МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Санкт-Петербургский государственный университет телекоммуникаций им. проф. М. А. Бонч-Бруевича

Факультет Инфокоммуникационных сетей и систем

**Курсовая работа**

по дисциплине

«Объектно-ориентированное программирование»

по теме

**«Разработка приложения для управления базой данных»**

Вариант 3

Выполнил: Чугуненко А.С.

Курс: 2

Группа: ИКПИ-74

Принимает: Петрова О. Б.

Санкт-Петербург

2018

Содержание

Постановка задачи………………………………………...………….….………3

Разработка структуры программы………………...……………………..……..4

Выбор и обоснование используемых классов………………..……….……….4

Разработка отдельных модулей…..………….…………………….…..….……4

Поля и методы класса…………..………………………………………….……4

Файл Database.h………………………..…………………………………….…..4

Файл Database.cpp…………………………………………………………….…5

Файл AddDialogWindow.h………………….…….……………………………..6

Файл AddDialogWindow.cpp……………………………………………………7

Файл ChangeDialogWindow.h…………………….……………………………..8

Файл ChangeDialogWindow.cpp………………...………………………………8

Файл MainWindow.h……………………………………….……………………9

Файл MainWindow.cpp………………..………………………………………...10

Файл main.cpp…………………………………………………………………...16

Разработка интерфейса пользователя…………………………………….……17

Общий вид программы………………………………………………………….17

Импорт БД из файла……………………………………………………..……...17

Экспорт БД в файл……………………………………………………..………..18

Окно добавления записи………………………………………………...………18

Окно поиска записей……………………………………………………...……..19

Заключение……………………………………………………...………………..19

Список литературы………………………………………………………..……..19

**Постановка задачи:**

Создать базу данных, работающую на основе контейнера, который следует разработать в процессе выполнения курсовой работы. В качестве прототипа контейнера можно использовать контейнеры vector или list из библиотеки STL. Разрабатываемая база данных должна быть создана использованием C++ и библиотеки Qt4. Среда программировании Qt Creator.

В качестве элементов контейнеров должны использоваться объекты класса, характеризующего файл.

Необходимо обеспечивать выполнение операций, таких как:

* + создание базы данных,
  + объединение баз данных,
  + добавления и удаления записей,
  + редактирования записей,
  + просмотра базы данных,
  + поиск данных (найти данные, удовлетворяющие заданному критерию),
  + запись на диск,
  + чтения с диска.

Интерфейсная часть программы должна содержать следующие компоненты:

* + главное меню,
  + всплывающее меню,
  + строку состояния,
  + панель инструментов,
  + подсказки по командам меню и панели инструментов.

Пояснительная записка должна содержать:

* + постановку задачи,
  + выбор и обоснование типа разрабатываемого контейнера,
  + разработку структуры программы.
  + разработку отдельных модулей.
  + выбор и обоснование используемых классов,
  + разработку собственных классов,
  + разработку интерфейса пользователя,
  + инструкцию пользователя,
  + заключение,

**Разработка структуры программы:**

Программа содержит в себе следующие модули:

* Заголовочные файлы
  + AddDialogWindow.h
  + ChangeDialogWindow.h
  + Database.h
  + MainWindow.h
* Исходные файл
  + AddDialogWindow.cpp
  + ChangeDialogWindow.cpp
  + Database.cpp
  + main.cpp
  + MainWindow.cpp

**Выбор и обоснование используемых классов:**

MainWindow — класс для работы с пользовательским интерфейсом.

QDialog — класс для добавления и редактирования записей в базе.

QLineEdit— представляет собой редактор однострочного текста.

## QValidator—проверка корректности ввода чисел.

## QList — это структура данных, представляющая собой упорядоченный набор связанных друг с другом элементов.

Database — класс-контейнер для хранения данных о файлах.

**Разработка отдельных модулей:**

**Поля и методы класса**

QString name- поле содержащее название файла.

int size-поле, содержащие размер файла.

QString type-поле, содержащие тип файла.

QString date-поле, содержащие дату создания файла.

