

МИНОБРАЗОВАНИЯ РОССИИ
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Кафедра «Информационная безопасность систем и технологий»

Курсовая работа
по дисциплине «Технологии и методы программирования»
на тему «Система проверки лицензии программы оффлайн»
ПГУ.100503.С1115.КР.18ПИ117.01.ПЗ

Специальность – 10.05.03 Информационная безопасность автоматизированных систем
Специализация – Защищенные автоматизированные системы управления

Выполнил студент : И.С. – Нестеров И.С.

Группа: 18ПИ1

Руководитель:

М.Ю. Лупанов М.Ю.

Работа защищена с оценкой хор

Дата защиты 10.06.19

Реферат

Отчет о курсовой работе содержит 41 страница, 3 таблицы, 6 рисунков, 3 источника, 5 приложений.

VISUAL STUDIO, C++, КЛАСС, МЕТОД, ПАРАМЕТРЫ, ФУНКЦИЯ, ДИАГРАММА, МОДУЛЬ, DOXYGEN, ХЭШ, ИМЯ ПОЛЬЗОВАТЕЛЯ, MD5, WINDOWS.

Объектом курсовой работы является изучение и практическое применение программных средств Microsoft Visual Studio.

Цель курсовой работы — разработка модуля проверки лицензионного ключа программы в режиме оффлайн, а так же документирование проекта с использованием пакета Doxygen.

В ходе выполнения курсовой работы была изучена работа с многомодульными проектами, строками, алгоритмами хеширования данных.

В результате выполнения курсовой работы был создан модуль проверки лицензии программы оффлайн.

УТВЕРЖДАЮ

Зав. кафедрой ИИС

к. с. н. с. доцент

Берников С.А.

3.04.2019г.

ЗАДАНИЕ

на курсовую работу

по теме: Система проверки лицензий программы оффлайн

1 Дисциплина: Технологии и методы программирования

2 Студент: Нестеров И.С.

3 Группа: ИСИИ

4 Исходные данные на работу:

4.1 Цель: разработка модуля для проверки лицензий программы без выхода в сеть

Интернет

4.2 Требования к модулю:

- модуль должен обеспечивать выполнение операции проверки лицензий ключа на подлинность;
- модуль должен содержать в себе секретную строку для формирования ключа, на основании которого проводится проверка лицензий;
- тип интерфейса, используемого в программе, – командная строка;
- интерфейс программы должен содержать:
 - имя пользователя;
 - лицензионный ключ;
 - результат проверки;
- модуль должен функционировать в операционной системе Windows 10;
- программа должна быть документирована с использованием пакета документирования Doxygen.

5 Структура работы

5.1 Пояснительная записка (содержание работы):

- словесная модель программы;
- диаграмма вариантов использования и диаграмма классов;
- диаграмма последовательностей и диаграмма деятельности;
- разработка модуля;
- разработка модульных тестов;

разработка функциональных тестов

3.2. Экспериментальная часть

обнаружение дефектов в модуле с помощью модульного тестирования;

- проверка модуля на соответствие техническому заданию с помощью квалификационного тестирования;

6. Календарный и диаграммный планы работы:

- оформление ТЗ к 6 марта 2019;
- разработка диаграмм вариантов использования к 20 марта 2019;

и диаграмм классов

- разработка диаграмм последовательностей и к 3 апреля 2019;

диаграмм деятельности

- разработка модели модуля к 17 апреля 2019;

- разработка модульных тестов и выполнение к 1 мая 2019;
- тестирования

- разработка функциональных тестов и к 15 мая 2019;
- привычное тестирование

- разработка документации к 29 мая 2019;

- оформление отчета о курсовой работе к 29 мая 2019;




Дата защиты работы « » 2019г.

Руководитель работы

Задание получил « » 2019г.

Студент

Проверил/провер

 M.I.O. Дуняев
 N.S. Пестерев
 M.I.O. Дуняев

Содержание

Введение.....	6
1 Пояснительная записка.....	7
1.1 Словесная модель программы.....	7
1.2 Разработка диаграмм вариантов использования и классов.....	7
1.3 Разработка диаграмм последовательностей и деятельности	9
1.4 Разработка программного модуля.....	11
1.5 Разработка модульных тестов.....	12
1.6 Разработка функциональных тестов.....	12
1.7 Документирование модуля.....	13
2 Экспериментальная часть.....	14
2.1 Обнаружение дефектов в модуле с помощью модульного тестирования.....	14
2.2 Проверка модуля на соответствие техническому заданию с помощью квалифицированного тестирования.....	15
Заключение.....	17
Список используемых источников.....	18
Приложение А.....	19
Приложение Б.....	20
Приложение В.....	28
Приложение Г	29
Приложение Д.....	31

Введение

Лицензия на программное обеспечение является юридическим документом, регулирующим использование и распространение программного обеспечения, охраняемого авторским правом. Все программное обеспечение, не находящееся в свободном доступе, защищено авторским правом. Типичная лицензия дает конечному пользователю разрешение на использование одного или более экземпляров программного обеспечения в работе, не влеча за собой нарушение прав издателя, в соответствии с законом об авторском праве. По сути, лицензия действует как обещание от издателей программного обеспечения, не подавать в суд на конечного пользователя, пользующегося правами, принадлежащими издателю программного обеспечения. Лицензирование программного обеспечения, развиваясь, меняет представление о том, как организации и индивидуальные пользователи приобретают и применяют программы. Для компаний используемая ими модель лицензирования может решающим образом повлиять на прибыль. Для этого и используется система проверки лицензии[1].

Объектом исследования курсовой работы является система проверки лицензионного ключа.

Целью курсовой работы является выработка практических навыков разработки многомодульных проектов на языке C++.

Задачей работы является разработка модуля проверки лицензии программы в режиме оффлайн.

Данная задача была решена с помощью программного продукта Microsoft Visual Studio.

1.1 Словесная модель программы

Модуль проверки лицензии в режиме оффлайн содержит в себе реализацию алгоритма хэширования, добавления секретного ключа к строке пользовательских данных, сравнение двух строк.

Для успешной разработки модуля проверки лицензии и выполнения заданий по курсовой работе необходимо было определиться с алгоритмом хэширования, по которому будет вычисляться хэш ключа для дальнейшей работы. Для разработки данного модуля был использован алгоритм хэширования MD5[2].

В качестве секретного ключа модуля была использована строка "18P11".

Для сравнения двух переменных строкового типа был использован встроенный оператор ==.

Для удобства использования модуль был разделен на функции.

Функция `to_hex` была использована для преобразования десятичного числа в его шестнадцатеричный эквивалент.

Функция `get_md5` была использована для реализации алгоритма хэширования MD5.

Функция `keyMaker` была использована для добавления к строке с данными пользователя секретного ключа, а затем преобразования данной строки с помощью функции `get_md5`.

