Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт кибербезопасности и цифровых технологий (ИКБ)
(наименование института, филиала)
Кафедра КБ-1 «Защита информации»
(наименование кафедры)

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ НА ПРОИЗВОДСТВЕННУЮ ПРАКТИКУ

(указать вид практики: учебная / производственная)

ПРАКТИКА ПО ПОЛУЧЕНИЮ ПРОФЕССИОНАЛЬНЫХ УМЕНИЙ И ОПЫТА ПРОФЕССИОНАЛЬНОЙ ДЕЯТЕЛЬНОСТИ

(указать тип практики в соответствии с учебным планом)

Студенту 3 курса учебной группы БАСО-03-20
Ускову Константину Александровичу
(фамилия, имя и отчество)

Место и время практики: Кафедра КБ-1 «Защита информации» 09.02.2023 – 31.05.2023
Должность на практике (при наличии): студент

1. СОДЕРЖАНИЕ ПРАКТИКИ:
1.1. Изучить: методы шифрования Виженера и Хилла
1.2. Практически выполнить: разработку и создание телеграмм бота для шифрования текстов
1.3. Ознакомиться: с программным интерфейсом приложения бота
2. ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ: нет
3. ОРГАНИЗАЦИОННО-МЕТОДИЧЕСКИЕ УКАЗАНИЯ: составить отчёт о проделанной работе в соответствии с ГОСТ

Руководитель практики от кафедры
«09» февраля 2023 г. _____ (Е.О. Карамышева)
Подпись *ФИО*

Задание получил:
«09» февраля 2023 г. _____ (К.А. Усков)
Подпись *ФИО*

СОГЛАСОВАНО:
И.о. заведующего кафедрой:
«09» февраля 2023 г. _____ (С.В. Артёмова)
Подпись *ФИО*

Проведенные инструктажи:

Охрана труда: «09» февраля 2023 г.

Инструктирующий		(Е.О. Карамышева)
	Подпись	*Расшифровка, должность*
Инструктируемый		(Усков К.А.)
	Подпись	*Расшифровка*

Техника безопасности: «09» февраля 2023 г.

Инструктирующий		(Е.О. Карамышева)
	Подпись	*Расшифровка, должность*
Инструктируемый		(К.А. Усков)
	Подпись	*Расшифровка*

Пожарная безопасность: «09» февраля 2023 г.

Инструктирующий		(Е.О. Карамышева)
	Подпись	*Расшифровка, должность*
Инструктируемый		(К.А. Усков)
	Подпись	*Расшифровка*

_____ «09» февраля 2023 г.

Инструктирующий		(Е.О. Карамышева)
	Подпись	*Расшифровка, должность*
Инструктируемый		(К.А. Усков)
	Подпись	*Расшифровка*

С правилами внутреннего распорядка ознакомлен: «09» февраля 2023 г.

	(К.А. Усков)
Подпись	*Расшифровка*

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт кибербезопасности и цифровых технологий (ИКБ)
(наименование института, филиала)
Кафедра КБ-1 «Защита информации»
(наименование кафедры)

ОТЧЁТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

(указать вид практики: учебная / производственная)

ПРАКТИКА ПО ПОЛУЧЕНИЮ ПРОФЕССИОНАЛЬНЫХ УМЕНИЙ И ОПЫТА ПРОФЕССИОНАЛЬНОЙ ДЕЯТЕЛЬНОСТИ

(указать тип практики в соответствии с учебным планом)

приказ Университета о направлении на практику от 19 января 2023 г. № 70-У

Отчет представлен к рассмотрению:
Студент группы *БАСО-03-20* «31» мая 2023 г. _____/К.А. Усков
(подпись и расшифровка подписи)

Отчет утвержден.
Допущен к защите:

Руководитель практики от кафедры «31» мая 2023 г. _____/ Е.О. Карамышева
(подпись и расшифровка подписи)

Москва 2023

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

СОВМЕСТНЫЙ РАБОЧИЙ ГРАФИК ПРОВЕДЕНИЯ ПРОИЗВОДСТВЕННОЙ ПРАКТИКИ

студента К.А. Ускова 3 курса группы БАСО-03-20 очной формы обучения, обучающегося по направлению подготовки 10.05.03 «Информационная безопасность автоматизированных систем», профиль «Создание автоматизированных систем в защищенном исполнении».

Неделя	Сроки выполнения	Этап	Отметка о выполнении
1-4	09.02.2023 - 09.03.2023	Получение индивидуального задания по производственной практике	выполнено
4-8	09.03.2023 – 09.04.2023	Исследование предметной области, подбор литературных источников по данной теме.	выполнено
8-12	09.04.2023 – 09.05.2023	Разработка и создание телеграм бота для шифрования текстов.	выполнено
12-16	09.05.2023- 31.05.2023	Оформление отчета и подготовка к защите	выполнено

Руководитель практики от кафедры _____ /Е.О. Карамышева, старший преподаватель/

Обучающийся _____ /К.А. Усков/

Согласовано:

И.о. заведующего кафедрой: _____ /С.В. Артёмова, д.т.н./

Содержание

ВВЕДЕНИЕ

Одним из способов передачи информации по открытому каналу связи в тайне от других является её передача в преобразованном виде т.е. использование алгоритмов шифрования.

Целью этой работы будет создания телеграмм бота способного шифровать и расшифровывать сообщения введённые пользователем по указанному ключу. Будут рассмотрены и реализованы два алгоритма шифрования с закрытым ключом – таблица Виженера и шифр Хилла.

1 Описание алгоритмов шифрования

Описание алгоритмов будет согласно их программной реализации т.е. будут присутствовать небольшие изменения упрощающие автоматическое шифрование с использование телеграмм бота. Однако суть алгоритмов останется прежней.

1.1 Шифр Виженера

Алгоритм шифрования относится к симметричным и представляет собой алгоритм многоалфавитной замены. Для шифрования используется таблица. Строки и столбцы которой обозначены символами алфавита. В простом случае первая строка задает одно алфавитную замену, а каждая последующая строка получается сдвигом верхней на одну позицию вправо. Задается ключ произвольной длины, если длина ключа меньше длины открытого текста, то он повторно записывается. Символ закрытого текста вычисляется как элемент таблицы со столбцом в виде символа открытого текста и строкой с символом ключевой информации, которая имеет ту же длину. Расшифровка соответственно заключается в определении столбца по строке и элементу.

При программной реализации для удобства символам будет задано соответствие целым неотрицательным числа, а именно как индекс вхождение символа типа char в список символов, который будет представлен в листинге программы. В качестве символов будут использоваться те что можно ввести с клавиатуры компьютера и телефона, за исключением символов эмоджи. Ключ и открытый текст записывается последовательностью чисел. Закрытый текст A' вычисляется по формуле (1.1) которая представляет собой алгоритм замены.

$$A'_i = f(A_i) + K_j (mod\ n), \tag{1.1}$$

где A'i– символ закрытого текста с номером i,Ai – символ открытого текста с номером i, Kj–символ ключевой информации с номеров j, j = i (modk), k–длина ключа, n–мощность алфавита, f–функция замены которая задаёт первую строку таблицы Виженера. В алгоритме она реализована, как инверсное шифрование по блокам в 10 символов. Т.е. будут блоки с номерами символов от 0 до 9; от 10 до 19; от 20 до 29 и т.д. Причём в последнем блоке число символом может быть меньше 10 в силу того, что мощность алфавита не кратна 10. В каждом блоке символ заменяется на противоположный, противоположность заключается в том, что расстояние от одного конца будет равно расстоянию до другого конца: номер 3, заменяется на 9-3=6; номер 16 заменяется на 29-(16-10) = 23; номер 71 заменяется на 79-(71%10)=78.

Если не сдвигать первую строку, а задавать по другому правилу, то функция замены f будет зависеть не только от открытого текста, но и от ключа.

Расшифровка аналогична процессу шифрования за исключением того, что используется функция обратная f.

1.2 Шифр Хилла

Алгоритм шифрования построен на использовании закрытого ключа по которому можно зашифровывать и расшифровывать сообщения. И использует принцип замены по блокам. Поскольку в алгоритме используются матричные операции, то необходимо дать однозначное соответствие алфавита и последовательности целых неотрицательных чисел: 0, 1, 2, …, n (n–мощность алфавита). Соответствие будет задано, как и в шифре Виженера.

Составляется ключ длинной k^2, где k–натуральное число отличное от 1. Причём чтобы исключить слишком длительные вычисления, связанный с большим порядком матрицы, число k будет не больше 7. В качестве ключа может использоваться ключевое слово или фраза длинной отличной от k^2, но тогда нужно будут добавить символы из алфавита, чтобы получить длину k^2. Ключевая информация заполняет квадратную матрицу K порядка k. Алго-

ритм заполнения матрицы представляет собой заполнение змейкой т.е. первая строка заполняется слева направо, вторая – слева направо, третья – слева направо и т.д. Исходная информация делится на блоки длинной k символов. Если открытый текст не кратен k, то необходимо дописать символы. В программе будут дописываться пробелы. Закрытый текст I' получается по формуле (1.2).

$$I' = I \times K(mod n), \tag{1.2}$$

где I' и Iматрицы размерности [1xk] соответствующие закрытому и открытому текстам, k–длина ключа, K–матрица размерности [kxk] составленная по ключу, n – мощность алфавита. Т.е. для шифрования необходимо последовательно перемножить информационные блоки и записать в той же последовательности. Как отмечалось ранее перед умножение необходимо дать соответствие алфавита и целых неотрицательных чисел. Процесс расшифровки аналогичен процессу шифрования за исключением того, что вместо матрицы K используется K' –матрица обратная матрице K по модулю n.

2 Создание телеграмм бота

Для создания бота необходимо воспользоваться телеграмм ботом BotFather, процесс создания показан на рисунке 2.1.

