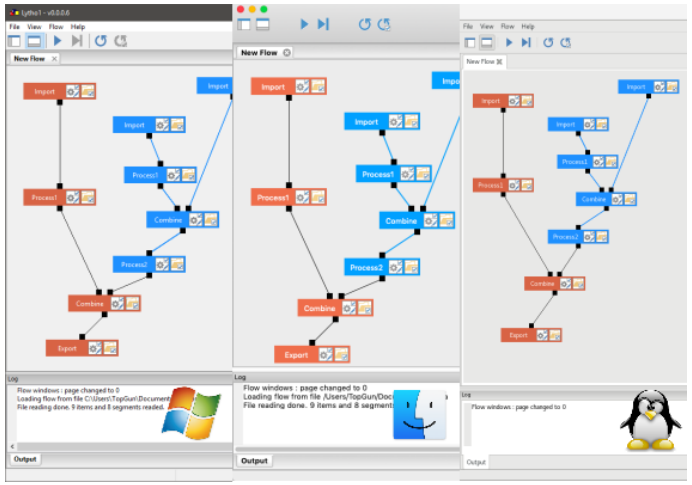
МИЭТ
Национальный исследовательский университет «МИЭТ»
Институт интегральной электроники (группы ЭН-34-35, каф. ПКИМС)

Кроссплатформенная разработка программного обеспечения



Лабораторная работа №4

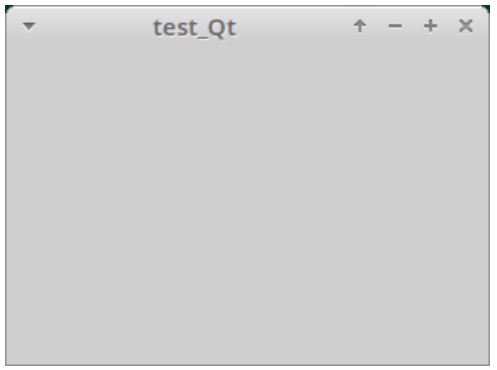
Знакомство с фреймворком Qt5

Фреймворк Qt

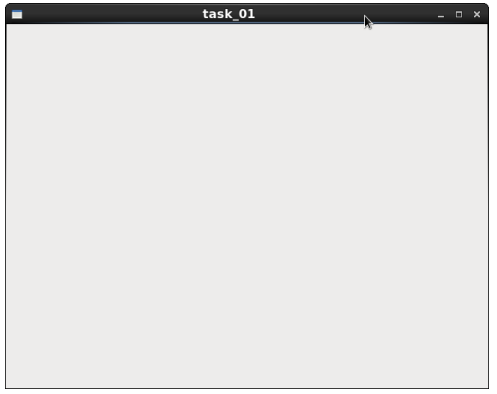
```
#include <QApplication>
#include <QWidget>

int main(int argc, char *argv[]) {
  QApplication app(argc, argv);

  QWidget wgt;
  wgt.show();


  return app.exec();
}
```

Windows 10

OpenSUSE Linux



CentOS Linux



Первое приложение на Qt


```
#include <QApplication>
#include <QWidget>

int main(int argc, char *argv[]) {
  QApplication app(argc, argv);

  QWidget wgt;
  wgt.show();


  return app.exec();
}
```

Компиляция приложения на Qt в консоли

Вариант 1

```
g++
-I/opt/Qt5.5.1/5.5/gcc_64/include
-I/opt/Qt5.5.1/5.5/gcc_64/include/QtGui
-I/opt/Qt5.5.1/5.5/gcc_64/include/QtCore
-I/opt/Qt5.5.1/5.5/gcc_64/include/QtWidgets
-I/opt/Qt5.5.1/5.5/gcc_64/mkspecs/linux-g++
-L/opt/Qt5.5.1/5.5/gcc_64/lib
-lQt5Gui
-lQt5Core
-lQt5Widgets
-fPIC
-o ./main
./main.cpp
```

Вариант 2

```
qmake -project
qmake
make
```

Основные модули Qt

- `QtCore` – ядро фреймворка.
- `QtGUI` – компоненты для создания интерфейсов.
- `QtWidgets` – модуль для работы с виджетами.
- `QtOpenGL` – инструменты для работы с библиотеками, написанными по спецификации OpenGL.
- `QtNetwork` – функции для работы с сетевыми соединениями.

