Оглавление

[Введение 3](#_Toc135273062)

[1. Аналитический обзор аналогов 4](#_Toc135273063)

[2. Проектирование 6](#_Toc135273064)

[3. Разработка программного средства 9](#_Toc135273065)

[4. Руководство пользователя 10](#_Toc135273066)

[5. Разбор алгоритма 13](#_Toc135273067)

[6. Криптостойкость 16](#_Toc135273068)

[Списки используемых источников 18](#_Toc135273069)

[Приложение А 19](#_Toc135273070)

[Приложение Б 22](#_Toc135273071)

# Введение

Современные методы обработки, передачи и накопления информации способствовали появлению угроз, связанных с возможностью потери, искажения и раскрытия данных, адресованных или принадлежащих конечным пользователям. Поэтому обеспечение информационной безопасности компьютерных систем и сетей является одним из ведущих направлений развития информационных технологий.

Защита информации — деятельность по предотвращению утечки защищаемой информации, несанкционированных и непреднамеренных воздействий на защищаемую информацию

Криптография — это наука о безопасном общении в присутствии враждебного поведения. Большинство людей понимает роль криптографии в сохранении конфиденциальности коммуникации (например, приложения для обмена сообщениями с шифрованием), однако она также используется для аутентификации происхождения, проверки целостности и для определения того, что сообщение пришло от конкретного человека и не было подделано, и что эти свойства не могут быть опровергнуты отправителем.

В рамках курсового проектирования будет разработан и реализован в виде программного средства модифицированный метод шифрования Цезаря.

Наш шифр будет использовать куда больше ключей для шифрования чем шифр Цезаря.

В модифицированном шифре идет привязка ключей к времени, которое меняется каждый день и час. А вот для расшифровки зашифрованного сообщения нам ключи уже не нужны, они будут передаваться в нашем сообщения как ее часть.

# Аналитический обзор аналогов

При проектировании и разработке своего метода в первую очередь должно учитываться положение о том, что злоумышленник обладает полной информацией о подробностях ее реализации. Неизвестным остается только ключ, или алгоритм шифрования.

Для того, чтобы улучшить разрабатываемый метода и его программную реализацию, предлагается рассмотреть существующие в открытом доступе решения, имеющие сходство с разрабатываемым алгоритмом.

Аналогами нашего шифра будут шифр Цезаря и Аффинный шифр.

Алгоритм шифрования Цезаря заключается в замене каждого символа входящего сообщения на символ, который находится на некотором константном расстоянии с правой или левой стороны. Расстояние при этом называют – ключом.

Программа не поддерживает интерфейс пользователя и вся ее работа показана в командной строке, это показана на рисунок 1.1.

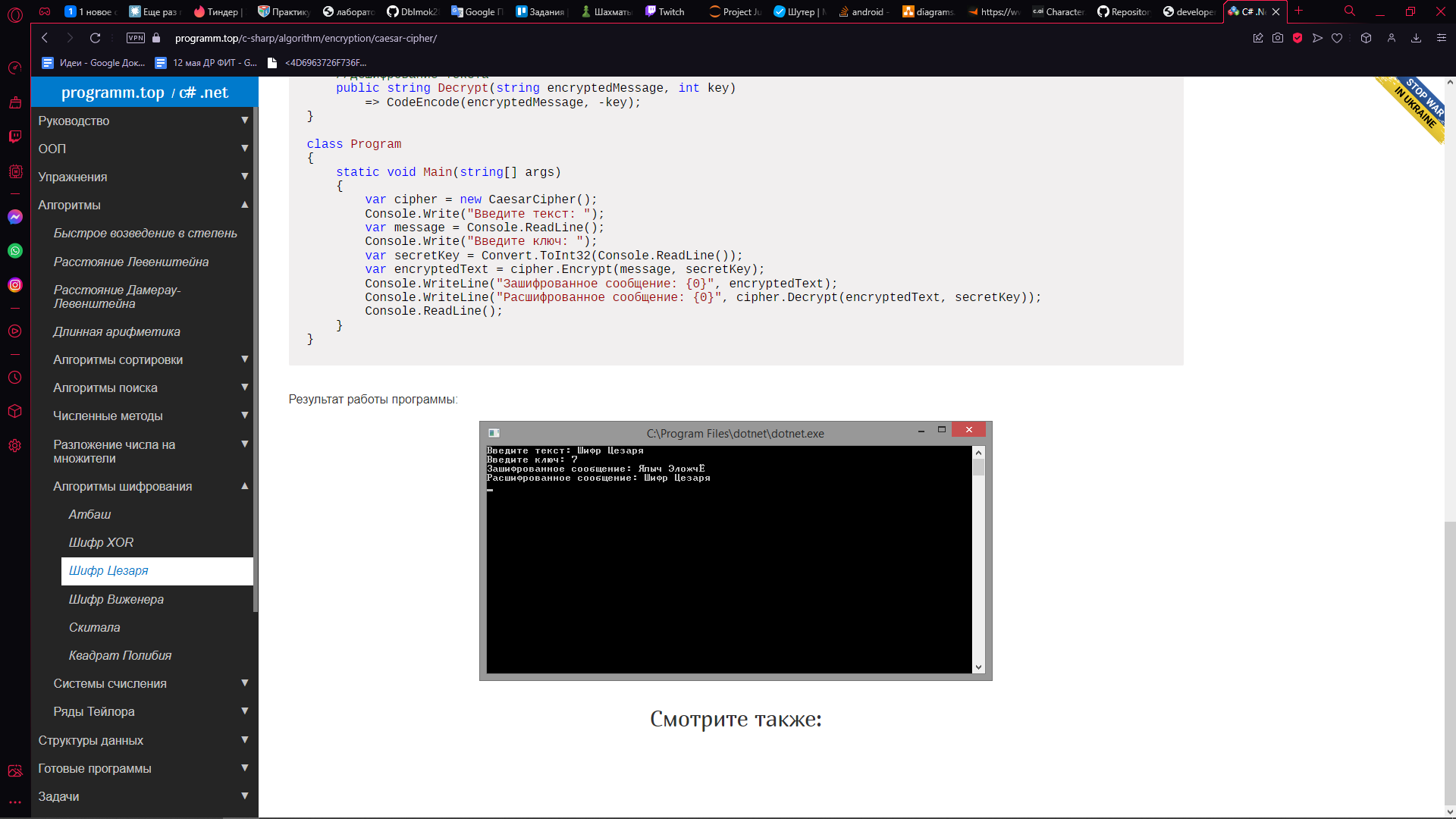


Рисунок. 1.1 - Использование программы Шифр Цезаря

Аффинный шифр является криптографическим методом на основе моно-буквенные замены шифра, то есть сказать, что оригинал письма заменяется только одной другой буквой, в отличие от шифра Хилла. Это код, который легко понять, но также один из самых простых для взлома. Создатель аффинного шифра неизвестен.

Для шифрования и расшифрования используются две формулы: для шифрования: y = (A\*x+B) mod N и для дешифрования: x = A-1(x-B) mod N

Здесь N- размер алфавита, а A и B ключи. Ключи выбираются так, чтобы они были взаимно простыми числами. В дешифровании A-1 число обратное к A число по модулю N (1=A\*A-1 mod N).

Пример использования программного средства для аффинного шифра, рисунок 1.2.

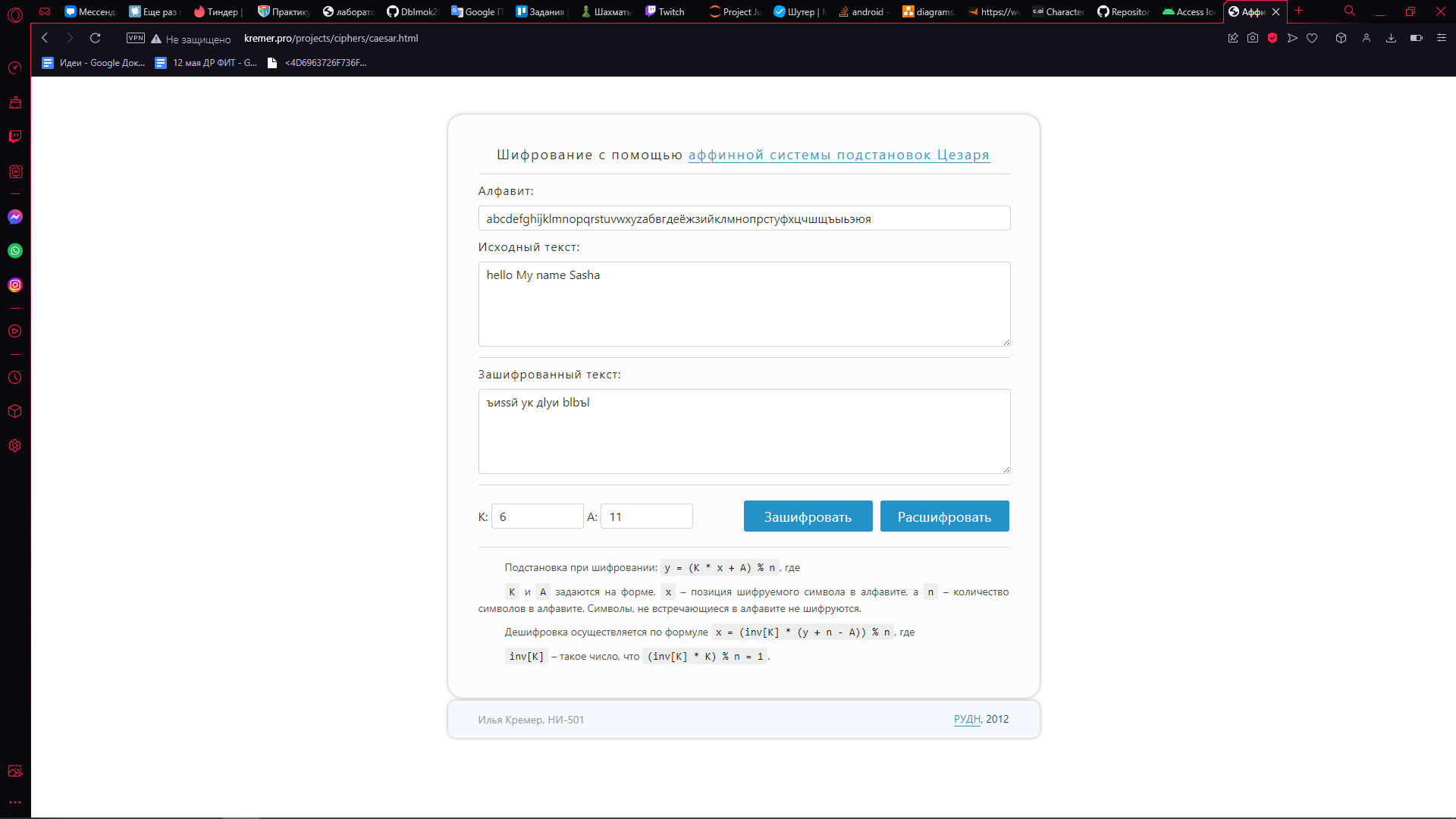


Рисунок. 1.2 – Пример использования

Рассмотрев аналоги, мы понимаем, что они используют ключи с ограничениями, в шифре Цезаря максимальное значение ключа зависит от длины используемого алфавита, а в аффинном шифре у нас используются два ключа и они могут быть только взаимно простыми ключами. Поэтому нам следует сделать 2 вещи: увеличить количество используемых ключей и сделать их без ограничений.

