МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Московский государственный технологический университет «СТАНКИН»

(ФГБОУ ВО «МГТУ «СТАНКИН»)

Институт
информационных систем
и технологий

Кафедра
информационных систем

КУРСОВАЯ РАБОТА

по дисциплине «**Объектно-ориентированное программирование**»

Тема: «**Шифрование текстовых сообщений**»

Студент
группы ИДБ-21-06 _____ **Бабурян А.М.**
подпись

Руководитель
к.т.н., доцент _____ **Разумовский А.И.**
подпись

Москва

2023 г.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Шифрование — это процесс преобразования информации (текста, данных и т.д.) в непонятный для постороннего понимания вид. Обычно для этого используются определенные алгоритмы и ключи шифрования, которые позволяют зашифровать сообщение и открыть доступ к нему только уполномоченным пользователям, которые имеют соответствующий ключ для расшифровки. В данной курсовой работе реализован шифр Цезаря и его модификации — шифры Гронсфельда и Виженера.

Целью данной курсовой работы является создания программы способной генерировать цифровые ключи и использовать их для шифрования введенного пользователем текста.

Для программной реализации продукта был выбран язык C++ в совокупности с преимуществами объектно-ориентированного подхода к программированию. ООП хорошо подходит для разработки базы данных, благодаря своим основным принципам: абстракции, инкапсуляции и полиморфизма.

ГЛАВА 1. СРЕДА И СРЕДСТВА РАЗРАБОТКИ

1.1 ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Объектно-ориентированное программирование (ООП) — это методология разработки программ, основанная на использовании объектов, которые являются основными компонентами программы. Каждый объект содержит данные (поля) и функции (методы) для работы с этими данными. ООП позволяет реализовывать сложные программные системы, разбивая их на более мелкие и логически связанные части — объекты. Это облегчает понимание и поддержку программы, а также повышает ее модульность, гибкость и переиспользуемость кода. ООП также предоставляет возможности для создания и использования наследуемых свойств и методов, абстракции и полиморфизма, что открывает дополнительные возможности для создания более эффективных и гибких программных решений.

Принципы ООП:

- Инкапсуляция: возможность скрыть внутреннюю реализацию класса и позволить взаимодействовать с ним только через определенные интерфейсы (методы и свойства класса), что обеспечивает безопасность и увеличивает уровень абстракции.

- Полиморфизм: возможность использования объектов различных классов с общим интерфейсом, что позволяет использовать один и тот же код для работы с объектами разного типа.

- Абстракция: возможность определения абстрактных классов, которые представляют только основные характеристики объектов, и конкретных классов, которые уже наследуют конкретные свойства и методы.

До появления ООП доминирующей моделью разработки было процедурное программирование. Но по мере того, как системы становились сложнее, процедурный подход начал пробуксовывать. Сопровождение и развитие кода стало занимать очень много времени. А все из-за того, что процедуры не позволяли в должной мере отделить компоненты системы друг от друга; изменение одних процедур влияло на поведение других. Для решения данной проблемы придумали объектно-ориентированное программирование.

Одна из основных задач программиста — это борьба со сложностью, гораздо удобней при описании возможностей будущей системы говорить в терминах существующих вещей, а не просто вырванными из контекста кусками кода. Объектный подход позволяет разделить программу на независимые и изолированные компоненты. И изменение одних никак не влияет на поведение других. Класс позволяет концентрироваться на отдельной части системы, понимать и работать с ней, уменьшая общую сложность задачи.

ООП предоставляет новый уровень абстракции, что позволяет оперировать понятиями из реального мира, которые более привычны и понятны для большинства людей.

На сегодняшний день существуют более 2500 языков программирования высокого уровня. Это объясняется направленностью конкретных языков на определенные предметные области, а также тем, что появление новых языков дает возможность разработчикам решать все более сложные задачи. В наше время количество прикладных языков программирования, реализующих парадигму ООП превышает количество языков, реализующих иные парадигмы. Наиболее популярные объектно-ориентированные языки программирования — C++, Delphi, C#, Java,Python, Ruby и другие.

1.2 ЯЗЫК ПРОГРАММИРОВАНИЯ C++

C++ — компилируемый, статически типизированный язык программирования общего назначения.

Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности.

C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником — языком C, — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

Исследование алгоритмов и структур данных является одной из основ программирования, а также богатым полем элегантных технологий и сложных математических изысканий. Каждая программа зависит от алгоритмов и структур данных, но редко бывает нужно изобретать новые алгоритмы. Даже в сложной программе, например в компиляторе или Web-браузере, структуры данных по большей части являются массивами, списками, деревьями и хэш-таблицами. Когда программе нужна более изощренная структура, будет основываться на этих более простых структурах.

По сути, это способы хранить и организовывать данные, для более эффективного решения конкретных задач. Данные можно представить поразному. В зависимости от того, что это за данные и что вы собираетесь с ними делать, одно представление подойдёт лучше других. Вы выбираете самую подходящую структуру, основываясь на данных и на том, как они будут обрабатываться.

В языке программирования C++ термин Стандартная Библиотека означает коллекцию классов и функций, написанных на базовом языке. Стандартная Библиотека поддерживает несколько основных контейнеров, функций для работы с этими контейнерами, объектов-функции, основных типов строк и

потоков (включая интерактивный и файловый ввод-вывод), поддержку некоторых языковых особенностей, и часто используемые функции для выполнения таких задач, как, например, нахождение квадратного корня числа. Стандартная Библиотека языка C++ также включает в себя спецификации стандарта ISO C90 стандартной библиотеки языка Си.

Функциональные особенности Стандартной Библиотеки объявляются внутри пространства имен std. Наибольшей частью стандартной библиотеки C++ является библиотека STL (Standard Template Library – Стандартная Библиотека Шаблонов).

Библиотека STL содержит пять основных видов компонентов:

- контейнер (container): управляет набором объектов в памяти.
- итератор (iterator): обеспечивает для алгоритма средство доступа к содержимому контейнера.
- алгоритм (algorithm): определяет вычислительную процедуру.
- функциональный объект (function object): инкапсулирует функцию в объекте для использования другими компонентами.

Контейнеры библиотеки STL можно разделить на четыре категории: последовательные, ассоциативные, контейнеры-адаптеры и псевдоконтейнеры. На пример по два контейнера каждого типа: vector и list, set и map,stack и queue, bitset и valarray. В контейнерах для хранения элементов используется семантика передачи объектов по значению. Другими словами, при добавлении контейнер получает копию элемента. Если создание копии нежелательно, то используют контейнер указателей на элементы. Присвоение элементов реализуется с помощью оператора присваивания, а их уничтожение происходит с использованием деструктора.

В библиотеке STL для доступа к элементам в качестве посредника используется обобщённая абстракция, именуемая итератором. Каждый контейнер поддерживает «свой» вид итератора, который представляет собой «модернизированный» интеллектуальный указатель, «знающий» как получить доступ к элементам конкретного контейнера. Стандарт C++ определяет пять

категорий итераторов: входные, выходные, однонаправленные, двунаправленные и произвольного доступа. STL — кроссплатформенная библиотека.

1.3 СРЕДА РАЗРАБОТКИ VISUAL STUDIO

Для разработки программного продукта использовалась интегрированная среда разработки Visual Studio, так как она обладает всем необходимым функционалом для разработки полноценного современного приложения с использованием всех современных технологий.

Microsoft Visual Studio — это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Интегрированная среда разработки (IDE) представляет собой многофункциональную программу, которую можно использовать для различных аспектов разработки программного обеспечения [2].

Функциональность Visual Studio охватывает все этапы разработки программного обеспечения, предоставляя современные инструменты для написания кода, проектирования графических интерфейсов, сборки, отладки и тестирования приложений. Возможности Visual Studio могут быть дополнены путем подключения необходимых расширений.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно ориентированных языках программирования)

или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

Далее перечислены некоторые популярные возможности Visual Studio, которые помогут вам повысить продуктивность разработки программного обеспечения:

1. Рефакторинг

Рефакторинг включает в себя такие операции, как интеллектуальное переименование переменных, извлечение одной или нескольких строк кода в новый метод, изменение порядка параметров методов и многое другое.

