**Правительство Российской Федерации**

---

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

# Отчёт к лабораторной работе №1

## по дисциплине

## «Языки программирования»


Работу выполнил

Студент группы СКБ223 _____ А. Я. Сорочайкин

подпись, дата

Работу проверил _____ С. А. Булгаков

подпись, дата

# Содержание

## Постановка задачи

Разработать программу с использованием библиотеки Qt позволяющую ознакомиться с имеющимися элементами пользовательского интерфейса. Окно программы должно содержать два элемента: органайзер (слева) и панель с текстом (справа). На вкладках органайзера расположить виджеты, сгруппировав их в соответствии с Qt: Widgets and Layouts. При наведении курсора мышки на виджет на панели теста появляется его описание.

Для реализации органайзера использовать класс QTabWidget. Для реализации панели с текстом использовать класс QFrame (задав ему режим отрисовки рамки) и класс QLabel (задав ему режим переноса строк).

# 1. Алгоритм решения задачи

Для решения задачи, был разработан класс Widget, являющийся наследником класса QWidget, и содержащий в себе вкладки органайзера и панель с текстом. Также, были разработаны классы, наследники классов виджетов из списка, соответствующего QT 5.15 Widget Classes.

# 2. Решение задачи

Все классы, наследники классов виджетов из списка, соответствующего QT 5.15 Widget Classes, были разработаны по одному принципу. Ниже, приведен пример для класса *myQDateEdit,* разработанного на основе класса QDateEdit.

## 2.1. Класс myQDateEdit

Класс состоит из, конструктора, защищенного метода *mouseMoveEvent*, защищенного сигнала *info.*

### 2.1.1. Конструктор

Принимает на вход *QString s* и применяет метод *setText* с аргументом s к полю класса *frame_text.*

### 2.1.2. Метод mouseMoveEvent

С помощью ключевого слова *emit* испускает сигнал *info* и передает в него параметр типа QString, с описанием виджета *DateEdit.*

Класс Widget разработан на основе класса QWidget.

Панель с текстом разработана с использованием класса QFrame и QLabel.

## 2.2. Класс Widget

Класс состоит из полей класса *Qframel* frame, QLabel* frame_text,* конструктора, публичного слота *changeInfo.*

### 2.2.1. Слот changeInfo

Принимает на вход *QString s* и применяет метод *setText* с аргументом s к полю класса *frame_text*

### 2.2.2. Конструктор

Инициализирует все поля класса, создает все объекты классов виджетов, которые должны быть отображены во

вкладках органайзера. Производит коннект всех классов виджетов, которые должны быть отображены во вкладках органайзера, с слотом *changeInfo*, и добавляет каждый в соответствующий ему Layout. Производит настройку графической составляющей приложения.

## 3. Тестирование программы

Общий вид окна программы приведен на рисунке 1**.**

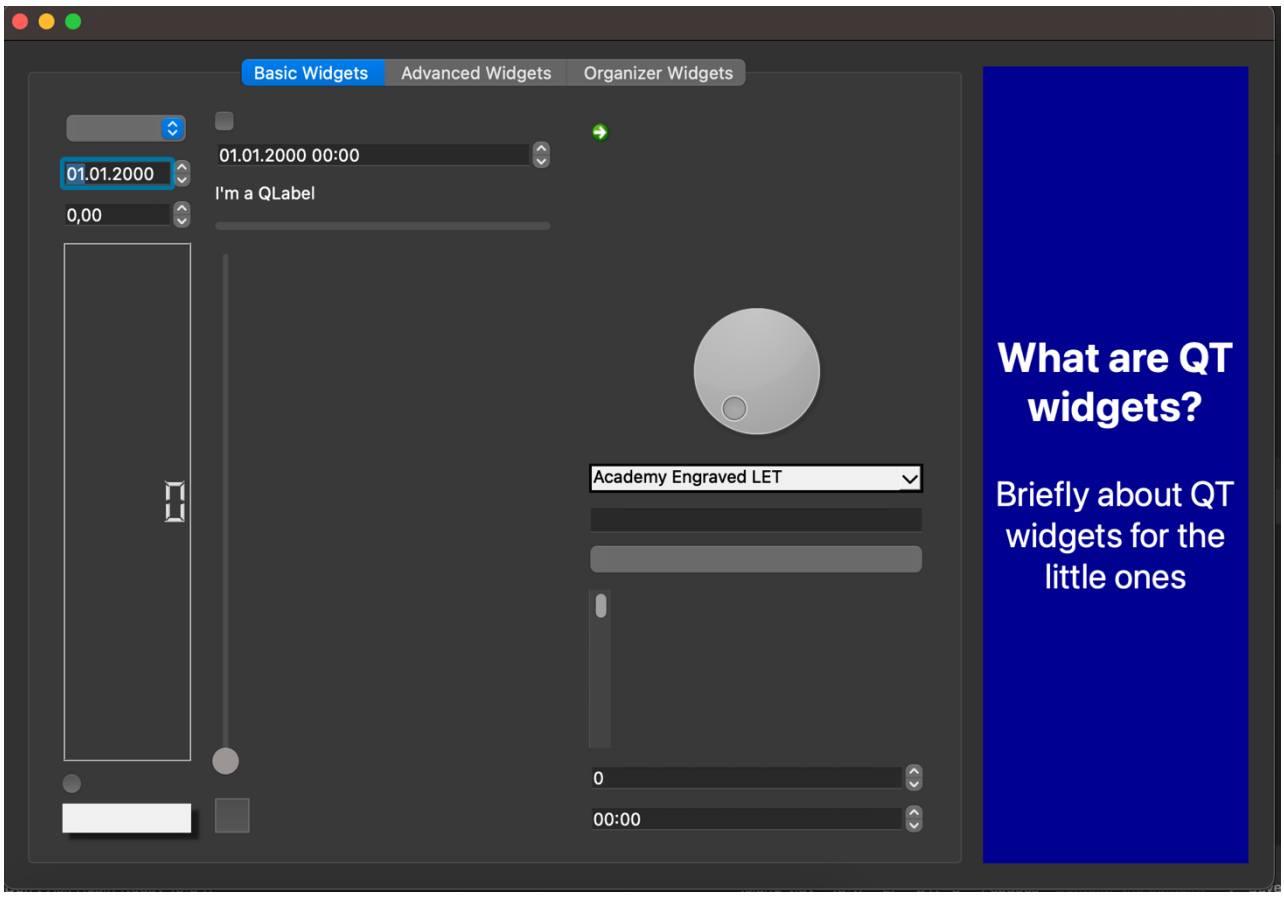


*Рис. 1*

На рисунке 2 приведен пример отображения информации о виджете при наведении на него курсором мыши.

