Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Факультет электронно-информационных систем

Кафедра ИИТ

Лабораторная работа №1-2

за 5 семестр

По дисциплине: «ОСиСП»

Выполнил:

студент 3 курса

группы ПО-4(1)

Жук В. А.

Проверила:

Дряпко А.В.

2021

Лабораторная работа №1-2

Цель работы: приобрести практические навыки проектирования и разработки приложений с графическим пользовательским интерфейсом в ОС Windows средствами Qt.

Вариант 7

Задание:

Изображение выглядит как стол

Автоматически созданное описание

**Текст программы:**

1. **main.cpp**

#include "mainwindow.h"

#include <QApplication>

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

{

QApplication a(argc, argv);

MainWindow w;

w.show();

*return* a.exec();

}

1. **mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QDebug>

bool **remove\_line**(*const* char \*filename, int index) {

QVector<QString> vec;

QFile file(filename);

*if* (!file.*open*(QIODevice::*ReadOnly*))

*return* *false*;

QString str;

str = file.readAll();

vec = (str.split(QRegExp("[\n]")).toVector());

vec.pop\_back();

*if* (vec.size() <= index)

*return* *false*;

vec.erase(vec.begin() + index);

QFile out("list.txt");

*if* (!out.*open*(QIODevice::*WriteOnly*))

*return* *false*;

*for* (int i = 0; i < vec.size(); ++i) {

out.write(vec[i].toStdString().data());

out.write("\n");

}

out.*close*();

*return* *false*;

}

void MainWindow::**accept**(int index) {

int helpRow = ui->table\_toDo->rowCount();

helpRow--;

QFile read("list.txt");

*if* (!read.*open*(QIODevice::*ReadOnly*))

*return*;

*if*(ui->table\_toDo->item(helpRow,0)->text() != "" && read.*size*() != 0) {

ui->table\_toDo->item(index,0)->setBackground(Qt::*green*);

listAccept[currectList[row].toInt()] = '1';

save.clear();

currectList.clear();

listData.clear();

listTask.clear();

QString buffer;

buffer = read.readAll(); *//* *Буфер*

QStringList aa = buffer.split(QRegExp("[-\n]"));

aa.pop\_back();

*for* (int i = 0, j = 0; i < aa.size(); i += 3, j++) {

int indD = i;

int indT = i;

indT +=2;

listData.append(aa[indD]);

save.append(listData[j]);

save.append("-");

save.append(listAccept[j]);

save.append("-");

listTask.append(aa[indT]);

save.append(listTask[j]);

save.append("\n");

}

QFile file("list.txt");

*if* (!file.*open*(QIODevice::*WriteOnly*))

*return*;

*for* (int i = 0; i < save.size(); ++i) {

file.write(save[i].toStdString().data());

}

file.*close*();

loadingDoings();

}

read.*close*();

}

void MainWindow::**loadingDoings**() {

noticeTextToday.clear();

noticeTextOneDayNext.clear();

noticeTextThreeDaysNext.clear();

save.clear();

ui->table\_toDo->clear(); *//* *Очистка* *таблицы*

ui->table\_toDo->setRowCount(0); *//* *Удаляем* *все* *строки* *таблицы*

currectDo = 0; *//* *Обнуляем* *счетчик* *строк* *таблицы*

ui->table\_toDo->setHorizontalHeaderLabels(QStringList() << tr(ui->calendarWidget->selectedDate().toString().toStdString().data()));

currectList.clear();

listData.clear();

listAccept.clear();

listTask.clear();

QString buffer;

QFile read("list.txt");

*if* (!read.*open*(QIODevice::*ReadOnly*))

*return*;

*if*(read.*size*() != 0) {

buffer = read.readAll(); *//* *Буфер*

save = buffer.split('\*');

QStringList aa = buffer.split(QRegExp("[-\n]"));

aa.pop\_back();

*for* (int i = 0; i < aa.size(); i += 3) {

int indD = i;

int indA = i;

int indT = i;

indA +=1;

indT +=2;

listData.append(aa[indD]);

listAccept.append(aa[indA]);

listTask.append(aa[indT]);

