ОБЛАСТНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
«ТОМСКИЙ ТЕХНИКУМ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ»

Специальность 09.02.07 «Информационные системы и программирование»

РАЗРАБОТКА ПРОГРАММЫ ШИФРОВКИ СООБЩЕНИЙ

Пояснительная записка
к учебной практике
УП.22.09.02.07.603.12.ПЗ

Студент
«__»_____ 2022 г. _____ Левицкий П. Д.
Руководитель
«__»_____ 2022 г. _____ Маюнова А. Ю.

Томск 2022

Содержание

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.

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.

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.

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).

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.

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.

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.

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.

2. 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.

3. 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.

4. 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.

5. Функция построения диаграмм для трех способов дешифровки.

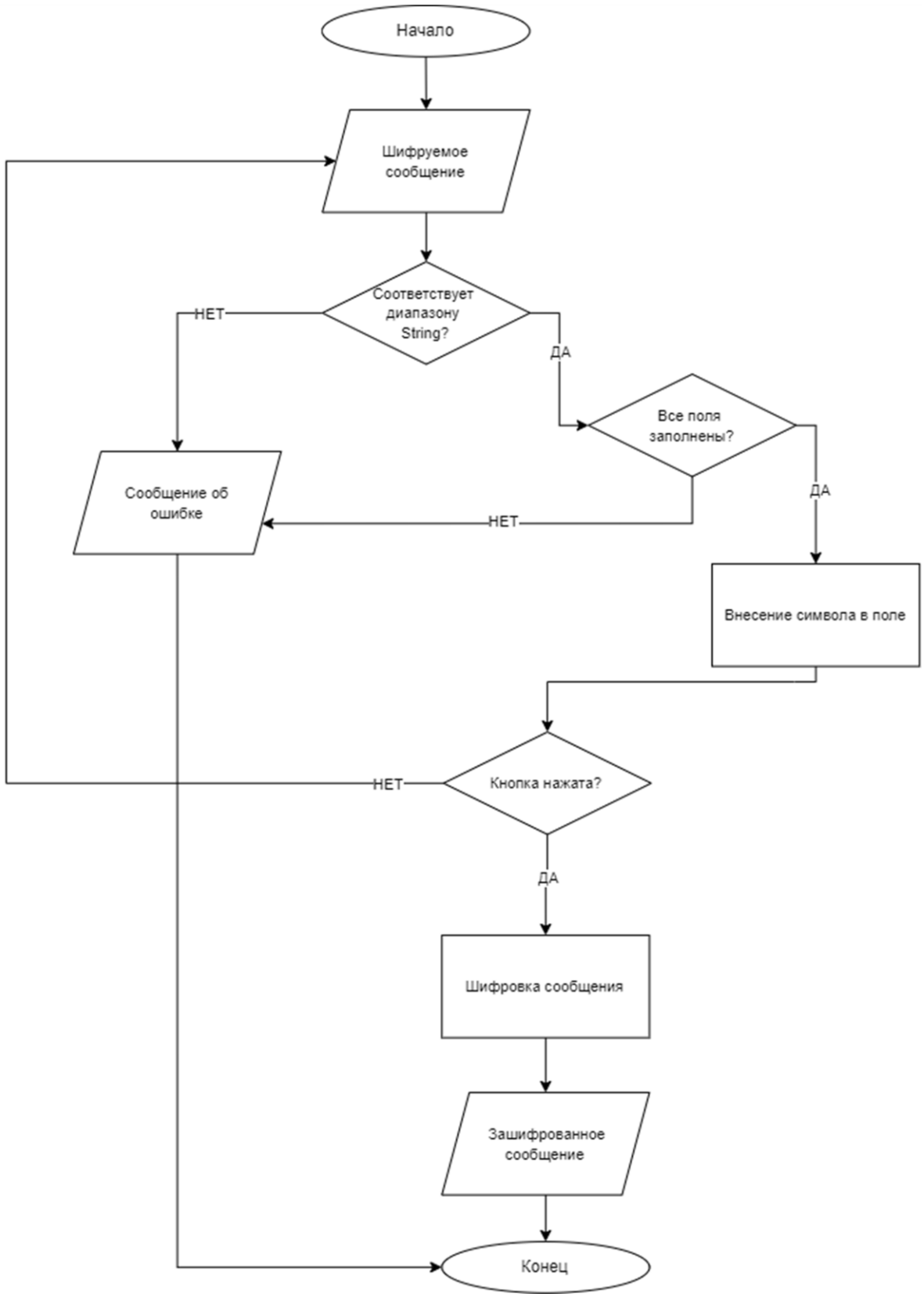
Входные данные: для построения частотной диаграммы по исходному текста – исходный текст, для построения частотной диаграммы по обработанному тексту – обработанный текст соответственно;