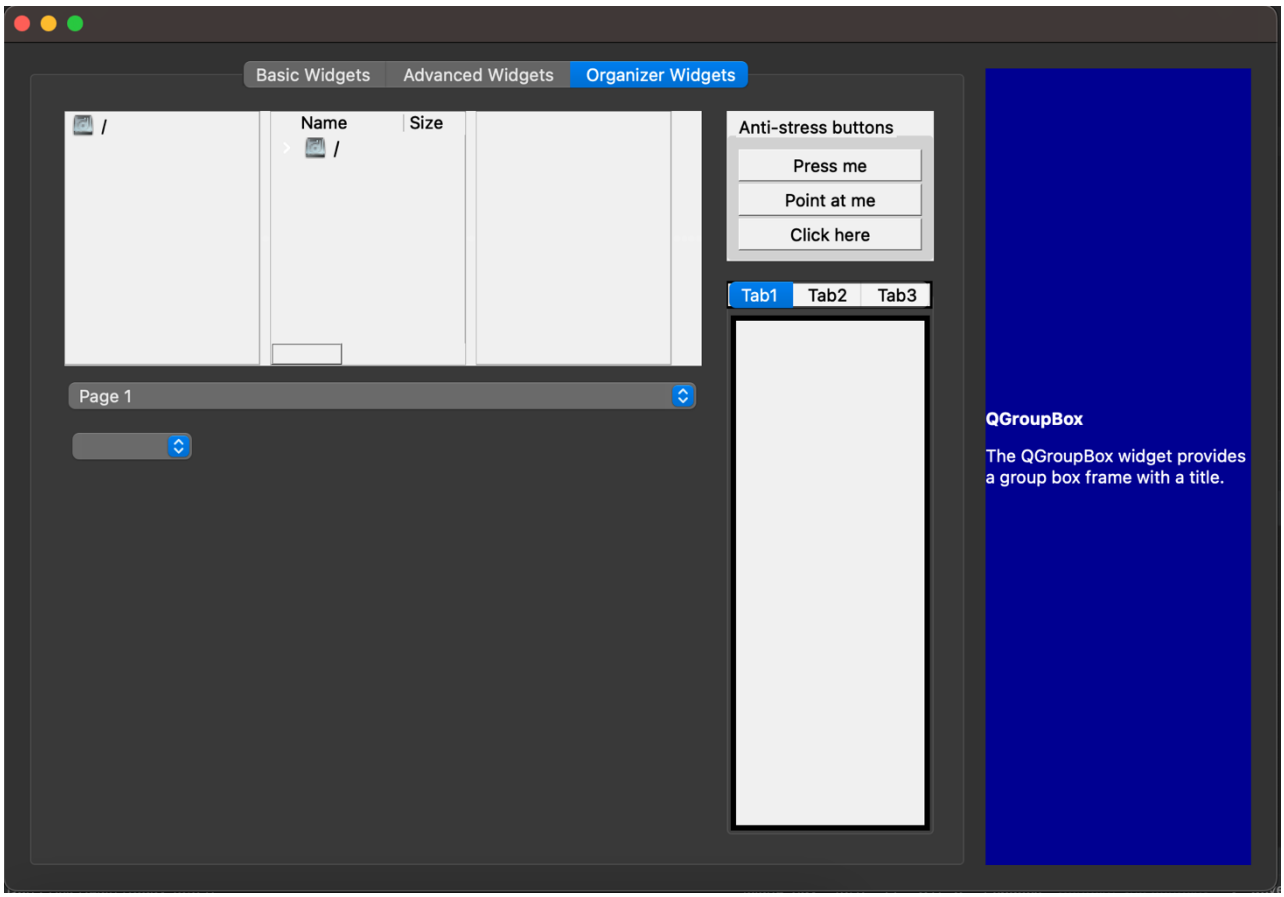


*Рис. 2*

На рисунках 3, 4 и 5 приведен общий вид вкладок Basic Widgets, Advanced Widgets и Organaizer Widgets.

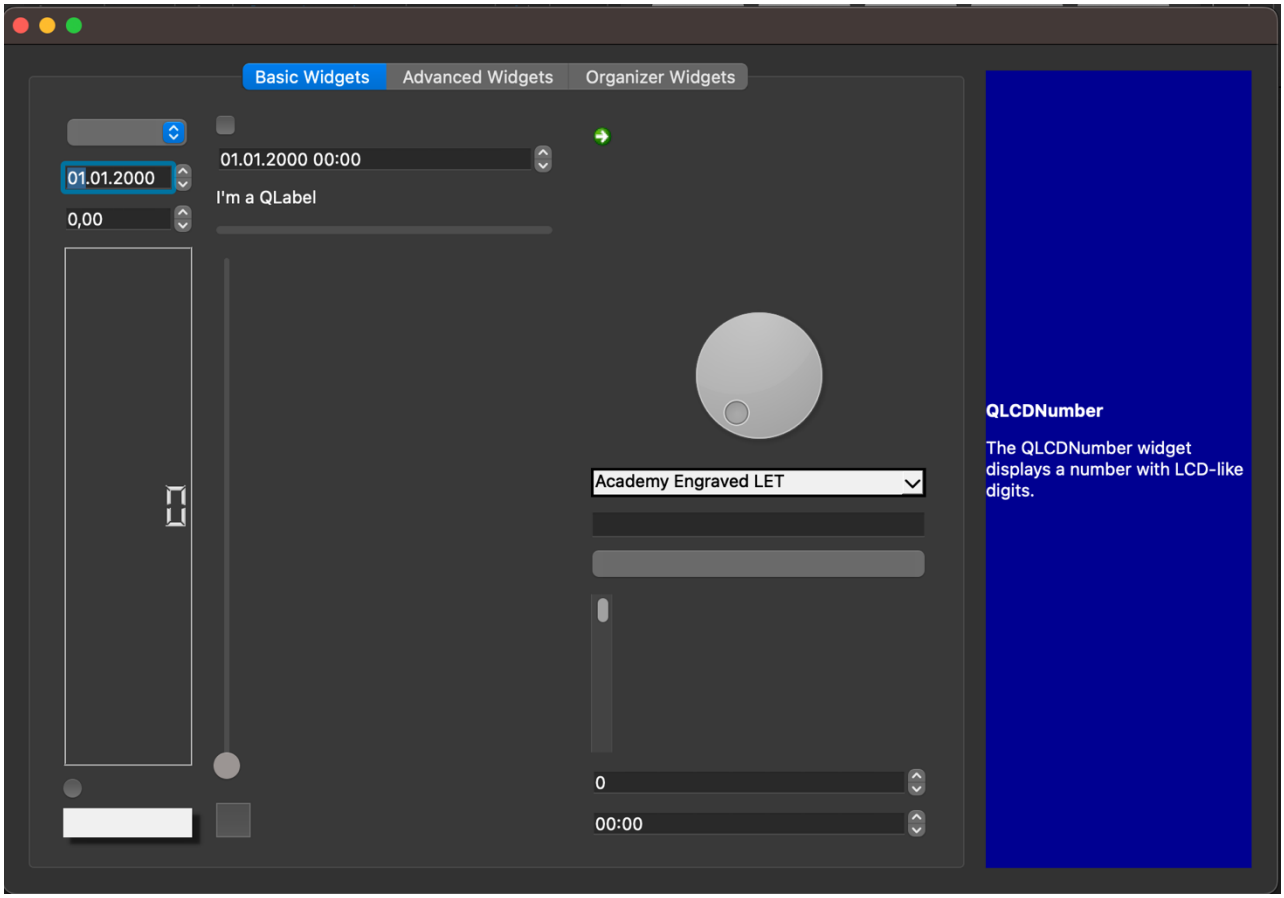
Basic Widgets
Advanced Widgets
Organizer Widgets
01.01.2000
0,00
01.01.2000 00:00
I'm a QLabel
Academy Engraved LET
0
00:00
QLCDNumber
The QLCDNumber widget displays a number with LCD-like digits.