**Файл Database.h:**

#ifndef DATABASE\_H

#define DATABASE\_H

#include <QString>

#include <QList>

class **database** {

public:

int count;

bool status;

struct **file**

{

QString name; //Название файла

int size; //Размер файла

QString type; //Тип файла

QString date; //Дата создания

};

**database**();

void **addNew**(file file2);

void **change**(int number, QString name, int size, QString type, QString date);

void **remove**(int i);

void **clear**();

QString **print**(int i);

QString **get\_name**(int i);

int **get\_size**(int i);

QString **get\_type**(int i);

QString **get\_date**(int i);

~**database**();

QList<file> data;

private:

int a;

};

#endif // DATABASE\_H

**Файл Database.cpp:**

#include "Database.h"

database::**database**()

{

count=0;

status=false;

a=0;

}

void database::**addNew**(file file2)

{

data<<file2;

count++;

}

void database::**change**(int number, QString name, int size, QString type, QString date)

{

file new\_file;

new\_file.name = name;

new\_file.size = size;

new\_file.type = type;

new\_file.date = date;

data[number]=new\_file;

}

QString database::**print**(int i)

{

if(a==4)

a=0;

file new\_file;

new\_file = data[i];

QString text;

if(a==0)text=new\_file.name;

if(a==1)text.setNum(new\_file.size);

if(a==2)text=new\_file.type;

if(a==3)text=new\_file.date;

a++;

return text;

}

void database::**remove**(int i)

{

data.removeAt(i);

count--;

}

void database::**clear**()

{

data.clear();

count=0;

}

QString database::**get\_name**(int i)

{

return data[i].name;

}

int database::**get\_size**(int i)

{

return data[i].size;

}

QString database::**get\_type**(int i)

{

return data[i].type;

}

QString database::**get\_date**(int i)

{

return data[i].date;

}

database::~**database**()

{

}

**Файл AddDialogWindow.h:**

#ifndef DIALOG\_H

#define DIALOG\_H

#include <QDialog>

#include <QLineEdit>

#include <QBoxLayout>

#include <QLabel>

#include <QPushButton>

#include <QValidator>

class **AddDialogWindow** : public QDialog

{

Q\_OBJECT

protected:

QLineEdit \*name\_line;

QLineEdit \*size\_line;

QLineEdit \*type\_line;

QLineEdit \*date\_line;

QLabel \*name\_label;

QLabel \*size\_label;

QLabel \*type\_label;

QLabel \*date\_label;

QLabel \*size\_label\_back;

QPushButton \*add\_button;

QHBoxLayout \*name\_hbox;

QHBoxLayout \*size\_hbox;

QHBoxLayout \*type\_hbox;

QHBoxLayout \*date\_hbox;

QVBoxLayout \*vbox;

public:

**AddDialogWindow**(QWidget \*parent);

QString **get\_name**();

int **get\_size**();

QString **get\_type**();

QString **get\_date**();

void **set\_name**(QString name);

void **set\_size**(int size);

void **set\_type**(QString type);

void **set\_date**(QString date);

~***AddDialogWindow***();

};

#endif // DIALOG\_H

**Файл AddDialogWindow.cpp:**

#ifndef DIALOG\_H

#define DIALOG\_H

#include <QDialog>

#include <QLineEdit>

#include <QBoxLayout>

#include <QLabel>

#include <QPushButton>

#include <QValidator>

class **AddDialogWindow** : public QDialog

{

Q\_OBJECT

protected:

QLineEdit \*name\_line;

QLineEdit \*size\_line;

QLineEdit \*type\_line;

QLineEdit \*date\_line;

QLabel \*name\_label;

QLabel \*size\_label;

QLabel \*type\_label;

QLabel \*date\_label;

QLabel \*size\_label\_back;

QPushButton \*add\_button;

QHBoxLayout \*name\_hbox;

QHBoxLayout \*size\_hbox;

QHBoxLayout \*type\_hbox;

QHBoxLayout \*date\_hbox;

QVBoxLayout \*vbox;

public:

**AddDialogWindow**(QWidget \*parent);

QString **get\_name**();

int **get\_size**();

QString **get\_type**();

QString **get\_date**();

void **set\_name**(QString name);

void **set\_size**(int size);

void **set\_type**(QString type);

void **set\_date**(QString date);

~***AddDialogWindow***();

};

