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 разминочной работе №2

Тема работы: Шифрование

Выполнил

студент: гр.251003 Жук Я. С.

Проверил: Фадеева Е.П.

Минск 2023

Содержание

[1 Постановка задачи 3](#_Toc133570898)

[1.1 Начальная постановка 3](#_Toc133570899)

[1.2 Конечная постановка задачи 3](#_Toc133570900)

[2 Метод решения 4](#_Toc133570901)

[Описание алгоритмов решения задачи 6](#_Toc133570902)

[3 Структура данных 8](#_Toc133570903)

[4 Результаты расчетов 11](#_Toc133570904)

[Приложение А 12](#_Toc133570905)

[Приложение Б 22](#_Toc133570906)

[Б.1 Проверка расчётов 22](#_Toc133570907)

# Постановка задачи

## Начальная постановка

Чтобы зашифровать текст из 121 буквы, его можно записать в матрицу [11, 11], затем прочитать по спирали из центра (6, 6). Исходная информация находится в виде файла. Реализовать алгоритмы зашифровки и расшифровки.

## Конечная постановка задачи

Зашифровать текст, полученный из файла, путём представления этой строки в виде матрицы [n, n] и прохождению по спирали из центра. Количество таких операций, начальное направления прохождения (вверх, вниз, влево или вправо) по спирали и направление (по часовой или против часовой стрелки) определяется посредством РРЧ-генератора. При зашифровке текста создаётся файл с данными, перечисленными раннее, и результат шифрования. При расшифровке создаётся файл с расшифрованным текстом. Каждый файл при создании имеет имя по умолчанию в зависимости от вида операции. Если пользователь после её проведения выберет опцию сохранения файла, то, при необходимости, он может изменить имя конечного файла. В случае, если пользователь решит не сохранять результат, созданный файл будет удалён. Пользователь может вводить имя файла, расположенного в определенной директории, либо ввести абсолютный путь.

# Метод решения

Представим, что введенная строка размером 25 символов имеет вид матрицы 5х5:

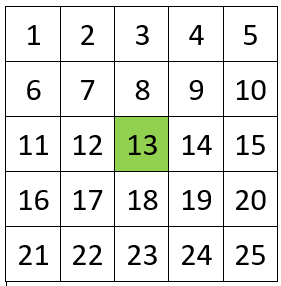


Рисунок 1 – Матрица

Прочтение по спирали означает то, что элементы строки с определенным индексом будут записываться в новую строку до тех пор, пока новая строка не будет заполнена полностью.

Центральным элементом является элемент с номером 13. При прохождении по спирали по часовой стрелке и с начальным направлением влево нужно менять направление в следующих случаях (выделены жёлтым цветом):

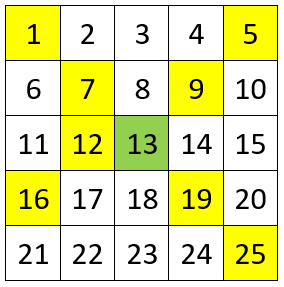


Рисунок 2 – Матрица

Проанализировав номера индексов, можно прийти к выводу, что чтобы найти места, в которых необходимо изменить направление движения, необходимо знать номер центрального элемента, количество рядов матрицы и то, какой сейчас оборот спирали происходит. Таким образом, формула для нахождения каждого места, в котором необходимо изменить направление имеет данный вид:

где *centre* – номер центрального элемента, *raws* – размерность квадратной матрицы, *b* – номер оборота спирали в данный момент. Знак операции зависит от того, какой направление движения в данный момент. Так как всего направлений 4, то и случаев есть такое же количество:

* , если вверх с левой стороны:

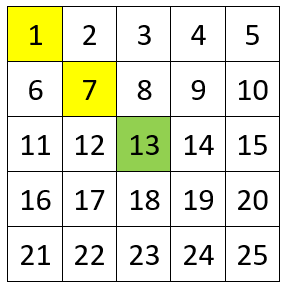


Рисунок 3 – Движение происходит вверх с левой стороны

* , если вверх с правой стороны:

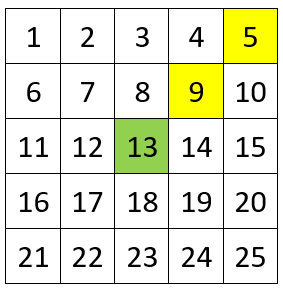


Рисунок 4 – Движение происходит вверх с правой стороны

* , если вниз с левой стороны;
* , если вниз с правой стороны.

Данный оборот по спирали будет завершен в том случае, когда текущее направление движения будет равно начальному. В таком случае будет увеличено значение переменной *b*.

# Описание алгоритмов решения задачи

Таблица 3 – Описание алгоритмов решения задачи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  п.п. | Наименование алгоритма | Название алгоритма | Формальные  параметры | Предпола-гаемый тип реализации |
| 1 | Основной алгоритм | Вызов следующих подпрограмм:  LoadTextFromFile, SetMatrixLength, Coding,  Decoding. |  |  |
| 2 | LoadTextFromFile  (  FName,  Res  ) | Прочитывает текст из файла с именем FName в переменную Res. В случае неверного пути/имени файла возвращает пустую строку | FName - получает от фактического параметра значение;  Res - получает от фактического параметра адрес, возвращаемый параметр. | Функция.  Res – возвращаемый функцией параметр |
| 3 | SetMatrixLength  (  S  ) | Устанавливает такую длину строки S, чтобы её можно было представить в виде нечетной квадратной матрицы. Заполняет новые элементы спецсимволом | S – получает от фактического параметра адрес. | Процедура |
| 4 | Coding  (  TextFUser,  BName  ) | Кодирует текст TextFUser и сохраняет его в файл с именем BName | TextFUser - получает от фактического параметра адрес;  BName - получает от фактического параметра адрес с защитой. | Процедура |

Продолжение Таблицы3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | Decoding  (  FName,  BName,  Res  ) | Расшифровывает текст из файла специального формата с именем FName, сохраняет результат в файл с именем BName и передает результат в переменную Res. | FName, BName - получает от фактического параметра его значение;  Res  - получает от фактического параметра адрес с защитой, возвращаемый параметр; | Функция.  Res – возвращаемый функцией параметр |

# Структура данных

Таблица 4 – Структура данных основной программы

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| Answer, TypeC | String | Ответы пользователя |
| FName, Text | String | Название файла и текст, который будет в нём сохранён соответственно |
| WrongAnswer | String | Отвечает за правильный формат ответа пользователя |

Таблица 5 – Структура данных алгоритма SetMatrixLength(S)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| I | Integer | Счётчик цикла | Локальный |
| N | Integer | Размерность необходимой нечетной квадратной матрицы | Локальный |
| Len | Integer | Начальная длина строки S | Локальный |
| S | String | Строка, считанная из файла | Формальный |

Таблица 6 – Структура данных алгоритма LoadTextFromFile(FName, Res)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| FFromUser | TextFile | Файл, из которого необходимо прочитать текст | Локальный |
| FName | String | Имя файла, из которого необходимо прочитать текст | Формальный |
| Res | String | Прочитанный текст | Формальный |

Таблица 7 – Структура данных алгоритма Coding(TextFUser, BName)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| I | Integer | Индекс элемента новой строки | Локальный |
| J | Integer | Счетчик цикла | Локальный |
| K | Integer | Индекс элемента матрицы | Локальный |
| N | Integer | Длина строки | Локальный |
| B | Integer | Номер оборота | Локальный |
| Centre | Integer | Индекс центра | Локальный |
| Cycles | Integer | Количество обходов по спирали | Локальный |
| Raws | Integer | Размерность матрицы | Локальный |
| DClockwise | Boolean | Направление по часовой стрелке | Локальный |
| Change | Boolean | Необходимость изменения направления | Локальный |
| Direction, Start | Пользовательский | Текущее и начальное направление | Локальный |
| ResOfCycle | String | Текст, полученный после данного обхода | Локальный |
| ResF | TextFile | Файл для сохранения результата | Локальный |
| TextFUser | String | Текст для кодирования | Формальный |
| BName | String | Название файла для сохранения по умолчанию | Формальный |

