Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

РАЗРАБОТКА HTTP-СЕРВЕРА

БГУИР КР 1-40 02 01 201 ПЗ

Студент: Д. Г. Альховик

Руководитель: Е. Д. Игнатьев

Минск 2018

Оглавление

[ВВЕДЕНИЕ 5](#_Toc6435853)

[1 ОБЗОР ИСТОЧНИКОВ 6](#_Toc6435854)

[2 СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ 11](#_Toc6435855)

[3 ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ 12](#_Toc6435856)

[Поля: 12](#_Toc6435857)

[Поля: 12](#_Toc6435858)

[Поля: 13](#_Toc6435859)

[Поля: 14](#_Toc6435860)

[4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ 15](#_Toc6435861)

[5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 18](#_Toc6435862)

[6 ТЕСТИРОВАНИЕ 23](#_Toc6435863)

[ЗАКЛЮЧЕНИЕ 25](#_Toc6435864)

[**ПРИЛОЖЕНИЕ А** 28](#_Toc6435865)

[**ПРИЛОЖЕНИЕ Б** 29](#_Toc6435866)

[**ПРИЛОЖЕНИЕ В** 30](#_Toc6435867)

# ВВЕДЕНИЕ

Во все времена информация была и остаётся одним из основных жизненно необходимых ресурсов человека. Она является основным безопасным способом передачи опыта от одного человека другому, а, следовательно, получая и обрабатывая большие объемы информации человек получает знания необходимые ему для жизнедеятельности и самосовершенствования.

С течением времени человечество накопило очень много сведений об окружающем нас мире, но сохранить и осмыслить столько данных человек не способен. Тогда возникает вопрос: как не потерять весь тот опыт, который человечество накапливало веками?

Выходом стало фиксирование полученной информации во всевозможных хранилищах, таких как книги. Однако и книга не всегда удобна для использования из-за объёмных размеров и трудности быстрого поиска необходимых сведений. Тогда на помощь пришли вычислительные машины, которые способны хранить огромные объёмы информации при помощи баз данных.

Хотя работа с базами данных не всегда является удобной напрямую, так как иногда необходим определенный осмысленный поиск среди хранящийся информации, либо же некоторые данные и вовсе не предоставляют ценности без соответствующей обработки и т.д.

Отличным решением для данной проблемы станет использование технологии «клиент-сервер». В данном случае это удобно, так как пользователь может доступным для него образом отправить запрос на сервер, который сумеет грамотно его обработать и выполнить, чтобы удовлетворить потребность клиента. А дабы работа клиента с сервером могла быть оперативной из любой точки земного шара следует использовать «всемирную сеть» как способ передачи запросов.

Целью данного курсового проекта будет разработка сервера, позволяющего сохранять, обрабатывать и возвращать необходимые данные по запросу пользователя и работающего по протоколу http(HyperText Transfer Protocol).

Таким образом в результате должны получить готовый http-сервер, способный работать с несколькими типами клиентов для хранения и обработки их информации на удалённом носителе.

В соответствии с поставленной целью определены следующие задачи:

* Организация обработки запроса пользователя
* Преобразование данных для их хранения
* Запись данных в базу данных
* Чтение данных из базы данных
* Организация работы по протоколу http

# ОБЗОР ИСТОЧНИКОВ

Для решения поставленных задач необходимо определить нужные компоненты для создания http-сервер. Для создания связи между клиентом и сервером нужно использовать соответствующую сетевую архитектуру. Для её реализации и построения корректной работы с базами данных также необходимо использовать подходящий для этого язык программирования.

Важно также выбрать базу данных, которая позволила бы хранить различные типы данных для более универсального использования сервера. Связь между протоколом и сервером воспользуемся стандартными http-протоколами.

1.1 Архитектура «Клиент-сервер»

«Клиент-сервер» — вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами.

Фактически клиент и сервер — это программное обеспечение. Обычно эти программы расположены на разных вычислительных машинах и взаимодействуют между собой через вычислительную сеть посредством сетевых протоколов, но они могут быть расположены также и на одной машине. Программы-серверы ожидают от клиентских программ запросы и предоставляют им свои ресурсы в виде данных (например, загрузка файлов посредством HTTP, FTP, BitTorrent, потоковое мультимедиа или работа с базами данных) или в виде сервисных функций (например, работа с электронной почтой, общение посредством систем мгновенного обмена сообщениями или просмотр web-страниц во всемирной паутине).

Поскольку одна программа-сервер может выполнять запросы от множества программ-клиентов, её размещают на специально выделенной вычислительной машине, настроенной особым образом, как правило, совместно с другими программами-серверами, поэтому производительность этой машины должна быть высокой. Из-за особой роли такой машины в сети, специфики её оборудования и программного обеспечения, её также называют сервером, а машины, выполняющие клиентские программы, соответственно, клиентами.

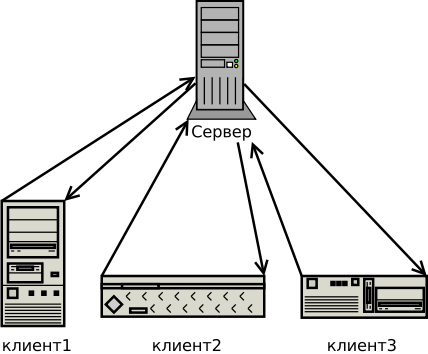


Рисунок 1.1 – Сетевая архитектура «Клиент-сервер»

Преимуществами данного подхода являются:

* Отсутствие дублирования кода программы-сервера программами-клиентами.
* Так как все вычисления выполняются на сервере, то требования к компьютерам, на которых установлен клиент, снижаются.
* Так как все вычисления выполняются на сервере, то требования к компьютерам, на которых установлен клиент, снижаются.

1.2 Обзор протокола HTTP

HyperText Transfer Protocol – это протокол передачи данных прикладного уровня, предназначенный передачи произвольных данных. Основой HTTP является технология «клиент-сервер».

Основным объектом манипуляции в HTTP является ресурс, на который указывает URI (Uniform Resource Identifier) в запросе клиента. Обычно такими ресурсами являются хранящиеся на сервере файлы, но ими могут быть логические объекты или что-то абстрактное. Особенностью протокола HTTP является возможность указать в запросе и ответе способ представления одного и того же ресурса по различным параметрам: формату, кодировке, языку и т. д.

Обмен сообщениями идёт по обыкновенной схеме «запрос-ответ». Для идентификации ресурсов HTTP использует глобальные URI.

1.3.1 Сервлеты

Сервлет является интерфейсом Java, реализация которого расширяет функциональные возможности сервера. Сервлет взаимодействует с клиентами посредством принципа запрос-ответ.

Хотя сервлеты могут обслуживать любые запросы, они обычно используются для расширения веб-серверов. Для таких приложений технология Java Servlet определяет HTTP-специфичные сервлет классы. Пакеты javax.servlet и javax.servlet.http обеспечивают интерфейсы и классы для создания сервлетов.

Описать работу сервлета можно так: на сервер приходит запрос от клиента, запрос содержит внутри себя URL и параметры. Сервер имеет специальный конфигурационный файл, который ему сообщает о том, какой сервлет надо выполнить в случае прихода определенного URL. Сервлет выполняется (там вы можете использовать параметры) и создает HTML-страницу, которая отсылается клиенту.

Сервер по сути является контейнером (теперь уже не визуальных компонентов), который загружает сервлеты, выполняет их, вызывая определенные методы и получив от них результат, отправляет его клиенту.

1.3.2 Контейнер сервлетов Apache Tomcat

Tomcat (в старых версиях — Catalina) — контейнер сервлетов с открытым исходным кодом, разрабатываемый Apache Software Foundation. Реализует спецификацию сервлетов, спецификацию JavaServer Pages (JSP) и JavaServer Faces (JSF). Написан на языке Java.

Tomcat используется в качестве самостоятельного веб-сервера, в качестве сервера контента в сочетании с веб-сервером Apache HTTP Server, а также в качестве контейнера сервлетов в серверах приложений JBoss и GlassFish.

Для данного проекта нам подойдёт такой компонент как Coyote, который поддерживает протокол HTTP 1.1 для веб-серверов или контейнера приложений. Coyote прослушивает входящие соединения на определённом TCP порту сервера, пересылает запросы в механизм Tomcat для обработки запросов и отправляет ответ назад запрашивающему клиенту.