}

}

*for* (int i = 0; i < listData.size(); ++i) {

*if*(ui->calendarWidget->selectedDate().toString().toStdString().data()==listData[i]) {

ui->table\_toDo->setRowCount(ui->table\_toDo->rowCount() + 1);

currectList.append(QString::number(i));

QTableWidgetItem \* item;

item = *new* QTableWidgetItem;

item->setText(listTask[i]);

ui->table\_toDo->setItem(currectDo, 0, *item*);

*if*(listAccept[i] == '1') {

ui->table\_toDo->item(currectDo,0)->setBackground(Qt::*green*);

}

currectDo++;

}

}

*for* (int i = 0; i < listData.size(); ++i) {

*if*(DateNext.toString()==listData[i] && listAccept[i] != '1') { *//* *Уведомления* *за* *день*

noticeTextOneDayNext += listTask[i];

noticeTextOneDayNext += '\n';

}

*if*(DateThreeNext.toString()==listData[i] && listAccept[i] != '1') { *//* *Уведомления* *за* *день*

noticeTextThreeDaysNext += listTask[i];

noticeTextThreeDaysNext += '\n';

}

*if*(QDate::currentDate().toString() ==listData[i] && listAccept[i] != '1') { *//* *Уведомления* *по* *текущей* *дате*

noticeTextToday += listTask[i];

noticeTextToday += '\n';

}

}

}

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

ui->setupUi(*this*);

currectDo = ui->table\_toDo->rowCount(); *//* *определяем* *количество* *уже* *записаных* *данных*

popUp = *new* PopUp();

DateNext=QDate::currentDate();

DateThreeNext=QDate::currentDate();

DateNext = DateNext.addDays(1);

DateThreeNext = DateThreeNext.addDays(3);

loadingDoings();

QStringList s;

s.append(QDate::currentDate().toString());

ui->table\_toDo->setHorizontalHeaderLabels(s);

ui->table\_toDo->horizontalHeader()->setStretchLastSection(*true*); *//* *по* *ширине* *строки*

*if*(noticeTextToday != "") {

timer = *new* QTimer();

connect(timer, SIGNAL(timeout()), *this*, SLOT(NoticeToday()));

timer->start(1000); *//* *И* *запустим* *таймер*

}

*else* {

*if*(noticeTextOneDayNext != "") {

timerNextDay = *new* QTimer();

connect(timerNextDay, SIGNAL(timeout()), *this*, SLOT(NoticeOneDayNext()));

timerNextDay->start(1000); *//* *И* *запустим* *таймер*

}

*else* {

*if*(noticeTextThreeDaysNext != "") {

timerThreeNextDay = *new* QTimer();

connect(timerThreeNextDay, SIGNAL(timeout()), *this*, SLOT(NoticeThreeDaysNext()));

timerThreeNextDay->start(1000); *//* *И* *запустим* *таймер*

}

}

}

}

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

*delete* ui;

}

void MainWindow::**on\_calendarWidget\_selectionChanged**() {

loadingDoings(); *//Загрука* *задач* *из* *файла*

}

void MainWindow::**on\_add\_doFortable\_clicked**() {

flag = *true*;

*if*(ui->table\_toDo->takeItem(currectList.size(), 0) == NULL) ui->table\_toDo->setRowCount(ui->table\_toDo->rowCount() + 1);

QTableWidgetItem \* item;

item = *new* QTableWidgetItem;

item->setText("");

ui->table\_toDo->setItem(currectDo, 0, *item*);

}

void MainWindow::**on\_showCurrectDate\_clicked**() {

ui->calendarWidget->setSelectedDate(QDate::currentDate());

loadingDoings();

}

void MainWindow::**on\_table\_toDo\_clicked**(*const* QModelIndex &index) {

row = index.row();

*if*(currectList.size() < row)

row = currectList[row].toInt();