Таблица 7 – Структура данных алгоритма Decoding(FName, BName, Res)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| I | Integer | Индекс элемента новой строки | Локальный |
| J | Integer | Счетчик цикла | Локальный |
| K | Integer | Индекс элемента матрицы | Локальный |
| N | Integer | Длина строки | Локальный |

Продолжение Таблицы 7

|  |  |  |  |
| --- | --- | --- | --- |
| B | Integer | Номер оборота | Локальный |
| Centre | Integer | Индекс центра | Локальный |
| Cycles | Integer | Количество обходов по спирали | Локальный |
| Raws | Integer | Размерность матрицы | Локальный |
| DClockwise | Boolean | Направление по часовой стрелке | Локальный |
| Change | Boolean | Необходимость изменения направления | Локальный |
| Direction, Start | Пользовательский | Текущее и начальное направление | Локальный |
| ResOfCycle | String | Текст, полученный после данного обхода | Локальный |
| GavenF, ResF | TextFile | Файл для расшифровки и для сохранения результата соответственно | Локальный |
| ArrOfDir | Array of byte | Массив направлений, используемых для зашифровки | Локальный |
| TextFUser | String | Текст для кодирования | Локальный |
| FName | String | Имя файла для расшифровки | Формальный |
| Res | String | Текст, полученный при расшифровке | Формальный |
| BName | String | Название файла для сохранения по умолчанию | Формальный |

# Результаты расчетов

Результаты расчетов при зашифровки:

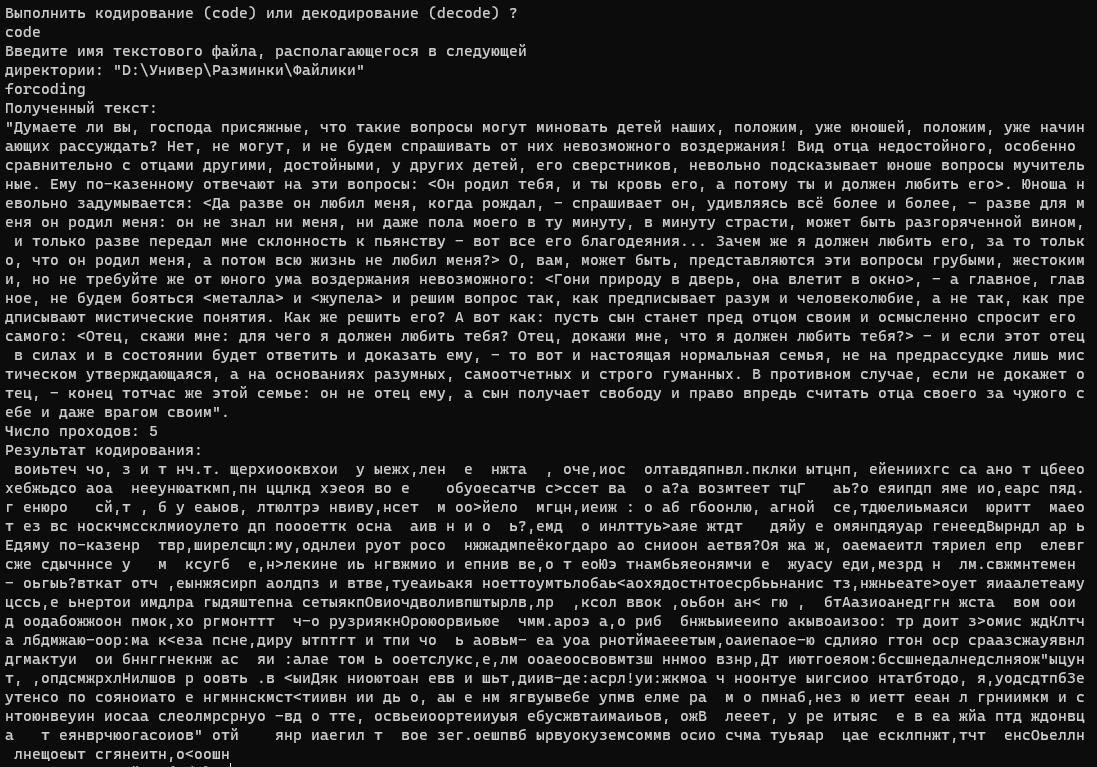


Рисунок 5 – Результаты расчетов зашифровки

Результаты расчетов расшифровки:

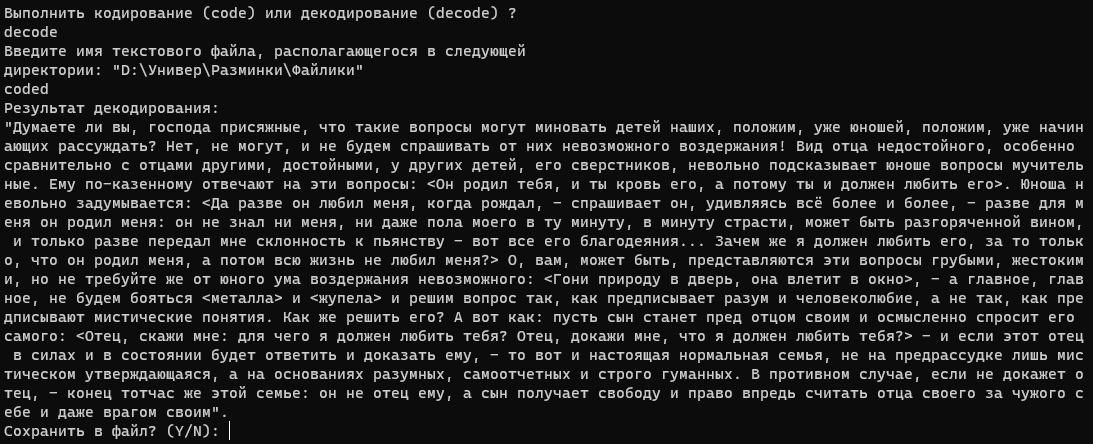


Рисунок 6 – Результаты расчетов расшифровки

Приложение А

(обязательное)

Исходный код программы

program CodingAndDecoding;

{

Program for coding and decoding the text

from files from the directory

}

uses

SysUtils;

// Subprogram for setting neccessary length

// for the text to code S

procedure SetMatrixLength(var s: string);

var

i, n, len: integer;

// n – length of raws in matrix

// len – start length

Begin

len := Length(s);

n := round(sqrt(len));

// The sqrt of a length must be odd as the centre

// of 'matrix' has to be clear

While not odd(n) or (n \* n < len) do

inc(n);

setlength(s, n \* n);

// Fill the added length with a special symbol, which

// will not be displayed to the user when the output

for i := len + 1 to n \* n do

s[i] := '#';

End;

// Subprogram for loading a text Fname

// from a file and checking it for right format

function LoadTextFromFile(FName: String): String;

var

FFromUser: TextFile;

// FFromUser – text file from user with text

Begin

{$I-}

AssignFile(FFromUser, FName);

Reset(FFromUser);

{$I+}

if IOResult <> 0 then

Writeln('Неверный путь/название файла!')

else

begin

Readln(FFromUser, Result);

CloseFile(FFromUser);