1.4 Система управления базами данных MongoDB

MongoDB — документоориентированная система управления базами данных (СУБД) с открытым исходным кодом, не требующая описания схемы таблиц. Классифицирована как NoSQL, использует JSON-подобные документы и схему базы данных. Написана на языке C++.

Система поддерживает ad-hoc-запросы: они могут возвращать конкретные поля документов и пользовательские JavaScript-функции. Поддерживается поиск по регулярным выражениям. Также можно настроить запрос на возвращение случайного набора результатов.

Данная система подходит нам больше всего, так как она не хранит информацию в таблицах, что удобно нам, потому что мы не знаем изначально, в каком виде информация будет поступать нам на сервер. Это даёт нам возможность хранить объекты целиком, не конвертируя содержимое данных.

1.5 Обзор аналогов

Сервис разрабатывается для сохранения нужной нам информации на удаленном сервере по заданному заголовку, что позволяет нам, не имея логина и пароля, а лишь зная нужную комбинацию символов, получать сохраненную ранее информацию или записывать новую.

Важной особенностью также является то, что это не будет какое-то приложение, которое необходимо скачивать и устанавливать на своём устройстве, так как всё взаимодействие с пользователем происходит через страницу интернет-браузера по заданному адресу. Это даёт широкий спектр возможностей и преимуществ перед другими аналогами, так как делает сервис легко запускаемым с любого устройства, поддерживающего работу браузера, и доступным для любого пользователя.

Основными аналогами данного сервиса является Listik, который является сервисом онлайн заметок для хранения страниц браузера, нужной информации или файлов. Из приложений можно выделить такие крупные проекты как Simplenote и Evernote.

1.5.1 Онлайн сервис заметок Listik

Listik – онлайн сервис заметок, разработанный компанией Sinta в 2013 году, который предназначен для общего использования в быту. Довольно простой интерфейс в сочетании с приятным дизайном делает Listik отличным сервисом для хранения своих мыслей, небольших фалов и скриншотов браузера.

В сервисе каждый пользователь имеет свой кабинет, в котором он может работать и составлять свои планы или хранить нужные данные.

Из минусов можно заметить лишь отсутствие совместной работы пользователей, что не дает использовать данный сервис для работы в компании.

1.5.2 Приложение Simpleton

Приложение позволяет синхронизировать записи между всеми устройствами без оформления подписки. Пользователь имеет возможность обмениваться заметками с другими и вести с ними совместную работу. Интерфейс Simplenote очень прост и удобен в управлении.

Программа поддерживается всеми популярными операционными системами. Ключевой особенностью Simplenote является исключительная скорость и эффективность работы.

1.5.3 Приложение Evernote

Evernote позволяет создавать как простые, так и сложные рабочие процессы, используя комбинацию заметок и тегов, чтобы все было организовано.

Одной из лучших возможностей для сбора исследований является расширение Web Clipper (поддерживается в Chrome, Firefox и Safari), которое позволяет сэкономить целые веб-страницы, включая текст, изображения и PDF-файлы, одним щелчком мыши. Доступ к заметкам можно получить на ноутбуках, мобильных устройствах. Другие функции включают возможность установки напоминаний, представления стилей PowerPoint и объединение их друг с другом.

Недавние обновление включает новые таблицы и интеграцию с Siri для тех, кто использует версию iOS.

1.6 Постановка задачи

На основе контейнера сервлетов Tomcat создать сервер, способный обрабатывать запросы клиента, выполнять соответствующие операции и связывать нужные HTML страницы с соблюдением протокола HTTP.

Среди основного функционала необходимо создать простой и понятный набор инструментов, помогающий связать запросы пользователя и работу с базой данных MongoDB. Также необходимо создать простой и понятный для пользователя интерфейс для удобной работы с базой данных.

Среди дополнительных задач, которые следовало бы реализовать: создание личных кабинетов, возможность самостоятельно изменять дизайн страниц, расширить возможности хранения различных типов данных.

# СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ

В данном разделе описана структурная схема работы приложения. Сама структурная схема представлена в приложении А.

После изучения теоретических аспектов разрабатываемого программного модуля, появилась возможность разбить систему на отдельные функциональные модули, что позволяет отойти от проектирования монолитного стиля проектирования приложения и перейти к модульности, что позволит разбить приложение на заменяемые блоки, которые облегчат разработку и обновление приложения.

Каждый модуль обладает необходимым функционалом для решения одной или нескольких задач. Модули взаимодействуют с помощью различных данных между собой позволяя приложению работать.

Работа программы разбита на четыре основных блоков:

* блок пользовательского ввода информации,
* блок обработки запроса,
* блок работы с сервером,
* блок работы с базой данных.
  1. Блок пользовательского ввода информации

Блок приёма видео потока работает как клиентская часть приложения. Блок пользовательского ввода информации содержит функции для получения от пользователя необходимых данных, преобразования этих данных в соответствующий запрос и передачи этого запроса во второй блоке, блоке обработки запроса, поэтому связь между ними односторонняя.

1.2 Блок обработки запроса

Блок обработки запроса служит для преобразования запроса, пришедшего из блока пользовательского ввода информации в соответствующий ряд команд для выполнения блока работы с сервером.

1.3 Блок работы с сервером

Блок работы с сервером при получении определенных команд отвечает за переброску команд в блок работы с базой данных для корректного их выполнения. Также этот блок обрабатывает результаты выполнения блока работы с базой данных и передаче их соответствующим функциям для отображения в правильном виде полученных данных, поэтому он обладает широким спектром возможностей и является основным звеном связывающим и согласующем работу базы данных и пользователя.

1.4 Блок работы с базой данных

Блок работы с базой данных отвечает за корректную запись и удаления требуемой по запросу информации. Этот блок также может отправлять данные в блок работы с сервером, в которых будет находится результаты выполнения поставленных задач или запрашиваемой информации. Более сложной частью работы данного блока также является изменение уже существующих данных в базе, так как сочетает в себе грамотную работу по поиску и замены определённого поля в элементе.

# ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

Рассмотрим подробно функционирование программы. Для этого проведем анализ основных модулей программы и рассмотрим их зависимости. А также проанализируем все классы, которые входят в состав кода программы, и рассмотрим назначение всех методов, свойств и переменных класса.

Условно программный модуль можно разделить на следующие части:

* блок обработвки запросов;
* блок работы с базой данных.

3.1 Блок обработки запросов

Java — сильно типизированный объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой компьютерной архитектуре с помощью виртуальной Java-машины. Данный язык удобен для проекта, так как он хорошо совместим с Tomcat и имеет библиотеки для работы с сервлетами.

В данном

1. Класс Cipher:

## Поля:

– QString input – необходима для хранения введённой пользователем информации.

– QString output – необходима для хранения преобразованной информации.

Методы:

– Cipher() – конструктор, в котором инициализируются поля класса.

– virtual ~Chipher() – виртуальный деструктор.

– virtual void setInput(QString \_input) – функция для получения доступа к полю класса QString input..

– virtual void setOutput(QString \_output) – функция для получения доступа к полю класса QString output.

– QString getInput() – функция установки значения поля класса QString input.

– QString getOutput() – функция установки значения поля класса QString input.

– virtual void doCipher()=0 – функция, в которой информация из QString input шифруется и записывается в QString output.

– virtual QString descpiption()=0 – функция, которая читает информацию из соответствующего файла и записывает её в строку.

– static Cipher\* createCipher(int name\_ID, QString \_input, QString \_smth) – функция для реализации патерна «Фабрика», в качестве значения int name\_ID ставится номер нужного шифра.

1. Класс CipherCaesar:

## Поля:

– int shift – необходима для хранения «шага».

Методы:

– CipherCaesar() – конструктор, в котором инициализируются поля класса как «тестовые».

– CipherCaesar(QString \_input, QString \_shift=”3”) – конструктор, в котором инициализируются поля класса.

– ~CipherCaesar() – переопределение виртуального деструктора.

– void setShift(int \_shift) – функция установки значения поля класса int shift.

– int getShift() – функция для получения доступа к полю класса int shift.

– void doCipher() – переопределение функции под шифр Цезаря.

– bool isCapitalLetter(char) – функция возвращающая значение true, если предаваемая переменная является заглавной буквой английского алфавита.

