|  |
| --- |
| Сопроводительный документ |
| SQLObfuscator |
| https://github.com/BadCatsss/SQLObfuscator |

|  |
| --- |
| 1.1.2019 |

Оглавление

[Введение 2](#_Toc21198252)

[Приложение SQLObfuscator 2](#_Toc21198253)

[Назначение приложения 2](#_Toc21198254)

[Принцип работы 2](#_Toc21198255)

[Особенности использования 2](#_Toc21198256)

[Структура Базы данных 2](#_Toc21198257)

[Выполняемые операции 2](#_Toc21198258)

[Пример работы приложения 3](#_Toc21198259)

[Технические подробности 6](#_Toc21198260)

[Файл Obfuscator.cpp 6](#_Toc21198261)

[Подробности реализации 9](#_Toc21198262)

[Obfuscator 9](#_Toc21198263)

[addToErrorList 9](#_Toc21198264)

[printErrorList 10](#_Toc21198265)

[openDatabaseFile 10](#_Toc21198266)

[createConnection 10](#_Toc21198267)

[addOperators 11](#_Toc21198268)

[changeForeginKey 11](#_Toc21198269)

[changeLacCid 12](#_Toc21198270)

[changeCellName 12](#_Toc21198271)

[changeAdress 12](#_Toc21198272)

[relocatePosition 13](#_Toc21198273)

[makeObfuscation 13](#_Toc21198274)

[Файл Obfuscator.h 14](#_Toc21198275)

[Подробности реализации 15](#_Toc21198276)

[Файл main.cpp 17](#_Toc21198277)

[Теоретические сведения 18](#_Toc21198278)

[Коэффициент для координат широты: 19](#_Toc21198279)

[Коэффициент для координат долготы: 19](#_Toc21198280)

## Введение

## Приложение SQLObfuscator

## Назначение приложения

Приложение SQLObfuscator создано с целью автоматического произведения процедуры обфускации для заданного файла база данных, при помощи средств языка c++ и фреймворка Qt.

## Принцип работы

Приложение не имеет оконного графического интерфейса, предоставляя, взамен, для коммуникации и обмена данными с пользователем – интерфейс командной строки.

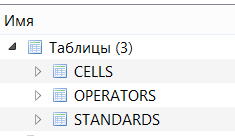
# Особенности использования

Приложение работает с файлами с расширением “.db”.



## Структура Базы данных

Предпологается, что база данных, с которой будет работать приложение SQLObfuscator – имеет следующую структуру:

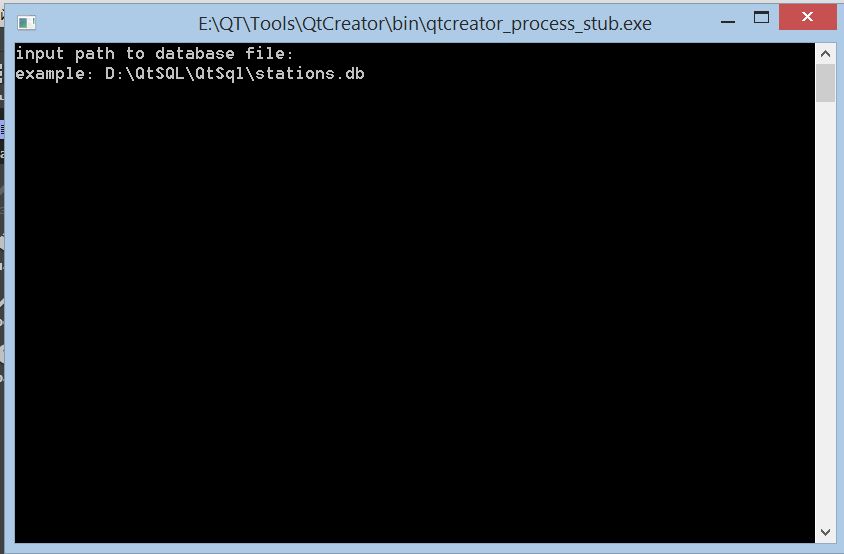


Так же, предполагается, что таблица CELLS имеет следующие столбцы: LAC, CELLID,CELL\_NAME,ADR,LAT,LON.

