Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики» (СибГУТИ)

Кафедра ПМиК

**Лабораторная работа № 15**

**по дисциплине «Современные технологии программирования»**

**«Телефонная книга»**

Выполнил: студент 4 курса ф. ИВТ, гр. ИП-813

Пещеров Вячеслав Александрович

Проверил: ассистент

к. ПМиК Агалаков А.А.

Новосибирск, 2021

Оглавление

[**Задание** 3](#_Toc90056353)

[**Реализация** 4](#_Toc90056354)

[**Демонстрация работы** 5](#_Toc90056355)

[**Вывод** 15](#_Toc90056356)

[**Список литературы** 16](#_Toc90056357)

[**Приложение** 17](#_Toc90056358)

[Листинг 1. test.cpp 17](#_Toc90056359)

[Листинг 2. phonebook.cpp 19](#_Toc90056360)

[Листинг 3.phoneb.cpp 25](#_Toc90056361)

# **Задание**

Объектно-ориентированный анализ, проектирование иреализация приложения под Windows «Телефонная книга».

В процессе выполнения работы студенты изучают:

* этапы разработки приложений в технологии ООП;
* элементы технологии визуального программирования;
* ассоциативный контейнер stl – multimap;
* потоковые классы для работы с файлами на диске;
* диаграммы языка UML для документирования разработки.

Функциональные требования к приложению.

Приложение должно обеспечивать пользователю:

* ввод, редактирование и сохранение имён абонентов городской телефонной сети и номеров их телефонов
* записи должны храниться и отображаться в отсортированном по именам порядке;
* поиск по имени;
* удаление записи;
* очистку книги.

Функциональные требования представлены диаграммой прецедентов (use- case диаграммой) расположенной ниже.

# **Реализация**

В данной работе мы создали приложение “телефонная книга” и все необходимые для работы с ней операции, такие как: ввод , редактирование, и сохранение имён абонентов и их номеров телефона, сортировка записей, поиск по имени, удаление и очистка книги.