}

void MainWindow::**on\_del\_doFortable\_clicked**() {

*if*(row >= 0 && ui->table\_toDo->rowCount() > 0) remove\_line("list.txt", row);

loadingDoings();

}

void MainWindow::**on\_save\_doFortable\_clicked**() {

*if*(flag) {

int helpRow = ui->table\_toDo->rowCount();

helpRow--;

*if*(ui->table\_toDo->item(helpRow,0)->text() != "") {

QFile file("list.txt");

*if* (!file.*open*(QIODevice::*WriteOnly*))

*return*;

*for* (int i = row; i < ui->table\_toDo->rowCount(); i++) {

save.append(ui->calendarWidget->selectedDate().toString().toStdString().data());

save.append("-");

save.append("0");

save.append("-");

save.append(ui->table\_toDo->item(i,0)->text().toStdString().data());

save.append("\n");

}

*for* (int i = 0; i < save.size(); ++i) {

file.write(save[i].toStdString().data());

}

file.*close*();

loadingDoings();

}

}

flag = *false*;

}

void MainWindow::**on\_accept\_clicked**() {

*if*(row >= 0 && ui->table\_toDo->rowCount() > 0) accept(row);

loadingDoings();

}

void MainWindow::**NoticeToday**() {

popUp->setPopupText("Сегодня необходимо :\n" + noticeTextToday);

popUp->show();

timer->stop();

*if*(noticeTextOneDayNext != "") {

timerNextDay = *new* QTimer();

connect(timerNextDay, SIGNAL(timeout()), *this*, SLOT(NoticeOneDayNext()));

timerNextDay->start(1000); *//* *И* *запустим* *таймер*

}

*else* {

*if*(noticeTextThreeDaysNext != "") {

timerThreeNextDay = *new* QTimer();

connect(timerThreeNextDay, SIGNAL(timeout()), *this*, SLOT(NoticeThreeDaysNext()));

timerThreeNextDay->start(1000); *//* *И* *запустим* *таймер*

}

}

}

void MainWindow::**NoticeThreeDaysNext**() {

popUp->setPopupText("Через 3 дня необходимо :\n" + noticeTextThreeDaysNext);

popUp->show();

timerThreeNextDay->stop();

}

void MainWindow::**NoticeOneDayNext**() {

popUp->setPopupText("Завтра необходимо :\n" + noticeTextOneDayNext);

popUp->show();

timerNextDay->stop();

*if*(noticeTextThreeDaysNext != "") {

timerThreeNextDay = *new* QTimer();

connect(timerThreeNextDay, SIGNAL(timeout()), *this*, SLOT(NoticeThreeDaysNext()));

timerThreeNextDay->start(1000); *//* *И* *запустим* *таймер*

}

}

1. **mybutton.h**

#include "popup.h"

#include <QPainter>

#include <QApplication>

#include <QDesktopWidget>

PopUp::**PopUp**(QWidget \*parent) : QWidget(*parent*)

{

setWindowFlags(Qt::*FramelessWindowHint* | *//* *Отключаем* *оформление* *окна*

Qt::*Tool* | *//* *Отменяем* *показ* *в* *качестве* *отдельного* *окна*

Qt::*WindowStaysOnTopHint*); *//* *Устанавливаем* *поверх* *всех* *окон*

setAttribute(Qt::*WA\_TranslucentBackground*); *//* *Указываем,* *что* *фон* *будет* *прозрачным*

setAttribute(Qt::*WA\_ShowWithoutActivating*); *//* *При* *показе,* *виджет* *не* *получается* *фокуса* *автоматически*

animation.setTargetObject(*this*); *//* *Устанавливаем* *целевой* *объект* *анимации*

animation.setPropertyName("popupOpacity"); *//* *Устанавливаем* *анимируемое* *свойство*

connect(&animation, &QAbstractAnimation::finished, *this*, &PopUp::hide); */\** *Подключаем* *сигнал* *окончания*