2. IntelliSense

IntelliSense — это набор функций, отображающих сведения о коде непосредственно в редакторе и в некоторых случаях автоматически создающих небольшие отрывки кода. По сути, это базовая документация, встроенная в редактор, с которой вам не приходится искать информацию гдето еще. Функции IntelliSense зависят от языка. Примеры функций доступные для языка C++:

- Волнистые линии и быстрые действия.

  Волнистые линии обозначают ошибки или потенциальные проблемы кода прямо во время ввода. Эти визуальные подсказки позволяют устранять проблемы немедленно и не ждать, пока ошибка будет обнаружена во время сборки или запуска программы. Если навести указатель мыши на волнистую линию, на экран будут выведены дополнительные сведения об ошибке.
- Поиск

  Среда Visual Studio может показаться сложной, ведь там столько разных меню, параметров и свойств. Чтобы быстро находить функции интегрированной среды разработки и элементы кода, в Visual Studio представлен единый компонент поиска (CTRL+Q).
- Live Share

Предоставляет возможности совместного редактирования и отладки в реальном времени независимо от типа приложения или языка программирования. Вы можете мгновенно и безопасно поделиться своим проектом и, при необходимости, сеансами отладки, экземплярами терминалов, веб-приложениями, голосовыми звонками и многим другим.

- Иерархия вызовов

  В окне Иерархия вызовов показаны методы, вызывающие выбранный метод. Это может быть полезно, если вы собираетесь изменить или удалить метод или хотите отследить ошибку.

- CodeLens

  CodeLens помогает находить ссылки на код, изменения кода, связанные ошибки, рабочие элементы, проверки кода и модульные тесты — все это, не выходя из редактора [4].

Основным расширением файла, ассоциированным с Microsoft Visual Studio, является SLN – Visual Studio Solution File (Файл решения Visual Studio), при открытии которого в программу загружаются все данные и проекты, связанные с разрабатываемым программным решением.

1.4 WINDOWS FORMS И .NET FRAMEWORK

Windows Forms — это технология интеллектуальных клиентов для NET Framework. Она представляет собой набор управляемых библиотек, упрощающих выполнение стандартных задач, таких как чтение из файловой системы и запись в нее. С помощью такой среды разработки, как Visual Studio, можно создавать интеллектуальные клиентские приложения Windows Forms, которые отображают информацию, запрашивают ввод от пользователей и обмениваются данными с удаленными компьютерами по сети.

Windows Forms позволяет разрабатывать интеллектуальные клиенты. Интеллектуальный клиент — это приложение с полнофункциональным графическим интерфейсом, простое в развертывании и обновлении, способное

работать при наличии или отсутствии подключения к Интернету и использующее более безопасный доступ к ресурсам на локальном компьютере по сравнению с традиционными приложениями Windows.

В Windows Forms форма — это визуальная поверхность, на которой выводится информация для пользователя. Обычно приложение Windows Forms строится путем помещения элементов управления на форму и написания кода для реагирования на действия пользователя, такие как щелчки мыши или нажатия клавиш. Элемент управления — это отдельный элемент пользовательского интерфейса, предназначенный для отображения или ввода данных.

При выполнении пользователем какого-либо действия с формой или одним из ее элементов управления создается событие. Приложение реагирует на эти события с помощью кода и обрабатывает события при их возникновении. Подробнее см. в разделе Создание обработчиков событий в Windows Forms.

Windows Forms включает широкий набор элементов управления, которые можно добавлять на формы: текстовые поля, кнопки, раскрывающиеся списки, переключатели и даже веб-страницы. Список всех элементов управления, которые можно использовать в форме, представлены в разделе Элементы управления для использования в формах Windows Forms. Если существующий элемент управления не удовлетворяет потребностям, в Windows Forms можно создать пользовательские элементы управления с помощью класса UserControl.

В состав Windows Forms входят многофункциональные элементы пользовательского интерфейса, позволяющие воссоздавать возможности таких сложных приложений, как Microsoft Office. Используя элементы управления ToolStrip и MenuStrip, можно создавать панели инструментов и меню, содержащие текст и рисунки, подменю и другие элементы управления, такие как текстовые поля и поля со списками.

Используя функцию перетаскивания конструктора Windows Forms в Visual Studio, можно легко создавать приложения Windows Forms. Достаточно выделить элемент управления курсором и поместить его в нужное место на форме. Для преодоления трудностей, связанных с выравниванием элементов

управления, конструктор предоставляет такие средства, как линии сетки и линии привязки. И при использовании Visual Studio, и при компиляции из командной строки вы можете использовать элементы управления FlowLayoutPanel, TableLayoutPanel и SplitContainer для создания сложных макетов форм за меньшее время.

Наконец, если нужно создать свои собственные элементы пользовательского интерфейса, пространство имен System.Drawing содержит широкий набор классов, необходимых для отрисовки линий, кругов и других фигур непосредственно на форме. [1]

ГЛАВА 2. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

2.1 ПРИНЦИП РАБОТЫ ПРОГРАММЫ

Данный программный продукт позволяет пользователю зашифровать текст шифром Цезаря или его модификациями (шифры Гронсфельда и Виженера).

Шифр Цезаря — это вид шифра подстановки, в котором каждый символ в открытом тексте заменяется символом, находящимся на некотором постоянном числе позиций левее или правее него в алфавите. Например, в шифре со сдвигом вправо на 3, А была бы заменена на Г, Б станет Д, и так далее.

Шифр Гронсфельда представляет собой модификацию шифра Цезаря числовым ключом. Для этого под буквами исходного сообщения записывают цифры числового ключа. Если ключ короче сообщения, то его запись циклически повторяют. Шифр-текст получают примерно, как в шифре Цезаря, но отсчитывают по алфавиту не третью букву, а выбирают ту букву, которая смещена по алфавиту на соответствующую цифру ключа.

Принцип шифра Виженера заключается в том, что каждая буква в исходном шифруемом тексте сдвигается по алфавиту не на фиксированное, а переменное количество символов. Величина сдвига каждой буквы задается ключом (паролем) — секретным словом или фразой, которая используется для

шифрования и расшифровки. Этот шифр идентичен шифру Гронсфельда, но вместо ряда ключей используется кодовое слово, которое позже будет преобразовано в набор шагов для сдвига подобно методу Гронсфельда. Выбирая способ шифрования Виженера, пользователь должен будет сам ввести ключевое слово.

2.2 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРОДУКТА

Весь функционал программы был помещен в DLL (англ. Dynamic Link Library — «библиотека динамической компоновки», «динамически подключаемая библиотека») — динамическая библиотека, позволяющая многократное использование различными программными приложениями [1].

В Windows библиотека динамической компоновки является исполняемым файлом, который выступает в качестве общей библиотеки функций и ресурсов. Она позволяет исполняемому файлу вызывать функции или использовать ресурсы, хранящиеся в отдельном файле. Библиотека DLL не является отдельным исполняемым файлом. Библиотеки DLL выполняются в контексте приложений, которые их вызывают.

Преимущества DLL включают в себя:

- экономия памяти;
- экономия места на диске;
- увеличение пропускной способности;
- сокращение подкачки;
- универсальность;
- многопримениемость.

Интерфейс программы был реализован с помощью платформы Windows Forms для C++/CLI .NET framework.

Рассмотрим полиморфную иерархию классов в нашем программном продукте. Базовый класс содержит:

1. Объект print_key типа string.
2. Алфавиты кириллицы и латиницы, реализованные через vector<char>.
3. Конструктор
4. Конструктор копирования
5. Виртуальный метод copy, который вернет указатель на класс
6. Виртуальный метод Cipher, который вернет зашифрованное сообщение.
7. Виртуальный деструктор.