# **Демонстрация работы**

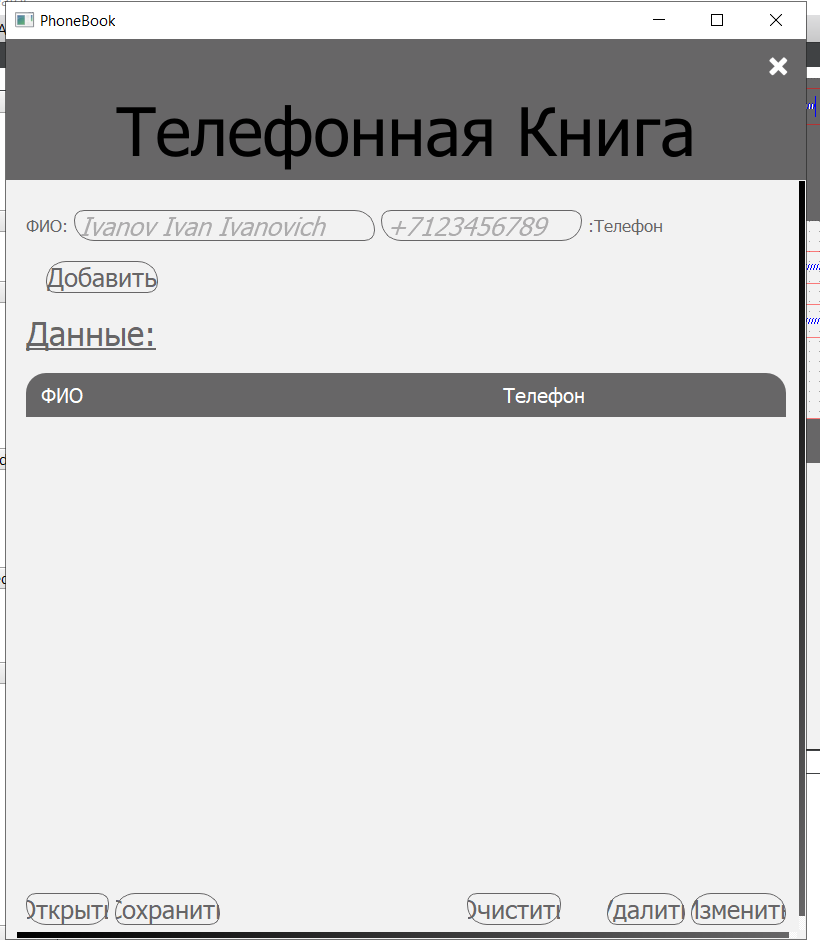


Рисунок 1. Интерфейс программы



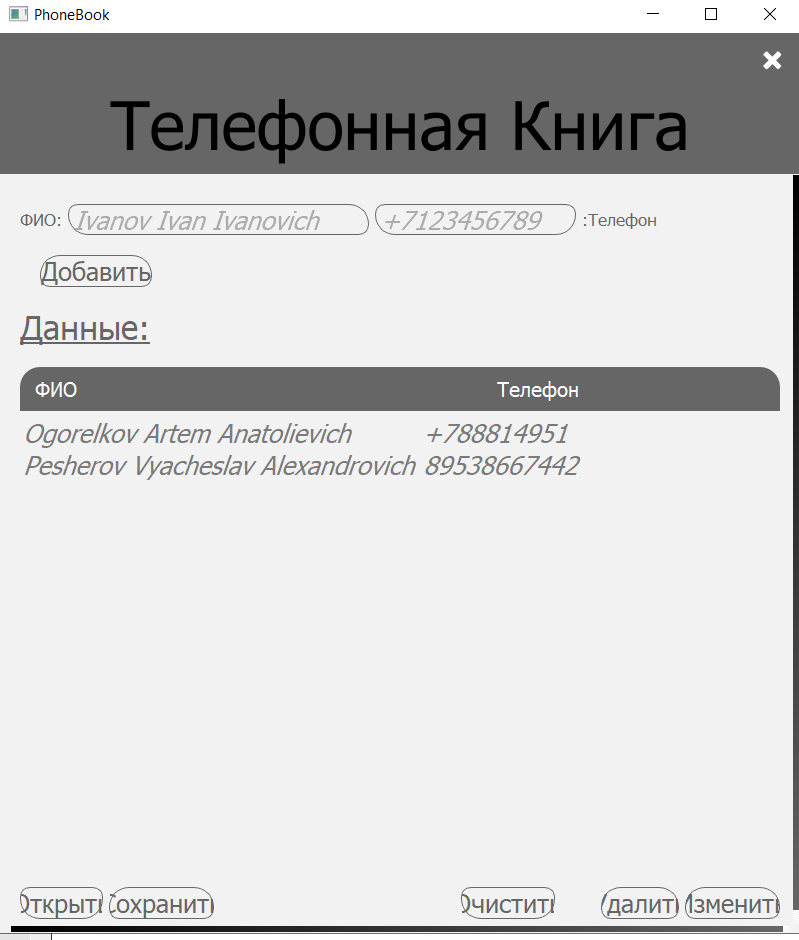