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

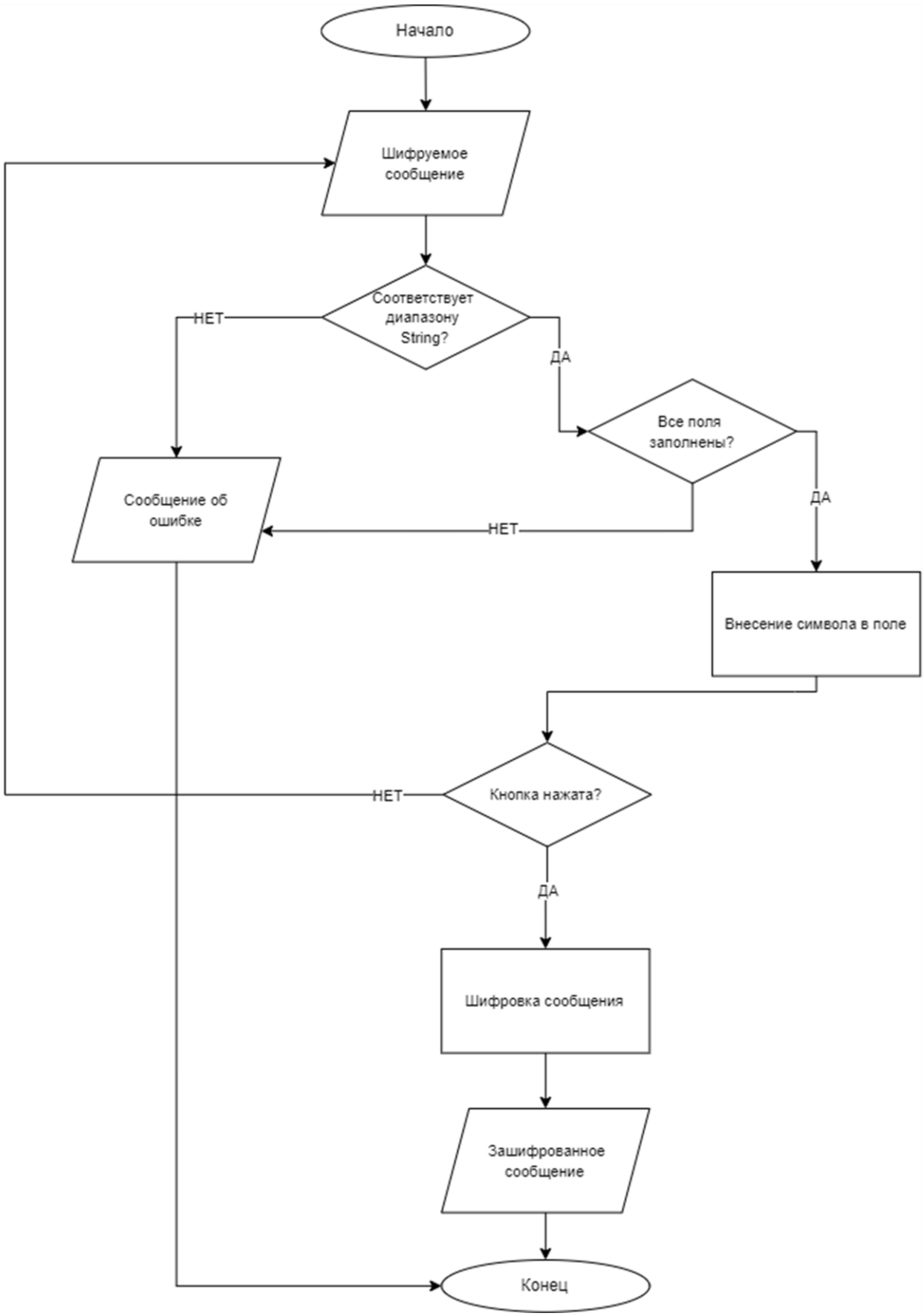
Описание функции: при расшифровке сообщения осуществляется вывод на экран двух диаграмм, отражающих частоту появления символов в первичных и обработанных сообщениях.