Функция `keyCheck` была использована для проверки созданного по данным пользователя ключа на соответствие входному. Созданный ключ генерируется с использованием функции `keyMaker`.

1.2 Разработка диаграмм вариантов использования и классов

Диаграмма вариантов использования представляет собой последовательность действий (транзакций), выполняемых системой в ответ на событие, инициируемое некоторым внешним объектом (действующим лицом). Вариант использования описывает типичное взаимодействие между пользователем и системой.

Действующее лицо (actor) – это роль, которую пользователь играет по отношению к системе. Действующие лица представляют собой роли, а не конкретных людей или наименования работ. Несмотря на то, что на диаграммах вариантов использования они изображаются в виде стилизованных человеческих фигурок, действующее лицо может также

быть внешней системой, которой необходима некоторая информация от данной системы. Показывать на диаграмме действующих лиц следует только в том случае, когда им действительно необходимы некоторые варианты использования. [3]

Диаграмма вариантов использования для модуля приведена на рисунке 1.

Диаграмма классов — это структурная диаграмма, на которой показано множество классов, интерфейсов, коопераций и отношений между ними. В UML диаграмма классов является типом диаграммы статической структуры. Она описывает структуру системы, показывая её классы, их атрибуты и операторы, и также взаимосвязи этих классов.

Диаграмма классов UML - это граф, узлами которого являются элементы статической структуры проекта (классы, интерфейсы), а дугами - отношения между узлами (ассоциации, наследование, зависимости).[3]

Диаграмма классов приведена на рисунке 2.



Рисунок 1 - Диаграмма вариантов использования

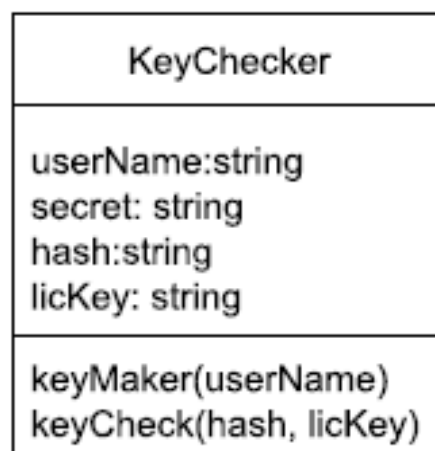


Рисунок 2 - Диаграмма классов

1.3 Разработка диаграмм последовательностей и деятельности

Диаграмма последовательностей отображает взаимодействие объектов в динамике.

Диаграмма последовательностей относится к диаграммам взаимодействия UML, описывающим поведенческие аспекты системы, но рассматривает взаимодействие объектов во времени. Другими словами, диаграмма последовательностей отображает временные особенности передачи и приема сообщений объектами.

Данные диаграммы содержат объекты, которые взаимодействуют в рамках сценария, сообщения, которыми они обмениваются, и возвращаемые результаты, связанные с сообщениями. Впрочем, часто возвращаемые результаты обозначают лишь в том случае, если это не очевидно из контекста.

Объекты обозначаются прямоугольниками с подчеркнутыми именами, сообщения - линиями со стрелками, возвращаемые результаты - пунктирными линиями со стрелками. Прямоугольники на вертикальных линиях под каждым из объектов показывают "время жизни" объектов. Впрочем, довольно часто их не изображают на диаграмме, все это зависит от индивидуального стиля проектирования. [3]

Диаграмма последовательностей приведена на рисунке 3.

Для моделирования процесса выполнения операций в языке UML используются диаграммы деятельности. Диаграмма деятельности - это своеобразная блок-схема, которая описывает последовательность выполнения операций во времени. Их можно использовать для моделирования динамических аспектов поведения системы. Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, а переход в следующее состояние срабатывает только при завершении этой операции в предыдущем состоянии.

В диаграммах деятельности используются пиктограммы "действие", "переход", "выбор" и "линии синхронизации". В языке UML действие изображается в виде прямоугольника с закругленными углами, переходы - в виде направленных стрелок, элементы выбора - в виде ромбов, линии синхронизации - в виде горизонтальных и вертикальных линий.

Состояние действия является специальным случаем состояния с некоторым входным действием и, по крайней мере, одним выходящим из состояния переходом. Когда действие или деятельность в некотором состоянии завершается, поток управления сразу переходит в следующее состояние действия или деятельности. Для описания этого потока используются переходы, показывающие путь из одного состояния действия или деятельности

в другое.

Простые последовательные переходы встречаются наиболее часто, но их одних недостаточно для моделирования любого потока управления. Как и в блок-схеме, вы можете включить в модель выбор, который описывает различные пути выполнения в зависимости от значения некоторого булевского выражения.

Простые и ветвящиеся последовательные переходы в диаграммах деятельности используются чаще всего. Однако можно встретить и параллельные потоки, и это особенно характерно для моделирования бизнес-процессов. В UML для обозначения разделения и слияния таких параллельных потоков выполнения используются линии синхронизации, которые рисуются в виде жирной вертикальной или горизонтальной линии.[3]

Диаграмма последовательностей приведена на рисунке 4.