– bool isUppercaseLetter(char) – функция возвращающая значение true, если предаваемая переменная является строчной буквой английского алфавита.

– QString description() – переопределение функции под шифр Цезаря.

1. Класс CipherVizhener:

## Поля:

– QString key – необходима для хранения ключа.

Методы:

– CipherVizhener() – конструктор, в котором инициализируются поля класса как «тестовые».

– CipherVizhener(QString \_input, QString \_key=”a”) – конструктор, в котором инициализируются поля класса.

– ~CipherVizhener() – переопределение виртуального деструктора.

– void setKey(QString \_key) – функция установки значения поля класса QString key.

– QString getKey() – функция для получения доступа к полю класса QString key.

– void doCipher() – переопределение функции под шифр Виженера.

– bool isCapitalLetter(char) – функция возвращающая значение true, если предаваемая переменная является заглавной буквой английского алфавита.

– bool isUppercaseLetter(char) – функция возвращающая значение true, если предаваемая переменная является строчной буквой английского алфавита.

– QString description() – переопределение функции под шифр Виженера.

1. Класс TranslationNS:

## Поля:

– QString key – необходима для хранения «шага».

Методы:

– TranslationNS() – конструктор, в котором инициализируются поля класса как «тестовые».

–TranslationNS(QString \_input, QString \_key=”a”) – конструктор, в котором инициализируются поля класса.

– ~TranslationNS() – переопределение виртуального деструктора.

– void seеNS(int \_ns) – функция установки значения поля класса int ns.

– int getKey() – функция для получения доступа к полю класса int ns.

– void doCipher() – переопределение функции для перевода чисел из десятичной системы счисления.

– QString description() – переопределение функции для перевода чисел из десятичной системы счисления.

Для упрощения работы с классами в данной работе используются преобразованные элементы стандартной библиотеки шаблонов (STL), а конкретно контейнер QString.

Также для дальнейшего улучшения программы используется патерн «Фабрика», который позволяет без значительных изменений в коде программы расширять её функционал.

# РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

В данном разделе приведён алгоритм шифрования данных по методу Цезаря. Он построен на принципе посимвольного преобразования входной строки и записи в поле выходной.

После выбора пользователя алгоритма Цезаря, ввода входных данных и нажатия кнопки «DO», программа выполняет функцию, в результате которой создаётся указатель на класс советующего алгоритма.

void MainWindow::on\_doButton\_clicked()

{

QString input=ui->inputLineEdit->text();

QString key=ui->keyLineEdit->text();

…

if(ui->caesarRadioButton->isChecked())

{

Cipher \*p=Cipher::createCipher(1, input, key);

…

}

…

}

Здесь l – это индивидуальный номер(name\_ID), известный разработчику, метода шифрования. Вторые два параметра input и key – это входные данные, т.е. текст для преобразования и «шаг». В ходе работы конструктора вызывается метод void doCipher(), который и создаёт зашифрованную строку и помещает в поле QString output.

void CipherCaesar::doCipher()

{

QString str=getInput(), res;

if(str.isEmpty())

{

setOutput(" ");

return;

}

for (int i=0; i<str.length(); ++i)

{

if(str[i].isLetter())

{

int ch=str[i].cell();

if (isUppercaseLetter(ch))

{

ch+=shift;

while(!isUppercaseLetter(ch))

{

if(ch>'z')

{

ch-=26;

}

else

{

ch+=26;

}

}

res[i]=static\_cast<char>(ch);

continue;

}

ch+=shift;

while(!isCapitalLetter(ch))

{

if(ch>'Z')

{

ch-=26;

}

else

{

ch+=26;

}

}

res[i]=static\_cast<char>(ch);

continue;

}

res[i]=str[i];

}

setOutput(res);

}

Изначально для удобства работы создаются переменные str, в которую помещается исходный текст, а также переменная для записи результата res.

После идёт проверка на существование введенной строки.

Затем осуществляется проход по контейнеру, в который уже занесена вся имеющаяся информация и преобразуются только те значения, которые являются символом английского алфавита. Если символ не является буквой, он записывается в выходную строку без изменений.

Если же символ является буквой, то его Unicode записывается в рабочую переменную int ch с помощью метода QChar::cell(). Далее к полученному Unicode добавляется «шаг», что и является ключевым моментом шифра Цезаря.

Поскольку «шаг» мог быть таким, что в результате изменения Unicode рабочей переменной вышел за пределы английского алфавита, то во избежание изменения регистра введенной буквы и несоответствия её символу английского алфавита выполняются соответствующие проверки.

После записи всего шифра результат заноситься в поле хранения результата QString output, которое будет позже выведено на экран.

Полученную результирующую строку заносим в текстовое поле с помощью функции:

void MainWindow::on\_doButton\_clicked()

{

…

if(ui->caesarRadioButton->isChecked())

{

Cipher \*p=Cipher::createCipher(1, input, key);

QString result=(\*p).getOutput();

ui->resultOutput->setText(result);

}

…

}

После чего производиться вывод на экран строки с результатом работы в соответствующем поле интерфейса.

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Для корректной работы разработанного программного средства нужны следующие файлы и папки:

— исполняемый файл;

— необходимые для работы приложения DLL файлы.

При соблюдении требуемых условий дополнительной установки компонентов не требуется.

Разработанное программное средство просто в использовании. Ниже будут даны описания последовательностей действий, необходимых для успешной генерации шифра.

После запуска программы откроется главное рабочее окно.

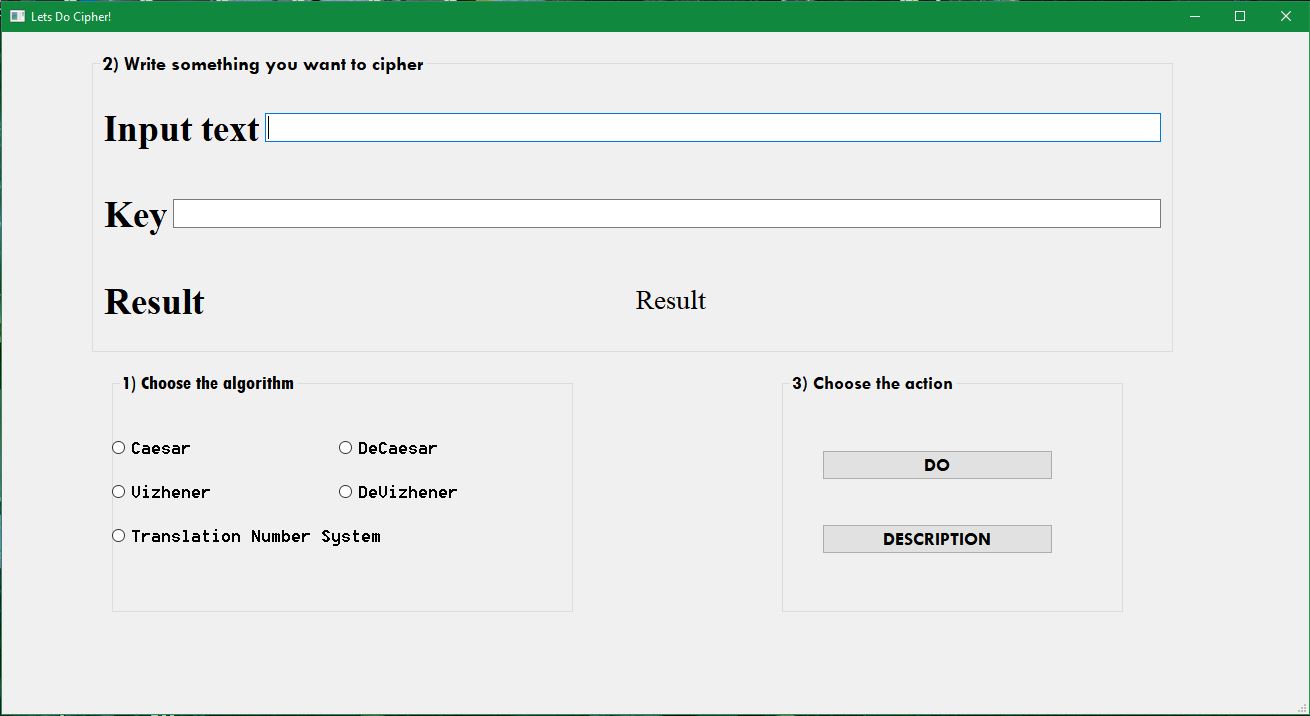


Рисунок 1

Далее в появившемся списке необходимо выбрать один из предложенных алгоритмов кодирования.