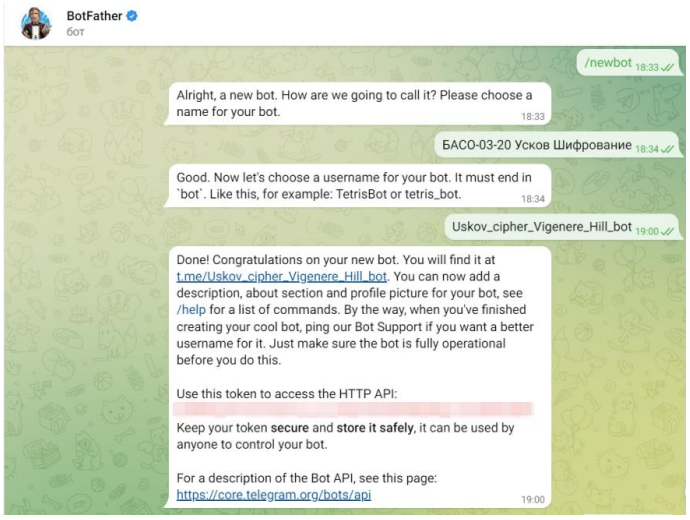

Рисунок 2.1 – Получение API для работы с ботом

Для настройки интерфейса бота используется тот же BotFather. Был настроен текст при первом запуске бота, который описывает что бот может делать. Было составлено описание бота в его профиле, добавлена иконка бота и три команды: /cipher_vigenere; /cipher_hill; /help. По умолчанию так же присутствует команда /start.

На рисунке 2.2 показан бот при его первом знакомстве.

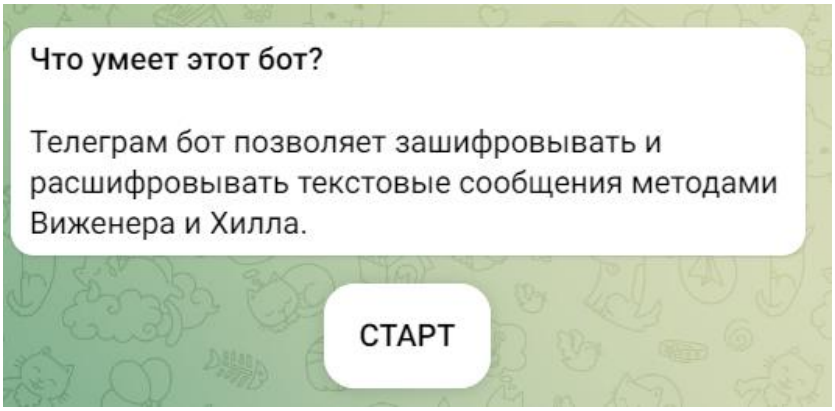

Рисунок 2.2 – Приветствие бота

На рисунке 2.3 показано описание бота в его профиле вместе с его иконкой.

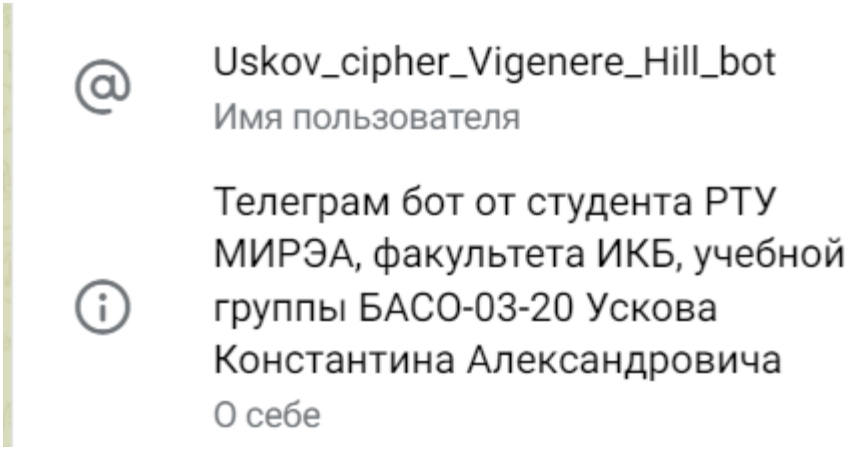

Рисунок 2.3 – Описание бота в профиле

На рисунке 2.4 показано меню для выбора команд.

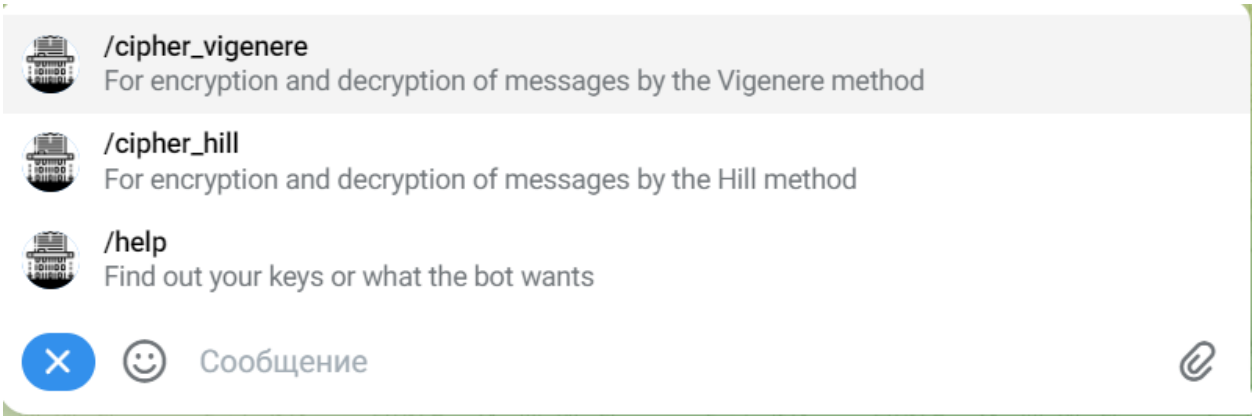

Рисунок 2.4 – Команды бота

Команда /start приветствует пользователя по его имени и выводит две кнопки для выбора алгоритма шифрования.

Команда /cipher_vigenere сообщает о том, что было выбрано шифрование методов Виженера и выводит 3 кнопки для ввода ключа, для шифрования и расшифровывания. Причём последние две операции не доступны, если не был введён ключ.

Команда /cipher_hill сообщает о том, что было выбрано шифрование методов Хилла и выводит 3 кнопки для ввода ключа, для шифрования и расшифровывания. Причём последние две операции не доступны, если не был введён ключ.

Команда /help сообщает пользователю о том, в каком режиме сейчас находится бот т.е. какое действие ожидает бот. Так же эта команда выводит две кнопки, что пользователь мог узнать какие ключи он ввёл для методом Виженера и Хилла.

3 Программная реализация бота

3.1 Общие сведения

Разрабатывать телеграмм бота будем в интегрированной среде разработки VisualStudio 2022CommunityEdition. В качестве проекта выберем консольное приложение на языке C# с платформой .NET 6.0.

Нужно подключить пакеты для работы с телеграмм ботом, а так же пакеты работы с SQLite для хранения информации о пользователях. Подключенные пакеты представлены на рисунке 3.

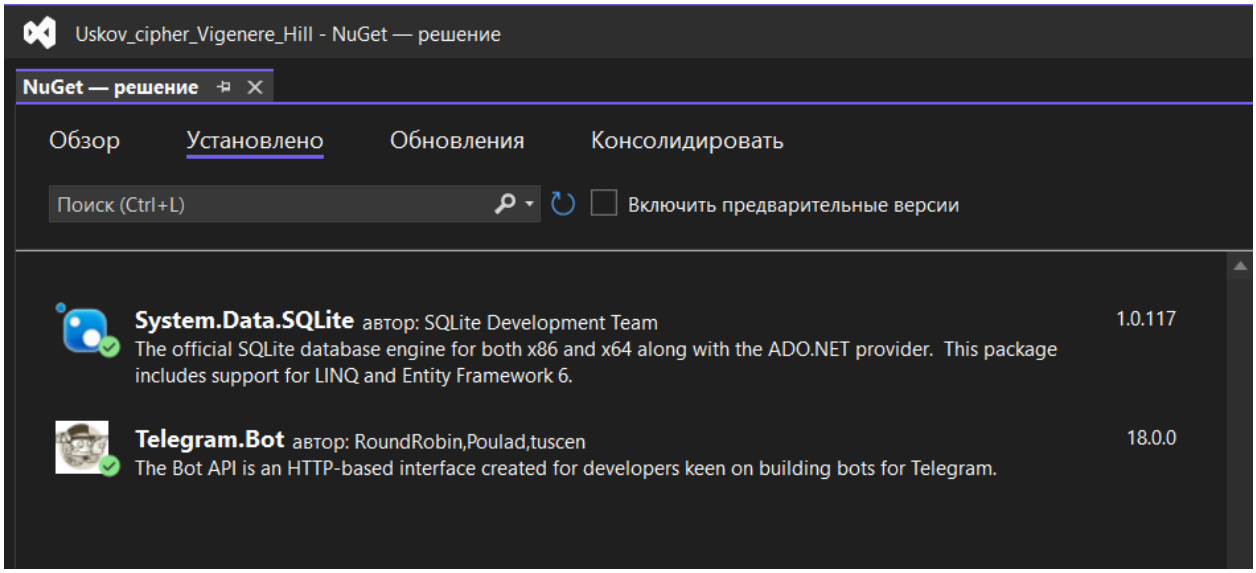

Рисунок 3 – Установленные пакеты NuGet

3.2 Классы и их описания

3.2.1 Класс «SQLite»

Класс «SQLite» предназначен для работы со встроенной БД SQLite, он содержит методы характеризующую запись и извлечение необходимых данных из БД.

Рассмотрим каждый метод по отдельности:

1) Метод «public static void Write(object obj)» необходим для записи данных в БД. Параметр obj служит для возможности использования метода по таймеру. Соответственно метод будет запускаться каждые 10 минут. Данные для записи берутся из статического поля класса Program.

2) Метод «public static void Read()» необходим для чтения данных из БД в процессе запуска телеграмм бота. Данные записываются в статическое поле класса Program.

3.2.2 Класс «HelpFunction»

Класс «HelpFunction» содержит в себе вспомогательные «функции». В нём реализованы методы для работы с двумерными массивами (логически – матрицами): умножение матриц, вычисление определителя, вычисление обратной матрицы по модулю, вычисление алгебраических дополнений.

Класс содержит поле «public static List<char> alphavit» которое служит в качестве алфавита и используется для преобразования символов в код.