*Рис. 3*

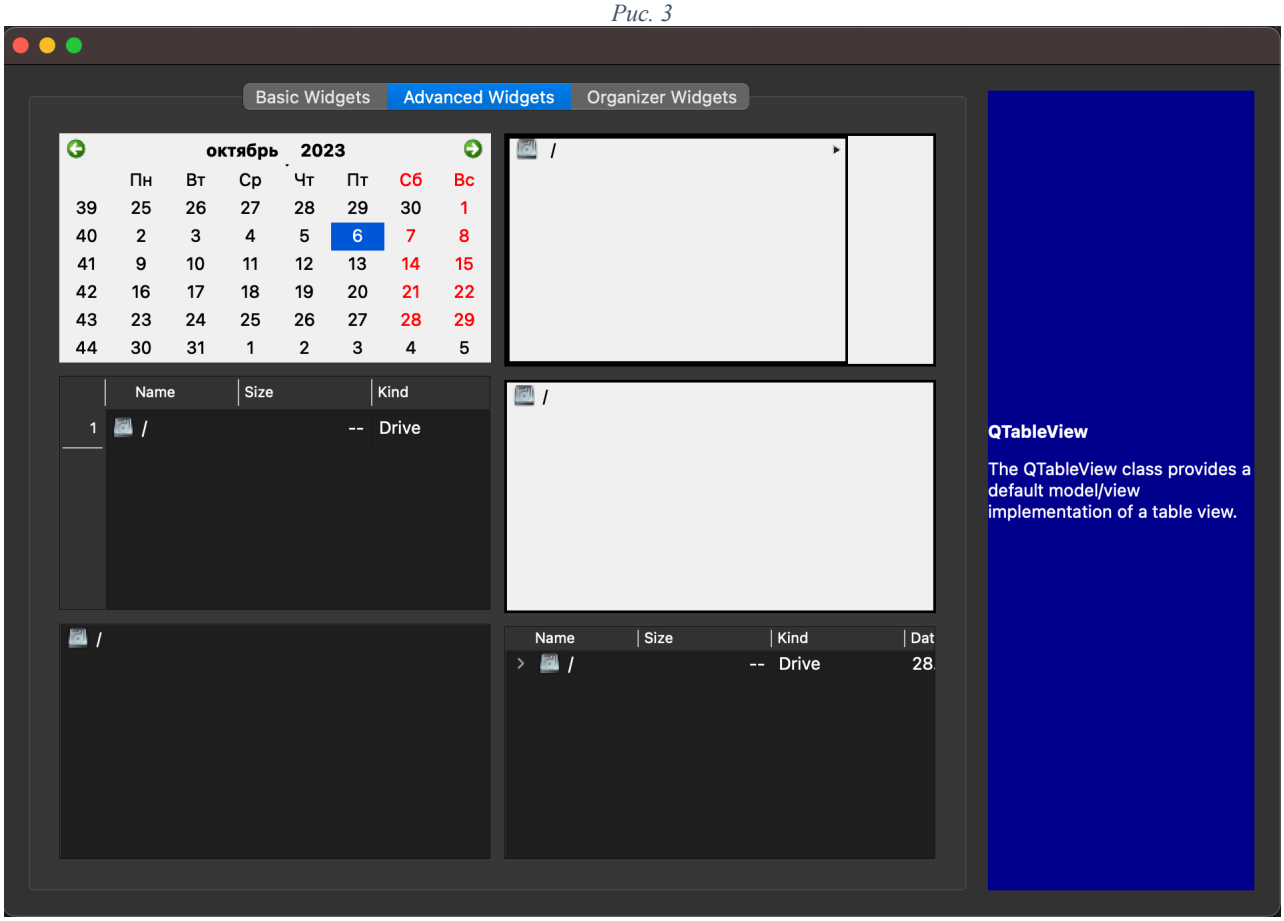
Basic Widgets
Advanced Widgets
Organizer Widgets
октябрь 2023
Пн Вт Ср Чт Пт Сб Вс
39 25 26 27 28 29 30 1
40 2 3 4 5 6 7 8
41 9 10 11 12 13 14 15
42 16 17 18 19 20 21 22
43 23 24 25 26 27 28 29
44 30 31 1 2 3 4 5
Name Size Kind
1 / -- Drive
/
/
Name Size Kind Dat
/ -- Drive 28
QTableView
The QTableView class provides a default model/view implementation of a table view.

*Рис. 4*

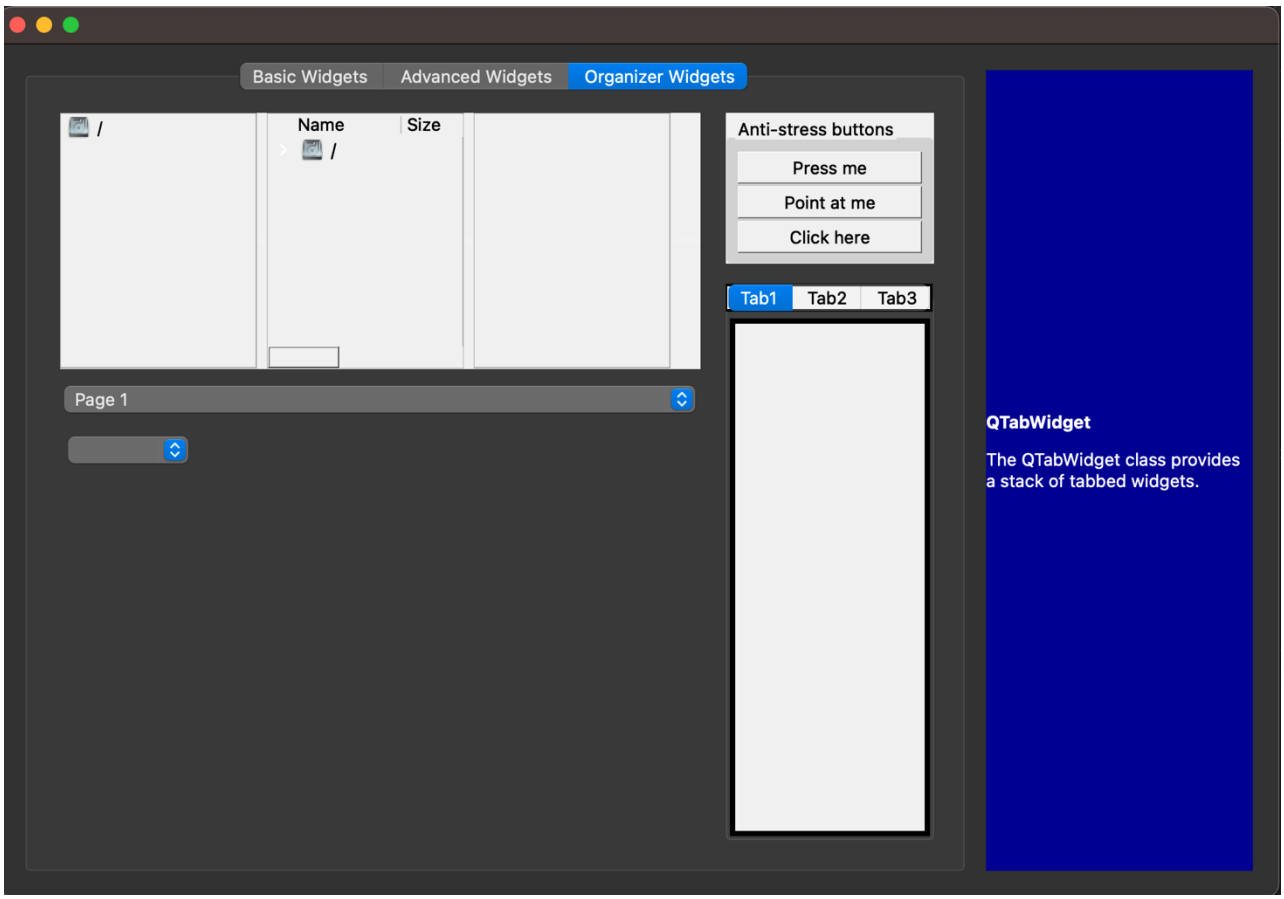
Basic Widgets
Advanced Widgets
Organizer Widgets
/
Name
Size
/
Page 1
Anti-stress buttons
Press me
Point at me
Click here
Tab1
Tab2
Tab3
QTabWidget
The QTabWidget class provides a stack of tabbed widgets.