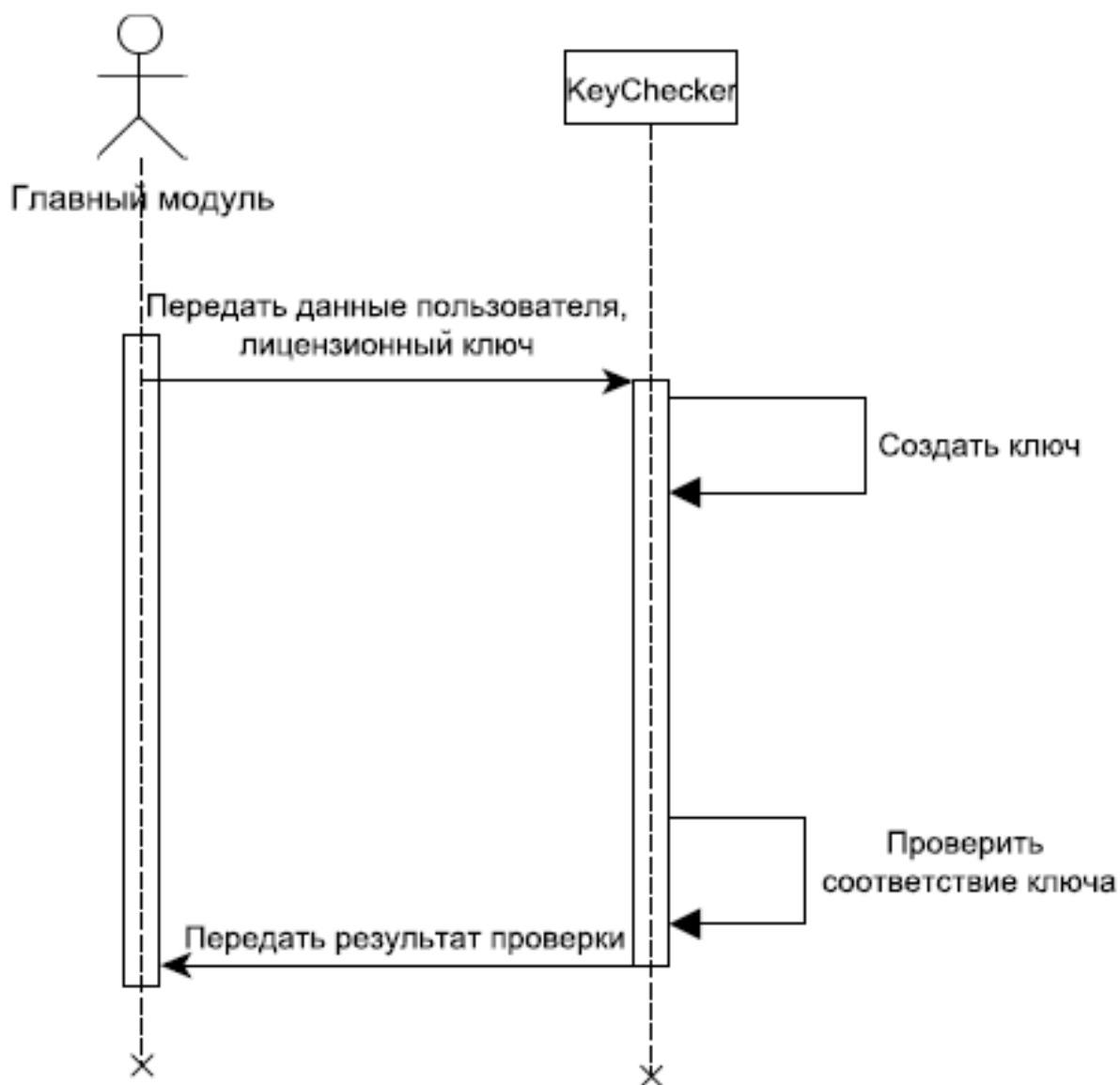


Рисунок 3 - Диаграмма последовательностей

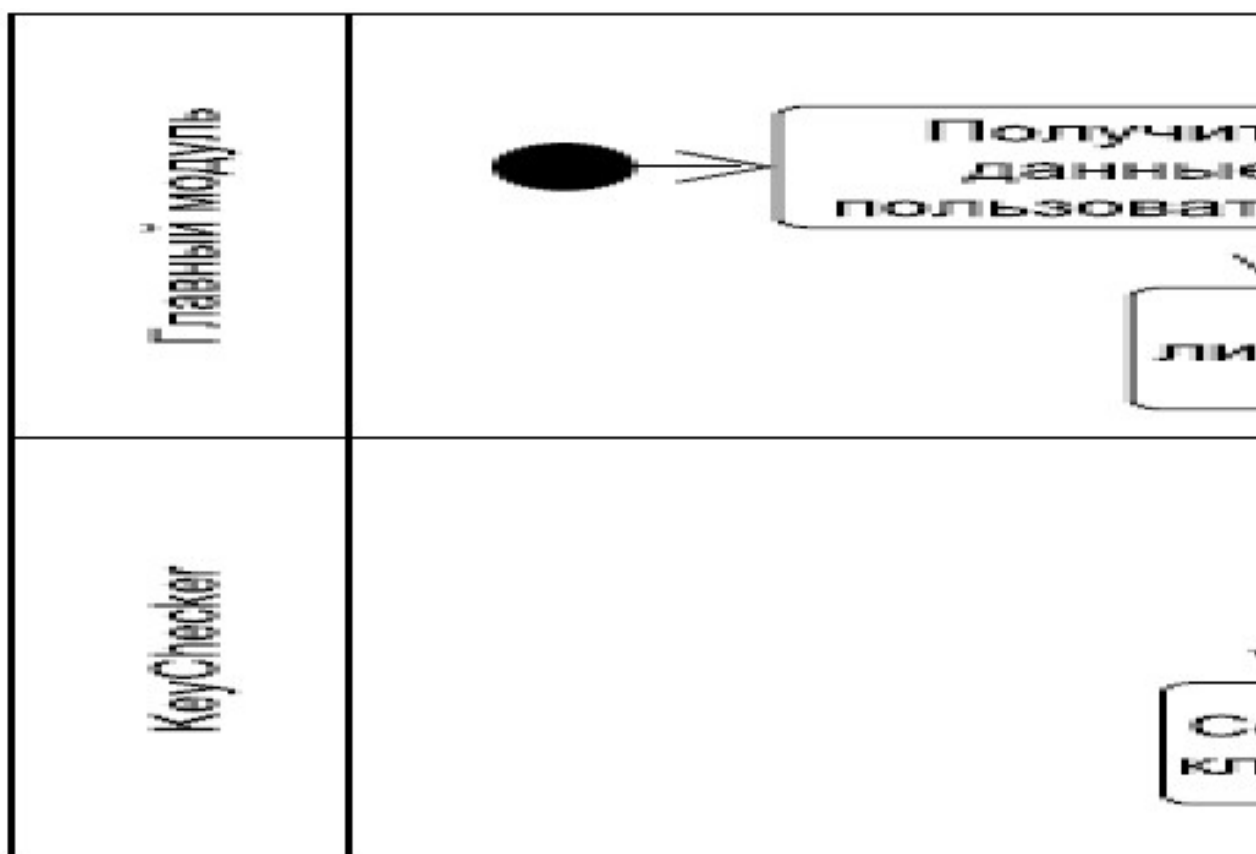


Рисунок 4 - Диаграмма деятельности

1.4 Разработка программного модуля

Модуль проверки лицензии программы в режиме оффлайн был разработан в соответствии со всеми требованиями, представленными в техническом задании. Модуль полностью соответствует словесному описанию, представленному в пункте 1.1. Код заголовочного модуля представлен в приложении А. Код реализации класса представлен в приложении Б.

Для демонстрации работы модуля была разработана демонстрационная программа. Данная программа принимает на вход данные пользователя и лицензионный ключ, вводимые с клавиатуры. Выходным параметром программы является сообщение о результате проверки ключа на подлинность. Код демонстрационной программы представлен в приложении В.

Пример запуска программы представлен на рисунке 5.

```

Input user name
Nesterov
Input key
e10dc11e20281b3c732116bfaebc8adf
Key is true

```

Рисунок 5 - Пример работы модуля

1.5 Разработка модульных тестов

Для проверки работоспособности модуля были разработаны модульные тесты. Результат представлен в таблице 1.