# Проектирование

Среда разработки или IDE (сокр. от англ. Integrated Development Environment – интегрированная среда разработки) – специальный программный комплекс, предназначенный для полного цикла написания и тестирования программ на определенном языке.

Среда разработки включает в себя:

* текстовый редактор;
* Транслятор ([компилятор](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80) и/или [интерпретатор](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D1%80%D0%B5%D1%82%D0%B0%D1%82%D0%BE%D1%80));
* средства автоматизации сборки;
* отладчик.

Иногда содержит также средства для интеграции с [системами управления версиями](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%D0%BC%D0%B8) и разнообразные инструменты для упрощения конструирования г[рафического интерфейса пользователя](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F). Многие современные среды разработки также включают браузер классов, инспектор объектов и [диаграмму](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0) [иерархии классов](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%8F_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%BE%D0%B2) для использования при [объектно-ориентированной разработке](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) ПО.

Мы используем для реализации нашего алгоритма Visual Studio 2022 и язык C#. Окно среды разработки представлено на рисунке 2.1.

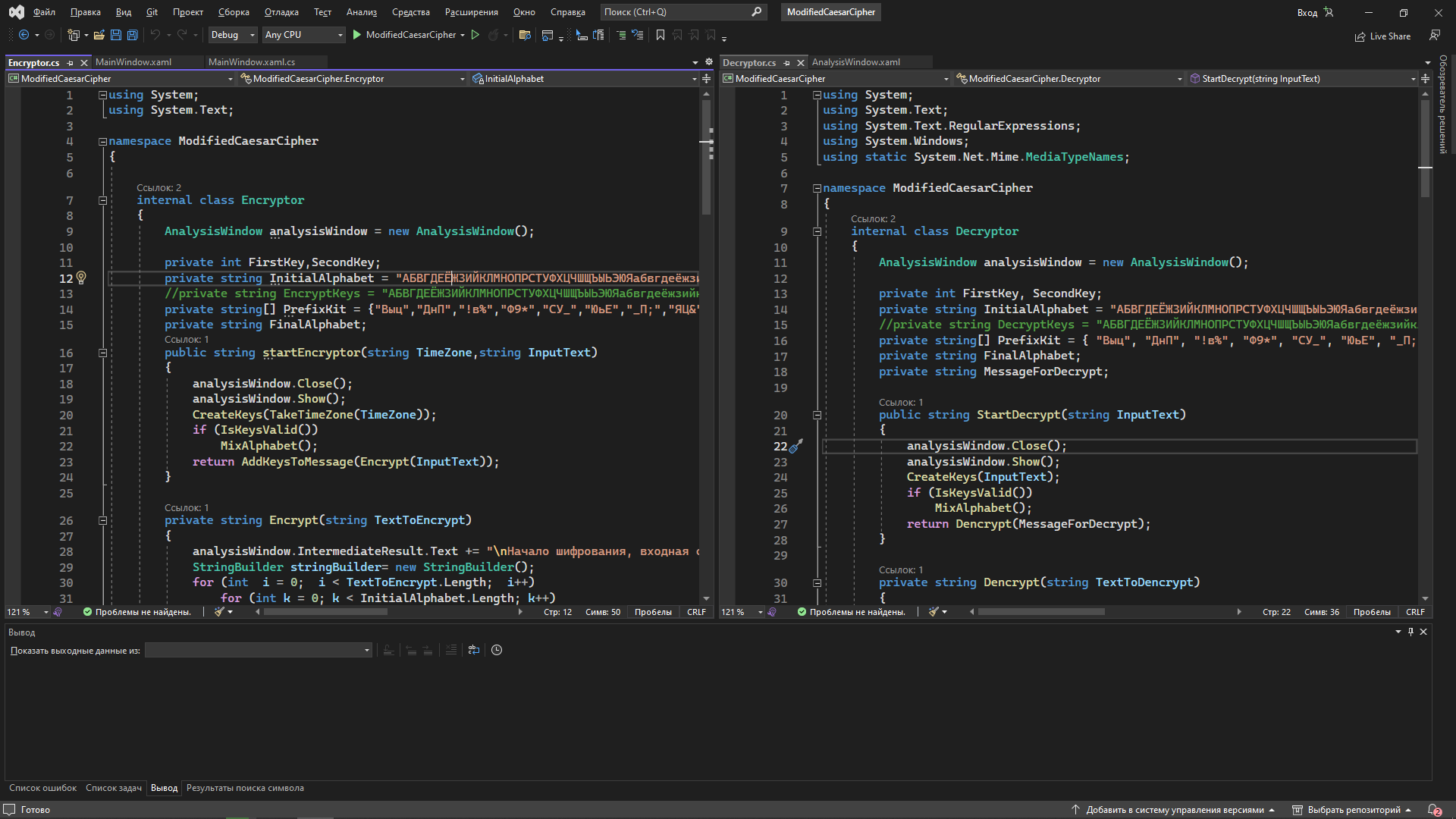


Рисунок. 2.1 – Среда разработки

Пройдемся по Visual Studio 2022. Сверху у нас меню, в котором есть вкладки такие как Файл, Вид, Проект, Сборка, Отладка и т.д., рисунок 2.2.

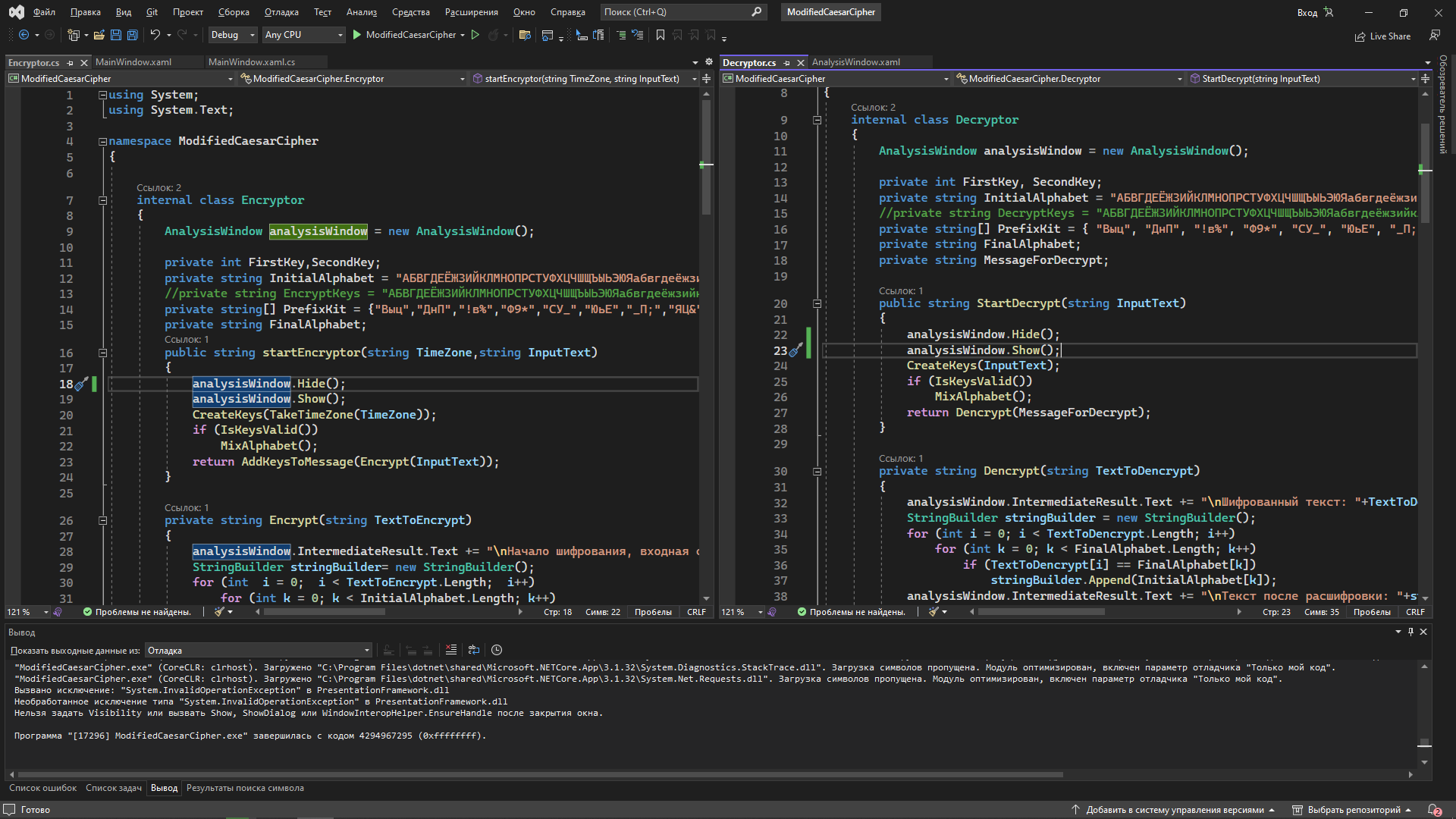


Рисунок. 2.2 – Верхнее меню

Справа обозреватель решений, в нем расположена наша иерархия проекта, рисунок 2.3.