Код из файла dll.h:

```
class DLL_API Caesar: public Base
{
private:
  int key;
  void enc(vector<char>& alf, int key, char& c);
public:
  Caesar();
  string Cipher(string ins);
  Caesar(const Caesar& c);
  Caesar* copy();
  ~Caesar();
};

class DLL_API Gronsfeld: public Base
{
  vector<int> key_generate();
  vector<int> keys;
  void enc(vector<char>& alf, vector<int>& keys, char& c, int& ind);
public:
  Gronsfeld();
  string Cipher(string ins);
  Gronsfeld(const Gronsfeld& g);
  Gronsfeld* copy();
  ~Gronsfeld();
};

class DLL_API Vigenere: public Base
{
  string key_word;
  void enc(vector<char>& alf, char& c, mit& it, mit& sit, mit& lit);
public:
  Vigenere();
  string Cipher(string ins);
```

```
  Vigenere(const Vigenere& v);
  Vigenere* copy();
  ~Vigenere();
};
```

Также был реализован класс DB, содержащий в себе вектор <Base*>, по умолчанию в нем хранятся три объекта, указатели на разные способы шифрования.

Объявление класса DB в файле dll.h:

```
class DLL_API DB
{
  vector<Base*> v;
public:
  DB();
  Base* operator[](string s);
  DB(const DB& db);
  DB& operator=(const DB& db);
  ~DB();
};
```

Реализация классов в файле dll.cpp:

```
Base::Base()
{
    lat_down =
{ 'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z' };
    lat_up =
{ 'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z'
};
    kir_down =
{ 'а','б','в','г','д','е','ё','ж','з','и','й','к','л','м','н','о','п','р','с','т','у','ф','х','ц','ч','ш','щ','ъ','ы','
ь','э','ю' };
```

```
    kir_up =
{ 'А','Б','В','Г','Д','Е','Ё','Ж','З','И','Й','К','Л','М','Н','О','П','Р','С','Т','У','Ф','Х','Ц','Ч','Ш
','Щ','Ъ','Ы','Ь','Э','Ю' };
}

Base::Base(const Base& m)
{
    print_key = m.print_key;
    lat_down = m.lat_down;
    lat_up = m.lat_up;
    kir_down = m.kir_down;
    kir_up = m.kir_up;
}
Base::~Base(){}
Caesar::Caesar()
{
    key = 0;
}
void Caesar::enc(vector<char>& alf, int key, char& c)
{
    vector<char>::iterator it = find(alf.begin(), alf.end(), c);
    int i = distance(alf.begin(), it) + key;
    c = alf[i % alf.size()];
}
string Caesar::Cipher(string ins)
{
    key = rand() % 26;
```

```
    for (char& c : ins)
    {
        if (binary_search(lat_down.begin(), lat_down.end(), c))
            enc(lat_down, key, c);

        else if (binary_search(lat_up.begin(), lat_up.end(), c))
            enc(lat_up, key, c);

        else if (binary_search(kir_down.begin(), kir_down.end(), c))
            enc(kir_down, key, c);

        else if (binary_search(kir_up.begin(), kir_up.end(), c))
            enc(kir_down, key, c);
    }

    print_key = to_string(key);
    return ins;
}
Caesar::Caesar(const Caesar& c)
{
    key = c.key;
}
Caesar* Caesar::copy()
{
    return new Caesar(*this);
}
Caesar::~Caesar(){}
```

```
Vigenere::Vigenere(){}

int index(char s, vector<char>& alf)
{
   vector<char>::iterator it = find(alf.begin(), alf.end(), s);
   int i = distance(alf.begin(), it);
   return i;
};

void Vigenere::enc(vector<char>& alf, char& c, mit& it, mit& sit, mit& lit)
{
   vector<char>::iterator iter = find(alf.begin(), alf.end(), c);
   int t = distance(alf.begin(), iter);
   c = alf[(t + it->first) % alf.size()];
   it++;
   if (it == lit)
      it = sit;
}

string Vigenere::Cipher(string ins)
{
   unordered_multimap <int, char> key;
   for (char& c : print_key)
   {
      if (binary_search(lat_down.begin(), lat_down.end(), c))
         key.insert({ index(c, lat_down), c});
```

```
    else if (binary_search(lat_up.begin(), lat_up.end(), c))
        key.insert({ index(c, lat_up), c });


    else if (binary_search(kir_down.begin(), kir_down.end(), c))
        key.insert({ index(c, kir_down), c });

    else if (binary_search(kir_up.begin(), kir_up.end(), c))
        key.insert({ index(c, kir_up), c });
}

mit it = key.begin(), sit = key.begin(), lit = key.end();

for (char& c : ins)
{
    if (binary_search(lat_down.begin(), lat_down.end(), c))
        enc(lat_down, c, it, sit, lit);

    else if (binary_search(lat_up.begin(), lat_up.end(), c))
        enc(lat_up, c, it, sit, lit);

    else if (binary_search(kir_down.begin(), kir_down.end(), c))
        enc(kir_down, c, it, sit, lit);

    else if (binary_search(kir_up.begin(), kir_up.end(), c))
        enc(kir_up, c, it, sit, lit);
```

```
    }

    return ins;
}

Vigenere::Vigenere(const Vigenere& v) : Base(v)
{
    key_word = v.key_word;
}
Vigenere* Vigenere::copy()
{
    return new Vigenere(*this);
}
Vigenere::~Vigenere() { }
vector<int> Gronsfeld::key_generate()
{
    vector<int> keys_1(rand() % 100), keys_2(rand() % 100);
    for_each(keys_1.begin(), keys_1.end(), [](int& i) {i = 1 + rand() % 26; });
    for_each(keys_2.begin(), keys_2.end(), [](int& i) {i = 1 + rand() % 26; });

    sort(keys_1.begin(), keys_1.end());
    sort(keys_2.begin(), keys_2.end());
    pair<vector<int>::iterator, vector<int>::iterator> it;

    vector<int> keys_intersection;

    set_intersection(
```

```
        keys_1.begin(), keys_1.end(),
        keys_2.begin(), keys_2.end(),
        back_inserter(keys_intersection)
    );

    sort(keys_intersection.begin(), keys_intersection.end());

    vector<int> final_keys;
    for (int i = 0; i < 3 + rand() % 5; i++)
    {
        it = equal_range(keys_1.begin(), keys_1.end(), 1 + rand() % 26, [](int i, int j)
{ return (i > j); });

        final_keys.push_back(it.second - it.first + rand() % 10);
    }
    random_shuffle(final_keys.begin(), final_keys.end());

    return final_keys;
}

void Gronsfeld::enc(vector<char>& alf, vector<int>& keys, char& c, int& ind)
{
    vector<char>::iterator it = find(alf.begin(), alf.end(), c);
    int t = distance(alf.begin(), it);
    c = alf[(t + keys[ind % keys.size()]) % alf.size()];
    ind++;
}
```

```
Gronsfeld::Gronsfeld(){}



string Gronsfeld::Cipher(string ins)
{
    keys = key_generate();
    int i = 0;
    for (char& c : ins)
    {
        if (binary_search(lat_down.begin(), lat_down.end(), c))
            enc(lat_down, keys, c, i);
        else if (binary_search(lat_up.begin(), lat_up.end(), c))
            enc(lat_up, keys, c, i);
        else if (binary_search(kir_down.begin(), kir_down.end(), c))
            enc(kir_down, keys, c, i);
        else if (binary_search(kir_up.begin(), kir_up.end(), c))
            enc(kir_up, keys, c, i);
    }
    print_key = "( ";

    for (int i : keys)
        print_key += to_string(i) + ' ';

    print_key += ')';
```

```
    return ins;
}


Gronsfeld::Gronsfeld(const Gronsfeld& g): Base(g)
{
    keys = g.keys;
}

Gronsfeld* Gronsfeld::copy()
{
    return new Gronsfeld(*this);
}

Gronsfeld::~Gronsfeld(){}

DB::DB()
{
    v.push_back(new Caesar);
    v.push_back(new Gronsfeld);
    v.push_back(new Vigenere);
}

Base* DB::operator[](string s)
{
    if (s == "Шифр Цезаря")
```

```
        return v[0];

    if (s == "Шифр Гронсфельда")
        return v[1];

    if (s == "Шифр Виженера")
        return v[2];
}



DB::DB(const DB& db)
{
    for (vector<Base*>::iterator it = v.begin(); it != v.end(); it++)
        delete* it;
    v.clear();

    for (vector<Base*>::const_iterator it = db.v.begin(); it != db.v.end(); it++)
    {
        Base* new_obj = (*it)->copy();
        v.push_back(new_obj);
    }
}

DB& DB::operator=(const DB& db)
{
    if (this == &db)
```

```
        return *this;

    for (vector<Base*>::iterator it = v.begin(); it != v.end(); it++)
        delete* it;
    v.clear();

    for (vector<Base*>::const_iterator it = db.v.begin(); it != db.v.end(); it++)
    {
        Base* new_obj = (*it)->copy();
        v.push_back(new_obj);
    }

    return *this;
}

DB::~DB()
{
    for (int i = 0; i < v.size(); i++)
        delete v[i];
    v.clear();
}
```