#endif // DIALOG\_H

**Файл ChangeDialogWindow.h:**

#ifndef CHANGEDIALOGWINDOW\_H

#define CHANGEDIALOGWINDOW\_H

#include "AddDialogWindow.h"

class **ChangeDialogWindow** : public AddDialogWindow

{

Q\_OBJECT

public:

**ChangeDialogWindow**();

~***ChangeDialogWindow***();

};

#endif // CHANGEDIALOGWINDOW\_H

**Файл ChangeDialogWindow.cpp:**

#include "ChangeDialogWindow.h"

ChangeDialogWindow::**ChangeDialogWindow**() : AddDialogWindow(nullptr)

{

this->setWindowTitle("Изменение");

add\_button->setText("Изменить");

connect(add\_button,SIGNAL(clicked(bool)),this,SLOT(accept()));

}

ChangeDialogWindow::~***ChangeDialogWindow***()

{

**Файл MainWindow.h:**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QLineEdit>

#include <QBoxLayout>

#include <QLabel>

#include <QPushButton>

#include <QTableWidget>

#include <QMenuBar>

#include <QDebug>

#include <QMenu>

#include <QToolBar>

#include <QStatusBar>

#include <QFile>

#include <QFileDialog>

#include <QTextStream>

#include <QTextCodec>

#include <QHeaderView>

#include <QMessageBox>

#include "Database.h"

#include "AddDialogWindow.h"

#include "ChangeDialogWindow.h"

class **MainWindow** : public QMainWindow

{

Q\_OBJECT

private:

QLabel \*label\_search;

QLabel \*label\_count;

QPushButton \*add;

QPushButton \*change;

QPushButton \*remove;

QLineEdit \*search\_line;

QLabel \*search\_label;

QLabel \*status\_label;

QMenuBar \*menuBar;

QMenu \*menu;

QMenu \*menu\_2;

QToolBar \*tool\_bar;

QStatusBar \*status\_bar;

QTableWidget \*table;

database \*data;

public:

**MainWindow**(QWidget \*parent);

~***MainWindow***();

void **clear\_database**();

void **open\_file**();

public slots:

void **add\_click**();

void **show\_database**();

void **remove\_click**();

void **change\_click**();

void **search**(QString search);

void **save\_to\_file**();

void **open\_from\_file**();

void **new\_database**();

void **merger**();

void **info\_show**();

void **instructions\_show**();

};

#endif // MAINWINDOW\_H

**Файл MainWindow.cpp:**

#include "MainWindow.h"

MainWindow::**MainWindow**(QWidget \*parent) : QMainWindow(parent)

{

this->setMinimumSize(700,500);

this->setWindowTitle("Файлы");

data = new database();

menuBar = new QMenuBar;

menu = new QMenu("Файл");

menu->addAction("Новый",this,SLOT(new\_database()));

menu->addAction("Открыть",this,SLOT(open\_from\_file()));

menu->addAction("Сохранить",this,SLOT(save\_to\_file()));

menu->addSeparator();

menu->addAction("Слияние",this,SLOT(merger()));

menu\_2 = new QMenu("Справка");

menu\_2->addAction("Инструкция",this,SLOT(instructions\_show()));

menu\_2->addSeparator();

menu\_2->addAction("О программе",this,SLOT(info\_show()));

menuBar->addMenu(menu);

menuBar->addSeparator();

menuBar->addMenu(menu\_2);

tool\_bar = new QToolBar;

tool\_bar->setMovable(false);

status\_bar = new QStatusBar;

add = new QPushButton("Добавить");

change = new QPushButton("Изменить");

remove = new QPushButton("Удалить");

search\_line = new QLineEdit;

label\_search = new QLabel("Поиск:");

label\_count = new QLabel;

search\_label = new QLabel;

status\_label = new QLabel;

table = new QTableWidget(0,4);

table->setHorizontalHeaderLabels(QStringList()<<"Название"<<"Размер"<<"Тип"<<"Дата создания");

table->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch);

tool\_bar->addWidget(add);

tool\_bar->addWidget(change);

tool\_bar->addWidget(remove);

tool\_bar->addSeparator();

tool\_bar->addWidget(label\_search);

tool\_bar->addWidget(search\_line);