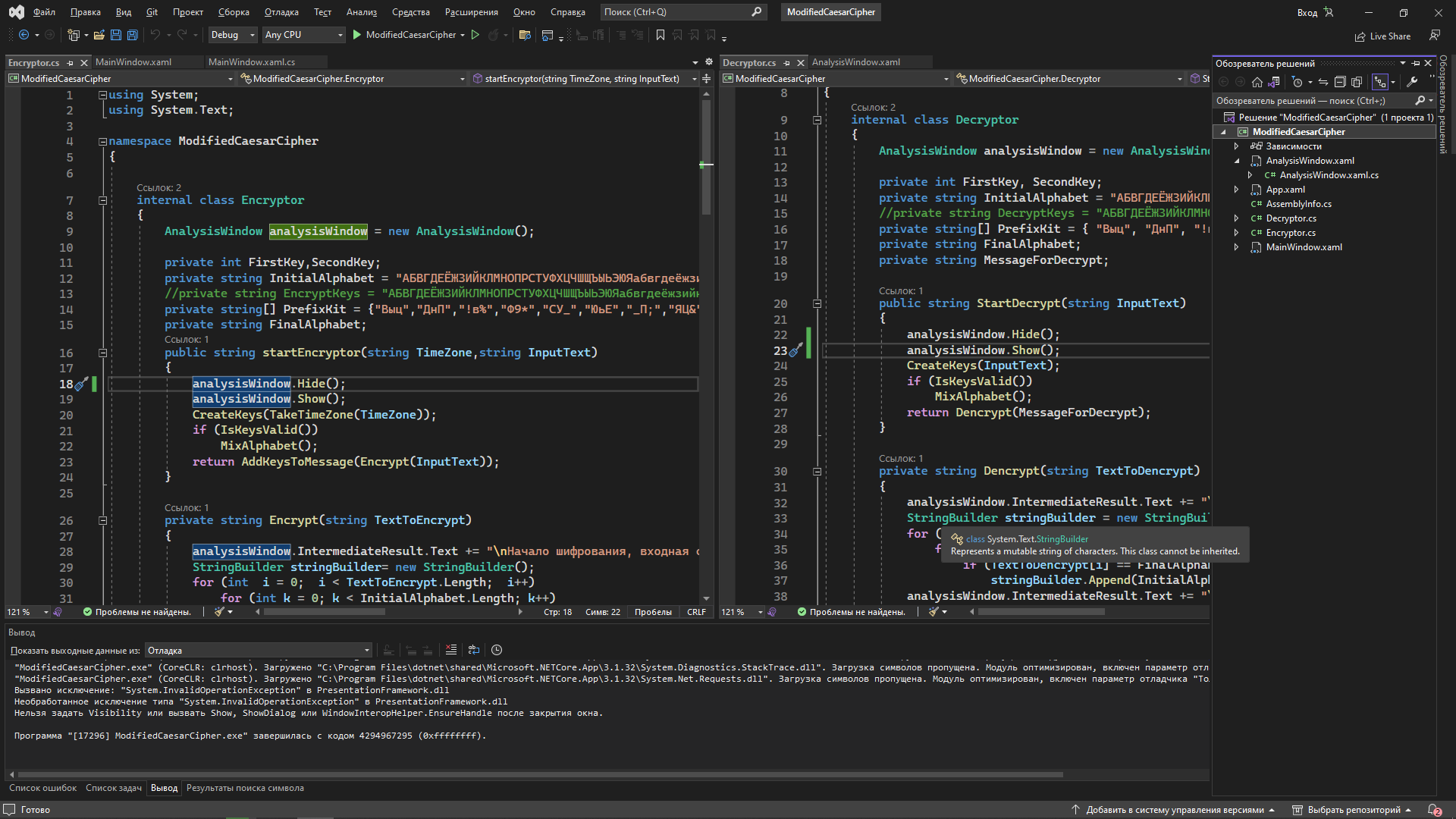


Рисунок. 2.3 – Обозреватель решений

Так как мы используем WPF для создания проекта, то надо пройтись и по конструктору нашего приложения, рисунок 2.4.

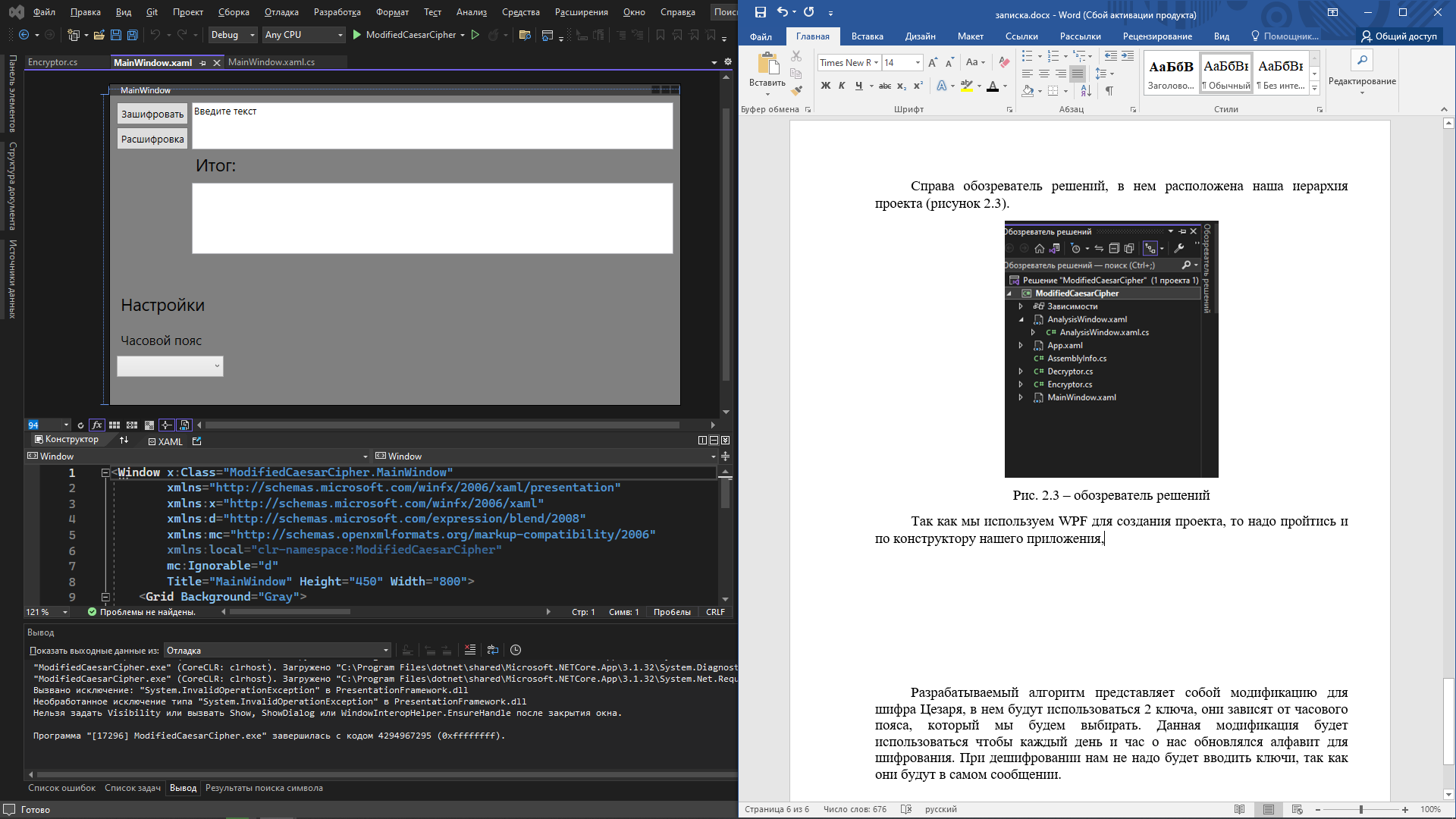


Рисунок. 2.4 – Окно конструктора

В центре нам показано окно нашего приложения, снизу настройка элементов и самого окна.

# Разработка программного средства

Программное средство, реализующее модифицированный алгоритм Цезаря будет разработано в Visual Studio 2022 на языке C# с использованием технологией WPF.

Среди основных функций приложения можно выделить:

* Генерация ключей из часового пояса.
* Отображение информации о ходе работы.
* Выбор часового пояса.
* Открытие файла
* Сохранение в файл

Разработанное приложение имеет файловую структуру, представленную на рисунке 3.1.

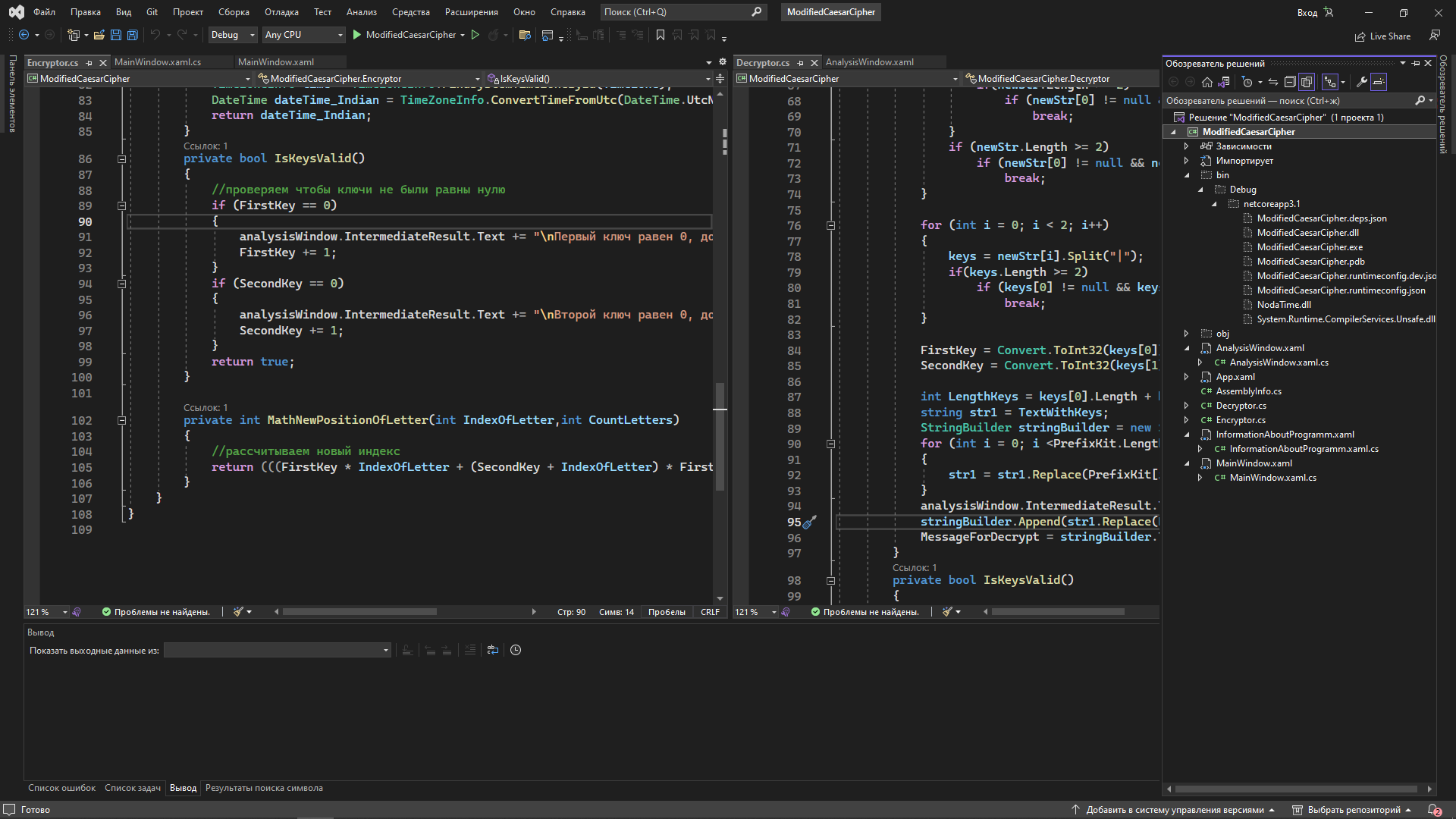


Рисунок 3.1 – Файловая структура

Страница “MainWindow” является главным файлом, с помощью которого происходит выполнение программы. Файл “Encrypt” содержит алгоритм для шифрования информации, а файл “Decrypt” содержит алгоритм для расшифрования. На странице “AnalysisWindow” у нас будет появляться информация о проделанных шагах. Страница “InformationAboutProgramm” содержит окно с информацией о программе.

Для корректной работы программы необходимо обеспечить защиту работы пользователя от ошибок и сбоев. Для этого используются конструкция типа «if-else». Она служат для «отлавливания» исключений с последующей их обработкой. Это необходимо, чтобы при вводе некорректной информации программа не получала сбоев. В нашем случае, были проведены проверки на наличие текста для шифрования или дешифрования.

# Руководство пользователя

Для использования проектируемого программного средства нужно запустить exe-файл с именем ModifiedCaesarCipher.exe. При запуске открывается основное окно приложения, представленное на рисунке 4.1.