2.3 ВЗАИМОДЕЙСТВИЕ С ПОЛЬЗОВАТЕЛЕМ

Пользователь взаимодействует с программным средством через визуальный интерфейс. Сначала пользователю предлагается ввести текст и выбрать вид шифрования. Окно представлено на рис. 1.

При выборе шифра Цезаря и шифра Гронсфельда программа автоматически генерирует ключи и сразу выводит зашифрованный текст примеры работы обоих методов представлены на рис. 2 и рис. 3 соответственно.

Если был выбран шифр Виженера, то. Программа предложит ввести ключевое слово в отдельном окне, после чего выведет зашифрованное сообщение в том же текстовом блоке, что и для предыдущих методов шифровки. Пример работы шифра Виженера представлены на рис. 4 - 6.

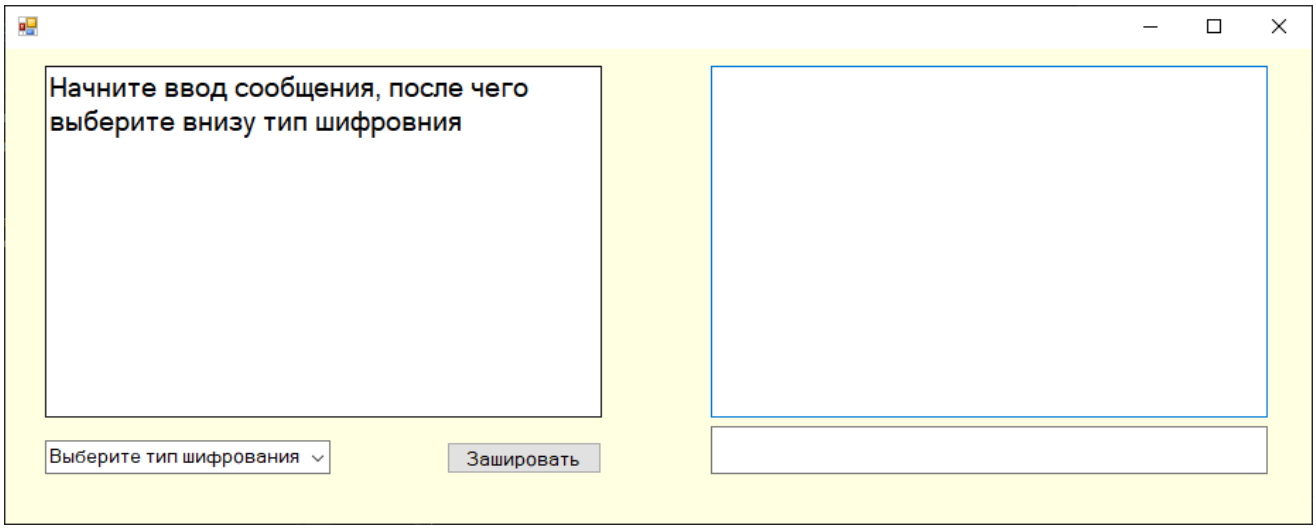

Рис. 1 Начальное Окно

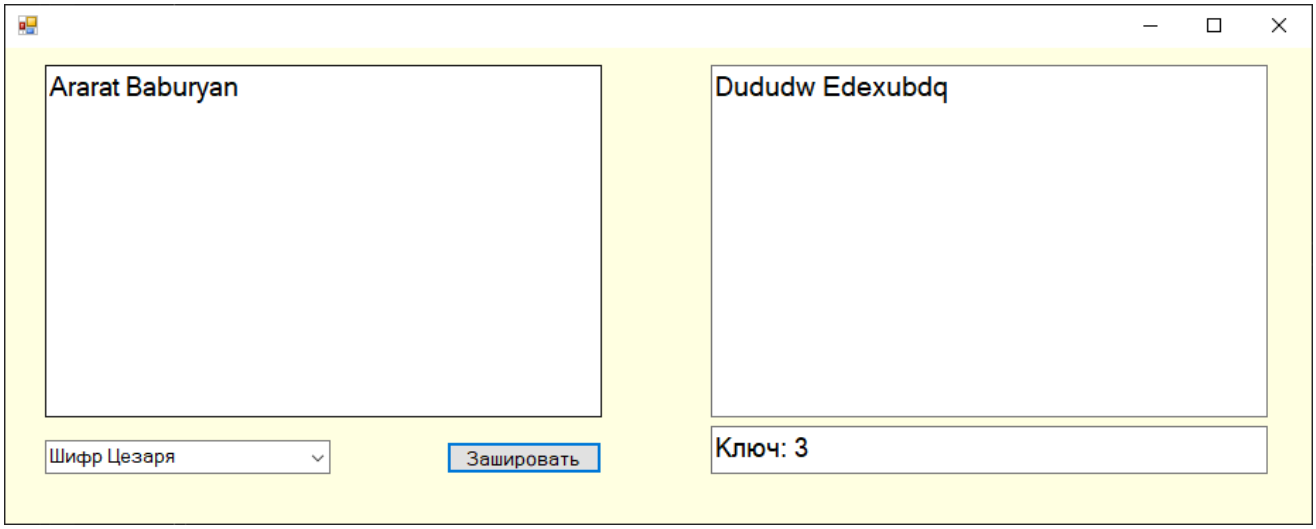

Рис. 2 Пример работы Шифра Цезаря

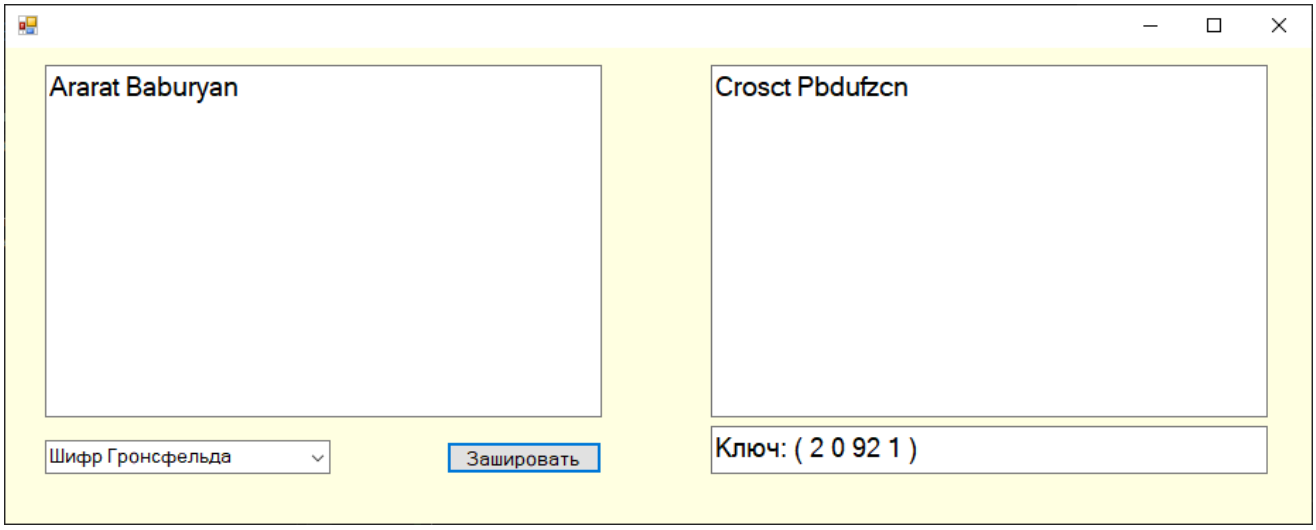

Рис. 3 Пример работы Шифра Гронсфельда

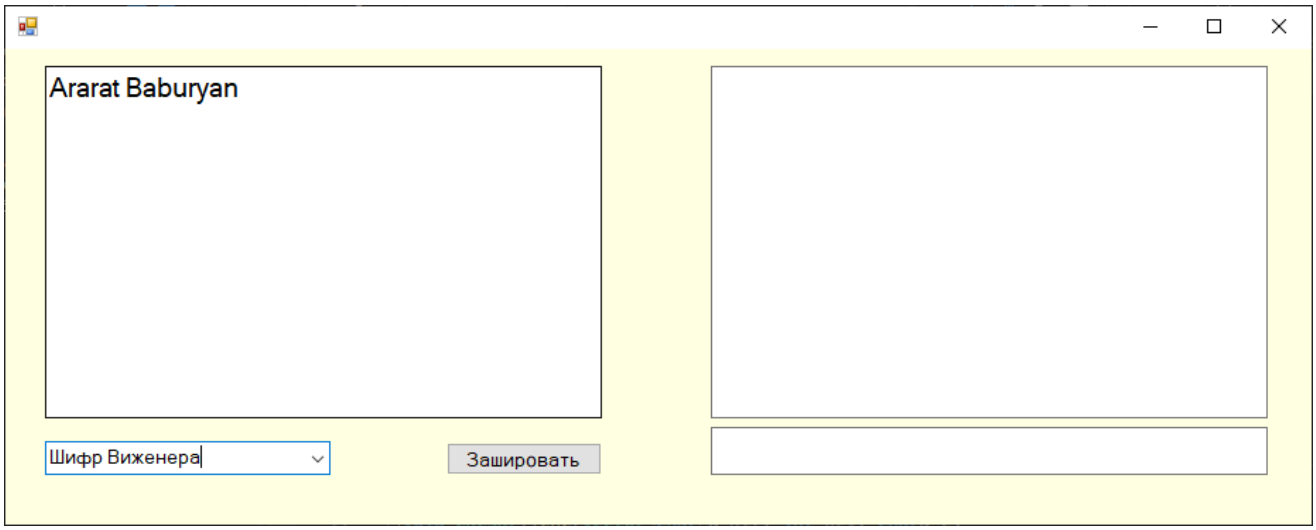

Рис. 4 Ввод сообщения и выбор Шифра Виженера

