');

Encoding(TextFileName);

Writeln('The operation was successful');

end;

'2':

begin

// This option is used to decode a text file

Writeln('Install two text files in the following

folder: Files -> Encoded.');

Writeln('One file should be encoded text and the

second should be key');

Writeln('Enter the file names in sequence. First the

encoded text and then the key');

Writeln('You will see the decoded text in the Files

-> Decoded folder');

// This loop continues until the user enters valid

// file names

repeat

Writeln('File name for encoded text:');

Readln(TextFileName);

Writeln('File name for key:');

Readln(KeyFileName);

until FileExists('..\..\..\Files\Encoded\' +

TextFileName + '.txt') and

FileExists('..\..\..\Files\Encoded\' +

KeyFileName + '.txt');

Decoding(TextFileName, KeyFileName);

Writeln('The operation was successful');

end;

else

begin

// If the user enters an invalid choice, this message

// will be displayed

Writeln('Unavailable option! Try again');

isCorrect := False;

end;

end;

until isCorrect;

Readln;

end.

unit En\_Decoding;

{

This unit contains procedures and functions for encoding

and decoding messages

using a matrix transposition cipher algorithm. The matrix

transposition cipher algorithm

involves arranging characters of the message in a matrix in

a certain order, and then

reading off the characters in a different order to obtain

the encoded message.

The decoding procedure reverses the encoding process to

obtain the original message

}

interface

type

// The TEncodedWithKey record is used to store both the key

// and the encoded text

// It is used in the Encoding procedure to return both

// values to the main program

TEncodedWithKey = Record

Key: string;

EncodedStr: string;

End;

// The Encoding procedure takes a file name as input, reads

// the text from the file,

// encodes it using a specific algorithm, and writes the

// encoded text and key to output files

procedure Encoding(const ATextFileName: string);

// The Decoding procedure takes two file names as input - one

// for the encoded text and

// one for the key, reads the text from both files, decodes // the text using the same

// algorithm that was used to encode it, and writes the decoded text to an output file.

procedure Decoding(const ATextFileName, AKeyFileName: string);

implementation

uses

System.SysUtils;

Type

TDirection = array [0..3] of char;

TMatrix = array of array of Char;

TAction = procedure (var AMatrix: TMatrix; var AStr:

string; const I, J, L: Integer);

// TDirection - the TDirection type is an array of

// characters that represent

// the four directions used in the algorithm: Up, Right,

// Down, and Left.

// TMatrix - type is a dynamic array that is used to store

// the matrix of

// characters used in the algorithm.

// TAction - type is a procedure type that takes a matrix

// of characters, a string,

// and three integer arguments. It is used to perform the

// main action of the

// algorithm - either encoding or decoding.

const

Сlockwise: TDirection = ('U','R','D','L');

AntiСlockwise: TDirection = ('U','L','D','R');

TrashChar = #152;

AmountToReset = 2;

DirectionAmount = 4;

// Сlockwise is a TDirection array that represents the

// clockwise

// directions used in the algorithm.

// AntiСlockwise is a TDirection array that represents the

// counterclockwise

// directions used in the algorithm.

// TrashChar is a character that is used as a placeholder

// when a

// character is deleted from the matrix.

// AmountToReset is an integer that represents the number

// of times a character

// must pass through the entire matrix in order to be

// considered

// "fully encoded" by the algorithm

// DirectionAmount is an integer that represents the number

// of

// directions used in the algorithm.

// These procedures are used to write characters to and from

// a matrix.

// WriteToMatrix writes the character from AStr to AMatrix at

// coordinates I, J.

// WriteToStr writes the character from AMatrix at

// coordinates I, J to AStr at index L.

procedure WriteToMatrix(var AMatrix: TMatrix; var AStr: string; const I, J, L: Integer);

begin

AMatrix[I, J] := AStr[L];

end;

procedure WriteToStr(var AMatrix: TMatrix; var AStr: string; const I, J, L: Integer);

begin

AStr[L]:= AMatrix[I, J];

end;

// Set the directions and current direction based on input

// number

// Assign clockwise or anticlockwise directions based on

// input number

// Update the current direction index based on input number

procedure GetDirections(const ANum: Integer; var ADirections: TDirection;

var ACurrDir: Integer);

begin

case ANum of

0:

begin

ADirections := Сlockwise;

ACurrDir := 0;

end;

1:

begin

ADirections := Сlockwise;

ACurrDir := 1;

end;

2:

begin

ADirections := Сlockwise;

ACurrDir := 2;

end;

3:

begin

ADirections := Сlockwise;

ACurrDir := 3;

end;

4:

begin

ADirections := AntiСlockwise;

ACurrDir := 0;

end;

5:

begin

ADirections := AntiСlockwise;

ACurrDir := 1;

end;