- `QtSql` – компоненты для работы с базами данных на основе SQL.
- `QtXml` – компоненты для обработки XML, специального формата хранения файлов.
- `QtXmlPatterns` – инструменты для работы с языками, которые обрабатывают данные XML и организуют к нему доступ.
- `QtScript` – классы внутреннего скриптового языка Qt Scripts.
- `QtSvg` – компоненты для обработки векторной графики.
- `QtMultimedia` – инструменты для работы с мультимедиа-файлами.
- `QtWebEngine` – ядро браузера Chromium, адаптированное под Qt.
- `QtTest` – компоненты для тестирования приложений.
- `Qt3Support` – поддержка старых версий фреймворка.
- `QtCLucene` – инструменты для автоматического поиска.

Базовый класс: QObject

Класс QObject – базовый класс для большинства классов в Qt(QWidget, QApplication, QThread и др).

Конструктор:

```
QObject (Qobject *parent = Q_NULLPTR)
```

Важные методы :

```
public:
  QMetaObject::Connection connect() //Соединить сигнал со слотом
  bool          disconnect()        //Разъединить сигнал и слот
  virtual bool  event()             //Метод-обработчик событий
  virtual bool  eventFilter()       //Метод-обработчик событий другого объекта
  void          installEventFilter() //Передача управления событиями др. объекту
  bool          isWidgetType()      //Является ли объект виджетом
  bool          isWindowType()      //Является ли объект окном
  void          moveToThread()      //Перенести объект в отдельный поток
  QObject*      parent()            //Получить родителя
  void          removeEventFilter()  //Отменить управление событиями др. объекта
  void          setParent           //Установить родителя
  int           startTimer()        //Запустить таймер
  Qthread*      thread()            //Получить рабочий поток объекта
protected:
  int           recievers()         //Количество получателей сигнала
  QObject*      sender()            //Отправитель сигнала
  virtual void  timerEvent()        //Метод-обработчик событий по таймеру
```

Базовый класс: QWidget

Класс QWidget – базовый класс для примитивов графического интерфейса.

Конструктор:

```
QWidget(QWidget* parent = Q_NULLPTR, Qt::WindowFlags flags = ...)
```

Важные методы :

```
public:
    void resize()               //Изменить размер
    void setLayout()            //Установить слой
    void setMouseTracking()     //Включить отслеживание мыши
    void setStatusTip()         //Установить подсказку статусной строки
    void setToolTip()           //Установить всплывающую подсказку
    void setWindowFlags()       //Установить флаги окна
    bool underMouse()           //Расположен ли объект под курсором мыши
    void update()               //Обновить окно
protected:
    virtual void closeEvent()           //Окно закрылось
    virtual void enterEvent()           //Курсор оказался в области виджета
    virtual void keyPressEvent()        //Клавиша клавиатуры нажата
    virtual void keyReleaseEvent()      //Клавиша клавиатуры отжата
    virtual void leaveEvent()           //Курсор вышел из области виджета
    virtual void mouseDoubleClickEvent()//Двойной щелчок мыши
    virtual void mouseMoveEvent()       //Мышь изменила положение
    virtual void mousePressEvent()      //Клавиша мыши нажата
    virtual void mouseReleaseEvent()    //Клавиша мыши отжата
    virtual void moveEvent()            //Окно передвинулось
    virtual void paintEvent()           //Окно необходимо перерисовать
    virtual void resizeEvent()          //Окно изменило размер
    virtual void showEvent()            //Окно открылось
    virtual void wheelEvent()           //Колесо мыши было прокручено
```

Типы отображения окон

```
#include <QApplication>
#include <QWidget>
#include <Qt>

int main(int argc, char *argv[]) {
  QApplication app(argc, argv);

  QWidget wgt;
  wgt.resize(300, 100);
  wgt.setWindowTitle("New title");

  wgt.setWindowFlags(     );

  wgt.show();

  return app.exec();
}
```

