

Человеко-машинное взаимодействие

Лекция 2

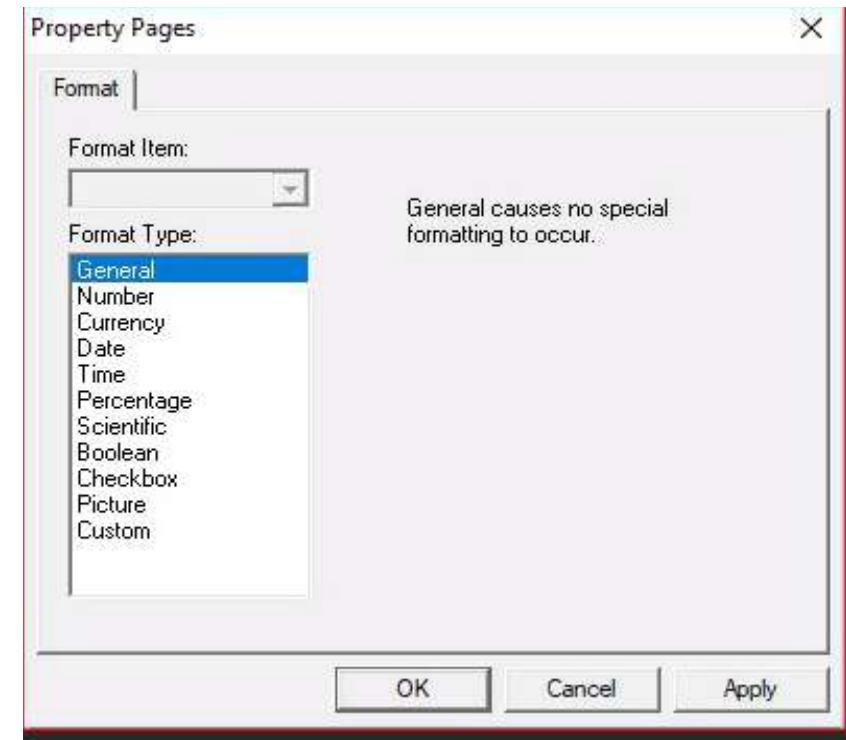
Мерзлякова Екатерина Юрьевна
к.т.н. доцент ПМИК

Диалоговые окна

Собственные
Стандартные
Окна сообщений

Правила создания диалоговых окон

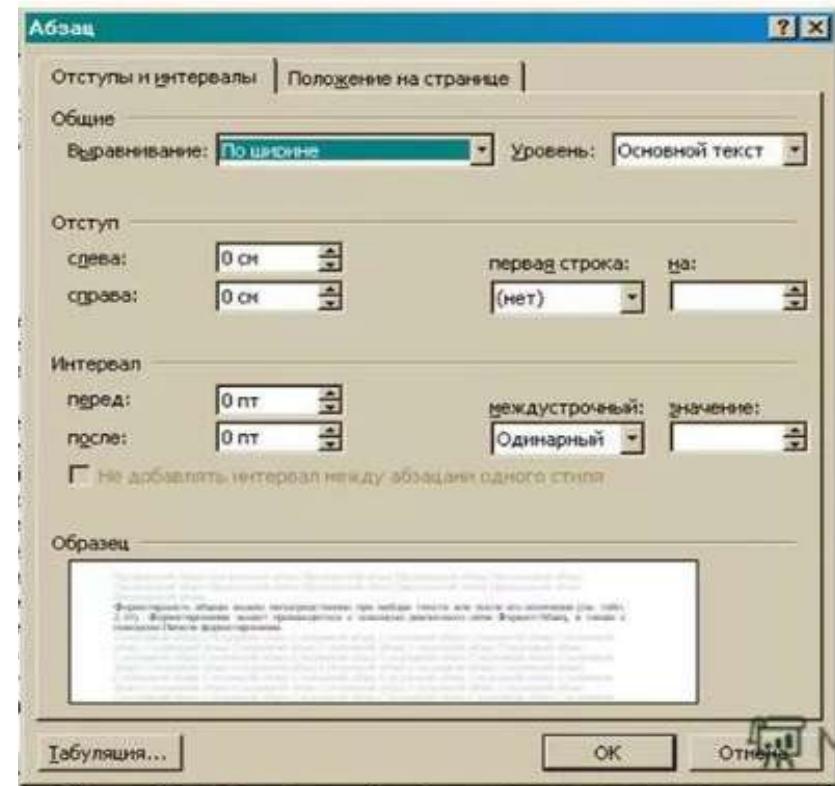
Стремитесь к тому,
чтобы диалоговое окно
не содержало ничего
лишнего и было как
можно проще.



В диалоговом окне настроек программы
желательны только основные кнопки
например **Ok**, **Cancel**, **Apply**

Правила создания диалоговых окон

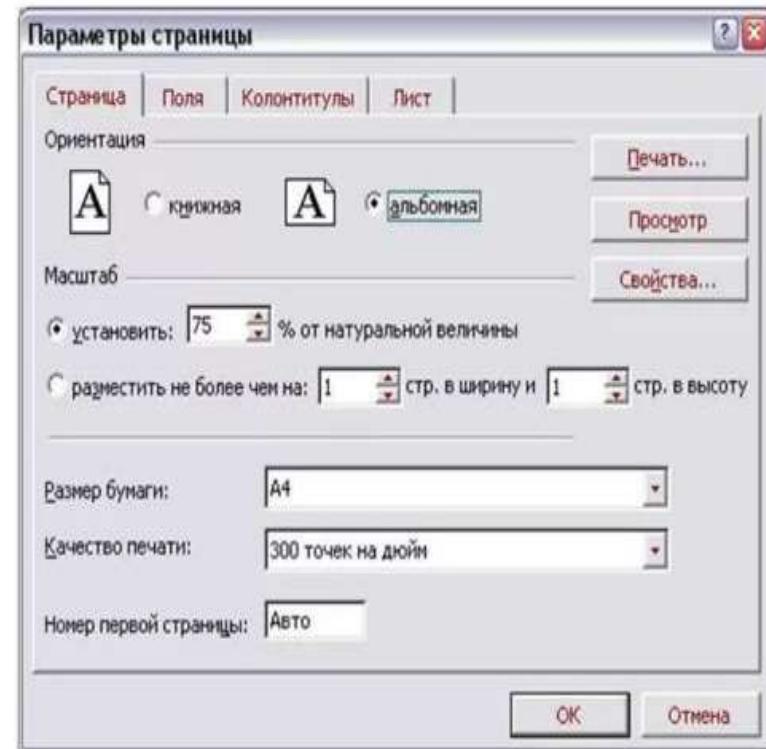
Объединяйте виджеты в логические группы, снабжая их прямоугольной рамкой и подписью.



Используйте горизонтальные и вертикальные линии для разделения.

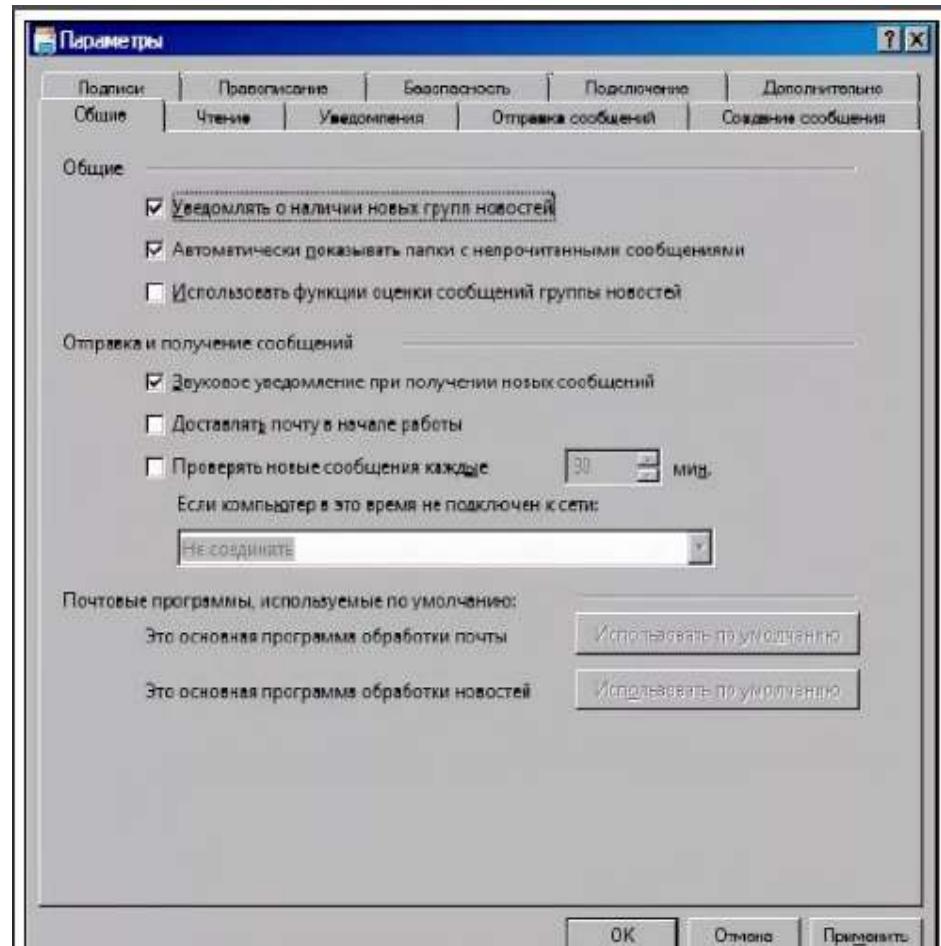
Правила создания диалоговых окон

Никогда не делайте
содержимое
диалогового окна
прокрутивающимся.
Если окно содержит
много элементов, то
постарайтесь разбить их
на группы и разместить
их **с помощью вкладок.**



Правила создания диалоговых окон

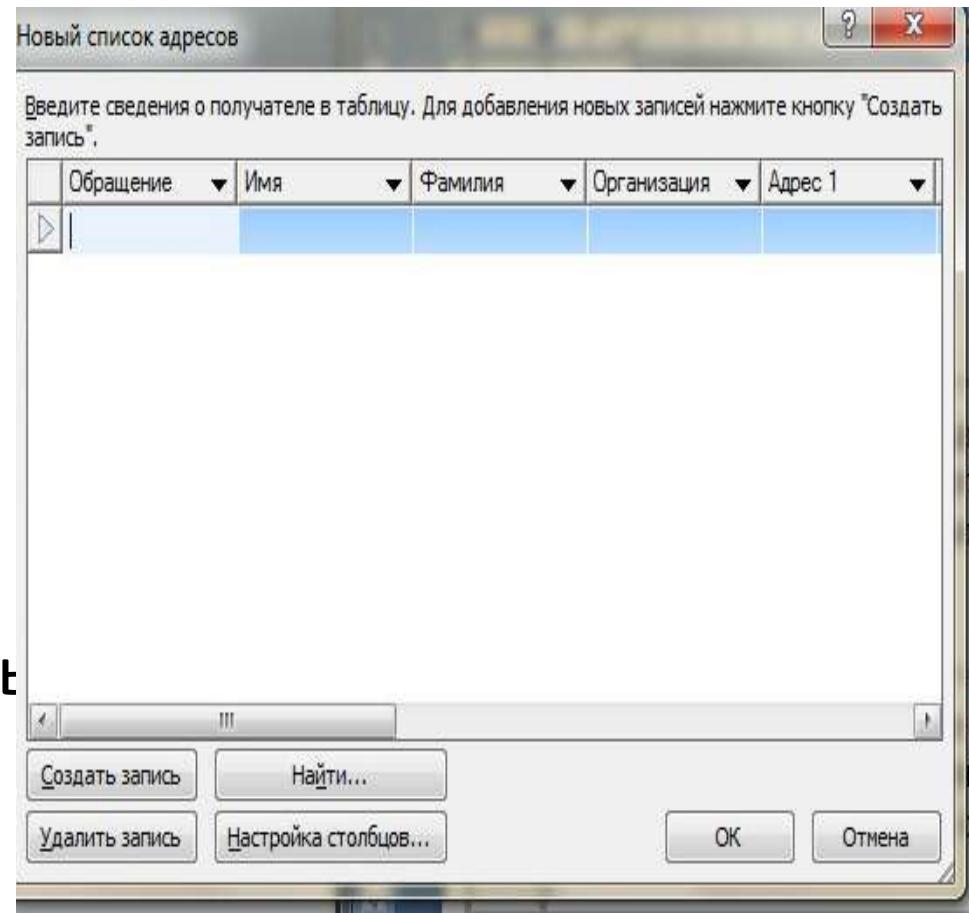
Нежелательно, чтобы вкладки в диалоговом окне занимали более одного ряда – это усложняет поиск.



Правила создания диалоговых окон

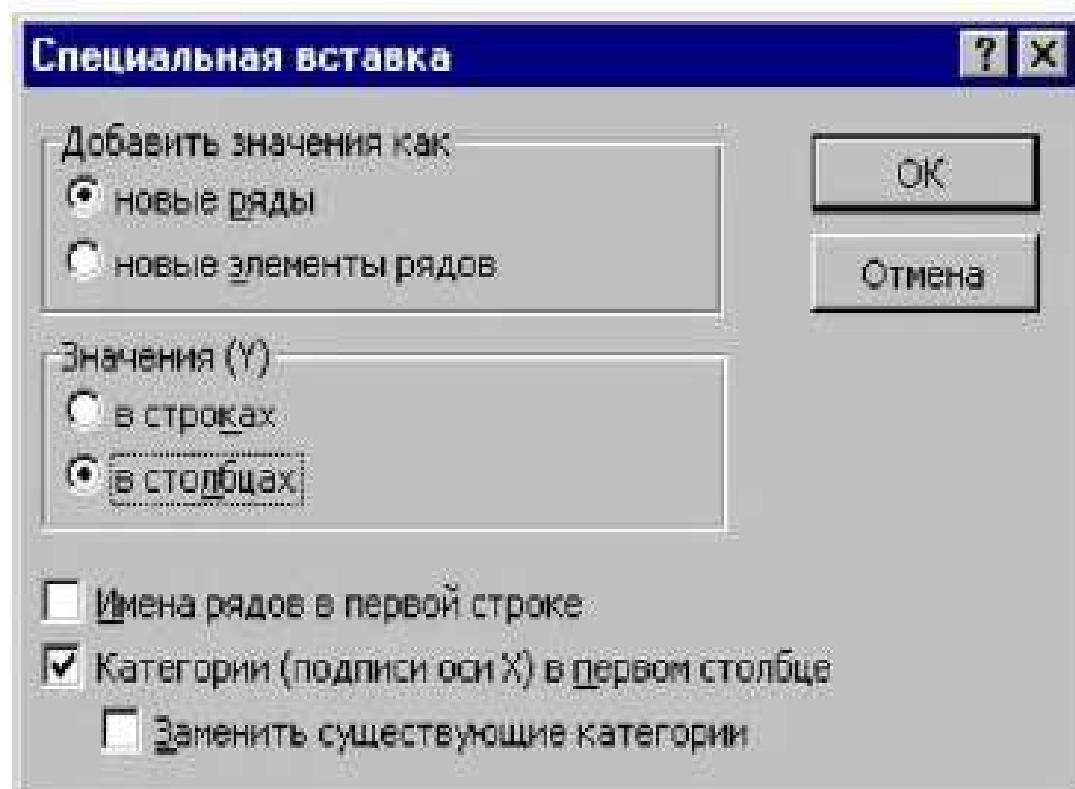
**Избегайте создания
диалоговых окон с
неизменяемыми
размерами.**

Пользователь всегда
должен иметь
возможность увеличить
или уменьшить
размеры окна по
своему усмотрению.



Правила создания диалоговых окон

Сложные диалоговые окна лучше снабжать дополнительной кнопкой **Help (?)**, при нажатии на которую должно открываться окно контекстной помощи.



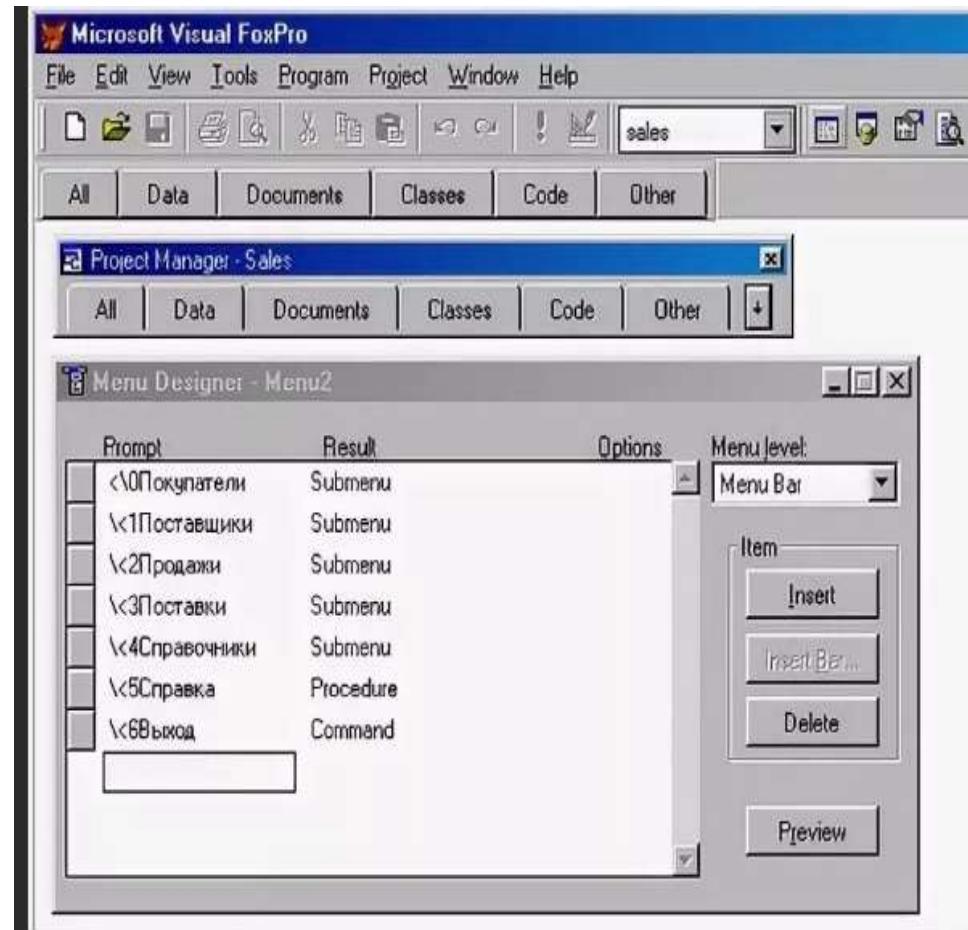
Правила создания диалоговых окон

Команды меню,
вызывающие диалоговые
окна, **должны**
оканчиваться
многоточием, например,
Open... Settings...
Это делается для того,
чтобы пользователь
заранее знал, что нажатие
команды меню приведет к
открытию диалогового
окна.



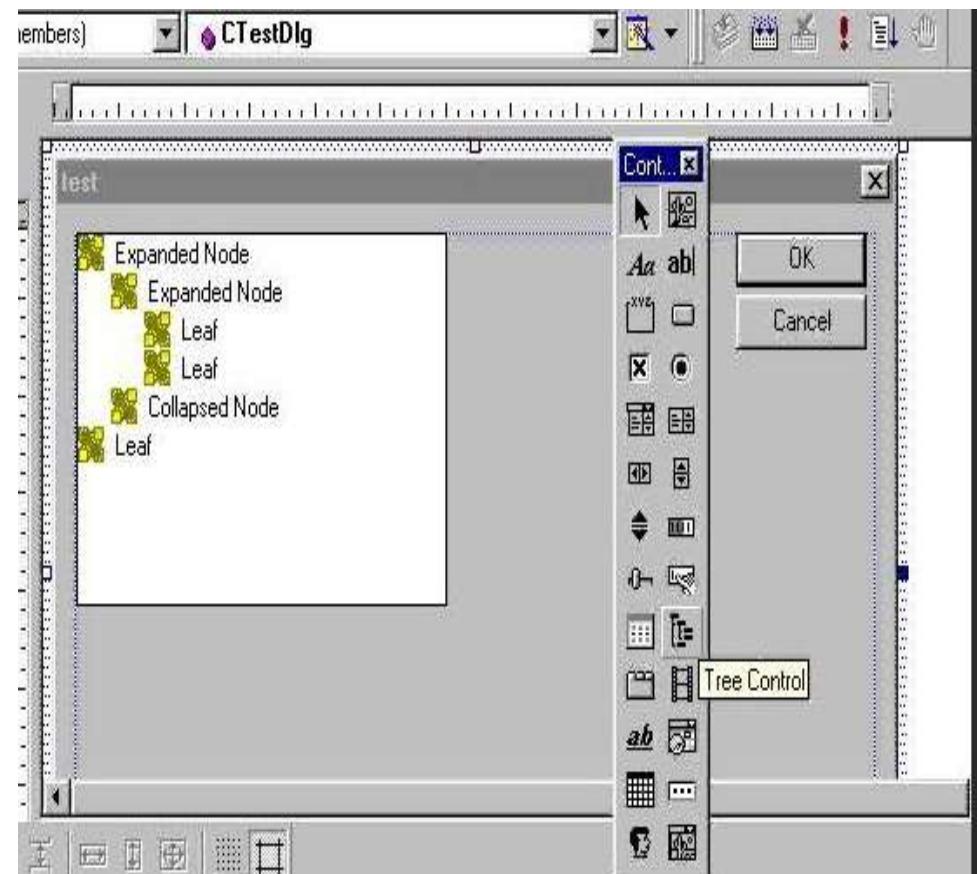
Правила создания диалоговых окон

Старайтесь не добавлять меню в диалоговые окна.
Меню должны использоваться в окне основной программы.



Правила создания диалоговых окон

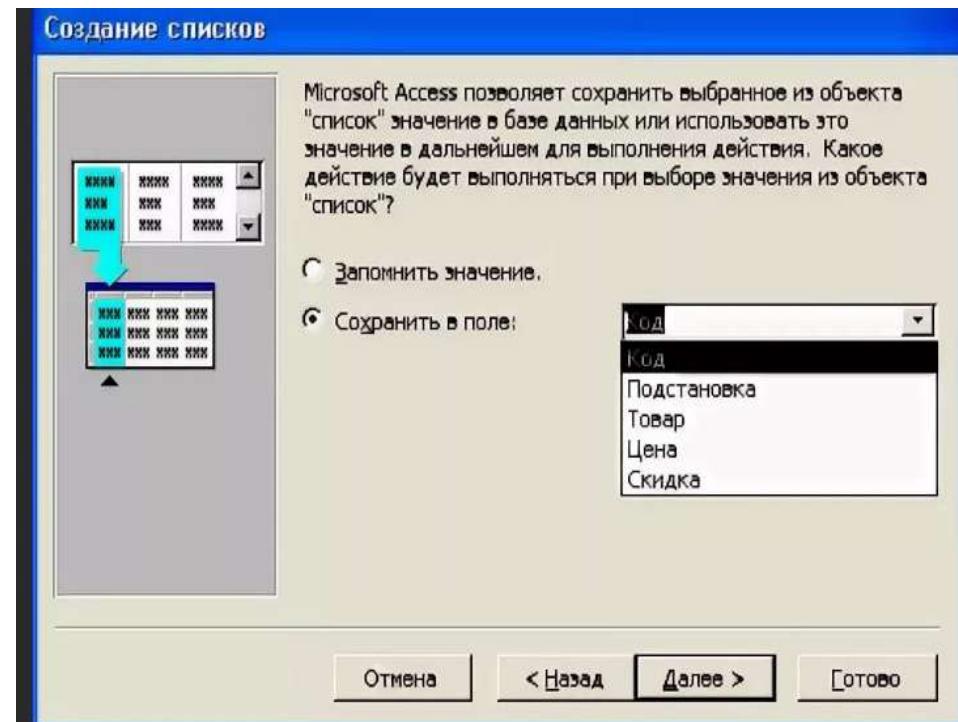
По возможности
используйте
стандартные виджеты,
хорошо знакомые
пользователям. Не
забывайте, что для
освоения новых
элементов управления
может понадобиться
дополнительное время



Правила создания диалоговых окон

Для показа настроек
избегайте
использования цвета.

В большинстве случаев
текст – лучшая
альтернатива



Правила создания диалоговых окон

Alt	Показать скрытую строку меню
Alt + D	Выбрать адресную строку
Alt + P	Показать панель предварительного просмотра в Проводнике Windows 8
Alt + Tab	Переключаться последовательно между открытыми окнами вперед
Alt + Shift + Tab	Переключаться последовательно между открытыми окнами назад
Alt + Esc	Переключаться между запущенными программами в порядке их запуска
Alt + F4	Закрыть текущее окно или открыть окно завершения работы на Рабочем столе
Alt + Enter	Открыть окно "Свойства" выбранного элемента
Alt + Стрелка Влево	Перейти в предыдущий каталог (Назад)
Alt + Стрелка Вправо	Перейти в следующий каталог (Вперед)
Alt + Стрелка Вверх	Перейти вверх на один уровень в Проводнике (как простая стрелка вверх в Windows XP)
Alt + пробел	Создать новое окно с тем же содержимым, что и текущее

Необходимо снабдить все элементы окна клавишами быстрого вызова, которые позволят, нажав букву совместно с клавишей Alt, установить фокус на нужном элементе.