Writeln('Полученный текст:');

Writeln(Result);

end;

End;

// Subprogram for coding a text TextFUser

// from given file and saving it to the file

// with name BName

procedure Coding(var TextFUser: String; Bname: String);

type

TDirection = (Up = 0, Right, Down, Left);

var

i, j, n, k, b, centre, cycles, raws: integer;

DClockwise, change: boolean;

Direction, Start: TDirection;

ResOfCycle: String;

ResF: TextFile;

// I, j, k – for cycles

// n – the length of raws

// b – the number of cycle

// cycles – amount of cycles

// Dclockwise, start – direction

begin

Randomize;

// Preparation of required parameters (length of matrix,

// its series, center; number of times to encode text)

n := Length(TextFUser);

raws := round(sqrt(n));

centre := round((n + 1) / 2);

cycles := Random(19) + 1;

setlength(ResOfCycle, n);

// Creating the result file

AssignFile(ResF, Bname);

Rewrite(ResF);

// Choosing by random the direction (clockwise)

// Writting value of cycles and direction to the file

DClockwise := boolean(Random(2));

Writeln(ResF, 'Clockwise: ', DClockwise);

Writeln(ResF, 'Number of cycles: ', cycles);

Writeln('Число проходов: ', cycles);

for j := 1 to cycles do

Begin

// Choosing by random a start direction (TDirection)

Direction := TDirection(Random(4));

Start := Direction;

ResOfCycle[1] := TextFUser[centre];

case Direction of

Up:

begin

k := centre - raws;

Writeln(ResF, 'U');

end;

Right:

begin

k := centre + 1;

Writeln(ResF, 'R');

end;

Down:

begin

k := centre + raws;

Writeln(ResF, 'D');

end;

Left:

begin

k := centre - 1;

Writeln(ResF, 'L');

end;

end;

ResOfCycle[2] := TextFUser[k];

if DClockwise then

Begin

If Direction <> high(TDirection) then

inc(Direction)

Else

Direction := Low(TDirection)

End

Else

Begin

if Direction <> Low(TDirection) then

dec(Direction)

Else

Direction := High(TDirection);

End;

i := 3;

b := 1;

// Unwind the old text 'matrix' and assign each element

// of this'matrix' to a new element of the new text

While i <= n do

begin

// The decision of changing the direction to the next

change := False;

// Calculating an index of a next element of the

// unwinding 'matrix' and checking for the change and

// 'borders' of 'matrix'

case Direction of

Up:

if (k - raws) < (centre - raws \* b - b) then

change := True

else

k := k - raws;

Right:

if (k = (centre - raws \* b + b)) or (k = (centre

+ raws \* b + b)) or (k = (centre + b)) then

Begin

change := True;

If (Direction=Start) and (k mod raws <> 0) then

Begin

k := k + 1;

ResOfCycle[i] := TextFUser[k];

inc(i);

inc(b);

End;

End

else

k := k + 1;

Down:

if ((k + raws) > (centre + raws \* b + b)) then

change := True

else

k := k + raws;

Left:

if (k = (centre - raws \* b - b)) or (k = (centre

+ raws \* b - b)) or (k = (centre - b)) then

Begin

change := True;

If (Direction=Start)and((k-1) mod raws<>0) then

Begin

k := k - 1;

ResOfCycle[i] := TextFUser[k];

inc(i);

inc(b);

End;

End

else

k := k - 1;

end;

// Changing the direction to the next

if change then

Begin

if DClockwise then

Begin

If Direction <> high(TDirection) then

inc(Direction)

Else

Direction := Low(TDirection)

End

Else

Begin

if Direction <> Low(TDirection) then

dec(Direction)

Else

Direction := High(TDirection);

End;

// If new direction is the start one (Up/Down) then

// inc value of the turn

If (Direction = Start) and ((Start = Down) or

(Start = Up)) then

inc(b);

