ОБЛАСТНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

«ТОМСКИЙ ТЕХНИКУМ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ»

Специальность 09.02.04 «Информационные системы (по отраслям)»

РАЗРАБОТКА ПРОГРАММЫ ШИФРОВКИ СООБЩЕНИЙ

Пояснительная записка

к учебной практике

УП.18.09.02.04.243.01.ПЗ

|  |  |  |
| --- | --- | --- |
| Студент  «\_\_»\_\_\_\_\_\_\_\_ 2022 г. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Левицкий П. Д. |
| Руководитель  «\_\_»\_\_\_\_\_\_\_\_ 2022 г. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Маюнова А. Ю. |
|  |  |  |
|  |  |  |
|  |  |  |

Томск 2022

Содержание

Введение 3

1 ОБЩАЯ ЧАСТЬ 4

1.1 Цель разработки (Описание предметной области) 4

1.2 Средства и среда разработки 5

1.3 Описание языка программирования 7

1.4 Теоретический раздел 8

2 СПЕЦИАЛЬНАЯ ЧАСТЬ 9

2.1 Постановка задачи 9

2.2 Эскизный проект 10

2.3 Технический проект 12

2.4 Инструкция пользователя 17

ЗАКЛЮЧЕНИЕ 18

Перечень используемых источников 18

Приложение A. Листинг программы 19

Приложение Б. Результат работы программы 32

*Утв.*

*Н. контр.*

*Пров.*

*Разраб.*

*Изм*

Лист

№ докум.

## Подпись

Дата

*Лит.*

## Лист

## Листов

## Т

ТТИТ 603 гр.

**ОПП.17.230401.232.01.ПЗ**

### Левицкий П. Д.

Пояснительная записка

36

2

**Введение**

В современном мире люди заботятся о своей информационной безопасности и конфиденциальности. С целью помочь этим людям издавна разрабатывались разные решения для обеспечения конфиденциальности переписки — от видоизменения древнеегипетских иероглифов до применения ключа шифрования AES.

Зачастую простым пользователям не нужны такие сложные решения.

Исходя из этого, есть вариант, который поможет пользователям с этой задачей. Этот вариант – создание компьютерной программы, которая позволила бы шифрацию и дешифрацию простых сообщений гораздо удобнее, быстрее и экономичнее чем ранее она велась, когда использовались другие решения.

Но программа должна решать основные задачи предметной области, и предоставлять максимальное удобство пользователю при работе. При этом начальная подготовка программы к работе должна быть сведена к минимуму, с целью того, чтобы не тратить время на установку и настройку программного продукта.

Безусловно, подобные программы уже существуют, но каждая из них имеет свои минусы и плюсы. Но создание новой программы, которая будет учитывать весь предыдущий опыт разработки программного обеспечения для данной предметной области, а также устранит ошибки, которые ранее были допущены в разработке программ.

Таким образом, главной целью является создание простой программы, которая позволила бы шифрацию\дешифрацию и упрощала работу пользователя.

# Лист

*№ докум.*

### Подпись

## Изм

### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

3

4

**Цель разработки**

Требуется разработать программу, которая позволяет зашифровать и расшифровать сообщение тремя различными способами.

Первый способ. Имеется текст – ключ шифра, содержащий достаточное количество слов. Определяется частота появления каждого символа в этом тексте и в зависимости от нее символу присваивается код – число или буква. При зашифровке сообщения каждый символ заменяется его кодом.

Второй способ. Каждому символу присваивается код в зависимости от очередности его появления в тексте – ключ.

Третий способ. Символы шифруемого текста заменяются на символы, полученные из исходного путем сдвига их кода ASCII на заданную величину. Кроме того, между символами сообщения помещаются дополнительные символы, выбранные случайным образом (их количество постоянно или подчиняется определенной закономерности). При расшифровке сообщения вывод на экран двух диаграмм, отражающих частоту появления символов в первичных и обработанных текстах.

При выборе пользователем опции «Шифровать» входные данные будут вводиться, выбираться способ шифрования, шифроваться и затем выводиться в шифрованном виде.

При выборе пользователем опции «Расшифровать» предварительно зашифрованные данные будут вводиться, выбираться способ, которым были предварительно зашифрованы входные данные, расшифровываться и выводиться в расшифрованном виде.

Приложение рассчитано на одного активного пользователя, в целях безопасности не хранит данные о своей работе после завершения процесса. Предполагается использование ПО на ПК, либо же на тонком клиенте. Интеграция со сторонним ПО отсутствует.

Анализ конкурентов.

У ПО много аналогов, основными характеристиками являются удобство для конечного пользователя, оптимизация и user-friendly интерфейс. Эти факторы необходимо учитывать при разработке ПО.

# Лист

*№ докум.*

### Подпись

## Изм

### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

**Средства и среда разработки**

draw.io обладает широким спектром функционала, предельно понятным и простым интерфейсом. Получить доступ к ресурсу можно получить практически с любого устройства и браузера.

ERWin обладает более широким функционалом — например, он имеет в своем наборе фигур те, которые отсутствуют в draw.io и способен проводить стоимостный анализ бизнес-процессов предприятий.

Так же в нем гораздо удобнее проводить декомпозицию бизнес-процесссов. Единственный недостаток — сложность получения лицензии для корпоративного использования.

На этапе реализации программного продукта должно быть уделено внимание таким деталям, как удобное обновление полезной нагрузки среды разработки и удобство процесса разработки. Для этого была выбрана MS Visual Studio 2022.

Среда позволяет написание, средства отладки и сборки кода, а также последующей публикации приложений. Помимо стандартного редактора и отладчика, которые есть в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для улучшения процесса разработки.

Среда разработки располагает редактируемым дизайном, огромным количеством расширений и приятным UI.

# Лист

*№ докум.*

### Подпись

## Изм

### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

5

Важно отметить, что в качестве вспомогательных средств на всех этапах проектирования были использованы такие вспомогательные средства, как Github и LibreOffice Writer.

Github используется по прямому назначению — удобная система контроля версий, хранение документации и скомпилированного кода.

LibreOffice Writer был выбран по причине наличия того же функционала и поддержки расширений .doc и .docx что и в MS Word с той лишь разницей, что распространяется этот текстовый процессор по лицензии LGPL v3 (в составе офисного пакета Libre Office).

# Лист

*№ докум.*

### Подпись

## Изм

### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

6

**Описание языка программирования**

В качестве средства реализации программного кода был выбран язык программирования C#, являющийся объектно- и компонентно-ориентированным языком программирования. Обладает рядом положительных качеств:

* C# – это объектно-ориентированный, простой и в то же время мощный язык программирования, позволяющий разработчикам создавать многофункциональные приложения.
* C# относится к языкам компилируемого типа, поэтому он обладает всеми преимуществами таких языков.
* C# объединяет лучшие идеи современных языков программирования Java, C++, Visual Basic и т.д.
* Из-за большого разнообразия синтаксических конструкций и возможности работать с платформой .Net, C# позволяет быстрее, чем любой другой язык, разрабатывать программные решения.
* C# отличается надежностью и наличием большого количества синтаксических конструкций.

# Лист

*№ докум.*

### Подпись

## Изм

### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

7

**Теоретический раздел**

Ниже приведен список терминов и их определения:

* Пользователь — человек, который имеет, имел, или, возможно, будет иметь доступ в систему для совершения операций.
* Диаграмма - графическое представление данных линейными отрезками.
* Сообщение — набор символов, введенных в программу для шифровки\дешифровки.
* Шифрование - обратимое преобразование информации в целях сокрытия от неавторизованных лиц.
* Расшифрование - процесс преобразования зашифрованного сообщения в открытый текст, когда известен алгоритм шифрования.

# Лист

*№ докум.*

### Подпись

## Изм

### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

8

**Постановка задачи**

Целями данной работы является:

1. Составление технического задания;
2. Составление пояснительной записки, включающей в себя помимо прочего:
   * Теоретический раздел;
   * Эскизный проект;
   * Технический проект;
   * Пользовательские сценарии.
3. Написание программного продукта в соответствии с поставленными условиями.

# Лист

*№ докум.*

### Подпись

## Изм

### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

9

**Эскизный проект**

1. Функция шифровки сообщений 1-м способом.