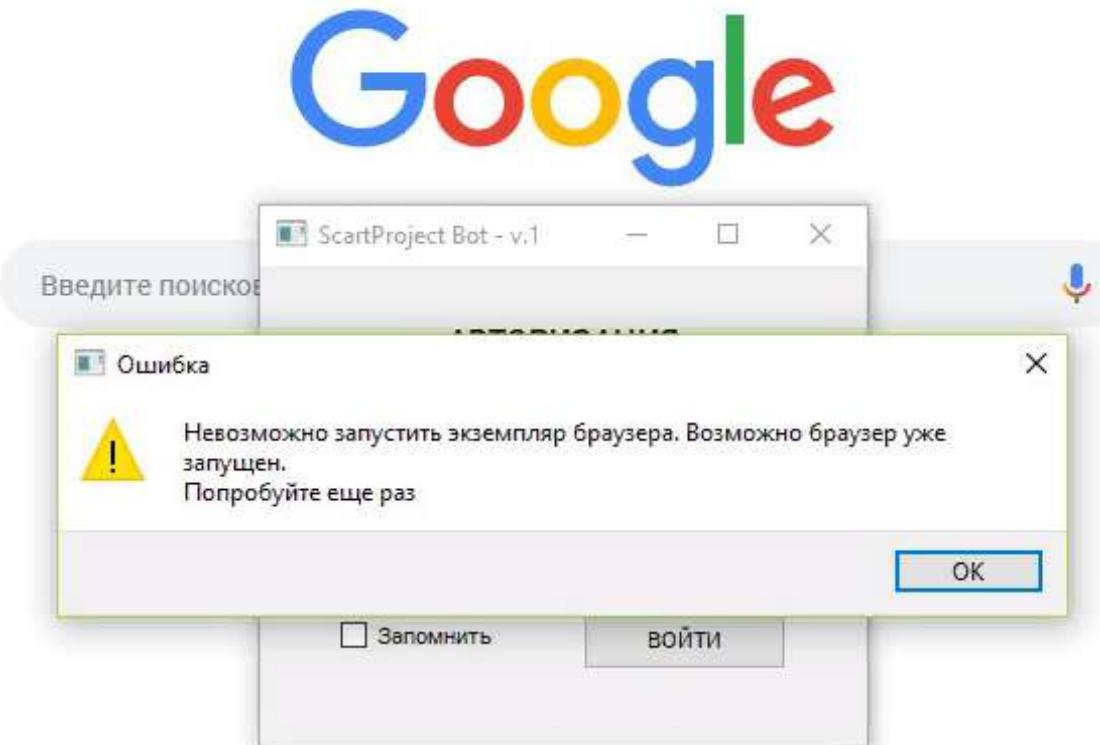
Класс QDialog

Модальные
Немодальные

`QDialog::setModal()`
`QDialog::isModal()`

Значение `true` означает модальный режим,
`false` – немодальный.

Модальные диалоговые окна

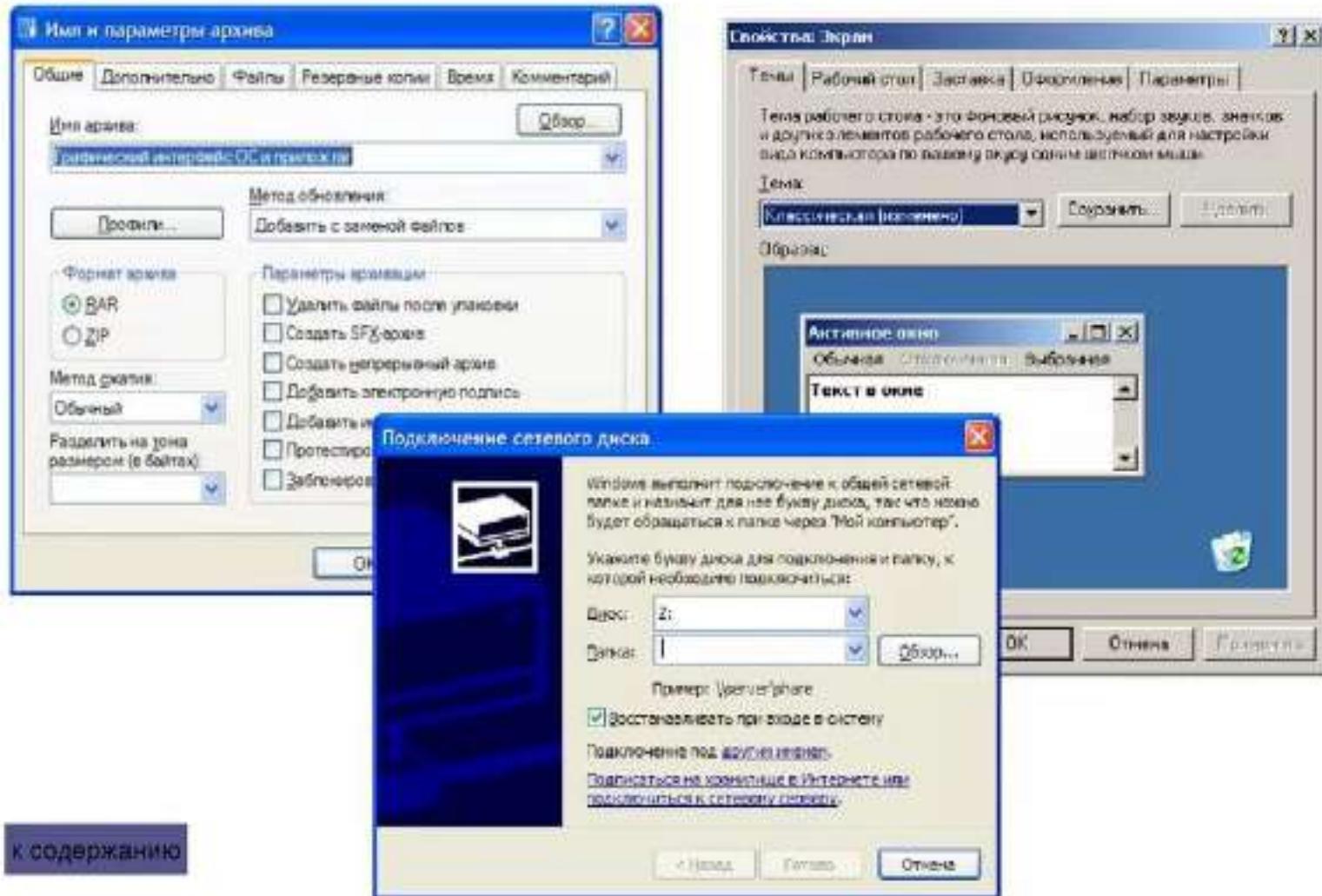


Модальные диалоговые окна

Цикл запускается вызовом слота exec(), который возвращает после закрытия диалогового окна значение целого типа, которое информирует о нажатой кнопке и может равняться: QDialog::Accepted или QDialog::Rejected, что соответствует кнопкам Ok и Cancel.

```
InputDialog* pdlg = new InputDialog(&data);
if (pdlg->exec()==QDialog::Accepted) {
    //Пользователь выбрать Accepted
    //Получить данные для дальнейшего анализа и обработки
    Data data = pdlg->getData;
    ...
}
delete pdlg;
```

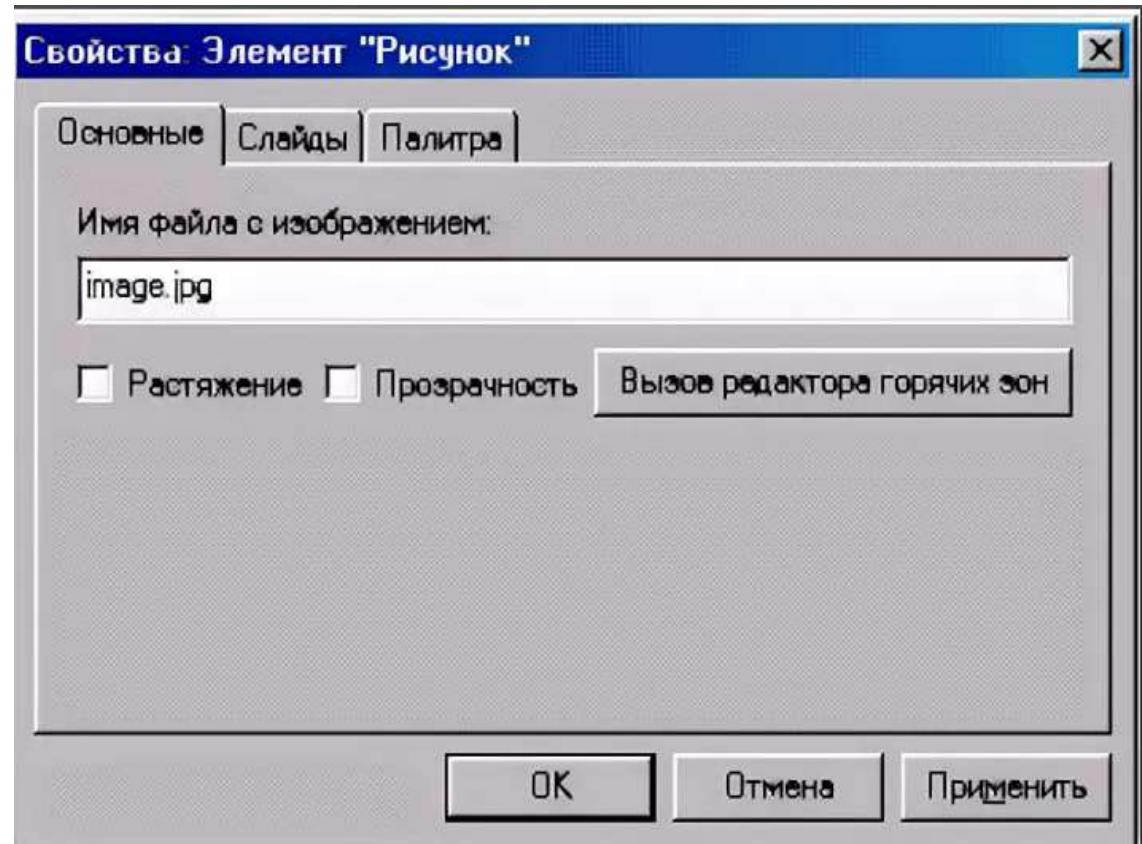
Немодальные диалоговые окна



[К содержанию](#)

Немодальные диалоговые окна

show()
hide()



Немодальные диалоговые окна

show()

вызовы raise()

и activateWindow()

hide()

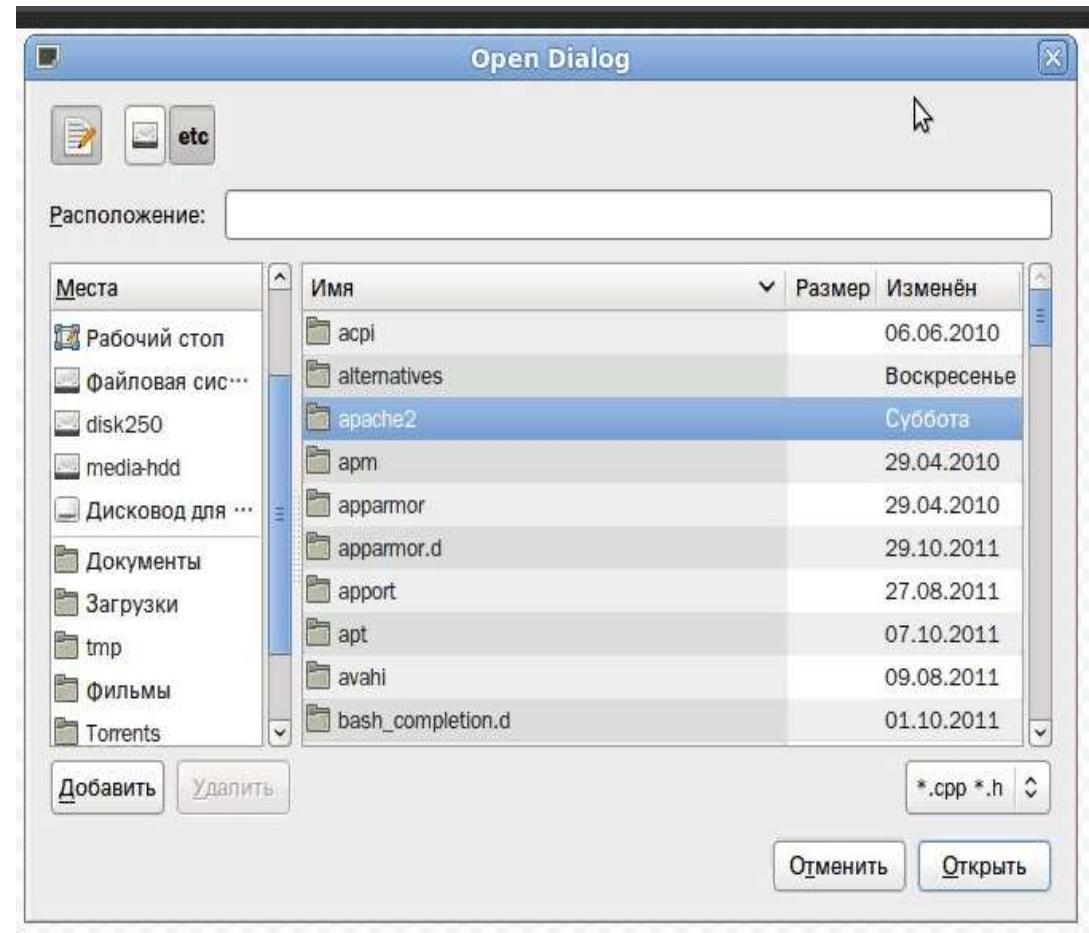


Стандартные диалоговые окна



Диалоговое окно выбора файлов

QFileDialog



Диалоговое окно выбора файлов

`getOpenFileName()`

`getOpenFileNames()`

`getSaveFileName()`

`getExistingDirectory()`

Диалоговое окно выбора файлов

```
QString str = QFileDialog::getOpenFileName(0, "Open Dialog", "", "*.cpp *.h");
```

Диалоговое окно настройки принтера

модуль QtPrintSupport
QT += printsupport

```
QPrinter printer;
QPrinterDialog *pPrintDialog = new QPrintDialog(&printer);
if(pPrinterDialog->exec() == QDialog::Accepted) {
    // печать
}
delete pPrinterDialog;
```

Диалоговое окно выбора цвета

```
QColor color = QColorDialog::getColor(blue);
if(!color.isValid()){
    //Cancel
}
```

Диалоговое окно выбора шрифта

```
bool bOk;  
QFont fnt = QFontDialog::getFont(&bOk);  
if(!bOk){  
    //Cancel  
}
```

Диалоговое окно ввода

QInputDialog

getText() – для текста или пароля

getInt() – для ввода целых чисел

getDouble() – для ввода чисел с плавающей точкой
двойной точности

getItem() – для выбора элемента

Стандартные диалоговые окна

Диалоговое окно ввода

QInputDialog

getInteger()

getDouble()

getText()

getItem()

- 1** указатель на виджет предка
- 2** заголовок диалогового окна
- 3** поясняющий текст

4	0	режим ввода паролей	текущая строка
5	-2147483647	текст для инициализации	список строк
6	2147483647		редактирование

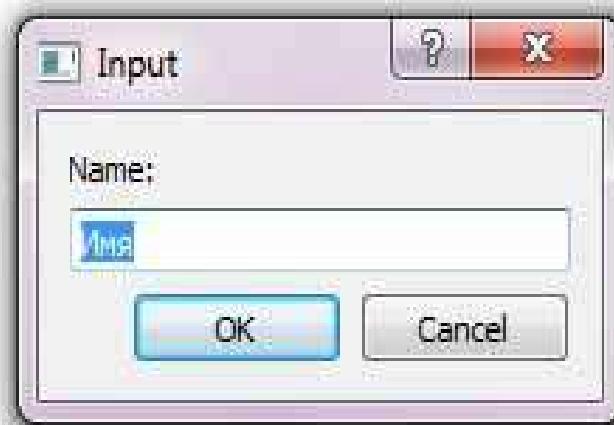
Ok или **Cancel**, и флаги окна

Диалоговое окно ввода

```
#include "mainwindow.h"
#include <QApplication>
#include <QInputDialog>
#include <QString>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    bool bOk;
    QString str =
        QInputDialog::getText(0,
            "Input", "Name:", QLineEdit::Normal,
            "Имя", &bOk);

    if (!bOk) {
        //была нажата кнопка Cancel
    }
    return a.exec();
}
```

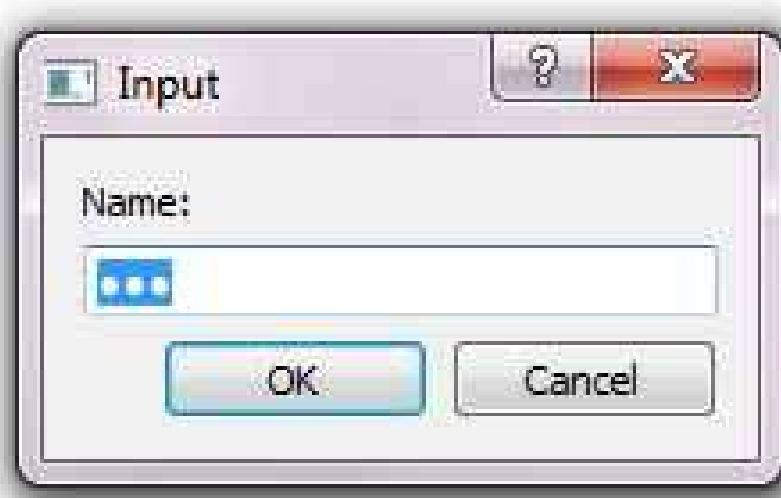


Стандартные диалоговые окна

Диалоговое окно ввода

```
#include "mainwindow.h"
#include <QApplication>
#include <QInputDialog>
#include <QString>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    bool bOk;
    QString str =
        QInputDialog::getText(0,
            "Input", "Name:", QLineEdit::Password,
            "Имя", &bOk);
    if (!bOk) {
        //была нажата кнопка Cancel
    }
    return a.exec();
}
```



Стандартные диалоговые окна

Диалоговое окно прогресса

QProgressDialog

> 3 секунд

QDialog

setMinimumDuration()
setTotalSteps()
setProgress()

Cancel

`canceled()`

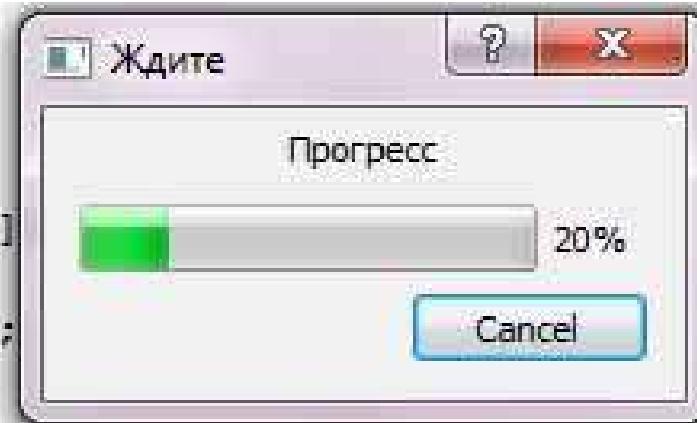
reset()
setAutoReset()
setAutoClose()

Стандартные диалоговые окна

Диалоговое окно прогресса

```
#include "mainwindow.h"
#include <QApplication>
#include <QProgressDialog>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    int n = 100000;
    QProgressDialog *pprd =
        new QProgressDialog("Прогресс", "&Cancel", 0, n);
    pprd->setMinimumDuration(0);
    pprd->setWindowTitle("Ждите");
    for(int i=0; i<n; ++i){
        pprd->setValue(i);
        a.processEvents();
        if (pprd->wasCanceled()) {
            break; }
    }
    pprd->setValue(n);
    delete pprd;
    return a.exec();
}
```



Диалоговое окно мастера

```
class Wizard: public QWizard {
private:
    QWizardPage* createPage(QWidget* pwgt, QString strTitle)
    {
        QWizardPage* ppage= new QWizardPage;
        ppage->setTitle(strTitle);

        QVBoxLayout* playout = new QVBoxLayout;
        playout->addWidget(pwgt);
        ppage->setLayout(playout);

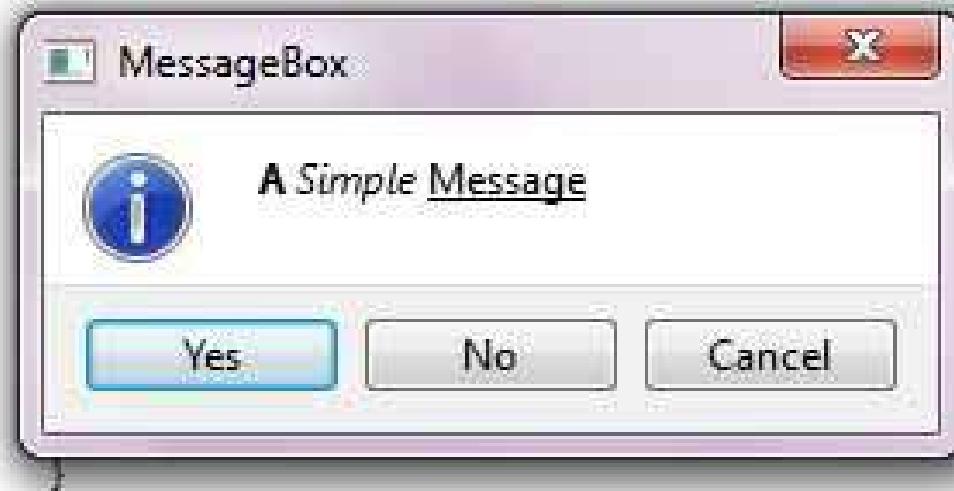
        return ppage;
    }

public:
    Wizard::Wizard(QWidget* pwgt = 0): QWizard(pwgt)
    {
        addPage(createPage(new QLabel("<H1>Label 1</H1>"), "One"));
        addPage(createPage(new QLabel("<H1>Label 2</H1>"), "Two"));
        addPage(createPage(new QLabel("<H1>Label 3</H1>"), "Three"));
    }
};
```

Стандартные диалоговые окна

Окно сообщений

```
#include "mainwindow.h"
#include <QApplication>
#include <QMessageBox>
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    QMessageBox* pmbx = new QMessageBox(
        "MessageBox", "<b>A</b> <i>Simple</i> <u>Message</u>",
        QMessageBox::Information,
        QMessageBox::Yes,
        QMessageBox::No,
        QMessageBox::Cancel |
        QMessageBox::Escape );
    int n = pmbx->exec();
    delete pmbx;
    if(n==QMessageBox::Yes) {
        /*нажата кнопка Yes*/
    }
    return a.exec();
}
```



Стандартные диалоговые окна

Окно сообщений

QMessageBox

setText()

setButtonText()

setWindowTitle()

setIcon()

Константа	Значение	Вид
NoIcon	0	-
Information	1	
Warning	2	
Critical	3	
Question	4	

Стандартные диалоговые окна

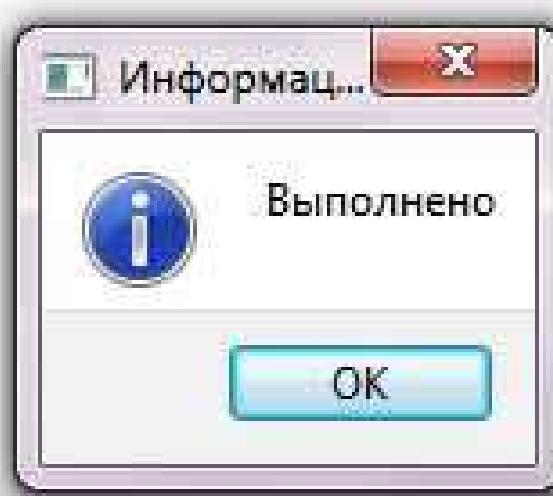
Первый параметр метода `setButtonText()`

Константа	Значение
NoButton	0
OK	1
Cancel	2
Yes	3
No	4
Abort	5
Retry	6
Ignore	7
YesAll	8
NoAll	9
Escape	0x200
Default	0x100

Стандартные диалоговые окна

Окно информационного сообщения

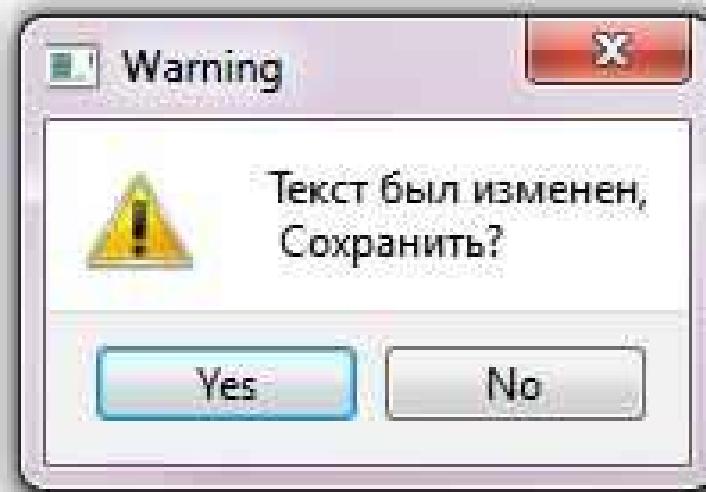
```
1 #include "mainwindow.h"
2
3 #include <QApplication>
4 #include <QMessageBox>
5
6 int main(int argc, char *argv[])
7 {
8     QApplication a(argc, argv);
9     QMessageBox::information(0, "Информация", "Выполнено");
10    return a.exec();
11 }
```