status\_bar->addWidget(status\_label);

status\_bar->addWidget(label\_count);

status\_bar->addWidget(search\_label);

this->setCentralWidget(table);

this->setMenuBar(menuBar);

this->addToolBar(tool\_bar);

this->setStatusBar(status\_bar);

show\_database();

connect(add,SIGNAL(clicked(bool)),this,SLOT(add\_click()));

connect(add,SIGNAL(clicked(bool)),this,SLOT(show\_database()));

connect(remove,SIGNAL(clicked(bool)),this,SLOT(remove\_click()));

connect(remove,SIGNAL(clicked(bool)),this,SLOT(show\_database()));

connect(change,SIGNAL(clicked(bool)),this,SLOT(change\_click()));

connect(change,SIGNAL(clicked(bool)),this,SLOT(show\_database()));

connect(search\_line,SIGNAL(textChanged(QString)),this,SLOT(search(QString)));

}

void MainWindow::**add\_click**()

{

AddDialogWindow log(nullptr);

log.setModal(true);

log.show();

if(log.*exec*()==QDialog::Accepted)

{

database::file new\_file;

new\_file.name = log.get\_name();

new\_file.size = log.get\_size();

new\_file.type = log.get\_type();

new\_file.date = log.get\_date();

data->addNew(new\_file);

data->status=false;

}

}

void MainWindow::**show\_database**()

{

table->setRowCount(data->count);

for(int row=0; row!=data->count; ++row){

for(int column=0; column!=4; ++column) {

QTableWidgetItem \*newItem = new QTableWidgetItem(tr("%1").arg(data->print(row)));

table->setItem(row, column, newItem);

}

}

label\_count->setText(QString("Записей: %0").arg(data->count));

if(data->status == true)

status\_label->setText("БД сохранена.");

else

status\_label->setText("БД не сохранена!");

}

void MainWindow::**remove\_click**()

{

int count = table->currentIndex().row();

if(count>=0)

{

data->remove(count);

search\_label->setText("");

data->status=false;

}

else

QMessageBox::warning(nullptr, "warning", "Выберите строку для удаления!");

}

void MainWindow::**change\_click**()

{

int count = table->currentIndex().row();

if(count>=0)

{

ChangeDialogWindow log;

log.setModal(true);

log.set\_name(data->get\_name(count));

log.set\_size(data->get\_size(count));

log.set\_type(data->get\_type(count));

log.set\_date(data->get\_date(count));

if(log.*exec*()==QDialog::Accepted)

{

data->change(count,log.get\_name(),log.get\_size(),log.get\_type(),log.get\_date());

search\_label->setText("");

data->status=false;

}

}

else

QMessageBox::warning(nullptr, "warning", "Выберите строку для изменения!");

}

void MainWindow::**search**(QString search)

{

if(search=="")

{

for(int i = 0; i<data->count;i++)

{

table->showRow(i);

}

search\_label->setText("");

}

else

{

int bbb =0;

for(int i = 0; i<data->count; i++)

{

if(search == data->get\_name(i))

{

table->showRow(i);

bbb++;

}

else table->hideRow(i);

}

search\_label->setText(QString("Совпадений: %0").arg(bbb));

}

}

void MainWindow::**info\_show**()

{

QDialog info;

info.setModal(true);

info.setWindowTitle("О программе");

QPushButton OK("OK");

QLabel about("Курсовая работа по дисциплине ООП на тему:\n"

"Базы данных 'Файл'\n\n"

"Выполнил:\n"

"Александр Чугуненко - 2018 год\n");

QVBoxLayout box;

box.addWidget(&about);

box.addWidget(&OK);

info.setLayout(&box);

info.show();

connect(&OK,SIGNAL(clicked(bool)),&info,SLOT(accept()));

if(info.*exec*()==QDialog::Accepted)

{

info.close();

}

}

void MainWindow::**instructions\_show**()

{

QDialog instruction;

instruction.setWindowTitle("Инструкция");

instruction.setModal(true);

QPushButton OK("OK");

QLabel about("\nСмотрите техническую документацию\n");

