# Code, file Program.cs

using System.IO;

using System;

namespace FileEncryption

{

class Program

{

static void Main(string[] args)

{

//get whether they want to encrypt or decrypt

int EncrypOrDecryp = Other.EncDec();

if (EncrypOrDecryp == 1)//encrypt

{

Console.Write("Please enter the full path of the file --> ");//get the path of the plaintext

string filePath = Console.ReadLine();

int option = Other.Menu();//get the encryption method

if (option == 1)//ceaser cipher

{

if (File.Exists(@filePath))//check for file

{

string plaintext = File.ReadAllText(@filePath);

Console.WriteLine("Current content of the file {0}", plaintext);//get content of the file

Console.Write("Enter the shift value --> ");//get the shift wanted

int shift = int.Parse(Console.ReadLine());

string ciphertext = EcDc.CeaserCipher.EncryptCC(plaintext, shift);//shift it

Console.Write("Enter the full path of the new file you would like to create --> ");//the new file created

string exitFilePath = Console.ReadLine();

Console.WriteLine("New content: {0}", ciphertext);//display the new content to be saves

File.WriteAllText(@exitFilePath, ciphertext);//save the file

}

}

else if (option == 2)//vigeneere

{

if (File.Exists(@filePath))

{

string plaintext = File.ReadAllText(@filePath);//get the content of the file

Console.WriteLine("Current content of the file {0}", plaintext);

Console.Write("Enter the shift text --> ");//get the shift text

string shift = Console.ReadLine();

string ciphertext = EcDc.Vigeneere.EncryptVi(plaintext, shift);//shift all of the character

Console.Write("Enter the full path of the new file you would like to create --> ");//get the address of the new file

string exitFilePath = Console.ReadLine();

Console.WriteLine("New content: {0}", ciphertext);//display new content

File.WriteAllText(@exitFilePath, ciphertext);//write the new content

}

}

else if (option == 3)//verman OTP

{

if (File.Exists(@filePath))

{

string plaintext = File.ReadAllText(@filePath);//get content

Console.WriteLine("Current content of the file {0}", plaintext);

string key = "";

string ciphertext = EcDc.Vernam.EncryptVe(plaintext, ref key);//generate key and encrypt

Console.Write("Enter the full path of the new file you would like to create --> ");

string exitFilePath = Console.ReadLine();

Console.Write("Enter the full path of the new file you would like to create for the key --> ");//get the new key file

string exitKeyPath = Console.ReadLine();

Console.WriteLine("New content: {0}", ciphertext);//display the new content

File.WriteAllText(@exitFilePath, ciphertext);//write the new content

File.WriteAllText(exitKeyPath, key);//write the key to a file

}

}

}

else if (EncrypOrDecryp == 2)//decrypt

{

Console.Write("Please enter the full path of the file --> ");//get the ciphertext file

string filePath = Console.ReadLine();

int option = Other.Menu();//get method

if (option == 1)//ceaser

{

if (File.Exists(@filePath))

{

string ciphertext = File.ReadAllText(@filePath);//get content

Console.WriteLine("Current content of the file {0}", ciphertext);

Console.Write("Enter the shift value --> ");//get the shift value

int shift = int.Parse(Console.ReadLine());

string plaintext = EcDc.CeaserCipher.DecryptCC(ciphertext, shift);//decrypt the ciphertext

Console.Write("Enter the full path of the new file you would like to create --> ");//get the new file path

string exitFilePath = Console.ReadLine();

Console.WriteLine("New content: {0}", plaintext);//display new content

File.WriteAllText(@exitFilePath, plaintext);//write new content

}

}

else if (option == 2)//vigeneere

{

if (File.Exists(@filePath))

{

string ciphertext = File.ReadAllText(@filePath);//get content of the file

Console.WriteLine("Current content of the file {0}", ciphertext);

Console.Write("Enter the shift text --> ");//get the shift text

string shift = Console.ReadLine();

string plaintext = EcDc.Vigeneere.DecryptVi(ciphertext, shift);//decrypt with key

Console.Write("Enter the full path of the new file you would like to create --> ");//get new file path

string exitFilePath = Console.ReadLine();

Console.WriteLine("New content: {0}", plaintext);//display new content

File.WriteAllText(@exitFilePath, plaintext);//write new content

}

}

else if (option == 3)//vernam OTP

{

if (File.Exists(@filePath))