Стандартные диалоговые окна

Окно предупреждающего сообщения

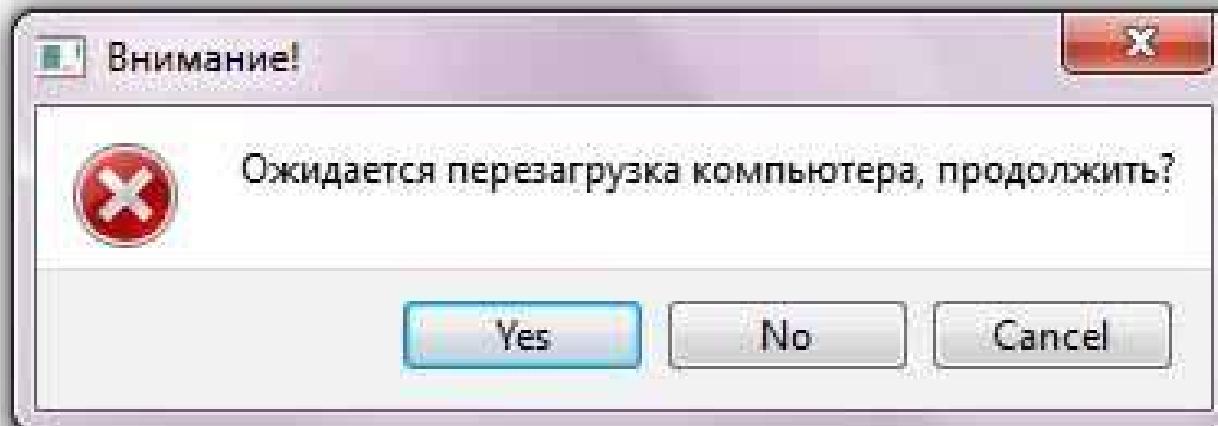
```
#include "mainwindow.h"
#include <QApplication>
#include <QMessageBox>
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    int n = QMessageBox::warning(0,
        "Warning",
        "Текст был изменен,\nп. Сохранить?",
        "Yes",
        "No",
        QString(),
        0,
        1);
    if(!n){ //Saving the changes!
    }
    return a.exec();
}
```



Стандартные диалоговые окна

Окно критического сообщения

```
6 QApplication a(argc, argv);
7 int n = QMessageBox::critical(0,
8     "Внимание!",
9     "Ожидается перезагрузка компьютера, продолжить?",
10    QMessageBox::Yes | QMessageBox::Default,
11    QMessageBox::No,
12    QMessageBox::Cancel | QMessageBox::Escape
13 );
14 if (n == QMessageBox::Yes) { //Do it
15 }
16 return a.exec();
17 }
```



Стандартные диалоговые окна

Окно сообщения About

```
1 #include "mainwindow.h"
2
3 #include <QApplication>
4 #include <QMessageBox>
5
6 int main(int argc, char *argv[])
7 {
8     QApplication a(argc, argv);
9     QMessageBox::about(0, "About", "My Program");
10    return a.exec();
11 }
```



Предоставление помощи

Всплывающие подсказки

```
1 #include "mainwindow.h"
2 #include <QApplication>
3 #include <QPushButton>
4 int main(int argc, char *argv[])
5 {
6     QApplication a(argc, argv);
7     QPushButton * pcmd = new QPushButton("Ok");
8     pcmd->show();
9     pcmd->setToolTip("Это кнопка");
10    return a.exec();
11 }
12 }
```



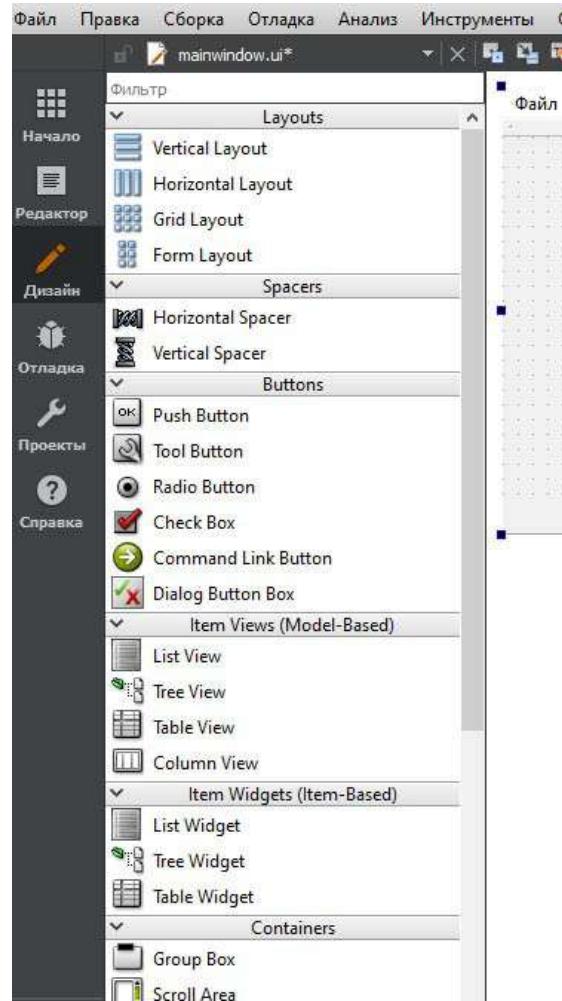
Предоставление помощи

Подсказка «Что это?»

```
#include "mainwindow.h"
#include <QApplication>
#include <QLabel>
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    QLabel lbl("Нажмите ?",
               Qt::WindowTitleHint|Qt::WindowSystemMenuHint
               |Qt::WindowContextHelpButtonHint);
    lbl.setWhatsThis("<i>Это метка</i><br><b>Class QLabel</b>");
    lbl.show();
    return a.exec();
}
```

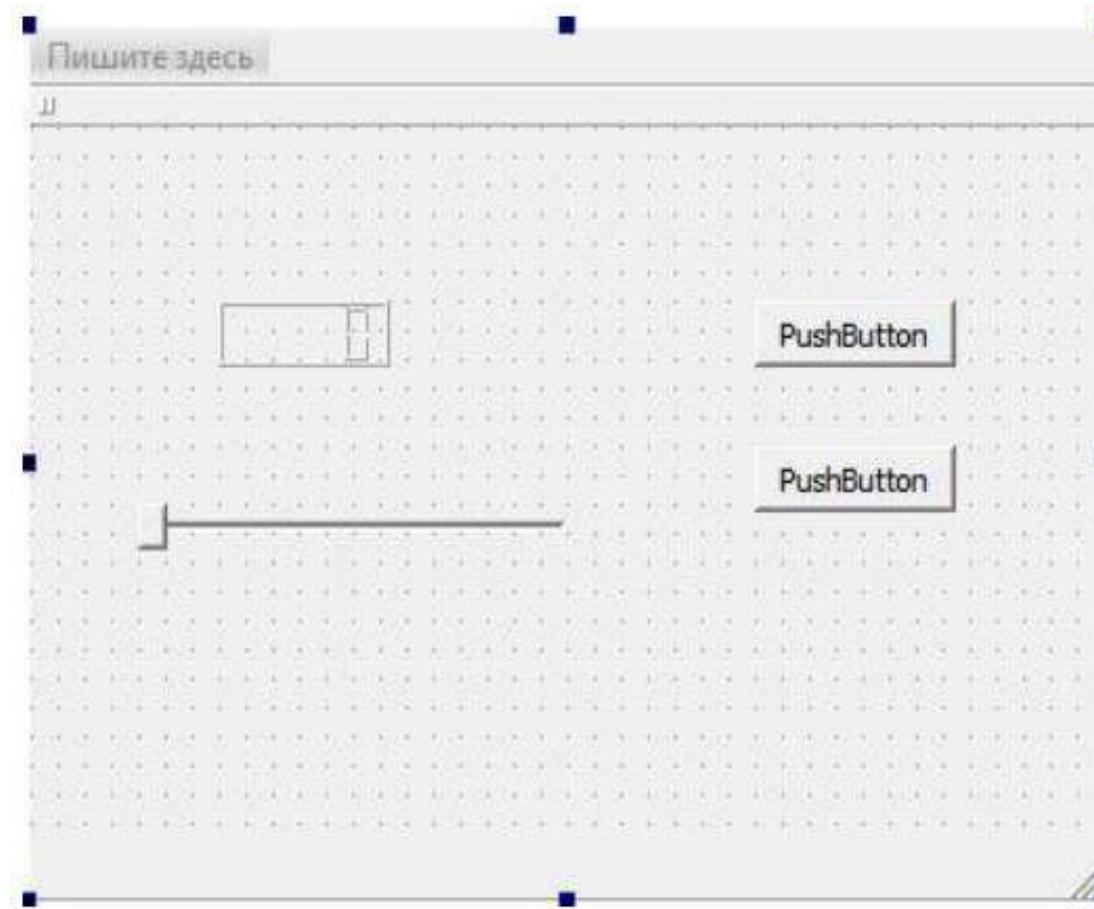


Создание собственного диалогового окна

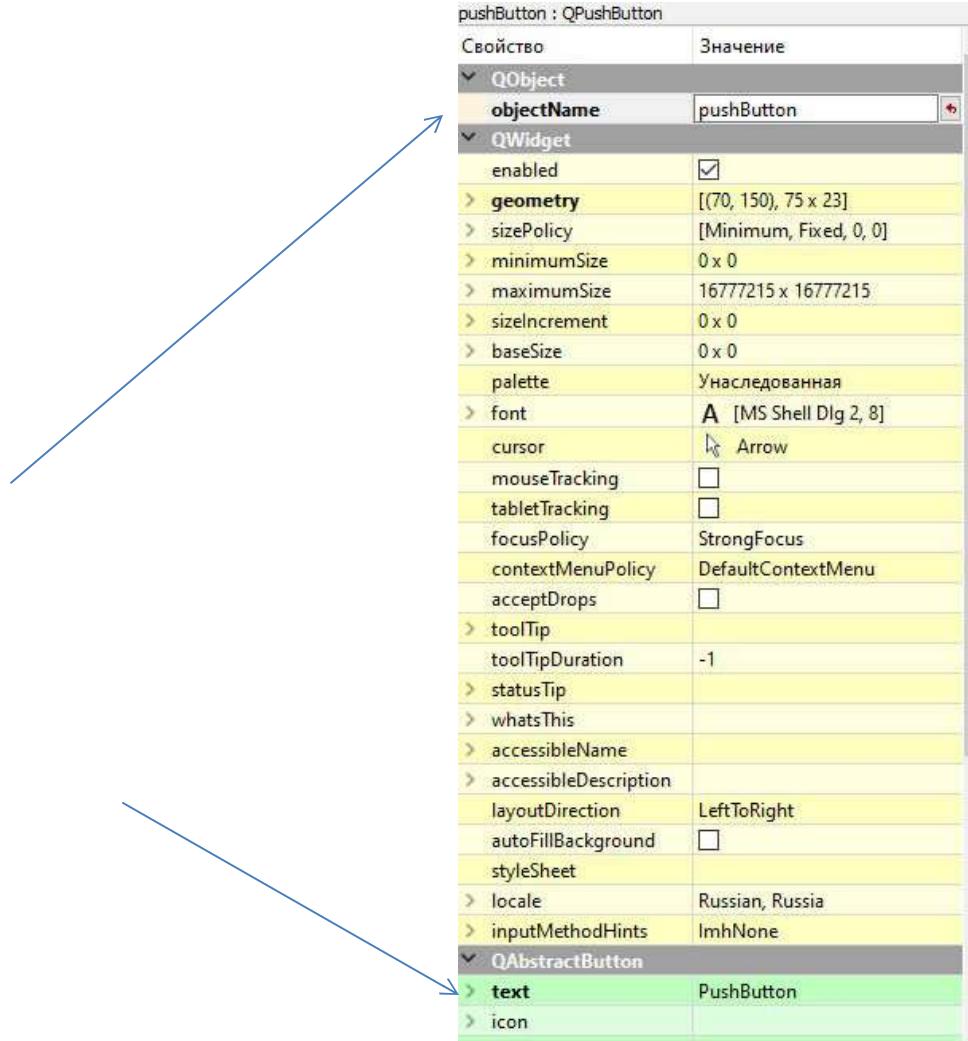


Создание собственного диалогового окна

PushButton
Horizontal Slider
LCDNumber



Создание собственного диалогового окна



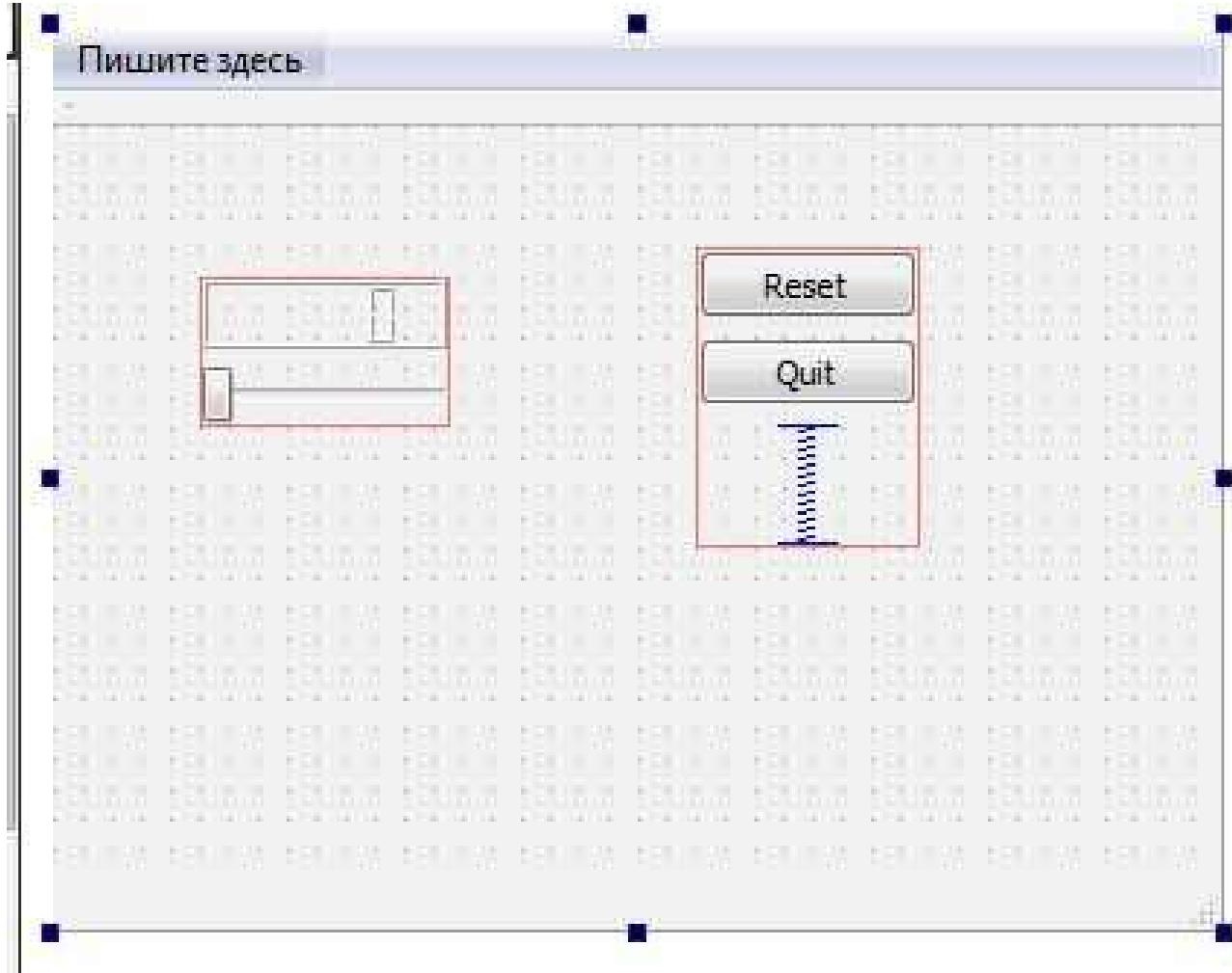
Текст- **&Reset**

ObjectName - **butReset**

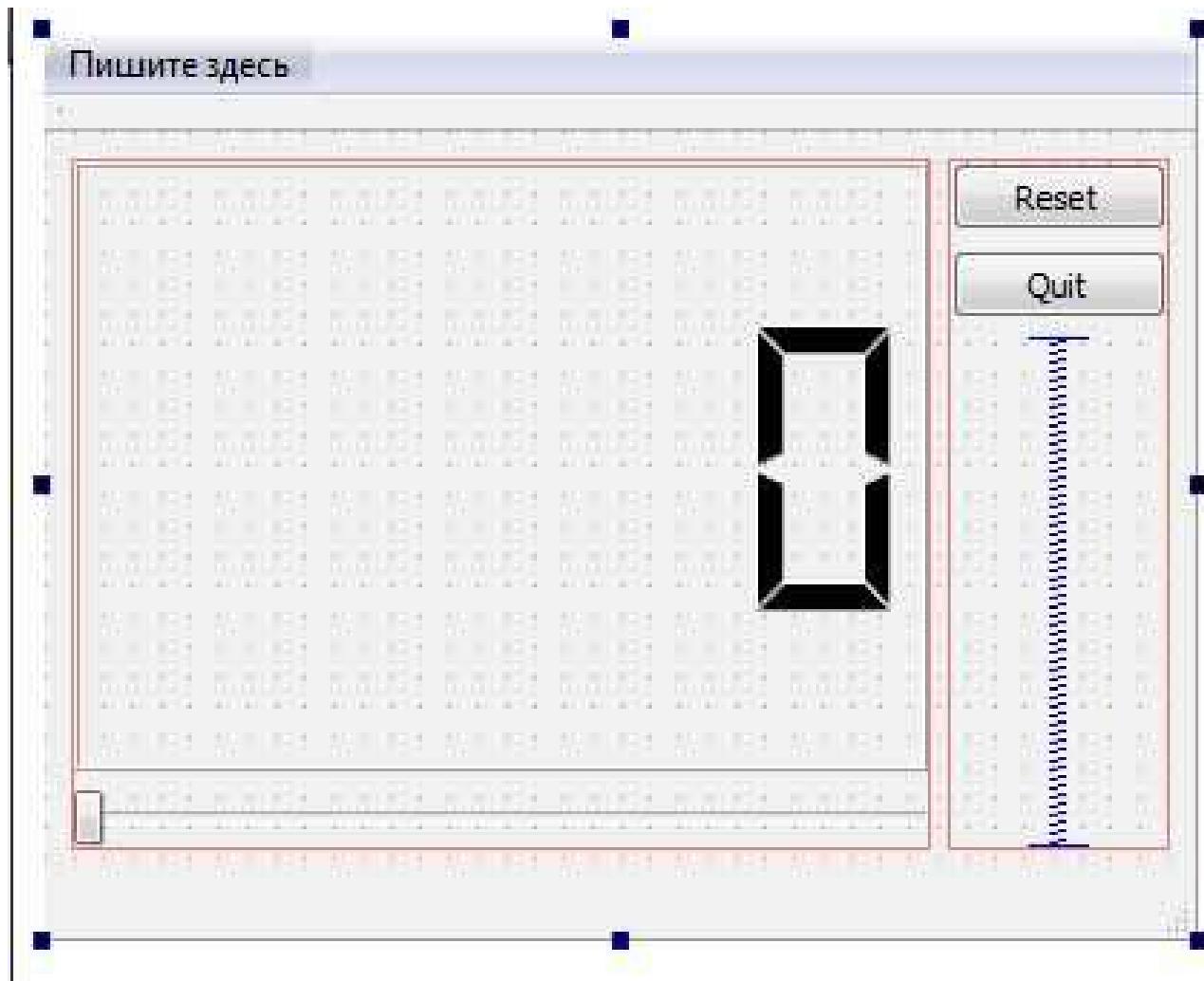
Текст - **&Quit**

ObjectName - **butQuit**

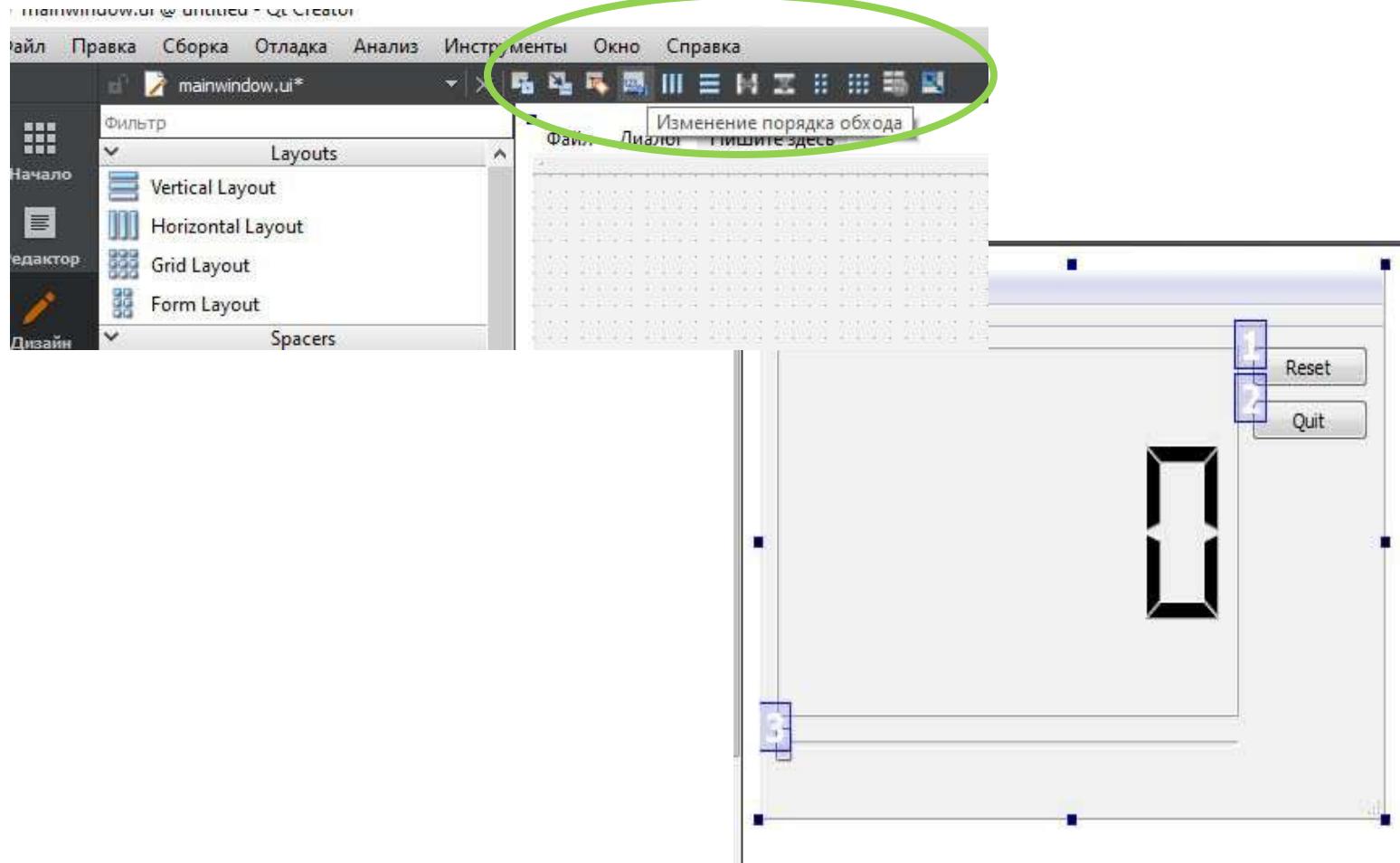
Создание собственного диалогового окна



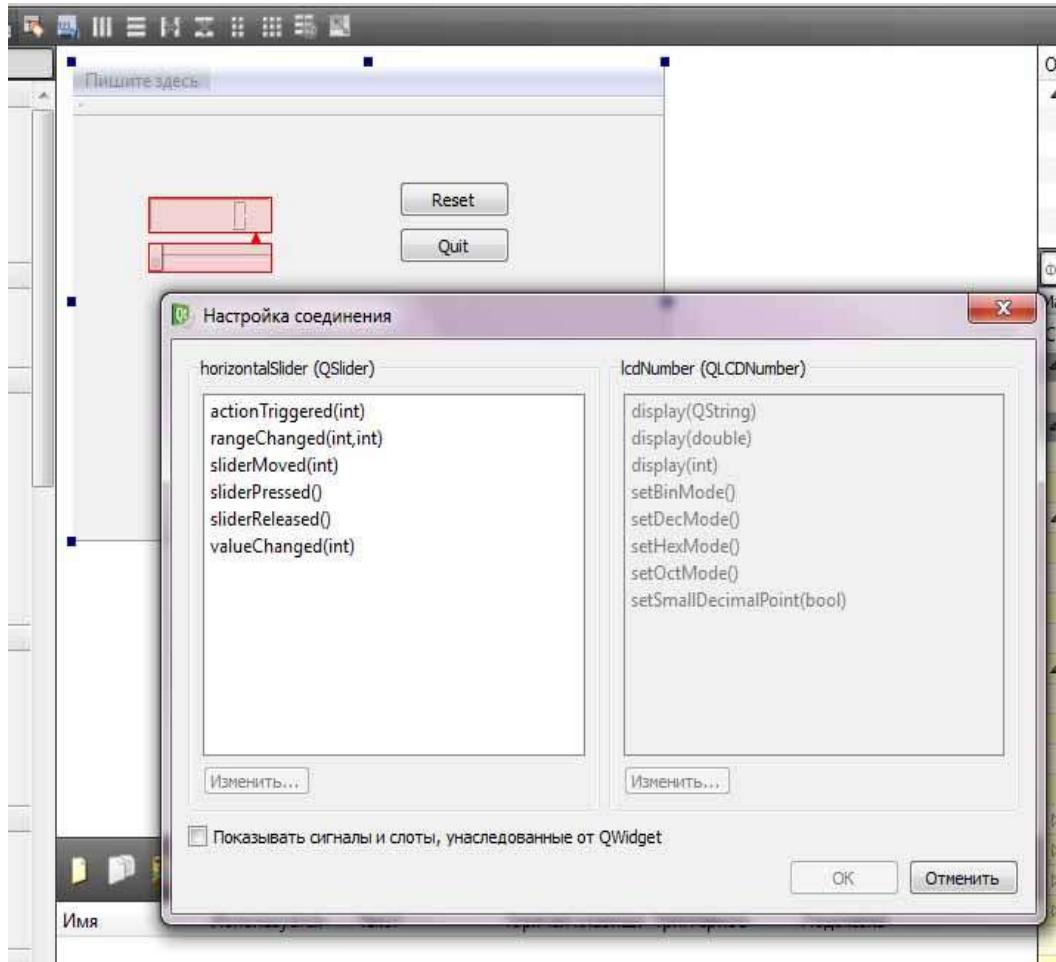
Создание собственного диалогового окна



Создание собственного диалогового окна



Создание собственного диалогового окна



В левой колонке выберите сигнал **valueChanged(int)**, в правой – **display(int)**.