Таблица 1 - Модульные тесты

Тест	Входные данные	Ожидаемый результат
Тест на вводимое имя пользователя	Nesterov e10dc11e20281b3c732116bfaebc8adf	Ключ верный
	Nes7er0v 5aa532ca0d6556265231b2b43998e451	Ключ верный
	Нес7ер0в 1baf5444c536f3424d06e7bce641c269	Ключ верный
	Нес7ер0в_Илья fef05e6a4735e12c7fb7588cf3e26eec	Ключ верный
Тест на вводимый ключ	Nesterov e10dc11e20281b3c732116bfaebc8adf	Ключ верный
	Nesterov e10dc11e20281b	Ключ не верный
	Nesterov e10дц11e20281б3с732116бфаебц8адф	Выдать сообщение об ошибке принятого на вход ключа(наличие русских букв в ключе)

1.6 Разработка функциональных тестов

В связи с тем, что модульные тесты практически совпадают с тестированием функций keyCheck и keyMaker, на данном этапе был разработан тест только для функции get_md5.

Результат представлен в таблице 2.

Таблица 2 -Тестирование функции get_md5

Вводимая строка	Ожидаемый результат
Nesterov	c835ba6f9401329effc49f482c31e009
Nes7er0v	f19d5e97df949f9b71e183dc36409485
Нес7ер0в	c511cf1db5b2b7543126ae1e61a13e04
Нес7ер0в_Илья	fc0b0ed223603adc8b93236f78925b3c

1.7 Документирование модуля

Модуль был задокументирован с использованием программного пакета документирования Doxygen. Результат представлен в приложении Д.

2.1 Обнаружение дефектов в модуле с помощью модульного тестирования

Модуль был протестирован на предмет выявления дефектов. Тесты были произведены на основании модульных и функциональных тестов. Результат представлен в таблице 3.

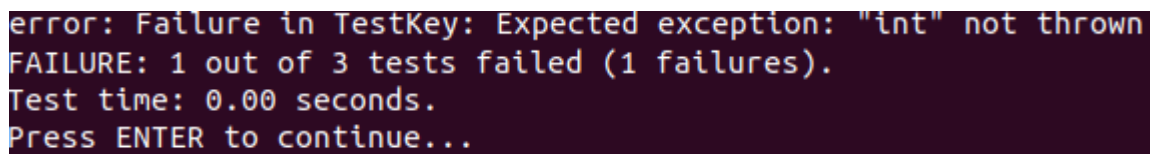
Таблица 3 - Результаты тестирования

Тест	Входные данные	Ожидаемый результат	Полученный результат	Результат тестирования
Тест на вводимое имя пользователя	Nesterov e10dc11e20281b3c 732116bfaebc8adf	Ключ верный	Key is true	Тест пройден
	Nes7er0v 5aa532ca0d655626 5231b2b43998e45 1	Ключ верный	Key is true	Тест пройден
	Нес7ер0в 1baf5444c536f342 4d06e7bce641c269	Ключ верный	Key is true	Тест пройден
	Нес7ер0в_Илья fef05e6a4735e12c7 fb7588cf3e26eec	Ключ верный	Key is true	Тест пройден
Тест на вводимый ключ	Nesterov e10dc11e20281b3c 732116bfaebc8adf	Ключ верный	Key is true	Тест пройден
	Nesterov e10dc11e20281b	Ключ не верный	Key is false	Тест пройден
	Nesterov e10dc11e20281b3c 732116bfaebc8adf	Выдать сообщение об ошибке принятого на вход ключа(наличие русских букв в ключе)	Key is false	Тест пройден
Тест функции get_md5	Nesterov	c835ba6f9401329ef fc49f482c31e009	c835ba6f9401329ef fc49f482c31e009	Тест пройден
	Nes7er0v	f19d5e97df949f9b7 1e183dc36409485	f19d5e97df949f9b7 1e183dc36409485	Тест пройден
	Нес7ер0в	c511cf1db5b2b7543 126ae1e61a13e04	c511cf1db5b2b754 3126ae1e61a13e04	Тест пройден
	Нес7ер0в_Илья	fc0b0ed223603adc8 b93236f78925b3c	fc0b0ed223603adc 8b93236f78925b3c	Тест пройден

2.2 Проверка модуля на соответствие техническому заданию с помощью квалифицированного тестирования.

Для проведения квалифицированного тестирования было использовано юнит-тестирование с использованием библиотеки UnitTest++. Данная библиотека достаточно проста в освоении, и, кроме того, для среды разработки CodeLite есть плагин, интегрирующий библиотеку в среду разработки и позволяющий упростить ее использование.

Была написана программа для проведения тестирования. Код программы приведен в приложении Г. Результат работы программы представлен на рисунке 6.



```
error: Failure in TestKey: Expected exception: "int" not thrown
FAILURE: 1 out of 3 tests failed (1 failures).
Test time: 0.00 seconds.
Press ENTER to continue...
```

Рисунок 6 - Результат тестирования

Заключение

В рамках настоящей курсовой работы был предложен алгоритм реализации модуля проверки лицензионного ключа программы в режиме оффлайн.

На первом этапе было составлено словесное описание модуля.

На втором этапе была разработана диаграммы вариантов использования и классов.

На третьем этапе были составлены диаграммы последовательностей и деятельности.

На четвертом этапе был разработан модуль проверки лицензии в режиме оффлайн.

На пятом этапе была произведена разработка модульных тестов.

На шестом этапе была произведена разработка функциональных тестов.

На седьмом этапе было проведено обнаружение дефектов в модуле с помощью тестирования.

Все поставленные перед разработчиком задачи были выполнены в полном объеме.

Список используемых источников

1. Лицензирование программного обеспечения// habr.com. - 2016.
URL: <https://habr.com/ru/post/275995/>
2. Хэш-функция MD5// habr.com. - 2011. URL: <https://habr.com/ru/sandbox/26876/>
3. UML диаграммы URL: http://book.uml3.ru/sec_1_5

Приложение А

(обязательное)

Код заголовочного модуля

```
#pragma once
#include <string>
#include <iostream>
#include <cmath>

class KeyChecker
{
private:
    std::string to_hex(unsigned int value);
    unsigned int F(unsigned int X, unsigned int Y, unsigned
int Z);
    unsigned int G(unsigned int X, unsigned int Y, unsigned
int Z);
    unsigned int H(unsigned int X, unsigned int Y, unsigned
int Z);
    unsigned int I(unsigned int X, unsigned int Y, unsigned
int Z);
    unsigned int rotate_left(unsigned int value, int shift);
    std::string get_md5(std::string in);
    std::string secret="18PI1", hash;
    std::string keyMaker(std::string str);
public:
    bool keyCheck(std::string userName, std::string licKey);
};
```

Приложение Б

(обязательное)