{

string ciphertext = File.ReadAllText(@filePath);//get content

Console.WriteLine("Current content of the file {0}", ciphertext);

Console.Write("Enter the full path of the key file --> ");//get the key from the file

string keyPath = Console.ReadLine();

string key = File.ReadAllText(@keyPath);//

Console.WriteLine("Current content of the file {0}", key);

string plaintext = EcDc.Vernam.DecryptVe(ciphertext, key);//decrypt the ciphertext

Console.Write("Enter the full path of the new file you would like to create --> ");//get new file location

string exitFilePath = Console.ReadLine();

Console.WriteLine("New content: {0}", plaintext);//display new content

File.WriteAllText(@exitFilePath, plaintext);//write new content

}

}

}

}

}

class Other

{

public static int Menu()//menu function

{

int option;

bool validUserInput;

do

{//display options

Console.WriteLine("0. Exit");

Console.WriteLine("1. Ceaser Cipher");

Console.WriteLine("2. Vigenere Cipher");

Console.WriteLine("3. Vernam Cipher");

validUserInput = int.TryParse(Console.ReadLine(), out option);//look for specific values

} while (!validUserInput || option < -1 || option > 4);

return option;

}

public static int EncDec()//the other menu, same as other one but different options

{

int option;

bool validUserInput;

do

{

Console.WriteLine("0. Exit");

Console.WriteLine("1. Encrypt");

Console.WriteLine("2. Decrypt");

validUserInput = int.TryParse(Console.ReadLine(), out option);

} while (!validUserInput || option < -1 || option > 3);

return option;

}

}

}

# Code, file EncryptDecrypt.cs

using System;

namespace EcDc

{

class CeaserCipher

{

public static string EncryptCC(string plaintext, int shift)//encryption for ceaser cipher

{

char letter;

string ciphertext = "";

//encrypt it

//outer loop: step through the plaintext, one character at a time

plaintext = plaintext.ToUpper(); //working in upper case

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

{

letter = plaintext[i];

for (int n = 0; n < shift; n++)

{

if (letter == 'Z')

{

letter = 'A'; //loop back to beginning of alphabet

}

else

{

letter++;

}

}

ciphertext += letter;

}

return ciphertext;

}

public static string DecryptCC(string ciphertext, int shift)

{

char letter;

string plaintext = "";

//encrypt it

//outer loop: step through the ciphertext, one character at a time

ciphertext = ciphertext.ToUpper(); //working in upper case

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

{

letter = ciphertext[i];

for (int n = 0; n < shift; n++)

{

if (letter == 'A')

{

letter = 'Z'; //loop back to beginning of alphabet

}

else

{

letter--;

}

}

plaintext += letter;

}

return plaintext;

}

}

class Vigeneere

{

public static string EncryptVi(string plaintext, string key)

{

char letter;//declare and intitialise variables

string ciphertext = "";

int j = 0;

key = key.ToLower();

letter = ' ';

for (int i = 0; i < plaintext.Length; i++, j++)//j is for the key, i is for the plaintext

{

if (j == key.Length)

{

j = 0;

}//loop for checking if the key needs resetting

char character = plaintext[i];

char keyLetter = key[j];

int shift = Convert.ToInt32(keyLetter) - 97;

letter = Convert.ToChar(CeaserCipher.EncryptCC(character.ToString(), shift));//ceaser cipher the induvidual characters by the shift of one of the key characters

ciphertext += letter;//add to the ciphertext

}

return ciphertext;

}

public static string DecryptVi(string ciphertext, string key)

{

char letter;

string plaintext = "";

int j = 0;

key = key.ToLower();

letter = ' ';

for (int i = 0; i < ciphertext.Length; i++, j++)//i for the ciphertext and j for the key

{

if (j == key.Length)

{

j = 0;

}//check the key back to the start if needed

char character = ciphertext[i];

char keyLetter = key[j];

int shift = Convert.ToInt32(keyLetter) - 97;

letter = Convert.ToChar(CeaserCipher.DecryptCC(character.ToString(), shift));//ceaser shift the characters back

plaintext += letter;

}

return plaintext;

}

}

class Vernam

{

static string GenerateRandom(int length)//generate key length of the text

{

Console.WriteLine(length);

Random rnd = new Random();

char[] randomstring = new char[length];//new char[] for key

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

{

randomstring[i] = Convert.ToChar(rnd.Next(97, 122));//character generation

}

string output = new string(randomstring);

return output;