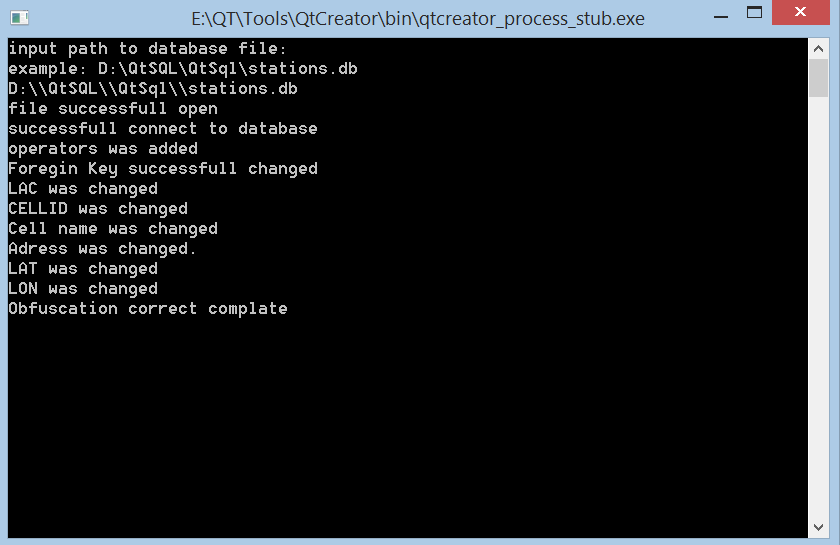
## Выполняемые операции

* Добавление операторов в таблицу OPERATORS
* Замена внешнего ключа в таблице CELLS
* Замена значений столбцов LAC и CELLID в таблице CELLS
* Замена значения столбца CELL\_NAME в таблице CELLS
* Замена значения столбца ADR в таблице CELLS
* Замена значений столбцов LAT и LON в таблице CELLS

## Пример работы приложения

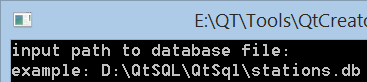


Стартовый диалог приложения.

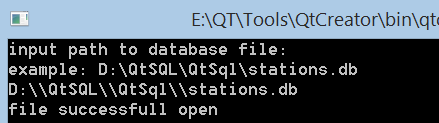


Приложение при процессе обфускации.

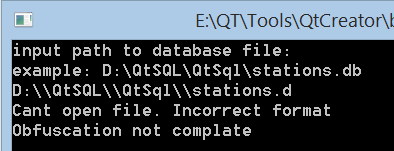
При старте приложения, для пользователя будет выведено диалоговое сообщение, предлагающее ввести путь к файлу базы данных.



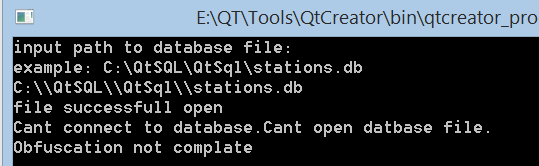
В случае, корректности введенных данных – будет выведено сообщение об успешном открытии файла:



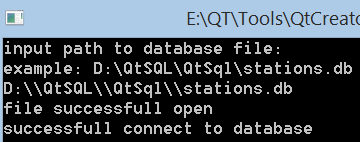
- иначе, пользователь получит сообщение об ошибке, указывающее на невозможность открытия указанного файла и о том, что процедура обфускации не завершена – в случае неверно указанного формата файла, не соответствующего расширению .db:



Или же, при верно указанном расширении, отсевающем файле по указанному пути – пользователь получит сообщение об успешности открытия файла – т.к его расширение соответствует требуемому, но получит сообщение об ошибке на этапе пробного запроса к файлу базы – с целью проверки возможности чтения из нее:



- в противном случае, если указанное расширение файла – корректно и по указанному пути – существует требуемый файл, пользователь получит сообщение о успешном открытии и подключении (отправке пробного запроса) к файлу.



