МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

 «БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра ИИТ

ЛАБОРАТОРНАЯ РАБОТА №3

По дисциплине: «ОСИСП»

Выполнил:

Студент ФЭИС

3-го курса, группы ПО-5

Пищик А.В

Проверила:

Дряпко А.В.

Брест 2021

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

Вариант 15

*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 компиляциии..

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

***mainwindow.h:***

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QProcess>

#include "interface.h"

#include <QLocale>

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

void load\_mode\_plugins();

private slots:

void on\_processor\_clicked(bool checked);

void result\_update(int size\_cols, QStringList header, QStringList mass);

void update\_str(string s);

void on\_mother\_clicked();

void on\_os\_clicked();

void on\_info\_programm\_clicked();

void on\_ram\_clicked();

void on\_hard\_clicked();

void on\_apply\_clicked();

void on\_reboot\_clicked();

private:

Ui::MainWindow \*ui;

QList<Interface\*> \_mode;

};

#endif // MAINWINDOW\_H

***main.cpp:***

#include "mainwindow.h"

#include <QApplication>

#include <QLabel>

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

{

qputenv("QT\_DEBUG\_PLUGINS", QByteArray("1"));

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

***mainwindow.cpp***

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "helper.h"

#include "about.h"

#include <QDir>

#include <QJsonObject>

#include <QPluginLoader>

#include <QDebug>

#include <Windows.h>

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

{

AllocConsole();

ShowWindow(GetConsoleWindow(), SW\_HIDE); // hide console

ui->setupUi(this);

ui->tableWidget->setEditTriggers(0);

bool checked = true;

on\_processor\_clicked(checked);

load\_mode\_plugins();

}

string \*res = new string[100];

QStringList header, result\_data, mass;

int size\_str = 0;

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::load\_mode\_plugins()

{

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();

qDebug() << name;

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

\_mode.append(mode);

}

}

}

void MainWindow::result\_update(int size\_cols, QStringList header, QStringList mass) {

string result[size\_cols];

string s;

ifstream f1;

f1.open("test1.txt");

int i = 0;

while (!f1.eof()) {

getline(f1, s);

result[i] = s;

i++;

}

f1.close();

Helper a;

a.destroy\_file("test1.txt");

for (int i = 0; i < size\_cols; i++)

{

res[i] = result[i];

}

ui->tableWidget->setRowCount(size\_cols); // Устанавливаем количество строк на size\_cols

ui->tableWidget->setColumnCount(1);

for(int i = 0; i < size\_cols; i++) {

ui->tableWidget->verticalHeader()->setSectionResizeMode(i, QHeaderView::Stretch);

}

ui->tableWidget->horizontalHeader()->setSectionResizeMode(0, QHeaderView::Stretch);

ui->tableWidget->setVerticalHeaderLabels(header);

ui->tableWidget->horizontalHeader()->setVisible(false);

for(int i = 0; i < size\_cols; i++) {

QString data = QString::fromStdString(res[i]);

result\_data.append(data);

}

int countt = 0;

for (int i = countt; i < ui->tableWidget->rowCount(); ++i) {

QTableWidgetItem \* item;

for (int j = 0; j < ui->tableWidget->columnCount(); ++j) {

item = new QTableWidgetItem;

for(int s = 0; s < size\_str; s++) {

if(i==s)

item->setText(result\_data.value(mass[s].toInt()));

ui->tableWidget->setItem(i, j, item);

}

}

}

countt++;

ui->tableWidget->resizeColumnsToContents();

ui->tableWidget->resizeRowsToContents();

}

void MainWindow::update\_str(string s) {

string new\_str;

bool flag = false;

int flag1 = 0;

for (int i = 0; i < s.length(); i++) {

if (s[i] == '|') {

flag = false;

flag1++;

}

if (flag1 == 1) {

new\_str += '\n';

flag1++;

}

if (flag == true) {

new\_str += s[i];

}

if (s[i] == '\n' || (s[i] == ' ' && s[++i] == ' ')) {

flag = false;

}

if (s[i] == '=') {

size\_str++;

flag = true;

}

}

ofstream ff;

ff.open("test1.txt");

ff << new\_str;

ff.close();

}

void MainWindow::on\_processor\_clicked(bool checked) {

size\_str = 0;

ui->tableWidget->setVisible(true);

ui->switch\_info->setText("Информация о процессоре");

header.clear(), result\_data.clear(), mass.clear();

Helper d;

const char \*take = d.set\_apply\_do(0).data(); //proc

system(take);

Helper a;

update\_str(a.read\_file\_info());

ui->tableWidget->setRowCount(0);

ui->tableWidget->setColumnCount(0);

for(int i = 0; i < size\_str; i++) {

QString m = QVariant(i).toString();

mass.append(m);

}

result\_update(size\_str, header<<"Частота на ядро"<<"Имя процессора"<<"Количество ядер"<<"Количество потоков", mass);

}

void MainWindow::on\_mother\_clicked() {

size\_str = 0;

ui->tableWidget->setVisible(true);

header.clear(), result\_data.clear(), mass.clear();

ui->switch\_info->setText("Информация о Материнской плате");

Helper d;

const char \*take = d.set\_apply\_do(1).data(); //mather

system(take);

Helper a;

update\_str(a.read\_file\_info());

ui->tableWidget->setRowCount(0);

ui->tableWidget->setColumnCount(0);

for(int i = 0; i < size\_str; i++) {

QString m = QVariant(i).toString();

mass.append(m);

}

result\_update(size\_str, header<<"Производитель"<<"Модель", mass);

}

void MainWindow::on\_os\_clicked() {

size\_str = 0;

ui->tableWidget->setVisible(true);

header.clear(), result\_data.clear(), mass.clear();

ui->switch\_info->setText("Информация об Операционной системе");

