МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

КАФЕДРА «ИИТ»

Лабораторная работа №3

По дисциплине «ОСиСП»

Выполнила:

студент ФЭИС

группы ПО-5

Нерода А. А.

Проверил:

Дряпко А. В.

Брест 2021

**Цель: научиться разрабатывать и использовать динамические библиотеки (DLL) с использованием Qt.**

**Задание:**

Доработать программу, разработанную в лабораторной работе No1-2, внеся следующие изменения:

1) Выбрать 3 вспомогательные функции и вынести их описание и реализацию в динамическую

библиотеку helper.dll. В основном приложении осуществить загрузку реализованных функций

во время работы программы (at run-time, с использованием объекта QLibrary) и их вызов.

2) Выбрать вспомогательный класс и вынести его описание и реализацию в динамическую биб-

лиотеку helper\_class.dll. В основном приложении осуществить загрузку реализованного класса

во время компиляции (at compile-time).

3) Реализовать окно «О программе» в виде объекта динамической библиотеки about.dll с указа-

нием автора программы, группы, курса и краткого описания разработанного приложения.

Осуществить импорт указанной библиотеки и отображение соответствующего окна при выборе

пункта меню «О программе».

4) Реализовать расширения для приложения, позволяющие изменять оформление пунктов меню

(шрифт, размер, начертание и т.д.). Соответствующие изменения должны происходить при

выборе специального пункта меню. Создать как минимум три расширения такого типа.**Код программы:**

**main.cpp**  
  
#include "player.h"

#include <QApplication>

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

QApplication a(argc, argv);

Player w;

w.show();

*return* a.exec();

}

**player.cpp**

#include "player.h"

#include "ui\_player.h"

#include "about.h"

#include "helper.h"

*typedef* QString (\*Helper\_set\_name\_window)(); *//установка* *имени* *окна*

*typedef* QString (\*Helper\_set\_first\_name)(); *//установка* *типа*

*typedef* QString (\*Helper\_set\_second\_name)(); *//установка* *пути*

Player::Player(QWidget \*parent) : QMainWindow(parent), ui(*new* Ui::Player) {

ui->setupUi(*this*);

*//* *установка* *имени* *окна* *приложения*

QLibrary \*helper\_set = *new* QLibrary("Helper");

Helper\_set\_name\_window set\_window\_name\_ = (Helper\_set\_name\_window)helper\_set->resolve("set\_window\_name");

QString windowTitle = set\_window\_name\_();

*delete* helper\_set;

*this*->setWindowTitle(windowTitle);

*//* *установка* *имени* *первого* *заголовка* *playlistview*

QLibrary \*helper\_set\_f\_n = *new* QLibrary("Helper");

Helper\_set\_first\_name set\_window\_name\_first = (Helper\_set\_first\_name)helper\_set\_f\_n->resolve("set\_first\_name\_HeaderLabels");

QString set\_name\_first = set\_window\_name\_first();

*delete* helper\_set\_f\_n;

*//* *установка* *имени* *второго* *заголовка* *playlistview*

QLibrary \*helper\_set\_s\_n = *new* QLibrary("Helper");

Helper\_set\_second\_name set\_window\_name\_second = (Helper\_set\_second\_name)helper\_set\_s\_n->resolve("set\_second\_name\_HeaderLabels");

QString set\_name\_second = set\_window\_name\_second();

*delete* helper\_set\_s\_n;

ui->just\_playingTrack->setVisible(*false*);

playListModel = *new* QStandardItemModel(*this*);

ui->playlistView->setModel(playListModel);

playListModel->setHorizontalHeaderLabels(QStringList() << tr(set\_name\_first.toStdString().data())<< tr(set\_name\_second.toStdString().data()));

ui->playlistView->hideColumn(1);

ui->playlistView->verticalHeader()->setVisible(*false*);

ui->playlistView->setSelectionMode(QAbstractItemView::SingleSelection); *//* *Разрешаем* *выделять* *только* *одну* *строку*

ui->playlistView->setEditTriggers(QAbstractItemView::NoEditTriggers); *//* *Отключаем* *редактирование*

ui->playlistView->horizontalHeader()->setStretchLastSection(*true*); *//* *по* *ширине* *строки*

player = *new* QMediaPlayer(*this*);

playlist = *new* QMediaPlaylist(player);

player->setPlaylist(playlist);

playlist->setPlaybackMode(QMediaPlaylist::Loop);

load\_mode\_plugins();

connect(playlist, &QMediaPlaylist::currentIndexChanged, [*this*](int index) {

int delLast = count\_tracks\_num;