}

public static string EncryptVe(string plaintext, ref string key)

{

string ciphertext;

key = GenerateRandom(plaintext.Length);//generate key

char[] cipherChar = new char[plaintext.Length];

Console.WriteLine("key: {0}", key.Length);//display the key

for (int i = 0; i < plaintext.Length; i++)//over each character

{

int letter = plaintext[i] ^ key[i];//xor the characters

cipherChar[i] = Convert.ToChar(letter + 97);

}

ciphertext = new string(cipherChar);//get it to the string

return ciphertext;

}

public static string DecryptVe(string ciphertext, string key)

{

string plaintext;

char[] plainChar = new char[ciphertext.Length];//get array for the new plaintext

for (int i = 0; i < ciphertext.Length; i++)//for every character

{

int letter = ciphertext[i] ^ key[i];//xor them

plainChar[i] = Convert.ToChar(letter);//add to char[]

}

plaintext = new string(plainChar);//to string

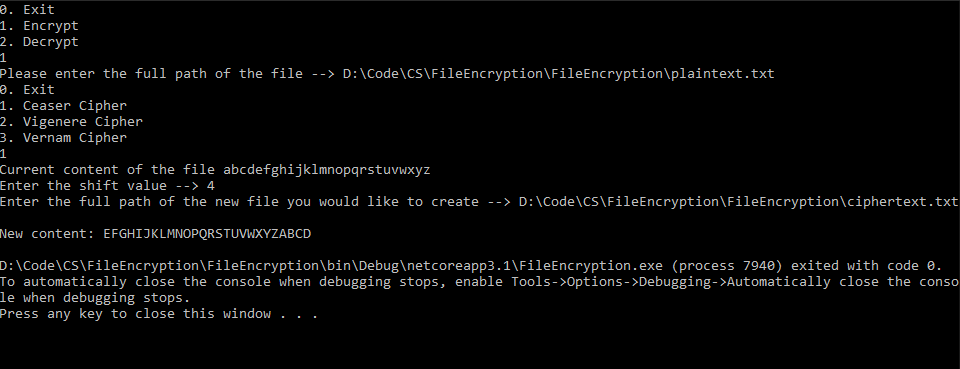
return plaintext;