End

else

Begin

// Assign the next element of the new text the

// desired value of the old one

ResOfCycle[i] := TextFUser[k];

inc(i);

End;

end;

// Assign new text instead of old

TextFUser := ResOfCycle;

End;

// Closing the result file

Writeln(ResF, TextFUser);

CloseFile(ResF);

end;

// Subprorgam for decoding a text from given file

// with special format

function Decoding(FName, Bname: String): String;

type

TDirection = (Up = 1, Right, Down, Left);

var

i, j, n, b, k, centre, cycles, raws: integer;

DClockwise, change: boolean;

Direction, Start: TDirection;

ResOfCycle, TextFUser: String;

GavenF, ResF: TextFile;

ArrOfDir: array of byte;

// I, j, k – for cycles

// n – the length of raws

// b – the number of cycle

// cycles – amount of cycles

// Dclockwise, start – direction

// ArrOfDir – array of directions

begin

{$I-}

AssignFile(GavenF, FName);

Reset(GavenF);

Readln(GavenF, TextFUser);

If Copy(TextFUser, 12, 1) = 'F' then

DClockwise := False

Else if Copy(TextFUser, 12, 1) = 'T' then

DClockwise := True

Else

change := False;

Readln(GavenF, TextFUser);

Val(Copy(TextFUser, 19), cycles, i);

If i <> 0 then

change := False;

{$I+}

if (IOResult <> 0) or not change then

Writeln('Некорректный файл для декодирования!')

Else

Begin

setlength(ArrOfDir, cycles);

for i := 0 to cycles - 1 do

Begin

Readln(GavenF, TextFUser);

if TextFUser = 'U' then

ArrOfDir[i] := 1

else if TextFUser = 'R' then

ArrOfDir[i] := 2

else if TextFUser = 'D' then

ArrOfDir[i] := 3

else if TextFUser = 'L' then

ArrOfDir[i] := 4;

End;

Readln(GavenF, TextFUser);

n := Length(TextFUser);

raws := round(sqrt(n));

centre := round((n + 1) / 2);

CloseFile(GavenF);

setlength(ResOfCycle, n);

for j := 1 to cycles do

Begin

ResOfCycle[centre] := TextFUser[1];

Direction := TDirection(ArrOfDir[cycles - j]);

Start := Direction;

case Direction of

Up:

k := centre - raws;

Right:

k := centre + 1;

Down:

k := centre + raws;

Left:

k := centre - 1;

end;

ResOfCycle[k] := TextFUser[2];

if DClockwise then

Begin

If Direction <> high(TDirection) then

inc(Direction)

Else

Direction := Low(TDirection)

End

Else

Begin

if Direction <> Low(TDirection) then

dec(Direction)

Else

Direction := High(TDirection);

End;

i := 3;

b := 1;

// Unwind the old text 'matrix' and assign each

// element of this 'matrix' to a new element of the

// new text

While i <= n do

begin

// The decision of changing the direction to next

change := False;

// Calculating an index of a next element of the

// unwinding 'matrix' and checking for the change

// and 'borders' of 'matrix'

case Direction of

Up:

if (k - raws) < (centre - raws \* b - b) then

change := True

else

k := k - raws;

Right:

if (k=(centre- raws\*b+b)) or (k=(centre+raws\*b

+b)) or (k=(centre+b)) then

Begin

change := True;

If (Direction=Start) and (k mod raws<>0) then

Begin

k := k + 1;

ResOfCycle[k] := TextFUser[i];

inc(i);

inc(b);

End;

End

else

k := k + 1;

Down:

if ((k + raws) > (centre + raws \* b + b)) then

change := True

else

k := k + raws;

Left:

if (k=(centre-raws\*b-b)) or (k=(centre+raws\*b

-b)) or (k = (centre - b)) then

Begin

change := True;

If(Direction=Start)and((k-1)mod raws<>0) then

Begin

k := k - 1;