Qt::Window

Qt::Dialog

Qt::Tool

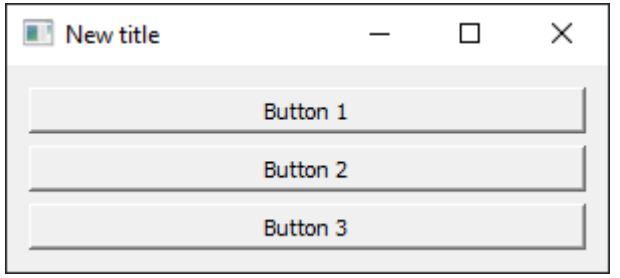
Qt::Popup

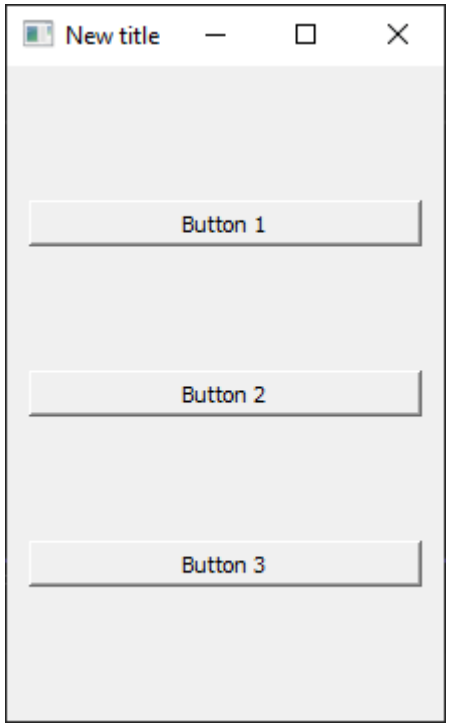
Добавление виджетов: QPushButton

```
#include <QApplication>
#include <QWidget>
#include <QPushButton>

int main(int argc, char *argv[]) {
  QApplication app(argc, argv);

  QWidget wgt;
  wgt.resize(300, 100);
  wgt.setWindowTitle("New title");
  wgt.show();

  QPushButton btn("Push me!", &wgt);
  wgt.show();

  return app.exec();
}
```

Менеджеры компоновки (Layout managers) (1)

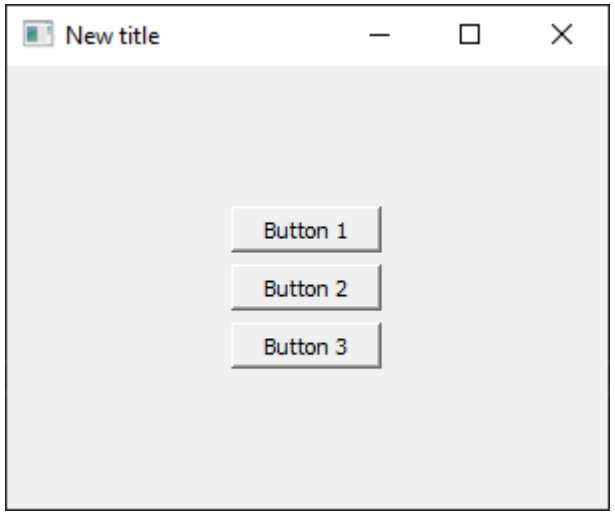
```
QPushButton btn1("Button 1", &wgt);
QPushButton btn2("Button 2", &wgt);
QPushButton btn3("Button 3", &wgt);

QGridLayout layout;

layout.addWidget(&btn1);
layout.addWidget(&btn2);
layout.addWidget(&btn3);

wgt.setLayout(&layout);

wgt.show();
```





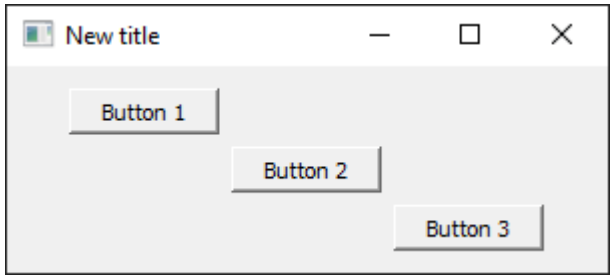
Менеджеры компоновки (Layout managers) (2)

```
QPushButton btn1("Button 1", &wgt);
QPushButton btn2("Button 2", &wgt);
QPushButton btn3("Button 3", &wgt);

QGridLayout layout;

layout.setAlignment(Qt::AlignCenter);

layout.addWidget(&btn1);
layout.addWidget(&btn2);
layout.addWidget(&btn3);

wgt.setLayout(&layout);

wgt.show();
```



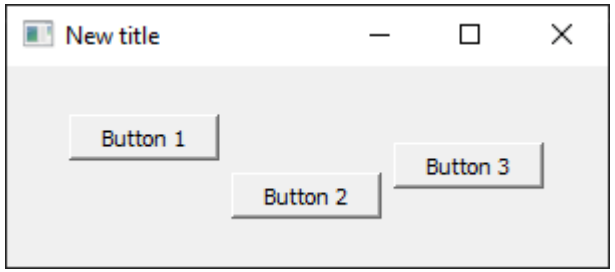
Менеджеры компоновки (Layout managers) (3)