*\** *анимации* *к* *слоты* *скрытия*

*\** *\*/*

*//* *Настройка* *текста* *уведомления*

label.setAlignment(Qt::*AlignHCenter* | Qt::*AlignVCenter*); *//* *Устанавливаем* *по* *центру*

*//* *И* *настраиваем* *стили*

label.setStyleSheet("QLabel { color : white; "

"margin-top: 6px;"

"margin-bottom: 6px;"

"margin-left: 10px;"

"margin-right: 10px; }");

*//* *Производим* *установку* *текста* *в* *размещение,* *...*

layout.addWidget(*&label*, 0, 0);

setLayout(*&layout*); *//* *которое* *помещаем* *в* *виджет*

*//* *По* *сигналу* *таймера* *будет* *произведено* *скрытие* *уведомления,* *если* *оно* *видимо*

timer = *new* QTimer();

connect(timer, &QTimer::timeout, *this*, &PopUp::hideAnimation);

}

void PopUp::***paintEvent***(QPaintEvent \*event)

{

Q\_UNUSED(event)

*/\** *А* *теперь* *настраиваем* *фон* *уведомления,*

*\** *который* *является* *прямоугольником* *с* *чёрным* *фоном*

*\** *\*/*

QPainter painter(*this*);

painter.setRenderHint(QPainter::*Antialiasing*); *//* *Включаем* *сглаживание*

*//* *Подготавливаем* *фон.* *rect()* *возвращает* *внутреннюю* *геометрию* *виджета* *уведомления,* *по* *содержимому*

QRect roundedRect;

roundedRect.setX(rect().x() + 5);

roundedRect.setY(rect().y() + 5);

roundedRect.setWidth(rect().width() - 10);

roundedRect.setHeight(rect().height() - 10);

*//* *Кисть* *настраиваем* *на* *чёрный* *цвет* *в* *режиме* *полупрозрачности* *180* *из* *255*

painter.setBrush(QBrush(QColor(0,0,0,180)));

painter.setPen(Qt::*NoPen*); *//* *Край* *уведомления* *не* *будет* *выделен*

*//* *Отрисовываем* *фон* *с* *закруглением* *краёв* *в* *10px*

painter.drawRoundedRect(roundedRect, 10, 10);

}

void PopUp::**setPopupText**(*const* QString &text)

{

label.setText(text); *//* *Устанавливаем* *текст* *в* *Label*

adjustSize(); *//* *С* *пересчётом* *размеров* *уведомления*

}

void PopUp::**show**()

{

setWindowOpacity(0.0); *//* *Устанавливаем* *прозрачность* *в* *ноль*

animation.setDuration(150); *//* *Настраиваем* *длительность* *анимации*

animation.setStartValue(0.0); *//* *Стартовое* *значение* *будет* *0* *(полностью* *прозрачный* *виджет)*

animation.setEndValue(1.0); *//* *Конечное* *-* *полностью* *непрозрачный* *виджет*

setGeometry(QApplication::desktop()->availableGeometry().width() - 36 - width() + QApplication::desktop() -> availableGeometry().x(),

QApplication::desktop()->availableGeometry().height() - 36 - height() + QApplication::desktop() -> availableGeometry().y(),

width(),

height());

QWidget::show(); *//* *Отображаем* *виджет,* *который* *полностью* *прозрачен*

animation.start(); *//* *И* *запускаем* *анимацию*

timer->start(15000); *//* *А* *также* *стартуем* *таймер,* *который* *запустит* *скрытие* *уведомления* *через* *3* *секунды*

}

void PopUp::**hideAnimation**()

{

timer->stop(); *//* *Останавливаем* *таймер*

animation.setDuration(1000); *//* *Настраиваем* *длительность* *анимации*

animation.setStartValue(1.0); *//* *Стартовое* *значение* *будет* *1* *(полностью* *непрозрачный* *виджет)*

animation.setEndValue(0.0); *//* *Конечное* *-* *полностью* *прозрачный* *виджет*

animation.start(); *//* *И* *запускаем* *анимацию*

}

void PopUp::**hide**()

