Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет

телекоммуникаций и информатики»

Кафедра ПМиК

Современные технологии программирования 2

Проект “Телефонная книга”

Выполнил: студенты 4 курса

Ф. ИВТ, группа: ИП-711

Мартасов И. О.

Панченко Д.Б.

Скирда А.А

Проверил: ассистент кафедры ПМиК

Агалаков Антон Александрович

Новосибирск, 2021

**Содержание**

1. Задание.
2. Текст программы.
3. Тестовые наборы данных.

**Задание**

Тема:

разработка приложения под Windows в технологии ООП.

Цель:

Объектно-ориентированный анализ, проектирование и реализация 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.

**Текст программы**

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()

{

}

Phoneb.h

#ifndef PHONEB\_H

#define PHONEB\_H

#include <QMultiMap>

#include <QString>

class phoneB

{

QMultiMap <QString, QString> phoneBook;

public:

phoneB();

void add(const QString &fullname, const QString &phone);

int find(const QString &fullname);

void del(const QString &fullname);

bool isUniqueName(const QString &fullname);

void clear();

~phoneB();

QMultiMap<QString, QString> getPhoneBook() const;

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

};

#endif // PHONEB\_H

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));

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()

{

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();

}

Phonebook.h

#ifndef PHONEBOOK\_H

#define PHONEBOOK\_H

#include <QMainWindow>

#include <QFileDialog>

#include <QTextStream>

#include <QListWidgetItem>

#include "phoneb.h"

namespace Ui

{

class PhoneBook;

}

class PhoneBook : public QMainWindow

{

Q\_OBJECT

public:

explicit PhoneBook(QWidget \*parent = 0);

~*PhoneBook*();

bool isUserInputValid(const QString &fullname, const QString &phoneNumber);

void open(const QString &filename);

void save(const QString &filename);

void clearInputFields();

void deleteItem();

private slots:

void on\_btnAdd\_clicked();

void on\_btnOpen\_clicked();

void on\_btnSave\_clicked();

void on\_btnFind\_clicked();

void on\_btnEdit\_clicked();

void on\_btnDelete\_clicked();

void on\_btnClear\_clicked();

void on\_listWidget\_itemClicked(QListWidgetItem \*item);

private:

Ui::PhoneBook \*ui;

phoneB \*pbObj = NULL;

QString filename;

};

#endif // PHONEBOOK\_H

**Тестовые наборы данных**

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"));

index = -1;

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

}

Test.h

#ifndef TEST\_H

#define TEST\_H

#include <QObject>

class test : public QObject

{

Q\_OBJECT

public:

explicit test(QObject \*parent = 0);

private slots:

void add();

void del();

void clear();

void isUniqueName();

void find();

};

#endif // TEST\_H