QVBoxLayout box;

box.addWidget(&about);

box.addWidget(&OK);

instruction.setLayout(&box);

instruction.show();

connect(&OK,SIGNAL(clicked(bool)),&instruction,SLOT(accept()));

if(instruction.*exec*()==QDialog::Accepted)

{

instruction.close();

}

}

void MainWindow::**save\_to\_file**()

{

if(data->count!=0&&data->status==false)

{

QFile outFile;

QString filename = QFileDialog::getSaveFileName(nullptr, "Сохранить", "", "\*.txt");

if(filename.isEmpty())

return;

if(!filename.endsWith(".txt"))

filename+=".txt";

outFile.setFileName(filename);

if(!outFile.*open*(QIODevice::WriteOnly | QIODevice::Text))

return;

QTextStream streamOut(&outFile);

streamOut.setCodec(QTextCodec::codecForName("UTF-8"));

for (int i=0;i<data->count;i++)

{

for (int j=0;j<4;j++)streamOut<<data->print(i)<<"|";

streamOut<<"\r\n";

}

outFile.*close*();

data->status=true;

show\_database();

}

else QMessageBox::warning(nullptr,"Warning","База данных пуста или уже сохранена!",QMessageBox::Yes);

}

void MainWindow::**open\_from\_file**()

{

if(data->status==true||data->count==0)

{

clear\_database();

data->status=true;

open\_file();

}

else

{

int answer=QMessageBox::warning(nullptr,"Warning","База данных не сохранена!"

"\nYes для сохранения текущей и открытия новой"

"\nNo для открытия новой без сохранения текущей"

"\nCancel для возврата"

,QMessageBox::Yes|QMessageBox::No|QMessageBox::Cancel);

if(answer==QMessageBox::Yes)

{

save\_to\_file();

clear\_database();

data->status=true;

open\_file();

}

else if(answer==QMessageBox::No)

{

clear\_database();

data->status=true;

open\_file();

}

}

}

void MainWindow::**open\_file**()

{

QFile inFile;

QString filename = QFileDialog::getOpenFileName(this, "Открыть", "", "\*.txt");

if(filename.isEmpty())

return;

inFile.setFileName(filename);

if(!inFile.*open*(QIODevice::ReadOnly | QIODevice::Text))

return;

QStringList dataList;

QString fileData;

while(!inFile.*atEnd*())

{

fileData = QString(inFile.readLine());

dataList=fileData.split("|");

database::file new\_file;

new\_file.name=dataList[0];

new\_file.size=dataList[1].toInt();

new\_file.type=dataList[2];

new\_file.date=dataList[3];

data->addNew(new\_file);

}

show\_database();

inFile.*close*();

}

void MainWindow::**merger**()

{

if(data->count!=0)

{

data->status=false;

open\_file();

}

else

QMessageBox::warning(nullptr,"Warning", "Слияние не может быть произведенно,\nтак как текущая БД пуста!",QMessageBox::Yes);

}

void MainWindow::**new\_database**()

{

if(data->status==true||data->count==0)

{

data->status=false;

clear\_database();

}

else

{

int answer=QMessageBox::warning(nullptr,"Warning","Yes для сохранения текущей БД в файл "

"и создания новой.\nNo для очистки текущей БД без сохранения"

" и создания новой\nCancel для возврата",

QMessageBox::Yes|QMessageBox::No|QMessageBox::Cancel);

if(answer==QMessageBox::Yes)

{

save\_to\_file();

data->status=false;

clear\_database();

}

else

if(answer==QMessageBox::No)

clear\_database();

}

}

void MainWindow::**clear\_database**()

{

data->clear();

show\_database();

}

MainWindow::~***MainWindow***(){}

**Файл main.cpp:**

#include "MainWindow.h"

#include <QApplication>

#include <QtDebug>

#include <QDebug>

#include <QFileDialog>

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

{

QApplication a(*argc*, argv);

MainWindow \*window = new MainWindow(nullptr);

window->show();

QFile styleSheetFile;

styleSheetFile.setFileName(":/style/style.css");

styleSheetFile.*open*(QFile::ReadOnly);

QString qssString = styleSheetFile.readAll();

a.setStyleSheet(qssString);

if(styleSheetFile.exists())