Next = index;

Back = index;

*if*(isNext) {

ui->playlistView->model()->setData(ui->playlistView->model()->index(Next, 0),QColor(255, 125, 0), Qt::BackgroundRole);

ui->playlistView->model()->setData(ui->playlistView->model()->index(--Back, 0),QColor(255, 255, 255), Qt::BackgroundRole);

}

*else* {

ui->playlistView->model()->setData(ui->playlistView->model()->index(Next, 0),QColor(255, 125, 0), Qt::BackgroundRole);

ui->playlistView->model()->setData(ui->playlistView->model()->index(++Back, 0),QColor(255, 255, 255), Qt::BackgroundRole);

}

ui->just\_playingTrack->setText(playListModel->data(playListModel->index(index, 0)).toString());

*if*(index == 0) ui->playlistView->model()->setData(ui->playlistView->model()->index(--delLast, 0),QColor(255, 255, 255), Qt::BackgroundRole);

*if*(index == --delLast) ui->playlistView->model()->setData(ui->playlistView->model()->index(0, 0),QColor(255, 255, 255), Qt::BackgroundRole);

}); *//подсветка* *проигрываемого* *трека*

}

void Player::**load\_mode\_plugins**() *//загрузка* *плагинов*

{

QDir dir(qApp->applicationDirPath());

dir.cd("plugins");

QStringList filter;

filter << "\*.dll";

QPluginLoader loader;

*foreach*(QFileInfo info, dir.entryInfoList(filter)) {

loader.setFileName(info.absoluteFilePath());

Interface\* mode = qobject\_cast<Interface\*>(loader.instance());

*if*(mode) {

QString name = loader.metaData().value("MetaData").toObject().value("Mode\_name").toString();

ui->mode\_list->addItem(name);

\_mode.append(mode);

}

}

}

Player::~**Player**() {

*delete* ui;

*delete* playListModel;

*delete* playlist;

*delete* player;

}

void Player::**on\_btn\_add\_clicked**() {

QStringList new\_files = QFileDialog::getOpenFileNames(*this*, tr("Open files"), QString(), tr("Audio Files (\*.mp3)"));

*if*(!new\_files.isEmpty()) {

QStringList write\_to\_globallist;

*if*(count\_tracks\_num != 0) {

*for*(int i = 0; i < new\_files.size(); i++) {

bool flag = *false*;

*for*(int j = 0; j < files\_list.size(); j++) {

*if*(new\_files[i] == files\_list[j]) {

flag = *true*;

}

}

*if*(!flag) {

write\_to\_globallist.append(new\_files[i]);

}

}

add\_playlist(write\_to\_globallist);

}

*else* add\_playlist(new\_files);

}

}

void Player::add\_playlist(QStringList list) {

*//* *устанавливаем* *данные* *по* *именам* *и* *пути* *к* *файлам*

*//* *в* *плейлист* *и* *таблицу* *отображающую* *плейлист*

*foreach* (QString filePath, list) {

files\_list.append(list);

count\_tracks\_num++;

QList<QStandardItem \*> items;

items.append(*new* QStandardItem(QDir(filePath).dirName()));

items.append(*new* QStandardItem(filePath));

playListModel->appendRow(items);

playlist->addMedia(QUrl(filePath));

}

ui->count\_tracks->setText(QString::number(count\_tracks\_num));

}

void Player::**on\_btn\_play\_clicked**() {

player->play();

ui->just\_playingTrack->setVisible(*true*);

}

void Player::**on\_btn\_previous\_clicked**() {

isNext = *false*;

playlist->previous();

}

void Player::**on\_btn\_pause\_clicked**() {

player->pause();

}

void Player::**on\_btn\_stop\_clicked**() {

player->stop();

ui->just\_playingTrack->setVisible(*false*);