Далее соединить кнопку **Quit** со слотом формы **close()**.

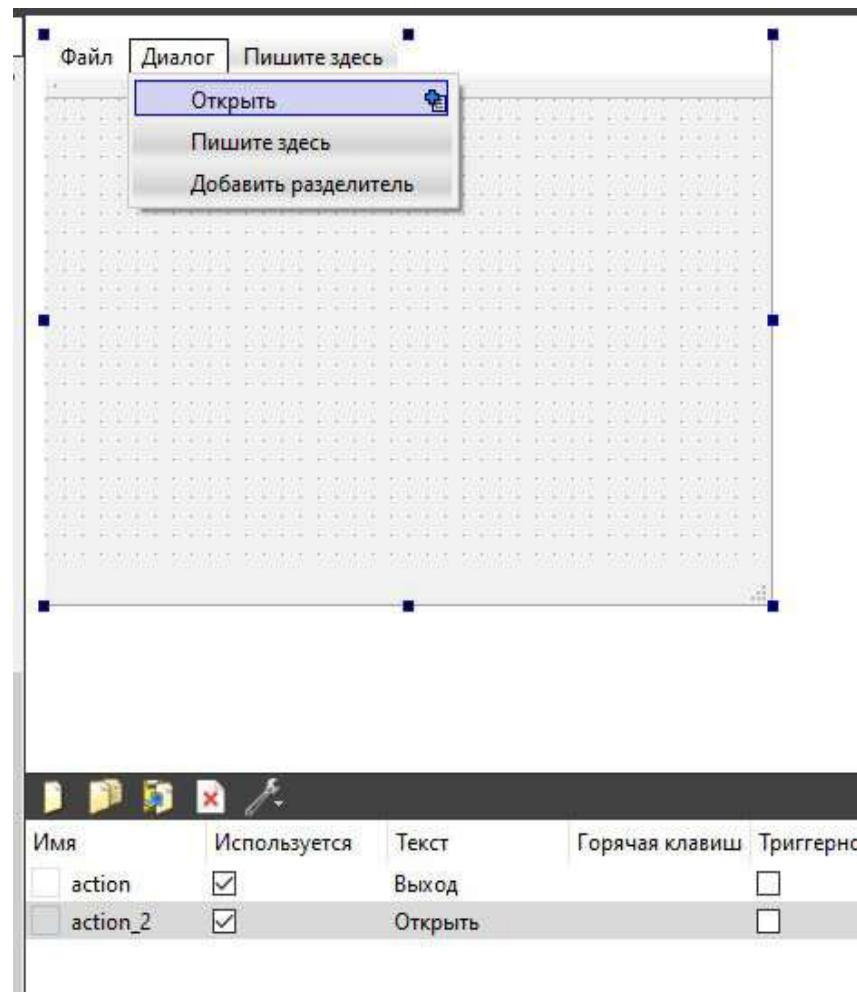
Для этого нужно выделить эту кнопку, затем переместить указатель мыши левой кнопкой в свободную область формы и отпустить, когда увидите красную большую стрелку.

Затем в окошке сигналов и слотов выделить опцию «Показать все сигналы и слоты» и соединить сигнал **clicked** со слотом **close()**.

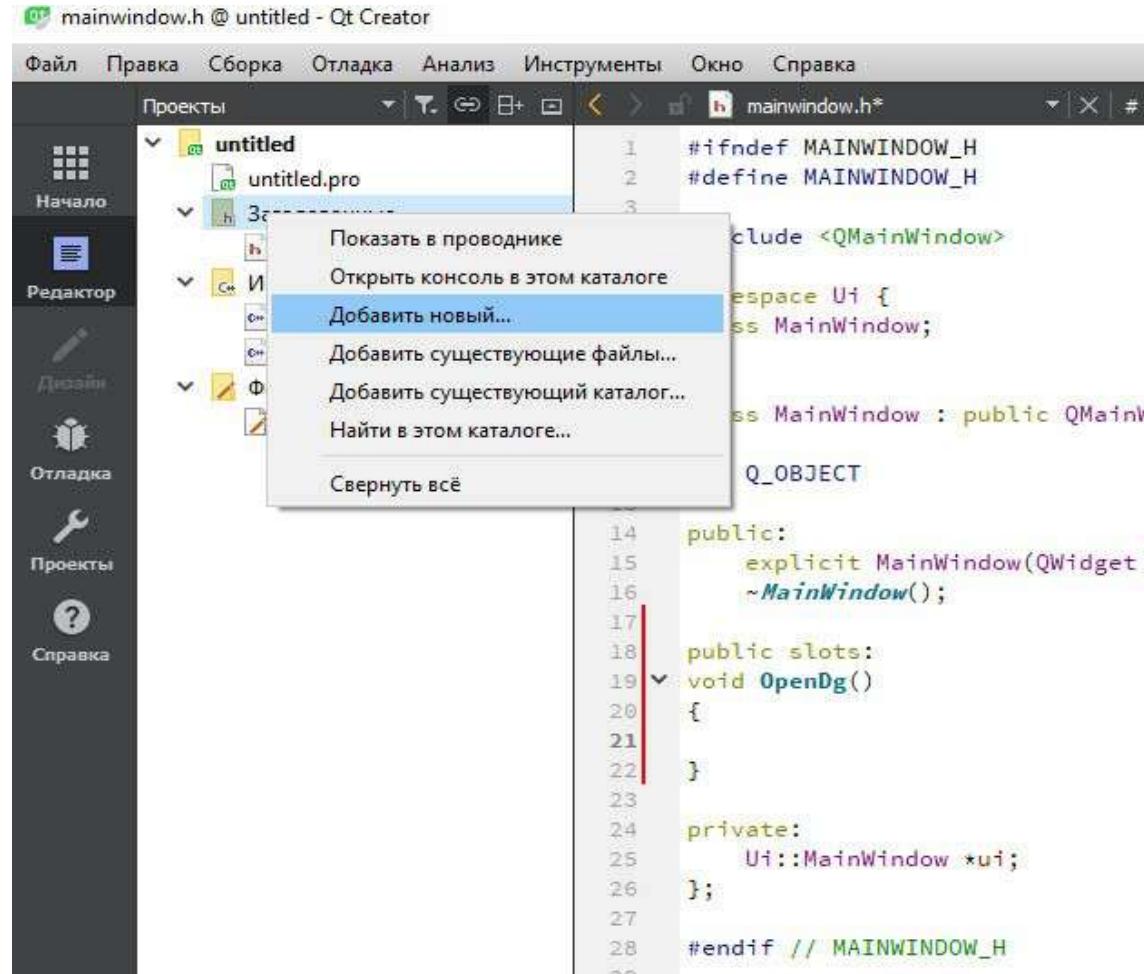
Кнопка **Reset** должна обнулять счетчик.

Для этого по кнопке нужно щелкнуть правой кнопкой мыши и выбрать «Перейти к слоту», затем слева выбрать сигнал **clicked**, нажать **OK** и в открывшемся обработчике написать: **ui->Num->display(0);**

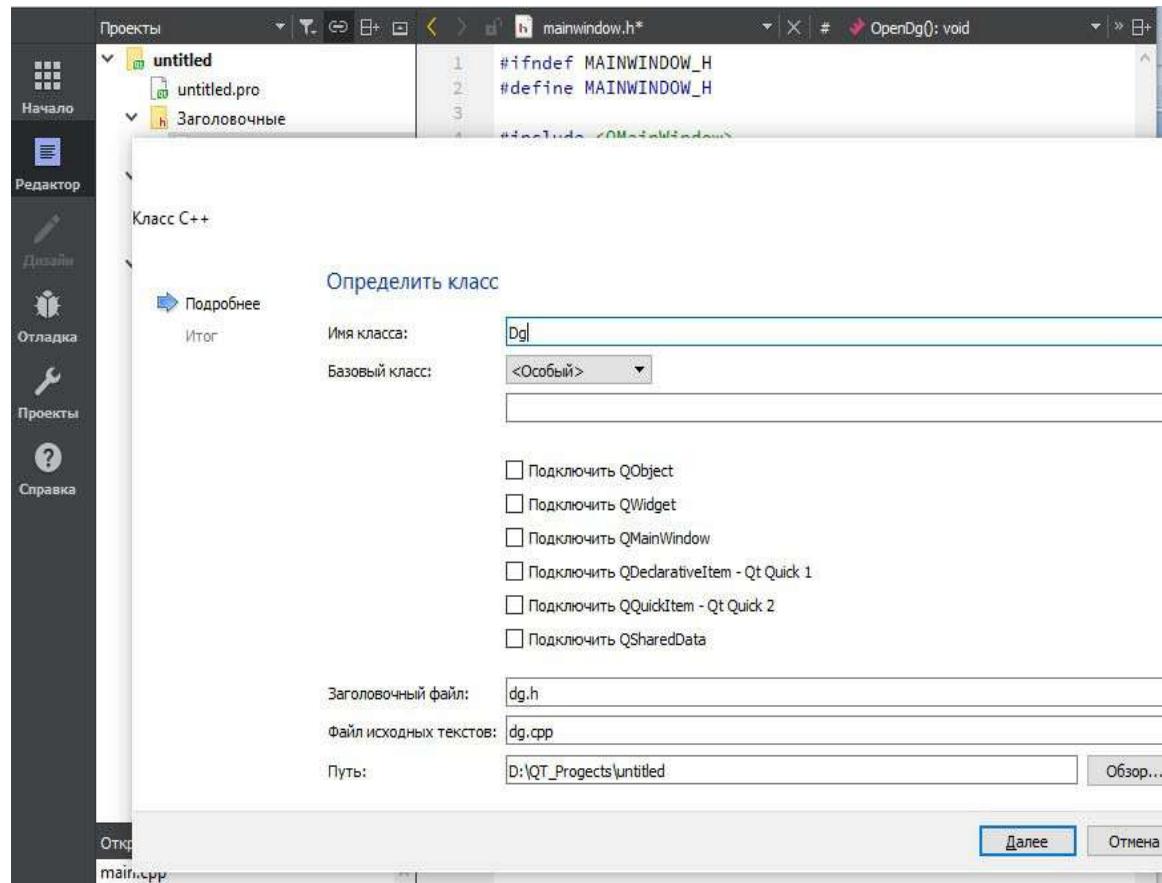
Создание собственного диалогового окна



Создание собственного диалогового окна



Создание собственного диалогового окна



Dg.h

```
1 ifndef DG_H
2 define DG_H
3 include <qdialog.h>
4
5 class QLabel;
6 class QLineEdit;
7 class QPushButton;
8
9 class Dg: public QDialog
10 {
11     Q_OBJECT
12 public:
13     Dg(QWidget* parent=0);
14
15 private slots:
16     void Wrt();
17     void enableWr(const QString &text);
18
19 private:
20     QLabel *label;
21     QLineEdit *lineEdit;
22     QPushButton *Wr_But;
23     QPushButton *closeButton;
24 };
25 #endif // DG_H
```

Диалог будет состоять из редактируемого текстового поля, поясняющего текста к нему и двух кнопок.

По одной кнопке будет происходить запись текста в файл, а по другой - диалог будет закрываться.

При этом диалог будет вызываться из основного меню главного окна.

Dg.cpp

```
1 #include <qlabel.h>
2 #include <qlayout.h>
3 #include <qlineedit.h>
4 #include <QPushButton.h>
5 #include "dg.h"
6 #include <qfile.h>
7 #include <qtextrstream.h>
8 #include <qmessagebox.h>
9
10
11 Dg::Dg(QWidget* parent): QDialog(parent)
12 {
13     this->setWindowTitle("Текстовый редактор");
14     label = new QLabel("Текст:", this);
15     lineEdit = new QLineEdit(this);
16     Wr_But = new QPushButton("записать", this);
17     Wr_But->setDefault(true);
18     Wr_But->setEnabled(false);
19     closeButton = new QPushButton("закрыть",this);
20
21     connect(lineEdit, SIGNAL(textChanged(const QString &)), this, SLOT(enableWr(const QString &)));
22     connect(Wr_But, SIGNAL(clicked()), this, SLOT(Wrt()));
23     connect(closeButton, SIGNAL(clicked()), this, SLOT(close()));
24 }
```

Dg.cpp

```
24  
25     QHBoxLayout *topLeft= new QHBoxLayout;  
26     topLeft->addWidget(label);  
27     topLeft->addWidget(lineEdit);  
28  
29     QVBoxLayout *left= new QVBoxLayout;  
30     left->addLayout(topLeft);  
31  
32     QVBoxLayout *right= new QVBoxLayout;  
33     right->addWidget(Wr_But);  
34     right->addWidget(closeButton);  
35     QHBoxLayout *mainLayout= new QHBoxLayout(this);  
36     mainLayout->setMargin(28);  
37     mainLayout->addLayout(left);  
38     mainLayout->addLayout(right);  
39 }
```

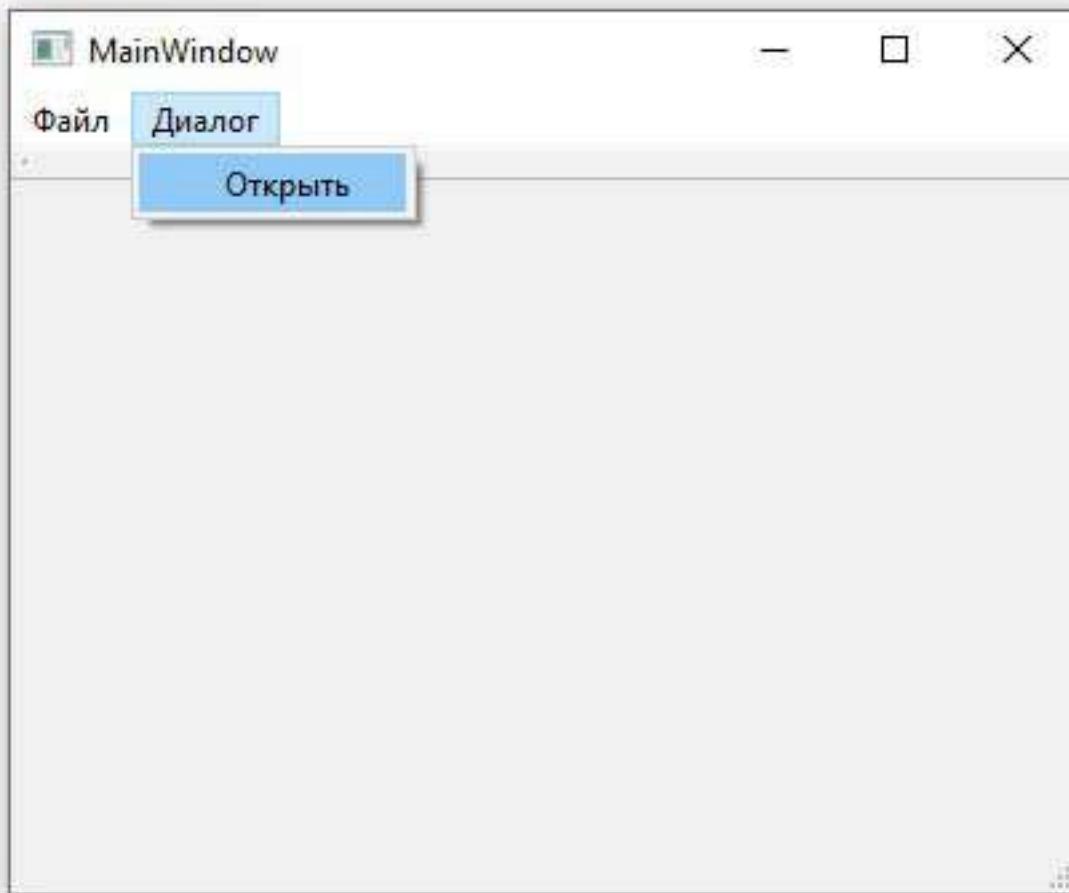
Dg.cpp

```
40
41 void Dg::Wrt()
42 {
43     //записать в файл
44     QFile file1("file1.txt");
45     file1.open(QIODevice::WriteOnly);
46     QTextStream stream(&file1);
47     stream << lineEdit->text();
48     file1.close();
49     QMessageBox::about(0, "Сообщение", "Текст записан");
50 }
51
52 void Dg:: enableWr(const QString &text)
53 {
54     Wr_But->setEnabled(!text.isEmpty());
55 }
```

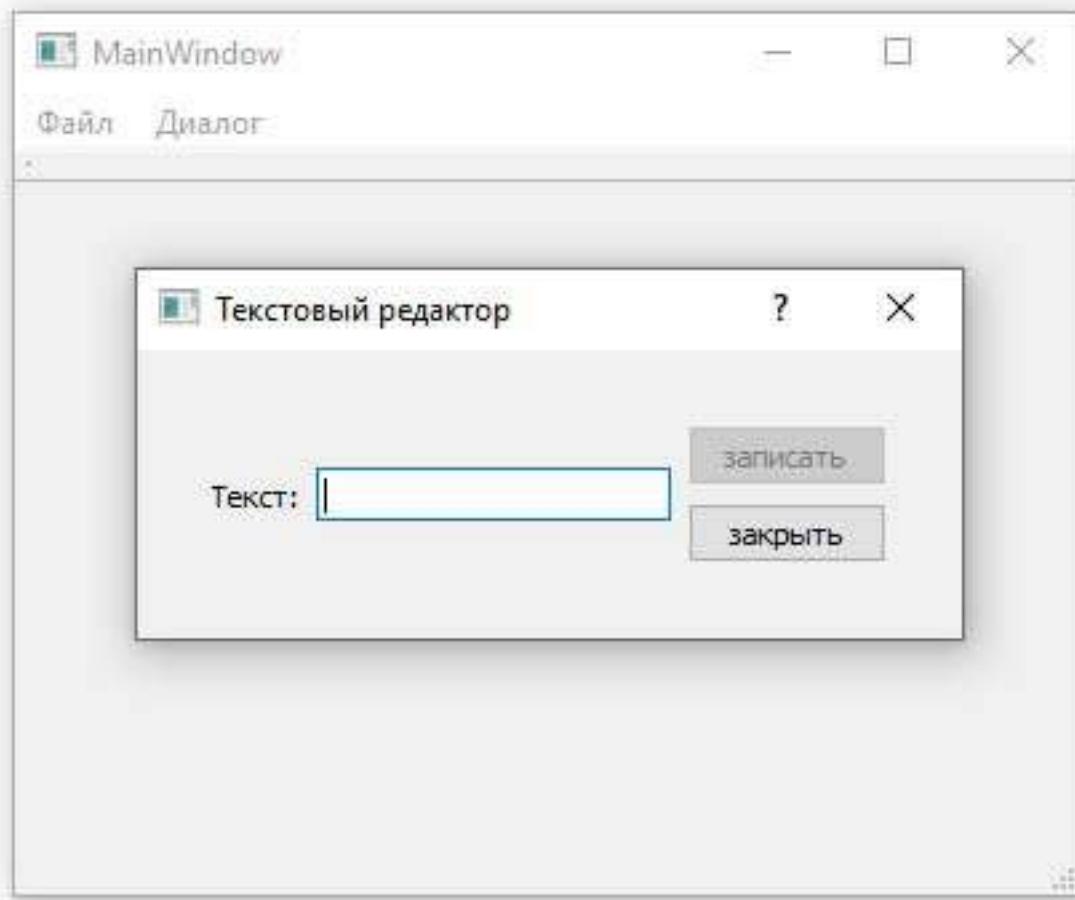
MainWindow.cpp

```
1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3 #include "dg.h"
4
5 void MainWindow::OpenDg()
6 {
7     Dg *dialog = new Dg;
8     dialog->setModal(true);
9     dialog->show();
10 }
11
12 MainWindow::MainWindow(QWidget *parent) :
13     QMainWindow(parent),
14     ui(new Ui::MainWindow)
15 {
16     ui->setupUi(this);
17     connect( ui->action_2, SIGNAL(triggered()), SLOT(OpenDg()) );
18 }
19
20 ~MainWindow()
21 {
22     delete ui;
23 }
```

Запуск приложения



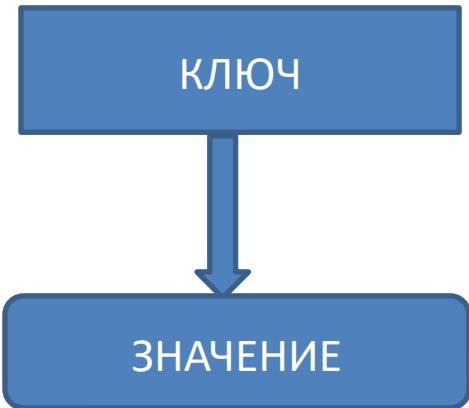
Запуск приложения



Человеко-машинное взаимодействие

Лекция 7

Мерзлякова Екатерина Юрьевна
к.т.н. доцент ПМИК



ОС Windows: HKEY_LOCAL_MACHINE\Software
HKEY_CURRENT_USER\Software

Linux: \$HOME/.qt
\$QTDIR/etc.

```
QSettings settings("PMK","Lab");
```

```
QCoreApplication::setOrganisationName("BHV") ;  
QCoreApplication:: setApplicationName("MyProgram");
```

setValue()

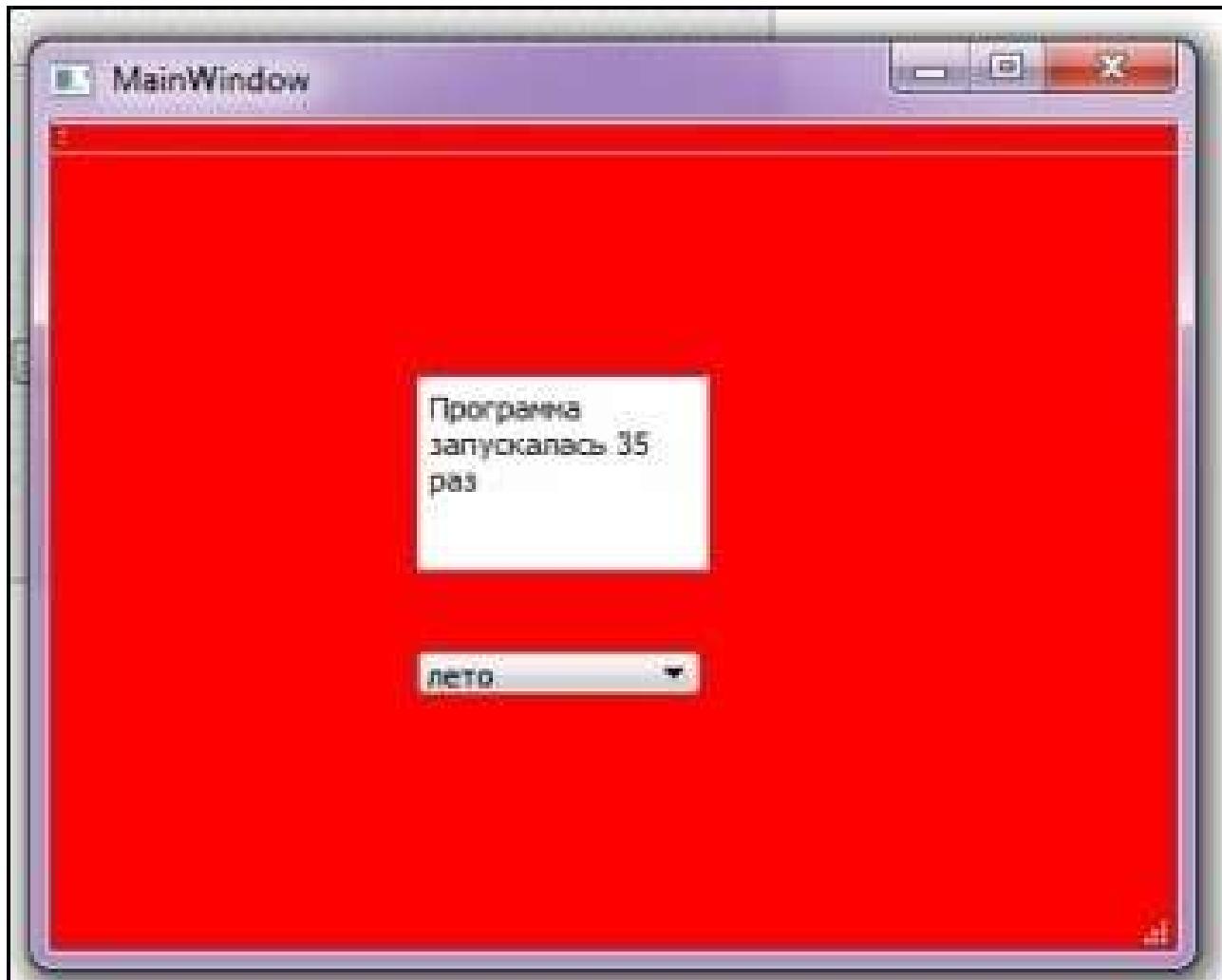
bool, double, int, QString, QRect, QImage, ...

```
settings.setValue("key1","val1"); //строка
settings.setValue("key2",3543); //число
settings.setValue("/Settings/key3",true);
```

value() → **QVariant**

```
QString str = settings.value("key1","").toString();
Int K = settings.value("key2",0).toInt();
Bool b = settings.value("/Settings/key3",false).toBool();
```

```
settings.remove("val1");
```