Рисунок 2,3. Добавление записей

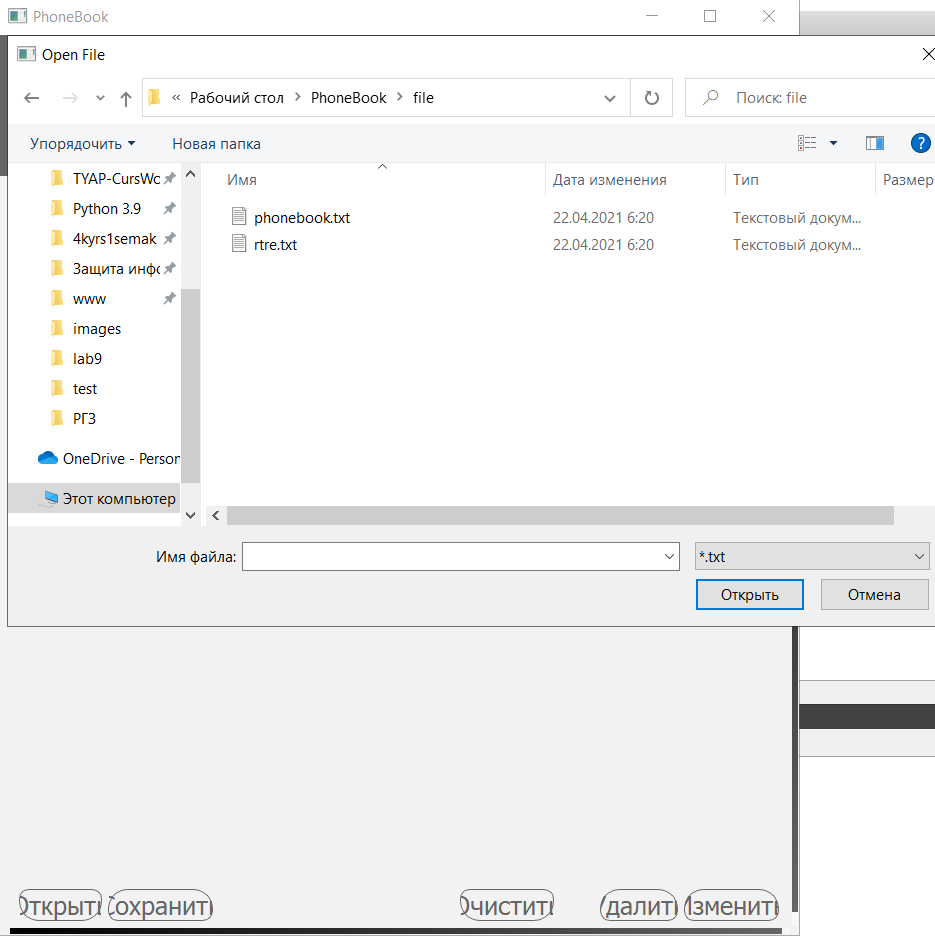


Рисунок 4. Открытие уже в заранее созданной базы данных.