Остановимся на каждом методе класса:

1) Метод «public static bool CheckErrorSymbol(string text)» возвращает истину, если в переменной text присутствует хотя бы один символ, которого нет в алфавите. И возвращает ложь, если все символы в тексте присутствуют в алфавите.

2) Метод «public static int ConvertSymbolToCode(char ch)» преобразует символ ch в код и возвращает это значение. Это необходимо для выполнения матричных операций в шифре Хилла, а так же в шифре Виженера – чтобы упростить процесс шифрования и расшифровки.

3) Метод «public static string SetDataToSQL(string text)» преобразует весь текст из переменной text в коды, причем длина каждого кода составляет 3 символа таким образом процесс декодирования однозначен. Этот метод необходим для записи симовлов в БД, поскольку она не обязательно поддерживает все символы из алфавита, к примеру «•™».

4) Метод «public static string ConvertCodeToSymbol(int code)» преобразует число (код) code в символ, для удобства символ представляется в виде строки, это значение и возвращается.

5) Метод «public static string GetDataFromSql(string text)» преобразует строку text состоящую из кодов в строку символов из алфавита. Т.е. является функцией декодирования, необходимой для отображения информации в кореектной форме, которая была получена из БД.

6) Метод «static int[,] DelRowCol(int[,] m, int _i, int _j)» удаляет из матрицы m, в виде двумерного массива строку с _i и столбец _j. И возвращает соответственно новую матрицу. Метод необходим для вычисления минора.

7) Метод «static decimal GetAlgDop(int[,] m, int _i, int _j)» возвращает алгебраическое дополнение указанного элемента (_i, _j) из матрицы m.

8) Метод «static decimal[] GetDetMatrix(int[,] m)» необходим для вычисления определите матрицы m. Причём для борьбы с переполнением он возвращает массив чисел, которые необходимо сложить, чтобы получить окончательно детерминант.

9) Метод «public static int GetOstat(decimal delim, int delit)» необходим для вычисления остатка от числа при делении числа delim на число delit, при условии того, что остаток является неотрицательным числом. Возвращает метод соответственно остаток от деления. Поскольку в языке программирования допускаются отрицательные остатки от деления, тогда как в алгоритмах шифрования остаток неотрицателен.

10) Метод «public static int GetOstat(int delim, int delit)» является перегрузкой предыдущего, его отличие в том, что делимое число представлено другим форматам, оно поддерживает меньшее число разрядом (для шифра Виженера).

11) Метод «public static int[,] MultiplyMatrixMod(int[,] leftMatrix, int[,] rightMatrix, int mod)» необходим для перемножения матриц leftMatrix и rightMatrix т.е. непосредственно для шифрования и расшифровывания информации по методу Хилла.

12) Метод «static int GetRevDet(int[,] m, int mod)» возвращает обратный элемент в кольце mod для определителя матрицы m .

13) Метод «public static int[,] GetRevMatrix(int[,] m, int mod)» возвращает обратную матрицу по модулю mod для матрицы m.

14) Метод «public static bool CheckDetMatrix(int[,] matrix, int mod)» возвращает истину, если не существует обратного элемента в кольце mod для определителя матрицы matrix. И значение ложь, если такой элемент существует.

15) Метод «public static int GetSqrt(int x)» необходим для вычисления квадратного корня из числа x. Причём если корень не является целым, то возвращается число равное сумме 10 и полученного корня с округление вверх. В коде корень вычисляется для проверки длинны ключа в шифровании методом Хилла, где заложено ограничение на максимальную длину в 49 символов, поэтому корень не может быть больше 7 и возвращаемая сумма в случае «ошибки» может быть одназначна распознана.

3.2.3 Класс «CipherVigenere»

Класс «CipherVigenere» необходим для расшифровывания и шифрования сообщений методом Виженера. Рассмотрим методы по отдельности:

1) Метод «public static string SetKey(long ID, string key)» принимает индетификатор пользователя ID и ключ key. И записывает этот ключ в поле соответствующего пользователя. Возвращает строку, которая является ответом пользователю.

2) Метод «static int GetCodeFirstRow(char ch)» принимает символ ch и возвращает код нового символа по алгоритму замены, описанному в разделе 1.1.

3) Метод «static string GetSymbolFirstRow(int number)» принимает номер символа и возвращает новый символ, полученный в результате обратного применения алгоритма замены.

4) Метод «public static string EncText(long ID, string text)» принимает индитификатор пользователя ID и строку text, которую необходимо зашифровать. С помощью ID определяется ключ пользователя и по нему происходит шифрование. Возвращает зашифрованное сообщение.

5) Метод «public static string DecText(long ID, string text)» принимает индитификатор пользователя ID и строку text, которую необходимо расшифровать. С помощью ID определяется ключ пользователя и по нему происходит расшифровка сообщения, результат расшифровки возвращается.

3.2.4 Класс «CipherHill»

Класс «CipherHill» необходим для расшифровывания и шифрования сообщений методом Виженера. Его методы:

1) Метод «public static string SetKey(long ID, string key)» принимает индетификатор пользователя и строку key, которая представляет собой ключ. Возвращает ответ пользователю об установки ключа. Как было описано в разделе 1.2. ключ может не быть степенью натурального числа, в таком случае к нему добавляются символы. А если ключ больше 49 символов, то в качестве ключа используются первые 49 символов. Так же возможно для ключа не сузествует обратной матрицы по модулю и в таком случае процесс расшифровки не получиться произвести, поэтому вводимый ключ будет последовательно сдвигаться на один символ k-1 раз, где k – длина ключа. Таким образом пользователю не придётся лишний раз возиться с подбором подходящего ключа.

Таким образом ключ вводимый пользователем и принятый программой могут отличаться, поэтому принятый ключ будет выведен пользователю. Хотя на самом деле если ввести повторно ключ пользователя, а не принятый программой, то программа примет тот же новый ключ.

2) Метод «public static string EncText(long ID, string text)» принимает индитификатор пользователя ID и строку text, которую необходимо зашиф-

ровать. С помощью ID определяется ключ пользователя и по нему происходит шифрование. Возвращает зашифрованное сообщение.

3) Метод «public static string DecText(long ID, string text)» принимает индитификатор пользователя ID и строку text, которую необходимо расшифровать. С помощью ID определяется ключ пользователя и по нему происходит расшифровка сообщения, результат расшифровки возвращается.

3.2.5 Класс «User_»

Класс «User_» описывает экземпляр пользователя со следующими полями:

1) Поле «public long ID» является индетификатором пользователя, который берется при получении обновлений в телеграм боте т.е. ответственность за уникальность значений лежит на мессенджере.

2) Поле «public int mode» характеризует режим работы бота. 0 – бот ожидает выбора своего функционала (нажатие на кнопку или ввода команды. 1 – бот ожидает ввод ключа для метода Виженера; 2 – бот ожидает ввод ключа для метода Хилла; 3 – бот ожидает ввод текста, который необходимо зашифровать по методу Виженера; 4 – бот ожидает ввод текста, который необходимо расшифровать по методу Виженера; 5 – бот ожидает ввод текста, который необходимо зашифровать по методу Хилла; 4 – бот ожидает ввод текста, который необходимо расшифровать по методу Хилла.

3) Поле «public string keyVigenere» необходимо для хранения ключа метода Виженера.

4) Поле «public string keyHill» необходимо для хранения ключа метода Хилла.

Класс также содержит несколько методов:

1) Метод «public static void RegisterUser(long ID)» добавляет с статическое поле Users класса Program нового пользователя с индетификаторо ID, режимом бота 0 и пустыми ключами.

2) Метод «public static int GetIndex(long ID)» возвращает индекс пользователя с индетификатором ID в списке статического поля Users класса Program, если пользователя нет в списке, то он автоматически добавляется и возвращается его индекс.

3.2.6 Класс «Program»

Класс «Program» является ключевым, он содержит поле «public static List<User_> Users» которое представляет собой совокупность всех пользователей, с которыми работает бот.

Рассмотрим методы класса:

1) Метод «static void Main(string[] args)» является входной точной при запуске бота. Он запускает метод для считывания данных из БД пользователей, создает бота-клиента по уникально токену, который был получен в разделе 2. Затем запускается обработка запросов для бота, создается и запускается таймер для записи данных пользователей в БД. На последнем шаге ожидается ввод строки из консоли, чтобы программа не завершалась.

2) Метод «async static Task Update(ITelegramBotClient botClient, Update update, CancellationToken token)» обрабатывает запросы от пользователей к телеграм боту. Пользователей может либо отправить сообщение (фото, текст, файл, команду), либо нажать на кнопку, которая не выводит сообщение в чате. Соответственно обработка запросов построена на эти двух параметрах.

Если пользователь отправил сообщение, то сначала проверяется есть ли в тексте те символы, которых нету в алфавите. Затем сообщение проверяется на предмет наличия команды, если команда не была введена, то проверяется режим бота, после чего сообщение обрабатывается как ключ, открытый текст или как закрытый текст для двух методом шифрования. Если режим бота 0 и оно не содержит команду, то сообщение «не обрабатывается», а именно бот сообщает, что не понял пользователя.

Если пользователь не отправлял сообщение, то запрос должен характеризовать нажатие какой либо кнопки бота. По нажатой кнопке сменяется ре-

жим работы бота и выводится сообщение пользователю о том, что бот в дальнейшем ожидает.

2) Метод «static Task Error(ITelegramBotClient botClient, Exception exception, CancellationToken cancellationToken)» предназначен для обработки ошибок, в частностности полученные из-за некорректного составления алгоритма (например обращение к массиву за его пределы). Данный метод не реализован.

4 Пример работы с ботом

После запуска консольного приложения, программа по указанному API обрабатывает все запросы, которые не были обработаны. Проверим функционал бота введя различные команды. После первого знакомства, нажав на кнопку «Старт» автоматически отправляется команда /start. Ответ бота показан на рисунке 4.1.