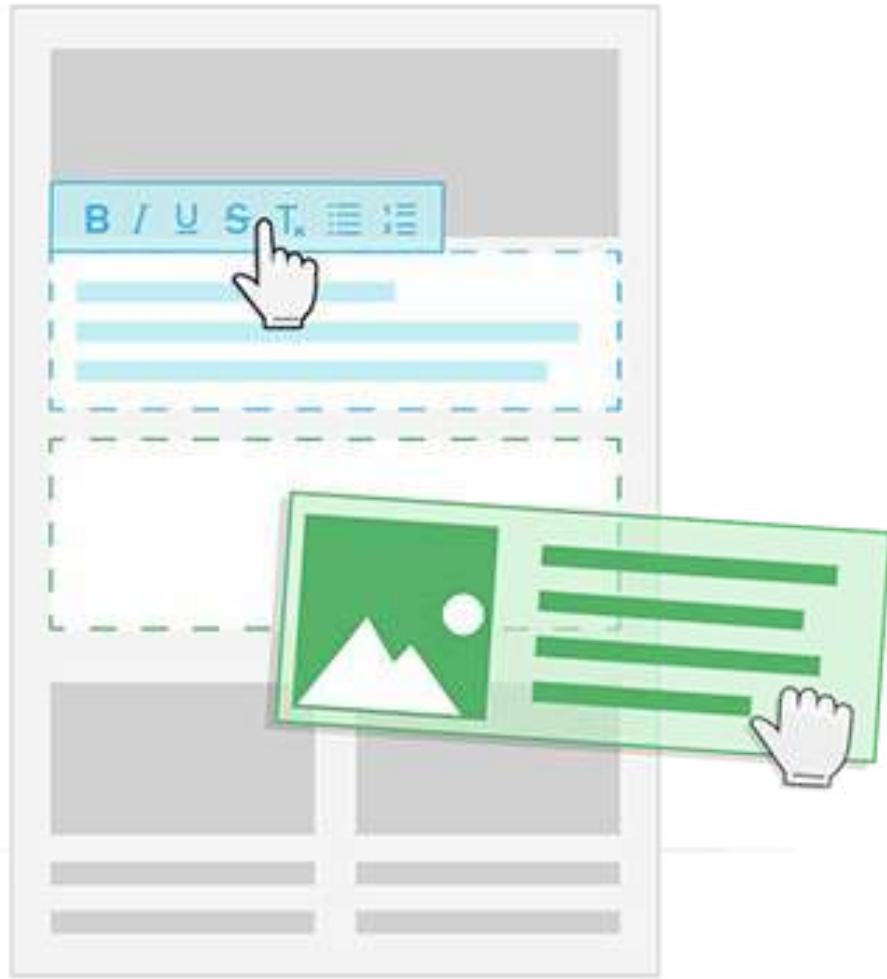
```
public:  
    explicit MainWindow(QWidget *parent = 0);  
    ~MainWindow();  
    void writeSet();  
    void readSet();  
  
private slots:  
    void on_comboBox_currentIndexChanged(int index);  
  
private:  
    Ui::MainWindow *ui;  
    QSettings My_set;  
    int K;  
    QString path="C:\\Qt\\MyProjects\\1\\untitled\\";
```

```
10
11 MainWindow::MainWindow(QWidget *parent) :
12     QMainWindow(parent),
13     ui(new Ui::MainWindow), My_set("pmk","lab")
14 {
15     ui->setupUi(this);
16     readSet();
17 }
```

```
24 void MainWindow::readSet() {
25     My_set.beginGroup("/Set");
26     int St = My_set.value("/Color",3).toInt();
27     K = My_set.value("/Count",1).toInt();
28     K++;
29     QString st = QString().setNum(K);
30     ui->textEdit->setText("Программа запускалась " + st + " раз");
31     ui->comboBox->setcurrentIndex(St);
32     My_set.endGroup();
33 }
34 }
```

```
2 // MainWindow.cpp
3
4 #include "mainwindow.h"
5 #include "ui_mainwindow.h"
6
7 #include <QApplication>
8 #include <QFile>
9 #include <QSettings>
10
11 #include <QColor>
12
13 #include <QComboBox>
14
15 #include <QSet>
16
17 #include <QTextStream>
18
19
20 // MainWindow::MainWindow()
21 // ~MainWindow()
22 // writeSet()
23 // delete ui;
24
25 // void MainWindow::writeSet()
26 // My_set.beginGroup("/Set");
27 // My_set.setValue("/Color",ui->comboBox->currentIndex());
28 // My_set.setValue("/Count",K);
29 // My_set.endGroup();
30
31 }
```

```
13  
46 void MainWindow::on_comboBox_currentIndexChanged(int index)  
47 {  
48  
49     QString S = path+QString().setNum(index)+".qss";  
50     QFile file(S);  
51     file.open(QFile::ReadOnly);  
52     QString strCSS = QLatin1String(file.readAll());  
53     setStyleSheet(strCSS);  
54  
55  
56 }
```





Ctrl + C + Ctrl + V

QClipboard



dataChanged()

```
connect(qApp->clipboard(),
SIGNAL(dataChanged()), pwgt,
SLOT(slotDataControl()) );
```

Помещение в буфер:

`setText(), setPixmap(), setImage(),
setMimeData() (любой тип).`

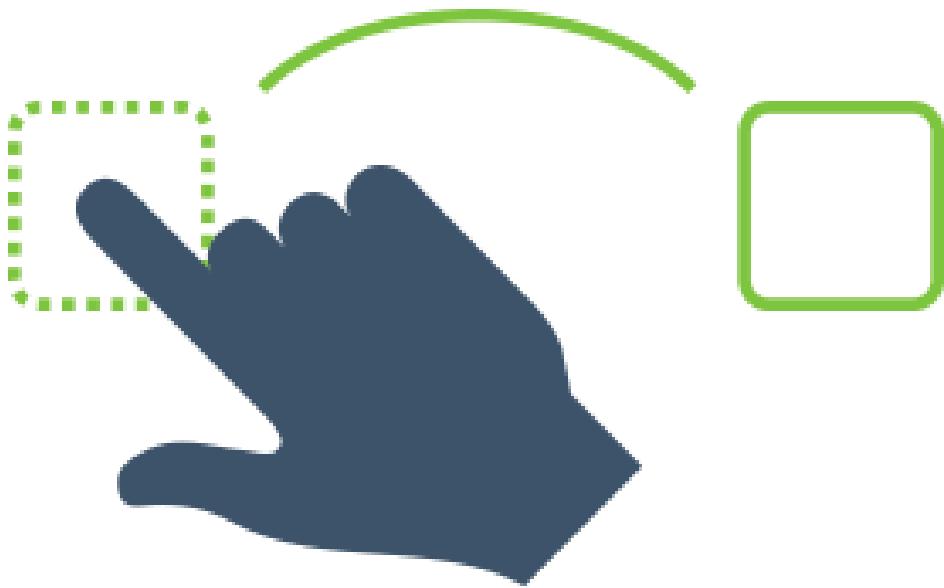
Например:

```
QClipboard* pcb = QApplication::clipboard();  
pcb->setText("My Text");
```

Получение данных из буфера: text(), image(), pixmap(), mimeData()

Например:

```
QClipboard* pcb = QApplication::clipboard();
QString      str = pcb->text();
if (!str.isNull()) {
    qDebug() << "Clipboard Text: " << str;
}
```



Перетаскивание - **QDrag**

Размещение данных различных типов при перетаскивании
– **QMimeType**

MIME - *Multipurpose Internet Mail Extension* (многоцелевые расширения почты Интернета)

MIME-тип	Описание
application/*	данные собственного типа, которые могут интерпретироваться только лишь вашим приложением
audio/*	Звуковые данные, например: audio/wav
image/*	Растровое изображение, например: image/png
model/*	Данные моделей, зачастую трехмерные, например model/vrml
text/*	Текст, например: text/plain , text/html
video/*	Видеоданные, например, video/mpeg

QMimeData

Цветовых значений	setColorData()
Растровых изображений	setImageData()
Текстовой информации	setText()
Гипертекстовой информации в формате HTML	setHtml()
Списков (ссылок) URL Этот метод часто применяется для перетаскивания файлов.	setUrls().

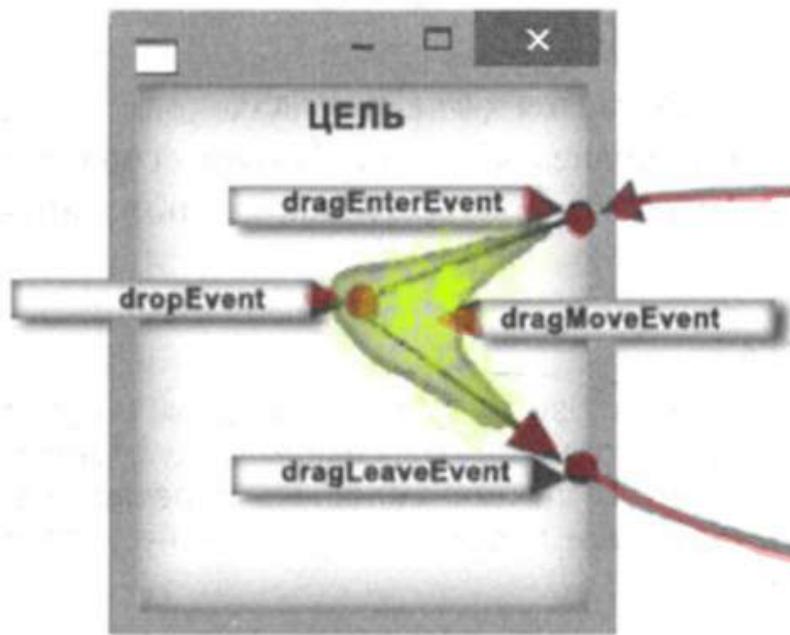
QMimeData

setData(,)

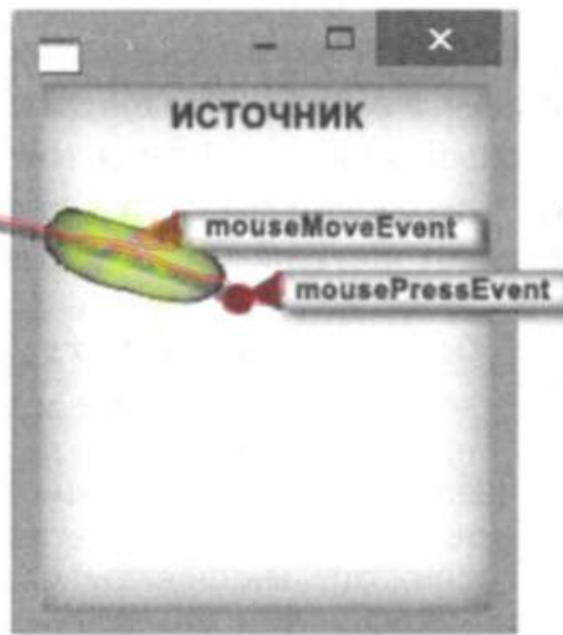
строка, характеризующая тип данных
сами данные в объекте класса QByteArray

перезаписать методы
formats() и retrieveData()

ОБЪЕКТ 1



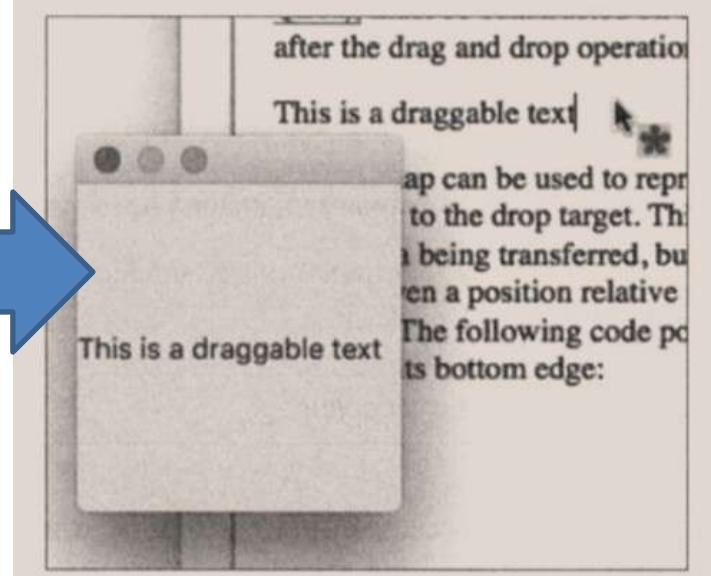
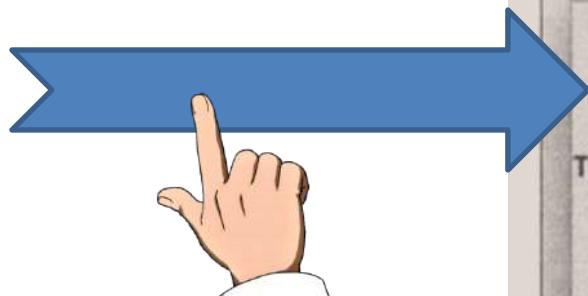
ОБЪЕКТ 2



mousePressEvent ()

позиция указателя мыши

mouseMoveEvent ()



(файл Drag.h)

```
#pragma once

#include <QtWidgets>

// =====
class Drag : public QLabel {
Q_OBJECT
private:
    QPoint m_ptDragPos;

    void startDrag()
    {
        QMimeData* pMimeData = new QMimeData;
        pMimeData->setText(text());
    }
}
```

```
pMimeData->setText(text());  
  
QDrag* pDrag = new QDrag(this);  
pDrag->setMimeData(pMimeData);  
pDrag->setPixmap(QPixmap(":/img1.png"));  
pDrag->exec();  
}  
}
```

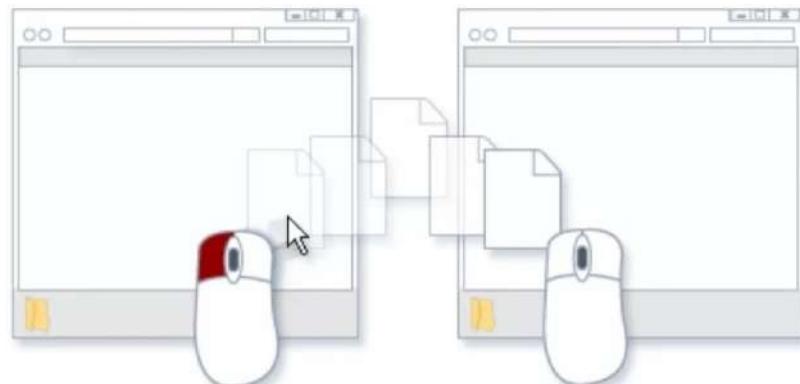
для копирования *Qt::CopyAction*

для перемещения *Qt::MoveAction*

для создания ссылки *Qt::LinkAction*

по умолчанию *Qt::MoveAction*

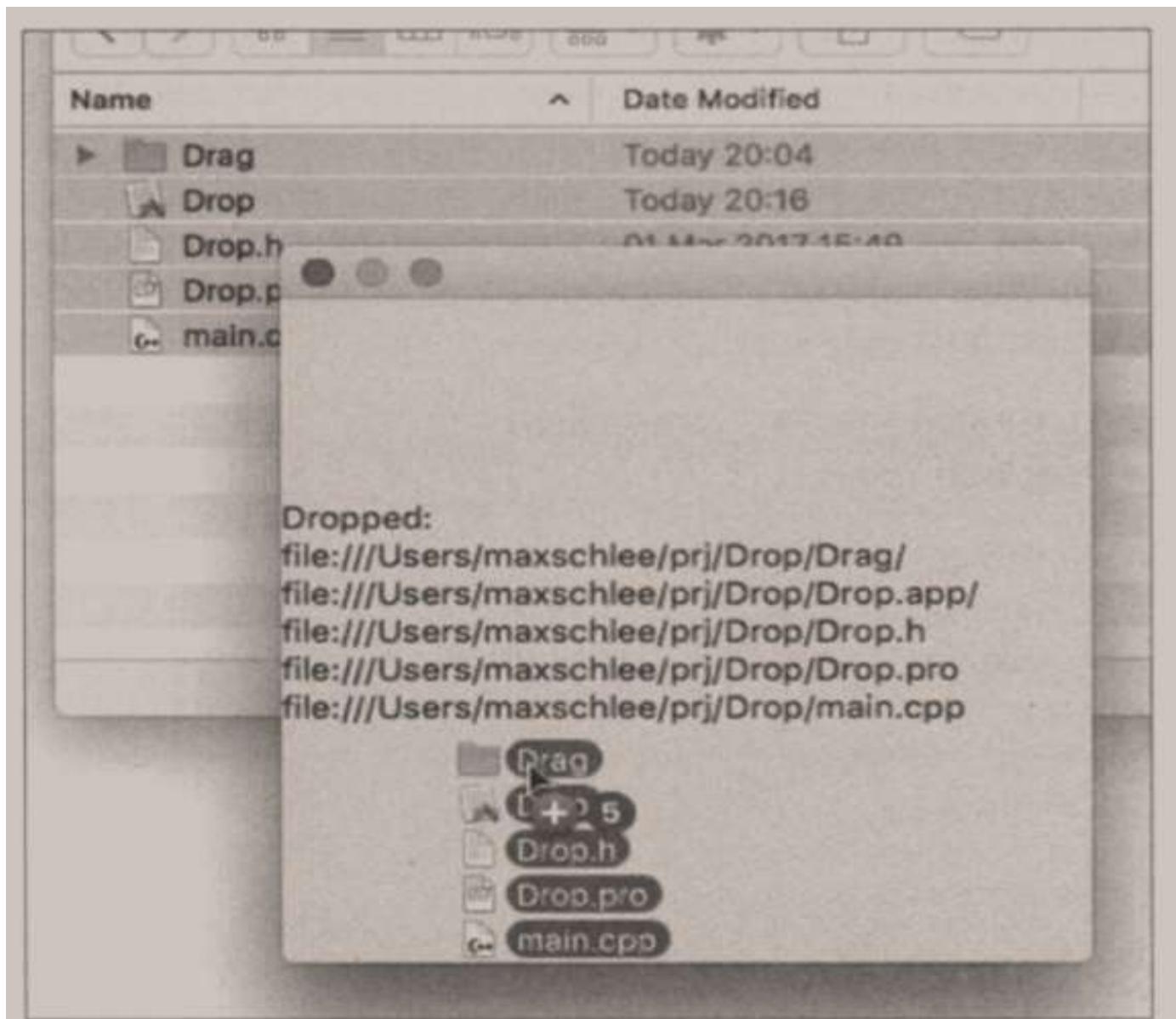
```
protected:  
    virtual void mousePressEvent(QMouseEvent* pe)  
    {  
        if (pe->button() == Qt::LeftButton) {  
            m_ptDragPos = pe->pos();  
        }  
        QWidget::mousePressEvent(pe);  
    }  
  
public:  
    Drag(QWidget* pwgt = 0) : QLabel("This is a draggable text", pwgt)  
    {}  
};
```



```
virtual void mouseMoveEvent (QMouseEvent* pe)
{
    if (pe->buttons() & Qt::LeftButton) {
        int distance = (pe->pos() - m_ptDragPos).manhattanLength();
        if (distance > QApplication::startDragDistance()) {
            startDrag();
        }
    }
    QWidget::mouseMoveEvent(pe);
}

public:
    Drag(QWidget* pwgt = 0) : QLabel("This is a draggable text", pwgt)
    {
    }
};

};
```



```
public:  
    Drop(QWidget* pwgt = 0) : QLabel("Drop Area", pwgt  
    {  
        setAcceptDrops(true);  
    }  
};
```

```
#pragma once

#include <QtWidgets>

// =====
class Drop : public QLabel {
Q_OBJECT

protected:
    virtual void dragEnterEvent(QDragEnterEvent* pe)
    {
        if (pe->mimeData()->hasFormat("text/uri-list")) {
            pe->acceptProposedAction();
        }
    }
}
```

```
virtual void dropEvent(QDropEvent* pe)
{
    QList<QUrl> urlList = pe->mimeType()->urls();
    QString      str;
    foreach(QUrl url, urlList) {
        str += url.toString() + "\n";
    }
    setText("Dropped:\n" + str);
}
```

System Tray

QSystemTrayIcon

setIcon()

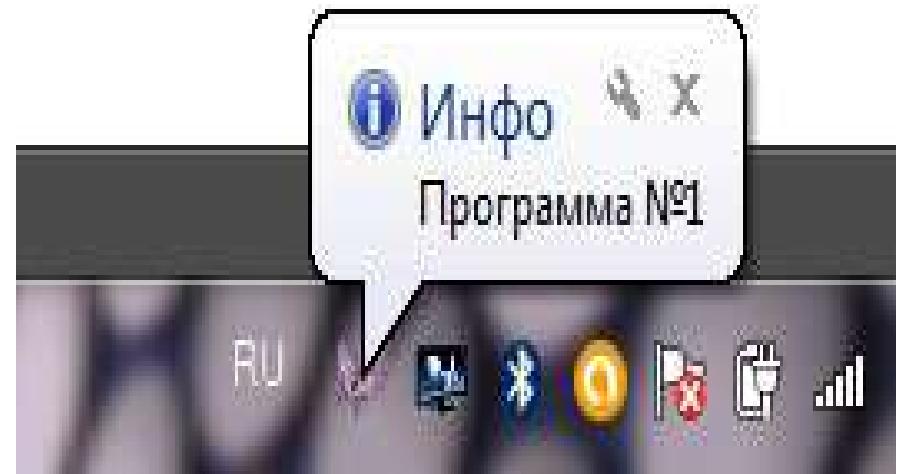
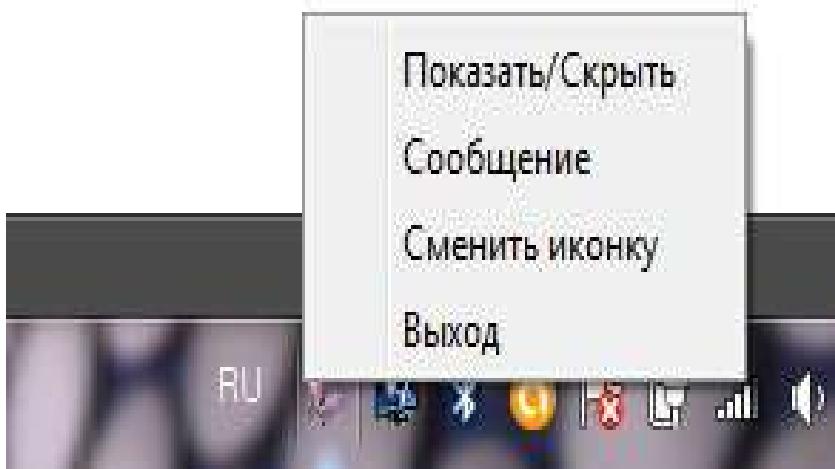
show()

showMessage()

setToolTip()

setContextMenu()

```
QSystemTrayIcon st;
QIcon icon("C:/Qt/MyProjects/2/s.png");
st.setIcon(icon);
st.show();
```



Всплывающая подсказка



```
5
6 int main(int argc, char *argv[])
7 {
8     QApplication a(argc, argv);
9     MainWindow w;
10    QApplication::setQuitOnLastWindowClosed(false);
11    return a.exec();
12 }
```

```
1 #ifndef MAINWINDOW_H
2 #define MAINWINDOW_H
3
4 #include <QMainWindow>
5 #include <QSystemTrayIcon>
6
7
8 // namespace Ui {
9 class MainWindow;
10 //}
11
12 class MainWindow : public QMainWindow
13 {
14     Q_OBJECT
15 private:
16     QSystemTrayIcon* micon;
17     QMenu* men;
18     bool sw;
```

```
1~  
21 protected:  
22     virtual void closeEvent(QCloseEvent*) ;  
23  
24 public:  
25     explicit MainWindow(QWidget *parent = 0);  
26     ~MainWindow();  
27  
28 public slots:  
29     void slotSH();  
30     void slotSM();  
31     void slotCI();  
32  
33 private:  
34     Ui::MainWindow *ui;  
35 };  
36  
37 #endif // MAINWINDOW_H
```

```
1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3
4 #include "QSystemTrayIcon"
5 #include "QAction"
6 #include "QMenu"
7
8 MainWindow::MainWindow(QWidget *parent) :
9     QMainWindow(parent),
10    ui(new Ui::MainWindow)
11 {
12     setWindowTitle("Приложение");
13
14     QAction* ShowHide= new QAction("Показать/Скрыть", this);
15     connect(ShowHide, SIGNAL(triggered()),this, SLOT(slotSH()));
16
17     QAction* ShowMessage= new QAction("Сообщение", this);
18     connect(ShowMessage, SIGNAL(triggered()),this, SLOT(slotSM()));
19
20     QAction* ChIcon= new QAction("Сменить иконку", this);
21     connect(ChIcon, SIGNAL(triggered()),this, SLOT(slotCI()));
22
23     QAction* Ext= new QAction("Выход", this);
24     connect(Ext, SIGNAL(triggered()),qApp, SLOT(quit()));
25 }
```