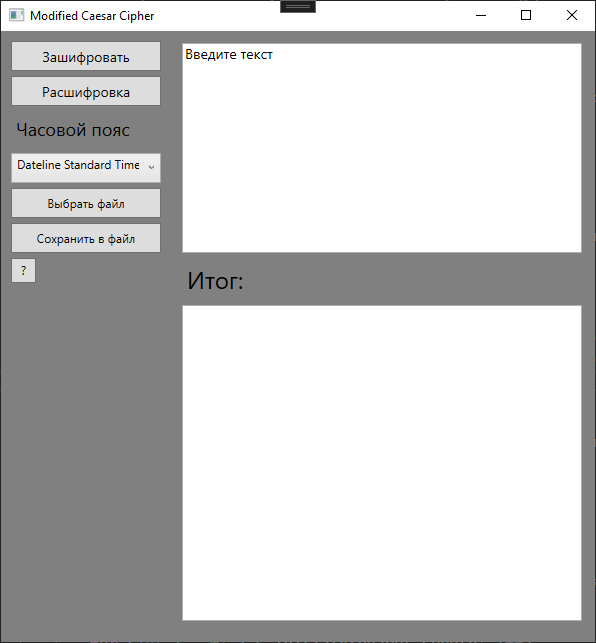


Рисунок 4.1 – Главное окно программы

В поле, где написано “Введите текст”, мы вводим наше сообщение, далее выбираем часовой пояс, по стандартным настройкам у нас стоит системное время. После нажимаем на кнопку “Зашифровать” и у нас открывается окно с дополнительной информацией шифрования, рисунок 4.2.

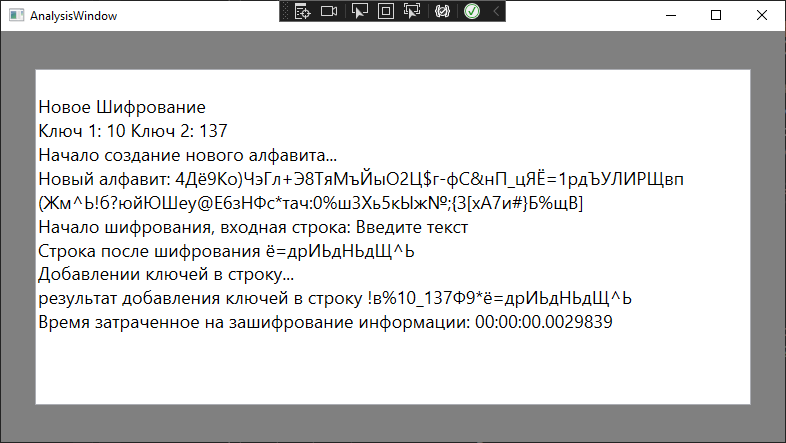


Рисунок. 4.2 – Окно информации

Для дешифрования открывается точно такое же окно, но с информацией о прошедшей дешифровки.

После этого в окне “Итог”, появляется зашифрованное сообщение, рисунок 4.3.

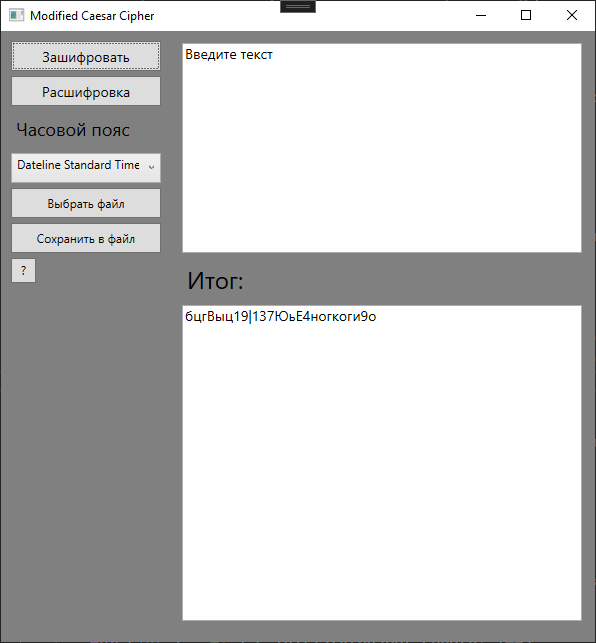


Рисунок 4.3 – Получено зашифрованное сообщение

Для дешифровки нам надо скопировать зашифрованное сообщение и вставить его в первый блок, и нажать кнопку “Расшифровать”.

Пользователь имеет может ознакомиться с информацией о проекте путем нажатия на кнопку «?», рисунок 4.3.

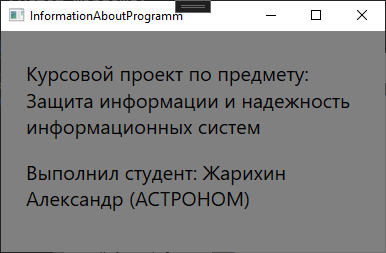


Рисунок 4.3 – Информация о проекте

Так же у нас есть возможность выгрузить информацию из .txt файла и сохранить в .txt файл, рисунок 4.4.

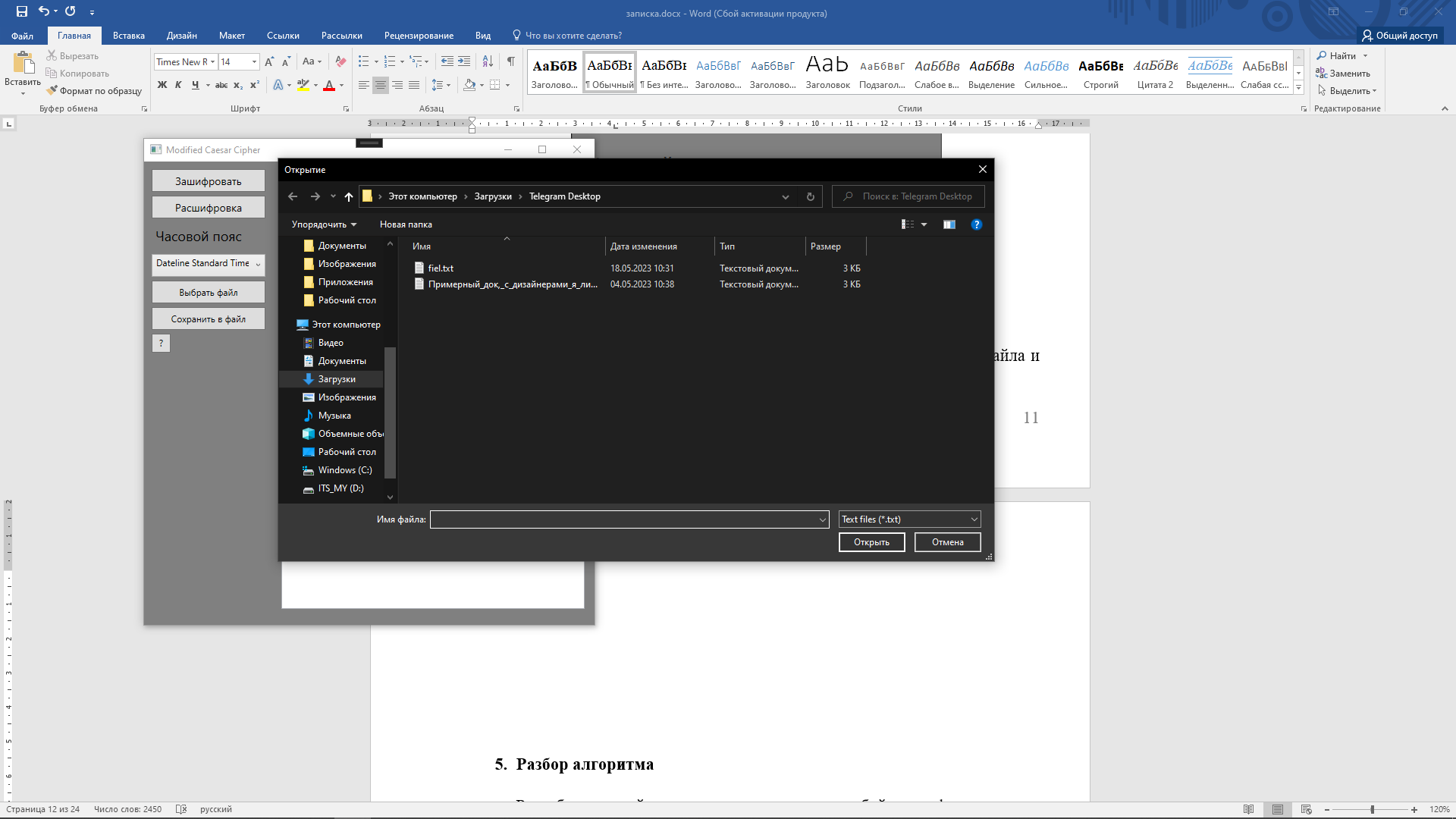


Рисунок 4.4 –Кнопки открыть файл и сохранить файл

При нажатии на кноп отвечающую за выбор часового пояса у нас появится выпадающий список, в котором мы можем выбрать часовой пояс для генерации ключей, рисунок 4.5.

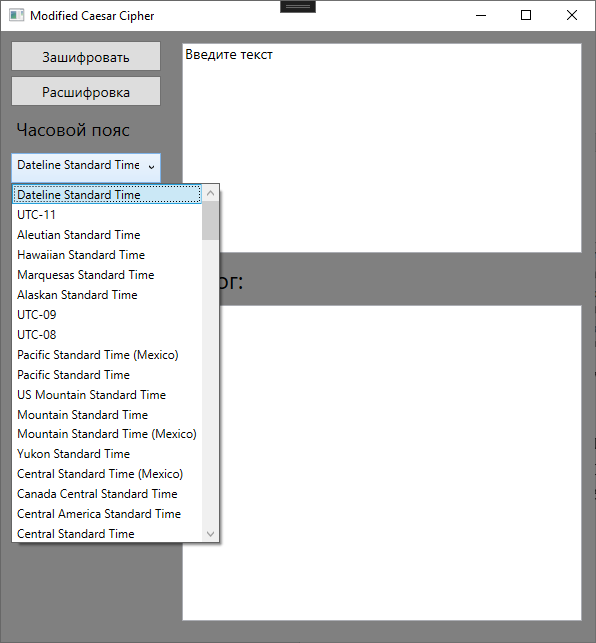


Рисунок 4.5 – Выбор часового пояса

# Разбор алгоритма

Разрабатываемый алгоритм представляет собой модификацию для шифра Цезаря, в нем будут использоваться 2 ключа, они зависят от часового пояса, который мы будем выбирать. Данная модификация будет использоваться чтобы каждый день и час у нас обновлялся алфавит для шифрования. При дешифровании нам не надо будет вводить ключи, так как они будут в самом зашифрованном сообщении.

Для начала мы должны взять нашу строку и время выбранного часового пояса и отправим их в метод startEncryptor().