Входные данные: введённое в поле сообщение, соответствующее диапазону типа данных String.

Выходные данные: зашифрованный текст\сообщение об ошибке;

Описание функции: пользователь вводит в поле сообщение;

Имеется текст – ключ шифра, содержащий достаточное количество слов. Определяется частота появления каждого символа в этом тексте и в зависимости от нее символу присваивается код – число или буква. При зашифровке сообщения каждый символ заменяется его кодом. Если какое-то поле будет пустым, то отобразится сообщение об ошибке.

1. Функция шифровки сообщений 2-м способом.

Входные данные: введённое в поле сообщение, соответствующее диапазону типа данных String;

Выходные данные: зашифрованный текст\сообщение об ошибке;

Описание функции: вводится сообщение. Каждому символу присваивается код в зависимости от очередности его появления в тексте – ключ. Если какое-то поле будет пустым, то отобразится сообщение об ошибке.

# Лист

*№ докум.*

### Подпись

## Изм

### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

10

1. Функция шифровки сообщений 3-м способом.

Входные данные: введённое в поле сообщение, соответствующее диапазону типа данных String;

Выходные данные: диаграмма\сообщение об ошибке;

Описание функции: вводится сообщение. Символы шифруемого текста заменяются на символы, полученные из исходного путем сдвига их кода ASCII на заданную величину. Кроме того, между символами сообщения помещаются дополнительные символы, выбранные случайным образом (их количество постоянно или подчиняется определенной закономерности). При расшифровке сообщения вывод на экран двух диаграмм, отражающих частоту появления символов в первичных и обработанных текстах. Если какое-то поле будет пустым, то отобразится сообщение об ошибке.

1. Функция дешифровки сообщений.

Входные данные: введённое в поле зашифрованное сообщение, соответствующее диапазону типа данных String, номер способа шифрования;

Выходные данные: сообщение, дешифрованное одним из трех способов\сообщение об ошибке;

Описание функции: вводится сообщение и способ, которым оно должно быть дешифровано. Если все поля заполнены верно, то пользователь получит результат в виде дешифрованного сообщения. Если какое-то поле будет пустым, то появится сообщение об ошибке.

1. Функция построения диаграмм для трех способов дешифровки.

Входные данные: для построения частотной диаграммы по исходному текста – исходный текст, для построения частотной диаграммы по обработанному тексту – обработанный текст соответственно;

Выходные данные: две диаграммы, для исходного и обработанного текста соответственно;

Описание функции: при расшифровке сообщения осуществляется вывод на экран двух диаграмм, отражающих частоту появления символов в первичных и обработанных сообщениях.

# Лист

*№ докум.*

### Подпись

## Изм

### Дата

### Лист

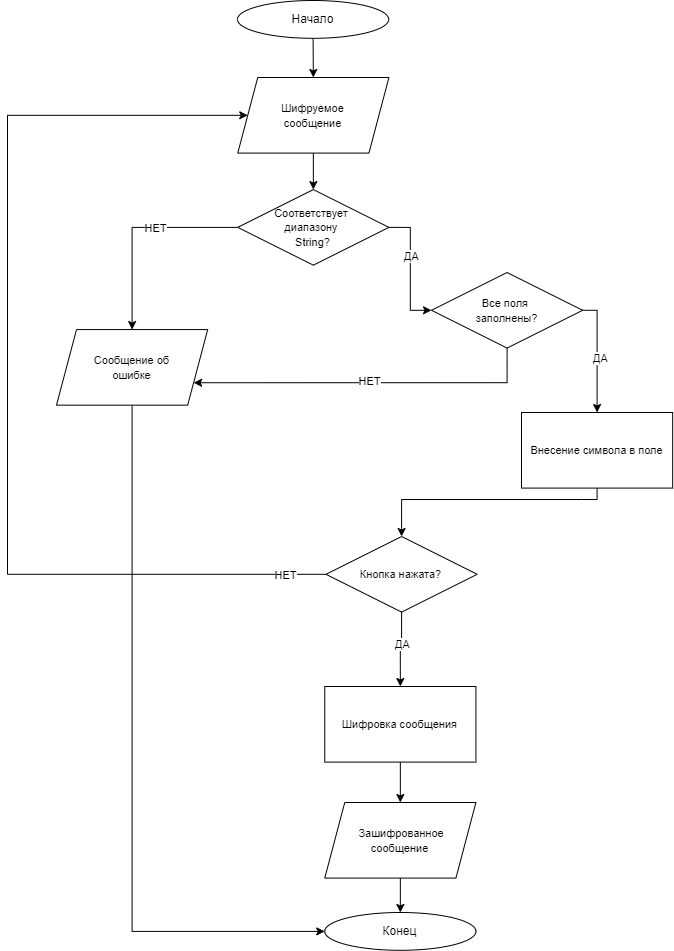
#### ОПП.17.230401.232.01.ПЗ

11

**Технический проект**

В приложении доступно три способа для шифровки введенного пользователем текста и три способа для дешифровки.

1. Функция шифровки сообщений первым способом:



# Лист

*№ докум.*

### Подпись

## Изм

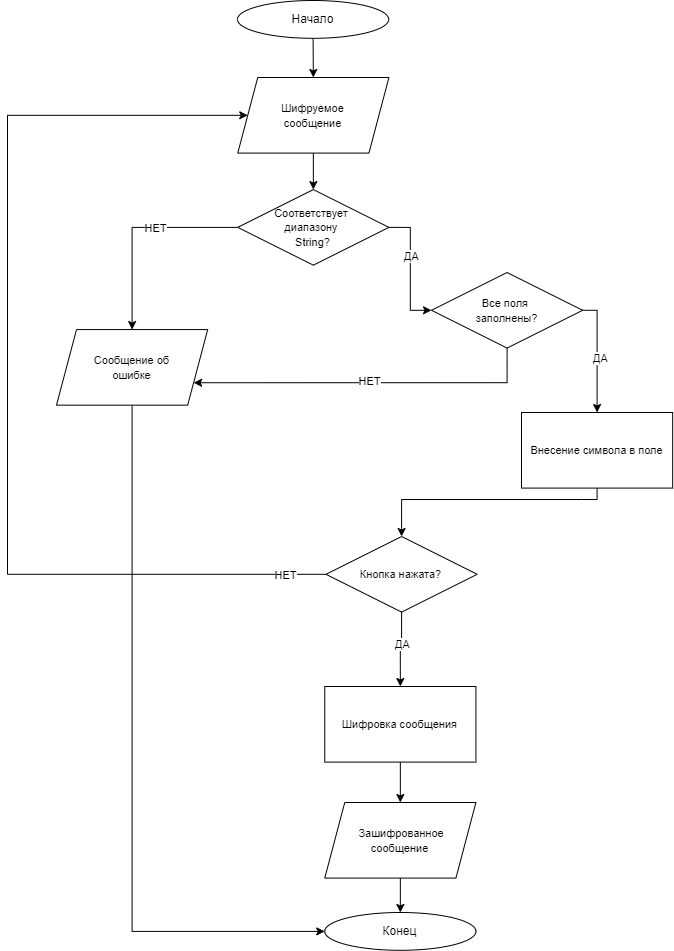
### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

12

1. Функция шифровки сообщений вторым способом:



# Лист

*№ докум.*

### Подпись

## Изм

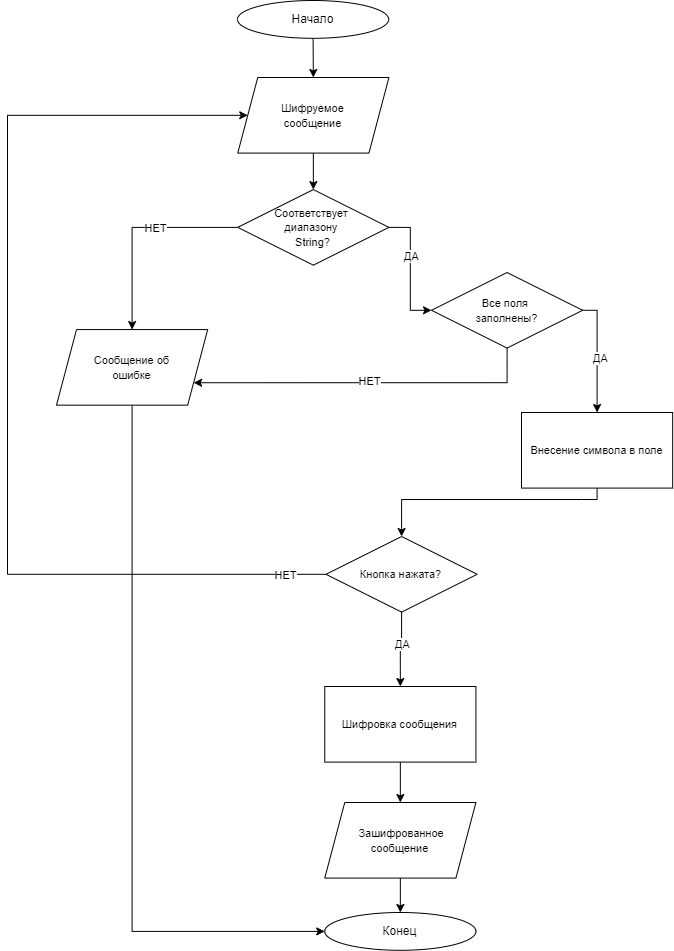
### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

13

1. Функция шифровки сообщений третьим способом:



# Лист

*№ докум.*

### Подпись

## Изм

### Дата

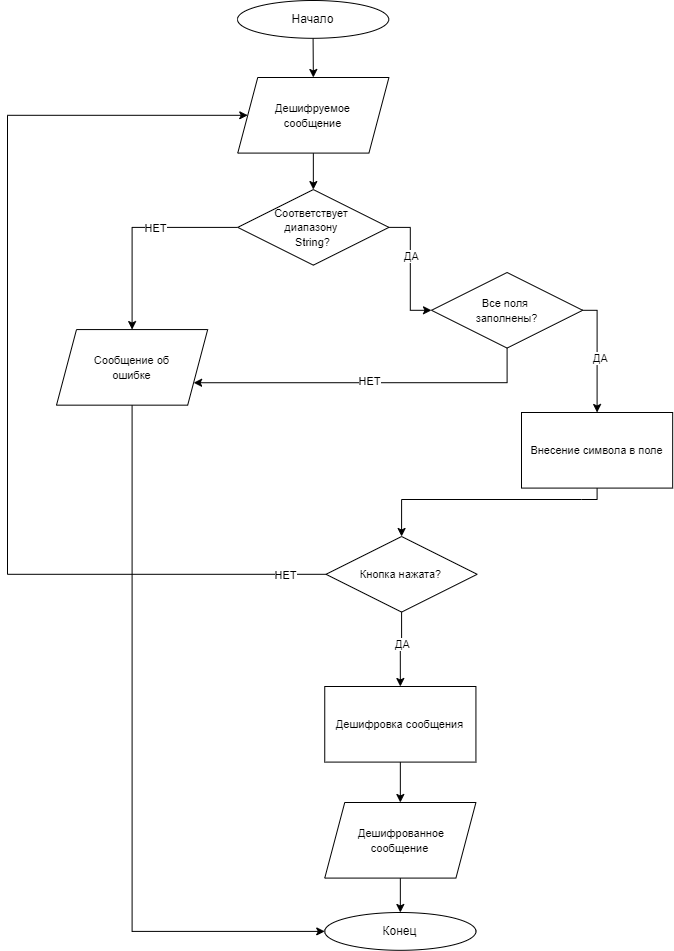
### Лист

#### ОПП.17.230401.232.01.ПЗ

14

15

1. Функция дешифровки сообщений:



# Лист

*№ докум.*

### Подпись

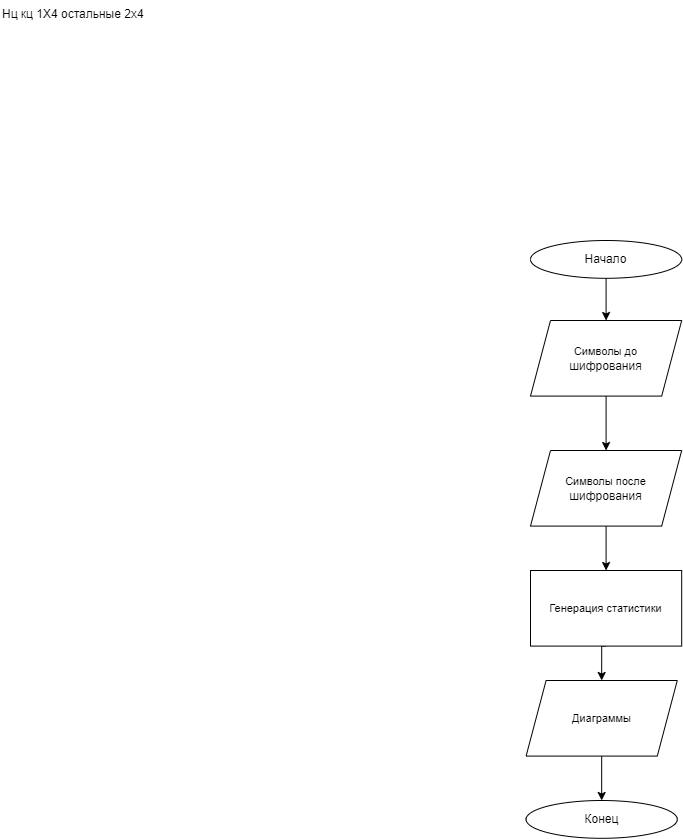
## Изм

### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

1. Функция построения диаграмм:



# Лист

*№ докум.*

### Подпись

## Изм

### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

16

**Инструкция пользователя**

В приложении доступно три способа для шифровки введенного пользователем текста и три способа для дешифровки.

1. Шифровка сообщений

Для шифрования текста любым из трех способов в окно консоли требуется ввести E/e и выбрать один из трех способов, при этом доступно краткое описание особенностей каждого способа;

Далее ввести шифруемый текст и нажать Enter (справедливо для первых двух способов). Для третьего способа необходимо ввести значение сдвига, запомнить это значение, запомнить сгенерированный программой Seed, ввести шифруемое сообщение и нажать Enter.

1. Дешифровка сообщений

Для дешифровки текста любым из трех способов в окно консоли требуется ввести D/d и выбрать один из трех способов, которым расшифровываемое сообщение было зашифровано. Помимо прочего, в этом разделе реализовано построение диаграмм, основывающихся на частоте вхождения уникальных символов в исходный и обработанный тексты.

# Лист

*№ докум.*

### Подпись

## Изм

### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

17

**Перечень используемых источников**

Ниже перечислены источники, использовавшиеся при составлении документации для данного проекта:

Построение UML-диаграмм:

https://habr.com/ru/post/566218/

Бизнес-логика:

https://metanit.com/sharp/

https://habr.com/ru/post/232009/

Основы и принципы криптографии:

http://sumk.ulstu.ru/docs/mszki/www.college.ru/UDP/texts/zi04.html

Составление документации:

https://intuit.ru/studies/courses/2195/55/lecture/15050?page=2

# Лист

*№ докум.*

### Подпись

## Изм

### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

18

Приложение A. Листинг программы

using System.Text;

#pragma warning disable CS8600 // Преобразование литерала, допускающего значение NULL или возможного значения NULL в тип, не допускающий значение NULL.