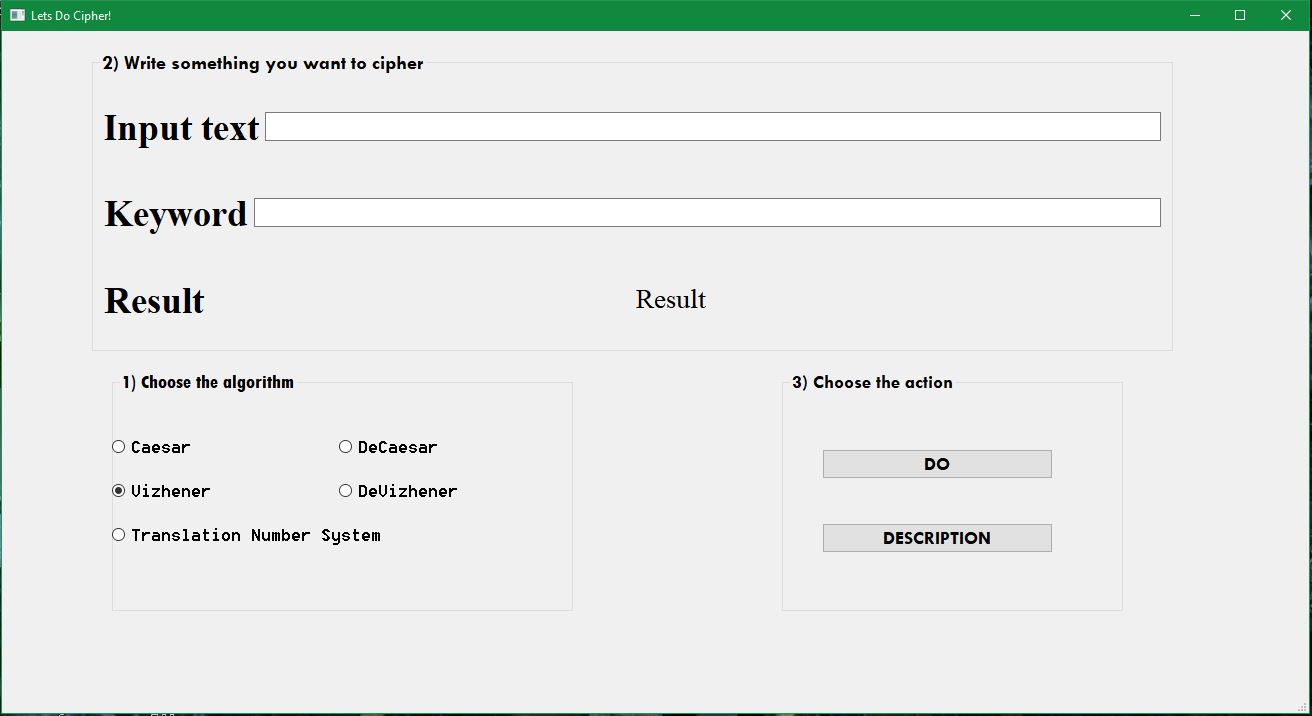


Рисунок 2

После чего необходимо ввести текст для шифрования и ключ.

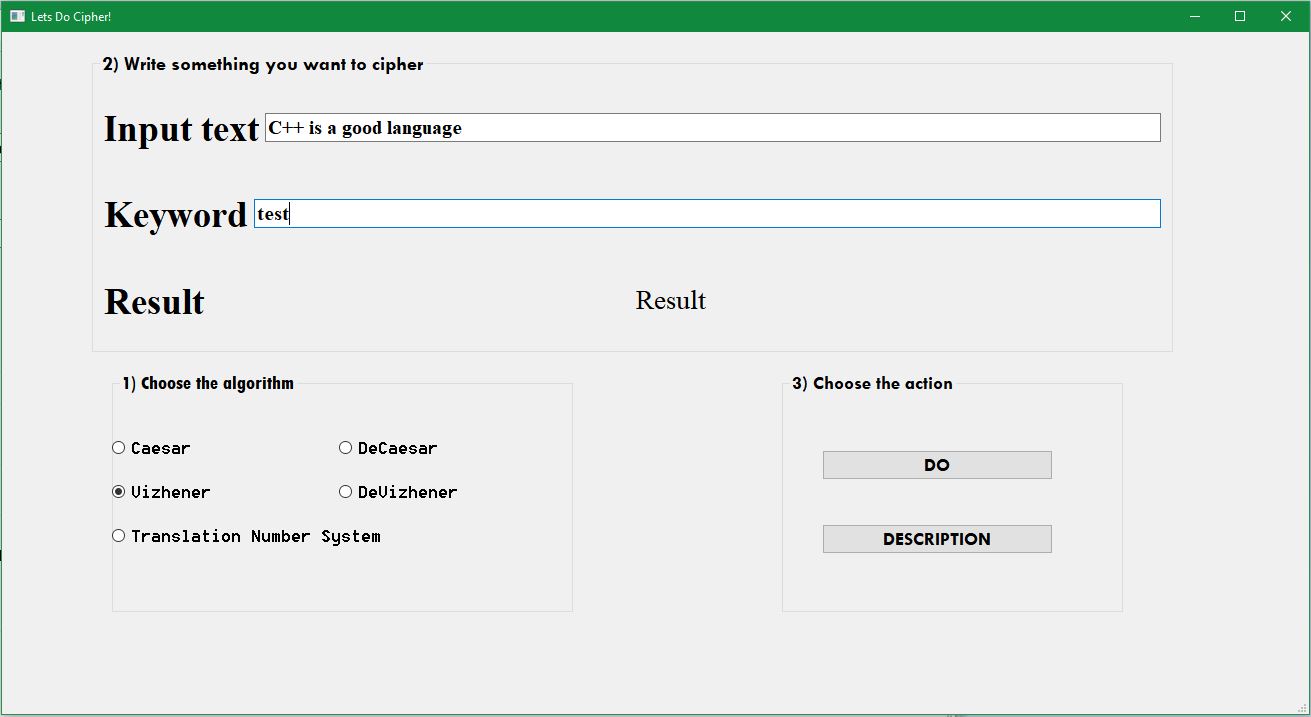


Рисунок 3

Затем для получения зашифрованного текста следует нажать кнопку «DO».