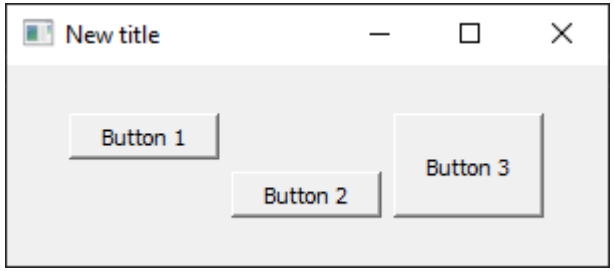
```
QPushButton btn1("Button 1", &wgt);
QPushButton btn2("Button 2", &wgt);
QPushButton btn3("Button 3", &wgt);

QGridLayout layout;

layout.setAlignment(Qt::AlignCenter);

layout.addWidget(&btn1, 0, 0);
layout.addWidget(&btn2, 1, 1);
layout.addWidget(&btn3, 2, 2);

wgt.setLayout(&layout);

wgt.show();
```



Менеджеры компоновки (Layout managers) (4)

```
QPushButton btn1("Button 1", &wgt);
QPushButton btn2("Button 2", &wgt);
QPushButton btn3("Button 3", &wgt);

QGridLayout layout;

layout.setAlignment(Qt::AlignCenter);

layout.addWidget(&btn1, 0, 0);
layout.addWidget(&btn2, 1, 1);
layout.addWidget(&btn3, 0, 2, 2, 1);

btn3.setSizePolicy(QSizePolicy::Preferred, QSizePolicy::Expanding);

wgt.setLayout(&layout);

wgt.show();
```





Обработка действий компонентов

```
void onClick() {
  QMessageBox::information(nullptr, "Button", "Clicked");
}

int main(int argc, char *argv[]) {
  QApplication app(argc, argv);

  QWidget wgt;
  wgt.resize(QSize(300, 100));
  wgt.setWindowTitle("New title");

  QPushButton btn1("Button 1", &wgt);
  btn1.connect(&btn1, &QPushButton::clicked, onClick);

  wgt.show();

  return app.exec();
}
```

Обработка событий виджетов

```
class MyWidget : public QWidget {
public:
  MyWidget();
private slots:
  void onClick();
};
```

```
void MyWidget::OnClick() {
  QMessageBox::information(this, "Button", "Clicked");
}
```

```
MyWidget::MyWidget() : QWidget(nullptr) {
  resize(QSize(300, 100));
  setWindowTitle("New title");
  QPushButton *btn1 = new QPushButton("Button 1", this);

  connect(btn1, SIGNAL(clicked()), this, SLOT(onClick()));
}
```

Класс главного окна QMainWindow

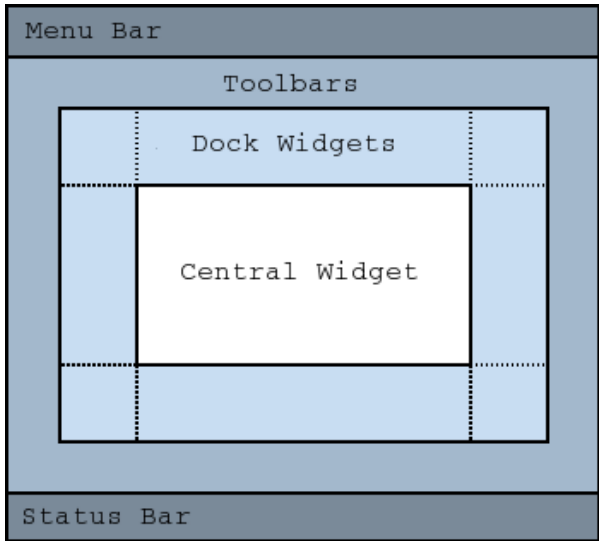
Класс QMainWindow– класс-виджет, поддерживающий основное меню (QMenuBar), панели инструментов (QToolBar), строку состояния (QStatusBar), присоединяемые виджеты (QDockWidget) и др.