6:

begin

ADirections := AntiСlockwise;

ACurrDir := 2;

end;

7:

begin

ADirections := AntiСlockwise;

ACurrDir := 3;

end;

end;

end;

// This procedure performs matrix traversal, based on the

// specified direction

// and current direction. It calls the specified action for

// each matrix element,

// allowing you to encode or decode the message. The matrix

// is traversed in a

// spiral pattern starting from the center, with the direction

// and reset frequency

// alternating in a predetermined manner.

procedure MatrixPass(var AMatrix: TMatrix; var AStr: string;

const ADirections: TDirection; ACurrDir: Integer;

const AAction :TAction);

var

ConsecutiveQuantity, ResetCounter, Counter: Integer;

I, J, L: Integer;

begin

ConsecutiveQuantity := 1;

Counter := 0;

ResetCounter := 0;

I := Length(AMatrix) shr 1;

J := I;

for L := 1 to Length(AStr) do

begin

AAction(AMatrix, AStr, I, J, L);

Case ADirections[ACurrDir] of

'U': Dec(I);

'D': Inc(I);

'L': Dec(J);

'R': Inc(J);

End;

Inc(Counter);

if Counter = ConsecutiveQuantity then

begin

Inc(ACurrDir);

ACurrDir := ACurrDir mod DirectionAmount;

Inc(ResetCounter);

Counter := 0;

if ResetCounter = AmountToReset then

begin

ResetCounter := 0;

Inc(ConsecutiveQuantity);

end;

end;

end;

end;

// Rounds up a single-precision floating-point number to the

// nearest integer

// greater than or equal to the number. If the fractional

// part of the number is

// greater than 0, then the result is rounded up to the next

// integer.

function Ceil(const X: Single): Integer;

begin

Result := Trunc(X);

if Frac(X) > 0 then

Inc(Result);

end;

// This procedure performs a traversal of the given matrix,

// modifying the

// characters in the string based on the current position and // action performed

// at each step. The traversal direction is determined by the // current direction

// and the direction change sequence. The consecutive

// quantity of steps in each

// direction before changing direction is determined by the

// amount to reset value.

function GenerateDecodedText(var AText, AKey: string): string;

var

MatrixOrder, I, J, K, L: Integer;

Matrix: TMatrix;

CurrDir: Integer;

Directions : TDirection;

begin

// Calculate the size of the matrix to be used to encode the text

MatrixOrder:= Ceil(sqrt(Length(AText)));

Inc(MatrixOrder, Ord(not Odd(MatrixOrder)));

// Create a matrix to store the encoded characters

SetLength(Matrix, MatrixOrder, MatrixOrder);

// Set up the result string to have the same length as the input string

SetLength(Result, Length(AText));

Result:= AText;

AText:= '';

// Loop over each character in the key

for K := Length(AKey) downto 1 do

begin

for I := 0 to MatrixOrder - 1 do

for J := 0 to MatrixOrder - 1 do

Matrix[I, J] := TrashChar;

// Get the direction and starting position for encoding

// the characters

GetDirections(StrToInt(AKey[K]), Directions, CurrDir);

// Pass over the matrix, encoding the characters

MatrixPass(Matrix, Result, Directions, CurrDir,

WriteToMatrix);

// Put the encoded characters back into the result string

L := 1;

for I := 0 to High(Matrix) do

for J := 0 to High(Matrix) do

if (Matrix[I, J] <> TrashChar) then

begin

Matrix[I, J] := Result[L];

Inc(L);

end;

// Get the direction and starting position for decoding

// the characters

GetDirections(StrToInt(AKey[K]), Directions, CurrDir);

// Pass over the matrix again, decoding the characters

// and putting them

// into the output string

MatrixPass(Matrix, Result, Directions, CurrDir,

WriteToStr);

end;

end;

// This procedure reads the encoded text and the encryption

// key from text files,

// generates the decoded text using the GenerateDecodedText

// function, and writes

// the decoded text to a new file.

procedure Decoding(const ATextFileName, AKeyFileName: string);

var

flOutput, flInput, flKey: TextFile;

InputString, InputKey: String;

begin

Assignfile(flOutput, '..\..\..\Files\Decoded\Decoded\_' + ATextFileName + '.txt');

Rewrite(flOutput);

Assignfile(flInput, '..\..\..\Files\Encoded\' + ATextFileName + '.txt');

Reset(flInput);

Assignfile(flKey, '..\..\..\Files\Encoded\' + AKeyFileName + '.txt');

Reset(flKey);

Read(flInput, InputString);

Read(flKey, InputKey);

Write(flOutput, GenerateDecodedText(InputString,

InputKey));

Close(flInput);

Close(flOutput);

Close(flKey);

end;

// This function generates an encoded version of the input

// string by repeatedly passing

// over a matrix of characters, modifying the string in each

// pass according to the

// current traversal direction and action. The number of

// passes is randomized.