*Рис. 5*

# Приложение А

## А.1 Исходный код main.cpp

```
#include "mainheader.h"

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    Widget w;
    w.show();
    return a.exec();
}
```

## А.2 Исходный код mainheader.h

```
#ifndef MAINHEADER_H
#define MAINHEADER_H

#include <QWidget>
#include <QComboBox>
#include <QMouseEvent>
#include <tab1.h>
#include <tab2.h>
#include <tab3.h>

class QFrame;
class QLabel;

class Widget : public QWidget
{
    Q_OBJECT
    QFrame *frame;
    QLabel *frame_text;
public:
    Widget();

public slots:
    void changeInfo(QString s);
};

#endif
```

## А.3 Исходный код func.cpp

```
#include "mainheader.h"

#include <QTabWidget>
#include <QHBoxLayout>
#include <QVBoxLayout>
#include <QLabel>
#include <QMouseEvent>
#include <QFileSystemModel>
#include <QTextEdit>

#include <QGraphicsDropShadowEffect>

Widget::Widget()
{
    QHBoxLayout *l = new QHBoxLayout(this);
    QTabWidget * tab = new QTabWidget(this);
    tab -> setMinimumSize(600, 600);
    l -> addWidget(tab);

    //Tab 1
```

```
    QWidget *tab1 = new QWidget(tab);
    tab->addTab(tab1, "&Basic Widgets");
    QHBoxLayout * lt1 = new QHBoxLayout(tab1);
    QVBoxLayout * lt2 = new QVBoxLayout();
    QVBoxLayout * lt3 = new QVBoxLayout();
    QVBoxLayout * lt4 = new QVBoxLayout();

    //QComboBox
    myComboBox * combo = new myComboBox(tab1);
    connect(combo, SIGNAL(info(QString)), this, SLOT(changeInfo(QString)));
    lt2 -> addWidget(combo);
    //QCheckBox
    myCheckBox * check = new myCheckBox(tab1);
    connect(check, SIGNAL(info(QString)), this, SLOT(changeInfo(QString)));
    lt3 -> addWidget(check);
    //QCommandLinkButton
    myQCommandLinkButton * commandLink = new myQCommandLinkButton(tab1);
    connect(commandLink, SIGNAL(info(QString)), this,
SLOT(changeInfo(QString)));
    lt4 -> addWidget(commandLink);
    //QDateEdit
    myQDateEdit * dateEdit = new myQDateEdit(tab1);
    connect(dateEdit, SIGNAL(info(QString)), this, SLOT(changeInfo(QString)));
    lt2 -> addWidget(dateEdit);
    //QTimeEdit
    myQDateTimeEdit * dateTimeEdit = new myQDateTimeEdit(tab1);
    connect(dateTimeEdit, SIGNAL(info(QString)), this,
SLOT(changeInfo(QString)));
    lt3 -> addWidget(dateTimeEdit);
    //QDial
    myQDial * qDial = new myQDial(tab1);
    connect(qDial, SIGNAL(info(QString)), this, SLOT(changeInfo(QString)));
    lt4 -> addWidget(qDial);
    //QDoubleSpinBox
    myQDoubleSpinBox * qDoubleSpinBox = new myQDoubleSpinBox(tab1);
    connect(qDoubleSpinBox, SIGNAL(info(QString)), this,
SLOT(changeInfo(QString)));
    lt2 -> addWidget(qDoubleSpinBox);
    //QFontComboBox
    myQFontComboBox * qFontComboBox = new myQFontComboBox(tab1);
    connect(qFontComboBox, SIGNAL(info(QString)), this,
SLOT(changeInfo(QString)));
    lt4 -> addWidget(qFontComboBox);
    //QLCDNumber
    myQLCDNumber * qLCDNumber = new myQLCDNumber(tab1);
    connect(qLCDNumber, SIGNAL(info(QString)), this,
SLOT(changeInfo(QString)));
    lt2 -> addWidget(qLCDNumber);
    //QLabel
    myQLabel * qLabel = new myQLabel(tab1);
    connect(qLabel, SIGNAL(info(QString)), this, SLOT(changeInfo(QString)));
    qLabel->setText("I'm a QLabel");
    lt3 -> addWidget(qLabel);
    //QLineEdit
    myQLineEdit * qLineEdit = new myQLineEdit(tab1);
    connect(qLineEdit, SIGNAL(info(QString)), this, SLOT(changeInfo(QString)));
    lt4 -> addWidget(qLineEdit);
    //QProgressBar
    myQProgressBar * qProgressBar = new myQProgressBar(tab1);
    connect(qProgressBar, SIGNAL(info(QString)), this,
SLOT(changeInfo(QString)));
    lt3 -> addWidget(qProgressBar);
    //QPushButton
    myQPushButton * qPushButton = new myQPushButton(tab1);
    connect(qPushButton, SIGNAL(info(QString)), this,
SLOT(changeInfo(QString)));
    lt4 -> addWidget(qPushButton);
    //QRadioButton
```

```
    myQRadioButton * qRadioButton = new myQRadioButton(tab1);
    connect(qRadioButton, SIGNAL(info(QString)), this,
SLOT(changeInfo(QString)));
    lt2 -> addWidget(qRadioButton);
    //QScrollBar
    myQScrollBar * qScrollBar = new myQScrollBar(tab1);
    connect(qScrollBar, SIGNAL(info(QString)), this,
SLOT(changeInfo(QString)));
    lt4 -> addWidget(qScrollBar);
    //QSizeGrip
    myQSizeGrip * qSizeGrip = new myQSizeGrip(tab1);
    connect(qSizeGrip, SIGNAL(info(QString)), this, SLOT(changeInfo(QString)));
    lt2 -> addWidget(qSizeGrip);
    //QSlider
    myQSlider * qSlider = new myQSlider(tab1);
    connect(qSlider, SIGNAL(info(QString)), this, SLOT(changeInfo(QString)));
    lt3 -> addWidget(qSlider);
    //QSpinBox
    myQSpinBox * qSpinBox = new myQSpinBox(tab1);
    connect(qSpinBox, SIGNAL(info(QString)), this, SLOT(changeInfo(QString)));
    lt4 -> addWidget(qSpinBox);
    //QTimeEdit
    myQTimeEdit * qTimeEdit = new myQTimeEdit(tab1);
    connect(qTimeEdit, SIGNAL(info(QString)), this, SLOT(changeInfo(QString)));
    lt4 -> addWidget(qTimeEdit);
    //QToolButton
    myQToolButton * qToolButton = new myQToolButton(tab1);
    connect(qToolButton, SIGNAL(info(QString)), this,
SLOT(changeInfo(QString)));
    lt3 -> addWidget(qToolButton);

    lt1->addLayout(lt2);
    lt1->addLayout(lt3);
    lt1->addLayout(lt4);
    //Tab2
    QWidget *tab2 = new QWidget(tab);
    tab->addTab(tab2, "&Advanced Widgets");
    QHBoxLayout * lt5 = new QHBoxLayout(tab2);
    QVBoxLayout * lt6 = new QVBoxLayout();
    QVBoxLayout * lt7 = new QVBoxLayout();

    QFileSystemModel *model = new QFileSystemModel;
    model->setRootPath(QDir::rootPath());

    //QCalendarWidget
    myQCalendarWidget * qCalendarWidget = new myQCalendarWidget(tab2);
    connect(qCalendarWidget, SIGNAL(info(QString)), this,
SLOT(changeInfo(QString)));
    lt6 -> addWidget(qCalendarWidget);

    //QColumnView
    myQColumnView * qColumnView = new myQColumnView(tab2);
    qColumnView->setModel(model);
    connect(qColumnView, SIGNAL(info(QString)), this,
SLOT(changeInfo(QString)));
    lt7 -> addWidget(qColumnView);

    //QListView
    myQListView * qListView = new myQListView(tab2);
    qListView->setModel(model);