Код реализации класса

```
#include "LicCheck.h"

unsigned int KeyChecker::F(unsigned int X, unsigned int Y,
unsigned int Z) { return ((X & Y) | ((~X) & Z)); }

unsigned int KeyChecker::G(unsigned int X, unsigned int Y,
unsigned int Z) { return (X & Z) | (Y & (~Z)); }

unsigned int KeyChecker::H(unsigned int X, unsigned int Y,
unsigned int Z) { return X ^ Y ^ Z; }

unsigned int KeyChecker::I(unsigned int X, unsigned int Y,
unsigned int Z) { return Y ^ (X | (~Z)); }

unsigned int KeyChecker::rotate_left(unsigned int value, int
shift) { return value << shift | value >> (32 - shift); }

std::string KeyChecker::to_hex(unsigned int value)
{
    std::string out;
    unsigned char hex;
    char hex_res[3];
    while (value)
    {
        hex = value % 256;
        _itoa_s(hex, hex_res, 16);
        if (hex_res[1] == '\\0')
        {
            hex_res[1] = hex_res[0];
            hex_res[0] = '0';
            hex_res[2] = '\\0';
        }
        out.append(hex_res);
        value /= 256;
    }
}
```

```

        return out;
    }

std::string KeyChecker::get_md5(std::string in)
{
    int length = in.length();
    int rest = length % 64;
    int size = 0;

    if (rest < 56)
        size = length - rest + 56 + 8;
    else
        size = length + 64 - rest + 56 + 8;

    unsigned char* msg_for_decode = new unsigned char[size];

    for (int i = 0; i < length; i++)
        msg_for_decode[i] = in[i];
    msg_for_decode[length] = 0x80;
    for (int i = length + 1; i < size; i++)
        msg_for_decode[i] = 0;

    __int64 bit_length = (unsigned int)(length) * 8;

    for (int i = 0; i < 8; i++)
        msg_for_decode[size - 8 + i] = (unsigned char)
(bit_length >> i * 8);

    unsigned int A = 0x67452301, B = 0xefcdab89, C =
0x98badcfe, D = 0x10325476;
    unsigned int T[64];

    for (int i = 0; i < 64; i++)
        T[i] = unsigned int(pow(2, 32) * fabs(sin(i + 1)));

```

```

unsigned int* X = (unsigned int*)(msg_for_decode);

unsigned int AA = 0, BB = 0, CC = 0, DD = 0;

for (int i = 0; i < size / 4; i += 16) {
    AA = A; BB = B; CC = C; DD = D;

    A = B + rotate_left((A + F(B, C, D) + X[i + 0] +
T[0]), 7);
    D = A + rotate_left((D + F(A, B, C) + X[i + 1] +
T[1]), 12);
    C = D + rotate_left((C + F(D, A, B) + X[i + 2] +
T[2]), 17);
    B = C + rotate_left((B + F(C, D, A) + X[i + 3] +
T[3]), 22);

    A = B + rotate_left((A + F(B, C, D) + X[i + 4] +
T[4]), 7);
    D = A + rotate_left((D + F(A, B, C) + X[i + 5] +
T[5]), 12);
    C = D + rotate_left((C + F(D, A, B) + X[i + 6] +
T[6]), 17);
    B = C + rotate_left((B + F(C, D, A) + X[i + 7] +
T[7]), 22);

    A = B + rotate_left((A + F(B, C, D) + X[i + 8] +
T[8]), 7);
    D = A + rotate_left((D + F(A, B, C) + X[i + 9] +
T[9]), 12);
    C = D + rotate_left((C + F(D, A, B) + X[i + 10] +
T[10]), 17);
    B = C + rotate_left((B + F(C, D, A) + X[i + 11] +
T[11]), 22);

```

```

        A = B + rotate_left((A + F(B, C, D) + X[i + 12] +
T[12]), 7);
        D = A + rotate_left((D + F(A, B, C) + X[i + 13] +
T[13]), 12);
        C = D + rotate_left((C + F(D, A, B) + X[i + 14] +
T[14]), 17);
        B = C + rotate_left((B + F(C, D, A) + X[i + 15] +
T[15]), 22);

```

```

        A = B + rotate_left((A + G(B, C, D) + X[i + 1] +
T[16]), 5);
        D = A + rotate_left((D + G(A, B, C) + X[i + 6] +
T[17]), 9);
        C = D + rotate_left((C + G(D, A, B) + X[i + 11] +
T[18]), 14);
        B = C + rotate_left((B + G(C, D, A) + X[i + 0] +
T[19]), 20);

```

```

        A = B + rotate_left((A + G(B, C, D) + X[i + 5] +
T[20]), 5);
        D = A + rotate_left((D + G(A, B, C) + X[i + 10] +
T[21]), 9);
        C = D + rotate_left((C + G(D, A, B) + X[i + 15] +
T[22]), 14);
        B = C + rotate_left((B + G(C, D, A) + X[i + 4] +
T[23]), 20);

```

```

        A = B + rotate_left((A + G(B, C, D) + X[i + 9] +
T[24]), 5);
        D = A + rotate_left((D + G(A, B, C) + X[i + 14] +
T[25]), 9);
        C = D + rotate_left((C + G(D, A, B) + X[i + 3] +

```

```

T[26]), 14);

    B = C + rotate_left((B + G(C, D, A) + X[i + 8] +
T[27]), 20);

    A = B + rotate_left((A + G(B, C, D) + X[i + 13] +
T[28]), 5);

    D = A + rotate_left((D + G(A, B, C) + X[i + 2] +
T[29]), 9);

    C = D + rotate_left((C + G(D, A, B) + X[i + 7] +
T[30]), 14);

    B = C + rotate_left((B + G(C, D, A) + X[i + 12] +
T[31]), 20);

    A = B + rotate_left((A + H(B, C, D) + X[i + 5] +
T[32]), 4);

    D = A + rotate_left((D + H(A, B, C) + X[i + 8] +
T[33]), 11);

    C = D + rotate_left((C + H(D, A, B) + X[i + 11] +
T[34]), 16);

    B = C + rotate_left((B + H(C, D, A) + X[i + 14] +
T[35]), 23);

    A = B + rotate_left((A + H(B, C, D) + X[i + 1] +
T[36]), 4);

    D = A + rotate_left((D + H(A, B, C) + X[i + 4] +
T[37]), 11);

    C = D + rotate_left((C + H(D, A, B) + X[i + 7] +
T[38]), 16);

    B = C + rotate_left((B + H(C, D, A) + X[i + 10] +
T[39]), 23);

    A = B + rotate_left((A + H(B, C, D) + X[i + 13] +
T[40]), 4);