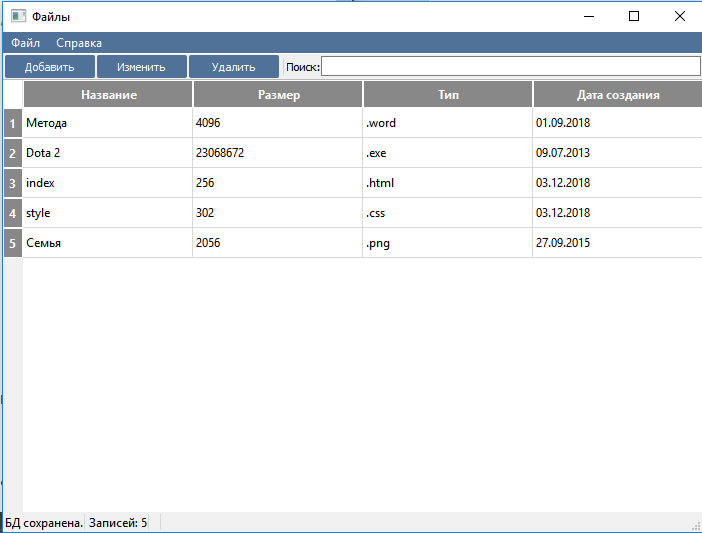
qDebug() << "[styleSheet loaded]";

return a.exec();