{

*//* *Если* *виджет* *прозрачный,* *то* *скрываем* *его*

*if*(getPopupOpacity() == 0.0){

QWidget::hide();

}

}

void PopUp::**setPopupOpacity**(float opacity)

{

popupOpacity = opacity;

setWindowOpacity(opacity);

}

float PopUp::**getPopupOpacity**() *const*

{

*return* popupOpacity;

}

1. **popup.h**

#ifndef POPUP\_H

#define POPUP\_H

#include <QWidget>

#include <QLabel>

#include <QGridLayout>

#include <QPropertyAnimation>

#include <QTimer>

*class* **PopUp** : *public* QWidget

{

Q\_OBJECT

*//* *Свойство* *полупрозрачности*

Q\_PROPERTY(float popupOpacity READ getPopupOpacity WRITE setPopupOpacity)

void **setPopupOpacity**(float opacity);

float **getPopupOpacity**() *const*;

*public*:

*explicit* **PopUp**(QWidget \*parent = 0);

*protected*:

void ***paintEvent***(QPaintEvent \*event); *//* *Фон* *будет* *отрисовываться* *через* *метод* *перерисовки*

*public* slots:

void **setPopupText**(*const* QString& text); *//* *Установка* *текста* *в* *уведомление*

void **show**(); */\** *Собственный* *метод* *показа* *виджета*

*\** *Необходимо* *для* *преварительной* *настройки* *анимации*

*\** *\*/*

*private* slots:

void **hideAnimation**(); *//* *Слот* *для* *запуска* *анимации* *скрытия*

void **hide**(); */\** *По* *окончании* *анимации,* *в* *данном* *слоте* *делается* *проверка,*

*\** *виден* *ли* *виджет,* *или* *его* *необходимо* *скрыть*

*\** *\*/*

*private*:

QLabel label; *//* *Label* *с* *сообщением*

QGridLayout layout; *//* *Размещение* *для* *лейбла*

QPropertyAnimation animation; *//* *Свойство* *анимации* *для* *всплывающего* *сообщения*

float popupOpacity; *//* *Свойства* *полупрозрачности* *виджета*

QTimer \*timer; *//* *Таймер,* *по* *которому* *виджет* *будет* *скрыт*

};

#endif *//* *POPUP\_H*

1. **mainwindow.h**

#include "popup.h"

QT\_BEGIN\_NAMESPACE

*namespace* **Ui** { *class* **MainWindow**; }

QT\_END\_NAMESPACE

*class* **MainWindow** : *public* QMainWindow

{

Q\_OBJECT

*public*:

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

~***MainWindow***();

void **accept**(int index);

void **loadingDoings**();

*private* slots:

void **on\_calendarWidget\_selectionChanged**();

void **on\_add\_doFortable\_clicked**();

void **on\_showCurrectDate\_clicked**();

void **on\_table\_toDo\_clicked**(*const* QModelIndex &index);

void **on\_del\_doFortable\_clicked**();

void **on\_save\_doFortable\_clicked**();

void **on\_accept\_clicked**();

void **NoticeToday**();

void **NoticeThreeDaysNext**();

void **NoticeOneDayNext**();

*private*:

Ui::MainWindow \*ui;

int currectDo = 0;

PopUp \*popUp;

QTimer \*timer, \*timerNextDay, \*timerThreeNextDay;

QDate DateNext, DateThreeNext;

bool flag = *false*;

QString noticeTextToday, noticeTextThreeDaysNext, noticeTextOneDayNext; *//* *Текст* *уведомлений*