```

```

D = A + rotate_left((D + H(A, B, C) + X[i + 0] +
T[41]), 11);
C = D + rotate_left((C + H(D, A, B) + X[i + 3] +
T[42]), 16);
B = C + rotate_left((B + H(C, D, A) + X[i + 6] +
T[43]), 23);

A = B + rotate_left((A + H(B, C, D) + X[i + 9] +
T[44]), 4);
D = A + rotate_left((D + H(A, B, C) + X[i + 12] +
T[45]), 11);
C = D + rotate_left((C + H(D, A, B) + X[i + 15] +
T[46]), 16);
B = C + rotate_left((B + H(C, D, A) + X[i + 2] +
T[47]), 23);

A = B + rotate_left((A + I(B, C, D) + X[i + 0] +
T[48]), 6);
D = A + rotate_left((D + I(A, B, C) + X[i + 7] +
T[49]), 10);
C = D + rotate_left((C + I(D, A, B) + X[i + 14] +
T[50]), 15);
B = C + rotate_left((B + I(C, D, A) + X[i + 5] +
T[51]), 21);

A = B + rotate_left((A + I(B, C, D) + X[i + 12] +
T[52]), 6);
D = A + rotate_left((D + I(A, B, C) + X[i + 3] +
T[53]), 10);
C = D + rotate_left((C + I(D, A, B) + X[i + 10] +
T[54]), 15);
B = C + rotate_left((B + I(C, D, A) + X[i + 1] +
T[55]), 21);

```



```

        A = B + rotate_left((A + I(B, C, D) + X[i + 8] +
T[56]), 6);
        D = A + rotate_left((D + I(A, B, C) + X[i + 15] +
T[57]), 10);
        C = D + rotate_left((C + I(D, A, B) + X[i + 6] +
T[58]), 15);
        B = C + rotate_left((B + I(C, D, A) + X[i + 13] +
T[59]), 21);

        A = B + rotate_left((A + I(B, C, D) + X[i + 4] +
T[60]), 6);
        D = A + rotate_left((D + I(A, B, C) + X[i + 11] +
T[61]), 10);
        C = D + rotate_left((C + I(D, A, B) + X[i + 2] +
T[62]), 15);
        B = C + rotate_left((B + I(C, D, A) + X[i + 9] +
T[63]), 21);

        A += AA;
        B += BB;
        C += CC;
        D += DD;
    }
    delete[]msg_for_decode;
    std::string res = to_hex(A) + to_hex(B) + to_hex(C) +
to_hex(D);
    return res;
}

std::string KeyChecker::keyMaker(std::string str)
{
    str += secret;
    return get_md5(str);
}

```

```
    }

    bool KeyChecker::keyCheck(std::string userName, std::string
licKey)
    {
        hash = keyMaker(userName);
        if (hash == licKey)
            return true;
        else return false;
    }
```

Приложение В

(обязательное)

Код демонстрационной программы

```
#include "LicCheck.h"

int main()
{
    std::string s, k;
    std::cout << "Input user name" << std::endl;
    std::cin >> s;
    std::cout << "Input key" << std::endl;
    std::cin >> k;
    KeyChecker q;
    if (q.keyCheck(s, k))
        std::cout << "Key is true" << std::endl;
    else std::cout << "Key is false" << std::endl;
}
```

Приложение Г

(обязательное)

Код юнит-тестирования

```
#include <UnitTest++/UnitTest++.h>
#include "LicCheck.h"
TEST (UserName)
{
    CHECK_EQUAL(true, KeyChecker().keyCheck("Nesterov",
"e10dc11e20281b3c732116bfaebc8adf"));
    CHECK_EQUAL(true, KeyChecker().keyCheck("Nes7er0v",
"5aa532ca0d6556265231b2b43998e451"));
    CHECK_EQUAL(true, KeyChecker().keyCheck("Нес7ер0в",
"1baf5444c536f3424d06e7bce641c269"));
    CHECK_EQUAL(true, KeyChecker().keyCheck("Нес7ер0в_Илья",
"fef05e6a4735e12c7fb7588cf3e26eec"));
}
TEST (TestKey)
{
    CHECK_EQUAL(true, KeyChecker().keyCheck("Nesterov",
"e10dc11e20281b3c732116bfaebc8adf"));
    CHECK_EQUAL(false, KeyChecker().keyCheck("Nesterov",
"e10dc11e20281b"));
    CHECK_THROW(KeyChecker().keyCheck("Nesterov",
"e10дц11e20281б3c732116бфаебц8адф"), int);
}
TEST (FuncMD5)
{
    CHECK_EQUAL("c835ba6f9401329effc49f482c31e009", md5("Nesterov"))
);
    CHECK_EQUAL("f19d5e97df949f9b71e183dc36409485", md5("Nes7er0v"))
);
    CHECK_EQUAL("c511cf1db5b2b7543126ae1e61a13e04", md5("Нес7ер0в"))
);
    CHECK_EQUAL("fc0b0ed223603adc8b93236f78925b3c", md5("Нес7ер0в_И
```

```
    лья" ) );  
    }  
    int main(int argc, char **argv)  
    {  
        return UnitTest::RunAllTests();  
    }
```

Приложение Д

(обязательное)

Документация проекта с использованием пакета Doxygen

License Cheking Proram

Generated by Doxygen 1.8.14

Contents

1	License checking module	1
2	Class Documentation	1
2.1	KeyChecker Class Reference	1
2.1.1	Member Function Documentation	2
2.1.2	Member Data Documentation	7
3	File Documentation	7
3.1	LicCheck/LicCheck.cpp File Reference	7
3.2	LicCheck/LicCheck.h File Reference	7
3.3	LicCheck/LicCheckMain.cpp File Reference	7
3.3.1	Function Documentation	8
	Index	9

1 License checking module

Author

Nesterov 18PI1

2 Class Documentation

2.1 KeyChecker Class Reference

```
#include <LicCheck.h>
```

Public Member Functions

- bool [keyCheck](#) (std::string userName, std::string licKey)
Function keyCheck: The function is used ckeck license key.

Private Member Functions

- `std::string to_hex` (unsigned int value)
Function to_hex: The function is used to transform symbol into the hex equivalent.
- `unsigned int F` (unsigned int X, unsigned int Y, unsigned int Z)
Function F: The function is used in md5 algorithm.
- `unsigned int G` (unsigned int X, unsigned int Y, unsigned int Z)
Function G: The function is used in md5 algorithm.
- `unsigned int H` (unsigned int X, unsigned int Y, unsigned int Z)
Function H: The function is used in md5 algorithm.
- `unsigned int I` (unsigned int X, unsigned int Y, unsigned int Z)
Function I: The function is used in md5 algorithm.
- `unsigned int rotate_left` (unsigned int value, int shift)
Function rotate_left: The function is used in md5 algorithm.
- `std::string get_md5` (std::string in)
Function get_md5: The function is used to transform the text into hash string using md5 algorithm.
- `std::string keyMaker` (std::string str)
Function keyMaker: The function is used to mske license key.