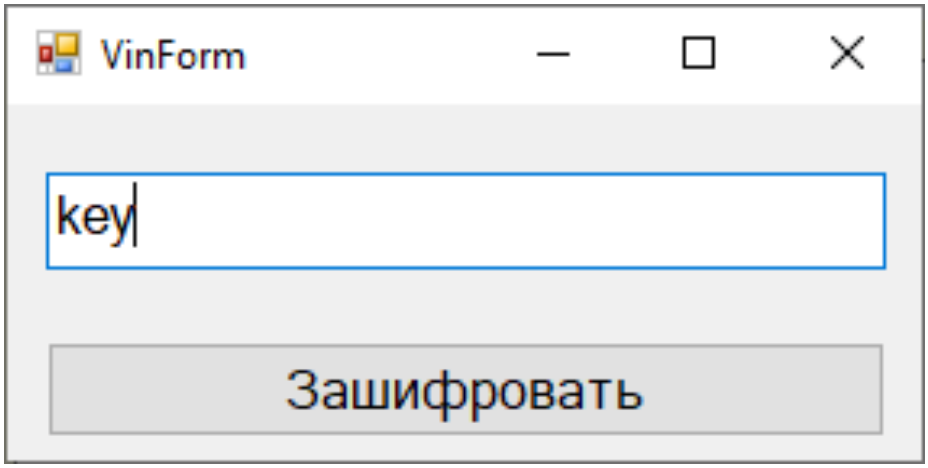

Рис. 5 Ввод ключа для Шифра Виженера

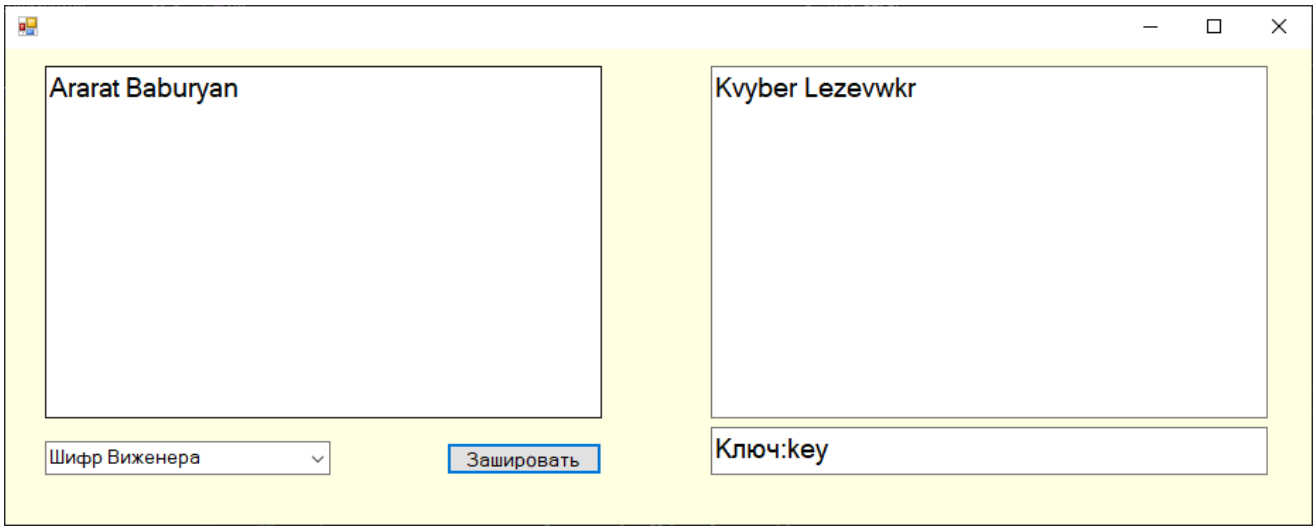

Рис. 6 Пример работы Шифра Виженера

ЗАКЛЮЧЕНИЕ

В данной работе мы разработали программу, способную генерировать цифровые ключи и использовать их для шифрования введенного пользователем текста. При разработки данного программного продукта были использованы принципы ООП и технологии C++/CLI .NET framework. Несмотря на свои скромные размеры, благодаря применению объектно-ориентированного подхода к проектированию, функционал данной программы всегда можно расширить, прибегая к минимальным изменениям существующего кода.

Использование DLL библиотеки позволяет более гибко разрабатывать и расширять интерфейс программы. Таким образом мы на практике научились проектировать и реализовывать программные продукты с использованием объектно-ориентированного подхода, изучили особенности работы с DLL библиотеками и разработали современный интерфейс на платформе .NET.