Конструктор:

```
QMainWindow (QWidget* parent = Q_NULLPTR, Qt::WindowFlags flags = ...)
```

Важные методы :

```
void        addDockWidget() //Добавить присоединяемый виджет
void        addToolBar()    //Добавить панель инструментов
QMenuBar*   menuBar()       //Получить указатель на объект панели меню
void        setStatusBar()  //Установить статусную строку
QStatusBar* statusBar()     //Получить указатель на объект статусной строки
```



Архитектура проекта


```h
#include <QMainWindow>

class MainWindow: public QMainWindow {
  Q_OBJECT
public:
  MainWindow();
};
```

```cpp
#include "MainWindow.h"

MainWindow::MainWindow() : QMainWindow(nullptr) {
  resize(400, 200);
  setWindowTitle("Main Window");
  show();
}
```

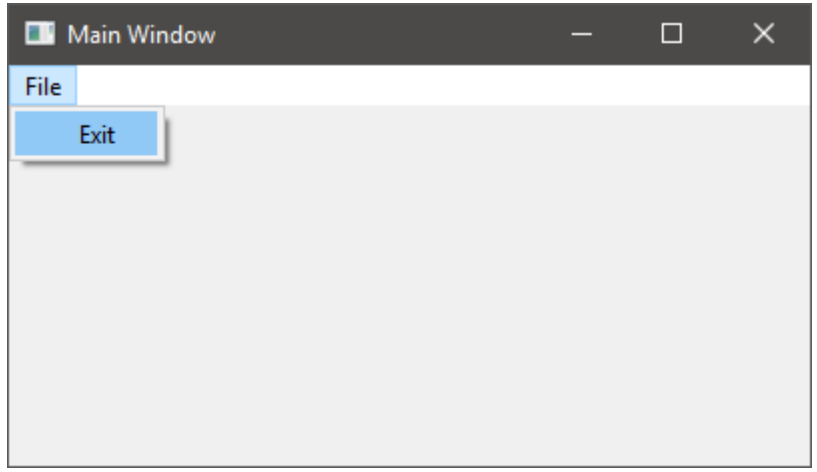
```cpp
#include <QApplication>
#include "MainWindow.hpp"

int main(int argc, char *argv[]) {
  QApplication app(argc, argv);

  MainWindow mainWindow;

  return app.exec();
}
```

Понятие действия в Qt

```
#include <QMainWindow>
#include <QAction>

class MainWindow : public QMainWindow {
  Q_OBJECT
private:
  QAction *actFileExit;
public:
  MainWindow();
};


#include "MainWindow.hpp"

MainWindow::MainWindow() : QMainWindow(nullptr) {
  resize(400, 200);
  setWindowTitle("Main Window");

  actFileExit = new QAction("&Exit", this);

  show();
}
```

Создание меню главного окна

```
#include "MainWindow.hpp"

#include <QMenu>
#include <QMenuBar>

MainWindow::MainWindow() : QMainWindow(nullptr) {
  resize(400, 200);
  setWindowTitle("Main Window");

  actFileExit = new QAction("&Exit", this);

  QMenu *menuFile = new QMenu("&File");
  menuFile->addAction(actFileExit);

  menuBar()->addMenu(menuFile);

  show();
}
```



Обработчики действий (1)

```
#include <QMainWindow>
#include <QAction>

class MainWindow : public QMainWindow {
  Q_OBJECT
private:
  QAction *actFileExit;
public:
  MainWindow();
private slots:
  void onMenuFileExit();
};
```

```
MainWindow::MainWindow() : QMainWindow(nullptr) {
  resize(400, 200);
  ...
  show();
}

void MainWindow::onMenuFileExit() {
  close();
}
```

Обработчики действий (2)

```
MainWindow::MainWindow() : QMainWindow(nullptr) {
  resize(400, 200);
  setWindowTitle("Main Window");

  actFileExit = new QAction("&Exit", this);
  connect(actFileExit,                  // кто источник сигнала
          SIGNAL(triggered()),          // какого именно сигнала
          this,                         // кто ловит сигнал
          SLOT(onMenuFileExit()));      // чем обрабатывает

  QMenu *menuFile = new QMenu("&File");
  menuFile->addAction(actFileExit);

  menuBar()->addMenu(menuFile);

  show();
}
```