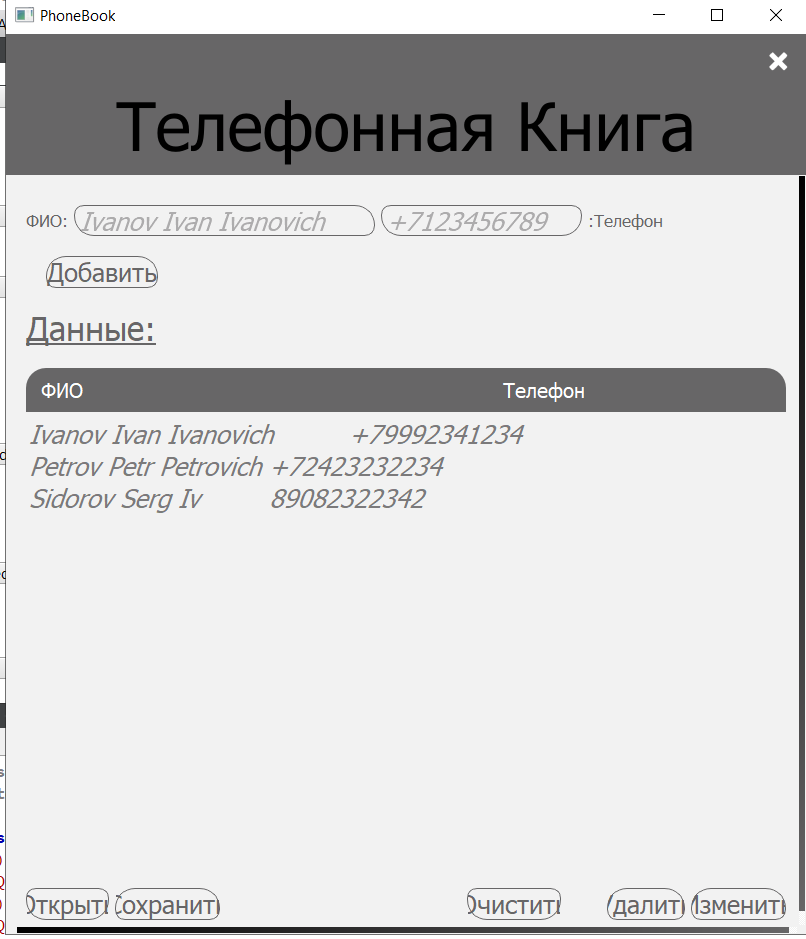


Рисунок 5. Загрузка выбранной базы данных

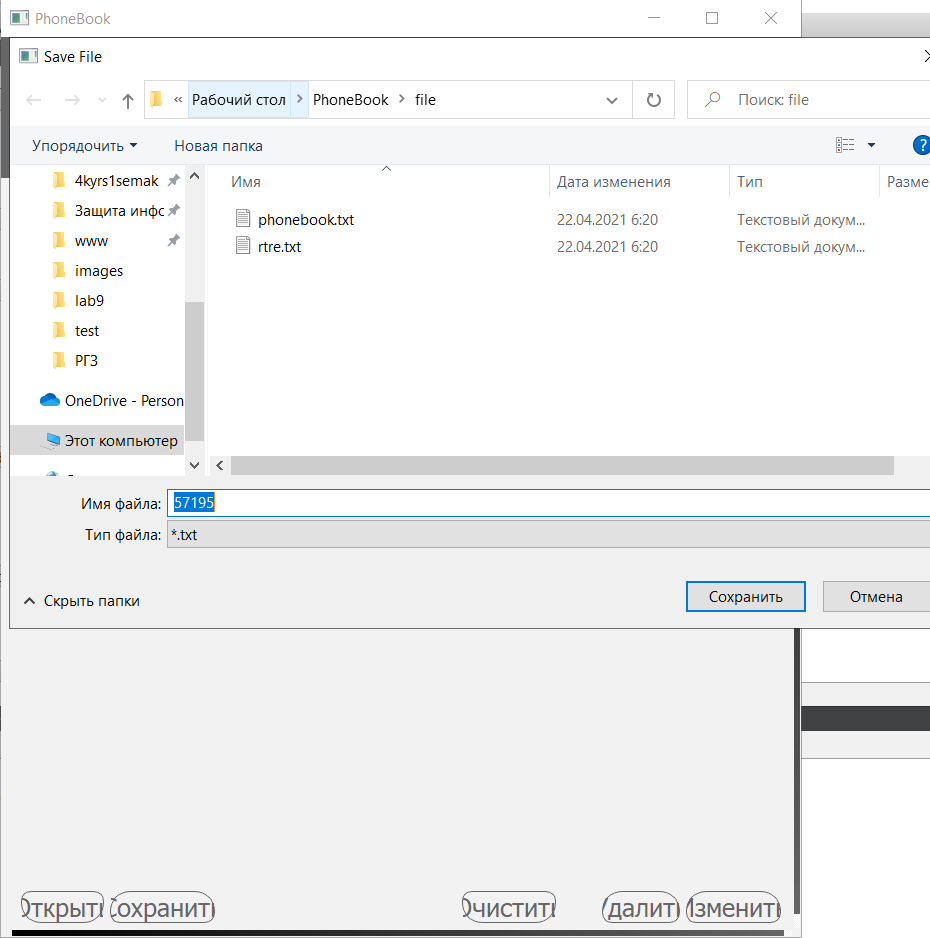


Рисунок 6. Сохранение данных

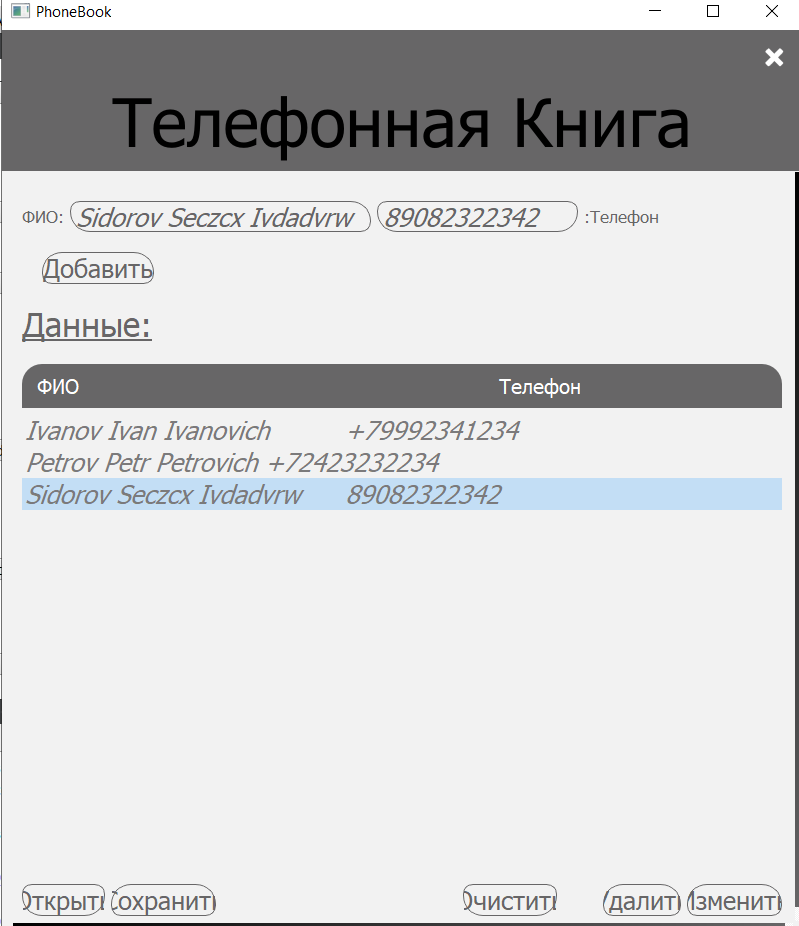


Рисунок 7. Редактирование записи