// The generated key is the sequence of traversal directions // in clockwise order,

// with an additional offset added for counterclockwise traversal.

function GenerateEncodedText(var AText: String): TEncodedWithKey;

const

MinPasses = 7;

MaxRandom = 420;

var

MatrixOrder, I, J, K, L: Integer;

Matrix: TMatrix;

CurrDir: Integer;

Directions : TDirection;

begin

// Calculate the size of the matrix to be used to encode

// the text

MatrixOrder:= Ceil(sqrt(Length(AText)));

Inc(MatrixOrder, Ord(not Odd(MatrixOrder)));

// Create a matrix to store the encoded characters

SetLength(Matrix, MatrixOrder, MatrixOrder);

// Initialize the result struct with the encoded string and

// an empty key

Result.Key := '';

Result.EncodedStr:= AText;

AText:= '';

// Randomize the number of passes to make over the matrix

Randomize;

for K := 1 to Random(MaxRandom) + MinPasses do

begin

// Initialize the matrix with trash characters

for I := 0 to MatrixOrder - 1 do

for J := 0 to MatrixOrder - 1 do

Matrix[I, J] := TrashChar;

// Get the traversal direction and sequence for the pass

GetDirections(Random(8), Directions, CurrDir);

// Add the current traversal direction to the key

if Directions = Сlockwise then

Result.Key := Result.Key + IntToStr(CurrDir)

else

Result.Key := Result.Key + IntToStr(CurrDir +

DirectionAmount);

// Pass over the matrix, modifying the encoded string

MatrixPass(Matrix, Result.EncodedStr, Directions,

CurrDir, WriteToMatrix);

// Put the modified characters back into the encoded

// string

L:= 1;

for I := 0 to High(Matrix) do

for J := 0 to High(Matrix) do

if Matrix[I, J] <> TrashChar then

begin

Result.EncodedStr[L] := Matrix[I, J];

Inc(L);

end;

end;

end;

// This procedure encodes the input text file by generating

// an encoded text and

// an encryption key. It reads the input text file, generates

// the encoded text

// and key using the GenerateEncodedText function, and writes

// the encoded text

// and key to separate files.

procedure Encoding(const ATextFileName: string);

var

flOutput, flInput, flKey: TextFile;

InputString: String;

EncodedWithKey: TEncodedWithKey;

begin

Assignfile(flInput, '..\..\..\Files\Decoded\' + ATextFileName + '.txt');

Reset(flInput);

Assignfile(flOutput, '..\..\..\Files\Encoded\Encoded\_' + ATextFileName + '.txt');

Rewrite(flOutput);

Assignfile(flKey, '..\..\..\Files\Encoded\Key\_' + ATextFileName + '.txt');

Rewrite(flKey);

Read(flInput, InputString);

EncodedWithKey:= GenerateEncodedText(InputString);

Write(flOutput, EncodedWithKey.EncodedStr);

Write(flKey, EncodedWithKey.Key);

Close(flInput);

Close(flOutput);

Close(flKey);

end;

end.

Приложение Б

(обязательное)

Тестовые наборы

**Тестовая ситуация: некорректный ввод данных**

**Тест 1**

Исходные данные: Некорректный ввод дат

Ожидаемый результат: Повторная попытка

Полученный результат:

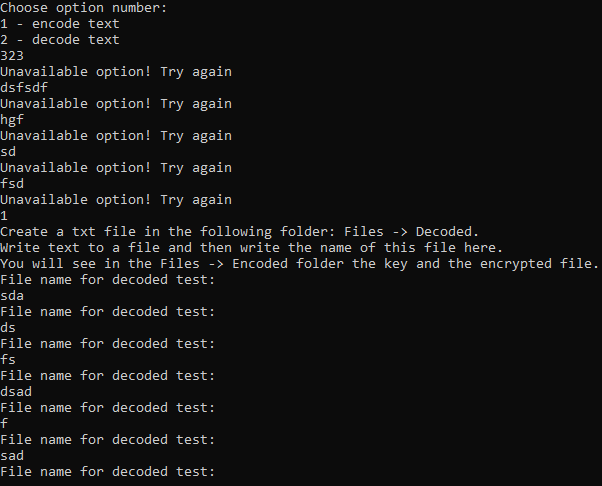


Рисунок 2 – Результаты расчетов

**Тестовая ситуация: корректный ввод данных**

Таблица 13 – Тестовые ситуации

|  |  |
| --- | --- |
| Тест | Кодирование |
| 1(a) |  |
| 2(a) |  |
| 2(a) |  |
| 2(б) |  |

Продолжение таблицы 13

|  |  |
| --- | --- |
| 3(а) |  |
| 3(б) |  |