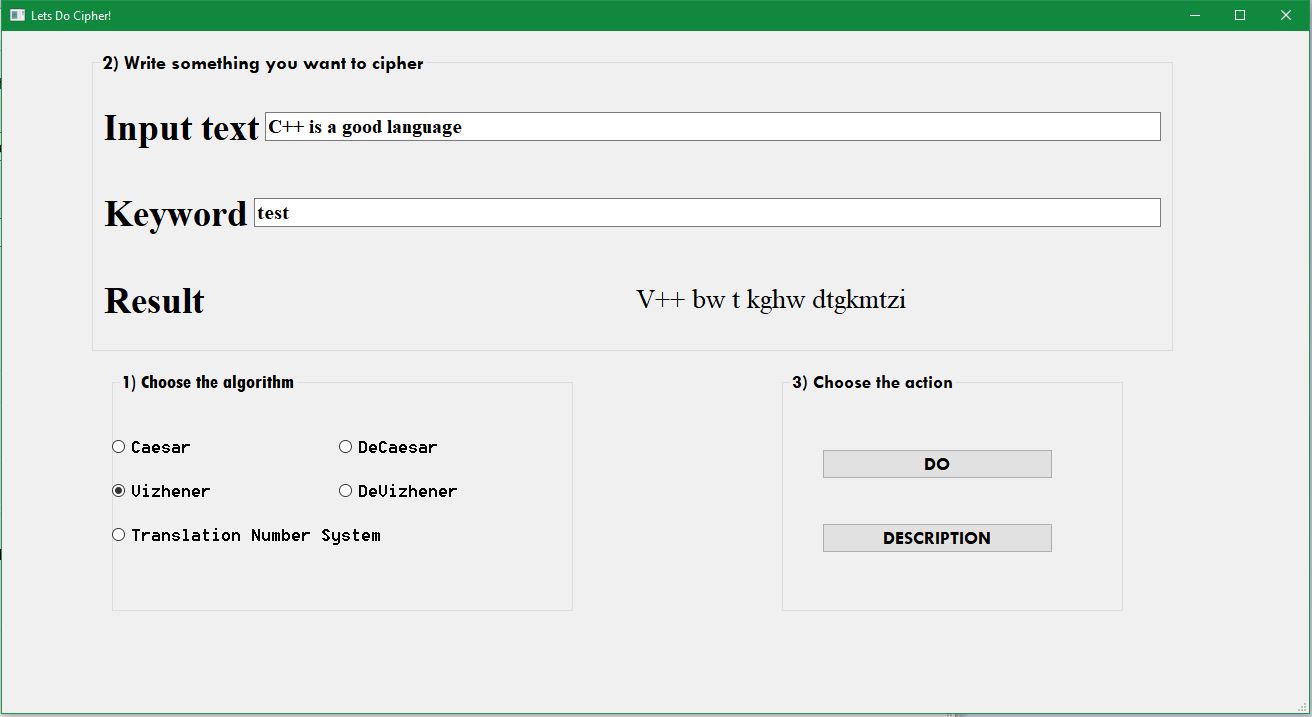


Рисунок 4

Для получения справочной информации о способе кодирования нужно нажать кнопку «DESRIPTION».

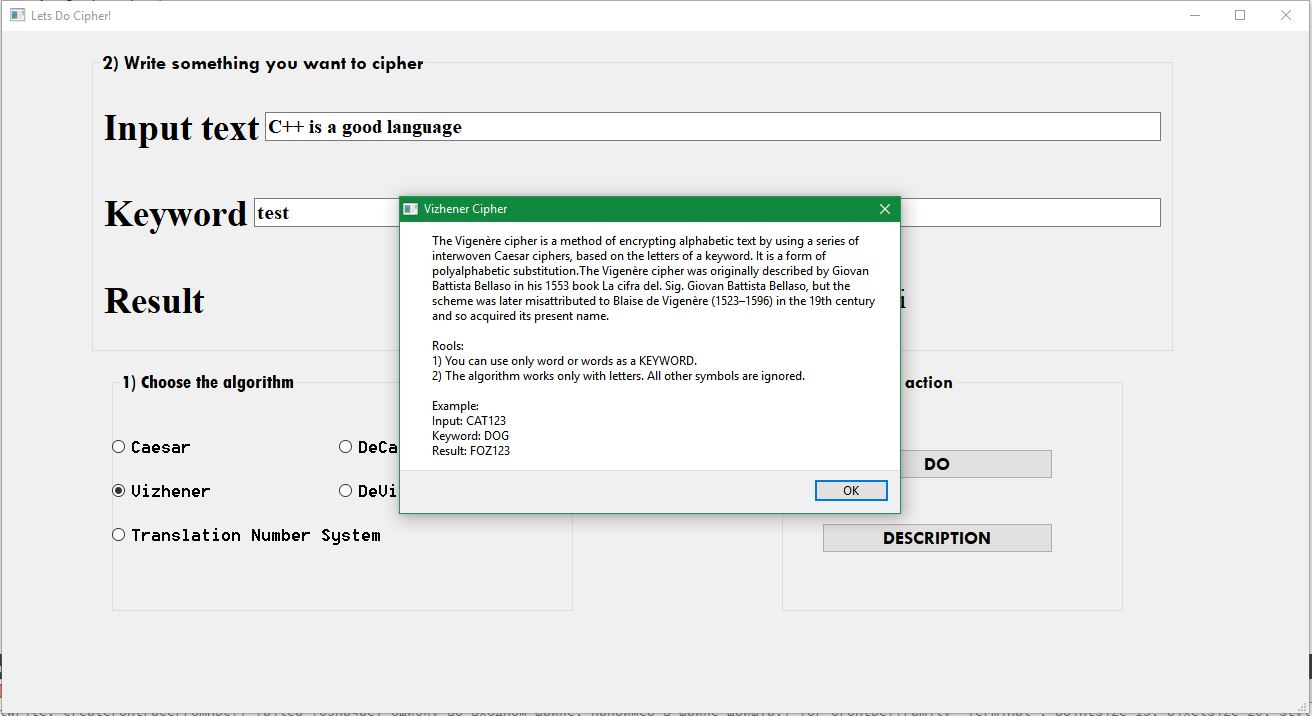


Рисунок 5

Для удобства пользования при наведении на поля ввода либо кнопки «DO» программа подсказывает, какую информацию необходимо ввести для корректной работы программы. При наведении на кнопку «DESCRIPTION» сообщает, что делает данная кнопка.