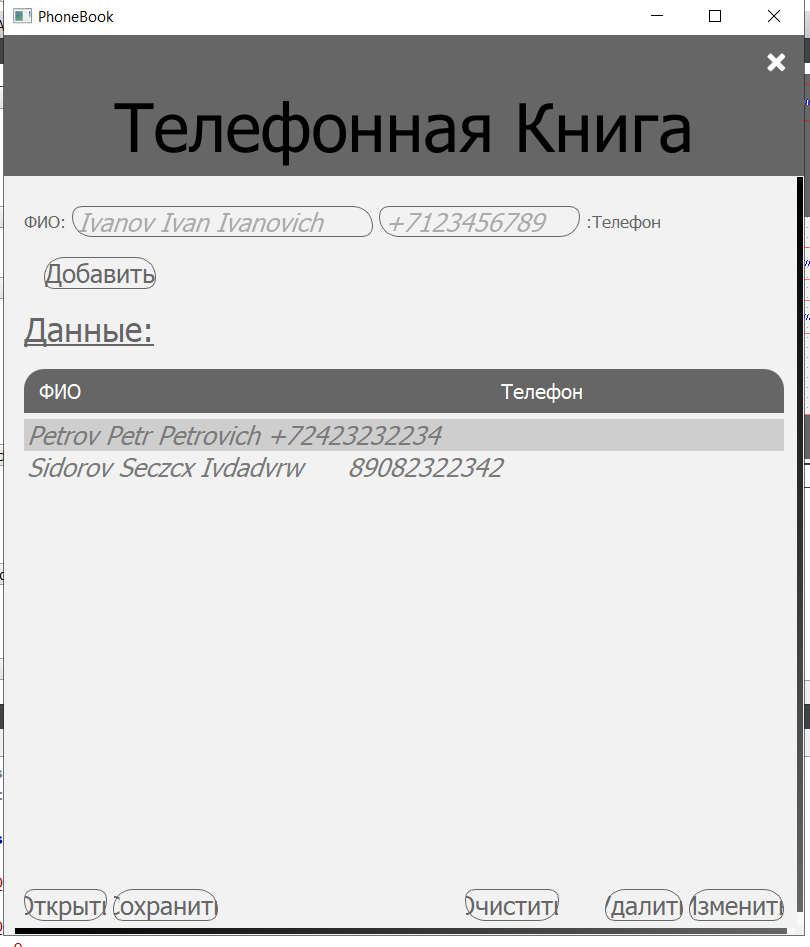


Рисунок 8. Удаление одной из записей

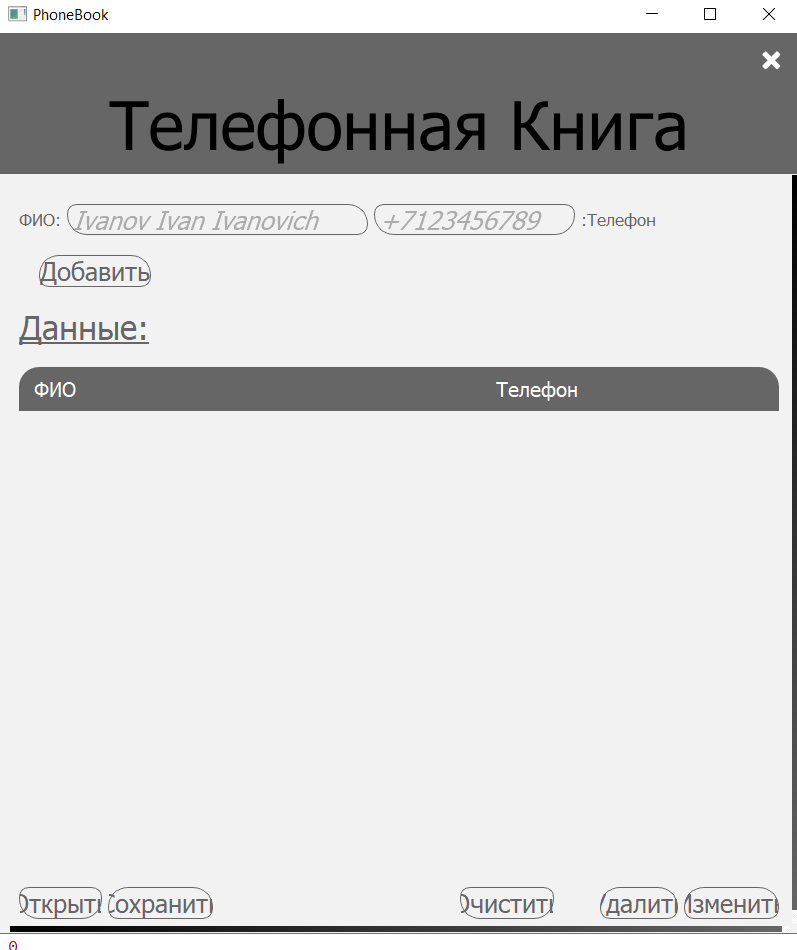


Рисунок 9. Удаление всех записей

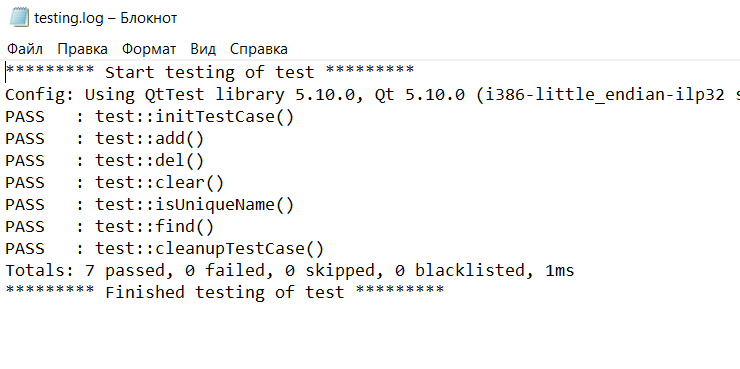


Рисунок 10. Тесты

# **Вывод**

Мы научились работать в среде QT, а также разрабатывать в ней приложения и модульные тесты для тестирования наших функций и классов языка Си. Данная среда также как и Visual Studio отлично подходит для создание и выполнения тестирования и автоматизации.