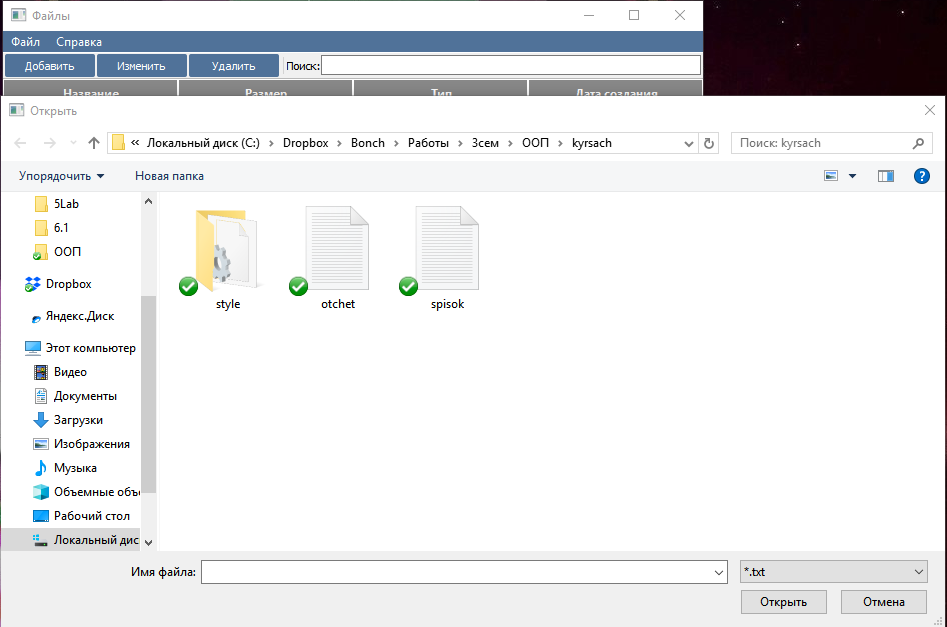
}

**Разработка интерфейса пользователя:**

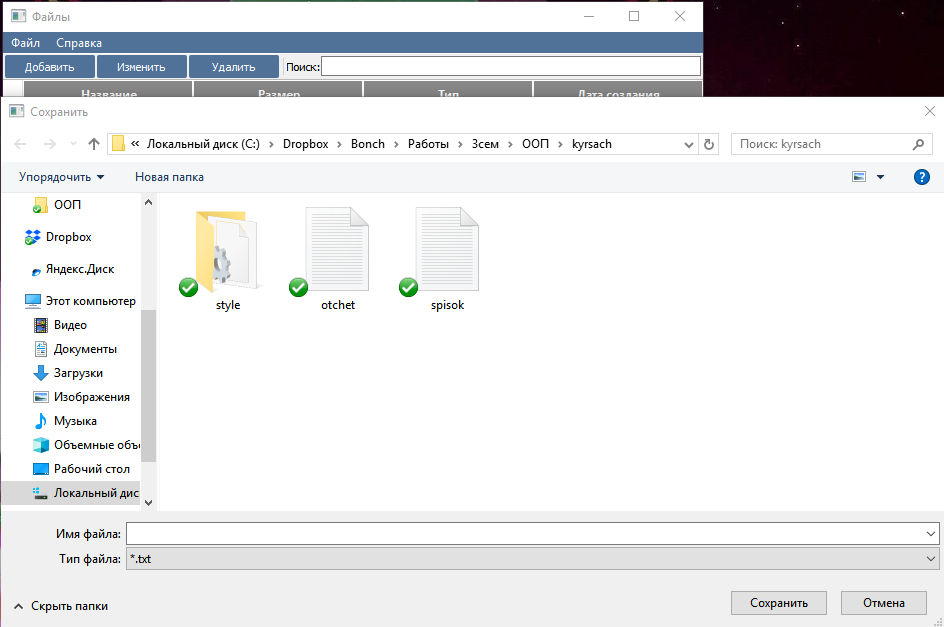
**Общий вид программы**



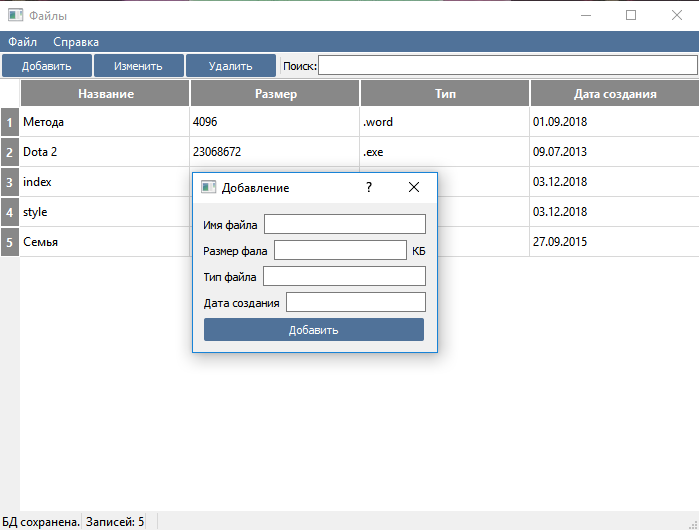
**Импорт БД из файла**



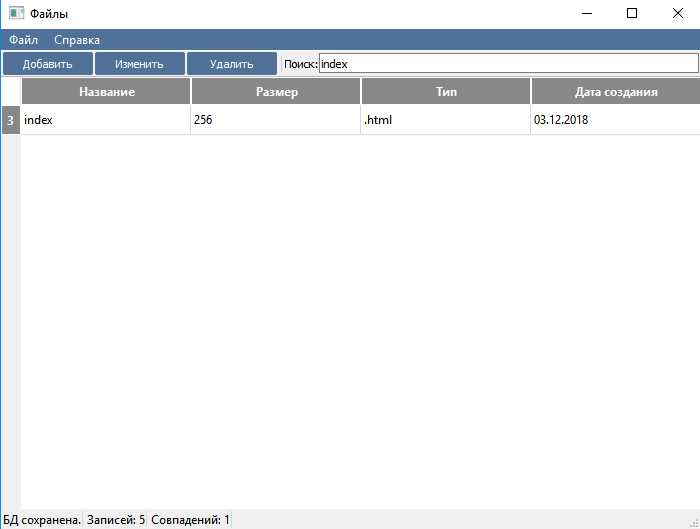
**Экспорт БД в файл**



**Окно добавления записи**



**Окно поиска записей**



**Заключение**

Поставленная задача была выполнена успешно. В ходе работы над курсовой работой были освоены основные возможности языка c++ и библиотеки Qt 4.

Написанная программа выполняет все поставленные задачи:

Главное окно содержит панель инструментов, главное меню, таблицу для отображения базы. В пункте файл содержатся подпункты новый, открыть, сохранить и слияние. В пункте таблица – добавить, изменить, удалить и поиск.

**Список литературы**

С. В. Козин. Лекции по дисциплине Объектно-ориентированное программирование.