    connect(qListView, SIGNAL(info(QString)), this, SLOT(changeInfo(QString)));
    lt7 -> addWidget(qListView);

    //QTableView
    myQTableView * qTableView = new myQTableView(tab2);
    qTableView->setModel(model);
    connect(qTableView, SIGNAL(info(QString)), this,
SLOT(changeInfo(QString)));
```

```
    lt6 -> addWidget(qTableView);

    //QTreeView
    myQTreeView * qTreeView = new myQTreeView(tab2);
    qTreeView->setModel(model);
    connect(qTreeView, SIGNAL(info(QString)), this, SLOT(changeInfo(QString)));
    lt7 -> addWidget(qTreeView);

    //QUndoView
    myQUndoView * qUndoView = new myQUndoView(tab2);
    qUndoView->setModel(model);
    connect(qUndoView, SIGNAL(info(QString)), this, SLOT(changeInfo(QString)));
    lt6 -> addWidget(qUndoView);

    lt5->addLayout(lt6);
    lt5->addLayout(lt7);
    //Tab3
    QWidget *tab3 = new QWidget(tab);
    tab->addTab(tab3, "&Organizer Widgets");
    QHBoxLayout * lt8 = new QHBoxLayout(tab3);
    QVBoxLayout * lt9 = new QVBoxLayout();
    QVBoxLayout * lt10 = new QVBoxLayout();

//    QGroupBox
    myQGroupBox * qGroupBox = new myQGroupBox(tab3);
    connect(qGroupBox, SIGNAL(info(QString)), this, SLOT(changeInfo(QString)));

    qGroupBox->setTitle("&Anti-stress buttons");
    QPushButton *p1 = new QPushButton(tr("&Press me"));
    QPushButton *p2 = new QPushButton(tr("&Point at me"));
    QPushButton *p3 = new QPushButton(tr("&Click here"));
    QVBoxLayout *groupLayout = new QVBoxLayout;
    groupLayout->addWidget(p1);
    groupLayout->addWidget(p2);
    groupLayout->addWidget(p3);
    qGroupBox->setLayout(groupLayout);

    lt10 -> addWidget(qGroupBox);
//    QSplitter
    myQSplitter * qSplitter = new myQSplitter(tab3);
    connect(qSplitter, SIGNAL(info(QString)), this, SLOT(changeInfo(QString)));

    QListView *listview = new QListView;
    QTreeView *treeview = new QTreeView;
    QTextEdit *textedit = new QTextEdit;
    listview->setModel(model);
    treeview->setModel(model);

    qSplitter->addWidget(listview);
    qSplitter->addWidget(treeview);
    qSplitter->addWidget(textedit);

    lt9 -> addWidget(qSplitter);

//    QSplitterHandle
    myQSplitterHandle *splitterHandleH = new
myQSplitterHandle(Qt::Orientation::Horizontal, qSplitter);
    myQSplitterHandle *splitterHandleV = new
myQSplitterHandle(Qt::Orientation::Vertical, qSplitter);

    connect(splitterHandleH, SIGNAL(info(QString)), this,
SLOT(changeInfo(QString)));
    connect(splitterHandleV, SIGNAL(info(QString)), this,
SLOT(changeInfo(QString)));

    //QStackedWidget
    myQStackedWidget * qStackedWidget = new myQStackedWidget(tab3);
    connect(qStackedWidget, SIGNAL(info(QString)), this,
```

```
SLOT(changeInfo(QString)));

    QWidget *firstPageWidget = new QWidget;
    QWidget *secondPageWidget = new QWidget;
    QWidget *thirdPageWidget = new QWidget;
    qStackedWidget->addWidget(firstPageWidget);
    qStackedWidget->addWidget(secondPageWidget);
    qStackedWidget->addWidget(thirdPageWidget);
    QComboBox *pageComboBox = new QComboBox;
    QComboBox *stackComboBox = new QComboBox(firstPageWidget);
    QTextEdit *stackTextEdit = new QTextEdit(secondPageWidget);
    QPushButton *stackPushButton = new QPushButton(thirdPageWidget);
    pageComboBox->addItem(tr("Page 1"));
    pageComboBox->addItem(tr("Page 2"));
    pageComboBox->addItem(tr("Page 3"));
    connect(pageComboBox, QOverload<int>::of(&QComboBox::activated),
            qStackedWidget, &QStackedWidget::setCurrentIndex);
    lt9 -> addWidget(pageComboBox);
    lt9 -> addWidget(qStackedWidget);

    //QTabWidget
    myQTabWidget * qTabWidget = new myQTabWidget(tab3);
    QWidget *firstPageWidget1 = new QWidget;
    QWidget *secondPageWidget2 = new QWidget;
    QWidget *thirdPageWidget3 = new QWidget;
    qTabWidget->addTab(firstPageWidget1, "Tab1");
    qTabWidget->addTab(secondPageWidget2, "Tab2");
    qTabWidget->addTab(thirdPageWidget3, "Tab3");
    connect(qTabWidget, SIGNAL(info(QString)), this,
SLOT(changeInfo(QString)));
    lt10 -> addWidget(qTabWidget);

    //Ending tab3
    lt8->addLayout(lt9);
    lt8->addLayout(lt10);

    frame = new QFrame(this);
    frame -> setFrameShape(QFrame::Box);
    frame -> setMinimumSize(200, 600);
    l -> addWidget(frame);

    frame_text = new QLabel(frame);
    frame_text->setMinimumSize(200, 600);
    frame_text->setWordWrap(true);
    frame_text->setStyleSheet("background-color:#00008B;");
    frame_text -> setText("<font size=\"20\"><b><center>What are QT
widgets?</font></b></center><br/>" "<font size=\"6\"><center>\n"
                          "Briefly about QT widgets for the little
ones</center>");


    //Some graphic adds

    QGraphicsDropShadowEffect *shadowEffect = new QGraphicsDropShadowEffect;
    shadowEffect->setBlurRadius(10);
    shadowEffect->setColor(QColor(0, 0, 0, 150));
    shadowEffect->setOffset(5, 5);

    qFontComboBox->setStyleSheet("background-color: #F0F0F0; color: #000000;
border: 2px solid #000000;");
    qFontComboBox->setGraphicsEffect(shadowEffect);

    qCalendarWidget->setStyleSheet("background-color: #F0F0F0; color:
#000000;");
    qCalendarWidget->setGraphicsEffect(shadowEffect);

    qGroupBox->setStyleSheet("background-color: #F0F0F0; color: #000000;");
    qGroupBox->setGraphicsEffect(shadowEffect);
```

```
    qListView->setStyleSheet("background-color: #F0F0F0; color: #000000;
border: 2px solid #000000;");
    qListView->setGraphicsEffect(shadowEffect);

    qColumnView->setStyleSheet("background-color: #F0F0F0; color: #000000;
border: 2px solid #000000;");
    qColumnView->setGraphicsEffect(shadowEffect);

    qTabWidget->setStyleSheet("background-color: #F0F0F0; color: #000000;
border: 2px solid #000000;");
    qTabWidget->setGraphicsEffect(shadowEffect);

    qSplitter->setStyleSheet("background-color: #F0F0F0; color: #000000;");
    qSplitter->setGraphicsEffect(shadowEffect);

    qSizeGrip->setStyleSheet("background-color: #F0F0F0; color: #000000;");
    qSizeGrip->setGraphicsEffect(shadowEffect);
}

void Widget::changeInfo(QString s) {
    frame_text -> setText(s);
}
```