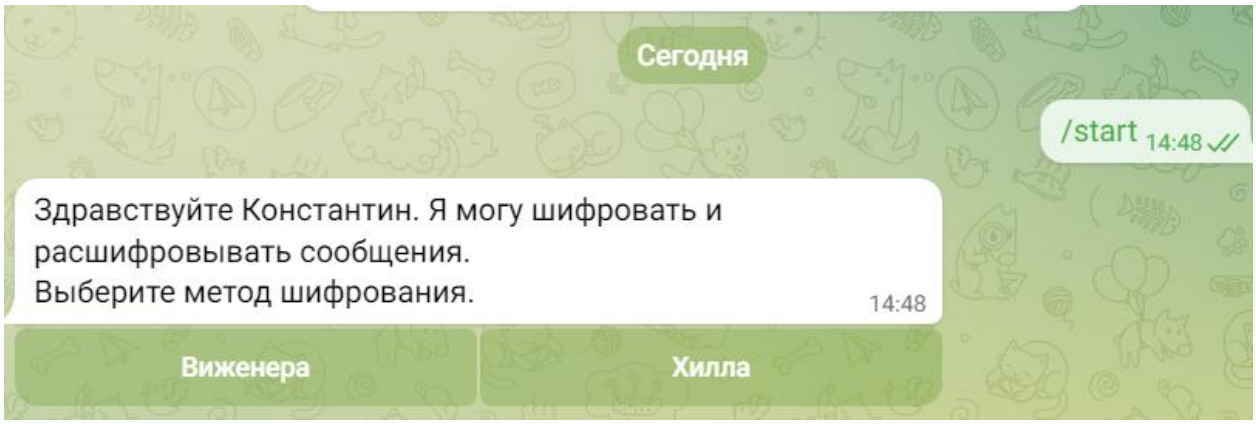

Рисунок 4.1 – Ответ бота на команду /start

Использовав меню команд, введём команду /help, результат ответа бота на эту команду и попытка просмотра ключей показана на рисунке 4.2.

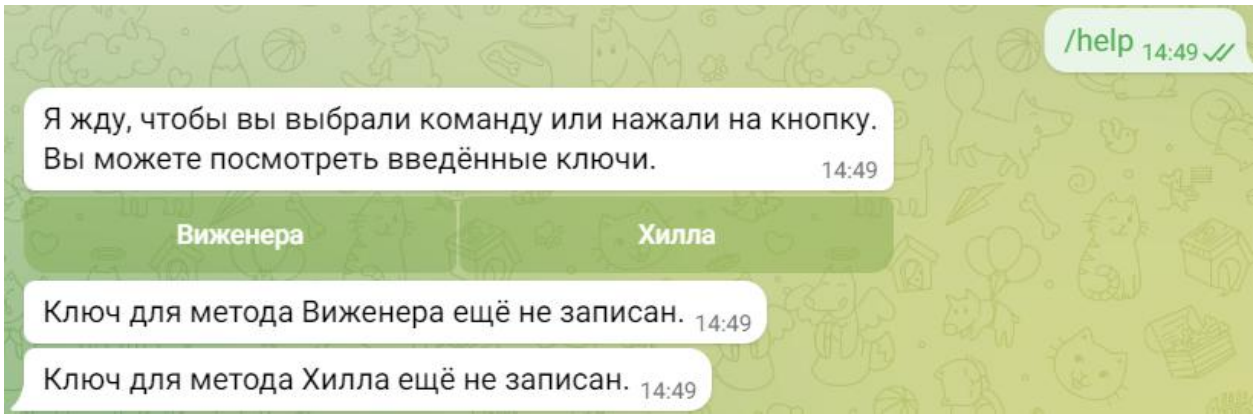

Рисунок 4.2 – Ответ бота на команду /help

На рисунке 4.3 показан ввод команды для метода Виженера, а также попытка зашифровать информацию, которая провалилась в силу отсутствия ключа и непосредственно ввод ключа с ответами бота.

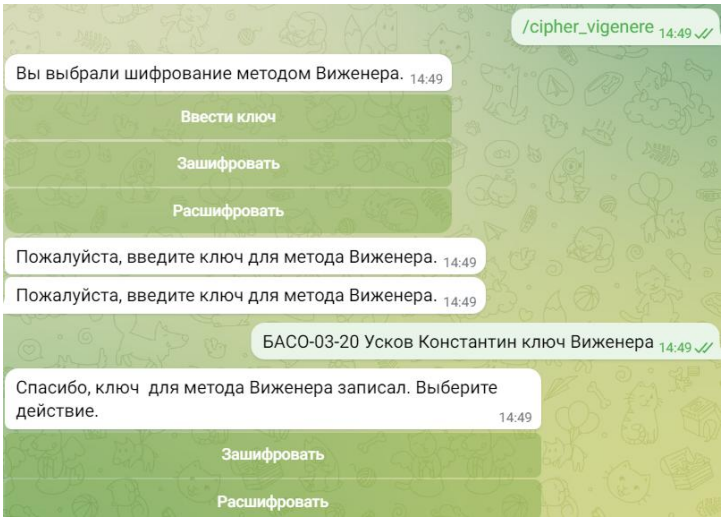

Рисунок 4.3 – Ввод ключа для метода Виженера

На рисунке 4.4 показано две попытки зашифровки сообщения. Первая провалилась, поскольку в алфавите нет кавычек такого формата, который использовался в сообщении.

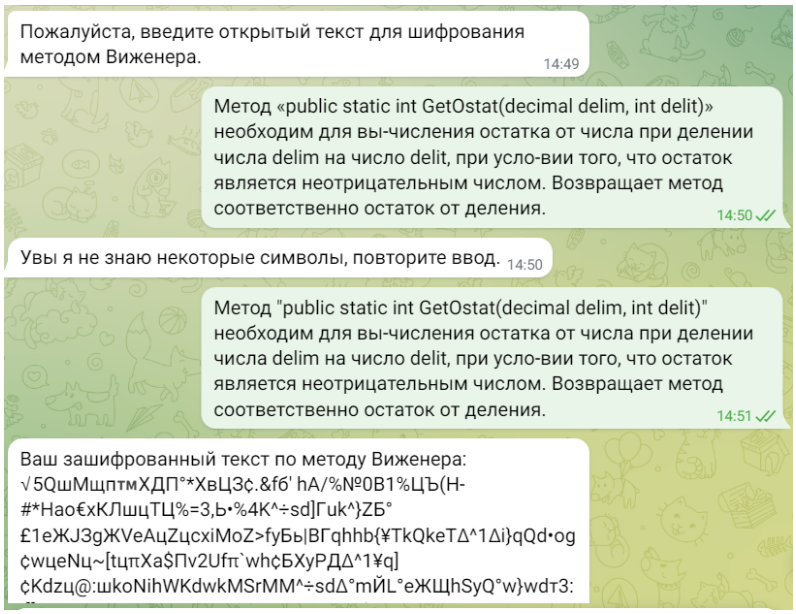

Рисунок 4.4 – Шифрование методом Виженера

На рисунке 4.5 показа расшифровка ранее зашифрованного сообщения.

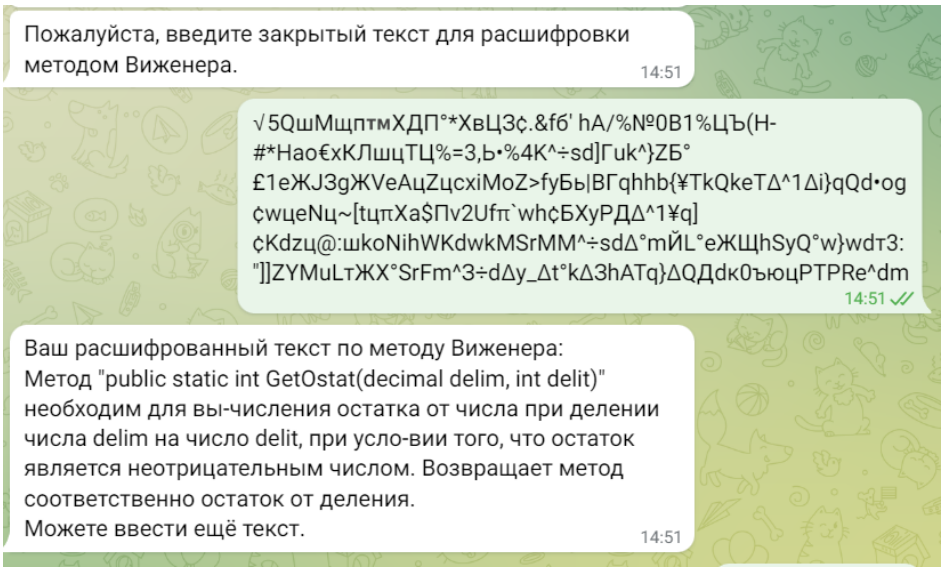

Рисунок 4.5 – Расшифровывание методом Виженера

На рисунке 4.5 показан ввод команды для шифрование методом Хилла, а также ввод ключа со всеми ответами бота.

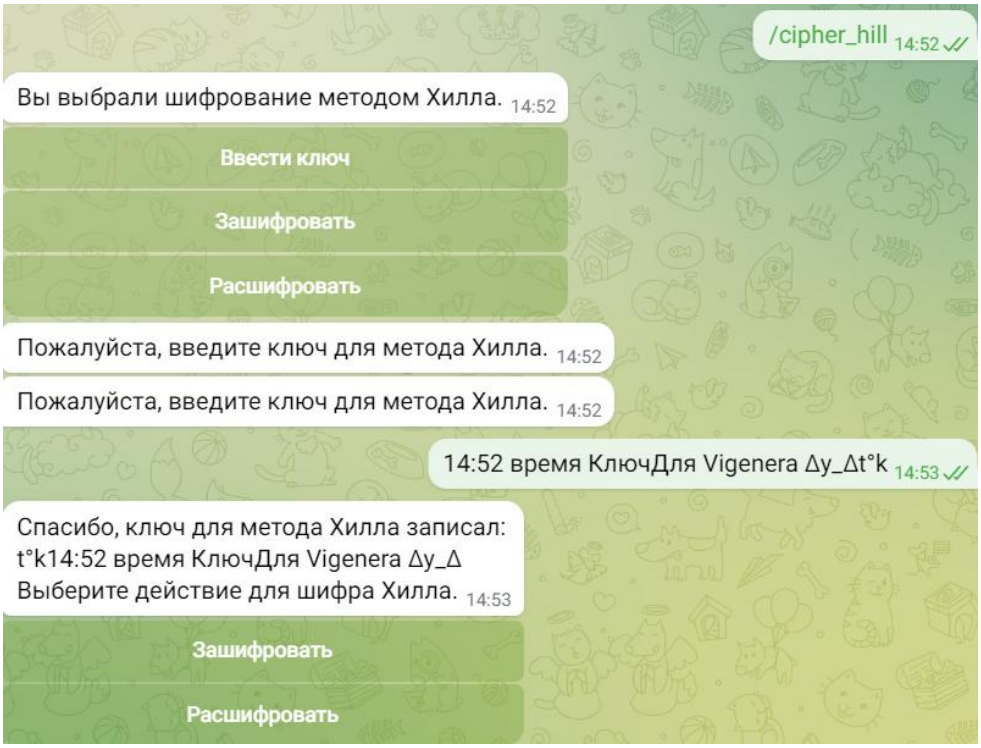

Рисунок 4.5 – Ввод ключа для метода Хилла

На рисунке 4.6 показано шифрование текста по методу Хилла.