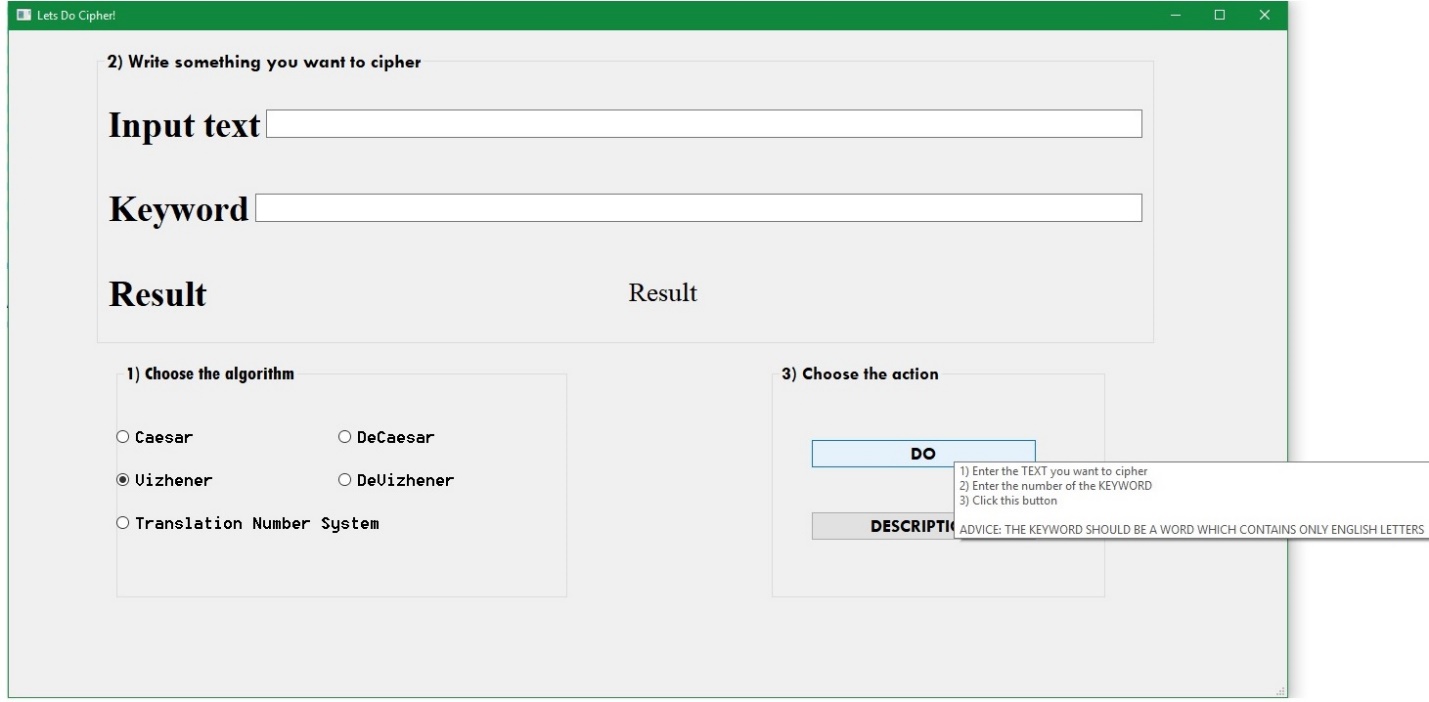


Рисунок 6

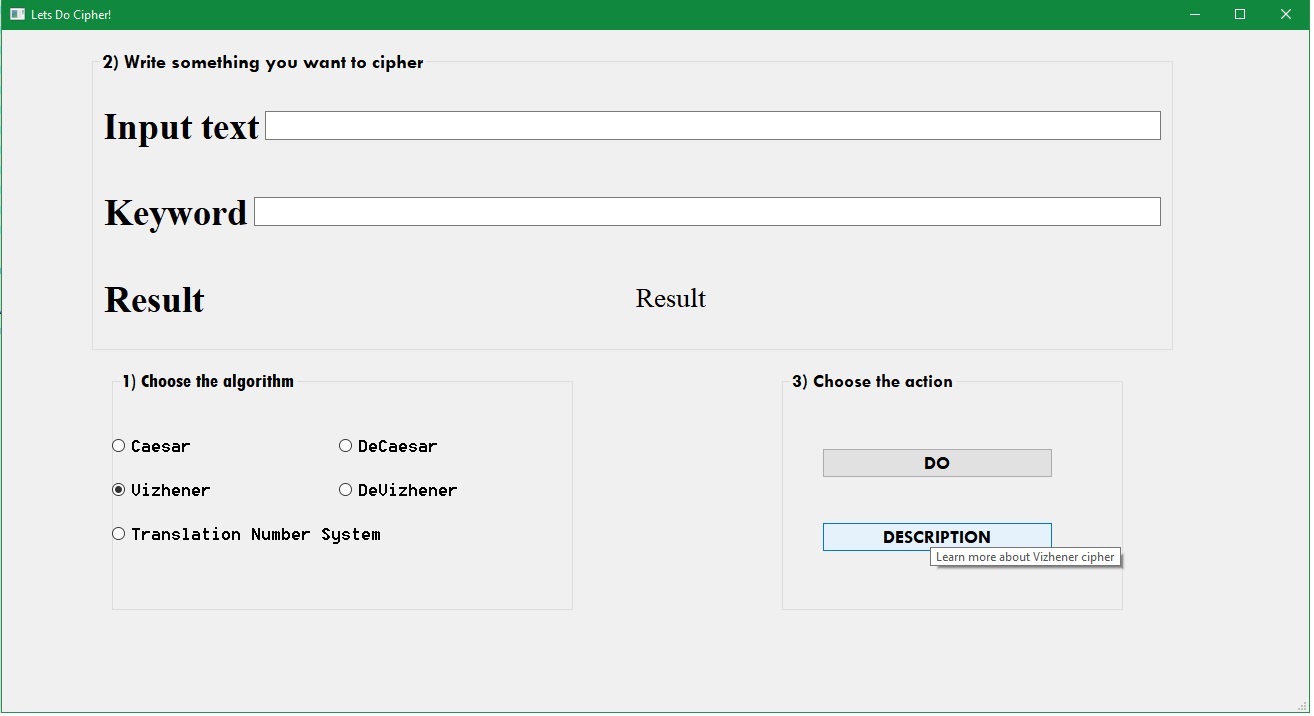


Рисунок 7

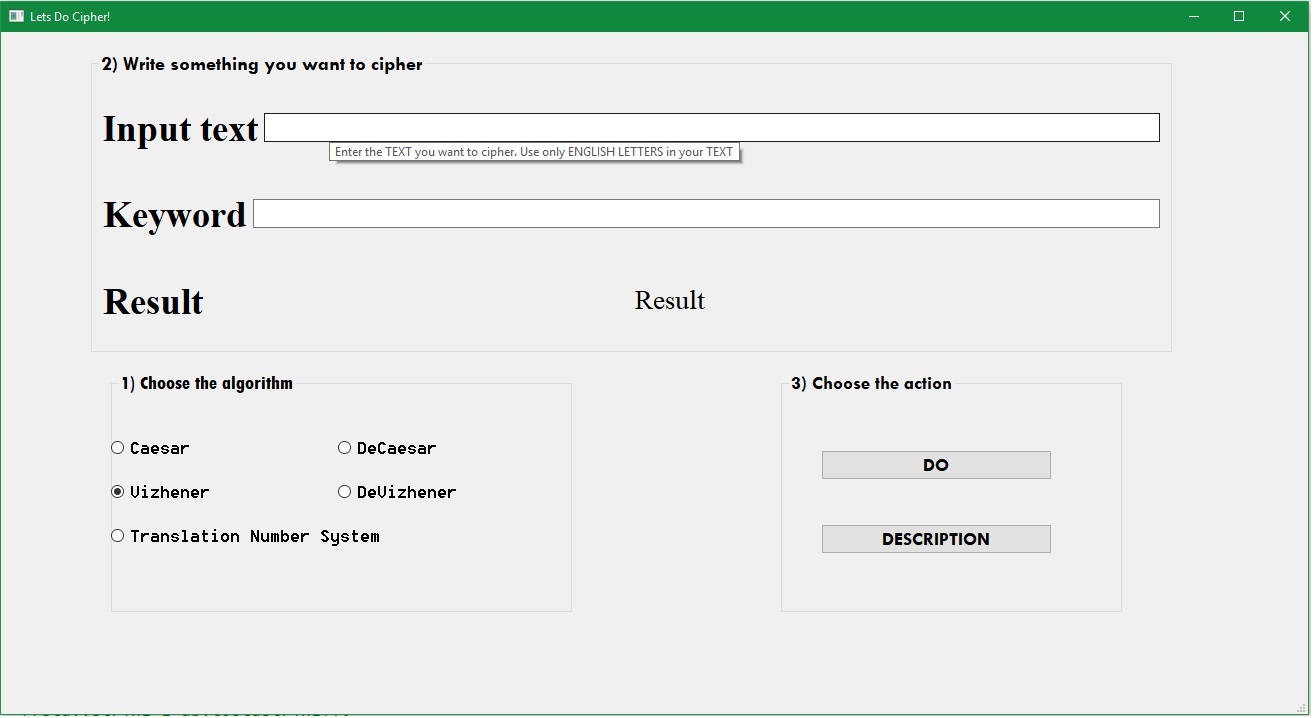


Рисунок 8

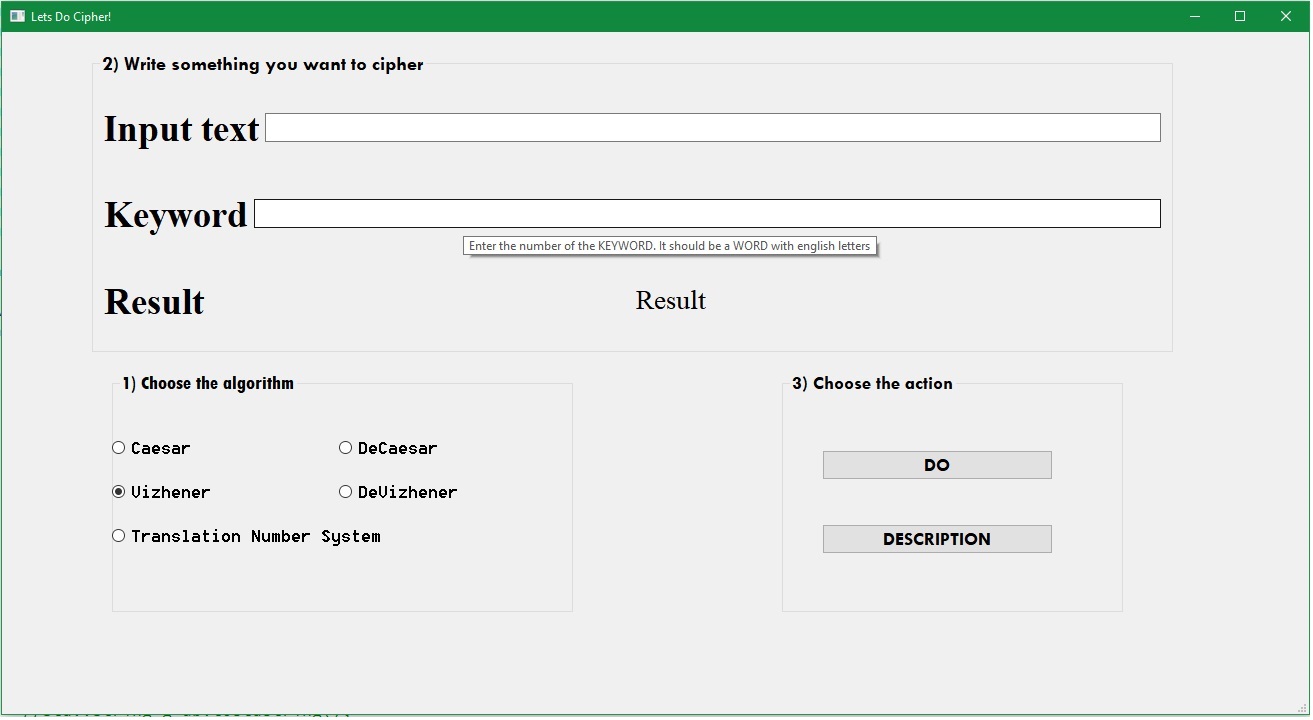


Рисунок 9

# ТЕСТИРОВАНИЕ

Проведём тестирование некоторых кнопок.

|  |  |  |  |
| --- | --- | --- | --- |
| Вид кнопки | Действие | Ожидаемый результат | Успешность тестирования |
| 1 | 2 | 3 | 4 |
| Radio Button | Нажатие на кнопку | Изменение текста некоторых полей, текста подсказок | да |
| Push Botton «DO» | Нажатие на кнопку | Вывод соответствующего шифра | да |
| Line Edit | Наведение мыши на поле | Вывод подсказки | да |
| Push Button  «DESRIPTION» | Нажатие на кнопку | Вывод окна с описанием алгоритма | да |

Результат тестирования Radio Button.