# A.4 Исходный код tab1.h

```
#ifndef TAB1_H
#define TAB1_H

#include <QFocusFrame>
#include <QDoubleSpinBox>
#include <QDateTimeEdit>
#include <QDateEdit>
#include <QCheckBox>
#include <QComboBox>
#include <QTabWidget>
#include <QHBoxLayout>
#include <QVBoxLayout>
#include <QLabel>
#include <QMouseEvent>
#include <QCommandLinkButton>
#include <QDial>
#include <QFontComboBox>
#include <QLCDNumber>
#include <QLineEdit>
#include <QMenu>
#include <QProgressBar>
#include <QPushButton>
#include <QRadioButton>
#include <QScrollArea>
#include <QScrollBar>
#include <QSizeGrip>
#include <QSlider>
#include <QSpinBox>
#include <QTabBar>
#include <QTabWidget>
#include <QTimeEdit>
#include <QToolBox>
#include <QToolButton>

class myComboBox : public QComboBox
{
    Q_OBJECT
public:
    myComboBox(QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
```

```
    void info(QString);
};

class myCheckBox : public QCheckBox
{
    Q_OBJECT
public:
    myCheckBox(QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
    void info(QString);
};

class myQCommandLinkButton : public QCommandLinkButton
{
    Q_OBJECT
public:
    myQCommandLinkButton(QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
        void info(QString);
};

class myQDateEdit  : public QDateEdit
{
    Q_OBJECT
public:
    myQDateEdit (QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
};

class myQDateTimeEdit  : public QDateTimeEdit
{
    Q_OBJECT
public:
    myQDateTimeEdit (QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
};

class myQDial  : public QDial
{
    Q_OBJECT
public:
    myQDial (QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
};

class myQDoubleSpinBox  : public QDoubleSpinBox
{
    Q_OBJECT
public:
    myQDoubleSpinBox (QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
};
```

```
class myQFontComboBox  : public QFontComboBox
{
    Q_OBJECT
public:
    myQFontComboBox (QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
};

class myQLCDNumber  : public QLCDNumber
{
    Q_OBJECT
public:
    myQLCDNumber (QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
};

class myQLabel  : public QLabel
{
    Q_OBJECT
public:
    myQLabel (QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
};

class myQLineEdit  : public QLineEdit
{
    Q_OBJECT
public:
    myQLineEdit (QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
};

class myQProgressBar  : public QProgressBar
{
    Q_OBJECT
public:
    myQProgressBar (QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
};

class myQPushButton  : public QPushButton
{
    Q_OBJECT
public:
    myQPushButton (QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
};

class myQRadioButton  : public QRadioButton
```

```
{
    Q_OBJECT
public:
    myQRadioButton (QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
};

class myQScrollBar  : public QScrollBar
{
    Q_OBJECT
public:
    myQScrollBar (QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
};

class myQSizeGrip  : public QSizeGrip
{
    Q_OBJECT
public:
    myQSizeGrip (QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
};

class myQSlider  : public QSlider
{
    Q_OBJECT
public:
    myQSlider (QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
};

class myQSpinBox  : public QSpinBox
{
    Q_OBJECT
public:
    myQSpinBox (QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
};

class myQTimeEdit  : public QTimeEdit
{
    Q_OBJECT
public:
    myQTimeEdit (QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
};

class myQToolButton  : public QToolButton
{
    Q_OBJECT
```

```
public:
    myQToolButton (QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
};
#endif
```