QStringList save; *//* *Сохраняемый* *список*

QStringList currectList; *//* *Текущий* *список*

QStringList listData; *//* *Список* *дат*

QStringList listTask; *//* *Список* *задач*

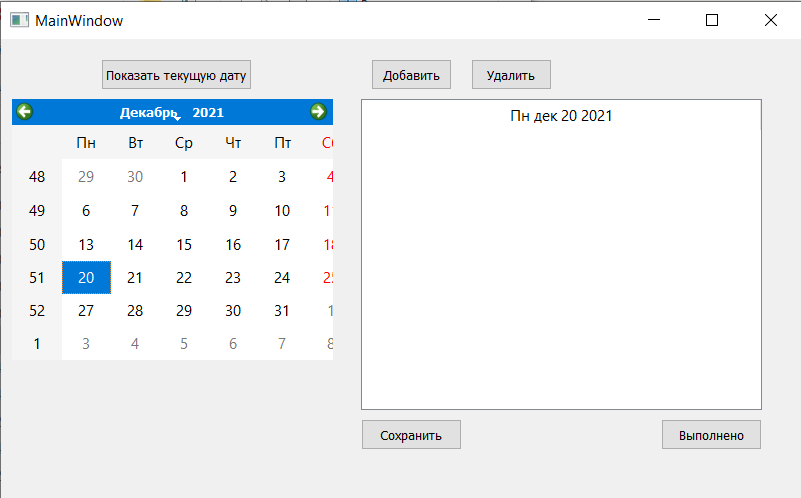
QStringList listAccept; *//* *Список* *выполненных* *задач*

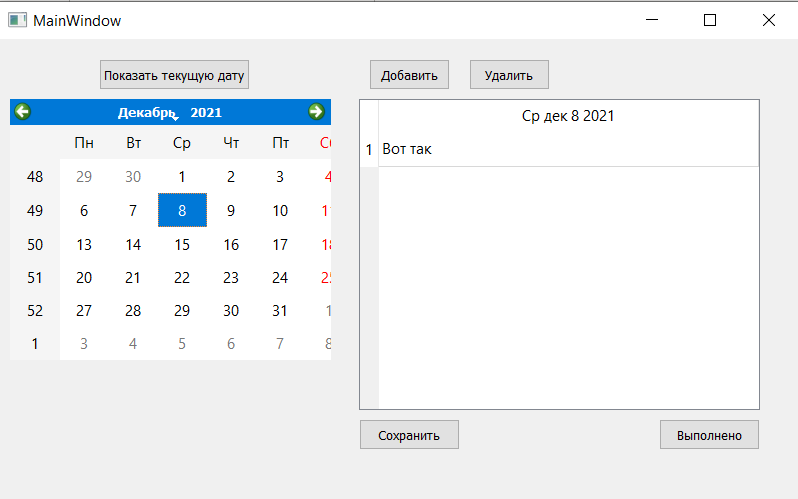
int row = -1;

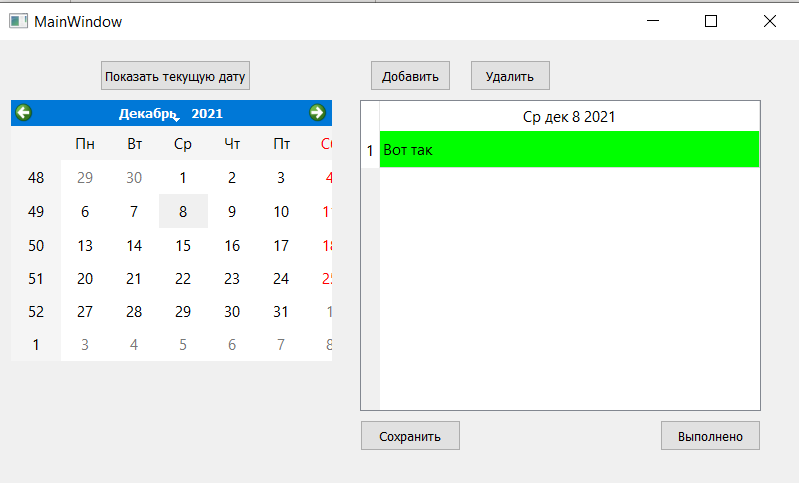
};

#endif *//* *MAINWINDOW\_H*

**Результаты тестирования программы:**







**Вывод:** приобрёл практические навыки проектирования и разработки приложений с графическим пользовательским интерфейсом в ОС Windows средствами Qt.