Private Attributes

- `std::string secret` = "18PI1"
Secret key: Secret key, used to make license key.
- `std::string hash`
Hash: Hash string, contained license key.

2.1.1 Member Function Documentation

2.1.1.1 F()

```
unsigned int KeyChecker::F (
    unsigned int X,
    unsigned int Y,
    unsigned int Z ) [private]
```

Function F: The function is used in md5 algorithm.

```
unsigned int KeyChecker::F(unsigned int X, unsigned int Y, unsigned int Z) { return ((X & Y) |
    ((~X) & Z)); }
```

2.1.1.2 G()

```
unsigned int KeyChecker::G (
    unsigned int X,
    unsigned int Y,
    unsigned int Z ) [private]
```

Function G: The function is used in md5 algorithm.

```
unsigned int KeyChecker::G(unsigned int X, unsigned int Y, unsigned int Z) { return (X & Z) |
    (Y & (~Z)); }
```

2.1.1.3 get_md5()

```
std::string KeyChecker::get_md5 (
    std::string in ) [private]
```

Function get_md5: The function is used to transform the text into hash string using md5 algorithm.

Returns

Hash string

```
std::string KeyChecker::get_md5(std::string in)
{
    int length = in.length();
    int rest = length % 64;
    int size = 0;

    if (rest < 56)
        size = length - rest + 56 + 8;
    else
        size = length + 64 - rest + 56 + 8;

    unsigned char* msg_for_decode = new unsigned char[size];

    for (int i = 0; i < length; i++)
        msg_for_decode[i] = in[i];
    msg_for_decode[length] = 0x80;
    for (int i = length + 1; i < size; i++)
        msg_for_decode[i] = 0;

    __int64 bit_length = (unsigned int)(length) * 8;

    for (int i = 0; i < 8; i++)
        msg_for_decode[size - 8 + i] = (unsigned char)(bit_length >> i * 8);

    unsigned int A = 0x67452301, B = 0xefcdab89, C = 0x98badcfe, D = 0x10325476;
    unsigned int T[64];

    for (int i = 0; i < 64; i++)
        T[i] = unsigned int(pow(2, 32) * fabs(sin(i + 1)));

    unsigned int* X = (unsigned int*)(msg_for_decode);

    unsigned int AA = 0, BB = 0, CC = 0, DD = 0;

    for (int i = 0; i < size / 4; i += 16) {
        AA = A; BB = B; CC = C; DD = D;

        A = B + rotate_left((A + F(B, C, D) + X[i + 0] + T[0]), 7);
        D = A + rotate_left((D + F(A, B, C) + X[i + 1] + T[1]), 12);
        C = D + rotate_left((C + F(D, A, B) + X[i + 2] + T[2]), 17);
        B = C + rotate_left((B + F(C, D, A) + X[i + 3] + T[3]), 22);

        A = B + rotate_left((A + F(B, C, D) + X[i + 4] + T[4]), 7);
        D = A + rotate_left((D + F(A, B, C) + X[i + 5] + T[5]), 12);
        C = D + rotate_left((C + F(D, A, B) + X[i + 6] + T[6]), 17);
        B = C + rotate_left((B + F(C, D, A) + X[i + 7] + T[7]), 22);

        A = B + rotate_left((A + F(B, C, D) + X[i + 8] + T[8]), 7);
```

```

D = A + rotate_left((D + F(A, B, C) + X[i + 9] + T[9]), 12);
C = D + rotate_left((C + F(D, A, B) + X[i + 10] + T[10]), 17);
B = C + rotate_left((B + F(C, D, A) + X[i + 11] + T[11]), 22);

A = B + rotate_left((A + F(B, C, D) + X[i + 12] + T[12]), 7);
D = A + rotate_left((D + F(A, B, C) + X[i + 13] + T[13]), 12);
C = D + rotate_left((C + F(D, A, B) + X[i + 14] + T[14]), 17);
B = C + rotate_left((B + F(C, D, A) + X[i + 15] + T[15]), 22);

A = B + rotate_left((A + G(B, C, D) + X[i + 1] + T[16]), 5);
D = A + rotate_left((D + G(A, B, C) + X[i + 6] + T[17]), 9);
C = D + rotate_left((C + G(D, A, B) + X[i + 11] + T[18]), 14);
B = C + rotate_left((B + G(C, D, A) + X[i + 0] + T[19]), 20);

A = B + rotate_left((A + G(B, C, D) + X[i + 5] + T[20]), 5);
D = A + rotate_left((D + G(A, B, C) + X[i + 10] + T[21]), 9);
C = D + rotate_left((C + G(D, A, B) + X[i + 15] + T[22]), 14);
B = C + rotate_left((B + G(C, D, A) + X[i + 4] + T[23]), 20);

A = B + rotate_left((A + G(B, C, D) + X[i + 9] + T[24]), 5);
D = A + rotate_left((D + G(A, B, C) + X[i + 14] + T[25]), 9);
C = D + rotate_left((C + G(D, A, B) + X[i + 3] + T[26]), 14);
B = C + rotate_left((B + G(C, D, A) + X[i + 8] + T[27]), 20);

A = B + rotate_left((A + G(B, C, D) + X[i + 13] + T[28]), 5);
D = A + rotate_left((D + G(A, B, C) + X[i + 2] + T[29]), 9);
C = D + rotate_left((C + G(D, A, B) + X[i + 7] + T[30]), 14);
B = C + rotate_left((B + G(C, D, A) + X[i + 12] + T[31]), 20);

A = B + rotate_left((A + H(B, C, D) + X[i + 5] + T[32]), 4);
D = A + rotate_left((D + H(A, B, C) + X[i + 8] + T[33]), 11);
C = D + rotate_left((C + H(D, A, B) + X[i + 11] + T[34]), 16);
B = C + rotate_left((B + H(C, D, A) + X[i + 14] + T[35]), 23);

A = B + rotate_left((A + H(B, C, D) + X[i + 1] + T[36]), 4);
D = A + rotate_left((D + H(A, B, C) + X[i + 4] + T[37]), 11);
C = D + rotate_left((C + H(D, A, B) + X[i + 7] + T[38]), 16);
B = C + rotate_left((B + H(C, D, A) + X[i + 10] + T[39]), 23);

A = B + rotate_left((A + H(B, C, D) + X[i + 13] + T[40]), 4);
D = A + rotate_left((D + H(A, B, C) + X[i + 0] + T[41]), 11);
C = D + rotate_left((C + H(D, A, B) + X[i + 3] + T[42]), 16);
B = C + rotate_left((B + H(C, D, A) + X[i + 6] + T[43]), 23);

A = B + rotate_left((A + H(B, C, D) + X[i + 9] + T[44]), 4);
D = A + rotate_left((D + H(A, B, C) + X[i + 12] + T[45]), 11);
C = D + rotate_left((C + H(D, A, B) + X[i + 15] + T[46]), 16);
B = C + rotate_left((B + H(C, D, A) + X[i + 2] + T[47]), 23);

A = B + rotate_left((A + I(B, C, D) + X[i + 0] + T[48]), 6);
D = A + rotate_left((D + I(A, B, C) + X[i + 7] + T[49]), 10);
C = D + rotate_left((C + I(D, A, B) + X[i + 14] + T[50]), 15);
B = C + rotate_left((B + I(C, D, A) + X[i + 5] + T[51]), 21);

A = B + rotate_left((A + I(B, C, D) + X[i + 12] + T[52]), 6);
D = A + rotate_left((D + I(A, B, C) + X[i + 3] + T[53]), 10);
C = D + rotate_left((C + I(D, A, B) + X[i + 10] + T[54]), 15);
B = C + rotate_left((B + I(C, D, A) + X[i + 1] + T[55]), 21);

A = B + rotate_left((A + I(B, C, D) + X[i + 8] + T[56]), 6);
D = A + rotate_left((D + I(A, B, C) + X[i + 15] + T[57]), 10);
C = D + rotate_left((C + I(D, A, B) + X[i + 6] + T[58]), 15);
B = C + rotate_left((B + I(C, D, A) + X[i + 13] + T[59]), 21);

A = B + rotate_left((A + I(B, C, D) + X[i + 4] + T[60]), 6);
D = A + rotate_left((D + I(A, B, C) + X[i + 11] + T[61]), 10);
C = D + rotate_left((C + I(D, A, B) + X[i + 2] + T[62]), 15);
B = C + rotate_left((B + I(C, D, A) + X[i + 9] + T[63]), 21);

A += AA;
B += BB;
C += CC;
D += DD;
}
delete[msg_for_decode;
std::string res = to_hex(A) + to_hex(B) + to_hex(C) +
    to_hex(D);
return res;
}