Helper d;

const char \*take = d.set\_apply\_do(4).data(); //OS

system(take);

Helper a;

update\_str(a.read\_file\_info());

ui->tableWidget->setRowCount(0);

ui->tableWidget->setColumnCount(0);

for(int i = 0; i < size\_str; i++) {

QString m = QVariant(i).toString();

mass.append(m);

}

result\_update(size\_str, header<<"Производитель OS"<<"Версия", mass);

}

void MainWindow::on\_ram\_clicked() {

size\_str = 0;

ui->tableWidget->setVisible(true);

header.clear(), result\_data.clear(), mass.clear();

ui->switch\_info->setText("Информация об Модулях оперативной памяти");

Helper d;

const char \*take = d.set\_apply\_do(2).data(); //ram

system(take);

Helper a;

update\_str(a.read\_file\_info());

ui->tableWidget->setRowCount(0);

ui->tableWidget->setColumnCount(0);

int r = size\_str/2;

for(int i = 0; i < r; i++) {

header<<"Партийный номер"<<"Частота";

}

for(int i = 0; i < size\_str; i++) {

QString m = QVariant(i).toString();

mass.append(m);

}

result\_update(size\_str, header, mass);

}

void MainWindow::on\_hard\_clicked() {

size\_str = 0;

ui->tableWidget->setVisible(true);

header.clear(), result\_data.clear(), mass.clear();

ui->switch\_info->setText("Информация об Накопителях");

Helper d;

const char \*take = d.set\_apply\_do(3).data(); //hard

system(take);

Helper a;

update\_str(a.read\_file\_info());

ui->tableWidget->setRowCount(0);

ui->tableWidget->setColumnCount(0);

int r = size\_str/2;

for(int i = 0; i < r; i++) {

header<<"Модель"<<"Размер";

}

for(int i = 0; i < size\_str; i++) {

QString m = QVariant(i).toString();

mass.append(m);

}

result\_update(size\_str, header, mass);

}

void MainWindow::on\_info\_programm\_clicked()

{

About ab(this);

ab.exec();

}

void MainWindow::on\_apply\_clicked()

{

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

}

void MainWindow::on\_reboot\_clicked()

{

qApp->quit();

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

}

***Mode.h***

#ifndef MODE\_H

#define MODE\_H

#include <QObject>

#include <interface.h>

class Mode : public QObject, public Interface

{

Q\_OBJECT

Q\_PLUGIN\_METADATA(IID "aida.Interface" FILE "Interface.json")

Q\_INTERFACES(Interface)

public:

Mode(QObject \*parent =0);

~Mode();

QString name();

virtual QFont changeView();

};

#endif // MODE\_H

***Mode.cpp***

#include "mode.h"

#include <QDebug>

Mode::Mode(QObject \*parent) : QObject(parent)

{

qDebug() << name() << "created";

}

Mode::~Mode()

{

qDebug() << name() << "destroy";

}

QString Mode::name()

{

return "mode 1";

}

QFont Mode::changeView()

{

qDebug() << name() << "print";

QFont newFont("Courier", 6, QFont::Bold, true);

return newFont;

}

***Interface.h***

#ifndef INTERFACE\_H

#define INTERFACE\_H

#include <QObject>

#include <QString>

#include <QFont>

class Interface

{

public:

virtual QFont changeView() = 0;//плагин выполняет действия над объектами в структуре Styles

};

Q\_DECLARE\_INTERFACE(Interface, "aida.Interface");

#endif // INTERFACE\_H

***Helper.h***

#ifndef HELPER\_H

#define HELPER\_H

#include "helper\_global.h"

#include <QString>

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

class HELPER\_EXPORT Helper

{

public:

Helper();

void destroy\_file(QString s);

string read\_file\_info();

string set\_apply\_do(int num);

};

#endif // HELPER\_H

***Helper.cpp***

#include "helper.h"

Helper::Helper()

{

}

void Helper::destroy\_file(QString s)

{

string name;

name = s.toStdString().c\_str();

const char \*r = name.data();

remove(r);

}

string Helper::read\_file\_info() {

ifstream f;

string s;

f.open("test.txt");

while (!f.eof()) {

char c = f.get();

s += c;

}

f.close();

destroy\_file("test.txt");

return s;

}

string Helper::set\_apply\_do(int num)

{

string \*mass = new string[5];

mass[0] = "wmic CPU get Name, NumberOfCores, NumberOfLogicalProcessors, Maxclockspeed /format:list >> test.txt"; //proc

mass[1] = "wmic baseboard get manufacturer, product /format:list >> test.txt"; //mather

mass[2] = "wmic memorychip get speed, PartNumber /format:list >> test.txt"; //ram

mass[3] = "wmic diskdrive get model, size /format:list >> test.txt"; //hard

mass[4] = "wmic OS get Manufacturer , Version /format:list >> test.txt"; //OS

return mass[num];

}

***About.h***

#ifndef ABOUT\_H

#define ABOUT\_H

#include "about\_global.h"

#include <QDialog>

class ABOUT\_EXPORT About : public QDialog

{

public:

explicit About(QWidget \*parent = nullptr);

};

#endif // ABOUT\_H

***About.cpp***

#include "about.h"

#include <QLabel>

#include <QGridLayout>

About::About(QWidget\* parent): QDialog(parent)

{

QGridLayout\* gridLayout = new QGridLayout(this);

setLayout(gridLayout);

gridLayout->addWidget(new QLabel("Программу сделал студент группы ПО-5, Пищик Алексей \nДанная программа выводит информацию о компьютере", this));

}

Вывод: научился разрабатывать и использовать динамические библиотеки (DLL) с использованием Qt.