# A.5 Исходный код tab1.cpp

```
#include <tab1.h>
#include <mainheader.h>

//QComboBox
myComboBox::myComboBox(QWidget * parent) : QComboBox(parent) {
    setMouseTracking(true);
}

void myComboBox::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QComboBox</b><p>The QComboBox widget is a combined button and
popup list<p>");
    QComboBox::mouseMoveEvent(e);
}

//QCheckBox
myCheckBox::myCheckBox(QWidget * parent) : QCheckBox(parent) {
    setMouseTracking(true);
}

void myCheckBox::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QCheckBox</b><p>The QCheckBox widget provides a checkbox with
a text label.<p>");
    QCheckBox::mouseMoveEvent(e);
}

//QCommandLinkButton
myQCommandLinkButton::myQCommandLinkButton(QWidget * parent) :
QCommandLinkButton(parent) {
    setMouseTracking(true);
}

void myQCommandLinkButton::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QCommandLinkButton</b><p>The QCommandLinkButton widget
provides a Vista style command link button..<p>");
    QCommandLinkButton::mouseMoveEvent(e);
}

//QDateEdit
myQDateEdit::myQDateEdit(QWidget * parent) : QDateEdit(parent) {
    setMouseTracking(true);
}

void myQDateEdit::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QDateEdit</b><p>The QDateEdit class provides a widget for
editing dates based on the QDateTimeEdit widget.<p>");
    QDateEdit::mouseMoveEvent(e);
}

//QDateTimeEdit
myQDateTimeEdit::myQDateTimeEdit(QWidget * parent) : QDateTimeEdit(parent) {
    setMouseTracking(true);
}

void myQDateTimeEdit::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QDateTimeEdit</b><p>The QDateTimeEdit class provides a widget
for editing dates and times.<p>");
```

```
    QDateTimeEdit::mouseMoveEvent(e);
}

//QDial
myQDial::myQDial(QWidget * parent) : QDial(parent) {
    setMouseTracking(true);
}

void myQDial::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QDial</b><p>The QDial class provides a rounded range control
(like a speedometer or potentiometer)<p>");
    QDial::mouseMoveEvent(e);
}

//QDoubleSpinBox
myQDoubleSpinBox::myQDoubleSpinBox(QWidget * parent) : QDoubleSpinBox(parent) {
    setMouseTracking(true);
}

void myQDoubleSpinBox::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QDoubleSpinBox</b><p>The QDoubleSpinBox class provides a spin
box widget that takes doubles.<p>");
    QDoubleSpinBox::mouseMoveEvent(e);
}

//QFontComboBox
myQFontComboBox::myQFontComboBox(QWidget * parent) : QFontComboBox(parent) {
    setMouseTracking(true);
}

void myQFontComboBox::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QFontComboBox</b><p>The QFontComboBox widget is a combobox
that lets the user select a font family.<p>");
    QFontComboBox::mouseMoveEvent(e);
}

//QLCDNumber
myQLCDNumber::myQLCDNumber(QWidget * parent) : QLCDNumber(parent) {
    setMouseTracking(true);
}

void myQLCDNumber::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QLCDNumber</b><p>The QLCDNumber widget displays a number with
LCD-like digits.<p>");
    QLCDNumber::mouseMoveEvent(e);
}

//QLabel
myQLabel::myQLabel(QWidget * parent) : QLabel(parent) {
    setMouseTracking(true);
}

void myQLabel::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QLabel</b><p>The QLabel widget provides a text or image
display.<p>");
    QLabel::mouseMoveEvent(e);
}

//QLineEdit
myQLineEdit::myQLineEdit(QWidget * parent) : QLineEdit(parent) {
    setMouseTracking(true);
}

void myQLineEdit::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QLineEdit</b><p>The QLineEdit widget is a one-line text
editor.<p>");
    QLineEdit::mouseMoveEvent(e);
}
```

```
//QProgressBar
myQProgressBar::myQProgressBar(QWidget * parent) : QProgressBar(parent) {
    setMouseTracking(true);
}

void myQProgressBar::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QProgressBar</b><p>The QProgressBar widget provides a
horizontal or vertical progress bar.<p>");
    QProgressBar::mouseMoveEvent(e);
}

//QPushButton
myQPushButton::myQPushButton(QWidget * parent) : QPushButton(parent) {
    setMouseTracking(true);
}

void myQPushButton::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QPushButton</b><p>The QPushButton widget provides a command
button.<p>");
    QPushButton::mouseMoveEvent(e);
}

//QRadioButton
myQRadioButton::myQRadioButton(QWidget * parent) : QRadioButton(parent) {
    setMouseTracking(true);
}

void myQRadioButton::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QRadioButton</b><p>The QRadioButton widget provides a radio
button with a text label.<p>");
    QRadioButton::mouseMoveEvent(e);
}

//QScrollBar
myQScrollBar::myQScrollBar(QWidget * parent) : QScrollBar(parent) {
    setMouseTracking(true);
}

void myQScrollBar::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QScrollBar</b><p>The QScrollBar widget provides a vertical or
horizontal scroll bar.<p>");
    QScrollBar::mouseMoveEvent(e);
}

//QSizeGrip
myQSizeGrip::myQSizeGrip(QWidget * parent) : QSizeGrip(parent) {
    setMouseTracking(true);
}

void myQSizeGrip::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QSizeGrip</b><p>The QSizeGrip class provides a resize handle
for resizing top-level windows.<p>");
    QSizeGrip::mouseMoveEvent(e);
}

//QSlider
myQSlider::myQSlider(QWidget * parent) : QSlider(parent) {
    setMouseTracking(true);
}

void myQSlider::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QSlider</b><p>The QSlider widget provides a vertical or
horizontal slider.<p>");
    QSlider::mouseMoveEvent(e);
}

//QSpinBox
```

```
myQSpinBox::myQSpinBox(QWidget * parent) : QSpinBox(parent) {
    setMouseTracking(true);
}

void myQSpinBox::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QSpinBox</b><p>The QSpinBox class provides a spin box
widget.<p>");
    QSpinBox::mouseMoveEvent(e);
}

//QTimeEdit
myQTimeEdit::myQTimeEdit(QWidget * parent) : QTimeEdit(parent) {
    setMouseTracking(true);
}

void myQTimeEdit::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QTimeEdit</b><p>The QTimeEdit class provides a widget for
editing times based on the QDateTimeEdit widget. <p>");
    QTimeEdit::mouseMoveEvent(e);
}

//QToolButton
myQToolButton::myQToolButton(QWidget * parent) : QToolButton(parent) {
    setMouseTracking(true);
}

void myQToolButton::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QToolButton</b><p>The QToolButton class provides a quick-
access button to commands or options, usually used inside a QToolBar.<p>");
    QToolButton::mouseMoveEvent(e);
}
```

## A.6 Исходный код tab2.h

```
#ifndef TAB2_H
#define TAB2_H

#include <QCalendarWidget>
#include <QColumnView>
#include <QDataWidgetMapper>
#include <QListView>
#include <QTableView>
#include <QTreeView>
#include <QUndoView>

class myQCalendarWidget  : public QCalendarWidget
{
    Q_OBJECT
public:
    myQCalendarWidget (QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
};

class myQColumnView  : public QColumnView
{
    Q_OBJECT
public:
    myQColumnView (QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
```

```
};

class myQListView  : public QListView
{
    Q_OBJECT
public:
    myQListView (QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
};

class myQTableView  : public QTableView
{
    Q_OBJECT
public:
    myQTableView (QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
};

class myQTreeView  : public QTreeView
{
    Q_OBJECT
public:
    myQTreeView (QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
};

class myQUndoView  : public QUndoView
{
    Q_OBJECT
public:
    myQUndoView (QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
};
#endif
```

# A.7 Исходный код tab2.cpp

```
#include <tab2.h>
#include <mainheader.h>

//QCalendarWidget
myQCalendarWidget::myQCalendarWidget(QWidget * parent) :
QCalendarWidget(parent) {
    setMouseTracking(true);
}

void myQCalendarWidget::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QCalendarWidget</b><p>The QCalendarWidget class provides a
monthly based calendar widget allowing the user to select a date.<p>");
    QCalendarWidget::mouseMoveEvent(e);
}

//QColumnView
```

```
myQColumnView::myQColumnView(QWidget * parent) : QColumnView(parent) {
    setMouseTracking(true);
}

void myQColumnView::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QColumnView</b><p>The QColumnView class provides a model/view
implementation of a column view. <p>");
    QColumnView::mouseMoveEvent(e);
}

//QListView
myQListView::myQListView(QWidget * parent) : QListView(parent) {
    setMouseTracking(true);
}

void myQListView::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QListView</b><p>The QListView class provides a list or icon
view onto a model. <p>");
    QListView::mouseMoveEvent(e);
}

//QTableView
myQTableView::myQTableView(QWidget * parent) : QTableView(parent) {
    setMouseTracking(true);
}

void myQTableView::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QTableView</b><p>The QTableView class provides a default
model/view implementation of a table view.<p>");
    QTableView::mouseMoveEvent(e);
}

//QTreeView
myQTreeView::myQTreeView(QWidget * parent) : QTreeView(parent) {
    setMouseTracking(true);
}

void myQTreeView::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QTreeView</b><p>The QTreeView class provides a default
model/view implementation of a tree view.<p>");
    QTreeView::mouseMoveEvent(e);
}

//QUndoView
myQUndoView::myQUndoView(QWidget * parent) : QUndoView(parent) {
    setMouseTracking(true);
}

void myQUndoView::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QUndoView</b><p>The QUndoView class displays the contents of
a QUndoStack.<p>");
    QUndoView::mouseMoveEvent(e);
}
```