СПИСОК ЛИТЕРАТУРЫ

1. Официальный сайт Microsoft [Электронный ресурс] Режим доступа: http://docs.microsoft.com

2. STL: стандартная библиотека шаблонов C++ [Электронный ресурс/статья] Режим доступа: https://tproger.ru/articles/stl-cpp

3. Романов С.С. Ключевые понятия и особенности объектно ориентированного программирования // Таврический научный обозреватель. 2016. №12-2 (17). URL: https://cyberleninka.ru/article/n/klyuchevye-ponyatiya-i-osobennosti obektno-orientirovannogo-programmirovaniya (дата обращения: 23.05.2022)

4. Большая российская энциклопедия [Электронный ресурс] Режим доступа: https://bigenc.ru/technology_and_technique/text/3958439

5. Официальный сайт Cppreference [Электронный ресурс] Режим доступа: https://ru.cppreference.com/w

6. Страуструп, Б. Язык программирования C++ / Б. Страуструп. - М.: Радио и связь, 2017. – 1136 с. [Бумажный ресурс]

7. Лафоре Р. Объектно-ориентированное программирование в C++ [Текст] / Р. Лафоре. — 4-е изд. — СПб.: Питер, 2004. — 928 с

8. Официальный сайт Wikipedia [Электронный ресурс] Режим доступа: https://ru.wikipedia.org/wiki

ПРИЛОЖЕНИЕ 1. ЛИСТИНГ КОДА

dllmain.cpp

```
// dllmain.cpp : Defines the entry point for the DLL application.
#include "framework.h"
#include <iostream>
#include <stdlib.h>
#include <time.h>

struct Leaks { ~Leaks() { _CrtDumpMemoryLeaks(); } };
Leaks _l_;

BOOL APIENTRY DllMain( HMODULE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    srand(time(NULL));
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}
```

dll.h

```
#ifdef DLL_EXPORTS
#define DLL_API __declspec(dllexport)
#else
#define DLL_API __declspec(dllimport)
#endif

#include<string>
#include<vector>
#include<algorithm>
#include<iostream>
#include<unordered_map>

using std::set_intersection;
using std::pair;
using std::random_shuffle;
using std::back_inserter;
using std::unordered_multimap;
using std::for_each;
using std::binary_search;
using std::find;
using std::distance;
using std::string;
using std::vector;
using std::to_string;
typedef unordered_multimap <int, char>::iterator mit;
```

```
extern DLL_API int open;
extern DLL_API bool flag;
extern DLL_API string KEY, text, enc_choose;

class DLL_API Base
{
public:
  string print_key;
  vector<char> lat_down, lat_up, kir_down, kir_up;
  Base();
  virtual string Cipher(string ins) = 0;
  Base(const Base& m);
  virtual Base* copy() = 0;
  virtual ~Base();
};
class DLL_API Caesar: public Base
{
private:
  int key;
  void enc(vector<char>& alf, int key, char& c);
public:
  Caesar();
  string Cipher(string ins);
  Caesar(const Caesar& c);
  Caesar* copy();
  ~Caesar();
};
```

```
class DLL_API Gronsfeld: public Base
{
  vector<int> key_generate();
  vector<int> keys;
  void enc(vector<char>& alf, vector<int>& keys, char& c, int& ind);
public:
  Gronsfeld();
  string Cipher(string ins);
  Gronsfeld(const Gronsfeld& g);
  Gronsfeld* copy();
  ~Gronsfeld();
};
class DLL_API Vigenere: public Base
{
  string key_word;
  void enc(vector<char>& alf, char& c, mit& it, mit& sit, mit& lit);
public:
  Vigenere();
  string Cipher(string ins);
  Vigenere(const Vigenere& v);
  Vigenere* copy();
  ~Vigenere();
};
```

```
class DLL_API DB
{
  vector<Base*> v;
public:
  DB();
  Base* operator[](string s);
  DB(const DB& db);
  DB& operator=(const DB& db);
  ~DB();
};
extern DLL_API DB db;
```

dll.cpp

```
#include "framework.h"
#include "dll.h"

DLL_API int open = 0;
DLL_API bool flag = false;
DLL_API DB db;
DLL_API string KEY = "Ключ: ", text = "", enc_choose = "";

Base::Base()
{
   lat_down = { 'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z' };
   lat_up =
{ 'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z' };
```

```
   kir_down =
{ 'а','б','в','г','д','е','ё','ж','з','и','й','к','л','м','н','о','п','р','с','т','у','ф','х','ц','ч','ш','щ','ъ','ы','ь','э
','ю' };
   kir_up =
{ 'А','Б','В','Г','Д','Е','Ё','Ж','З','И','Й','К','Л','М','Н','О','П','Р','С','Т','У','Ф','Х','Ц','Ч','Ш','
Щ','Ъ','Ы','Ь','Э','Ю' };
}

Base::Base(const Base& m)
{
   print_key = m.print_key;
   lat_down = m.lat_down;
   lat_up = m.lat_up;
   kir_down = m.kir_down;
   kir_up = m.kir_up;
}
Base::~Base(){}

Caesar::Caesar()
{
   key = 0;
}
void Caesar::enc(vector<char>& alf, int key, char& c)
{
   vector<char>::iterator it = find(alf.begin(), alf.end(), c);
   int i = distance(alf.begin(), it) + key;
   c = alf[i % alf.size()];
}
```

```
string Caesar::Cipher(string ins)
{
   key = rand() % 26;
   for (char& c : ins)
   {
      if (binary_search(lat_down.begin(), lat_down.end(), c))
         enc(lat_down, key, c);

      else if (binary_search(lat_up.begin(), lat_up.end(), c))
         enc(lat_up, key, c);

      else if (binary_search(kir_down.begin(), kir_down.end(), c))
         enc(kir_down, key, c);

      else if (binary_search(kir_up.begin(), kir_up.end(), c))
         enc(kir_down, key, c);
   }
   print_key = to_string(key);

   return ins;
}
Caesar::Caesar(const Caesar& c)
{
   key = c.key;
}
```

```
Caesar* Caesar::copy()
{
   return new Caesar(*this);
}
Caesar::~Caesar(){}
Vigenere::Vigenere(){}
int index(char s, vector<char>& alf)
{
   vector<char>::iterator it = find(alf.begin(), alf.end(), s);
   int i = distance(alf.begin(), it);
   return i;
};

void Vigenere::enc(vector<char>& alf, char& c, mit& it, mit& sit, mit& lit)
{
   vector<char>::iterator iter = find(alf.begin(), alf.end(), c);
   int t = distance(alf.begin(), iter);
   c = alf[(t + it->first) % alf.size()];
   it++;
   if (it == lit)
      it = sit;
}

string Vigenere::Cipher(string ins)
{
   unordered_multimap <int, char> key;
   for (char& c : print_key)
```

```
{
    if (binary_search(lat_down.begin(), lat_down.end(), c))
        key.insert({ index(c, lat_down), c});

    else if (binary_search(lat_up.begin(), lat_up.end(), c))
        key.insert({ index(c, lat_up), c });

    else if (binary_search(kir_down.begin(), kir_down.end(), c))
        key.insert({ index(c, kir_down), c });

    else if (binary_search(kir_up.begin(), kir_up.end(), c))
        key.insert({ index(c, kir_up), c });
}

mit it = key.begin(), sit = key.begin(), lit = key.end();

for (char& c : ins)
{
    if (binary_search(lat_down.begin(), lat_down.end(), c))
        enc(lat_down, c, it, sit, lit);

    else if (binary_search(lat_up.begin(), lat_up.end(), c))
        enc(lat_up, c, it, sit, lit);

    else if (binary_search(kir_down.begin(), kir_down.end(), c))
        enc(kir_down, c, it, sit, lit);
```

```
        else if (binary_search(kir_up.begin(), kir_up.end(), c))
            enc(kir_up, c, it, sit, lit);
    }

    return ins;
}

Vigenere::Vigenere(const Vigenere& v) : Base(v)
{
    key_word = v.key_word;
}

Vigenere* Vigenere::copy()
{
    return new Vigenere(*this);
}

Vigenere::~Vigenere() { }

vector<int> Gronsfeld::key_generate()
{

    vector<int> keys_1(rand() % 100),
                keys_2(rand() % 100);

    for_each(keys_1.begin(), keys_1.end(), [](int& i) {i = 1 + rand() % 26; });
    for_each(keys_2.begin(), keys_2.end(), [](int& i) {i = 1 + rand() % 26; });
```

```
    sort(keys_1.begin(), keys_1.end());
    sort(keys_2.begin(), keys_2.end());
    pair<vector<int>::iterator, vector<int>::iterator> it;

    vector<int> keys_intersection;
    set_intersection(
        keys_1.begin(), keys_1.end(),
        keys_2.begin(), keys_2.end(),
        back_inserter(keys_intersection)
    );

    sort(keys_intersection.begin(), keys_intersection.end());

    vector<int> final_keys;

    for (int i = 0; i < 3 + rand() % 5; i++)
    {
        it = equal_range(keys_1.begin(), keys_1.end(), 1 + rand() % 26, [](int i, int j)
{ return (i > j); });
        final_keys.push_back(it.second - it.first + rand() % 10);
    }

    random_shuffle(final_keys.begin(), final_keys.end());

    return final_keys;
}
```

```
void Gronsfeld::enc(vector<char>& alf, vector<int>& keys, char& c, int& ind)
{
   vector<char>::iterator it = find(alf.begin(), alf.end(), c);
   int t = distance(alf.begin(), it);
   c = alf[(t + keys[ind % keys.size()]) % alf.size()];
   ind++;
}

Gronsfeld::Gronsfeld(){}

string Gronsfeld::Cipher(string ins)
{
   keys = key_generate();
   int i = 0;
   for (char& c : ins)
   {
      if (binary_search(lat_down.begin(), lat_down.end(), c))
         enc(lat_down, keys, c, i);
      else if (binary_search(lat_up.begin(), lat_up.end(), c))
         enc(lat_up, keys, c, i);
      else if (binary_search(kir_down.begin(), kir_down.end(), c))
         enc(kir_down, keys, c, i);
      else if (binary_search(kir_up.begin(), kir_up.end(), c))
         enc(kir_up, keys, c, i);
   }
```

```
    print_key = "( ";

    for (int i : keys)
        print_key += to_string(i) + ' ';

    print_key += ')';

    return ins;
}

Gronsfeld::Gronsfeld(const Gronsfeld& g): Base(g)
{
    keys = g.keys;
}

Gronsfeld* Gronsfeld::copy()
{
    return new Gronsfeld(*this);
}
Gronsfeld::~Gronsfeld(){}

DB::DB()
{
    v.push_back(new Caesar);
    v.push_back(new Gronsfeld);
    v.push_back(new Vigenere);
}
```

```
Base* DB::operator[](string s)
{
   if (s == "Шифр Цезаря")
      return v[0];
   if (s == "Шифр Гронсфельда")
      return v[1];
   if (s == "Шифр Виженера")
      return v[2];
}

DB::DB(const DB& db)
{
   for (vector<Base*>::iterator it = v.begin(); it != v.end(); it++)
      delete* it;
   v.clear();

   for (vector<Base*>::const_iterator it = db.v.begin(); it != db.v.end(); it++)
   {
      Base* new_obj = (*it)->copy();
      v.push_back(new_obj);
   }
}

DB& DB::operator=(const DB& db)
{
   if (this == &db)
      return *this;
```

```
    for (vector<Base*>::iterator it = v.begin(); it != v.end(); it++)
        delete* it;
    v.clear();

    for (vector<Base*>::const_iterator it = db.v.begin(); it != db.v.end(); it++)
    {
        Base* new_obj = (*it)->copy();
        v.push_back(new_obj);
    }

    return *this;
}

DB::~DB()
{
    for (int i = 0; i < v.size(); i++)
        delete v[i];
    v.clear();
}
```