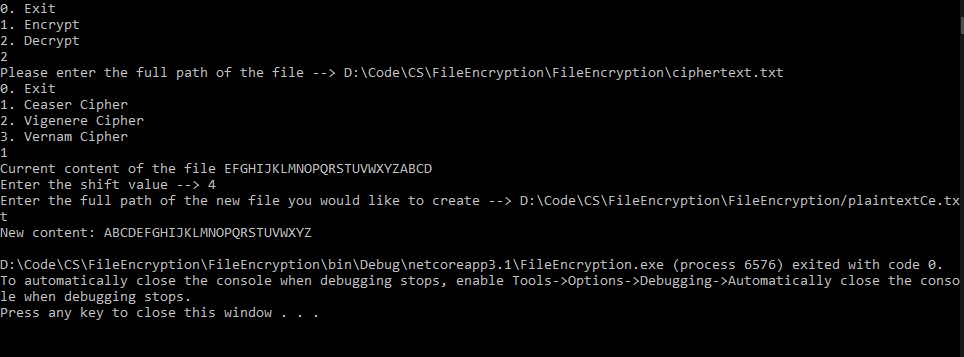
}

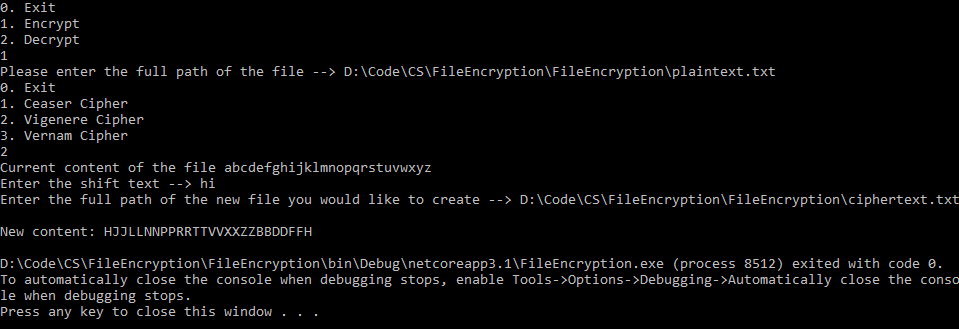
}

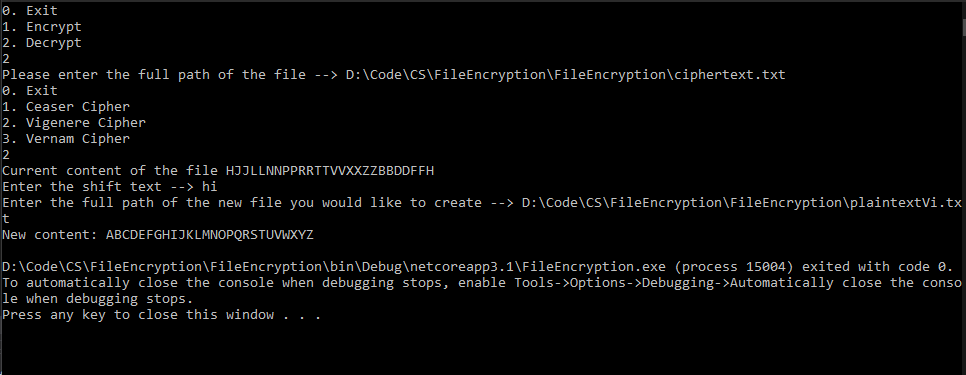
}

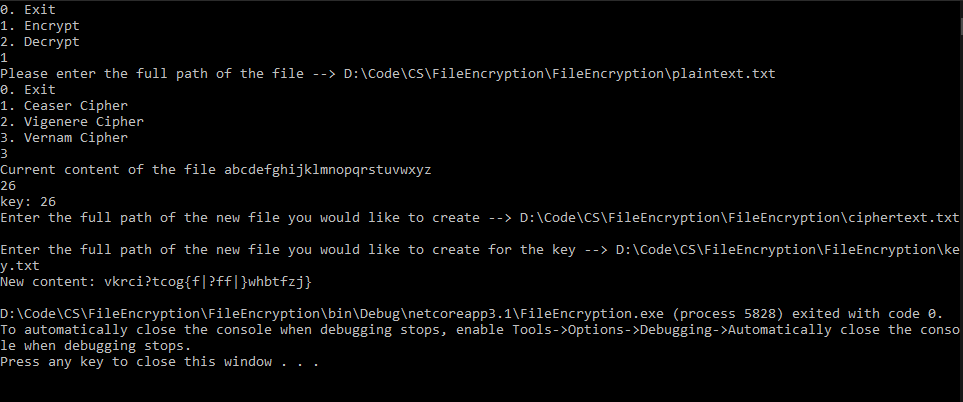
# Screenshots

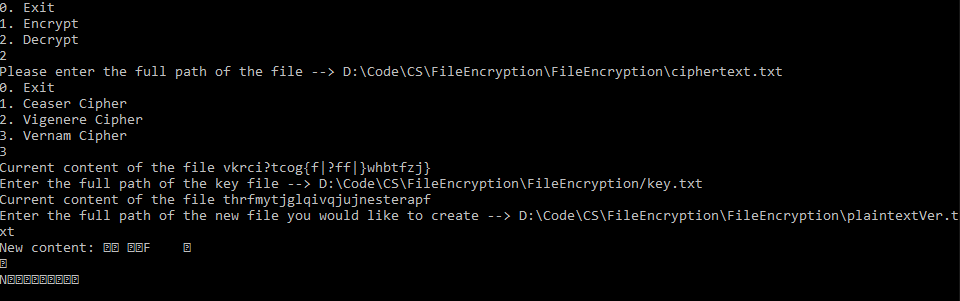




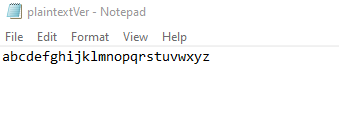








This one doesn’t appear to work but this is the file that it actually produces:

Which is then correct.

I think the problem is there is no runes for saying the data is text and it does not process it as that but control codes instead.

# Test table

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Outcome expected | Outcome actual | algorithm |
| abc…xyz; 4 | efg…xyzabcd | Same as expected | ceaser |
| abc…xyz; hi | HJJLLNNPPRRTTVVXXZZBBDDFFH | Same as expected | vigeneere |
| abc…xyz; random | vkrci?tcog{f|?ff|}whbtfzj} | Same as expected | vernam |

The reverse of this is also true (with a non-random key for vernam algorithm)