Рисунок 4.6 – Шифрование методом Хилла

На рисунке 4.7 показа расшифровка ранее зашифрованного сообщения.

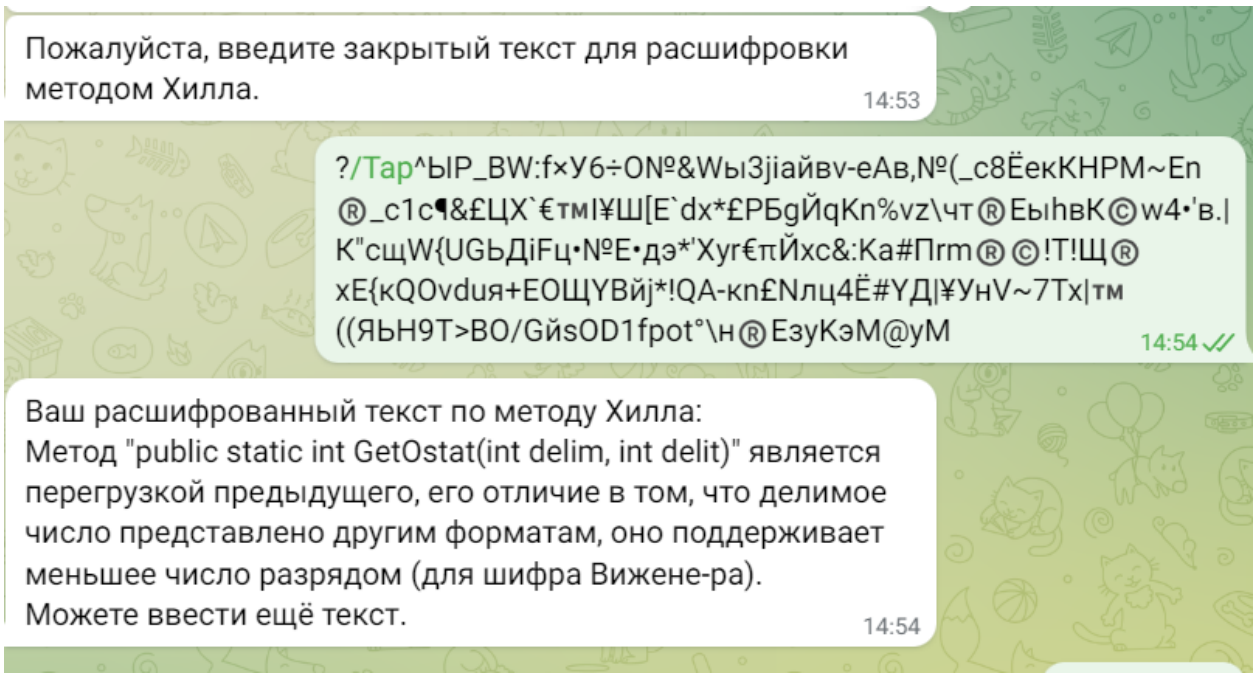

Рисунок 4.7 – Расшифровка сообщения методом Хилла

На рисунке 4.8 показан ввод команды /help и просмотр введённых ключей для методов Виженера и Хилла.

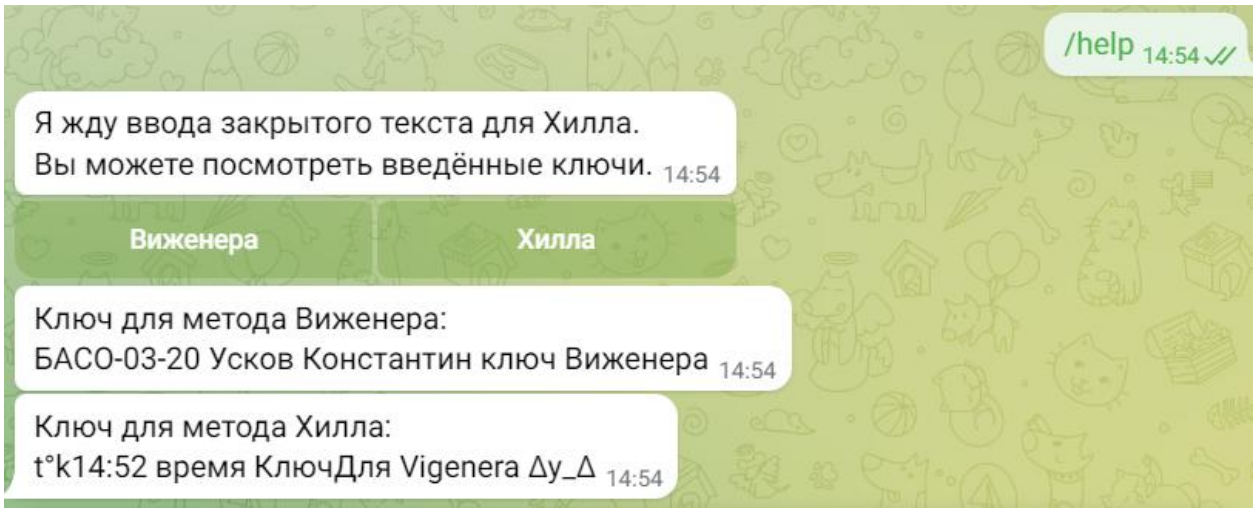

Рисунок 4.8 – Просмотр введённых ключей

ЗАКЛЮЧЕНИЕ

В процессе прохождения производственной практики были изучены методы шифрования Хилла и Виженера. На основании полученных знаний были реализованы программно шифры Виженера и Хилла с использованием языка программирования C#. После ознакомления с программных интерфейсом приложения бота (API) от telegram, было написано приложение, поддерживающее работу с API ключами для обработки обращений к созданному боту в мессенджере telegram.

Созданный бот поддерживает шифрование и расшифровывание текстовых сообщений методами Виженера и Хилла. Любой запрос к разработанному приложению возвращает ответ. Разработанный бот исключает такие ошибки, как неправильный ввод сообщения т.е. ввод некорректных символов или отправка фотографий и файлов вместо текста, а также попытку шифрования или расшифровывания сообщения без ввода ключа. Telegram бот допускает просмотр введённых ключей и раз в 10 мин сохраняет о них информацию, на случай если приложение будет остановлено. Для удобства понимания структуры бота, пользователей имеет возможность узнать в каком состоянии находится приложения т.е. запросить у бота чего он ожидает от пользователя, таким образом исключается недопонимание в случае, если пользователей давно не пользовался ботом.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Прайс М. C# 10 и .NET 6. Современная кросс-платформенная разработка, 6-е изд. – СПб.: Питер, 2023. – 848 с.

2. Ермакова А.Ю. Криптографические методы защиты информации [Электронный ресурс]: Учебно-методическое пособие / Ермакова А.Ю. — М.: МИРЭА – Российский технологический университет, 2021. — 1 электрон. опт. диск (CD-ROM).

3. Керниган Б., Пайк Р. Практика программирования – М.:Вильямс, 2019. – 288 с.

4. Introduction - A guide to Telegram.Bot library: [Электронный ресурс]. URL: https://telegrambots.github.io/book (Дата обращения: 06.04.2023).

5. Query Language Understood by SQLite: [Электронный ресурс]. URL: https://www.sqlite.org/lang.html (Дата обращения: 08.04.2023).

ПРИЛОЖЕНИЕ

Листинг программы.