После открытия и подключения к файлу базы данных – приложение начнет выполнять процедуру обфускации, производя замену данных для требуемых записей в соответствующей таблице (см. [выполняемые операции](#_Выполняемые_операции)).

В случае успешности/не успешности проведения одной из операции – пользователь получит соответствующее сообщение:

* Добавление операторов в таблицу OPERATORS:

Успех завершения операции:





* Замена внешнего ключа в таблице CELLS:

Успех завершения операции:



Операция не завершена:



* Замена значений столбцов LAC и CELLID в таблице CELLS:

Успех завершения операции:



Операция не завершена:

Для LAC:



Для CELLID:



* Замена значения столбца CELL\_NAME в таблице CELLS:

Успех завершения операции:



Операция не завершена:



* Замена значения столбца ADR в таблице CELLS:

Успех завершения операции:



Операция не завершена:



* Замена значений столбцов LAT и LON в таблице CELLS:
* Успех завершения операции:



Операция не завершена:

Для LAT или LON:



Финальное сообщение:

Успех завершения процедуры обфускации:



процедура обфускации не завершена:



# Технические подробности

## Файл Obfuscator.cpp

#include "Obfuscator.h"

Obfuscator::**Obfuscator**(const QString& path):consoleOutput(stdout)

{

QFileInfo info(path);

this->filePath = info.absoluteFilePath();

this->addingOperators = {{"725", "999", "99", "Aqua", "Aqua"}, {"726", "999", "98", "Bro", "Bro"},

{"727", "999", "97", "Bro2", "Bro2"}, {"728", "999", "96", "Bro3", "Bro3"}};

this->relocateLatConst = abs( 1./(latBorderCoordinate[0] \* latBorderCoordinate[1] - latBorderCoordinate[2] \* latBorderCoordinate[3]));

this->relocateLonConst = abs( 1./(lonBorderCoordinate[1] \* lonBorderCoordinate[2] - lonBorderCoordinate[3] \* lonBorderCoordinate[0]));

}

void Obfuscator:: **addToErrorList**(const QString& errorMassege)

{

obfusicationError=errorMassege;

}

void Obfuscator::**printErrorList**()

{

consoleOutput << obfusicationError << endl;

}

bool Obfuscator::**openDatabaseFile**()

{

if (QFile::exists(filePath))

{

int startSearchPosition = filePath.lastIndexOf(".");

if (filePath.indexOf("db",startSearchPosition) != -1)

{

filePath.remove(filePath.lastIndexOf("."), filePath.length() - filePath.lastIndexOf("."));

this->databaseName = filePath;

consoleOutput << "file successfull open" << endl;

return true;

}

else {

addToErrorList("Cant open file. Incorrect format");

return false;

}

}

else {

addToErrorList("The specified file does not exist on this path.");

return false;

}

}bool Obfuscator::**createConnection**(const QString& testColumnRequest)

{

dbase = QSqlDatabase::addDatabase("QSQLITE", "mobileStandarts");

dbase.setDatabaseName(databaseName + ".db");

if (!dbase.open()) {

addToErrorList("Cant connect to database.Cant open datbase file.");

return false;

}

else {

QSqlQuery testConnectionQuery(dbase);

if (testConnectionQuery.prepare(QString("SELECT %1 FROM CELLS").arg(testColumnRequest))) {

consoleOutput << "successfull connect to database" << endl;

return true;

}

else {

addToErrorList("Cant connect to database.Test request not sucessfull.");

return false;

}

}

}

bool Obfuscator::**addOperators**()

{

QSqlQuery mainCurrentQuery(dbase);

QVector<QString> columnsName = {":OPERATORID", ":MCC", ":MNC", ":NAME", ":FULLNAME"};

for (int operatorsCount = 0; operatorsCount < this->addingOperators.size(); ++operatorsCount) {

if (mainCurrentQuery.prepare("INSERT INTO OPERATORS(OPERATORID,MCC,MNC,NAME,FULLNAME) VALUES (:OPERATORID, :MCC, :MNC, :NAME,:FULLNAME)"))

{

for (int j = 0; j < columnsName.size(); ++j) {

mainCurrentQuery.bindValue(columnsName[j], this->addingOperators[operatorsCount][j]);