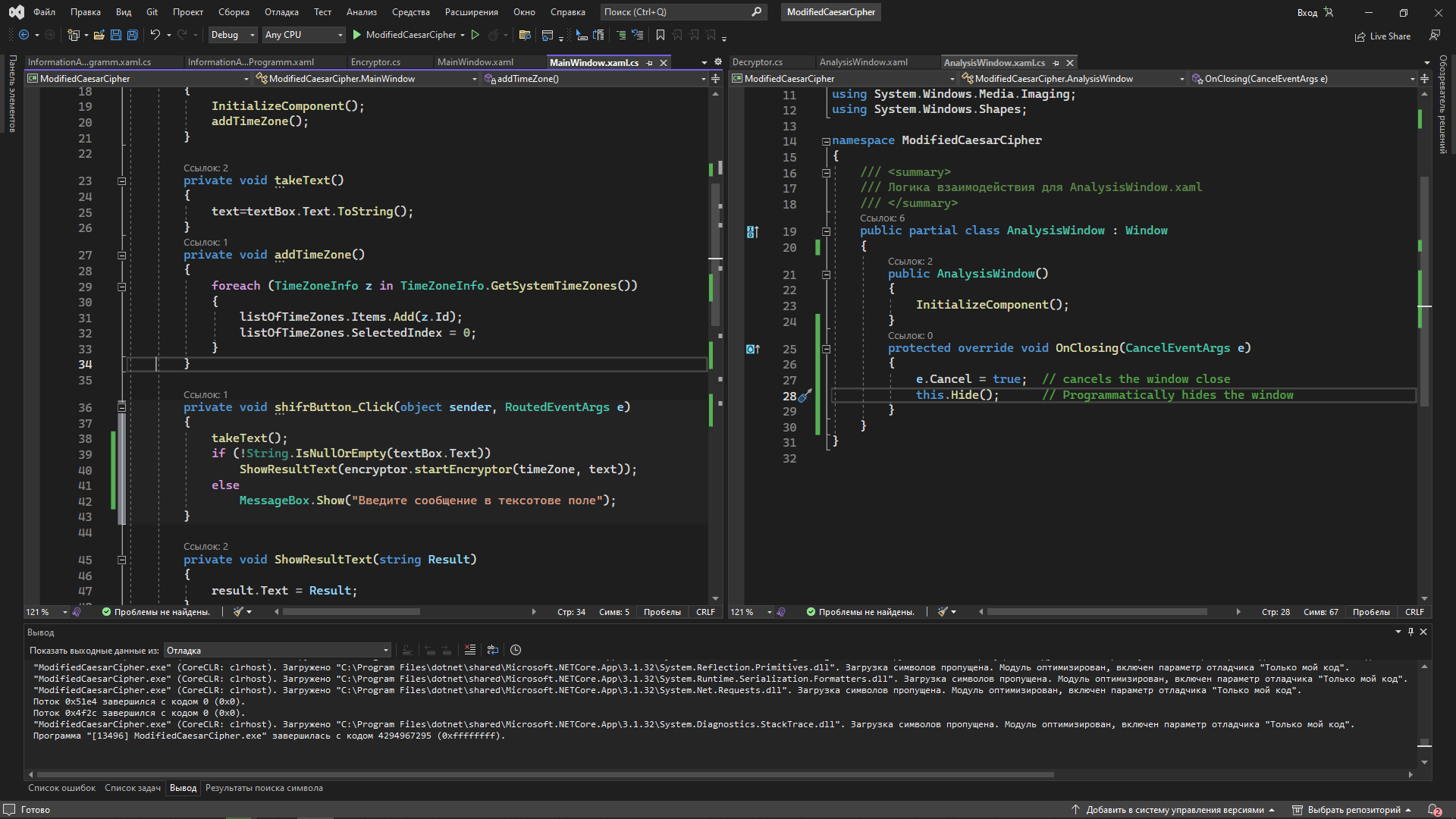


Рисунок 5.1 – Данные для шифрования

Далее в методе startEncryptor() мы создаем ключи, после чего проверяем их на правильность, затем перемешиваем алфавит и добавляем ключи в сообщение, рисунок 5.2.

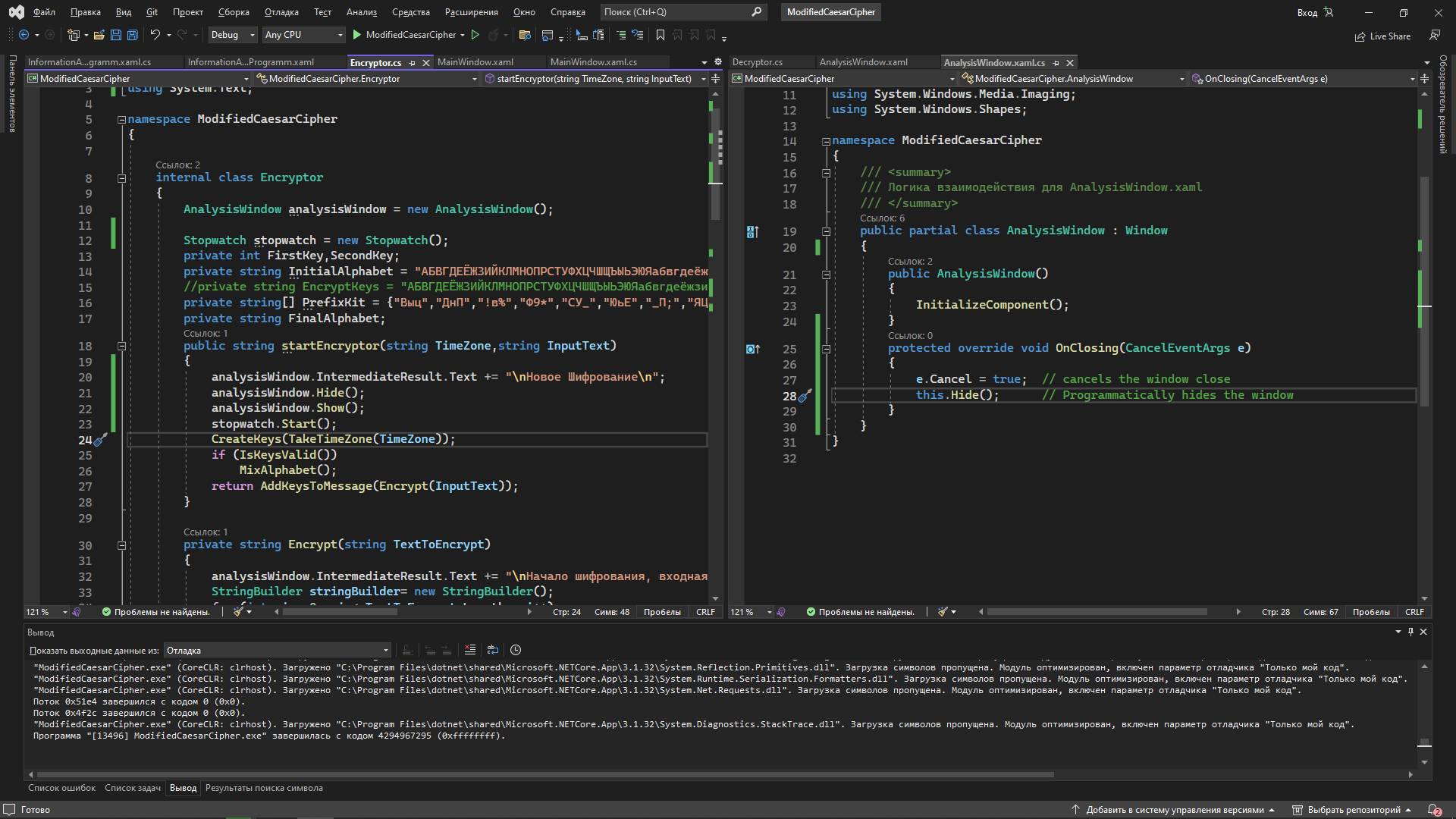


Рисунок 5.2 – Метод startEncryptor()

Метод CreateKeys(), рисунок 5.3.

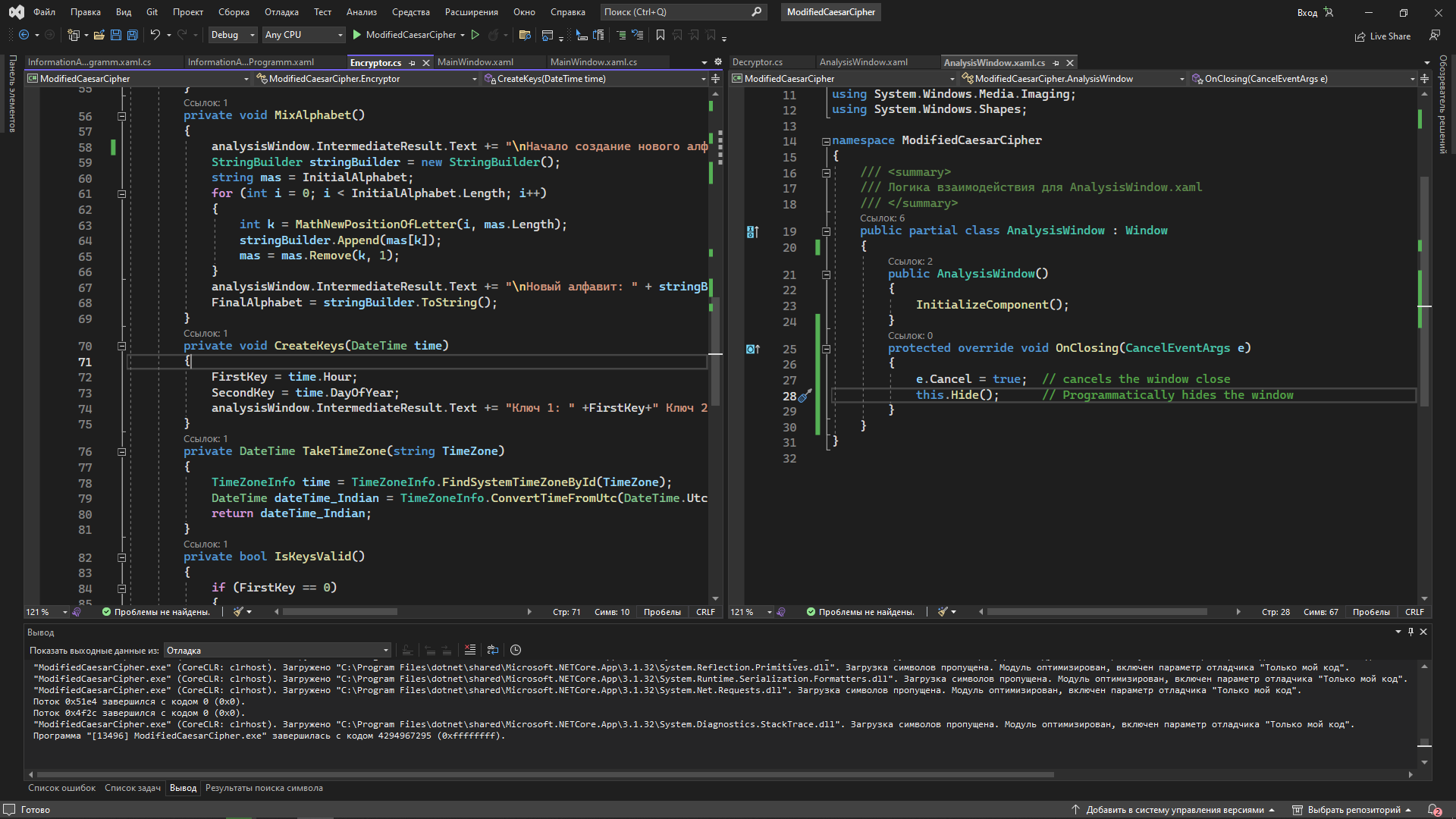


Рисунок 5.3 – Метод CreateKeys()

Метод IsKeysValid(), для проверки правильности ключей, рисунок 5.4.