# **Список литературы**

1. Подбельский В.В., Фомин С.С.инт Курс программирования на языке Си: учебник. – М.:ДМК Пресс, 2012 – 384 с.
2. Павловская Т.А. C#. Программирование на языке высокого уровня: Учебник для вузов. - СПб. : Питер, 2014 - 432 с. : ил. - (Серия "Учебник для вузов").
3. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft.NET Framework 4 на языке C# . 3-е изд.: - СПб.:Питер, 2012 - 928 с. : ил.
4. Котляров, В. П. Основы тестирования программного обеспечения : учебное пособие для СПО / В. П. Котляров. — Саратов : Профобразование, 2019 — 335 c. — ISBN 978-5-4488-0364-2. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/86202.html (дата обращения: 21.08.2020). — Режим доступа: для авторизир. пользователей

# **Приложение**

## Листинг 1. test.cpp

#include "test.h"

#include "phoneb.h"

#include <QtTest/QTest>

test::test(QObject \*parent) : QObject(parent) {}

void test::add()

{

phoneB p;

p.add("ivan", "+1234567890");

QMultiMap<QString, QString> res;

res.insert("ivan", "+1234567890");

QCOMPARE(res, p.getPhoneBook());

res.insert("iv", "+1234567890");

bool check = res == p.getPhoneBook();

QCOMPARE(false, check);

}

void test::del() {

phoneB p;

p.add("ivan", "+1234567890");

p.add("ivan1", "+1234567890");

int res = 2;

QCOMPARE(res, p.getPhoneBook().size());

p.del("ivan");

res = 1;

QCOMPARE(res, p.getPhoneBook().size());

}

void test::clear()

{

phoneB p;

p.add("ivan", "+1234567890");

p.add("ivan1", "+1234567890");

bool res = true;

p.clear();

QCOMPARE(res, p.getPhoneBook().isEmpty());

}

void test::isUniqueName()

{

phoneB p;

p.add("ivan", "+1234567890");

p.add("ivan1", "+1234567890");

bool res = true;

QCOMPARE(res, p.isUniqueName("ivan"));

res = false;

QCOMPARE(res, p.isUniqueName("petr"));

}

void test::find()

{

phoneB p;

p.add("petr", "+242342323423");

p.add("ivan1", "+1234567890");

p.add("ivan", "+1234567890");

int index = 2;

QCOMPARE(index, p.find("petr"));

index = 0;

QCOMPARE(index, p.find("ivan"));

}

//QTEST\_MAIN(test)

## Листинг 2. phonebook.cpp

#include "phonebook.h"

#include "ui\_phonebook.h"

PhoneBook::PhoneBook(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::PhoneBook)

{

ui->setupUi(this);

// this->setWindowFlags(Qt::MSWindowsFixedSizeDialogHint | Qt::CustomizeWindowHint | Qt::FramelessWindowHint);

this->setAttribute(Qt::WA\_TranslucentBackground, true);

ui->lEditPhoneNumber->setValidator(new QRegExpValidator(QRegExp("\\+?[0-9]+"), this));

ui->lEditFullname->setValidator(new QRegExpValidator(QRegExp("[A-Z][a-z]+ [A-Z][a-z]+ [A-Z][a-z]+"), this));

ui->btnFind->hide();

pbObj = new phoneB;

}

PhoneBook::~PhoneBook()

{

delete ui;

delete pbObj;

pbObj = NULL;

}

bool PhoneBook::isUserInputValid(const QString &fullname, const QString &phoneNumber)

{

if (fullname.isEmpty() || phoneNumber.isEmpty())

{

ui->lblError->setText("You need to fill in both fields!");

return false;

}

ui->lblError->setText("");

return true;

}

void PhoneBook::clearInputFields()

{

ui->lEditFullname->clear();

ui->lEditPhoneNumber->clear();

}

void PhoneBook::on\_btnAdd\_clicked()

{

QString fullname = ui->lEditFullname->text();

QString phone = ui->lEditPhoneNumber->text();

if (pbObj->isUniqueName(fullname))

{

ui->lblError->setText("Fullname must be unique!");

return;

}

else

{

ui->lblError->setText("");

}

if (isUserInputValid(fullname, phone))

{

pbObj->add(fullname, phone);

ui->listWidget->addItem(fullname + "\t" + phone);

ui->listWidget->sortItems();

clearInputFields();

}

}

void PhoneBook::open(const QString &filename)

{

QFile file(filename);

if (file.open(QIODevice::ReadOnly | QIODevice::Text))

{

pbObj->clear();

ui->listWidget->clear();

QTextStream in(&file);

QStringList lines = in.readAll().split("\n");

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

{

if(!lines[i].isEmpty())