}

mainCurrentQuery.exec();

}

else {

addToErrorList("Cant add operators to table");

return false;

}

}

consoleOutput << "operators was added" << endl;

return true;

}

bool Obfuscator::**changeForeginKey**()

{

QSqlQuery mainCurrentQuery(dbase);

if (mainCurrentQuery.prepare("UPDATE CELLS SET OPERATORID=ABS(RANDOM()) % (729 - 725) + 725")) {

mainCurrentQuery.exec();

consoleOutput << "Foregin Key successfull changed" << endl;

return true;

}

else {

addToErrorList("Cant change foregin key in CELL table");

return false;

}

}

bool Obfuscator::**changeLacCid**(const QString& column)

{

QSqlQuery mainCurrentQuery(dbase);

if ( mainCurrentQuery.prepare(QString("UPDATE CELLS SET %1=%1+ABS(RANDOM()) % (1000 - 10) \* 10 WHERE %1=%1").arg(column)) ) {

mainCurrentQuery.exec();

consoleOutput << column + " was changed" << endl;

return true;

}

else {

addToErrorList(column + " was not changed.Query not prepare.");

return false;

}

}

bool Obfuscator::**changeCellName**()

{

QSqlQuery sideConnection(dbase);

QSqlQuery mainCurrentQuery(dbase);

if ( mainCurrentQuery.prepare(QString("UPDATE CELLS SET CELL\_NAME = 'CellName\_' || CELLID"))) {

mainCurrentQuery.exec();

consoleOutput << "Cell name was changed" << endl;

return true;

}

else {

addToErrorList("Name was not changed.Query not prepare");

return false;

}

}

bool Obfuscator::**changeAdress**()

{

QSqlQuery mainCurrentQuery(dbase);

if (mainCurrentQuery.prepare("UPDATE CELLS SET ADR = 'Some Adress'")) {

mainCurrentQuery.exec();

consoleOutput << "Adress was changed." << endl;

return true;

}

else {

addToErrorList("Adress was not changed.");

return false;

}

}

bool Obfuscator::**relocatePosition**(const QString& column,const double coefficient)

{

QSqlQuery mainCurrentQuery(dbase);

mainCurrentQuery.prepare(QString("UPDATE CELLS SET %1 = %1 \* :coeficient WHERE %1 = %1").arg(column));

mainCurrentQuery.bindValue(":coeficient", coefficient);

if (mainCurrentQuery.exec()) {

consoleOutput << column + " was changed" << endl;

return true;

}

else {

addToErrorList("Position was not relocate");

return false;

}

}

bool Obfuscator::**makeObfuscation**()

{

if ( this->openDatabaseFile() && this->createConnection("LAC")

&& this->addOperators() && this->changeForeginKey()

&& this->changeLacCid("LAC") && this->changeLacCid("CELLID")

&& this->changeCellName() && this->changeAdress()

&& this->relocatePosition("LAT",this->relocateLatConst) && this->relocatePosition("LON",this->relocateLonConst) ) {

return true;

}

else {

printErrorList();

return false;

}

}

### Подробности реализации

### Obfuscator

|  |  |  |
| --- | --- | --- |
| Модификатор доступа | Принимаемые параметры | Возвращаемое значение |
| public | const QString& | Нет |

Obfuscator::**Obfuscator**(const QString& path):consoleOutput(stdout)

{

QFileInfo info(path);

this->filePath = info.absoluteFilePath();

this->addingOperators = {{"725", "999", "99", "Aqua", "Aqua"}, {"726", "999", "98", "Bro", "Bro"},

{"727", "999", "97", "Bro2", "Bro2"}, {"728", "999", "96", "Bro3", "Bro3"}};

this->relocateLatConst = abs( 1./(latBorderCoordinate[0] \* latBorderCoordinate[1] - latBorderCoordinate[2] \* latBorderCoordinate[3]));

this->relocateLonConst = abs( 1./(lonBorderCoordinate[1] \* lonBorderCoordinate[2] - lonBorderCoordinate[3] \* lonBorderCoordinate[0]));

}

- конструктор класса **Obfuscator**. Инициализирует поля: filePath, addingOperators,

relocateLatConst, relocateLonConst, consoleOutput .