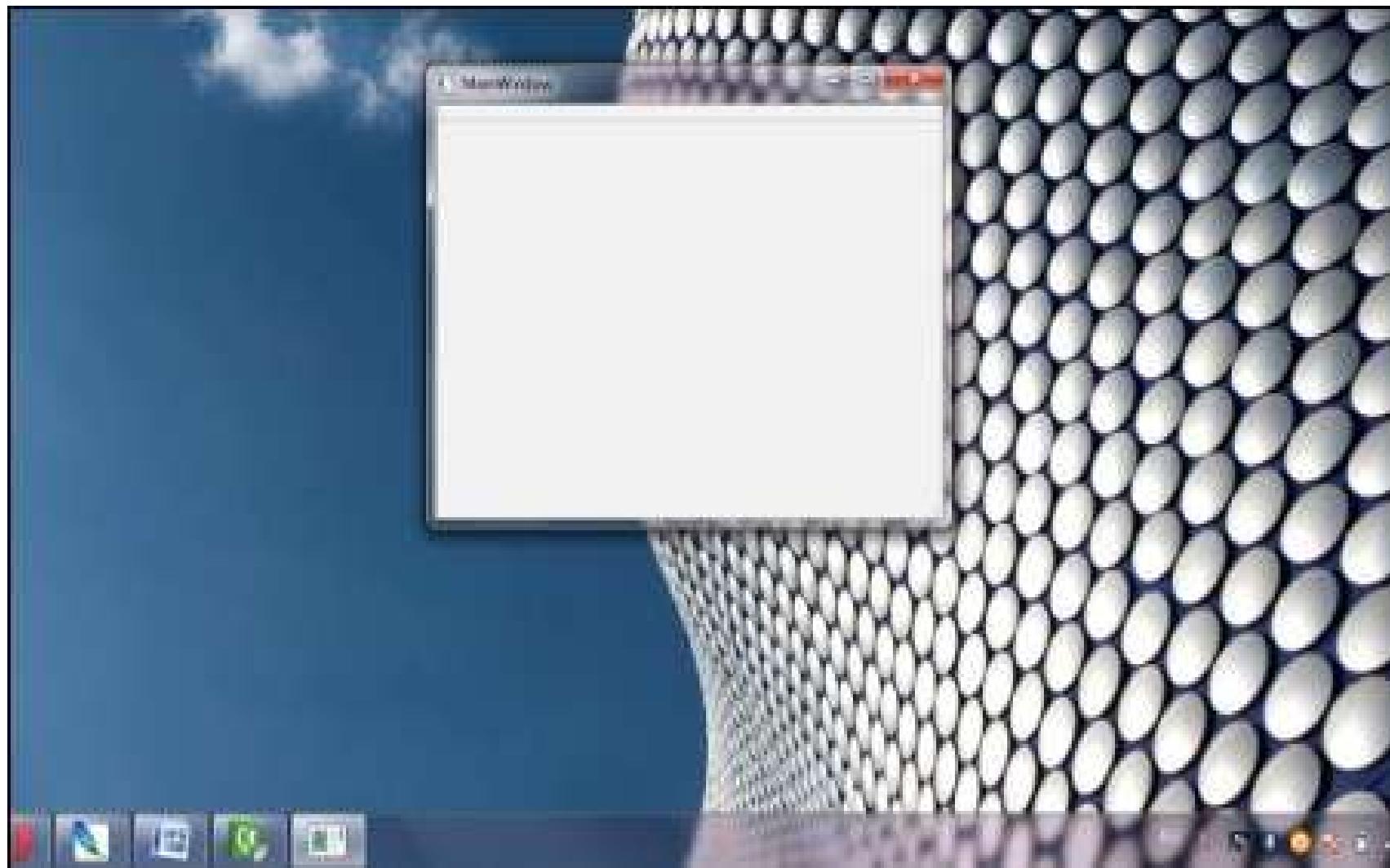
```
25
26     men= new QMenu(this);
27     men->addAction(ShowHide);
28     men->addAction(ShowMessage);
29     men->addAction(ChIcon);
30     men->addAction(Ext);
31
32
33     micon= new QSystemTrayIcon(this);
34     micon->setContextMenu(men);
35     micon->setToolTip ("Всплывающая подсказка");
36
37     slotCI();
38     micon->show();
39
40     ui->setupUi(this);
41 }
```

```
18 4 void MainWindow::closeEvent(QCloseEvent *parent)
19 {
20     if(micon->isVisible()) {hide();}
21 }
22 }
```

```
1 53  █ void MainWindow::slotSH()
2 54  {
3 55     setVisible(!isVisible());
4 56 }
```

```
8     void MainWindow::slotSM()
9     {
10         micon->showMessage("Инфо", "Программа №1", QSystemTrayIcon::Information, 3000);
11     }
12 }
```

```
63     void MainWindow::slotCI()
64     {
65         sw=!sw;
66         QString Pix = sw ? "C:\\Qt\\MyProjects\\3\\1.ico"
67                         : "C:\\Qt\\MyProjects\\3\\2.ico";
68
69         QIcon icon(Pix);
70         micon->setIcon(icon);
71     }
72 }
```





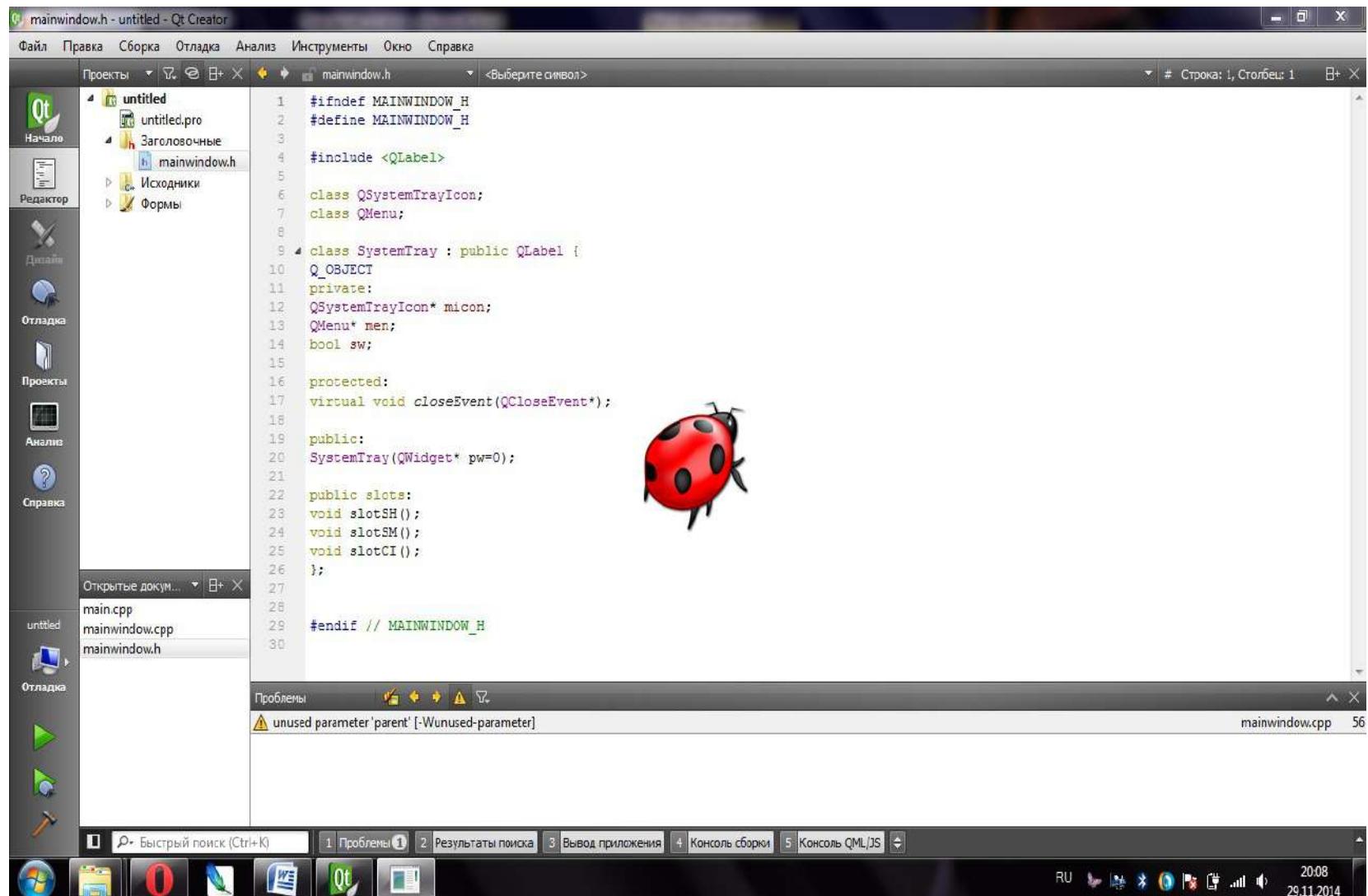
```
4 #include <QLabel>
5
6 class QSystemTrayIcon;
7 class QMenu;
8
9 class SystemTray : public QLabel {
10 Q_OBJECT
11 private:
12 QSystemTrayIcon* micon;
13 QMenu* men;
14 bool sw;
15
16 protected:
17 virtual void closeEvent(QCloseEvent*) ;
18
19 public:
20 SystemTray(QWidget* pw=0) ;
21
22 public slots:
23 void slotSH();
```

```
4 //include "QSystemTrayIcon"
5 #include "QAction"
6 #include "QMenu"
7 #include <QBitmap>
8 #include <QPixmap>
9
10 // SystemTray::SystemTray(QWidget* pw) : QLabel("Приложение", pw), sw(false)
11 {
12     setWindowTitle("Приложение");
13     QPixmap pix("C:\\Qt\\MyProjects\\3\\2.ico");
14     setPixmap(pix);
15     setMask(pix.mask());
16
17     QAction* ShowHide= new QAction("&Show/Hide", this);
```

```
1 #include "mainwindow.h"
2 #include <QApplication>
3 #include <QtGui>
4 #include <QSystemTrayIcon>
5
6 int main(int argc, char *argv[])
7 {
8     QApplication a(argc, argv);
9     SystemTray st;
10    QApplication::setQuitOnLastWindowClosed(false);
11    return a.exec();
12 }
```

QDesktopWidget

```
61 // void SystemTray::slotSH()
62 {
63     setVisible(!isVisible());
64     QDesktopWidget desktop;
65     QRect rect = desktop.availableGeometry(desktop.primaryScreen());
66     QPoint center = rect.center();
67     center.setX(center.x() - (this->width()/2));
68     center.setY(center.y() - (this->height()/2));
69     this->move(center);
70 }
71 }
```



Человеко-машинное взаимодействие

Лекция 4

Мерзлякова Екатерина Юрьевна
к.т.н. доцент ПМИК

Работа с файлами, каталогами, потоками

QDir – для работы с каталогами

QFile – для работы с файлами

QFileInfo – для получения файловой информации

QIODevice – абстрактный класс для ввода/вывода

QBuffer – для эмуляции файлов в памяти компьютера

Класс QIODevice

Класс QIODevice – это абстрактный класс, обобщающий устройство **ввода/вывода**, который содержит виртуальные методы для открытия и закрытия устройства ввода/вывода, а также для чтения и записи блоков данных или отдельных символов.

Класс QIODevice

QFile – класс для проведения операций с файлами

QBuffer – класс буфера, который позволяет записывать и считывать данные в массив **QByteArray**.

QAbstractSocket – базовый класс для сетевой коммуникации посредством сокетов.

QProcess – класс, предоставляющий возможность запуска процессов с дополнительными аргументами и позволяющий обмениваться информацией с этими процессами посредством методов, определенных в **QIODevice**.

Класс QIODevice

QIODevice::NotOpen – устройство не открыто

QIODevice::ReadOnly – открытие устройства только для чтения данных

QIODevice::WriteOnly – открытие только для записи данных

QIODevice::ReadWrite – для чтения и записи

QIODevice::Append – для добавления данных в конец файла

Класс QIODevice

QIODevice::Unbuffered – открытие для непосредственного доступа к данным без промежуточного буферизирования

QIODevice::Truncate – все данные устройства, по возможности, должны быть удалены при открытии

QIODevice::Text – текстовый файл (символы конца строки преобразуются в соответствие с текущей платформой). В Windows используется комбинация «\r\n», а в Mac и Unix – «\r»

Методы класса QIODevice

- openMode()
- read() и write()
- readAll()
- методами readLine() и getChar()
- seek()
- pos()

Класс QFile

```
QFile file;  
file.setName("file.dat");
```

QIODevice::isOpen()

close()

QFile::flush()

remove()

Класс QFile

```
QFile file1("file1.dat");
QFile file2("file2.dat");
if (file2.exists()){
    //файл уже существует. Перезаписать?
}
if(!file1.open(QIODevice::ReadOnly)){
    qDebug() << "Ошибка открытия для чтения";
}
if(!file2.open(QIODevice::WriteOnly)){
    qDebug() << "Ошибка открытия для записи";
}
```

QFile::exists()
QIODevice::read()
QIODevice::write()

Класс QFile

```
QFile::exists()  
QIODevice::read()  
QIODevice::write()  
  
{  
    char a[1024];  
    while(!file1.atEnd()) {  
        int nBlockSize = file1.read(a, sizeof(a));  
        file2.write(a, nBlockSize);  
    }  
    file1.close();  
    file2.close();  
}
```

Класс QFile чтение и запись файла

QIODevice::write()
IODevice::readAll()

```
QFile file1("file1.dat");
QFile file2("file2.dat");
if (file2.exists()){
    //файл уже существует. Перезаписать?
}
if(!file1.open(QIODevice::ReadOnly)){
    qDebug() << "Ошибка открытия для чтения";
```

Класс QFile чтение и запись файла

```
QIODevice::write()  
IODevice::readAll()
```

```
if(!file2.open(QIODevice::WriteOnly)){  
    qDebug() << "Ошибка открытия для записи";  
}  
QByteArray a = file1.readAll();  
file2.write(a);  
file1.close();  
file2.close();  
}
```

Класс QBuffer

Данный класс унаследован от класса QIODevice и представляет собой **эмуляцию** файлов в памяти компьютера.

- open()
- close()
- write()
- read()

Класс QTemporaryFile

```
6 int main(int argc, char *argv[])
7 {
8     QApplication a(argc, argv);
9
10    QTemporaryFile file;
11    if (file.open()) {
12        qDebug() << file.fileName();
13    }
14    file.close();
15    return a.exec();
16 }
17
```

Вывод приложения



untitled



Запускается C:\Qt\MyProjects\lekc2-2\build-untitled-Desktop
"C:/Users/Домовой/AppData/Local/Temp/untitled.Hp6304"

Работа с каталогами. Класс QDir

ОС Windows:	ОС UNIX:
C:\Windows\System	/usr/bin

QDir::current() — возвращает путь текущего каталога приложения;

QDir::root () — возвращает корневой каталог;

QDir::drives () — указатель на объект класса ***QFileInfo*** , содержащий список с узловыми каталогами. Для Windows это будут C:\, D:\ и т. д.;

QDir::home () — возвращает персональный каталог пользователя.

Работа с каталогами. Класс QDir

QApplication::applicationDirPath()

либо *QApplication::applicationFilePath()*

exists() – существование каталога

cd() - перемещение

cdUp() - аналогично

cd("..") = *cdUp()*

makeAbsolute() – относительный путь в абсолютный

mkdir() – создание каталога

Работа с каталогами. Класс QDir

rename() – переименование

В этот метод первым параметром нужно передать старый путь, а вторым — новый.

методом *rmdir()* – удаление пустых каталогов

Класс QDir. Просмотр содержимого каталога

entryList() и *entryInfoList()*

Первый возвращает список имен элементов (**QStringList**), а второй — информационный список (**QFileInfoList**).

Если вам нужно узнать только количество элементов, находящихся в директории, то просто вызовите метод **count()**.

Информация о файлах. Класс **QFileInfo**

Объект класса **QFileInfo** создается передачей в его конструктор пути к файлу, или объекта класса **QFile**.

isFile()

isDir()

true, false.

Путь к файлу. Имя файла

absoluteFilePath()

filePath()

fileName() имя и расширение файла

baseName() - имя

completeSuffix() - расширение

Класс QFileInfo. Дата и время файла

```
//Дата и время создания файла  
fileInfo.created().toString();  
  
//Дата и время последнего изменения файла  
fileInfo.lastModified().toString();  
  
//Дата и время последнего чтения файла  
fileInfo.lastRead().toString();
```

Получение атрибутов файла

isReadable() – возвращает *true*, если из указанного файла можно читать информацию;

isWriteable() – возвращает *true*, если в указанный файл можно записывать информацию;

isHidden() – возвращает *true*, если указанный файл является скрытым;

isExecutable() – возвращает *true*, если указанный файл можно исполнять.

В ОС UNIX это определяется не на основании расширения файла, как принято в DOS и ОС Windows, а из свойств самого файла.

Элементы выбора. Простой список

Элементы выбора представляют собой стандартные элементы графического интерфейса пользователя для отображения, модификации и выбора данных.

Класс **QListWidget** – это виджет списка, в котором пользователь может выбрать один или несколько элементов.

Простой список

`QListWidget::addItem()`

`clone()`

`addItem()`

Есть два варианта этого метода: для текста и
объекта класса `QListWidgetItem`.

Чтобы удалить все элементы, вызывают слот
`clear()`.

Простой список

QStringList → insertItems()

insertItem()

QListWidgetItem → insertItem()

QListWidgetItem::setIcon()

Отличие метода `insertItem()` от метода `addItem()` состоит в том, что он дает возможность явно указать позицию добавляемого элемента.

Простой список

setItemWidget() и itemWidget()

*Чтобы получить указатель на виджет, расположенный в элементе списка, в метод **itemWidget()** передается указатель на объект элемента списка.*

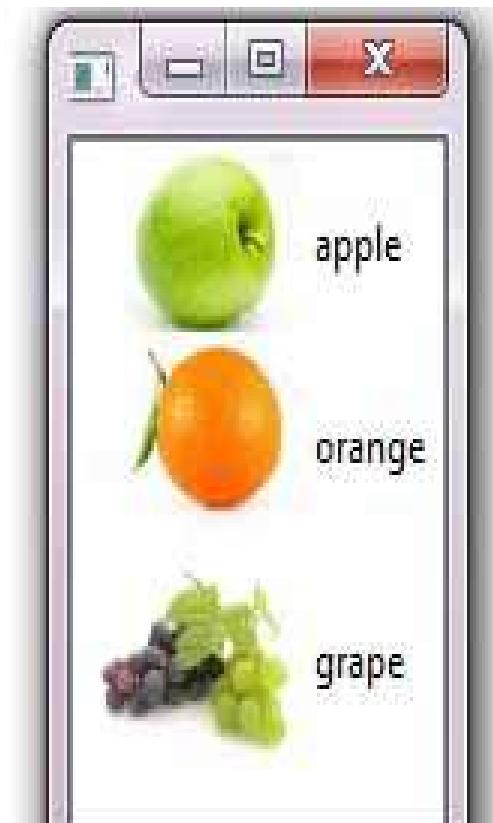
Пример простого списка

Чтобы получить указатель на виджет, расположенный в элементе списка, в метод **itemWidget()** передается указатель на объект элемента списка.

```
#include "mainwindow.h"
#include <QApplication>
#include <QListWidget>

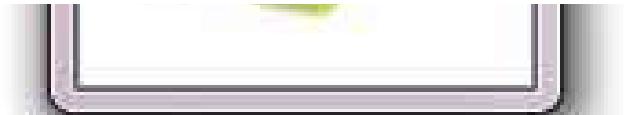
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    QStringList lst;
    QListWidget lwg;
    QListWidgetItem* pItem = 0;

    lwg.setIconSize(QSize(70, 70));
    lst<<"apple"<<"orange"<<"grape";
```



Пример простого списка

```
lwg.setIconSize(QSize(70, 70));
lst<<"apple"<<"orange"<<"grape";
foreach(QString str, lst) {
    pitem = new QListWidgetItem(str, &lwg);
    pitem->setIcon(QIcon ("C:\\Qt\\MyProjects\\"+str+".jpg"));
}
lwg.resize(125, 175);
lwg.show();
return a.exec();
```



Выбор элементов пользователем

QListWidget::currentItem() - указатель на выбранный элемент.
selectedItems() - список выбранных элементов.

Для этого режима выбора нужно установить при помощи метода **setSelectionMode()** значение: **QAbstractItemView::MultiSelection**.

Можно запретить выделение элементов, передав значение:
QAbstractItemView::NoSelection.

А для возможности выделения одного элемента:

QAbstractItemView::SingleSelection.

itemClicked

temDoubleClicked() с параметром **QListWidgetItem***.

После каждого изменения выделения элементов высыпается сигнал **itemSelectionChanged()**.

Изменение элементов пользователем

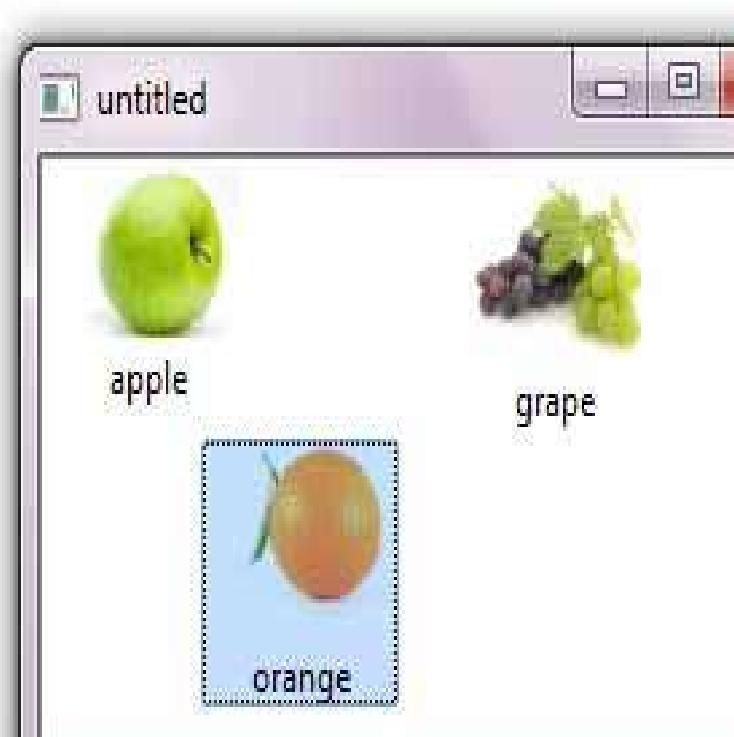
```
pitem->setFlags(Qt::ItemIsEditable |  
Qt::ItemisEnabled) ;
```

виджет **QListWidget** высылает сигналы
itemChanged(QListWidgetItem*) и
itemRenamed(QListWidgetItem*).

Режим пиктограмм

```
#include <iostream>
#include <QApplication>
#include <QListWidget>

main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    QListWidget lwg;
    QListWidgetItem* pitem = 0;
    QStringList lst;
    lwg.setIconSize(QSize(70, 70));
    lwg.setSelectionMode(
        QAbstractItemView::MultiSelection);
    lwg.setViewMode(QListView::IconMode);
```



Режим пиктограмм

```
lwg.setViewMode(QListView::IconMode);
lst<<"apple"<<"orange"<<"grape";
foreach(QString str, lst){
    pitem = new QListWidgetItem(str, &lwg);
    pitem->setIcon(QIcon("C:\\Qt\\MyProjects\\\"+str+".jpg"));
    pitem->setFlags(Qt::ItemIsEnabled|Qt::ItemIsSelectable|
Qt::ItemIsEditable|Qt::ItemIsDragEnabled);
}
lwg.resize(300, 160);
lwg.show();
return a.exec();
```

Сортировка элементов

sortItems()

При передаче в этот метод значения **Qt::AscendingOrder** сортировка будет по возрастанию, а при **Qt::DescendingOrder** – по убыванию.

если нужно отсортировать по дате или по числовому значению, необходимо унаследовать класс элемента **QListWidgetItem** и перезаписать в нем **operator<()**

Иерархические списки

Виджет **QTreeWidget** отображает элементы списка в иерархической форме и поддерживает возможность выбора пользователем одного или нескольких из них. Его часто применяют для показа содержимого дисков и каталогов в случае, когда область отображения не в состоянии разместить все элементы, появляются полосы прокрутки.

- **setItemWidget()**

Иерархические списки

```
QTreeWidgetItem* ptwi = new QTreeWidgetItem(pTreeView);
ptwi->setFlags(ptwiTemp->flags() | Qt::ItemIsUserCheckable);
ptwi->setCheckState(0, Qt::Checked);
ptwi->setText(0, "'Checkable Item');
```

Иерархические списки

```
if (ptwi->checkState(0) == Qt::Checked){  
//This item is checked  
}
```

Иерархические списки

QTreeWidgetItem и предоставляют возможность отображать несколько столбцов с данными.

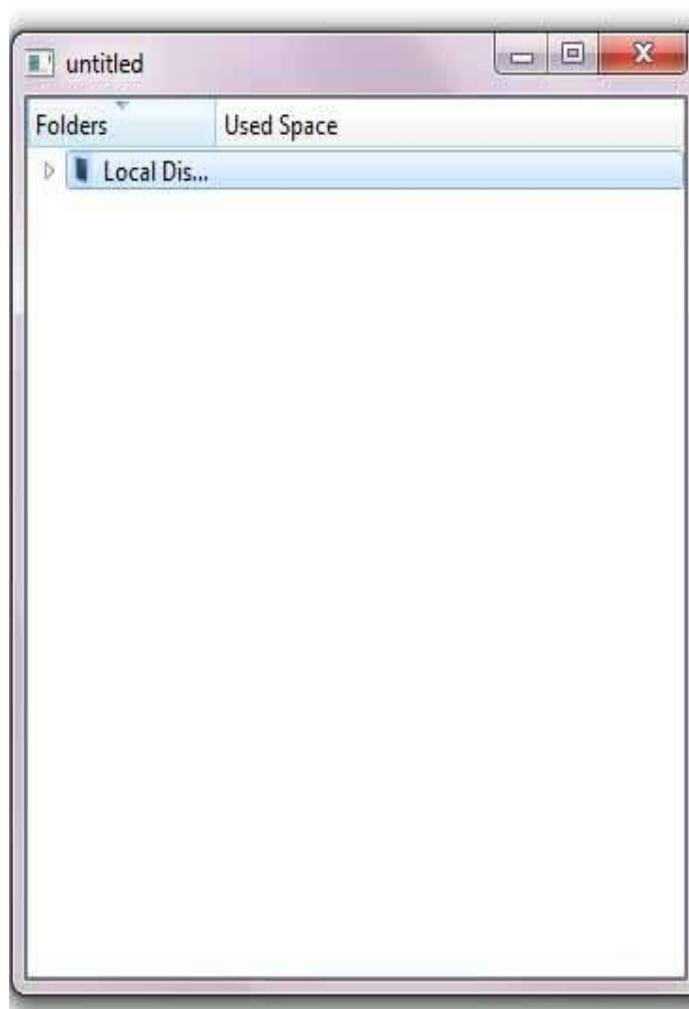
Содержит конструктор копирования и метод **clone()** для создания копий элементов.

addChildren(), **insertChildren()** - можно добавлять сразу несколько элементов.