# A.8 Исходный код tab3.h

```
#ifndef TAB3_H
#define TAB3_H

#include <QButtonGroup>
#include <QTabWidget>
#include <QStackedWidget>
#include <QSplitterHandle>
#include <QSplitter>
#include <QGroupBox>
```

```
class myQGroupBox  : public QGroupBox
{
    Q_OBJECT
public:
    myQGroupBox (QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
};

class myQSplitter  : public QSplitter
{
    Q_OBJECT
public:
    myQSplitter (QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
};

class myQSplitterHandle  : public QSplitterHandle
{
    Q_OBJECT
public:
    myQSplitterHandle (Qt::Orientation, myQSplitter *);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
};

class myQStackedWidget  : public QStackedWidget
{
    Q_OBJECT
public:
    myQStackedWidget (QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
};

class myQTabWidget  : public QTabWidget
{
    Q_OBJECT
public:
    myQTabWidget (QWidget *parent = 0);
protected:
    void mouseMoveEvent(QMouseEvent *e);
    signals:
            void info(QString);
};

//organizer widget Mtabwidget
#endif
```

## A.9 Исходный код tab3.cpp

```
#include <tab3.h>
#include <mainheader.h>

//QGroupBox
myQGroupBox::myQGroupBox(QWidget * parent) : QGroupBox(parent) {
```

```
    setMouseTracking(true);
}

void myQGroupBox::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QGroupBox</b><p>The QGroupBox widget provides a group box
frame with a title. <p>");
    QGroupBox::mouseMoveEvent(e);
}

//QSplitter
myQSplitter::myQSplitter(QWidget * parent) : QSplitter(parent) {
    setMouseTracking(true);
}

void myQSplitter::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QSplitter</b><p>The QSplitter class implements a splitter
widget.<p>");
    QSplitter::mouseMoveEvent(e);
}

//QSplitterHandle
myQSplitterHandle::myQSplitterHandle (Qt::Orientation o, myQSplitter * s) :
QSplitterHandle(o, s) {
    setMouseTracking(true);
}

void myQSplitterHandle::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QSplitterHandle</b><p>The QSplitterHandle class provides
handle functionality for the splitter. <p>");
    QSplitterHandle::mouseMoveEvent(e);
}

//QStackedWidget
myQStackedWidget::myQStackedWidget(QWidget * parent) : QStackedWidget(parent) {
    setMouseTracking(true);
}

void myQStackedWidget::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QStackedWidget</b><p>The QStackedWidget class provides a
stack of widgets where only one widget is visible at a time. <p>");
    QStackedWidget::mouseMoveEvent(e);
}

//QTabWidget
myQTabWidget::myQTabWidget(QWidget * parent) : QTabWidget(parent) {
    setMouseTracking(true);
}

void myQTabWidget::mouseMoveEvent(QMouseEvent *e) {
    emit info("<b>QTabWidget</b><p>The QTabWidget class provides a stack of
tabbed widgets.<p>");
    QTabWidget::mouseMoveEvent(e);
}
```

# Приложение Б

## Б.1 UML диаграмма

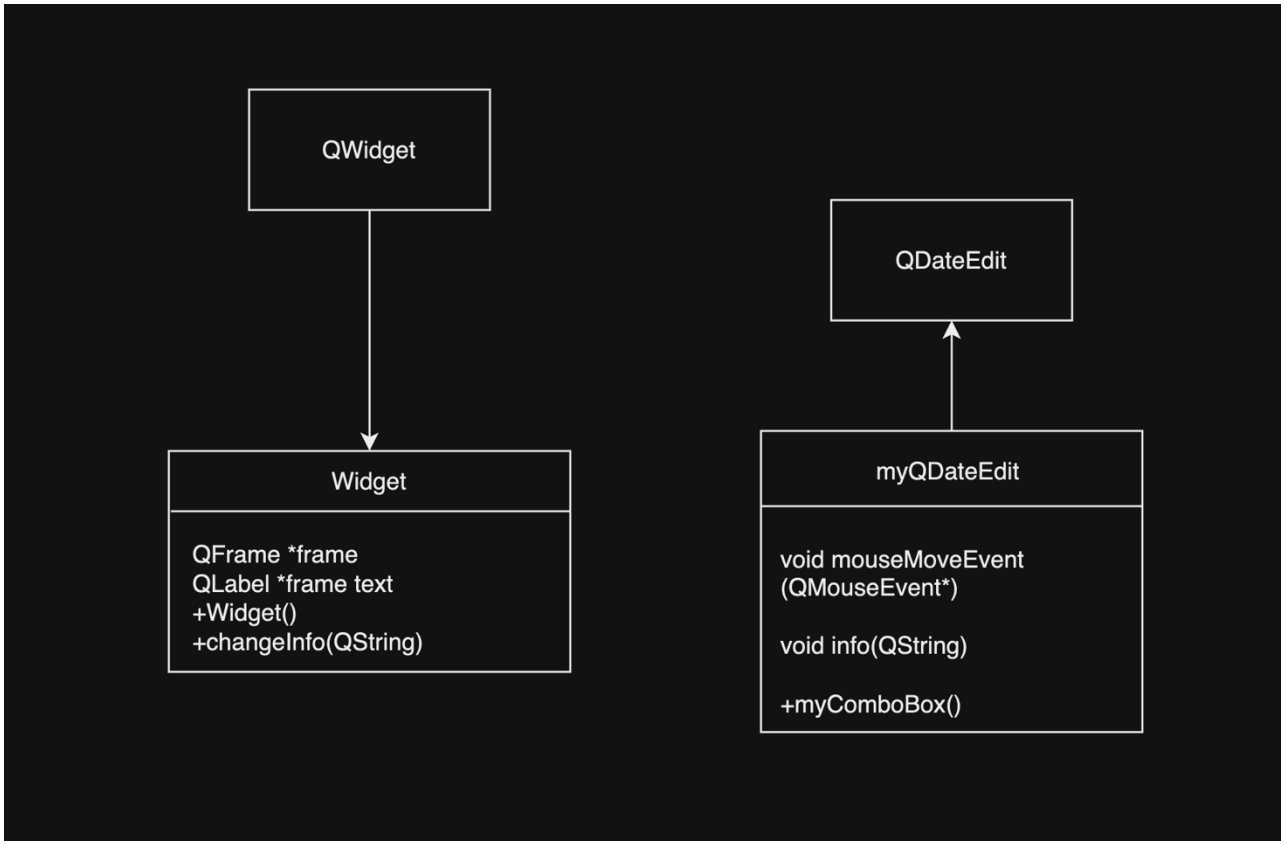
QWidget
QDateEdit
Widget
QFrame *frame
QLabel *frame text
+Widget()
+changeInfo(QString)
myQDateEdit
void mouseMoveEvent
(QMouseEvent*)
void info(QString)
+myComboBox()

*Рис. 6*