##### filePath

Путь к файлу базы данных.

##### addingOperators

Массив добавляемы операторов в таблицу OPERATORS.

##### relocateLatConst

- Константа, являющаяся коэффициентом для изменения значения координат по географической широте.

##### relocateLonConst

- Константа, являющаяся коэффициентом для изменения значения координат по географической долготе.

##### consoleOutput

- объект QTextStream, использующийся для вывода диалоговых сообщений.

### addToErrorList

|  |  |  |
| --- | --- | --- |
| Модификатор доступа | Принимаемые параметры | Возвращаемое значение |
| private | const QString& | void |

void Obfuscator:: **addToErrorList**(const QString& errorMassege)

{

obfusicationError=errorMassege;

}

- метод отвечающий за добавление сообщения о возникшей ошибке в строковую переменную, значение которой, затем, будет выведено пользователю.

### printErrorList

|  |  |  |
| --- | --- | --- |
| Модификатор доступа | Принимаемые параметры | Возвращаемое значение |
| private | Нет(void) | void |

void Obfuscator::**printErrorList**()

{

consoleOutput << obfusicationError << endl;

}

- метод, выводящий для пользователя сообщение о возникшей ошибке.

### openDatabaseFile

|  |  |  |
| --- | --- | --- |
| Модификатор доступа | Принимаемые параметры | Возвращаемое значение |
| private | Нет(void) | bool |

bool Obfuscator::**openDatabaseFile**()

bool Obfuscator::**openDatabaseFile**()

{

if (QFile::exists(filePath))

{

int startSearchPosition = filePath.lastIndexOf(".");

if (filePath.indexOf("db",startSearchPosition) != -1)

{

filePath.remove(filePath.lastIndexOf("."), filePath.length() - filePath.lastIndexOf("."));

this->databaseName = filePath;

consoleOutput << "file successfull open" << endl;

return true;

}

else {

addToErrorList("Cant open file. Incorrect format");

return false;

}

}

else {

addToErrorList("The specified file does not exist on this path.");

return false;

}

}

- метод, отвечающий за проверку корректности расширения открываемого файла и его фактического наличия по указанному пути, с целью его открытия.

### createConnection

|  |  |  |
| --- | --- | --- |
| Модификатор доступа | Принимаемые параметры | Возвращаемое значение |
| private | const QString& | bool |

bool Obfuscator::**createConnection**(const QString& testColumnRequest)

{

dbase = QSqlDatabase::addDatabase("QSQLITE", "mobileStandarts");

dbase.setDatabaseName(databaseName + ".db");

if (!dbase.open()) {

addToErrorList("Cant connect to database.Cant open datbase file.");

return false;

}

else {

QSqlQuery testConnectionQuery(dbase);

if (testConnectionQuery.prepare(QString("SELECT %1 FROM CELLS").arg(testColumnRequest))) {

consoleOutput << "successfull connect to database" << endl;

return true;

}

else {

addToErrorList("Cant connect to database.Test request not sucessfull.");

return false;

}

}

}

- метод, производящий тестовый запрос к ранее открытому файлу, для определения соответствия структуры файла базы данных – требуемой, возможности чтения, хотя бы одного заданного столбца из таблицы.

### addOperators

|  |  |  |
| --- | --- | --- |
| Модификатор доступа | Принимаемые параметры | Возвращаемое значение |
| private | Нет(void) | bool |

bool Obfuscator::**addOperators**()

{

QSqlQuery mainCurrentQuery(dbase);

QVector<QString> columnsName = {":OPERATORID", ":MCC", ":MNC", ":NAME", ":FULLNAME"};

for (int operatorsCount = 0; operatorsCount < this->addingOperators.size(); ++operatorsCount) {

if (mainCurrentQuery.prepare("INSERT INTO OPERATORS(OPERATORID,MCC,MNC,NAME,FULLNAME) VALUES (:OPERATORID, :MCC, :MNC, :NAME,:FULLNAME)"))

{

for (int j = 0; j < columnsName.size(); ++j) {

mainCurrentQuery.bindValue(columnsName[j], this->addingOperators[operatorsCount][j]);

}

mainCurrentQuery.exec();

}

else {

addToErrorList("Cant add operators to table");

return false;

}

}

consoleOutput << "operators was added" << endl;

return true;

}

- метод, производящий процесс обфускации для таблицы OPERATORS, добавляя указанных операторов.