{

QStringList list = lines[i].split(" ; ");

pbObj->add(list[0], list[1]);

ui->listWidget->addItem(list[0] + "\t" + list[1]);

}

}

ui->lblError->setText("");

file.close();

}

else

{

ui->lblError->setText("Error opening file");

}

}

void PhoneBook::on\_btnOpen\_clicked()

{

filename = QFileDialog::getOpenFileName(0, "Open File", "", "\*.txt");

open(filename);

}

void PhoneBook::save(const QString &filename)

{

QFile file(filename);

if (file.open(QIODevice::WriteOnly | QIODevice::Text))

{

QTextStream out(&file);

QMultiMap<QString, QString> phoneBoox = pbObj->getPhoneBook();

for(QMultiMap<QString, QString>::iterator it = phoneBoox.begin(); it != phoneBoox.end(); it++)

{

out << it.key() << " ; " << it.value() << "\n";

}

ui->lblError->setText("");

file.close();

}

else

{

ui->lblError->setText("Error writing to file");

}

}

void PhoneBook::on\_btnSave\_clicked()

{

filename = QFileDialog::getSaveFileName(0, "Save File", "", "\*.txt");

save(filename);

}

void PhoneBook::on\_btnFind\_clicked()

{

//setVisible(false);

int index = pbObj->find(ui->lEditFullname->text());

if (index == -1)

{

ui->lblError->setText("Cannot find the name");

}

else

{

ui->listWidget->setCurrentRow(index);

}

}

void PhoneBook::on\_btnEdit\_clicked()

{

QString fullname = ui->lEditFullname->text();

QString phone = ui->lEditPhoneNumber->text();

if (pbObj->isUniqueName(fullname))

{

ui->lblError->setText("Fullname must be unique!");

ui->lEditFullname->setFocus();

return;

}

if (isUserInputValid(fullname, phone))

{

deleteItem();

pbObj->add(fullname, phone);

ui->listWidget->addItem(fullname + "\t" + phone);

ui->listWidget->sortItems();

}

}

void PhoneBook::deleteItem()

{

QList<QListWidgetItem \*> selectedItems = ui->listWidget->selectedItems();

for (int i = 0; i < selectedItems.size(); ++i)

{

QStringList fields = selectedItems[i]->text().split('\t');

QString name = fields[0];

pbObj->del(name);

ui->listWidget->removeItemWidget(selectedItems[i]);

delete selectedItems[i];

}

clearInputFields();

}

void PhoneBook::on\_btnDelete\_clicked()

{

deleteItem();

}

void PhoneBook::on\_listWidget\_itemClicked(QListWidgetItem \*item)

{

int position = item->text().toStdString().find("\t");

std::string name = item->text().toStdString().erase(position, item->text().toStdString().length());

std::string phoneNumber = item->text().toStdString().erase(0, position + 1);

ui->lEditFullname->setText(QString::fromStdString(name));

ui->lEditPhoneNumber->setText(QString::fromStdString(phoneNumber));

}

void PhoneBook::on\_btnClear\_clicked()

{

ui->listWidget->clear();

pbObj->clear();

clearInputFields();

}

## Листинг 3.phoneb.cpp

#include "phoneb.h"

phoneB::phoneB()

{

}

QMultiMap<QString, QString> phoneB::getPhoneBook() const

{

return phoneBook;

}

void phoneB::setPhoneBook(const QMultiMap<QString, QString> &value)

{

phoneBook = value;

}

void phoneB::clear()

{

phoneBook.clear();

}

bool phoneB::isUniqueName(const QString &fullname)

{

auto result = phoneBook.find(fullname);

if (result == phoneBook.end())

{

return false;

}

return true;

}

void phoneB::add(const QString &fullname, const QString &phone)

{

phoneBook.insert(fullname, phone);

}

int phoneB::find(const QString &fullname)

{

int i = 0;

for(QMultiMap<QString, QString>::iterator it = phoneBook.begin(); it != phoneBook.end(); it++, i++)

{

if(it.key().contains(fullname))

{

return i;

}

}

return -1;

}

void phoneB::del(const QString &fullname)

{

phoneBook.remove(fullname);

}

phoneB::~phoneB()

{

}