1.1. Файл «Program.cs»

```
// See https://aka.ms/new-console-template for more information

using Telegram.Bot;
using Telegram.Bot.Types;
using Telegram.Bot.Types.ReplyMarkups;
using Uskov_cipher_Vigenere_Hill;

class Program
{
    public static List<User_> Users = new List<User_>();
    static void Main(string[] args)
    {
        SQLite.Read();
        var client = new TelegramBotCli-
ent("5999627978:AAGxvyj1Lio62FtbH5rN6Z9Hjtsnd5ScHDM");
        client.StartReceiving(Update, Error);
        int num = 0;
        TimerCallback tm = new TimerCallback(SQLite.Write);
        System.Threading.Timer timer = new System.Threading.Timer(tm, num,
0, 600000);
        Console.ReadLine();
    }
    async static Task Update(ITelegramBotClient botClient, Update update,
CancellationToken token)
    {
        var message = update.Message;
        var answer = update.CallbackQuery;
        if (message != null)
        {
            if (message.Text != null)
            {
                if (HelpFunction.CheckErrorSymbol(message.Text)) { await
botClient.SendTextMessageAsync(message.Chat.Id, "Увы я не знаю некоторые
символы, повторите ввод."); }
                else if (message.Text == "/start")
                {
                    User_.GetIndex(message.Chat.Id);
                    InlineKeyboardMarkup inlineKeyboard = new(new[] {
                        InlineKeyboardButton.WithCallbackData(text: "Виже-
нера", callbackData: "cipherVigenere"),
                        InlineKeyboardButton.WithCallbackData(text:
"Хилла", callbackData: "cipherHill") });
                    await botClient.SendTextMessageAsync(
                        chatId: message.Chat.Id,
                        text: $"Здравствуйте {message.Chat.FirstName ??
"человек без имени"}. Я могу шифровать и расшифровывать сообще-
ния.\nВыберите метод шифрования.",
                        replyMarkup: inlineKeyboard);
                }
                else if (message.Text == "/cipher_vigenere")
                {
                    InlineKeyboardMarkup inlineKeyboard = new(new[] {
```

```
                        new []{ InlineKeyboardBut-
ton.WithCallbackData(text: "Ввести ключ", callbackData: "setKeyVigenere")
},
                        new []{ InlineKeyboardBut-
ton.WithCallbackData(text: "Зашифровать", callbackData: "encVigenere") },
                        new []{ InlineKeyboardBut-
ton.WithCallbackData(text: "Расшифровать", callbackData: "decVigenere") }
});
                    await botClient.SendTextMessageAsync(
                        chatId: message.Chat.Id,
                        text: "Вы выбрали шифрование методом Виженера.",
                        replyMarkup: inlineKeyboard);
                }
                else if (message.Text == "/cipher_hill")
                {
                    InlineKeyboardMarkup inlineKeyboard = new(new[] {
                       new []{ InlineKeyboardButton.WithCallbackData(text:
"Ввести ключ", callbackData: "setKeyHill") },
                        new []{ InlineKeyboardButton.WithCallbackData(text:
"Зашифровать", callbackData: "encHill") },
                        new []{ InlineKeyboardButton.WithCallbackData(text:
"Расшифровать", callbackData: "decHill") } });
                    await botClient.SendTextMessageAsync(
                        chatId: message.Chat.Id,
                        text: "Вы выбрали шифрование методом Хилла.",
                        replyMarkup: inlineKeyboard);
                }
                else if (message.Text == "/help")
                {
                    string str = "";
                    switch (Users[User_.GetIndex(message.Chat.Id)].mode)
                    {
                        case 0:
                            str = "Я жду, чтобы вы выбрали команду или
нажали на кнопку.";
                            break;
                        case 1:
                            str = "Я жду ввода ключа для метода Вижене-
ра.";
                            break;
                        case 2:
                            str = "Я жду ввода ключа для метода Хилла.";
                            break;
                        case 3:
                            str = "Я жду ввода открытого текста для Виже-
нера.";
                            break;
                        case 4:
                            str = "Я жду ввода закрытого текста для Виже-
нера.";
                            break;
                        case 5:
                            str = "Я жду ввода открытого текста для Хил-
ла.";
                            break;
                        case 6:
                            str = "Я жду ввода закрытого текста для Хил-
ла.";
                            break;
                    }
                    InlineKeyboardMarkup inlineKeyboard = new(new[] {
```

```
                        InlineKeyboardButton.WithCallbackData(text: "Виже-
нера", callbackData: "getKeyVigenere"),
                        InlineKeyboardButton.WithCallbackData(text:
"Хилла", callbackData: "getKeyHill")});
                    await botClient.SendTextMessageAsync(
                        chatId: message.Chat.Id,
                        text: str + "\nВы можете посмотреть введённые клю-
чи.",
                        replyMarkup: inlineKeyboard);
                }
                else
                {
                    var mode = Us-
ers[User_.GetIndex(message.Chat.Id)].mode;
                    if (mode == 0) { await botCli-
ent.SendTextMessageAsync(message.Chat.Id, "Простите, но я не знаю что де-
лать, выберите команду или нажми на кнопки."); }
                    else if (mode == 1)
                    {
                        Users[User_.GetIndex(message.Chat.Id)].keyVigenere
= message.Text;
                        InlineKeyboardMarkup inlineKeyboard = new(new[] {
                        new []{ InlineKeyboardBut-
ton.WithCallbackData(text: "Зашифровать", callbackData: "encVigenere") },
                        new []{ InlineKeyboardBut-
ton.WithCallbackData(text: "Расшифровать", callbackData: "decVigenere") }
});
                        await botClient.SendTextMessageAsync(
                            chatId: message.Chat.Id,
                            text: "Спасибо, ключ  для метода Виженера за-
писал. Выберите действие.",
                            replyMarkup: inlineKeyboard);

                    }
                    else if (mode == 2)
                    {
                        InlineKeyboardMarkup inlineKeyboard = new(new[] {
                       new []{ InlineKeyboardButton.WithCallbackData(text:
"Зашифровать", callbackData: "encHill") },
                        new []{ InlineKeyboardButton.WithCallbackData(text:
"Расшифровать", callbackData: "decHill") } });
                        string check = CipherHill.SetKey(message.Chat.Id,
message.Text);
                        if (!check.Contains("Увы"))
                        {
                            Users[User_.GetIndex(message.Chat.Id)].mode =
0;
                            check += Us-
ers[User_.GetIndex(message.Chat.Id)].keyHill + "\nВыберите действие для
шифра Хилла.";
                        }
                        await botClient.SendTextMessageAsync(
                            chatId: message.Chat.Id,
                            text: check,
                            replyMarkup: inlineKeyboard);
                    }
                    else if (mode == 3)
                    {
                        string encText = CipherVigene-
re.EncText(message.Chat.Id, message.Text);
```

```
                        { await botCli-
ent.SendTextMessageAsync(message.Chat.Id, $"Ваш зашифрованный текст по ме-
тоду Виженера:\n{encText}\nМожете ввести ещё текст."); }
                    }
                    else if (mode == 4)
                    {
                        string decText = CipherVigene-
re.DecText(message.Chat.Id, message.Text);
                        { await botCli-
ent.SendTextMessageAsync(message.Chat.Id, $"Ваш расшифрованный текст по
методу Виженера:\n{decText}\nМожете ввести ещё текст."); }
                    }
                    else if (mode == 5)
                    {
                        string encText = Ci-
pherHill.EncText(message.Chat.Id, message.Text);
                        { await botCli-
ent.SendTextMessageAsync(message.Chat.Id, $"Ваш зашифрованный текст по ме-
тоду Хилла:\n{encText}\nМожете ввести ещё текст."); }
                    }
                    else if (mode == 6)
                    {
                        string decText = Ci-
pherHill.DecText(message.Chat.Id, message.Text);
                        { await botCli-
ent.SendTextMessageAsync(message.Chat.Id, $"Ваш расшифрованный текст по
методу Хилла:\n{decText}\nМожете ввести ещё текст."); }
                    }
                    else { await botCli-
ent.SendTextMessageAsync(message.Chat.Id, "Похоже я сломался."); }
                }
            }
            else { await botClient.SendTextMessageAsync(message.Chat.Id,
"Прости, но я не знаю что делать, выбери команду или нажми на кнопки."); }
        }
        else if (answer != null)
        {
            if (answer.Data == "setKeyVigenere")
            {
                Users[User_.GetIndex(answer.Message.Chat.Id)].mode = 1;
                await botCli-
ent.SendTextMessageAsync(answer.Message.Chat.Id, "Пожалуйста, введите ключ
для метода Виженера.");
            }
            else if (answer.Data == "setKeyHill")
            {
                Users[User_.GetIndex(answer.Message.Chat.Id)].mode = 2;
                await botCli-
ent.SendTextMessageAsync(answer.Message.Chat.Id, "Пожалуйста, введите ключ
для метода Хилла.");
            }
            else if (answer.Data == "encVigenere")
            {
                var keyVigenere = Us-
ers[User_.GetIndex(Convert.ToInt32(answer.Message.Chat.Id))].keyVigenere;
                if (keyVigenere.Length == 0)
                {
                    Us-
ers[User_.GetIndex(Convert.ToInt32(answer.Message.Chat.Id))].mode = 1;
```

```
                    await botCli-
ent.SendTextMessageAsync(answer.Message.Chat.Id, "Пожалуйста, введите ключ
для метода Виженера.");
                }
                else
                {
                    Us-
ers[User_.GetIndex(Convert.ToInt32(answer.Message.Chat.Id))].mode = 3;
                    await botCli-
ent.SendTextMessageAsync(answer.Message.Chat.Id, "Пожалуйста, введите от-
крытый текст для шифрования методом Виженера.");
                }
            }
            else if (answer.Data == "decVigenere")
            {
                var keyVigenere = Us-
ers[User_.GetIndex(Convert.ToInt32(answer.Message.Chat.Id))].keyVigenere;
                if (keyVigenere.Length == 0)
                {
                    Us-
ers[User_.GetIndex(Convert.ToInt32(answer.Message.Chat.Id))].mode = 1;
                    await botCli-
ent.SendTextMessageAsync(answer.Message.Chat.Id, "Пожалуйста, введите ключ
для метода Виженера.");
                }
                else
                {
                    Us-
ers[User_.GetIndex(Convert.ToInt32(answer.Message.Chat.Id))].mode = 4;
                    await botCli-
ent.SendTextMessageAsync(answer.Message.Chat.Id, "Пожалуйста, введите за-
крытый текст для расшифровки методом Виженера.");
                }
            }
            else if (answer.Data == "encHill")
            {
                var keyVigenere = Us-
ers[User_.GetIndex(Convert.ToInt32(answer.Message.Chat.Id))].keyHill;
                if (keyVigenere.Length == 0)
                {
                    Us-
ers[User_.GetIndex(Convert.ToInt32(answer.Message.Chat.Id))].mode = 2;
                    await botCli-
ent.SendTextMessageAsync(answer.Message.Chat.Id, "Пожалуйста, введите ключ
для метода Хилла.");
                }
                else
                {
                    Us-
ers[User_.GetIndex(Convert.ToInt32(answer.Message.Chat.Id))].mode = 5;
                    await botCli-
ent.SendTextMessageAsync(answer.Message.Chat.Id, "Пожалуйста, введите от-
крытый текст для шифрования методов Хилла.");
                }
            }
            else if (answer.Data == "decHill")
            {
                var keyVigenere = Us-
ers[User_.GetIndex(Convert.ToInt32(answer.Message.Chat.Id))].keyHill;
                if (keyVigenere.Length == 0)
                {
```

```
                    Us-
ers[User_.GetIndex(Convert.ToInt32(answer.Message.Chat.Id))].mode = 2;
                    await botCli-
ent.SendTextMessageAsync(answer.Message.Chat.Id, "Пожалуйста, введите ключ
для метода Хилла.");
                }
                else
                {
                    Us-
ers[User_.GetIndex(Convert.ToInt32(answer.Message.Chat.Id))].mode = 6;
                    await botCli-
ent.SendTextMessageAsync(answer.Message.Chat.Id, "Пожалуйста, введите за-
крытый текст для расшифровки методом Хилла.");
                }
            }
            else if (answer.Data == "cipherVigenere")
            {
                InlineKeyboardMarkup inlineKeyboard = new(new[] {
                        new []{ InlineKeyboardBut-
ton.WithCallbackData(text: "Ввести ключ", callbackData: "setKeyVigenere")
},
                        new []{ InlineKeyboardBut-
ton.WithCallbackData(text: "Зашифровать", callbackData: "encVigenere") },
                        new []{ InlineKeyboardBut-
ton.WithCallbackData(text: "Расшифровать", callbackData: "decVigenere") }
});
                await botClient.SendTextMessageAsync(
                    chatId: answer.Message.Chat.Id,
                    text: "Вы выбрали шифрование методом Виженера.",
                    replyMarkup: inlineKeyboard);
            }
            else if (answer.Data == "cipherHill")
            {
                InlineKeyboardMarkup inlineKeyboard = new(new[] {
                        new []{ InlineKeyboardButton.WithCallbackData(text:
"Ввести ключ", callbackData: "setKeyHill") },
                        new []{ InlineKeyboardButton.WithCallbackData(text:
"Зашифровать", callbackData: "encHill") },
                        new []{ InlineKeyboardButton.WithCallbackData(text:
"Расшифровать", callbackData: "decHill") } });
                await botClient.SendTextMessageAsync(
                    chatId: answer.Message.Chat.Id,
                    text: "Вы выбрали шифрование методом Хилла.",
                    replyMarkup: inlineKeyboard);
            }
            else if (answer.Data == "getKeyVigenere")
            {
                string keyVig = Us-
ers[User_.GetIndex(Convert.ToInt32(answer.Message.Chat.Id))].keyVigenere;
                if (keyVig.Length == 0) { keyVig = "Ключ для метода Вижe-
нера ещё не записан."; }
                else { keyVig = $"Ключ для метода Виженера:\n" + keyVig; }
                await botCli-
ent.SendTextMessageAsync(answer.Message.Chat.Id, keyVig);
            }
            else if (answer.Data == "getKeyHill")
            {
                string keyHill = Us-
ers[User_.GetIndex(Convert.ToInt32(answer.Message.Chat.Id))].keyHill;
                if (keyHill.Length == 0) { keyHill = "Ключ для метода Хил-
ла ещё не записан."; }
```

```
                else { keyHill = $"Ключ для метода Хилла:\n" + keyHill; }
                await botCli-
ent.SendTextMessageAsync(answer.Message.Chat.Id, keyHill);
            }
            else { await botCli-
ent.SendTextMessageAsync(answer.Message.Chat.Id, "Похоже я сломался."); }
        }
        else { Console.WriteLine("Неизвестный запрос\n"); }
    }
    static Task Error(ITelegramBotClient botClient, Exception exception,
CancellationToken cancellationToken)
    {
        throw new NotSupportedException();
    }
}
```