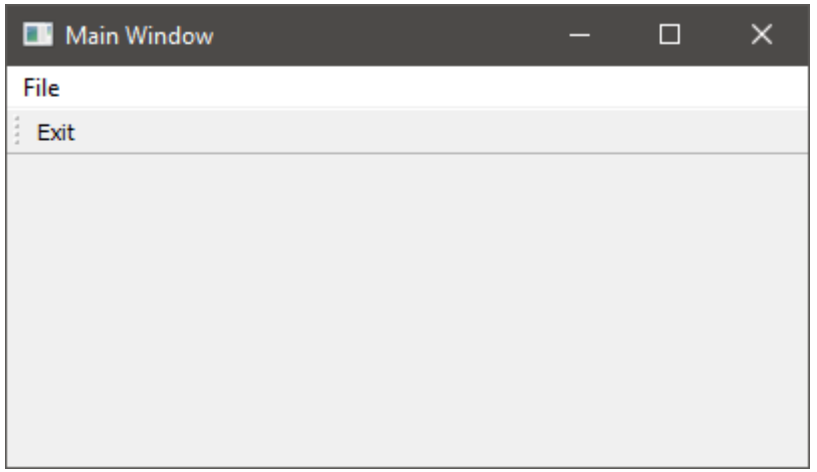
Обработчики действий (3)

```
connect(btn,                    // кто источник сигнала
        SIGNAL(clicked()),      // какого именно сигнала
        this,                   // кто ловит сигнал
        SLOT(onClick())         // чем обрабатывается сигнал
);
```

```
connect(btn,                        // кто источник сигнала
        &QPushButton::clicked,      // какого именно сигнала
        this,                       // кто ловит сигнал
        &MainWindow::onClick        // чем обрабатывается сигнал
);
```

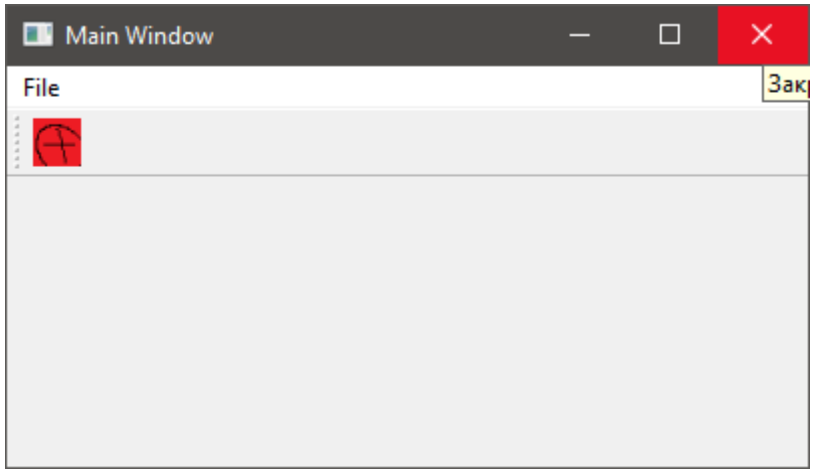
Добавление панели инструментов

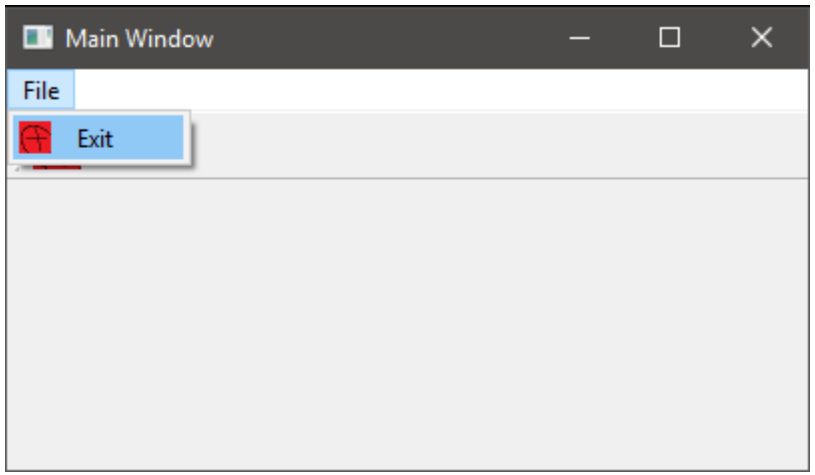
```
#include <QMenu>
#include <QMenuBar>
#include <QToolBar>

MainWindow::MainWindow() : QMainWindow(nullptr) {

  ...

  menuBar()->addMenu(menuFile);

  QToolBar *toolbar = new QToolBar(this);
  toolbar->addAction(actFileExit);
  addToolBar(Qt::TopToolBarArea, toolbar);

  show();
}
```



Добавление иконки к QAction

```
#include <QMenu>
#include <QMenuBar>
#include <QToolBar>

MainWindow::MainWindow() : QMainWindow(nullptr) {
  resize(400, 200);
  setWindowTitle("Main Window");

  actFileExit = new QAction(
                    QIcon("/home/student/KRPO_Lab_0x04/icon.png"),
                    "&Exit",
                    this);
  connect(actFileExit,
          SIGNAL(triggered()),
          this,
          SLOT(OnMenuFileExit()));

  QMenu *menuFile = new QMenu("File");
}
```