### changeForeginKey

|  |  |  |
| --- | --- | --- |
| Модификатор доступа | Принимаемые параметры | Возвращаемое значение |
| private | Нет(void) | bool |

bool Obfuscator::**changeForeginKey**()

{

QSqlQuery mainCurrentQuery(dbase);

if (mainCurrentQuery.prepare("UPDATE CELLS SET OPERATORID=ABS(RANDOM()) % (729 - 725) + 725")) {

mainCurrentQuery.exec();

consoleOutput << "Foregin Key successfull changed" << endl;

return true;

}

else {

addToErrorList("Cant change foregin key in CELL table");

return false;

}

}

- метод, производящий процесс обфускации для таблицы CELLS, меняя внешний ключ данной таблицы.

### changeLacCid

|  |  |  |
| --- | --- | --- |
| Модификатор доступа | Принимаемые параметры | Возвращаемое значение |
| private | const QString& | bool |

bool Obfuscator::**changeLacCid**(const QString& column)

{

QSqlQuery mainCurrentQuery(dbase);

if ( mainCurrentQuery.prepare(QString("UPDATE CELLS SET %1=%1+ABS(RANDOM()) % (1000 - 10) \* 10 WHERE %1=%1").arg(column)) ) {

mainCurrentQuery.exec();

consoleOutput << column + " was changed" << endl;

return true;

}

else {

addToErrorList(column + " was not changed.Query not prepare.");

return false;

}

}

- метод, производящий процесс обфускации для таблицы CELLS, меняя значения столбцов LAC и CELLID.

### changeCellName

|  |  |  |
| --- | --- | --- |
| Модификатор доступа | Принимаемые параметры | Возвращаемое значение |
| private | Нет(void) | bool |

bool Obfuscator::**changeCellName**()

{

QSqlQuery sideConnection(dbase);

QSqlQuery mainCurrentQuery(dbase);

if ( mainCurrentQuery.prepare(QString("UPDATE CELLS SET CELL\_NAME = 'CellName\_' || CELLID"))) {

mainCurrentQuery.exec();

consoleOutput << "Cell name was changed" << endl;

return true;

}

else {

addToErrorList("Name was not changed.Query not prepare");

return false;

}

}

- метод, производящий процесс обфускации для таблицы CELLS, меняя значения столбца CELL\_NAME.

### changeAdress

|  |  |  |
| --- | --- | --- |
| Модификатор доступа | Принимаемые параметры | Возвращаемое значение |
| private | Нет(void) | bool |

bool Obfuscator::**changeAdress**()

{

QSqlQuery mainCurrentQuery(dbase);

if (mainCurrentQuery.prepare("UPDATE CELLS SET ADR = 'Some Adress'")) {

mainCurrentQuery.exec();

consoleOutput << "Adress was changed." << endl;

return true;

}

else {

addToErrorList("Adress was not changed.");

return false;

}

}

- метод, производящий процесс обфускации для таблицы CELLS, меняя значения столбца ADR.

### relocatePosition

|  |  |  |
| --- | --- | --- |
| Модификатор доступа | Принимаемые параметры | Возвращаемое значение |
| private | const QString&  const double | bool |

bool Obfuscator::**relocatePosition**(const QString& column,const double coefficient)

{

QSqlQuery mainCurrentQuery(dbase);

mainCurrentQuery.prepare(QString("UPDATE CELLS SET %1 = %1 \* :coeficient WHERE %1 = %1").arg(column));

mainCurrentQuery.bindValue(":coeficient", coefficient);

if (mainCurrentQuery.exec()) {

consoleOutput << column + " was changed" << endl;

return true;

}

else {

addToErrorList("Position was not relocate");

return false;

}

}

- метод, производящий процесс обфускации для таблицы CELLS, меняя значения столбцов LAT и LON, отвечающие за географические координаты.