FirstForm.h

```
#pragma once

#include "VinForm.h"
#include "dll/dll.h"
#pragma comment(lib, "bin\\dll.lib")
#include <msclr\marshal_cppstd.h>

namespace Cursovaya {
	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Summary for FirstForm
	/// </summary>
	public ref class FirstForm : public System::Windows::Forms::Form
	{
	public:
		FirstForm(void)
		{
			InitializeComponent();
			//
```

```
            //TODO: Add the constructor code here
            //
        }
protected:
        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        ///
        ~FirstForm()
        {
            db.~DB();
            KEY.~basic_string();
            enc_choose.~basic_string();
            text.~basic_string();
            delete components;
        }

private: System::Windows::Forms::ContextMenuStrip^ contextMenuStrip1;
private: System::Windows::Forms::TextBox^ Encryption;
private: System::Windows::Forms::TextBox^ CleanText;
private: System::Windows::Forms::ComboBox^ EncChoose;
private: System::Windows::Forms::Button^ EncBut;
private: System::Windows::Forms::TextBox^ textBox1;
private: System::ComponentModel::IContainer^ components;
private:
        /// <summary>
        /// Required designer variable.
```

```
        /// </summary>

#pragma region Windows Form Designer generated code
        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        void InitializeComponent(void)
        {
            this->components = (gcnew
System::ComponentModel::Container());
            this->Encryption = (gcnew
System::Windows::Forms::TextBox());
            this->CleanText = (gcnew System::Windows::Forms::TextBox());
            this->EncChoose = (gcnew System::Windows::Forms::Combo-
Box());
            this->EncBut = (gcnew System::Windows::Forms::Button());
            this->textBox1 = (gcnew System::Windows::Forms::TextBox());
            this->SuspendLayout();
            //
            // Encryption
            //
            this->Encryption->BackColor =
System::Drawing::SystemColors::Window;
            this->Encryption->Font = (gcnew System::Drawing::Font(L"Mi-
crosoft Sans Serif", 14.25F, System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->Encryption->Location = System::Drawing::Point(499, 12);
```

```
            this->Encryption->Multiline = true;
            this->Encryption->Name = L"Encryption";
            this->Encryption->ReadOnly = true;
            this->Encryption->Size = System::Drawing::Size(394, 249);
            this->Encryption->TabIndex = 10;
            //
            // CleanText
            //
            this->CleanText->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14.25F, System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->CleanText->Location = System::Drawing::Point(28, 12);
            this->CleanText->Multiline = true;
            this->CleanText->Name = L"CleanText";
            this->CleanText->Size = System::Drawing::Size(394, 249);
            this->CleanText->TabIndex = 11;
            this->CleanText->MouseClick += gcnew
System::Windows::Forms::MouseEventHandler(this, &FirstForm::CleanText_MouseClick);
            //
            // EncChoose
            //
            this->EncChoose->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->EncChoose->FormattingEnabled = true;
```

```
this->EncChoose->Items->AddRange(gcnew cli::array< System::Object^ >(3) { L"Шифр Цезаря", L"Шифр Гронсфельда", L"Шифр Виженера" });
this->EncChoose->Location = System::Drawing::Point(28, 277);
this->EncChoose->Name = L"EncChoose";
this->EncChoose->Size = System::Drawing::Size(202, 24);
this->EncChoose->TabIndex = 14;
this->EncChoose->Text = L"Выберите тип шифрования";
//
// EncBut
//
this->EncBut->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
this->EncBut->Location = System::Drawing::Point(312, 278);
this->EncBut->Name = L"EncBut";
this->EncBut->Size = System::Drawing::Size(110, 23);
this->EncBut->TabIndex = 15;
this->EncBut->Text = L"Зашировать";
this->EncBut->UseVisualStyleBackColor = true;
this->EncBut->Click += gcnew System::EventHandler(this, &FirstForm::EncBut_Click);
//
// textBox1
//
this->textBox1->BackColor = System::Drawing::SystemColors::Window;
```

```
this->textBox1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
this->textBox1->Location = System::Drawing::Point(499, 267);
this->textBox1->Multiline = true;
this->textBox1->Name = L"textBox1";
this->textBox1->ReadOnly = true;
this->textBox1->Size = System::Drawing::Size(394, 34);
this->textBox1->TabIndex = 16;
//
// FirstForm
//
this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
this->AutoSizeMode = System::Windows::Forms::AutoSizeMode::GrowAndShrink;
this->BackColor = System::Drawing::SystemColors::Info;
this->ClientSize = System::Drawing::Size(924, 336);
this->Controls->Add(this->textBox1);
this->Controls->Add(this->EncBut);
this->Controls->Add(this->EncChoose);
this->Controls->Add(this->CleanText);
this->Controls->Add(this->Encryption);
this->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 7.875F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point, static_cast<System::Byte>(0)));
this->Location = System::Drawing::Point(10, 10);
```

```
            this->Margin = System::Windows::Forms::Padding(6);
            this->Name = L"FirstForm";
            this->Text = L"  ";
            this->Activated += gcnew System::EventHandler(this, &First-
Form::FirstForm_Activated);
            this->Load += gcnew System::EventHandler(this,
&FirstForm::FirstForm_Load);
            this->ResumeLayout(false);
            this->PerformLayout();
        }
#pragma endregion
    private:
        VinForm^ f;
        System::Void FirstForm_Load(System::Object^ sender, System::Event-
Args^ e);
        System::Void CleanText_MouseClick(System::Object^ sender,
System::Windows::Forms::MouseEventArgs^ e);
        System::Void EncBut_Click(System::Object^ sender, System::Event-
Args^ e);
        System::Void FirstForm_Activated(System::Object^ sender,
System::EventArgs^ e);
};
}
```