1.2. Файл «SQLite.cs»

```
using System.Data.SQLite;

namespace Uskov_cipher_Vigenere_Hill
{
    internal class SQLite
    {
        public static void Write(object obj)
        {
            using (var connection = new SQLiteConnection(@"Data Source = "
+ Environment.CurrentDirectory + @"\dataForCipherTelegram.db; Ver-
sion=3;"))
            {
                connection.Open();
                var command = new SQLiteCommand();
                command.Connection = connection;
                command.CommandText = "DROP TABLE IF EXISTS allData;";
                command.ExecuteNonQuery();
                command.CommandText = "CREATE TABLE allData (ID INTEGER
PRIMARY KEY, mode INTEGER, keyVigenere TEXT, keyHill TEXT)";
                command.ExecuteNonQuery();
                foreach (var user in Program.Users)
                {
                    string keyVigenere = HelpFunc-
tion.SetDataToSQL(user.keyVigenere);
                    string keyHill = HelpFunc-
tion.SetDataToSQL(user.keyHill);
                    command.CommandText = $"INSERT INTO allData(ID, mode,
keyVigenere, keyHill) VALUES({user.ID}, {user.mode}, '{keyVigenere}',
'{keyHill}')";
                    command.ExecuteNonQuery();
                }
            }
        }
        public static void Read()
        {
            if (File.Exists(Environment.CurrentDirectory +
@"\dataForCipherTelegram.db"))
            {
                using (var connection = new SQLiteConnection(@"Data Source
= " + Environment.CurrentDirectory + @"\dataForCipherTelegram.db; Ver-
sion=3;"))
                {
                    connection.Open();
```

```
                    var command = new SQLiteCommand();
                    command.Connection = connection;
                    command.CommandText = $"SELECT * FROM allData";
                    using (var reader = command.ExecuteReader())
                    {
                        if (reader.HasRows)
                        {
                            while (reader.Read())
                            {
                                Program.Users.Add(new User_
                                {
                                    ID = Con-
vert.ToInt32(reader.GetValue(0)),
                                    mode = Con-
vert.ToInt32(reader.GetValue(1)),
                                    keyVigenere = HelpFunc-
tion.GetDataFromSql(reader.GetValue(2).ToString()),
                                    keyHill = HelpFunc-
tion.GetDataFromSql(reader.GetValue(3).ToString())
                                });
                            }
                        }
                    }
                }
            }
        }
    }
}
```

1.3. Файл «User_.cs»

```
namespace Uskov_cipher_Vigenere_Hill
{
    internal class User_
    {
        public long ID { get; set; }
        public int mode { get; set; }
        public string keyVigenere { get; set; }
        public string keyHill { get; set; }
        public static void RegisterUser(long ID)
        {
            Program.Users.Add(new User_
            {
                ID = ID,
                mode = 0,
                keyVigenere = "",
                keyHill = ""
            });
        }
        public static int GetIndex(long ID)
        {
            if (Program.Users.FindIndex(user => user.ID == ID) == -1)
            {
                RegisterUser(ID);
            }
            return Program.Users.FindIndex(user => user.ID == ID);
        }
    }
}
```

1.4. Файл «HelpFunction.cs»

```
namespace Uskov_cipher_Vigenere_Hill
{
    internal class HelpFunction
    {
        public static List<char> alphavit =
"ЁΔАБВГ✓ДЕЖ№ЗИЙК√ЛМНОПР!С\"Т•™#У$Ф%Х&Ц'Ч(Ш)Щ*Ъ+Ы,Ь-Э.Ю/Я0а1
б2в3г4д5е6ж7з8и9й:к;л<м=н>о?п@рАсВтСуДфЕхФцГчНшІщЈъКыЬьМэНюОяРQёRSTUVWXYZ[
\\]^_`abcdefghijklmnopqrstuvwxyz{|}~¢£¥©€®°¶π×÷".ToList();
        public static bool CheckErrorSymbol(string text)
        {
            foreach (char ch in text.ToCharArray())
            {
                if (alphavit.IndexOf(ch) == -1) { return true; }
            }
            return false;
        }
        public static int ConvertSymbolToCode(char ch)
        {
            return alphavit.IndexOf(ch);
        }
        public static string SetDataToSQL(string text)
        {
            string newText = "";
            foreach (char ch in text)
            {
                int number = ConvertSymbolToCode(ch);
                if (number < 10) { newText += $"00{number}"; }
                else if (number < 100) { newText += $"0{number}"; }
                else { newText += number.ToString(); }
            }
            return newText;
        }
        public static string ConvertCodeToSymbol(int code)
        {
            return alphavit[Convert.ToInt32(code)].ToString();
        }
        public static string GetDataFromSql(string text)
        {
            string newText = "";
            for (int i = 0; i < text.Length; i += 3)
            {
                newText += ConvertCodeToSym-
bol(Convert.ToInt32(text.Substring(i, 3)));
            }
            return newText;
        }
        static int[,] DelRowCol(int[,] m, int _i, int _j)
        {//удаляет строку и столбец для вычисления минора
            int k = m.Length;
            int[,] newM1 = new int[m.GetLength(0) - 1, m.GetLength(1)];
            int x = 0;
            for (int i = 0; i < m.GetLength(0); i++)
            {
                if (i == _i)
                {
                    _i = -1;
                    continue;
                }

                for (int j = 0; j < m.GetLength(1); j++)
                {
```

```
                    newM1[x, j] = m[i, j];
                }
                x++;
            }
            int[,] newM2 = new int[newM1.GetLength(0), newM1.GetLength(1)
- 1];
            for (int i = 0; i < newM1.GetLength(0); i++)
            {
                x = 0;
                for (int j = 0; j < newM1.GetLength(1); j++)
                {
                    if (j == _j) continue;
                    newM2[i, x] = newM1[i, j];
                    x++;
                }
            }
            return newM2;
        }
        static decimal GetAlgDop(int[,] m, int _i, int _j)
        {//вычисление матр алг дополнений
            decimal algDop = 0;
            int k = m.GetLength(0) - 1;
            int[,] _m = DelRowCol(m, _i, _j);
            if (k == 1) { algDop = _m[0, 0]; }
            else
            {
                for (int j = 0; j < k; j++)
                {
                    algDop += _m[0, j] * GetAlgDop(_m, 0, j);
                }
            }
            int tmpI = (_i + _j) % 2;
            if (tmpI == 1) { return -algDop; }
            else { return algDop; }
        }
        static decimal[] GetDetMatrix(int[,] m)
        {//вычисление детерминанта
            decimal[] det_ = new decimal[m.GetLength(0)];
            //decimal det = 0;
            for (int i = 0; i < m.GetLength(0); i++)
            {
                det_[i] = m[0, i] * GetAlgDop(m, 0, i);
                //det += m[0, i] * GetAlgDop(m, 0, i);
            }
            return det_;
        }
        public static int GetOstat(decimal delim, int delit)
        {//правильный остаток
            if (delit != 0)
            {
                while ((delim < 0) || (delim >= delit))
                {
                    delim %= delit;
                    if (delim < 0) { delim += delit; }
                }
                return Convert.ToInt32(delim);
            }
            else { return -1; }
        }
        public static int GetOstat(int delim, int delit)
        {//правильный остаток
```

```
            if (delit != 0)
            {
                while ((delim < 0) || (delim >= delit))
                {
                    delim %= delit;
                    if (delim < 0) { delim += delit; }
                }
                return Convert.ToInt32(delim);
            }
            else { return -1; }
        }
        public static int[,] MultiplyMatrixMod(int[,] leftMatrix, int[,]
rightMatrix, int mod)
        {//перемножение матриц
            int row = leftMatrix.GetLength(0);
            int med = rightMatrix.GetLength(0);
            int col = rightMatrix.GetLength(1);
            int[,] result = new int[row, col];
            //for (int i = 0; i < row; i++) { result[i].resize(col); }
            for (int i = 0; i < row; i++)
            {
                for (int j = 0; j < col; j++)
                {
                    long sum = 0;
                    for (int k = 0; k < med; k++)
                    {
                        sum += leftMatrix[i, k] * rightMatrix[k, j];
                    }
                    result[i, j] = GetOstat(sum, mod);
                }
            }
            return result;
        }
        static int GetRevDet(int[,] m, int mod)
        {//обратный дет в кольце
            decimal[] det_ = GetDetMatrix(m);
            for (int i = 1; i < mod; i++)
            {
                decimal det = 0;
                for (int j = 0; j < det_.GetLength(0); j++)
                {
                    det += GetOstat(i * det_[j], mod);
                }
                if (det % mod == 1) return i;
            }
                return -1;
        }
        public static int[,] GetRevMatrix(int[,] m, int mod)
        {//обратная матрица по модулю
            int revDet = GetRevDet(m, mod);
            int k = m.GetLength(0);
            int[,] result = new int[k, k];
            for (int i = 0; i < k; i++)
            {
                for (int j = 0; j < k; j++)
                {
                    decimal tmpResult = revDet * GetAlgDop(m, i, j);
                    result[j, i] = GetOstat(tmpResult, mod);
                }
            }
            return result;
```

```
        }
        public static bool CheckDetMatrix(int[,] matrix, int mod)
        {//истина, если обр дет нет
            int k = GetRevDet(matrix, mod);
            if (k == -1) { return true; }
            return false;
        }
        public static int GetSqrt(int x)
        {
            int i;
            for (i = 1; i * i <= x; i++)
            {
                if (i * i <= x) { if (i * i == x) { return i; } }
            }
            return 10 + i;
        }
    }
}
```