}

void Player::**on\_btn\_next\_clicked**() {

isNext = *true*;

playlist->next();

}

void Player::on\_playlistView\_doubleClicked(*const* QModelIndex index) {

*for* (int ind = 0; ind < count\_tracks\_num; ++ind) {

ui->playlistView->model()->setData(ui->playlistView->model()->index(ind, 0),QColor(255, 255, 255), Qt::BackgroundRole);

}

playlist->setCurrentIndex(index.row());

on\_btn\_play\_clicked();

}

void Player::**on\_Info\_clicked**() { *//информация* *о* *программе*

*typedef* void (\*About)();

QLibrary \*aboutLib = *new* QLibrary("About");

About showWindowAboutProgramm = (About)aboutLib->resolve("about");

showWindowAboutProgramm();

*delete* aboutLib;

}

void Player::**on\_reboot\_clicked**() { *//перезагрузка* *программы*

qApp->quit();

QProcess::startDetached(qApp->arguments()[0], qApp->arguments());

}

void Player::**on\_apply\_clicked**() { *//применить*

QApplication::setFont(\_mode.at(ui->mode\_list->currentIndex())->changeView());

}

**player.h**

#ifndef PLAYER\_H

#define PLAYER\_H

#include <QMainWindow>

#include <QFileDialog>

#include <QStandardItemModel>

#include <QMediaPlayer>

#include <QMediaPlaylist>

#include <QProcess>

#include <QLibrary>

#include <QPluginLoader>

#include "interface.h"

QT\_BEGIN\_NAMESPACE

*namespace* **Ui** { *class* **Player**; }

QT\_END\_NAMESPACE

*class* Player : *public* QMainWindow {

Q\_OBJECT

*public*:

Player(QWidget \*parent = *nullptr*);

~Player();

void load\_mode\_plugins();

*private* *slots*:

void add\_playlist(QStringList list);

void on\_btn\_add\_clicked();

void on\_btn\_play\_clicked();

void on\_btn\_previous\_clicked();

void on\_btn\_pause\_clicked();

void on\_btn\_stop\_clicked();

void on\_btn\_next\_clicked();

void on\_playlistView\_doubleClicked(*const* QModelIndex index);

void on\_Info\_clicked();

void on\_reboot\_clicked();

void on\_apply\_clicked();

*private*:

Ui::Player \*ui;

QList<Interface\*> \_mode;

QStringList files\_list;

int count\_tracks\_num = 0;

QStandardItemModel \*playListModel;

QMediaPlayer \*player;

QMediaPlaylist \*playlist;

int Next, Back;

bool isNext = *true*;

};

#endif *//* *PLAYER\_H*

**about.h**

#ifndef ABOUT\_H

#define ABOUT\_H

#include "about\_global.h"

*extern* "C" ABOUT\_EXPORT void about();

#endif *//* *ABOUT\_H*

**about\_global.h**

#ifndef ABOUT\_GLOBAL\_H

#define ABOUT\_GLOBAL\_H

#include <QtCore/qglobal.h>

#if defined(ABOUT\_LIBRARY)

# define ABOUT\_EXPORT Q\_DECL\_EXPORT

#else

# define ABOUT\_EXPORT Q\_DECL\_IMPORT

#endif

#endif *//* *ABOUT\_GLOBAL\_H*

**helper.h**

#ifndef HELPER\_H

#define HELPER\_H

#include "helper\_global.h"

#include <QString>

*extern* "C" HELPER\_EXPORT QString set\_window\_name();

*extern* "C" HELPER\_EXPORT QString set\_first\_name\_HeaderLabels();

*extern* "C" HELPER\_EXPORT QString set\_second\_name\_HeaderLabels();

#endif *//* *HELPER\_H*

**helper\_global.h**

#ifndef HELPER\_GLOBAL\_H

#define HELPER\_GLOBAL\_H

#include <QtCore/qglobal.h>

#if defined(HELPER\_LIBRARY)

# define HELPER\_EXPORT Q\_DECL\_EXPORT

#else

# define HELPER\_EXPORT Q\_DECL\_IMPORT

#endif

#endif *//* *HELPER\_GLOBAL\_H*

**Результаты работы программы:**