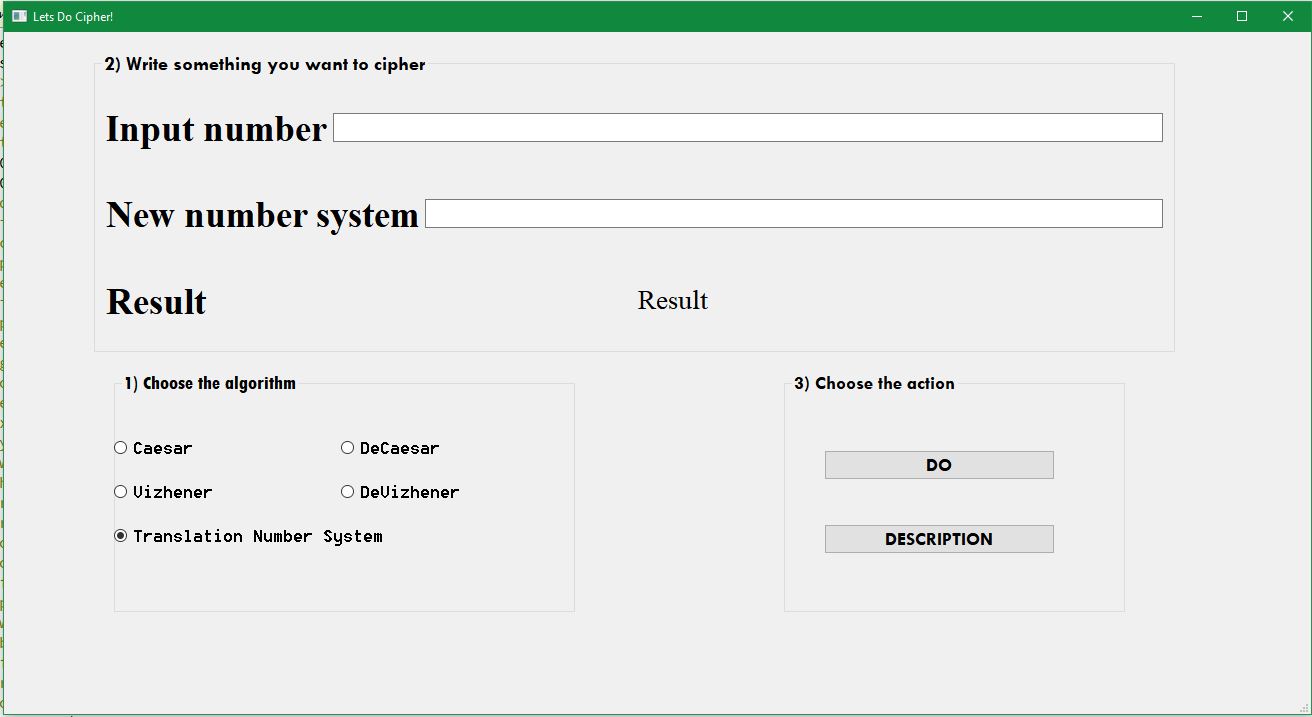


Рисунок 10

Результат тестирования Push Botton «DO».

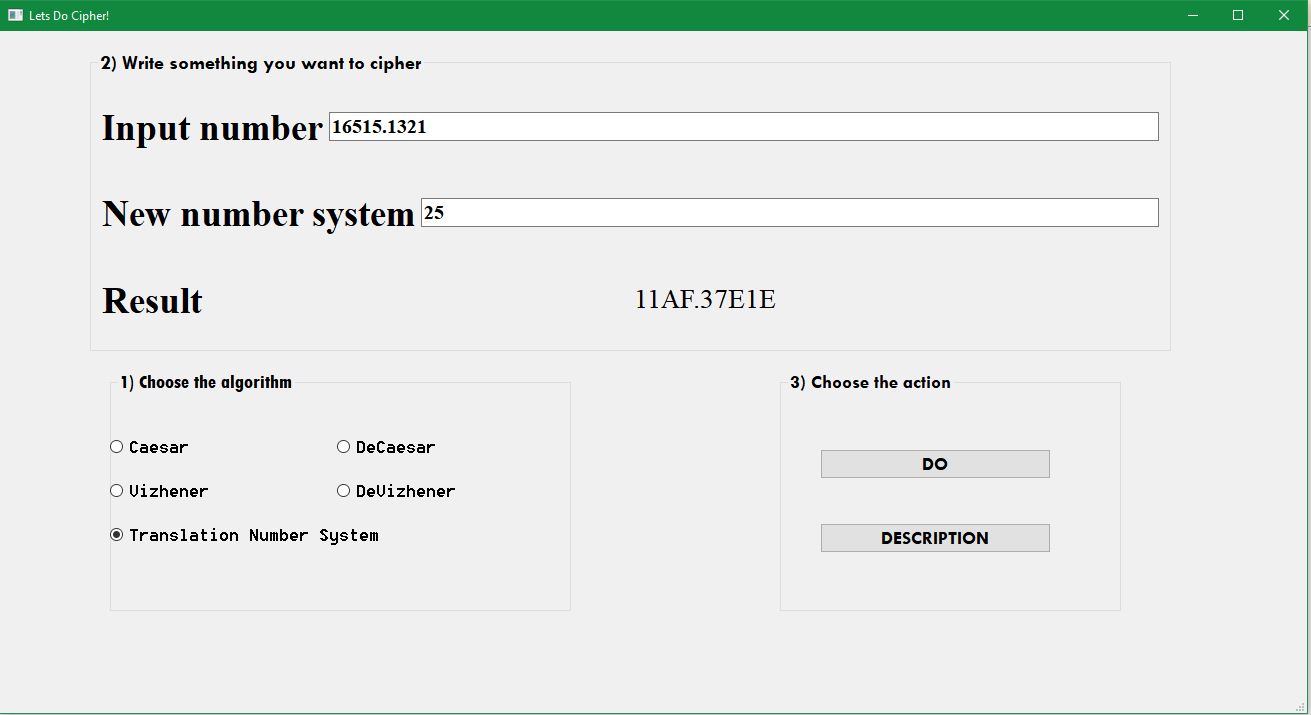


Рисунок 11

Результат тестирования Line Edit.