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

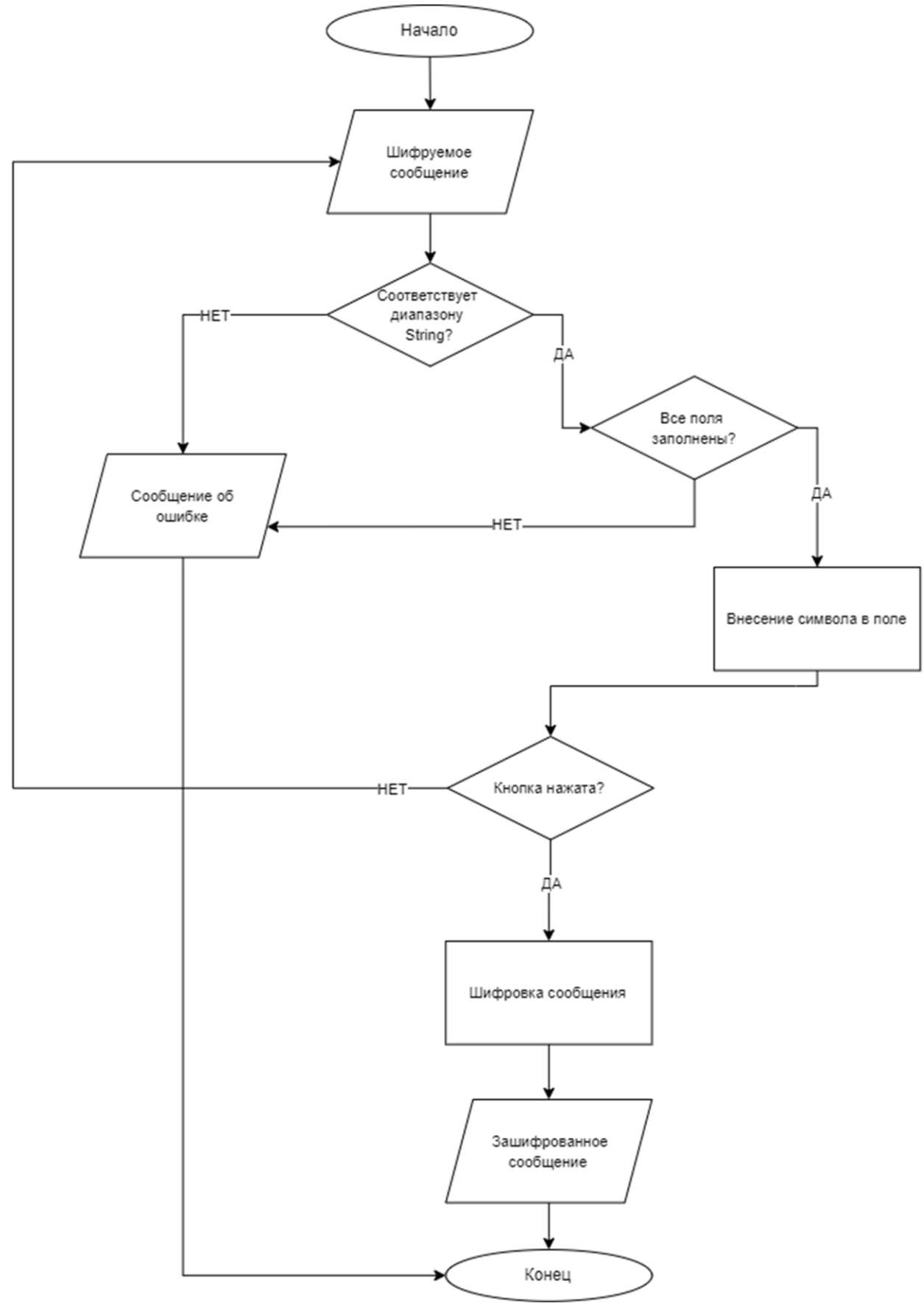
1. Функция шифровки сообщений первым способом:



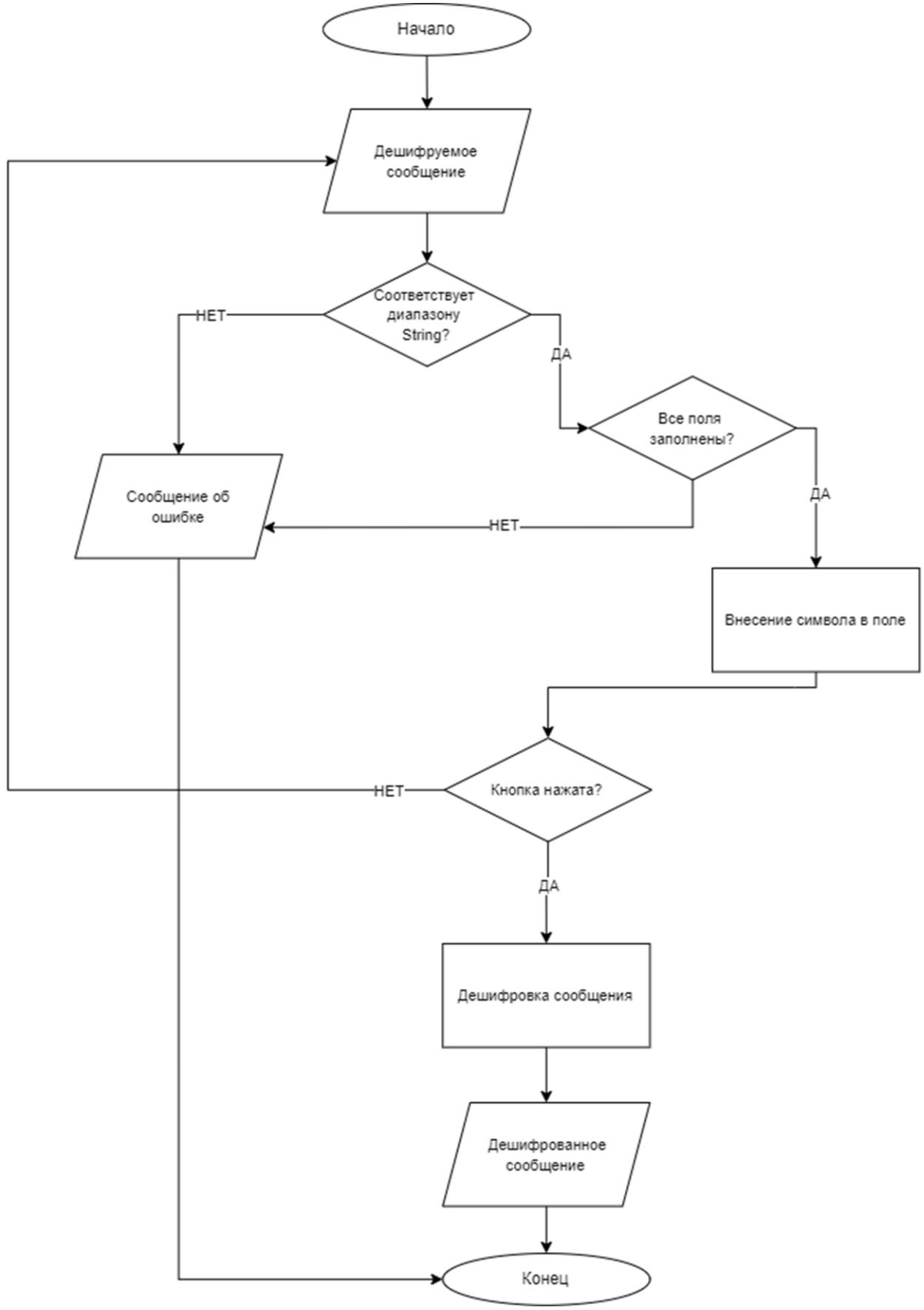
2. Функция шифровки сообщений вторым способом:



3. Функция шифровки сообщений третьим способом:



4. Функция дешифровки сообщений:



5. Функция построения диаграмм:

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.

2. 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.

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

```
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
```

```
#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
```

```
#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
```

```
#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
```

```
#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
```

```
#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
```

```
#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);
}
```

```
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() + "-!";
}
```

```
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',
'c',
'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',
'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',
' ',
',',
'!',
'.',
'?',
'–',
'_',
```

```
'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;
}
```

```
#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
```

```
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)));
}

}
}
```

Шифровка:

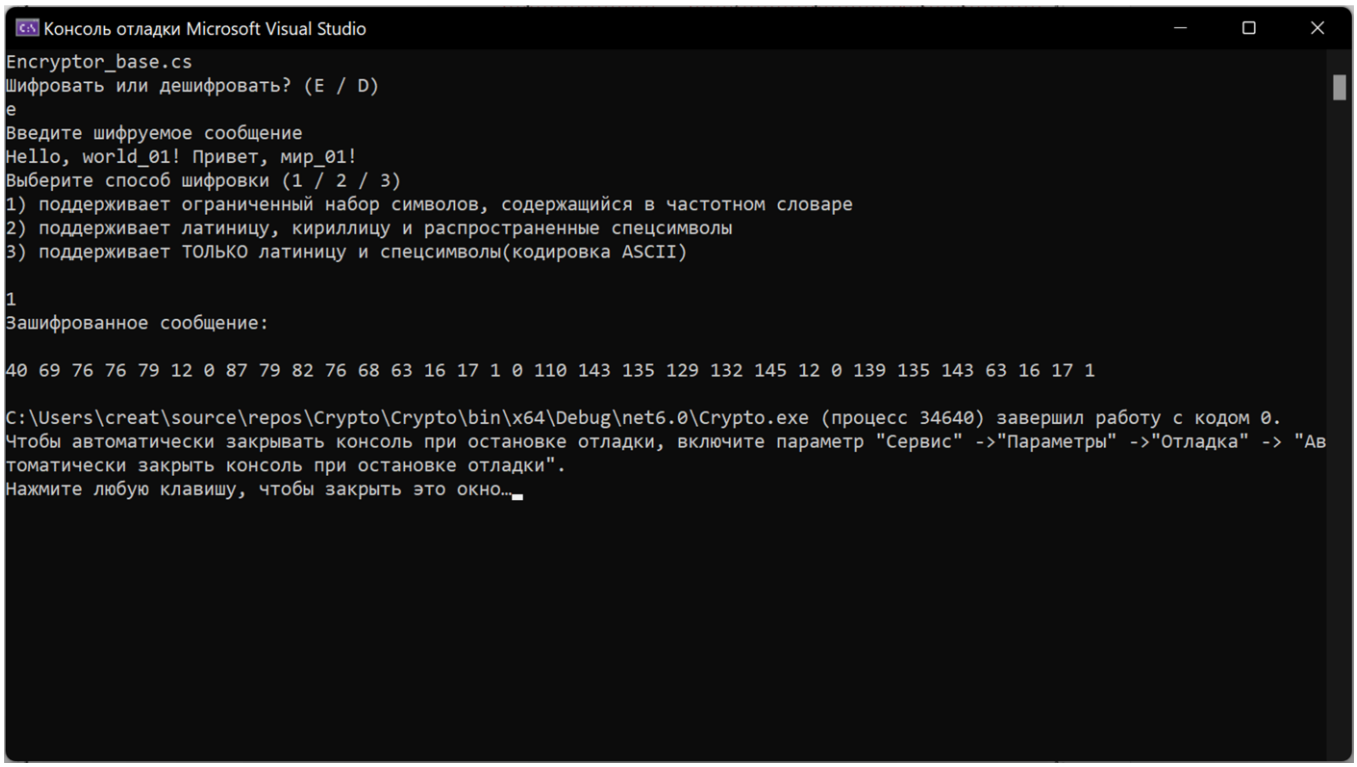


Рисунок 1 – шифровка сообщения первым способом

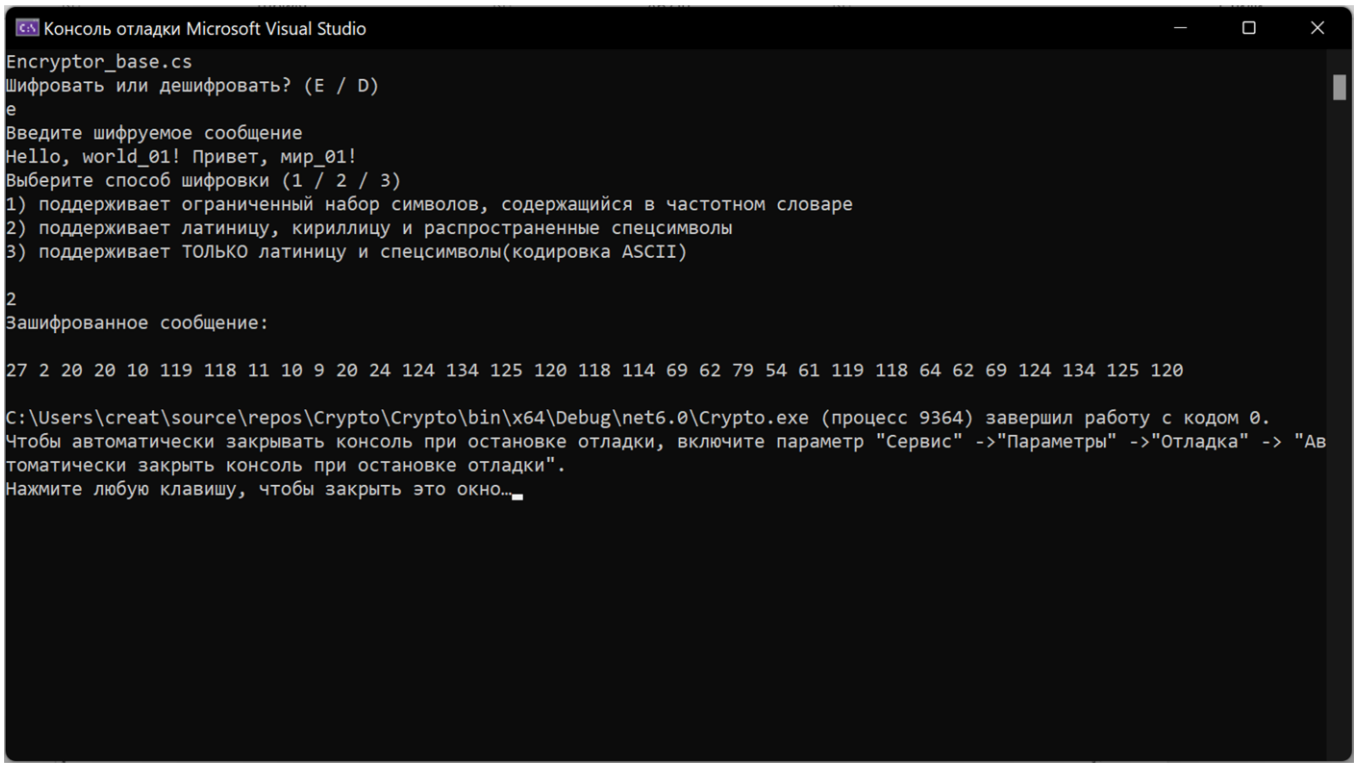