```

2.1.1.4 H()

```
unsigned int KeyChecker::H (
    unsigned int X,
    unsigned int Y,
    unsigned int Z ) [private]
```

Function H: The function is used in md5 algorithm.

```
unsigned int KeyChecker::H(unsigned int X, unsigned int Y, unsigned int Z) { return X ^ Y ^ Z;
}
```

2.1.1.5 I()

```
unsigned int KeyChecker::I (
    unsigned int X,
    unsigned int Y,
    unsigned int Z ) [private]
```

Function I: The function is used in md5 algorithm.

```
unsigned int KeyChecker::I(unsigned int X, unsigned int Y, unsigned int Z) { return Y ^ (X | (
    ~Z)); }
```

2.1.1.6 keyCheck()

```
bool KeyChecker::keyCheck (
    std::string userName,
    std::string licKey )
```

Function keyCheck: The function is used ckeck license key.

Returns

bool value: True, if key is correct, False, otherwise

```
bool KeyChecker::keyCheck(std::string userName, std::string licKey)
{
    hash = keyMaker(userName);
    if (hash == licKey)
        return true;
    else return false;
}
```

2.1.1.7 keyMaker()

```
std::string KeyChecker::keyMaker (
    std::string str ) [private]
```

Function keyMaker: The function is used to mske license key.

Returns

Lisense key string

```
std::string KeyChecker::keyMaker(std::string str)
{
    str += secret;
    return get_md5(str);
}
```

2.1.1.8 rotate_left()

```
unsigned int KeyChecker::rotate_left (
    unsigned int value,
    int shift ) [private]
```

Function rotate_left: The function is used in md5 algorithm.

```
unsigned int KeyChecker::rotate_left(unsigned int value, int shift) { return value
    << shift | value >> (32 - shift); }
```

2.1.1.9 to_hex()

```
std::string KeyChecker::to_hex (
    unsigned int value ) [private]
```

Function to_hex: The function is used to transform symbol into the hex equivalent.

Returns

Hex equivalent string

```
std::string KeyChecker::to_hex(unsigned int value)
{
    std::string out;
    unsigned char hex;
    char hex_res[3];
    while (value)
    {
        hex = value % 256;
        _itoa_s(hex, hex_res, 16);
        if (hex_res[1] == '\0')
        {
            hex_res[1] = hex_res[0];
            hex_res[0] = '0';
            hex_res[2] = '\0';
        }
        out.append(hex_res);
        value /= 256;
    }
    return out;
}
```

2.1.2 Member Data Documentation

2.1.2.1 hash

```
std::string KeyChecker::hash [private]
```

Hash: Hash string, contained license key.

2.1.2.2 secret

```
std::string KeyChecker::secret = "18PI1" [private]
```

Secret key: Secret key, used to make license key.

The documentation for this class was generated from the following files:

- [LicCheck/LicCheck.h](#)
- [LicCheck/LicCheck.cpp](#)

3 File Documentation

3.1 LicCheck/LicCheck.cpp File Reference

```
#include "LicCheck.h"
```

3.2 LicCheck/LicCheck.h File Reference

```
#include <string>
#include <iostream>
#include <cmath>
```

Classes

- class [KeyChecker](#)

3.3 LicCheck/LicCheckMain.cpp File Reference

```
#include "LicCheck.h"
```

Functions

- `int main()`

Function main.

3.3.1 Function Documentation

3.3.1.1 main()

```
int main ( )
```

Function main.

The function is used to demonstrate module's work

```
int main()
{
    std::string s, k;
    std::cout << "Input user name" << std::endl;
    std::cin >> s;
    std::cout << "Input key" << std::endl;
    std::cin >> k;
    KeyChecker q;
    if (q.keyCheck(s, k))
        std::cout << "Key is true" << std::endl;
    else std::cout << "Key is false" << std::endl;
}
```

Index

F

KeyChecker, [2](#)

G

KeyChecker, [2](#)

get_md5

KeyChecker, [3](#)

H

KeyChecker, [4](#)

hash

KeyChecker, [7](#)

I

KeyChecker, [5](#)

keyCheck

KeyChecker, [5](#)

KeyChecker, [1](#)

F, [2](#)

G, [2](#)

get_md5, [3](#)

H, [4](#)

hash, [7](#)

I, [5](#)

keyCheck, [5](#)

keyMaker, [5](#)

rotate_left, [6](#)

secret, [7](#)

to_hex, [6](#)

keyMaker

KeyChecker, [5](#)

LicCheck/LicCheck.cpp, [7](#)

LicCheck/LicCheck.h, [7](#)

LicCheck/LicCheckMain.cpp, [7](#)

LicCheckMain.cpp

main, [8](#)

main

LicCheckMain.cpp, [8](#)

rotate_left

KeyChecker, [6](#)

secret

KeyChecker, [7](#)

to_hex

KeyChecker, [6](#)