QTreeWidgetItem::setIcon(),
QtreeWidgetItem::setText().

Первый параметр обоих методов соответствует номеру столбца.

Иерархические списки



Folders	Used Space
Local Disk	
Directory9	9MB
Directory8	8MB
Directory7	7MB
Directory6	6MB
Directory5	5MB
Directory4	4MB
Directory3	3MB
Directory2	2MB
Directory19	19MB
Directory18	18MB
Directory17	17MB
Directory16	16MB
Directory15	15MB
Directory14	14MB
Directory13	13MB
Directory12	12MB
Directory11	11MB
Directory10	10MB
Directory1	1MB

Иерархические списки

```
3 //include <QTreeWidget>
4
5 int main(int argc, char *argv[])
6 {
7     QApplication a(argc, argv);
8     QTreeWidget twg;
9     QStringList lst;
10
11     lst<<"Folders"<<"Used Space";
```

Иерархические списки

```
lst<<"Folders"<<"Used Space";  
twg.setHeaderLabels(lst);  
twg.setSortingEnabled(true);  
QTreeWidgetItem * ptwg = new QTreeWidgetItem(&twg);  
ptwg->setText(0, "Local Disk @");  
ptwg->setIcon(0, QPixmap("C:\\Qt\\MyProjects\\drive.jpg"));
```

Иерархические списки

```
QTreeWidgetItem * ptwgItemDir = 0;  
for (int i=1; i<20; i++){  
    ptwgItemDir = new QTreeWidgetItem(ptwg);  
    ptwgItemDir->setText(0, "Directory"+QString::number(i));  
    ptwgItemDir->setText(1,  
        QString::number(i)+"MB");
```

Иерархические списки

```
    QString::number(i)+"MB");
    ptwgItemDir->setIcon(0, QPixmap("C:\\Qt\\MyProjects\\folder.jpg"));
}

twg.setItemExpanded(ptwg, true);
twg.resize(350, 400);
twg.show();

return a.exec();
```

Иерархические списки

При нажатии на заголовок столбца проводится сортировка элементов в убывающем или возрастающем порядке.

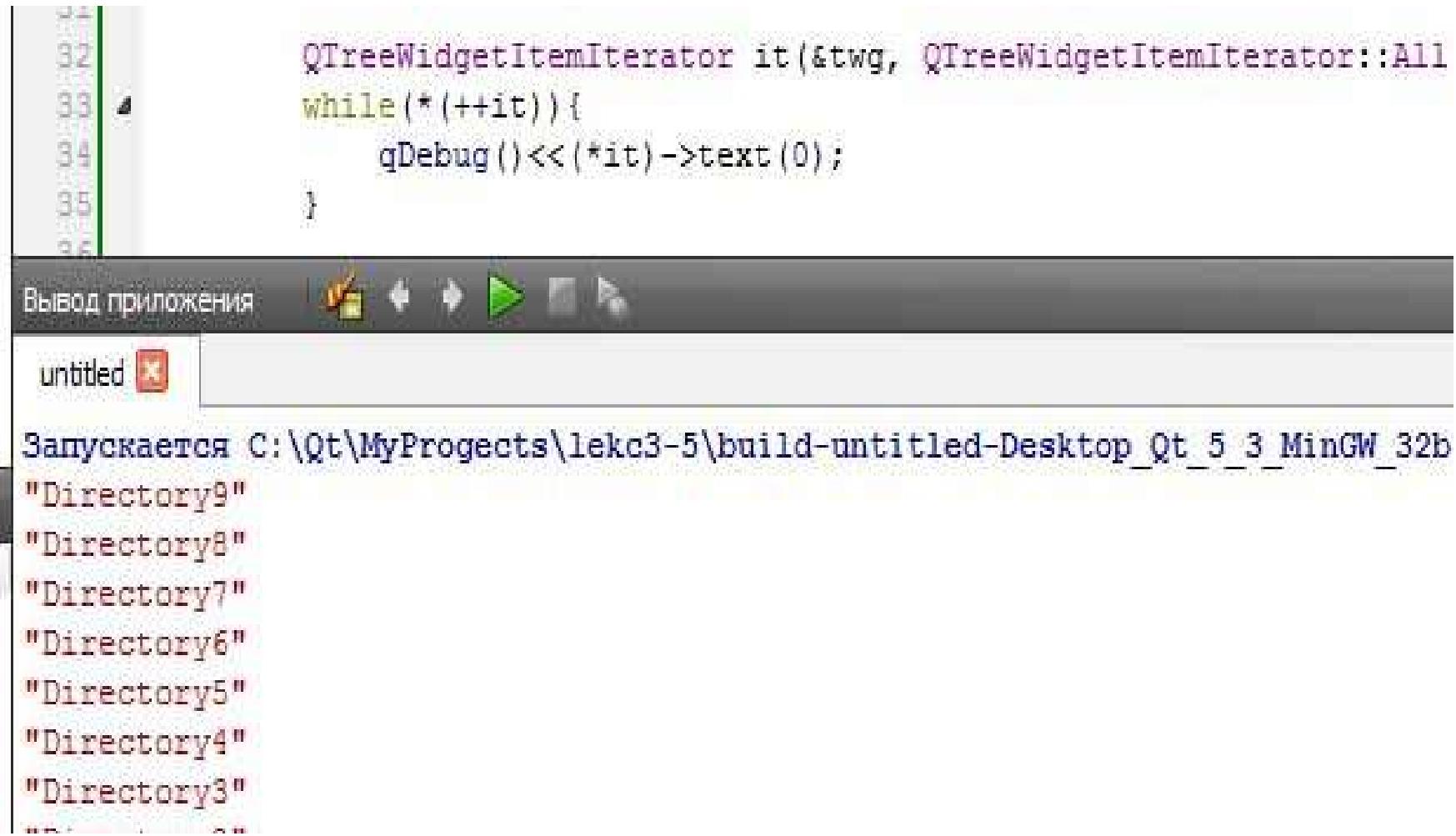
Подобную сортировку можно разрешить либо запретить методом `setSortingEnabled()`, передав `true` либо `false`.

Иерархические списки

По умолчанию пользователь может одновременно выбирать из списка только один из элементов.

Если этого недостаточно, то нужно вызвать метод **setSelectionMode()** с параметром **QAbstractItemView::MultiSelection**, который устанавливает режим множественного выделения.

Иерархические списки



The screenshot shows the Qt Creator interface with a code editor and a terminal window.

Code Editor:

```
21
22
23     QTreeWidgetItemIterator it(&twg, QTreeWidgetItemIterator::All)
24     while(*(++it)) {
25         qDebug() << (*it)->text(0);
26     }
27
28
29
30
31
32
33
34
35
36
```

Terminal Output:

Выход приложения

```
untitled 
```

Запускается C:\Qt\MyProjects\lekc3-5\build-untitled-Desktop_Qt_5_3_MinGW_32b

```
"Directory9"
"Directory8"
"Directory7"
"Directory6"
"Directory5"
"Directory4"
"Directory3"
"Directory2"
"Directory1"
```

Сигналы QTreeWidget

itemSelectionChanged() – сообщает об изменении выбранных элементов;

itemClicked(QTreeWidgetItem*, int) – отправляется после щелчка на элементе;

itemDoubleClicked(QTreeWidgetItem*, int) – отправляется при двойном щелчке мыши;

itemActivated(QTreeWidgetItem*, int) – отправляется при двойном щелчке мыши, а также при нажатии клавиши Enter на элементе.

QTreeWidget drag and drop

```
ptwi->setFlags(Qt::ItemIsDragEnabled | Qt::ItemIsEnabled);
```

Qt::ItemIsEditable

QTreeWidget сортировка

```
bool MyTreeWidgetItem::operator<(const QTreeWidgetItem& ptwiOther)
{
    bool bRet = false;
    int nColumn = treeWidget()->sortColumn();
    if (nColumn == 0) {
        QString strFormat="dd.MM.yyyy";
        bRet = QDate::fromString(text(nColumn))
            < QDate::fromString(ptwi.text(nColumn));
    }
    return bRet;
}
```

Таблицы

QTableWidget

```
const int n = 4;  
QTableWidget tbl(n, n); //размер таблицы  
QTableWidgetItem* ptwi = 0;  
QStringList lst;  
lst<<"Первый"<<"Второй"<<"Третий"<<"Четвертый";  
tbl.setHorizontalHeaderLabels(lst); //заголовки  
tbl.setVerticalHeaderLabels(lst); //заголовки  
for(int i=0; i<n; ++i){
```

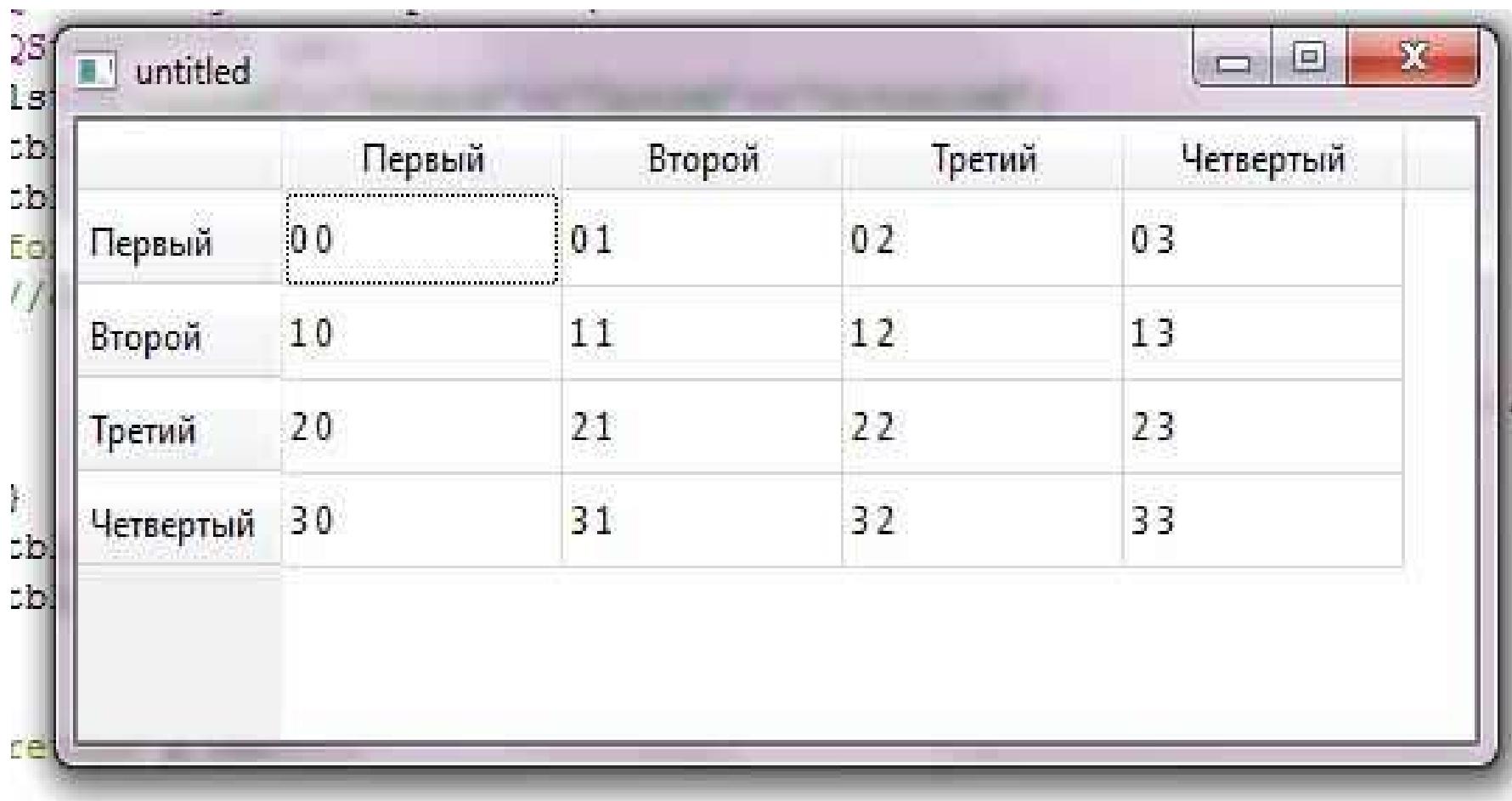
Таблицы

QTableWidget

```
for(int i=0; i<n; ++i){  
    //создание объектов ячеек  
    for(int j=0; j<n; ++j){  
        ptwi = new QTableWidgetItem(QString("%1 %2").arg(i).arg(j));  
        tbl.setItem(i, j, ptwi); }  
}  
tbl.resize(500, 200);  
tbl.show();
```

Таблицы

QTableWidget



The screenshot shows a Windows application window titled "untitled". Inside the window is a QTableWidget control containing a 4x5 grid of cells. The columns are labeled "Первый", "Второй", "Третий", "Четвертый", and an unlabeled column on the far right. The rows are labeled "Первый", "Второй", "Третий", "Четвертый", and an unlabeled row at the bottom. The cell at the intersection of the second row and the first column is highlighted with a dashed border.

	Первый	Второй	Третий	
Первый	0 0	0 1	0 2	0 3
Второй	1 0	1 1	1 2	1 3
Третий	2 0	2 1	2 2	2 3
Четвертый	3 0	3 1	3 2	3 3

Таблицы

QTableWidget

```
QStringList lst;
```

	Первый	Второй	Третий	Четвертый
Первый	Вставка текста			
Второй				
Третий				
Четвертый				

```
ls
tb
tb
fa
// tb
tb
tb
tb
tb
tb
tb
tb
tb
lb->setPixmap(QPixmap("C:/Qt/MyProjects/apple.png"));
```

Таблицы

QTableWidget

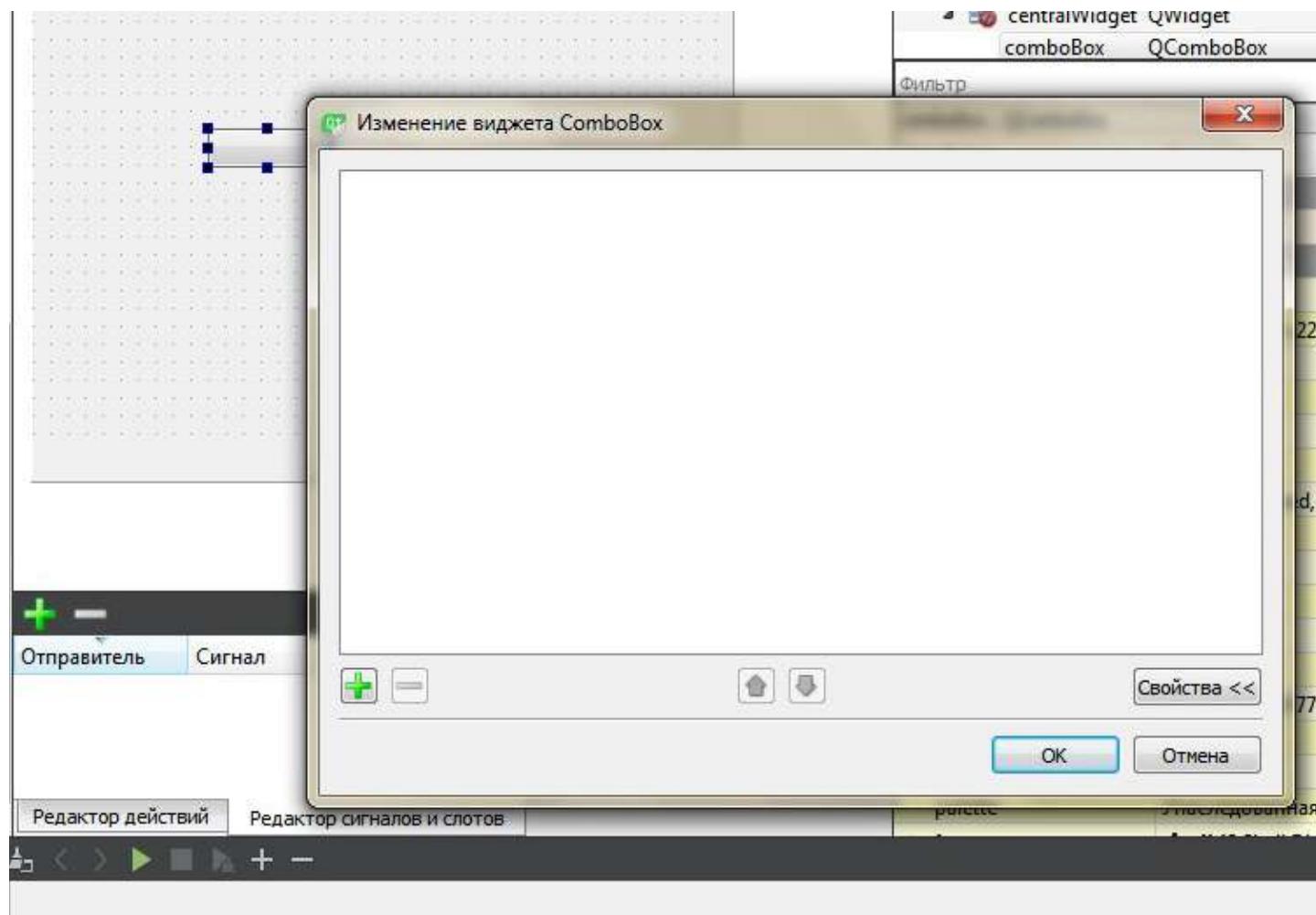
```
16  * for(int i=0; i<n; ++i){  
17  *     //создание объектов ячеек  
18  *     for(int j=0; j<n; ++j){  
19  *         ptwi = new QTableWidgetItem();  
20  *         tbl.setItem(i, j, ptwi); }  
21  *     }  
22  *     tbl.resize(500, 200);  
23  *     tbl.show();
```

Таблицы

QTableWidget

```
23     tbl.show();
24     tbl.item(0,0)->setText ("Вставка текста");
25     tbl.item(0,1)->setBackgroundColor(QColor(192,45,90));
26     tbl.item(0,2)->setIcon (QIcon("C:/Qt/MyProjects/apple.png"));
27     QLabel *lb = new QLabel;
28     lb->setPixmap (QPixmap("C:/Qt/MyProjects/apple.png"));
29     tbl.setCellWidget (0,3,lb);
```

Выпадающий список



Выпадающий список

Для добавления текста **addItem()**

для нескольких элементов в метод **addItems()** передается
указатель на объект класса **QStringList**.

иконки **setItemIcon()**

режим, исключающий повторы в списке
setDuplicatesEnabled(false).

Для удаления элементов – **clear()**.

какой из элементов является текущим **currentIndex()**

Чтобы добавлять элементы в список **setEditable(true)**.

После изменения пользователем текста выбранного элемента отправляется сигнал **editTextChanged(const QString&)**, и новый элемент добавляется в список.

Выпадающий список

После выбора элемента отправляются сразу два сигнала **activated()**:

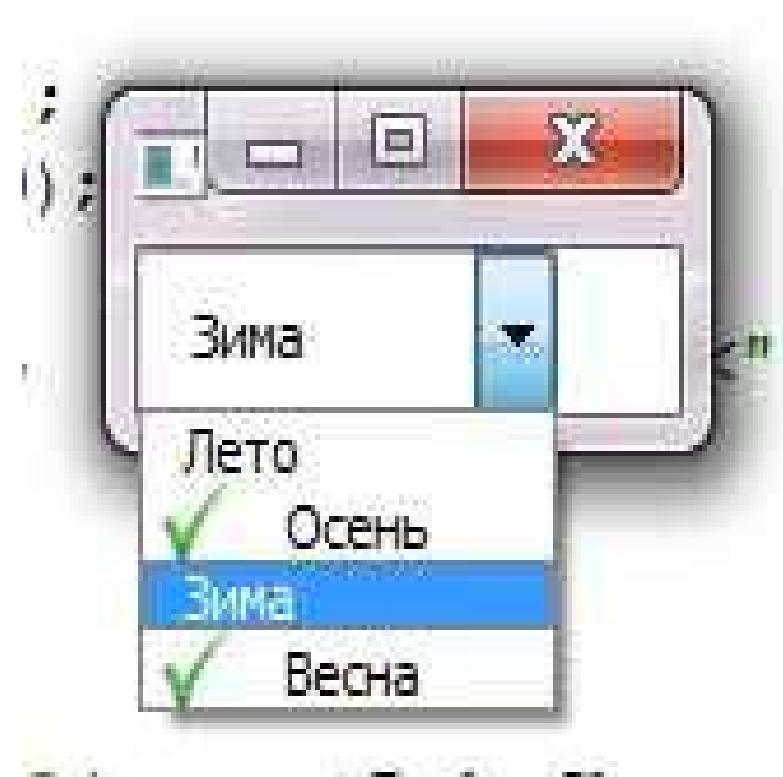
один с параметром типа int (индексом выбранного элемента),

а другой – с параметром типа const QString& (его значением).

currentIndexChanged() - для информирования о реальном изменении

с параметрами int и const QString& каждый.

Выпадающий список. Пример



Рассмотрим главную функцию

```
14 // int main(int argc, char *argv[])
15 {
16     QApplication a(argc, argv);
17     MyComb *cb = new MyComb(0);
18     QStringList lst;
19     lst<<" Лето "<<" Осень "<<" Зима "<<" Весна ";
20     cb->addItems(lst);
21     cb->setEditable(true);
22     cb->resize(90, 30);
23     cb->show();
24     QObject::connect(cb, SIGNAL(currentIndexChanged(int)),
25                       bb, SLOT(SetPic(int)));
26     return a.exec();
27 }
```

СЛОТ

```
6     bool flags[4];
7     void MyComb::SetPic(int index){
8         if (flags[index]==0)
9             {setItemIcon(currentIndex(),QIcon("C:/Qt/MyProjects/gal.png"));
10             flags[index]=1; }
11         else setItemIcon(currentIndex(),QIcon("")); }
12 }
```

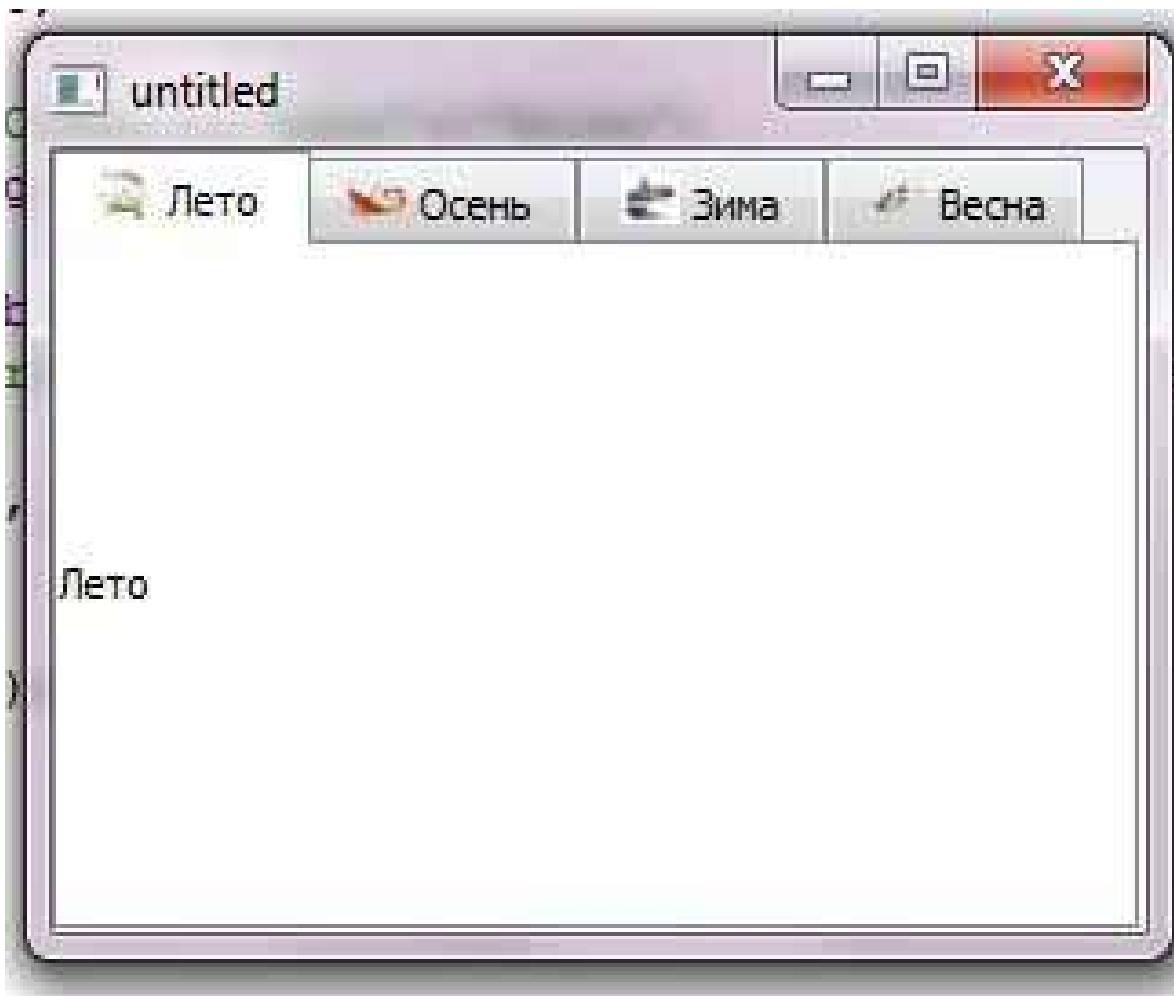
Определение класса

```
#ifndef CB_H
#define CB_H

#include <QComboBox>
class MyComb: public QComboBox
{
    Q_OBJECT
public:
    MyComb(QWidget* parent) : QComboBox(parent) {};
public slots:
    void SetPic(int);
};

#endif
```

Вкладки



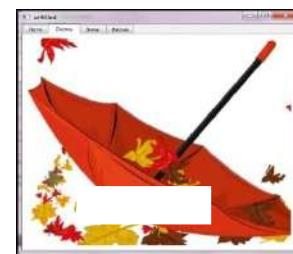
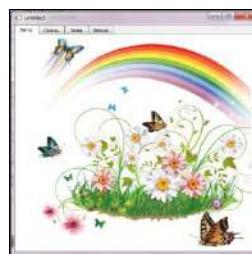
Вкладки

```
6 int main(int argc, char *argv[])
7 {
8     QApplication a(argc, argv);
9     QTabWidget tab;
10    QStringList lst;
11    int i=0;
12    lst<<"Лето"<<"Осень"<<"Зима"<<"Весна";
13    foreach(QString str, lst){
14        i++;
15        tab.addTab(new QLabel(str, &tab),
16        QPixmap("C:/Qt/MyProgects/" +QString("%1").arg(i)+".png"), str);
17    }
18    tab.resize(500, 500);
19    tab.show();
```

Вкладки

```
int QTabWidget::insertTab ( int index, QWidget * page, const  
QString & label)
```

Этот метод вставляет вкладку с указанной меткой *label* и страницей *page* в виджет со вкладками по указанному индексу *index* и возвращает индекс вставленной вкладки в панели вкладок.