Рисунок 2 – шифровка сообщения вторым способом

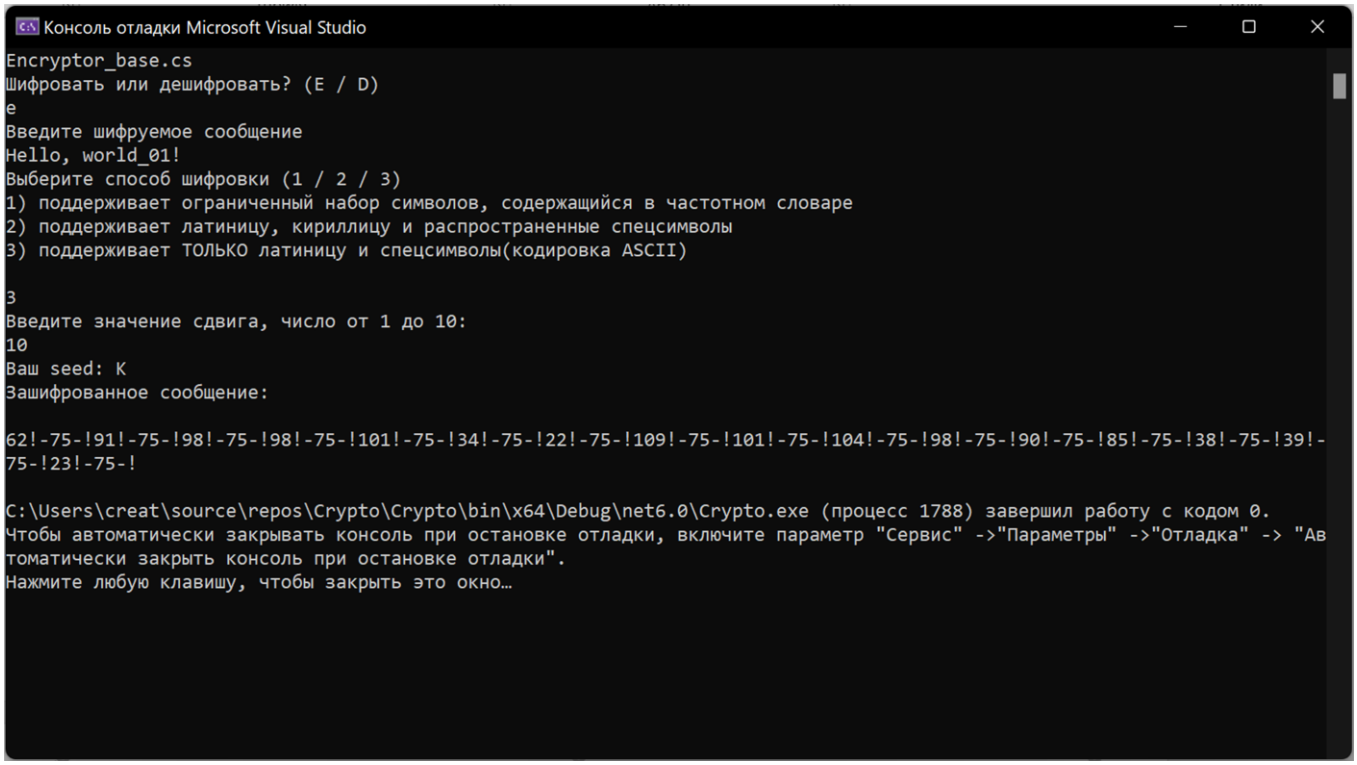


Рисунок 3 – шифровка сообщения третьим способом

Дешифровка:

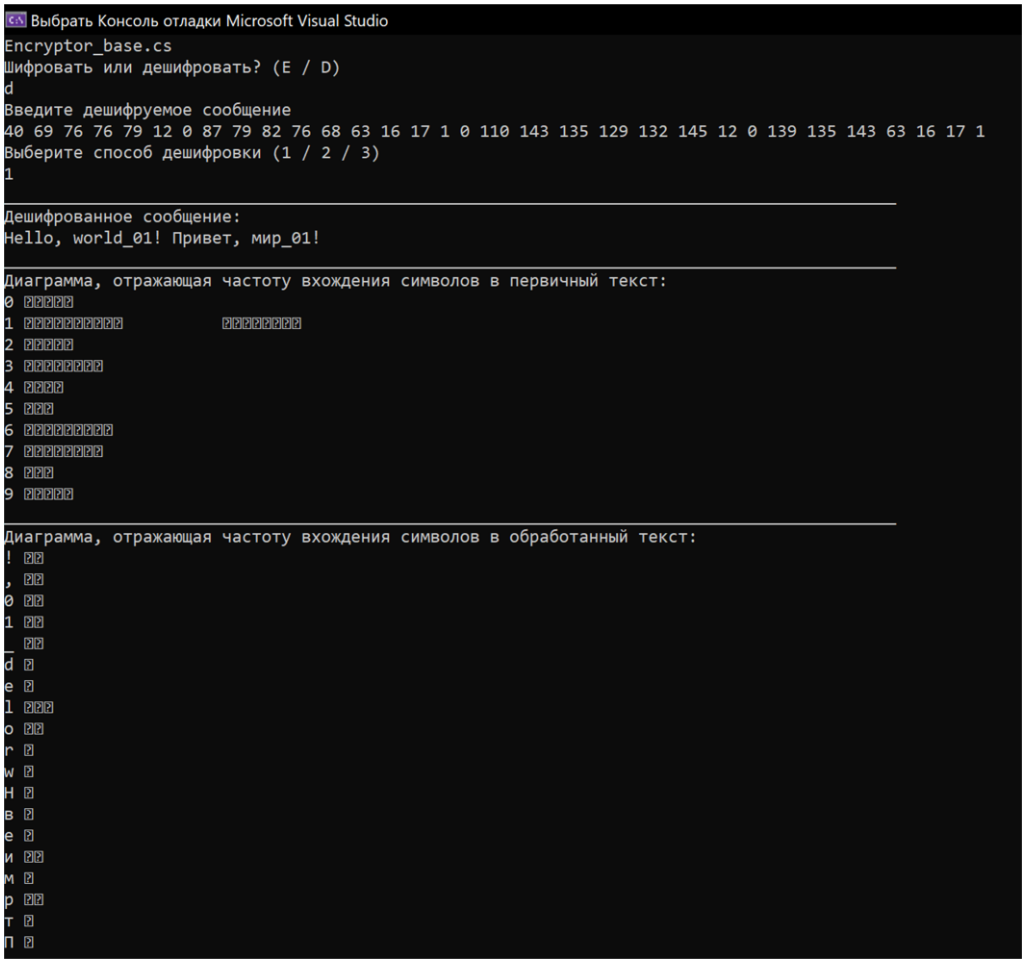


Рисунок 4 – дешифровка сообщения первым способом

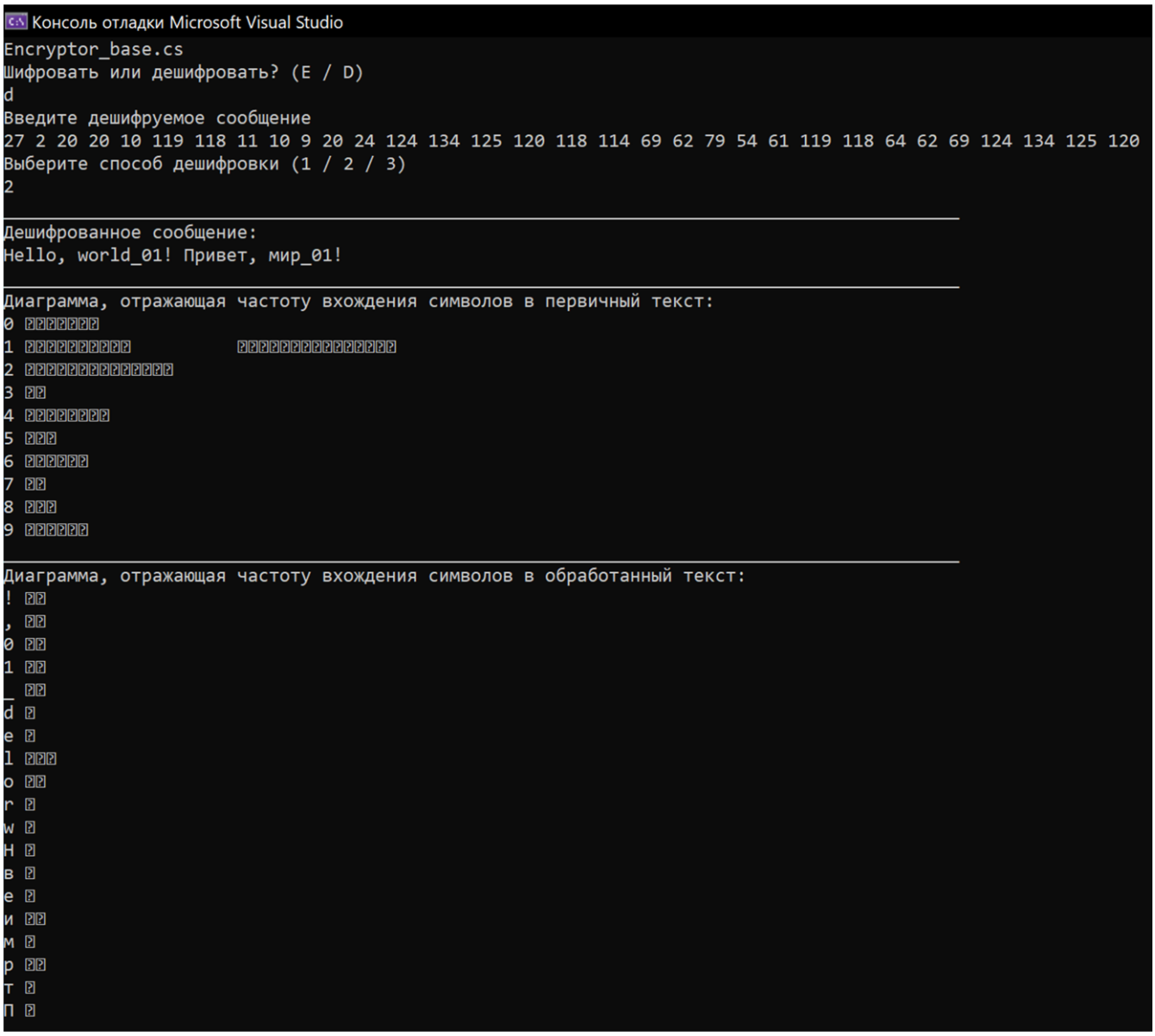