1.5. Файл «CipherVigenere.cs»

```
namespace Uskov_cipher_Vigenere_Hill
{
    internal class CipherVigenere
    {
        public static string SetKey(long ID, string key)
        {
            string answer = "Ключ для метода Виженера записал.";
            //if (key.Length > ) в телеграме длина сообщения не более 2^12
символов, БД поддреживает 2^16
            Program.Users[User_.GetIndex(ID)].keyVigenere = key;
            return answer;
        }
        static int GetCodeFirstRow(char ch)
        {
            int number = HelpFunction.ConvertSymbolToCode(ch);
            if (number + 1 + HelpFunction.alphavit.Count % 10 > HelpFunc-
tion.alphavit.Count) { return HelpFunction.alphavit.Count - 1 - (number %
10); }
            return (number / 10 + 1) * 10 - 1 - number % 10;
        }
        static string GetSymbolFirstRow(int number)
        {
            int index;
            if (number + 1 + HelpFunction.alphavit.Count % 10 > HelpFunc-
tion.alphavit.Count) { index = HelpFunction.alphavit.Count - 1 - (number %
10); }
            else { index = (number / 10 + 1) * 10 - 1 - number % 10; }
            return HelpFunction.ConvertCodeToSymbol(index);
        }
        public static string EncText(long ID, string text)
        {
            string key = Program.Users[User_.GetIndex(ID)].keyVigenere;
            string newText = "";
            for (int i = 0; i < text.Length; i++)
            {
                newText += HelpFunc-
tion.ConvertCodeToSymbol((GetCodeFirstRow(text[i]) + HelpFunc-
tion.ConvertSymbolToCode(key[i % key.Length])) % HelpFunc-
tion.alphavit.Count);
            }
```

```
            return newText;
        }
        public static string DecText(long ID, string text)
        {
            string key = Program.Users[User_.GetIndex(ID)].keyVigenere;
            string newText = "";
            for (int i = 0; i < text.Length; i++)
            {
                newText += GetSym-
bolFirstRow(HelpFunction.GetOstat(HelpFunction.ConvertSymbolToCode(text[i]
) - HelpFunction.ConvertSymbolToCode(key[i % key.Length]), HelpFunc-
tion.alphavit.Count));
            }
            return newText;
        }
    }
}
```

1.6. Файл «CipherHill.cs»

```
namespace Uskov_cipher_Vigenere_Hill
{
    internal class CipherHill
    {
        public static string SetKey(long ID, string key)
        {
            string answer = "Спасибо, ключ для метода Хилла записал:\n";
            if (key.Length > 49)
            {
                key = key.Substring(0, 49);
                answer = "Ключ оказался слишком длинным (больше 49 симво-
лов), я записал:\n";
            }
            else if (HelpFunction.GetSqrt(key.Length) > 10)
            {
                int start = HelpFunc-
tion.alphavit.IndexOf(key.ToCharArray()[key.Length - 1]) + 1;
                int tmp1 = key.Length;
                int tmp2 = HelpFunction.GetSqrt(tmp1) - 10;
                for (int i = 0; i < tmp2*tmp2 - tmp1; i++)
                {
                    if (start > HelpFunction.alphavit.Count) { start = 0;
}
                    key += HelpFunction.alphavit[start].ToString();
                    start++;
                }
                answer = "Ключ оказался неправильной длины (должна быть 4,
9, ...), я записал:\n";
            }
            int poryadok = Convert.ToInt32(Math.Sqrt(key.Length));
            int[,] matrixKey = new int[poryadok, poryadok];
            int step = 0;
            while (true)
            {
                string newKey = key.Substring(key.Length - step, step) +
key.Substring(0, key.Length - step);
                for (int i = 0, help = 0, index = 0; i < poryadok; i++)
                {
                    for (int j = 0; j < poryadok; j++)
                    {
```

```
                        matrixKey[i, help * (poryadok - 1) + Con-
vert.ToInt32(Math.Pow(-1, help)) * j] = HelpFunc-
tion.ConvertSymbolToCode(newKey[index]);
                        index++;
                    }
                    if (help == 0) { help = 1; }
                    else { help = 1; }
                }
                if (HelpFunction.CheckDetMatrix(matrixKey, HelpFunc-
tion.alphavit.Count))
                {
                    step++;
                    if (step == poryadok * poryadok)
                    {
                        answer = "Увы ключ не подходит. Я не смогу расшиф-
ровать методом Хилла.\nВведите другой ключ.";
                        break;
                    }
                    else { continue; }
                }
                else { Program.Users[User_.GetIndex(ID)].keyHill = newKey;
break; }
            }
            return answer;
        }
        public static string EncText(long ID,  string text)
        {
            string key = Program.Users[User_.GetIndex(ID)].keyHill;
            int poryadok = Convert.ToInt32(Math.Sqrt(key.Length));
            int[,] matrixKey = new int[poryadok, poryadok];
            for (int i = 0, help = 0, index = 0; i < poryadok; i++)
            {
                for (int j = 0; j < poryadok; j++)
                {
                    matrixKey[i, help * (poryadok - 1) + Con-
vert.ToInt32(Math.Pow(-1, help)) * j] = HelpFunc-
tion.ConvertSymbolToCode(key[index]);
                    index++;
                }
                if (help == 0) { help = 1; }
                else { help = 1; }
            }
            if (text.Length % poryadok != 0)
            {
                text += new string(' ', poryadok - (text.Length % por-
yadok));
            }
            int[,] matrixText = new int[text.Length / poryadok, poryadok];
            for (int i = 0, index = 0; i < text.Length / poryadok; i++)
            {
                for (int j = 0; j < poryadok; j++)
                {
                    matrixText[i, j] = HelpFunc-
tion.ConvertSymbolToCode(text[index]);
                    index++;
                }
            }
            int[,] matrixNewText = HelpFunc-
tion.MultiplyMatrixMod(matrixText, matrixKey, HelpFunc-
tion.alphavit.Count);
            string newText = "";
```

```
            for (int i = 0; i < matrixNewText.GetLength(0); i++)
            {
                for (int j = 0; j < matrixNewText.GetLength(1); j++)
                {
                    newText += HelpFunc-
tion.ConvertCodeToSymbol(matrixNewText[i, j]);
                }
            }
            return newText;
        }
        public static string DecText(long ID, string text)
        {
            string key = Program.Users[User_.GetIndex(ID)].keyHill;
            int poryadok = Convert.ToInt32(Math.Sqrt(key.Length));
            int[,] matrixKey = new int[poryadok, poryadok];
            for (int i = 0, help = 0, index = 0; i < poryadok; i++)
            {
                for (int j = 0; j < poryadok; j++)
                {
                    matrixKey[i, help * (poryadok - 1) + Con-
vert.ToInt32(Math.Pow(-1, help)) * j] = HelpFunc-
tion.ConvertSymbolToCode(key[index]);
                    index++;
                }
                if (help == 0) { help = 1; }
                else { help = 1; }
            }
            matrixKey = HelpFunction.GetRevMatrix(matrixKey, HelpFunc-
tion.alphavit.Count);
            if (text.Length % poryadok != 0)
            {
                text += new string(' ', poryadok - (text.Length % por-
yadok));
            }
            int[,] matrixText = new int[text.Length / poryadok, poryadok];
            for (int i = 0, index = 0; i < text.Length / poryadok; i++)
            {
                for (int j = 0; j < poryadok; j++)
                {
                    matrixText[i, j] = HelpFunc-
tion.ConvertSymbolToCode(text[index]);
                    index++;
                }
            }
            int[,] matrixNewText = HelpFunc-
tion.MultiplyMatrixMod(matrixText, matrixKey, HelpFunc-
tion.alphavit.Count);
            string newText = "";
            for (int i = 0; i < matrixNewText.GetLength(0); i++)
            {
                for (int j = 0; j < matrixNewText.GetLength(1); j++)
                {
                    newText += HelpFunc-
tion.ConvertCodeToSymbol(matrixNewText[i, j]);
                }
            }
            return newText;
        }
    }
}
```