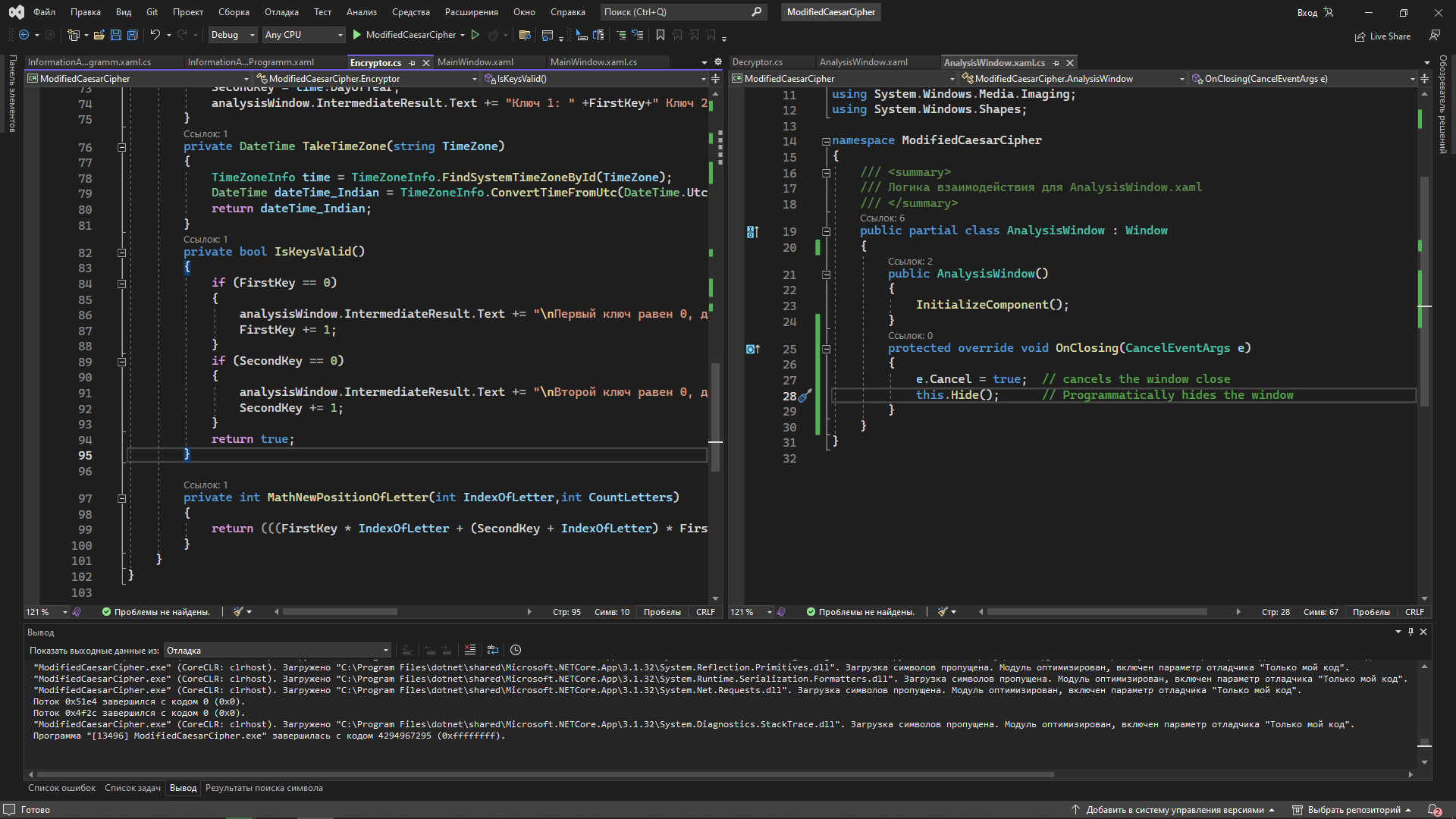


Рисунок 5.4 – Метод IsKeysValid()

Далее метод MixAlphabet(), здесь у нас на основе значений ключей создается новый перемешанный алфавит, из первоначального, длина алфавита составляет 100 символов, рисунок 5.5.

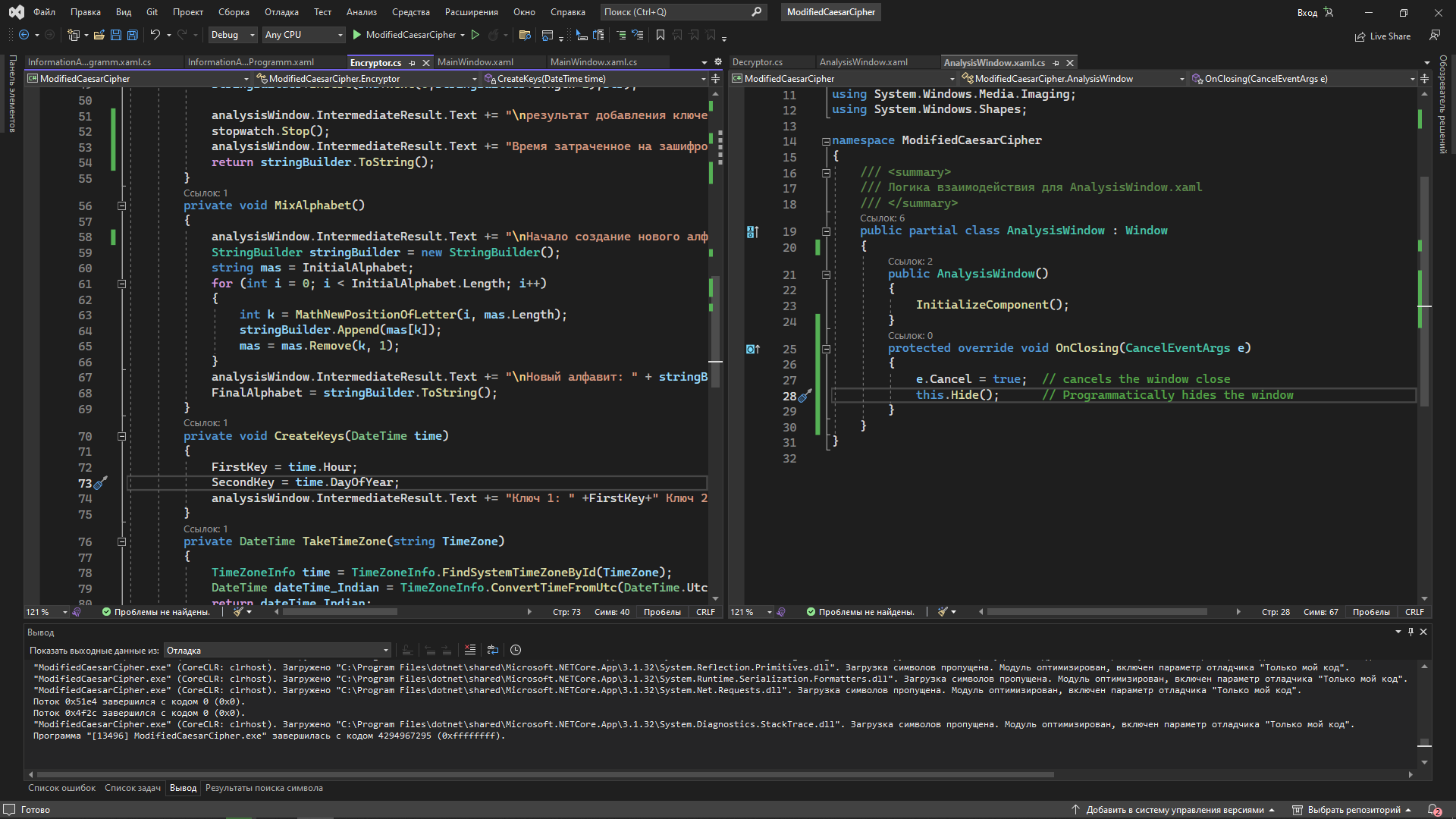


Рисунок 5.5 – Создание нового алфавита

Метод MathNewPositionOfLetter() возвращает нам индексы букв из первоначального алфавита для построения нового, здесь у нас используется собственная формула, рисунок 5.6.

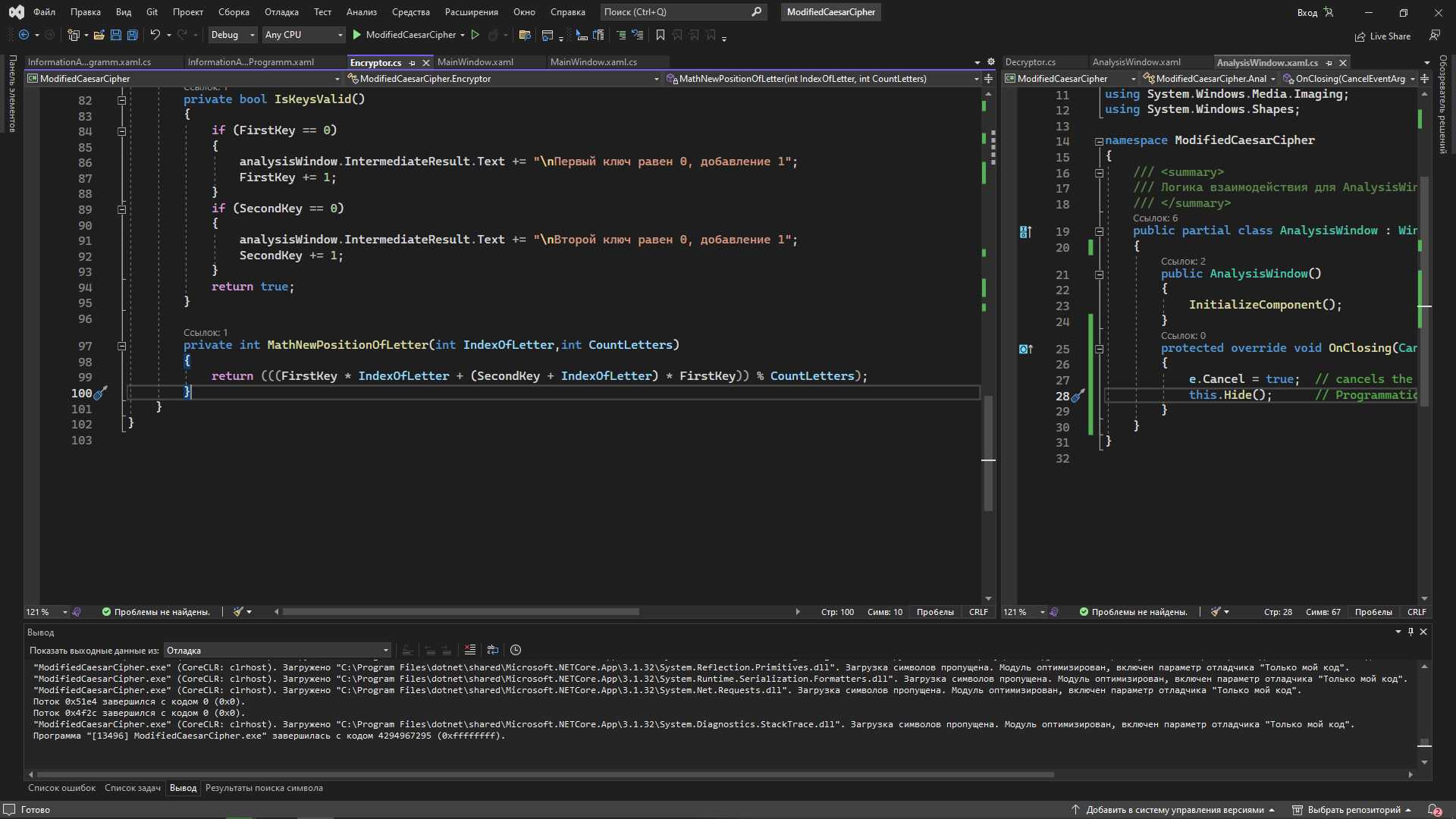


Рисунок 5.6 – Формула для нахождения индекса

После создания нового алфавита мы начинаем брать наше сообщение и искать индекс буквы в первоначальном сообщении, затем через найденный индекс мы получаем букву из нового алфавита. Таким образом у нас получается зашифрованное сообщение, метод Encrypt(), рисунок 5.7.