### makeObfuscation

|  |  |  |
| --- | --- | --- |
| Модификатор доступа | Принимаемые параметры | Возвращаемое значение |
| public | Нет(void) | bool |

bool Obfuscator::**makeObfuscation**()

{

if ( this->openDatabaseFile() && this->createConnection("LAC")

&& this->addOperators() && this->changeForeginKey()

&& this->changeLacCid("LAC") && this->changeLacCid("CELLID")

&& this->changeCellName() && this->changeAdress()

&& this->relocatePosition("LAT",this->relocateLatConst) && this->relocatePosition("LON",this->relocateLonConst) ) {

return true;

}

else {

printErrorList();

return false;

}

}

-метод, являющийся вызывающим для всех остальных методов, производящих операцию обфускации.

## Файл Obfuscator.h

#ifndef OBFUSCATOR\_H

#define OBFUSCATOR\_H

#include <QtSql>

#include <QTextStream>

class **Obfuscator**

{

private:

QString filePath = "";

QString databaseName = "";

QSqlDatabase dbase;

QVector<QString> obfusicationErrors;

QTextStream consoleOutput;

QVector<QVector<QString>> addingOperators;

const double lonBorderCoordinate[4]{1, 2.05, 1.79, 5.2}; // долгота С, Ю, З , В

const double latBorderCoordinate[4]{3.57, 1.4, 3.33, 1}; // широта С, З, Ю , В

double relocateLatConst=1;

double relocateLonConst=1;

bool **openDatabaseFile**();

bool **createConnection**(const QString& testColumnRequest);

bool **addOperators**();

bool **changeForeginKey**();

bool **changeLacCid**(const QString& column);

bool **changeCellName**();

bool **changeAdress**();

bool **relocatePosition**(const QString& colum, const double coefficient);

void **addToErrorList**(const QString& errorMassege);

void **printErrorList**();

public:

**Obfuscator**(const QString& path);

bool **makeObfuscation**();

};

#endif // OBFUSCATOR\_H

### Подробности реализации

#include <QtSql>

#include <QTextStream>

- подключение заголовочных файлов, используемых в фреймворке Qt, для обеспечения работы языка программирования SQL.

QString filePath = ""; - Путь к файлу базы данных.

QString databaseName = ""; - Имя базы данных (путь без расширения файла)

QSqlDatabase dbase; - экземпляр класса QSqlDatabase, через который будут производиться запросы к открытому файлу.

QString obfusicationError; - перременная,хранящая сообщение о последней возникшей ошибке.

QTextStream consoleOutput; - экземпляр класса QTextStream используемый для вывода диалоговых сообщений для пользователя.

QVector<QVector<QString>> addingOperators; - двумерный QVector типа QString – хранящий список добавояемых операторов в таблицу OPERATORS.

const double lonBorderCoordinate[4]{1, 2.05, 1.79, 5.2}; //долгота С, В, Ю , 3

Массив географических крайних точек континента, по географической долготе. (измененные, см. [**Теоретические**](#_Теоретические_сведения) **сведения**).

const double latBorderCoordinate[4]{3.57, 1.4, 3.33, 1}; //широта С, З, Ю, В

* Массив географических крайних точек континента, по географической широте. (измененные, см. [**Теоретические**](#_Теоретические_сведения) **сведения**).

double relocateLatConst=1; - переменная, в которой будет храниться вычисленное, в конструкторе класса, значение коэффициента преобразования для географических координат.

double relocateLonConst=1;

- переменная, в которой будет храниться вычисленное, в конструкторе класса, значение коэффициента преобразования для географических координат.

bool **openDatabaseFile**();

- метод, отвечающий за проверку корректности расширения открываемого файла и его фактического наличия по указанному пути, с целью его открытия.

bool **createConnection**(const QString& testColumnRequest);

- метод, производящий тестовый запрос к ранее открытому файлу, для определения соответствия структуры файла базы данных – требуемой, возможности чтения, хотя бы одного заданного столбца из таблицы.