ResOfCycle[k] := TextFUser[i];

inc(i);

inc(b);

End;

End

else

k := k - 1;

end;

// Changing the direction to the next

if change then

Begin

if DClockwise then

Begin

If Direction <> high(TDirection) then

inc(Direction)

Else

Direction := Low(TDirection)

End

Else

Begin

if Direction <> Low(TDirection) then

dec(Direction)

Else

Direction := High(TDirection);

End;

// If new direction is the start one (Up/Down)

// then inc value of the turn

If (Direction = Start) and ((Start = Down) or

(Start = Up)) then

inc(b);

End

else

Begin

// Assign the next element of the new text the

// desired value of the old one

ResOfCycle[k] := TextFUser[i];

inc(i);

End;

end;

// Assign new text instead of old

TextFUser := ResOfCycle;

End;

// Opening, writting into and closing the result file

AssignFile(ResF, Bname);

Rewrite(ResF);

Result := TextFUser;

Writeln(ResF, TextFUser);

CloseFile(ResF);

End;

end;

var

Answer, Text, TypeC, FName: String;

WrongAnswer: boolean;

const

BNameCode = 'coded.txt';

BNameDecode = 'decoded.txt';

// Default file names to save

begin

SetCurrentDir('D:\Универ\Разминки\Файлики');

Repeat

Writeln('Выполнить кодирование (code) или декодирование

(decode) ?');

Readln(TypeC);

WrongAnswer := True;

if (TypeC = 'code') or (TypeC = 'decode') then

begin

Writeln('Введите имя текстового файла,

располагающегося в следующей');

Writeln('директории: "D:\Универ\Разминки\Файлики"');

Readln(Text);

if TypeC = 'code' then

Begin

Text := LoadTextFromFile(Text + '.txt');

SetMatrixLength(Text);

Coding(Text, BNameCode);

Writeln('Результат кодирования:');

FName := BNameCode;

End

Else if TypeC = 'decode' then

Begin

Text := Decoding(Text + '.txt', BNameDecode);

Writeln('Результат декодирования:');

FName := BNameDecode;

End;

// Outputting the final text without the symbol

Text := StringReplace(Text, '#', '', [rfReplaceAll]);

Writeln(Text);

WrongAnswer := False;

end

Else

Writeln('Такой функции нет! Попробуйте еще раз!');

Until not WrongAnswer;

// If it's neccessary saving and renaming the final file

Write('Сохранить в файл? (Y/N): ');

Readln(Answer);

if Answer = 'Y' then

begin

Writeln('Введите имя файла, под которым хотите

сохранить');

Writeln(TypeC, 'd текст (enter, если ', FName, ')');

Readln(Answer);

If not(Answer = '') then

ReNameFile(FName, Answer + '.txt');

Writeln(TypeC, 'd запись сохранена');

End

Else

DeleteFile(FName);

Readln;

end.

Приложение Б

(обязательное)

Тестовые наборы

Тест 1

Исходный файл:



Рисунок 7 – Исходный файл

Зашифрованный файл:



Рисунок 8 – Зашифрованный файл

Расшифрованный файл:



Рисунок 9 – Расшифрованный файл

Исходный файл совпадает с расшифрованным.

Тест 2



Рисунок 10 – Проверка ввода команды

Тест 3

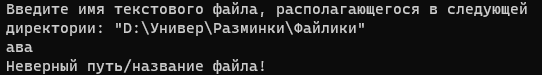


Рисунок 11 – Проверка наличия введенного файла по относительному пути

Тест 4

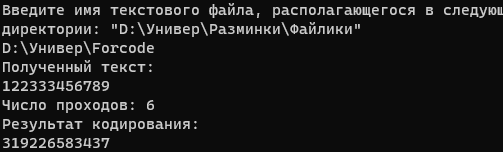


Рисунок 12 – Проверка наличия введенного файла по абсолютному пути