Строка состояния

```
void MyWindow::initStatusBar() {
  QStatusBar* status = new QStatusBar(this);
  setStatusBar(status);

  status->showMessage("Welcome to my prog!");

  status->setStatusTip("Status bar");
}
```

Работа с GDI: уровень Qt

```
class MainWindow : public QMainWindow {
  Q_OBJECT
private:
  QAction *actFileExit;
public:
  MainWindow();
private:
  void paintEvent(QPaintEvent *event);
private slots:
  void onMenuFileExit();
};
```


```
#include <QPainter>


void MainWindow::paintEvent(QPaintEvent *event) {
  QPainter p(this);
  p.drawLine(QPoint(100, 100), QPoint(150, 150));
}
```

Логическое разделение программного кода (1)

```
MainWindow::MainWindow() : QMainWindow(nullptr) {
    resize(400, 200);
    setWindowTitle("Main Window");

    actFileExit = new QAction(QIcon("icon.bmp"), "&Exit", t
    connect(actFileExit, SIGNAL(triggered()),
            this, SLOT(OnMenuFileExit()));

    QMenu *menuFile = new QMenu("File");
    menuFile->addAction(actFileExit);
    menuBar()->addMenu(menuFile);

    QToolBar* toolbar = new QToolBar(this);
    toolbar->addAction(actFileExit);
    toolbar->setMovable(true);
    addToolBar(Qt::TopToolBarArea, toolbar);

    QStatusBar* status = new QStatusBar(this);
    setStatusBar(status);
    status->showMessage("Welcome to my prog!");
    status->setStatusTip("Status bar");

    show();
}
```