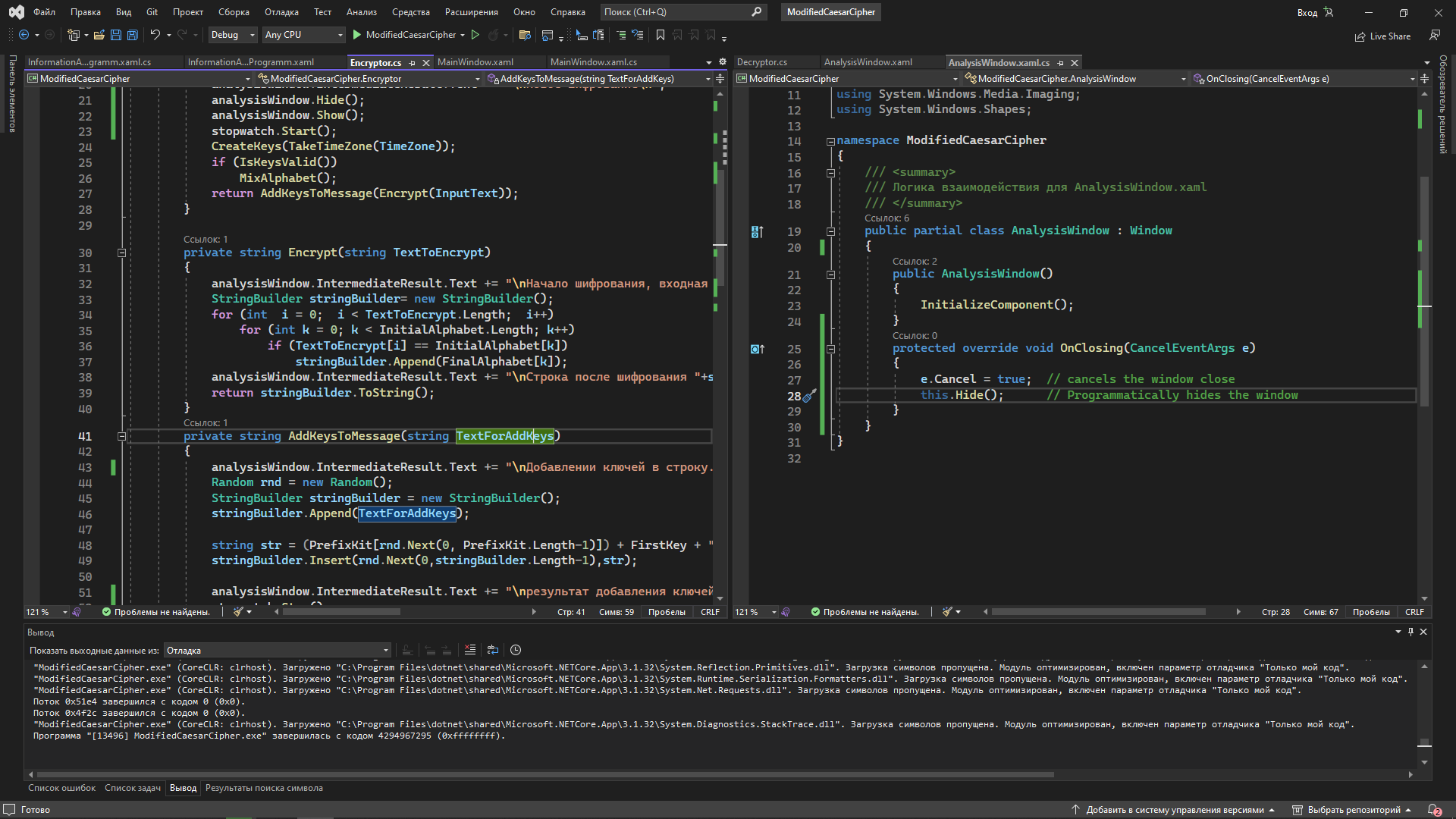


Рисунок 5.7 – Шифрование сообщения

Ну и наконец мы добавляем ключи в сообщение, метод AddKyesToMessage(), рисунок 5.8.

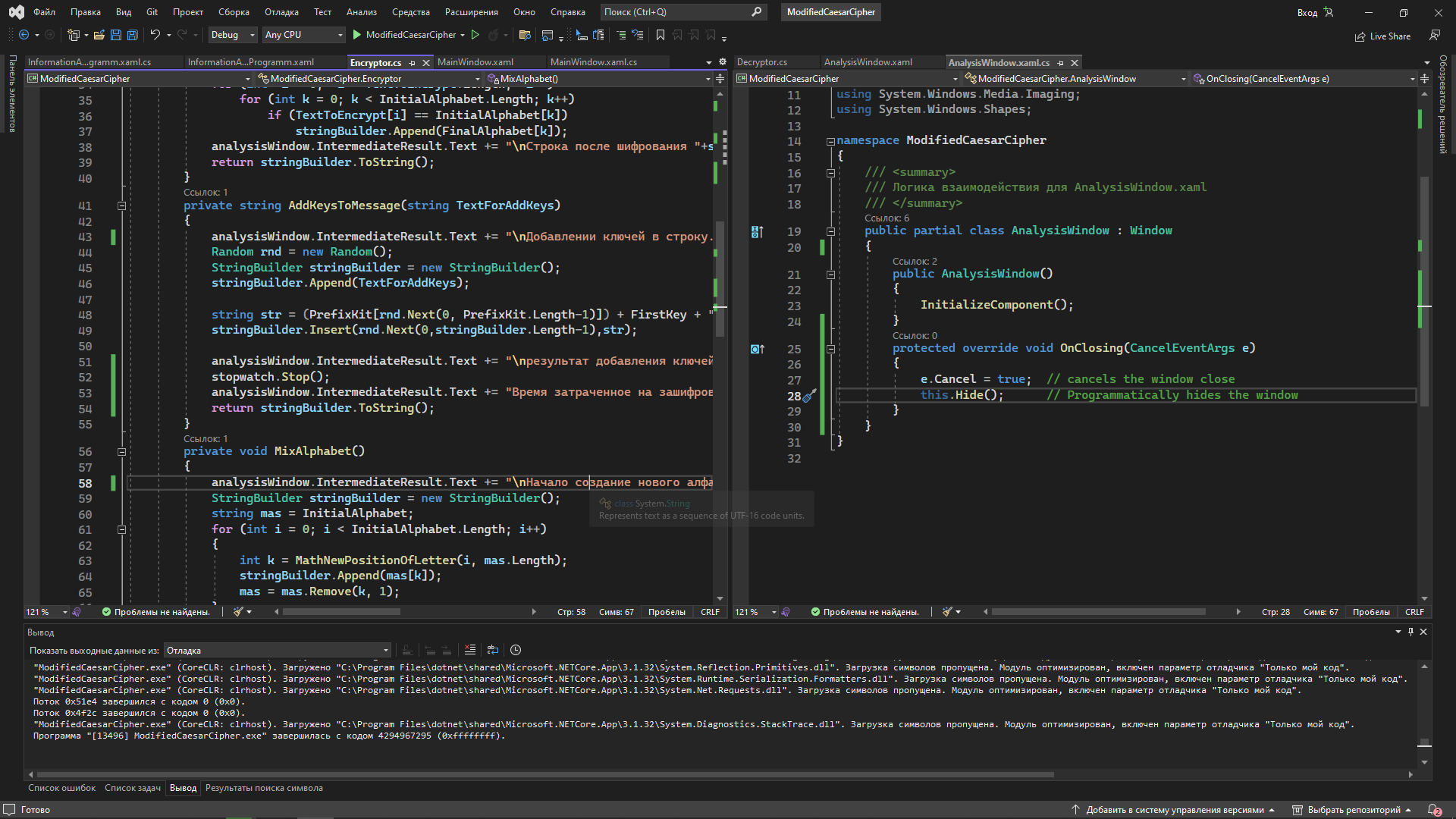


Рисунок 5.8 – Метод AddKyesToMessage()

Как работает данная схема, у нас есть пара ключей FirstKey и SecondKey, их мы должны добавить в сообщение, но так чтобы они не выделялись. Для этого у нас есть PrefixKit, это набор букв, которые являются началом и концом части в сообщении где у нас находятся ключи, всего их 8, но количество можно увеличить. Длина набора букв 3, однако это можно легко увеличить, что сделает вероятность возникновения такого же набора букв в сообщении низкой. Также они выбираются случайно и ставятся в случайное место в сообщении, таким образом в большом объеме информации легко можно потерять ключи, и они не будут бросаться в глаза.

Дешифрование работает похожим принципом, как и шифрование, только сначала мы должны достать ключи из сообщения, затем создать по этим ключам наш алфавит и расшифровать сообщение.

# Криптостойкость

Криптостойкость – основная характеристика для алгоритмов шифрования, которая обычно измеряется временем, необходимым для вскрытия того или иного метода шифрования при неких фиксированных ресурсах, имеющихся у злоумышленника.

Рассмотрим некоторые криптоаналитические методы, которые чаще всего используются в атаках.

Метод «грубой силы» подразумевает перебор всех возможных комбинаций ключа. Относится к классу методов поиска решения исчерпыванием всевозможных вариантов.

Метод «атака с известным открытым текстом» это вид криптоанализа, при котором в шифротексте присутствуют стандартные отрывки, смысл которых заранее известен аналитику.

Метод «Бандитский криптоанализ» Криптоаналитик может использовать так называемый «человеческий фактор», то есть пытаться с помощью шантажа, подкупа, пыток или иных способов получить информацию о системе шифрования или даже сам ключ шифрования. Например, дача взятки, как одна из разновидностей бандитского криптоанализа, может носить название «Вскрытие с покупкой ключа». Таким образом методика вскрытия построена на слабости людей как составной части системы защиты информации.