#pragma warning disable CS8604 // Возможно, аргумент-ссылка, допускающий значение NULL.

namespace CryptorNew

{

internal class Encryptor\_base

{

public static void Main(string[] args)

{

Console.OutputEncoding = System.Text.Encoding.UTF8;

Console.WriteLine("Encryptor\_base.cs");

//вспомогательный метод для шифровки/дешифровки первым способом

FreqDictGenerator(out List<char> freqsequence\_list);

//лист для шифровки/дешифроки вторым способом

List<char> positions\_list = PositionsListGenerator();

Console.WriteLine("Шифровать или дешифровать? (E / D)");

string choose = Convert.ToString(Console.ReadLine());

//ШИФРОВКА

if (choose == "E" || choose == "e")

{

Console.WriteLine("Введите шифруемое сообщение");

string input\_string = Console.ReadLine();

string output\_encrypted = "";

Console.WriteLine("Выберите способ шифровки (1 / 2 / 3)");

Console.WriteLine

(

$"1) поддерживает ограниченный набор символов, содержащийся в частотном словаре \n" +

$"2) поддерживает латиницу, кириллицу и распространенные спецсимволы \n" +

$"3) поддерживает ТОЛЬКО латиницу и спецсимволы(кодировка ASCII) \n"

);

int encryption\_method = Convert.ToInt32(Console.ReadLine());

switch (encryption\_method)

{

#region Encryption, method 1

case 1://шифрование на основе частоты появления символов в исходном тексте, готово

Console.WriteLine("Зашифрованное сообщение:\n");

EncryptAs1(freqsequence\_list, input\_string, ref output\_encrypted);

Console.WriteLine(output\_encrypted);

break;

#endregion

#region Encryption, method 2

case 2: //порядковый номер символа в листе символов, готово

//Console.WriteLine("Ёмкость словаря: " + positions\_list.Count);

Console.WriteLine("Зашифрованное сообщение:\n");

EncryptAs2(positions\_list, input\_string, out output\_encrypted);

Console.WriteLine(output\_encrypted);

break;

#endregion

# Лист

*№ докум.*

### Подпись

## Изм

### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

19

#region Encryption, method 3

case 3://шифрование посредством вывода кодов символов + сдвиг + межсимвольный мусор

//генерация межсимвольного мусора

int seed = SeedGenerator();//диапазон кодов символов, доступных пользователю для ввода одной клавишей

//составление внутреннего seed с маркерами и вывод пользовательского seed

Console.WriteLine("Введите значение сдвига, число от 1 до 10: ");

int ascii\_shift = Convert.ToInt32(Console.ReadLine());

string seed\_string = InternalSeedBuilder(seed);

UserSeedOut(seed);

Console.WriteLine("Зашифрованное сообщение: \n");

string encrypted\_symbols\_string = EncryptAs3(input\_string, ascii\_shift, seed\_string);

Console.WriteLine(encrypted\_symbols\_string);

break;

#endregion

default:

ErrorMessage();

break;

}

}

else if (choose == "D" || choose == "d")

{

Console.WriteLine("Введите дешифруемое сообщение");

string input\_encrypted = Console.ReadLine();

Console.WriteLine("Выберите способ дешифровки (1 / 2 / 3)");

int decryption\_method = int.Parse(Console.ReadLine());

switch (decryption\_method)

{

#region Decryption, method 1

case 1:

string return\_for\_analysis;

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine("Дешифрованное сообщение:");

return\_for\_analysis = DecryptAs1(freqsequence\_list, input\_encrypted);

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine("Диаграмма, отражающая частоту вхождения символов в первичный текст:");

StringAnalyzer(input\_encrypted);

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine("Диаграмма, отражающая частоту вхождения символов в обработанный текст:");

StringAnalyzer(return\_for\_analysis);

break;

#endregion

# Лист

*№ докум.*

### Подпись

## Изм

### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

20

#region Decryption, method 2

case 2:

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine("Дешифрованное сообщение:");

return\_for\_analysis = DecryptAs2(positions\_list, input\_encrypted);

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine("Диаграмма, отражающая частоту вхождения символов в первичный текст:");

StringAnalyzer(input\_encrypted);

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine("Диаграмма, отражающая частоту вхождения символов в обработанный текст:");

StringAnalyzer(return\_for\_analysis);

break;

#endregion

#region Decryption, method 3

case 3:

return\_for\_analysis = DecryptAs3(input\_encrypted);

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine("Диаграмма, отражающая частоту вхождения символов в первичный текст:");

StringAnalyzer(input\_encrypted);

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine("Диаграмма, отражающая частоту вхождения символов в обработанный текст:");

StringAnalyzer(return\_for\_analysis);

break;

#endregion

default:

ErrorMessage();

break;

}

}

else

{

ErrorMessage();

}

}

#region First encryptor methods

private static char[] EncryptAs1(List<char> freqsequence\_list, string input\_string, ref string output\_encrypted)

{

char[] inputChar\_arr = input\_string.ToCharArray();

inputChar\_arr = input\_string.ToCharArray();

for (int x = 0; x < input\_string.Length; x++)

{

int index\_of = freqsequence\_list.IndexOf(inputChar\_arr[x]);

output\_encrypted = output\_encrypted + index\_of + " ";

}

return inputChar\_arr;

}

#endregion

# Лист

*№ докум.*

### Подпись

## Изм

### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

21

#region Second encryptor methods

private static void EncryptAs2(List<char> dictionary, string input\_string, out string output\_encrypted)

{

char[] inputChar\_arr = input\_string.ToCharArray();

output\_encrypted = "";

for (int x = 0; x < input\_string.Length; x++)

{

int index\_of = dictionary.IndexOf(inputChar\_arr[x]);

output\_encrypted = output\_encrypted + index\_of + " ";

}

}

#endregion

#region Third encryptor methods

private static string EncryptAs3(string input\_string, int ascii\_shift, string seed\_string)

{

byte[] ascii\_bytes = Encoding.ASCII.GetBytes(input\_string);//получение байтов символов в кодировке ASCII (латиница)

string encrypted\_symbols\_string = "";

foreach (int ascii\_bytes\_of\_element in ascii\_bytes)

{

string encrypted\_symbol = (ascii\_bytes\_of\_element - ascii\_shift + seed\_string);//байтовое значение символа-сдвиг+добавление подстроки seed

encrypted\_symbols\_string += encrypted\_symbol;

}

return encrypted\_symbols\_string;

}

#endregion

#region First decryptor methods

private static string DecryptAs1(List<char> freqsequence\_list, string input\_encrypted)

{

string input\_index\_str = input\_encrypted;

string input\_index\_int = input\_index\_str.Replace(",", string.Empty);

int \_ind = 0;

int[] inputIndex\_arr = new int[\_ind];

string return\_for\_analysis = "";

inputIndex\_arr = input\_index\_int.Split(' ').Select(int.Parse).ToArray();

for (int i = 0; i < inputIndex\_arr.Length; i++)//повторяется, пока не закончатся символы в строке

{

for (int j = 0; j < freqsequence\_list.Count; j++)//перебирает все символы в словаре

{

if (inputIndex\_arr[i] == j)

{

return\_for\_analysis += Convert.ToString(freqsequence\_list[j]);

Console.Write(freqsequence\_list[j]);

}

}

}

Console.WriteLine();

return return\_for\_analysis;

}

#endregion

# Лист

*№ докум.*

### Подпись

## Изм

### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

22

23

#region Second encryptor methods

private static void EncryptAs2(List<char> dictionary, string input\_string, out string output\_encrypted)

{

char[] inputChar\_arr = input\_string.ToCharArray();

output\_encrypted = "";

for (int x = 0; x < input\_string.Length; x++)

{

int index\_of = dictionary.IndexOf(inputChar\_arr[x]);

output\_encrypted = output\_encrypted + index\_of + " ";

}

}

#endregion

#region Third encryptor methods

private static string EncryptAs3(string input\_string, int ascii\_shift, string seed\_string)

{

byte[] ascii\_bytes = Encoding.ASCII.GetBytes(input\_string);//получение байтов символов в кодировке ASCII (латиница)

string encrypted\_symbols\_string = "";

foreach (int ascii\_bytes\_of\_element in ascii\_bytes)

{

string encrypted\_symbol = (ascii\_bytes\_of\_element - ascii\_shift + seed\_string);//байтовое значение символа-сдвиг+добавление подстроки seed

encrypted\_symbols\_string += encrypted\_symbol;

}

return encrypted\_symbols\_string;

}

#endregion

#region First decryptor methods

private static string DecryptAs1(List<char> freqsequence\_list, string input\_encrypted)

{

string input\_index\_str = input\_encrypted;

string input\_index\_int = input\_index\_str.Replace(",", string.Empty);

int \_ind = 0;

int[] inputIndex\_arr = new int[\_ind];

string return\_for\_analysis = "";

inputIndex\_arr = input\_index\_int.Split(' ').Select(int.Parse).ToArray();

for (int i = 0; i < inputIndex\_arr.Length; i++)//повторяется, пока не закончатся символы в строке