Окно

Действие

Меню

Панель инструментов

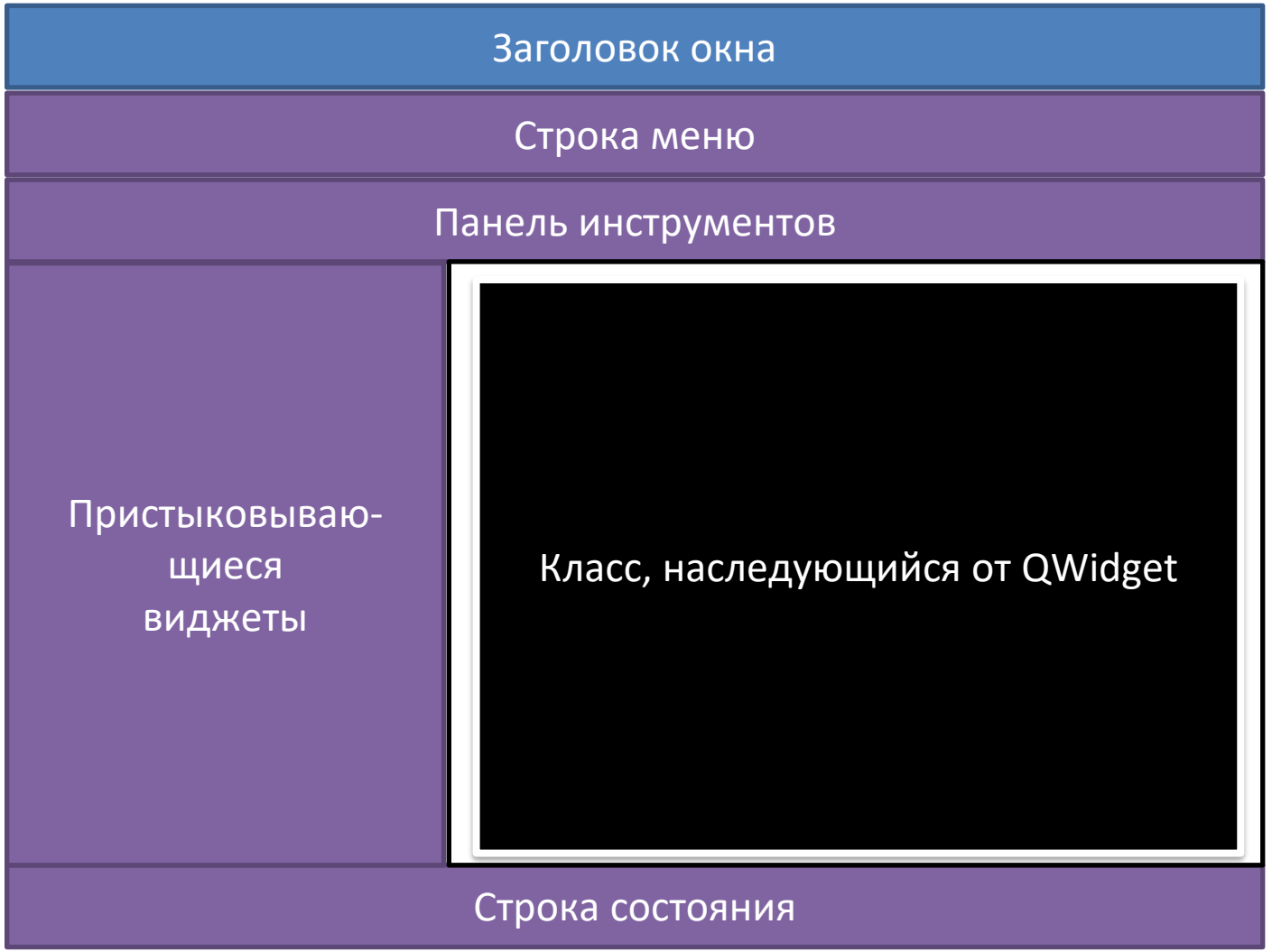
Строка состояния

Логическое разделение программного кода (2)

```
class MainWindow : public QMainWindow {
  Q_OBJECT
private:
  QAction *actFileExit;
public:
  MainWindow();
private:
  void initActions();
  void initMenuBar();
  void initToolBar();
  void initStatusBar();
private slots:
  void onMenuFileExit();
};
```

```
MainWindow::MainWindow() : QMainWindow(nullptr) {
  resize(400, 200);
  setWindowTitle("Main Window");

  initActions();
  initMenuBar();
  initToolBar();
  show();
}

void MainWindow::initActions() {
  actFileExit = new QAction(QIcon("icon.bmp"), "&Exit", this);
  connect(actFileExit, SIGNAL(triggered()), this, SLOT(OnMenuFileExit()));
}

void MainWindow::initMenuBar() {
  QMenu *menuFile = new QMenu("File");
  menuFile->addAction(actFileExit);
  menuBar()->addMenu(menuFile);
}

void MainWindow::initToolBar() {
  QToolBar* toolbar = new QToolBar(this);
  toolbar->addAction(actFileExit);
  toolbar->setMovable(true);
  addToolBar(Qt::TopToolBarArea, toolbar);
}
```

Корректная архитектура GUI-приложений



Центральный виджет

```
MainWindow::MainWindow() : QMainWindow(nullptr) {
  resize(400, 200);
  setWindowTitle("Main Window");

  InitActions();
  InitMenuBar();
  InitToolBar();

  painterWidget = new PainterWidget(this);
  setCentralWidget(painterWidget);

  show();
}
```

Рисование графических примитивов

```
void PainterWidget::paintEvent(QPaintEvent *event) {
  QPainter p(this);

  QPen pen(QColor(0, 0, 0) , 2, Qt::SolidLine);
  p.setPen(pen);

  p.drawLine(QPoint(10, 10), QPoint(100, 100));

  pen.setWidth(4);
  p.setPen(pen);

  p.drawLine(QPoint(50, 50), QPoint(200, 50));

  pen.setColor(QColor("#FF00FF"));
  p.setPen(pen);

  p.drawLine(QPoint(50, 50), QPoint(50, 100));
}
```

Пристыковывающиеся окна (1)

```
#include <QDockWidget>

MainWindow::MainWindow() : QMainWindow(nullptr) {
  resize(600, 400);
  setWindowTitle("Main Window");

  ...

  widget = new PainterWidget(this);
  setCentralWidget(widget);

  QDockWidget *dock = new QDockWidget("Log Window", this);
  dock->setWidget(new QWidget(this));
  addDockWidget(Qt::BottomDockWidgetArea, dock);

  show();
}
```

Пристыковывающиеся окна (2)

```
#include <QDockWidget>
#include <QTextEdit>

MainWindow::MainWindow() : QMainWindow(nullptr) {
  resize(600, 400);
  setWindowTitle("Main Window");

  ...

  widget = new PainterWidget(this);
  setCentralWidget(widget);

  QDockWidget *dock = new QDockWidget("Log Window", this);
  dock->setWidget(new QTextEdit(this));
  addDockWidget(Qt::BottomDockWidgetArea, dock);

  show();
}
```