Бандитский криптоанализ считается очень мощным способом взлома системы, а зачастую и наилучшим путём вскрытия шифров.

Шифр Цезаря не сможет долго простоять против брутфорса, а вот наш алгоритм за счет большей возможности в ключах продержится дольше. К сравнению в шифре цезаря возможное количество ключей зависит напрямую от длины алфавита, который используется, если русский алфавит, то длина будет 33 и возможное количество ключей 33.

Если мы рассматриваем случай, когда злоумышленник не знает, как работает алгоритм, то взлом усложняется тем, что у нас меняются ключи каждый час и день, мы можем с легкостью поменять формулу для расчетов новых букв в алфавит. Если же злоумышленник знает, как работает алгоритм, то все что ему остается подобрать ключи. Это он может сделать через брутфорс. Максимальное количество ключей к нашему алгоритму 8760.

# Заключение

Важность криптографии в современном мире нельзя недооценивать. Криптография — это наука о том, как сделать сообщения нечитаемыми для третьих лиц без надлежащего ключа дешифрования. Криптография имеет большое значение для защиты информации и может использоваться во многих областях, таких как финансы, медицина, военные и т. д.

В ходе проведенной работы мы разработали модифицированный шифр Цезаря, в котором используются ключи с большим количеством возможных комбинаций. Благодаря этому он становится более надежным чем стандартный шифр Цезаря, так же был придуман способ сокрытия ключей в нашем зашифрованном сообщении.

# Списки используемых источников

1. Википедия, шифр Цезаря https://ru.wikipedia.org/wiki/Шифр\_Цезаря
2. Онлайн калькулятор шифра Цезаря https://planetcalc.ru/1434/
3. Онлайн калькулятор аффинного шифра http://kremer.pro/projects/ciphers/caesar.html
4. Википедия, аффинный шифр https://ru.wikipedia.org/wiki/Аффинный\_шифр

# Приложение А

**Класс шифрования текста**

|  |
| --- |
| using System;  using System.Diagnostics;  using System.Text;  namespace ModifiedCaesarCipher  {  internal class Encryptor  {  AnalysisWindow analysisWindow = new AnalysisWindow();  Stopwatch stopwatch = new Stopwatch();  private int FirstKey,SecondKey;  private string InitialAlphabet = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯабвгдеёжзийклмнопрстуфхцчшщъыьэюя 1234567890!@#$%^&\*()\_+=№;%:?-{}[]";  private string[] PrefixKit = {"Выц","ДнП","!в%","Ф9\*","СУ\_","ЮьЕ","\_П;","ЯЦ&"};  private string FinalAlphabet;  public string startEncryptor(string TimeZone,string InputText)  {  //метод начала шифрования  analysisWindow.IntermediateResult.Text += "\nНовое Шифрование\n";  analysisWindow.Hide();  analysisWindow.Show();  stopwatch.Start();  CreateKeys(TakeTimeZone(TimeZone));  if (IsKeysValid())  MixAlphabet();  return AddKeysToMessage(Encrypt(InputText));  }  private string Encrypt(string TextToEncrypt)  {  //Шифруем сообщение  analysisWindow.IntermediateResult.Text += "\nНачало шифрования, входная строка: "+TextToEncrypt;  StringBuilder stringBuilder= new StringBuilder();  for (int i = 0; i < TextToEncrypt.Length; i++)  for (int k = 0; k < InitialAlphabet.Length; k++)  if (TextToEncrypt[i] == InitialAlphabet[k])  stringBuilder.Append(FinalAlphabet[k]);  analysisWindow.IntermediateResult.Text += "\nСтрока после шифрования "+stringBuilder.ToString();  return stringBuilder.ToString();  }  private string AddKeysToMessage(string TextForAddKeys)  {  //добавляем ключи в сообщение  analysisWindow.IntermediateResult.Text += "\nДобавлении ключей в строку...";  Random rnd = new Random();  StringBuilder stringBuilder = new StringBuilder();  stringBuilder.Append(TextForAddKeys);  string str = (PrefixKit[rnd.Next(0, PrefixKit.Length-1)]) + FirstKey + "|" + SecondKey+ (PrefixKit[rnd.Next(0, PrefixKit.Length - 1)]);  stringBuilder.Insert(rnd.Next(0,stringBuilder.Length-1),str);  analysisWindow.IntermediateResult.Text += "\nрезультат добавления ключей в строку "+stringBuilder.ToString()+"\n";  stopwatch.Stop();  analysisWindow.IntermediateResult.Text += "Время затраченное на зашифрование информации: "+stopwatch.Elapsed.ToString()+"\n";  return stringBuilder.ToString();  }  private void MixAlphabet()  {  //создаем новый алфавит  analysisWindow.IntermediateResult.Text += "\nНачало создание нового алфавита...";  StringBuilder stringBuilder = new StringBuilder();  string mas = InitialAlphabet;  for (int i = 0; i < InitialAlphabet.Length; i++)  {  int k = MathNewPositionOfLetter(i, mas.Length);  stringBuilder.Append(mas[k]);  mas = mas.Remove(k, 1);  }  analysisWindow.IntermediateResult.Text += "\nНовый алфавит: " + stringBuilder.ToString();  FinalAlphabet = stringBuilder.ToString();  }  private void CreateKeys(DateTime time)  {  //создание ключей, берем день в году и час  FirstKey = time.Hour;  SecondKey = time.DayOfYear;  analysisWindow.IntermediateResult.Text += "Ключ 1: " +FirstKey+" Ключ 2: " +SecondKey;  }  private DateTime TakeTimeZone(string TimeZone)  {  //берем часовой пояс  TimeZoneInfo time = TimeZoneInfo.FindSystemTimeZoneById(TimeZone);  DateTime dateTime\_Indian = TimeZoneInfo.ConvertTimeFromUtc(DateTime.UtcNow, time);  return dateTime\_Indian;  }  private bool IsKeysValid()  {  //проверяем чтобы ключи не были равны нулю  if (FirstKey == 0)  {  analysisWindow.IntermediateResult.Text += "\nПервый ключ равен 0, добавление 1";  FirstKey += 1;  }  if (SecondKey == 0)  {  analysisWindow.IntermediateResult.Text += "\nВторой ключ равен 0, добавление 1";  SecondKey += 1;  }  return true;  }  private int MathNewPositionOfLetter(int IndexOfLetter,int CountLetters)  {  //рассчитываем новый индекс  return (((FirstKey \* IndexOfLetter + (SecondKey + IndexOfLetter) \* FirstKey)) % CountLetters);  }  }} |

# Приложение Б

**Класс дешифрования текста**

|  |
| --- |
| using System;  using System.Diagnostics;  using System.Text;  using System.Text.RegularExpressions;  using System.Windows;  using static System.Net.Mime.MediaTypeNames;  namespace ModifiedCaesarCipher  {  internal class Decryptor  {  AnalysisWindow analysisWindow = new AnalysisWindow();  Stopwatch stopwatch = new Stopwatch();  private int FirstKey, SecondKey;  private string InitialAlphabet = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯабвгдеёжзийклмнопрстуфхцчшщъыьэюя 1234567890!@#$%^&\*()\_+=№;%:?-{}[]";  private string[] PrefixKit = { "Выц", "ДнП", "!в%", "Ф9\*", "СУ\_", "ЮьЕ", "\_П;", "ЯЦ&" };  private string FinalAlphabet;  private string MessageForDecrypt;  public string StartDecrypt(string InputText)  {  analysisWindow.IntermediateResult.Text += "\nНовое Деифрование\n";  analysisWindow.Hide();  analysisWindow.Show();  stopwatch.Start();  CreateKeys(InputText);  if (IsKeysValid())  MixAlphabet();  return Dencrypt(MessageForDecrypt);  }  private string Dencrypt(string TextToDencrypt)  {  //дешифрование сообщения  analysisWindow.IntermediateResult.Text += "\nШифрованный текст: "+TextToDencrypt;  StringBuilder stringBuilder = new StringBuilder();  for (int i = 0; i < TextToDencrypt.Length; i++)  for (int k = 0; k < FinalAlphabet.Length; k++)  if (TextToDencrypt[i] == FinalAlphabet[k])  stringBuilder.Append(InitialAlphabet[k]);  analysisWindow.IntermediateResult.Text += "\nТекст после расшифровки: "+stringBuilder.ToString()+"\n";  stopwatch.Stop();  analysisWindow.IntermediateResult.Text += "Время затраченное на дешифрование информации: " + stopwatch.Elapsed.ToString()+"\n";  return stringBuilder.ToString();  }  public void CreateKeys(string TextWithKeys)  {  //получаем ключи из сообщения отделяя их по префиксам  analysisWindow.IntermediateResult.Text = "Начало получения ключей из сообщения...";  string[] str=new string[2],newStr = new string[2], keys = new string[2];  for (int i = 0; i < PrefixKit.Length; i++)  {  str = TextWithKeys.Split(PrefixKit[i]);  if(str.Length >= 2)  if (str[0] != null && str[1] != null)  break;  }  for (int k = 0; k < PrefixKit.Length; k++)  {  for (int i = 0; i < 2; i++)  {  newStr = str[i].Split(PrefixKit[k]);  if(newStr.Length >= 2)  if (newStr[0] != null && newStr[1] != null)  break;  }  if (newStr.Length >= 2)  if (newStr[0] != null && newStr[1] != null)  break;  }    for (int i = 0; i < 2; i++)  {  keys = newStr[i].Split("|");  if(keys.Length >= 2)  if (keys[0] != null && keys[1]!=null)  break;  }  FirstKey = Convert.ToInt32(keys[0]);  SecondKey = Convert.ToInt32(keys[1]);  int LengthKeys = keys[0].Length + keys[1].Length;  string str1 = TextWithKeys;  StringBuilder stringBuilder = new StringBuilder();  for (int i = 0; i <PrefixKit.Length; i++)  {  str1 = str1.Replace(PrefixKit[i],"");  }  analysisWindow.IntermediateResult.Text += "\n Ключи: Первый ключ " + FirstKey+" Второй ключ "+SecondKey;  stringBuilder.Append(str1.Replace(keys[0] + "\_" + keys[1],""));  MessageForDecrypt = stringBuilder.ToString();  }  private bool IsKeysValid()  {  //проверяем чтобы ключи не были равны нулю  if (FirstKey == 0)  FirstKey += 1;  if (SecondKey == 0)  SecondKey += 1;  return true;  }  private void MixAlphabet()  {  //создаем новый алфавит  analysisWindow.IntermediateResult.Text += "\nНачало получения алфавита для расшифровки...";  StringBuilder stringBuilder = new StringBuilder();  string mas = InitialAlphabet;  for (int i = 0; i < InitialAlphabet.Length; i++)  {  int k = MathNewPositionOfLetter(i, mas.Length);  stringBuilder.Append(mas[k]);  mas = mas.Remove(k, 1);  }  analysisWindow.IntermediateResult.Text += "\n Алфивит для расшифровки "+stringBuilder.ToString();  FinalAlphabet = stringBuilder.ToString();  }  private int MathNewPositionOfLetter(int IndexOfLetter, int CountLetters)  {//рассчитываем новый индекс  return (((FirstKey \* IndexOfLetter + (SecondKey + IndexOfLetter) \* FirstKey)) % CountLetters);  }  }  } |