{

for (int j = 0; j < freqsequence\_list.Count; j++)//перебирает все символы в словаре

{

if (inputIndex\_arr[i] == j)

{

return\_for\_analysis += Convert.ToString(freqsequence\_list[j]);

Console.Write(freqsequence\_list[j]);

}

}

}

Console.WriteLine();

return return\_for\_analysis;

}

#endregion

# Лист

*№ докум.*

### Подпись

## Изм

### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

24

#region Second decryptor methods

private static string DecryptAs2(List<char> dictionary, string input\_encrypted)

{

string return\_for\_analysis = "";

string input\_index\_str = input\_encrypted;

string input\_index\_int = input\_index\_str.Replace(",", string.Empty);

int \_ind = 0;

int[] inputIndex\_arr = new int[\_ind];

inputIndex\_arr = input\_index\_int.Split(' ').Select(int.Parse).ToArray();

for (int i = 0; i < inputIndex\_arr.Length; i++)//повторяется, пока не закончатся символы в строке

{

for (int j = 0; j < dictionary.Count; j++)//перебирает все символы в словаре

{

if (inputIndex\_arr[i] == j)

{

return\_for\_analysis += Convert.ToString(dictionary[j]);

Console.Write(dictionary[j]);

}

}

}

Console.WriteLine();

return return\_for\_analysis;

}

#endregion

# Лист

*№ докум.*

### Подпись

## Изм

### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

#region Third decryptor methods

private static string DecryptAs3(string input\_encrypted)

{

string return\_for\_analysis = "";

Console.WriteLine("Введите seed, один символ:");

string seed\_string = DecryptionSeedBuilder();

Console.WriteLine("Введите значение сдвига");

int ascii\_shift = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine("Дешифрованное сообщение:");

string input\_encrypted\_without\_seed = input\_encrypted.Replace(seed\_string, " ");

//

List<int> int\_chars\_list = new();

string[] output\_arr = input\_encrypted\_without\_seed.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

foreach (string s in output\_arr)

{

if (!int.TryParse(s, out int temp))

{

throw new Exception("Wrong argument!");

}

else

{

int\_chars\_list.Add(temp);

}

}

foreach (int i in int\_chars\_list)

{

return\_for\_analysis += Convert.ToString((char)(i + ascii\_shift));

Console.Write((char)(i + ascii\_shift));

}

Console.WriteLine();

return return\_for\_analysis;

}

#endregion

private static void FreqDictGenerator(out List<char> freqsequence\_list)

{

string freqsequence\_string = AbbcccStringBuilder();//построение частотной строки

Dictionary<char, int> freqdict = SymbolRepeat\_Counter(freqsequence\_string);//пересчет вхождений каждого символа в частотную строку

//перемещение ключей(символов) в порядке увеличения числа вхождений в массив(теперь кол-во вхождений элемента == индекс этого же элемента в массиве)

freqsequence\_list = FreqSequenceListBuilder(freqdict);

}

# Лист

*№ докум.*

### Подпись

## Изм

### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

25

26

private static List<char> FreqSequenceListBuilder(Dictionary<char, int> freqdict)

{

List<char> freqsequence\_list = new()

{

(char)10060//заполнение нулевого индекса листа, подгонка индексации к частоте вхождения

};

freqsequence\_list = freqdict.Keys.ToList();

return freqsequence\_list;

}

private static Dictionary<char, int> SymbolRepeat\_Counter(string freqsequence\_string)

{

char alphabet\_char;

Dictionary<char, int> freqdict = new();

foreach (char ch in freqsequence\_string)

{

alphabet\_char = ch;

if (freqdict.ContainsKey(alphabet\_char))

freqdict[alphabet\_char]++;

else

freqdict.Add(alphabet\_char, 1);

}

return freqdict;

}

private static string DecryptionSeedBuilder()

{

char seed\_char = Convert.ToChar(Console.ReadLine()); //введенный символ конвертится из string в char

int seed\_char\_int = (int)seed\_char; //далее извлекается код символа

string seed\_string = "!-" + seed\_char\_int + "-!"; //и строится полноценная строка с маркерами начала\конца seed'a

return seed\_string;

}

private static string InternalSeedBuilder(int seed)//построение внутреннего seed для построения итоговой строки

{

return "!-" + seed.ToString() + "-!";

}

# Лист

*№ докум.*

### Подпись

## Изм

### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

27

private static List<char> PositionsListGenerator()//создание листа символов для дальнейшего вывода индексов символов

{

return new()

{

't',

'h',

'e',

'q',

'u',

'i',

'c',

'k',

'b',

'r',

'o',

'w',

'n',

'f',

'x',

'j',

'm',

'p',

's',

'v',

'l',

'a',

'z',

'y',

'd',

'g',

'T',

'H',

'E',

'Q',

'U',

'I',

'C',

'K',

'B',

'R',

'O',

'W',

'N',

'F',

'X',

'J',

'M',

'P',

'S',

'V',

'L',

'A',

'Z',

'Y',

'D',

'G',

'с',

'ъ',

'е',

'ш',

'ь',

'ж',

'щ',

'ё',

'э',

# Лист

*№ докум.*

### Подпись

## Изм

### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

28

'т',

'и',

'х',

'м',

'я',

'г',

'к',

'ф',

'р',

'а',

'н',

'ц',

'у',

'з',

'б',

'л',

'о',

'д',

'в',

'ы',

'п',

'й',

'ч',

'ю',

'С',

'Ъ',

'Е',

'Ш',

'Ь',

'Ж',

'Щ',

'Ё',

'Э',

'Т',

'И',

'Х',

'М',

'Я',

'Г',

'К',

'Ф',

'Р',

'А',

'Н',

'Ц',

'У',

'З',

'Б',

'Л',

'О',

'Д',

'В',

'Ы',

'П',

'Й',

'Ч',

'Ю',

' ',

',',

'!',

'.',

'?',

'-',

'\_',

# Лист

*№ докум.*

### Подпись

## Изм

### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

29

'1',

'2',

'3',

'4',

'5',

'6',

'7',

'8',

'9',

'0'

};

}

private static string AbbcccStringBuilder()//создание строки повторяющихся символов для дальнейшего частотного анализа (способ №1)

{

string symbol\_string = "";

int repeat = 0;

for (int symbol = 32; symbol <= 126; symbol++, repeat++)//с 32 по 126 символ(латиница+символы)

{

for (int i = repeat; i >= 0; i--) symbol\_string = symbol\_string.Insert(symbol\_string.Length, Convert.ToString(Convert.ToChar(symbol)));

}

//пропуск неподдерживаемых консолью символов

for (int symbol = 1040; symbol <= 1103; symbol++, repeat++)//с 1040 по 1103 символ(кириллица, оба регистра)

{

for (int i = repeat; i >= 0; i--) symbol\_string = symbol\_string.Insert(symbol\_string.Length, Convert.ToString(Convert.ToChar(symbol)));

}

return symbol\_string;

}

private static void ErrorMessage()//вывод сообщения об ошибке для случая ввода некорректного значения

{

Console.Write("Error. Unexpected symbol");

}

private static void UserSeedOut(int seed)//вывод пользовательского seed(для копирования)

{

Console.WriteLine("Ваш seed: " + (char)seed);

}

private static int SeedGenerator()//генератор численного seed

{

Random random = new();

int seed = random.Next(33, 125);//диапазон кодов символов, доступных пользователю для ввода одной клавишей

return seed;

}

# Лист

*№ докум.*

### Подпись

## Изм

### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

30

#region Second decryptor methods

private static string DecryptAs2(List<char> dictionary, string input\_encrypted)

{

string return\_for\_analysis = "";

string input\_index\_str = input\_encrypted;

string input\_index\_int = input\_index\_str.Replace(",", string.Empty);

int \_ind = 0;

int[] inputIndex\_arr = new int[\_ind];

inputIndex\_arr = input\_index\_int.Split(' ').Select(int.Parse).ToArray();

for (int i = 0; i < inputIndex\_arr.Length; i++)//повторяется, пока не закончатся символы в строке

{

for (int j = 0; j < dictionary.Count; j++)//перебирает все символы в словаре

{

if (inputIndex\_arr[i] == j)

{

return\_for\_analysis += Convert.ToString(dictionary[j]);