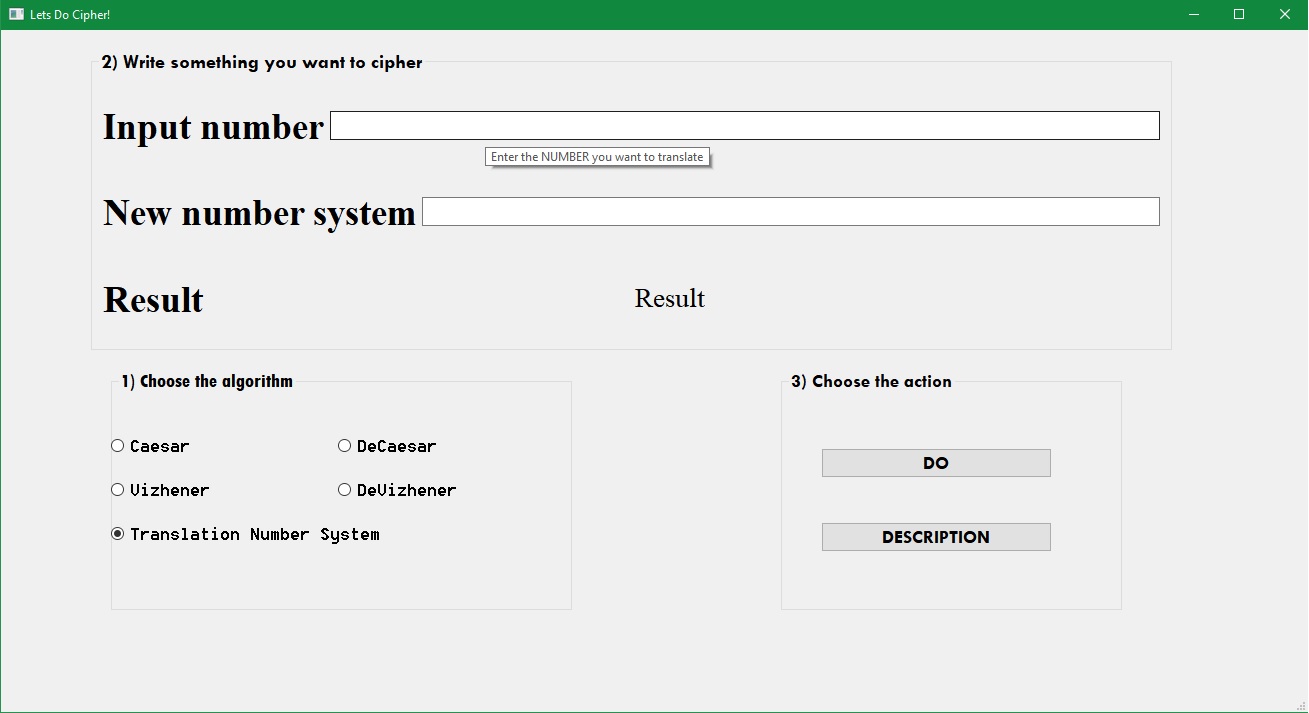


Рисунок 12

Результат тестирования Push Button «DESCRIPTION».

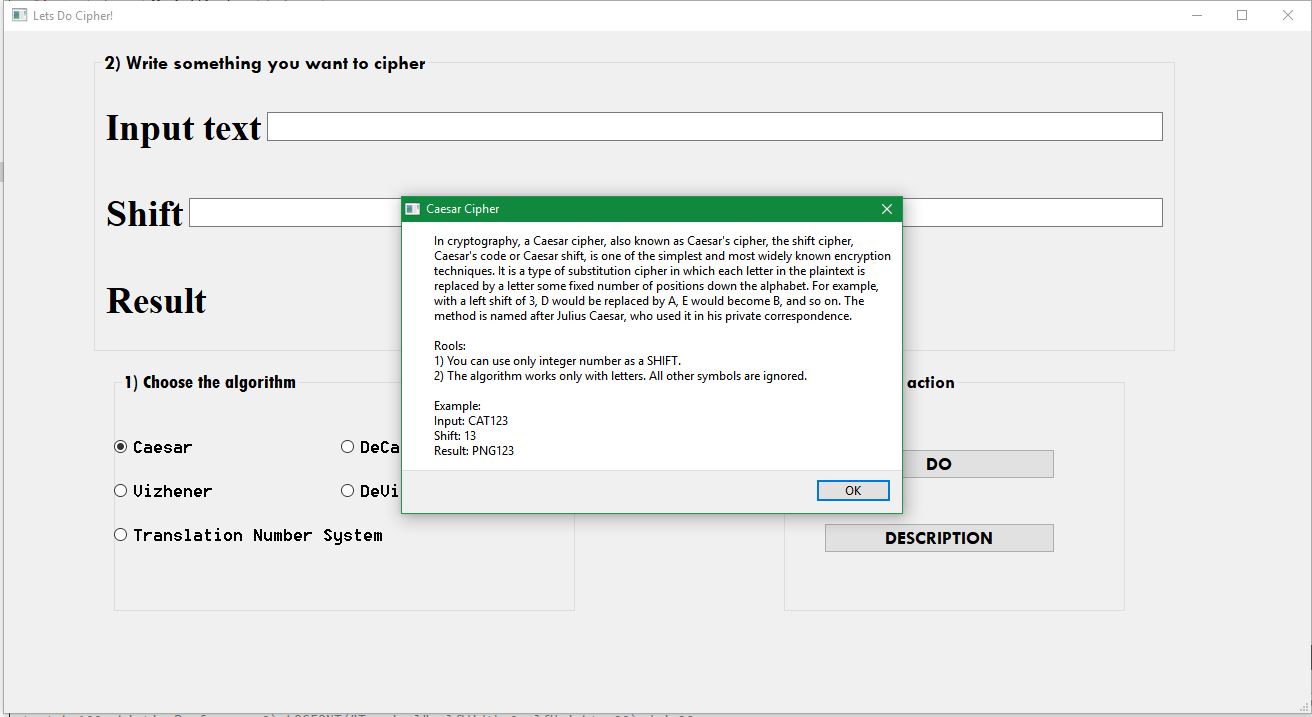


Рисунок 13

# ЗАКЛЮЧЕНИЕ

Во время выполнения курсовой работы были выполнены все поставленные задачи.

Создание программы, которая позволяет не только правильно шифровать переданную ей информацию, но и также помогать пользователю разобраться с использованным методом шифрования, узнать его историю создания и изучить основные принципы его работы.

Использование объектно-ориентированного подхода программирования дало возможность удобной реализации разработки шифратора. Но самым важным преимуществом подхода в данной программе является возможность использования патерна «Фабрика».

Применение патерна «Фабрика» позволяет постоянно усовершенствовать программу без особых изменений в коде проекта. А именно добавлять новые алгоритмы шифрования в функционал программы, и, как следствие, увеличивать её мощности.

Изучение среды разработки Qt Creator позволило создать простое и понятное для использования приложения при помощи стандартных библиотек языка С++. Эта среда является отличным способом создания хорошего фронтенда, так как на её основе уже были написаны такие приложения как Skype, Opera, Google Earth и т.д.

Подводя итог, было создано компьютерное приложение, работающее на разных устройствах, однако несмотря на проделанную работу, данное приложение ещё не раскрыло весь свой потенциал. Более детальная проработка пользовательского интерфейса, добавления новых методов шифрования и адаптация приложения для большего количества языков позволит создать мощное учебно-практическое средство защиты информации, которая играет важную составляющую каждого современного человека.

СПИСОК ЛИТЕРАТУРЫ

[1] Шилдт Г. С++, базовый курс, третье издание – Osborne, 2013.

[2] Макс Шлее: Qt 5.10. Профессиональное программирование на C++ – BHV, 2018 – 1072.

[3] Объектно-ориентированное программирование на языке С++: учеб. пособие по курсу «Объектно-ориентир.программирование» для студентов специальности«Вычислит. машины, системы и сети» всех форм обучения Ю.А.Луцик [и др.]. – Мн.: БГУИР, 2003.

[4] Демидович Е.М. «Конструирование программ и языки программирования» учебное пособие для студентов специальности «Вычислительные машины, системы и сети». – Мн.: БГУИР, 2001.

[5] Луцик Ю.А., Лукьянова И.В., Калабухов Е.В., Ковальчук А.М. Электронный учебно-методический комплекс по дисциплине «Конструирование программ и языки программирования» для студентов специальности I-40 02 01 «Вычислительные машины системы и сети» всех форм обучения, БГУИР, Минск, 2011.

[6] Бушкевич А.В. Конструирование программ и языки программирования: метод. указания по курсовому проектированию для студ. спец. I-40 02 01 «Вычислительные машины, системы и сети» всех форм обуч. / сост. А. В. Бушкевич, А. М. Ковальчук, И. В. Лукьянова. – Минск : БГУИР, 2009.

[7] Шилдт Г. Искусство программирования на C++ / Г. Шилд – СПб.: BHM, 2004.

# **ПРИЛОЖЕНИЕ А**

(обязательное)

Схема структурная

# **ПРИЛОЖЕНИЕ Б**

(обязательное)

Диаграмма классов

# **ПРИЛОЖЕНИЕ В**

(обязательное)

Схема алгоритма