Рисунок 5 – дешифровка сообщения вторым способом

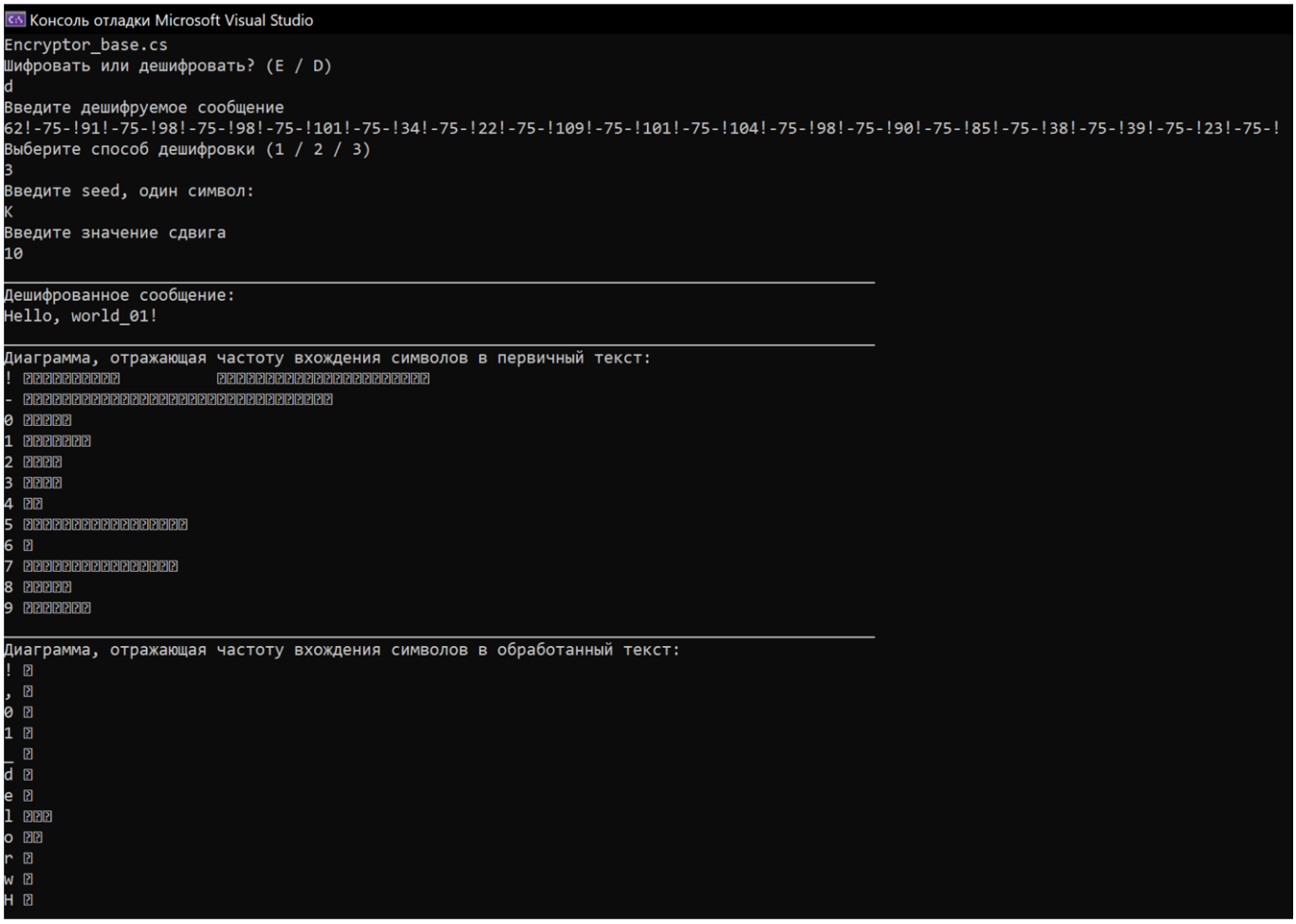


Рисунок 6 – дешифровка сообщения третьим способом