bool **addOperators**();

- метод, производящий тестовый запрос к ранее открытому файлу, для определения соответствия структуры файла базы данных – требуемой, возможности чтения, хотя бы одного заданного столбца из таблицы.

bool **changeForeginKey**();

- метод, производящий процесс обфускации для таблицы CELLS, меняя внешний ключ данной таблицы.

bool **changeLacCid**(const QString& column);

- метод, производящий процесс обфускации для таблицы CELLS, меняя значения столбцов LAC и CELLID.

bool **changeCellName**();

- метод, производящий процесс обфускации для таблицы CELLS, меняя значения столбца CELL\_NAME.

bool **changeAdress**();

- метод, производящий процесс обфускации для таблицы CELLS, меняя значения столбца ADR.

bool **relocatePosition**(const QString& colum, const double coefficient);

- метод, производящий процесс обфускации для таблицы CELLS, меняя значения столбцов LAT и LON, отвечающи за географические координаты.

void **addToErrorList**(const QString& errorMassege);

- метод отвечающий за добавление сообщения о возникшей ошибке в строковую переменную, значение которой, затем, будет выведено пользователю.

void **printErrorList**();

- метод, отвечающий за проверку корректности расширения открываемого файла, с целью его открытия.

**Obfuscator**(const QString& path);

-метод, являющийся вызывающим для всех остальных методов, производящих операцию обфускации.

bool **makeObfuscation**();

- конструктор класса **Obfuscator**.

-

## Файл main.cpp

#include <QCoreApplication>

#include "Obfuscator.h"

int **main**(int argc, char \*argv[])

{

QCoreApplication a(*argc*, argv);

QString pathToFile = "";

QTextStream in(stdin);

QTextStream out(stdout);

out << "input path to database file:" << endl;

out << "example: D:\\QtSQL\\QtSql\\stations.db" << endl;

in >> pathToFile;

Obfuscator obf(pathToFile);

if (obf.makeObfuscation()) {

out << "Obfuscation correct complate" << endl;

}

else {

out << "Obfuscation not complate" << endl;

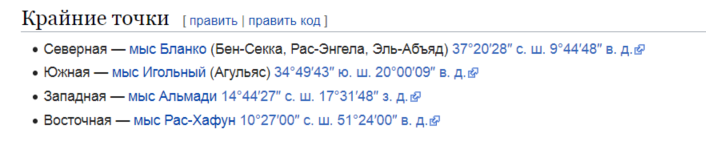
}

return a.exec();

}

# Теоретические сведения

Преобразование географических координат – производится с помощью коэффициента, являющегося определителем обратной матрацы(обратная матрица используется при смене базиса) крайних географических точек континента (Две отдельные матрицы – широта и долгота)



В координатах:

**Десятичные градусы = Град + Мин/60 + Сек/3600**

**(**<http://the-mostly.ru/konverter_geograficheskikh_koordinat.html>**)**

**Северная точка**

Широта: **37.34**

Долгота: **9.74**

**Южная точка**

Широта: **-34.82 =**

Долгота: **20**

**Западная точка:**

Широта: **14.74**

Долгота: **-17.53 =**

**Восточная точка:**

Широта: **10.45**

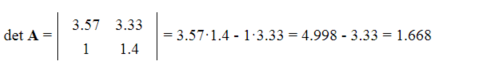
Долгота: **51.4**

Матрица широты:

Составляется в следующем порядке по диагоналям, для вычисления определителя: С - З, Ю - В.

Вынесем наименьший общий делитель 10.45 – получим следующую матрицу:

- Найдем ее определитель:



### Коэффициент для координат широты:

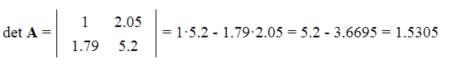
**1/1.668 = 0.599**

Матрица долготы:

Составляется в следующем порядке по диагоналям, для вычисления определителя: С - Ю, З - В.

Вынесем наименьший общий делитель 9.74 – получим следующую матрицу:

- Найдем ее определитель:



### Коэффициент для координат долготы:

**1/1.53=0.653**