Console.Write(dictionary[j]);

}

}

}

Console.WriteLine();

return return\_for\_analysis;

}

#endregion

# Лист

*№ докум.*

### Подпись

## Изм

### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

31

private static void StringAnalyzer(string parsed\_string)

{

string analyzer\_dict = "!\"#$%&'()\*+,-./0123456789:;<=>?@[\\]^\_`{|}~" +

"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ" +

"абвгдежзийклмнопрстуфхцчшщъыьэюяАБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";

Dictionary<char, int> dic = new();

foreach (char ch in analyzer\_dict)

dic.Add(ch, 0);

foreach (char ch in parsed\_string)

{

if (analyzer\_dict.Contains(ch.ToString()))

dic[ch]++;

}

foreach (var pair in dic)

if (pair.Value > 0)

Console.WriteLine("{0} {1}", pair.Key, string.Concat(Enumerable.Repeat("𝩋", pair.Value)));

}

}

}

# Лист

*№ докум.*

### Подпись

## Изм

### Дата

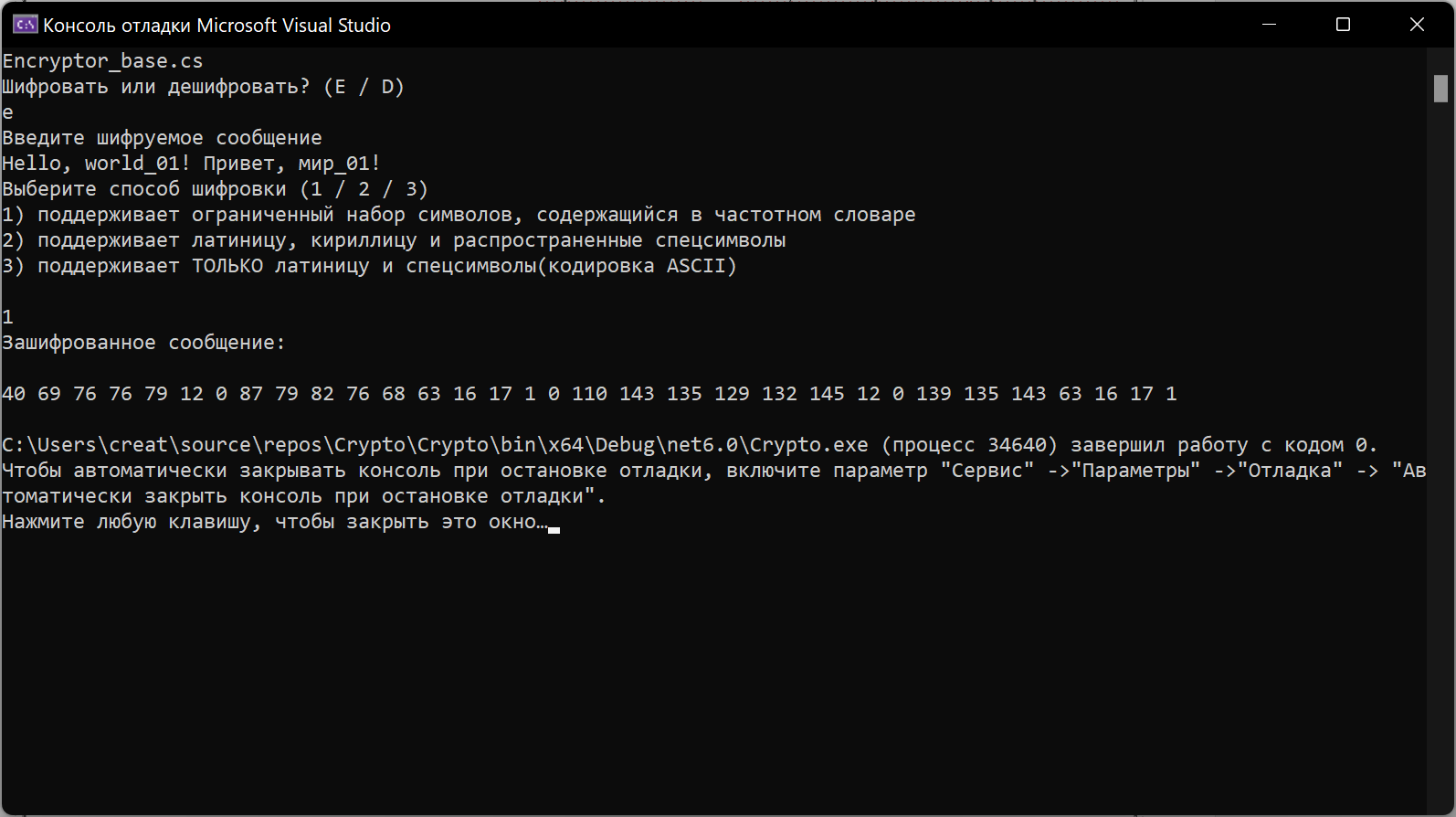
### Лист

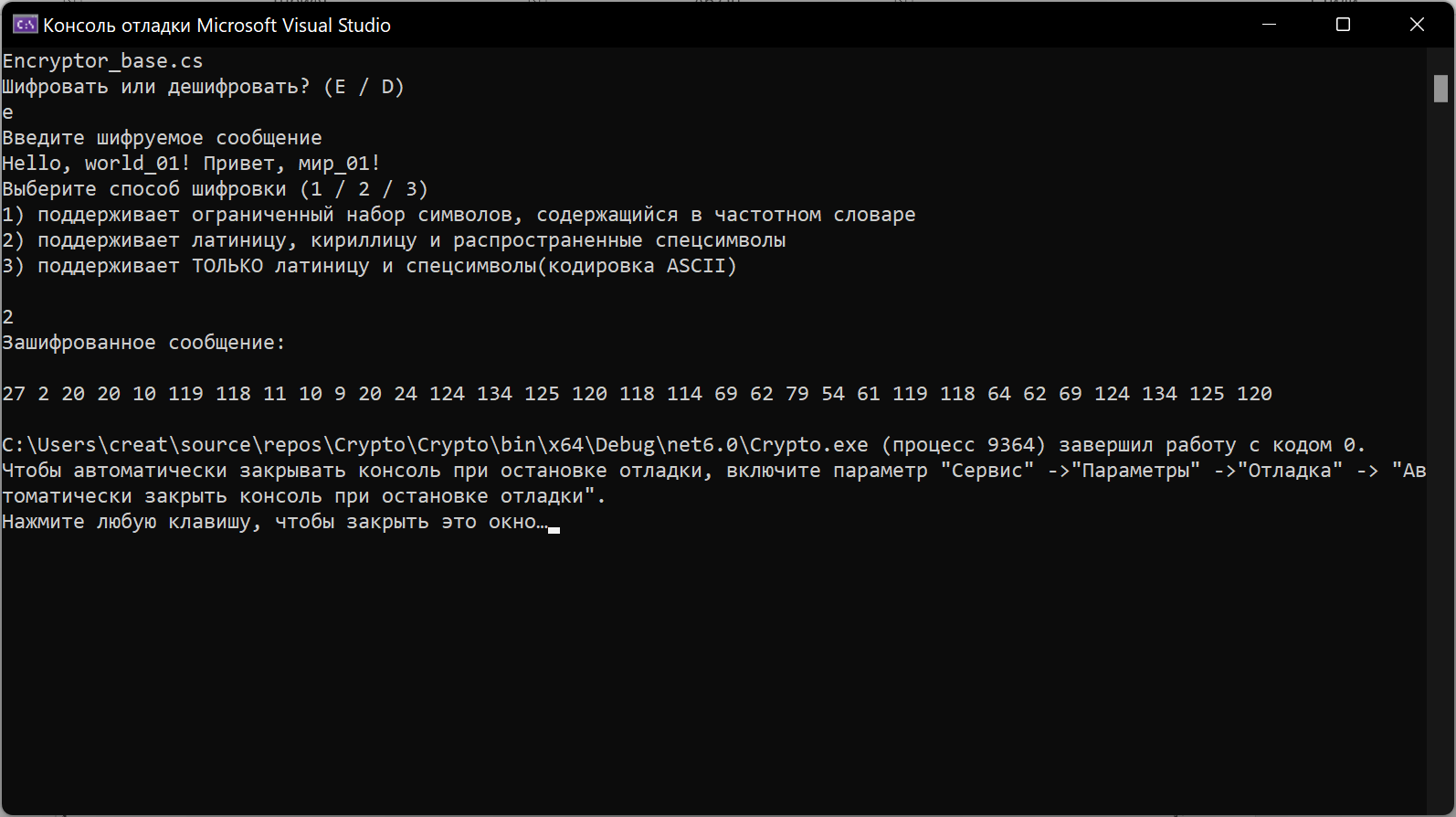
#### ОПП.17.230401.232.01.ПЗ

32

Приложение Б. Результат работы программы

Шифровка:





# Лист

*№ докум.*

### Подпись

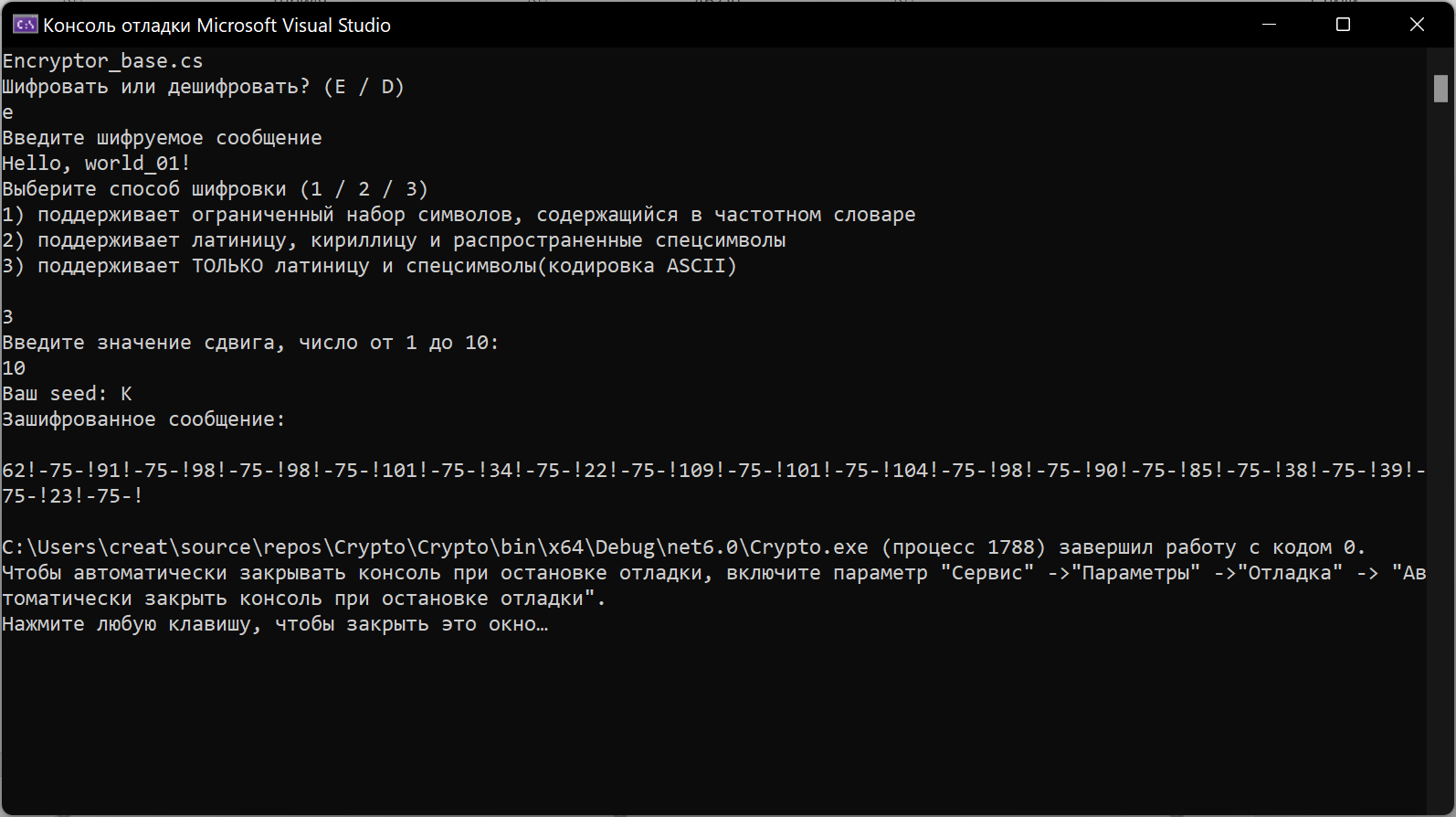
## Изм

### Дата

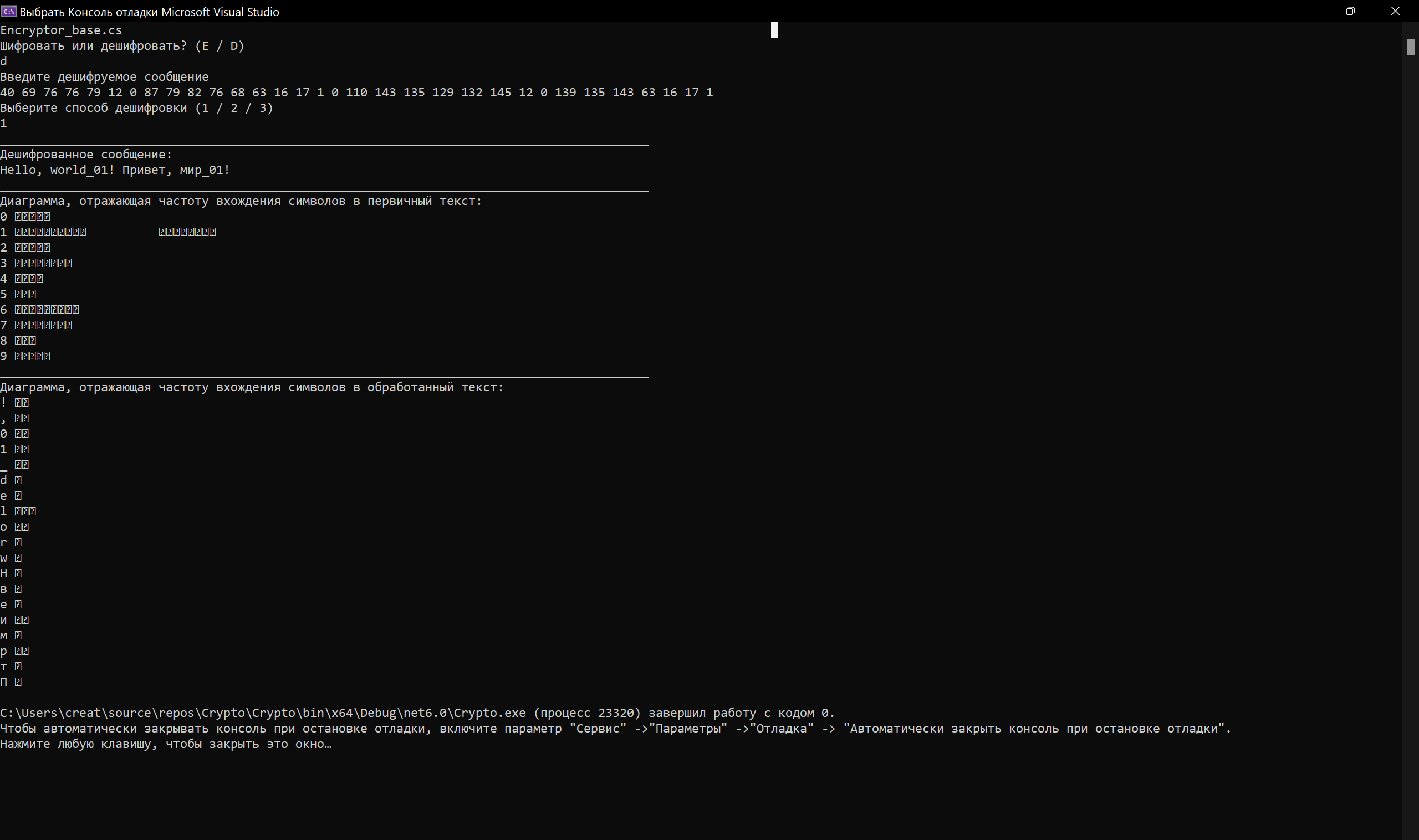
### Лист

#### ОПП.17.230401.232.01.ПЗ

33



Дешифровка:



# Лист

*№ докум.*

### Подпись

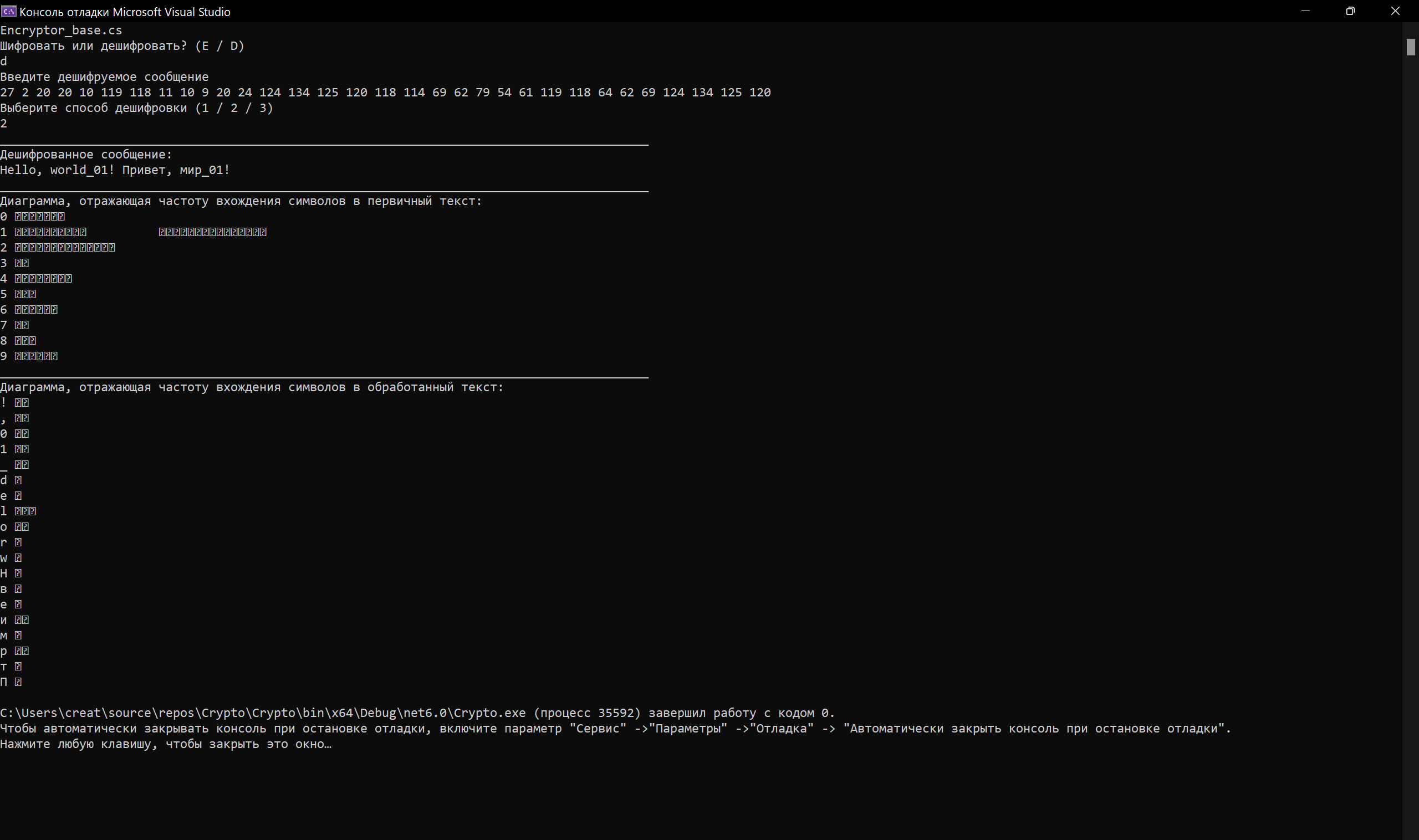
## Изм

### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

34



# Лист

*№ докум.*

### Подпись

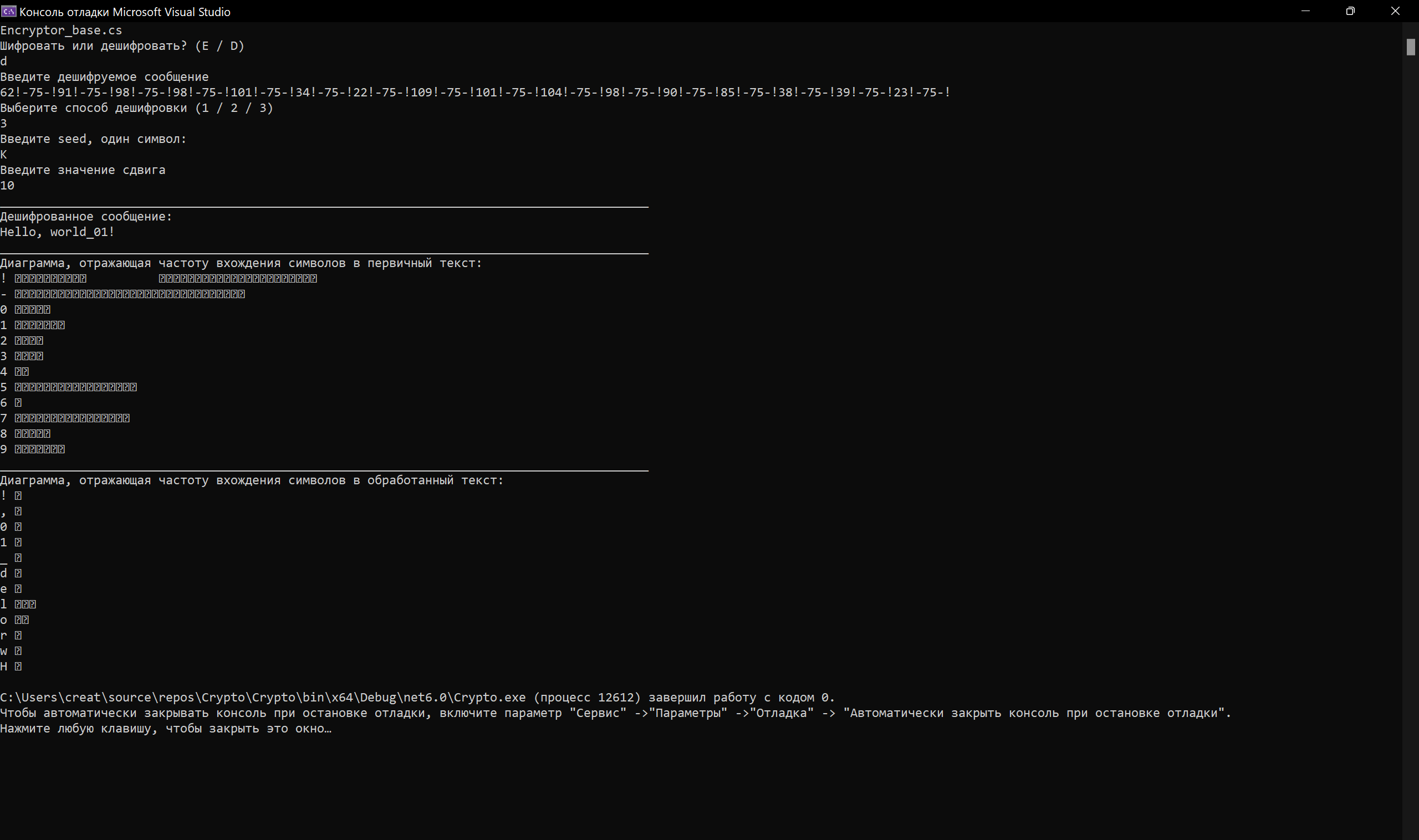
## Изм

### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ

35



# Лист

*№ докум.*

### Подпись

## Изм

### Дата

### Лист

#### ОПП.17.230401.232.01.ПЗ