FirstForm.cpp

```
#include "FirstForm.h"
using namespace System;
using namespace System::Windows::Forms;
using namespace Cursovaya;
[STAThreadAttribute]

int main(array<String^>^ args)
{
   ContextMenuStrip;
   Application::SetCompatibleTextRenderingDefault(false);
   Application::EnableVisualStyles();
   FirstForm form;
   Application::Run(% form);
}

Void FirstForm::FirstForm_Load(Object^ sender, EventArgs^ e)
{
   this->CleanText->Text = L"Начните ввод сообщения, после чего выберите внизу тип шифровния";
   return Void();
}
```

```
Void FirstForm::CleanText_MouseClick(Object^ sender, MouseEventArgs^ e)
{
   if(this->CleanText->Text == L"Начните ввод сообщения, после чего выберите внизу тип шифровния")
      this->CleanText->Text = L"";
   return Void();
}
Void FirstForm::EncBut_Click(Object^ sender, EventArgs^ e)
{
   text = msclr::interop::marshal_as<string>(CleanText->Text);
   enc_choose = msclr::interop::marshal_as<string>(EncChoose->Text);
   if (enc_choose == "Выберите тип шифрования")
      return Void();
   if (enc_choose == "Шифр Виженера" && open == 0)
   {
      f = gcnew VinForm();
      f->Show();
      open++;
      return Void();
   }
   KEY = "Ключ: ";
   text = db[enc_choose]->Cipher(text);
   Encryption->Text = gcnew String(text.c_str());
   KEY += db[enc_choose]->print_key;
   textBox1->Text = gcnew String(KEY.c_str());
   return Void();
}
```

```
Void FirstForm::FirstForm_Activated(Object^ sender, EventArgs^ e)
{
   if (flag)
   {
      KEY = "Ключ: ";
      text = msclr::interop::marshal_as<string>(CleanText->Text);
      db[enc_choose]->print_key = msclr::interop::marshal_as<string>(f->GetKey());
      text = db[enc_choose]->Cipher(text);
      Encryption->Text = gcnew String(text.c_str());
      KEY += db[enc_choose]->print_key;
      textBox1->Text = gcnew String(KEY.c_str());
   }
   return Void();
}
```

VinForm.h

```
#pragma once

namespace Cursovaya {
        using namespace System;
        using namespace System::ComponentModel;
        using namespace System::Collections;
        using namespace System::Windows::Forms;
        using namespace System::Data;
        using namespace System::Drawing;
        /// <summary>
```

```
/// Сводка для VinForm
/// </summary>
public ref class VinForm : public System::Windows::Forms::Form
{
public:
	VinForm(void)
	{
		InitializeComponent();
		//
		//TODO: добавьте код конструктора
		//
	}
protected:
	/// <summary>
	/// Освободить все используемые ресурсы.
	/// </summary>
	~VinForm()
	{
		delete components;
	}
private: System::Windows::Forms::Button^ VinBut;
private: System::Windows::Forms::TextBox^ VinKeyWord;
public:
	/// <summary>
	/// Обязательная переменная конструктора.
	/// </summary>
	System::ComponentModel::Container ^components;
```

```
#pragma region Windows Form Designer generated code
        /// <summary>
        /// Требуемый метод для поддержки конструктора — не изменяйте
        /// содержимое этого метода с помощью редактора кода.
        /// </summary>
        void InitializeComponent(void)
        {
            this->VinBut = (gcnew System::Windows::Forms::Button());
            this->VinKeyWord = (gcnew
System::Windows::Forms::TextBox());
            this->SuspendLayout();
            //
            // VinBut
            //
            this->VinBut->Font = (gcnew System::Drawing::Font(L"Mi-
crosoft Sans Serif", 12, System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->VinBut->Location = System::Drawing::Point(12, 73);
            this->VinBut->Name = L"VinBut";
            this->VinBut->Size = System::Drawing::Size(260, 30);
            this->VinBut->TabIndex = 0;
            this->VinBut->Text = L"Зашифровать";
            this->VinBut->UseVisualStyleBackColor = true;
            this->VinBut->Click += gcnew System::EventHandler(this,
&VinForm::VinBut_Click);
            //
            // VinKeyWord
```

```
            //
            this->VinKeyWord->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 12,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->VinKeyWord->Location = System::Drawing::Point(12, 21);
            this->VinKeyWord->Multiline = true;
            this->VinKeyWord->Name = L"VinKeyWord";
            this->VinKeyWord->Size = System::Drawing::Size(260, 30);
            this->VinKeyWord->TabIndex = 2;
            this->VinKeyWord->Text = L"Введите ключевое слово";
            this->VinKeyWord->MouseClick += gcnew
System::Windows::Forms::MouseEventHandler(this, &VinForm::VinKey-
Word_MouseClick);
            //
            // VinForm
            //
            this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
            this->AutoScaleMode = System::Windows::Forms::AutoScale-
Mode::Font;
            this->ClientSize = System::Drawing::Size(283, 110);
            this->Controls->Add(this->VinKeyWord);
            this->Controls->Add(this->VinBut);
            this->Name = L"VinForm";
            this->Text = L"VinForm";
            this->Load += gcnew System::EventHandler(this,
&VinForm::VinForm_Load);
            this->ResumeLayout(false);
            this->PerformLayout();
```

```
            }
#pragma endregion
        public:
            String^ GetKey();
            System::Void VinBut_Click(System::Object^ sender, System::Event-
Args^ e);
            System::Void VinKeyWord_MouseClick(System::Object^ sender,
System::Windows::Forms::MouseEventArgs^ e);
            System::Void VinForm_Load(System::Object^ sender, System::Event-
Args^ e);
        };
}
```

VinForm.cpp

```
#include "VinForm.h"
#include "FirstForm.h"

using namespace System;
using namespace System::Windows::Forms;
using namespace Cursovaya;

String^ VinForm::GetKey()
{
   return VinKeyWord->Text;
}

Void VinForm::VinBut_Click(Object^ sender, EventArgs^ e)
```

```
{
    if (VinKeyWord->Text->Length == 0 || VinKeyWord->Text == L"Введите ключевое слово")
    {
        VinKeyWord->Text = L"Введите ключевое слово";
        return Void();
    }
    open--;
    flag = true;
    this->Close();

    return Void();
}

Void VinForm::VinKeyWord_MouseClick(Object^ sender, MouseEventArgs^ e)
{
    if (VinKeyWord->Text == L"Введите ключевое слово")
        VinKeyWord->Text = L"";

    return Void();
}

System::Void Cursovaya::VinForm::VinForm_Load(System::Object^ sender, System::EventArgs^ e)
{
    return System::Void();
}
```