Вкладки

```
12 int i=0;
13 lst<<"Лето"<<"Осень"<<"Зима"<<"Весна";
14 foreach(QString str, lst) {
15     i++;
16     QWidget *window = new QWidget;
17     QLabel *lb = new QLabel;
18     lb->setPixmap(QPixmap("C:/Qt/MyProjects/" +QString("%1").arg(i)+".png"));
```

Вкладки

```
19 QHBoxLayout *layout = new QHBoxLayout;
20 layout->addWidget(lb);
21 window->setLayout(layout);
22 tab.insertTab(i, window, str);
23
24 tab.resize(500, 500);
25 tab.show();
```

Виджет панели инструментов

```
5 int main(int argc, char *argv[])
6 {
7     QApplication a(argc, argv);
8     QToolBox tbx;
9     QStringList lst;
10
11     lst<<"Linux"<<"Windows"<<"MacOSX"<<"Android";
12     foreach(QString str,lst) {
13         tbx.addItem(new QLabel(str, &tbx), QPixmap(str+".jpg"),str);
14     }
15     tbx.resize(100,80);
16     tbx.show();
17
18     return a.exec();
19 }
```



name: 8, 30, 8, 8, custom margin: 0, 0, 0, 0, minimum size: 69x189, maximum size: 69x189

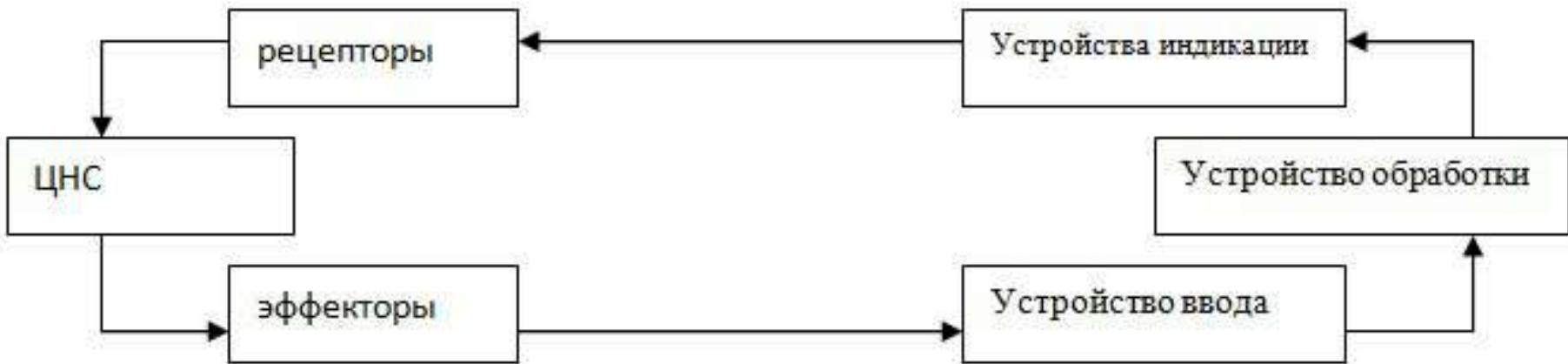
Человеко-машинное взаимодействие

Лекция 1

Мерзлякова Екатерина Юрьевна

к.т.н. доцент ПМИК

Основные вопросы человеко-машинного взаимодействия



Пользовательский интерфейс программы - это совокупность элементов, позволяющих пользователю программы управлять ее работой и получать требуемые результаты.

Основные вопросы человеко-машинного взаимодействия

Н0 С3ЙЧ4С Н4 Э70Й
С7Р0К3 84Ш Р4ЗУМ
ЧN7437 Э70
4870М47НЧ3СКН,
Н3 349УМЫ184ЯСЬ 06
Э70М. Г0Р9НСЬ.
ЛНШЬ ОПР393ЛЗННЫЗ
ЛЮ9Н М0ГУ7 ПРОЧN747Ь
Э70.



QtCreator

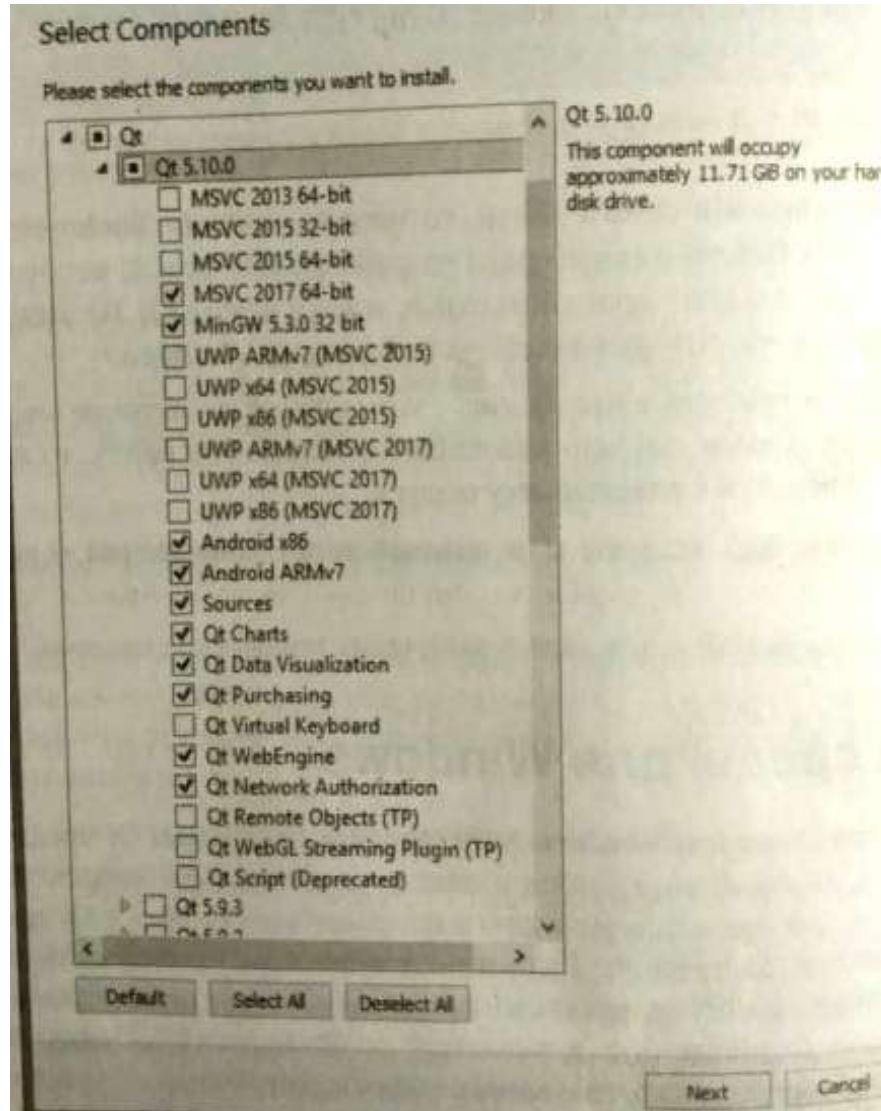
Microsoft Windows,
Mac OS X,
Linux,
FreeBSD и других клонов UNIX,
iOS, Android,
Windows Phone, Windows RT и BlackBerry.

Установка Qt 5.10

<https://www.qt.io/download>

https://disk.yandex.ru/d/QCj0Jujk_6_xnA

Установка Qt 5.10



MinGW 5.3.0 32 bit.

Применение Qt

Adobe Amazon AMD Bosh BMW BlackBerry

Canon Cisco Systems Disney Intel IBM

Panasonic Parallels Pioneer Philips Oracle

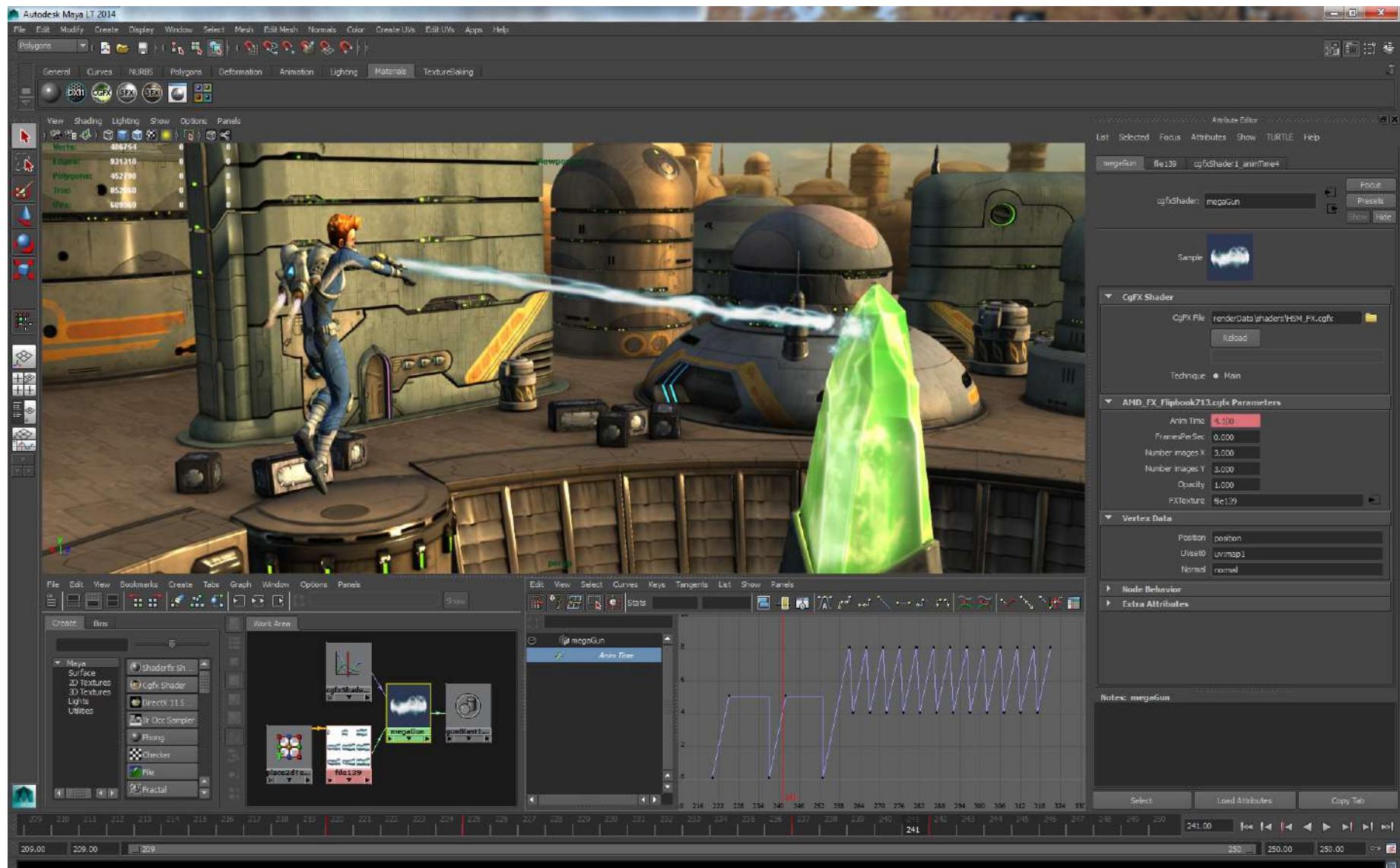
HP Goober Google Mersedes NASA NEC

Neonway Nokia Rakuten Samsung Siemens

Sony SUN Tesla Xerox Xillinx Yamaha

И др.

Применение Qt



Применение Qt



Viber: Звонки и Сообщения

Viber Media S.à r.l. Связь

3+

Есть платный контент.

⚠ У вас нет устройств.

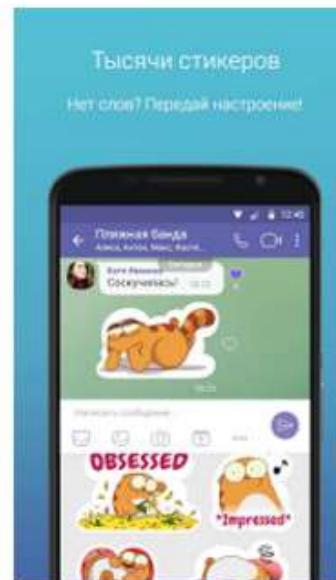
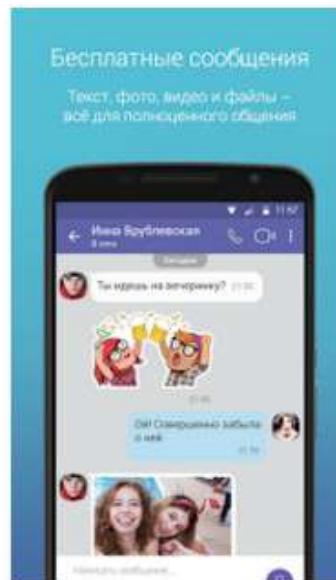
Лучший разработчик

★★★★★ 9 658 993

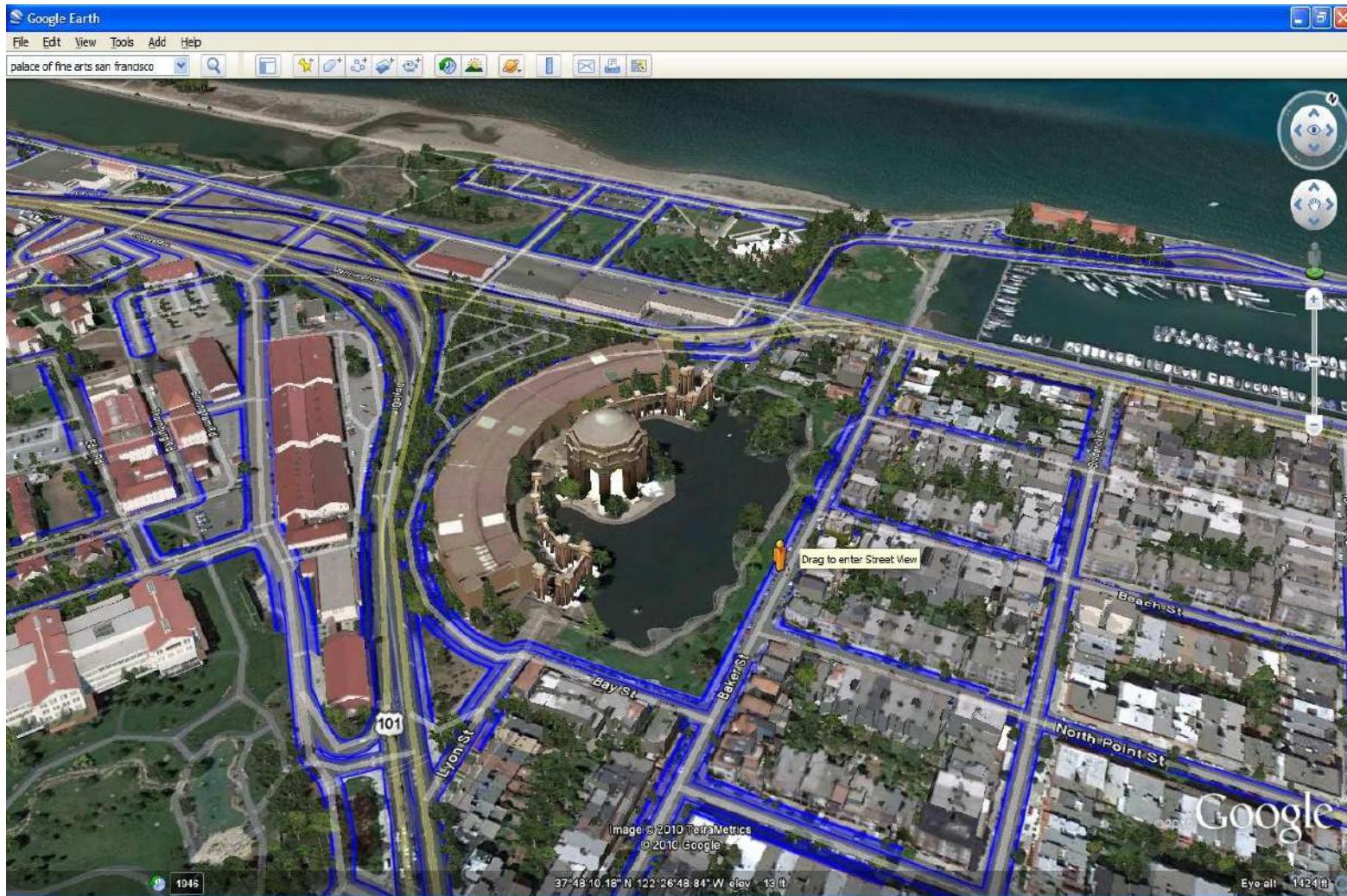


Добавить в список желаний

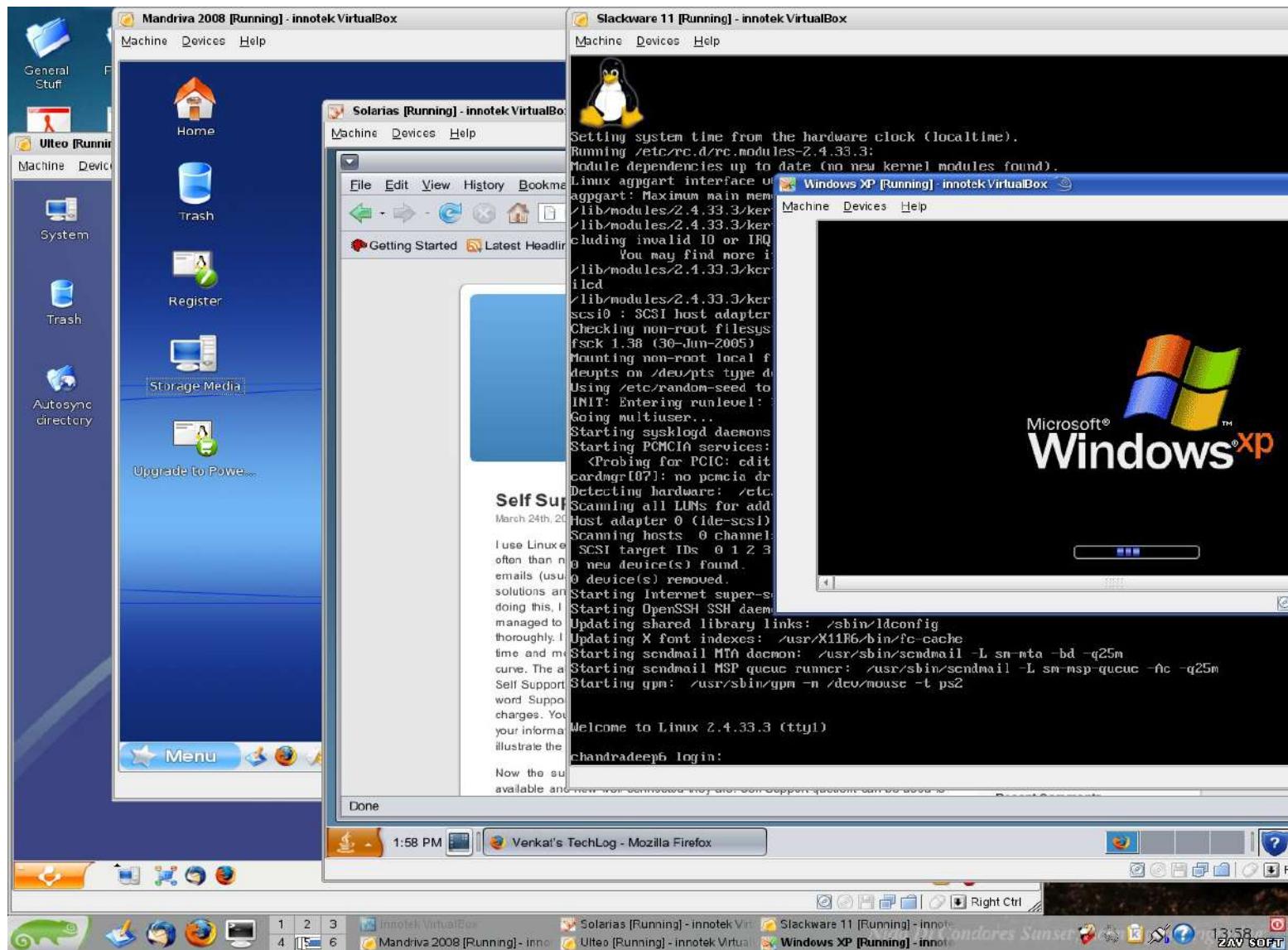
Установить



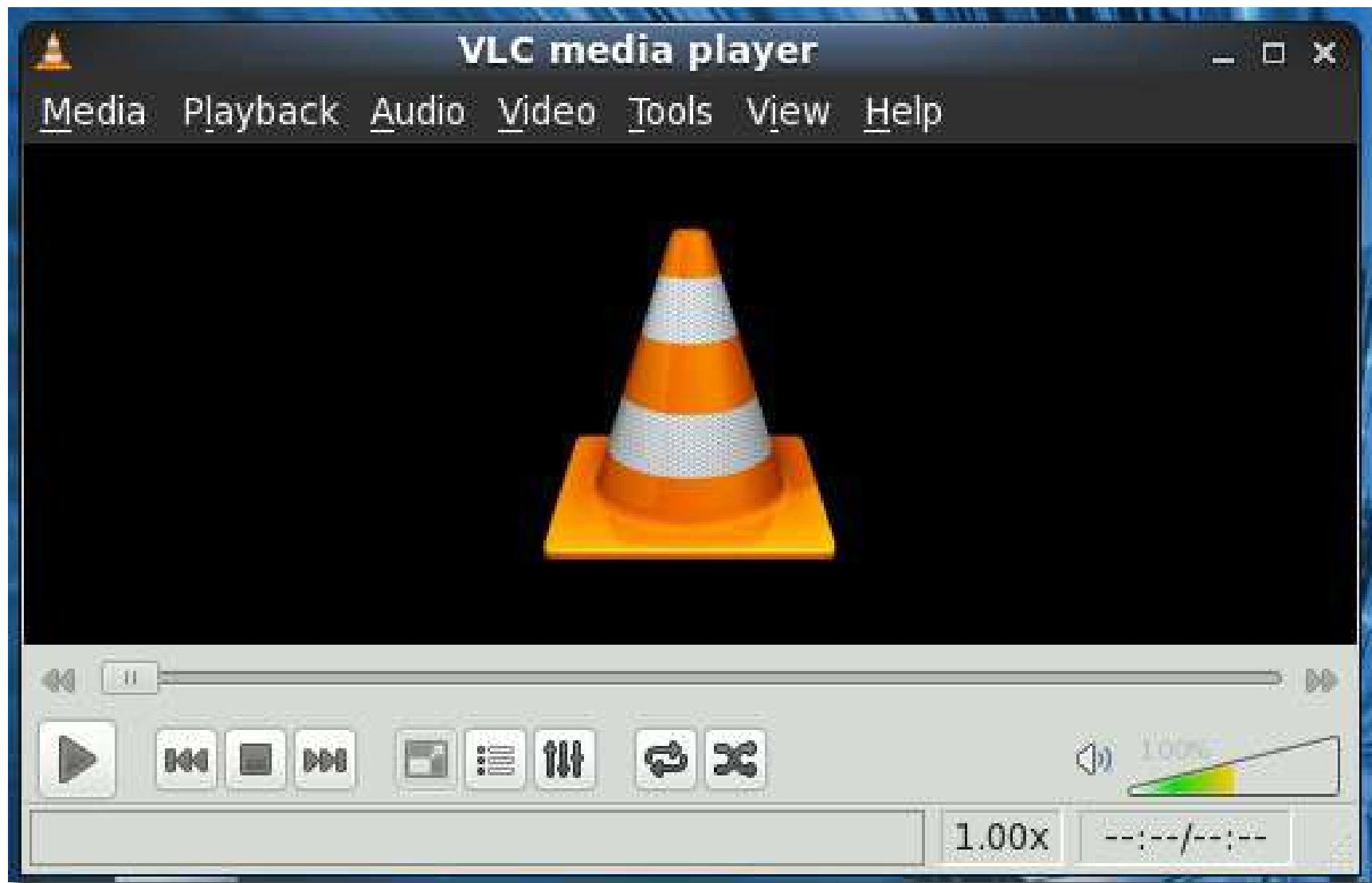
Применение Qt



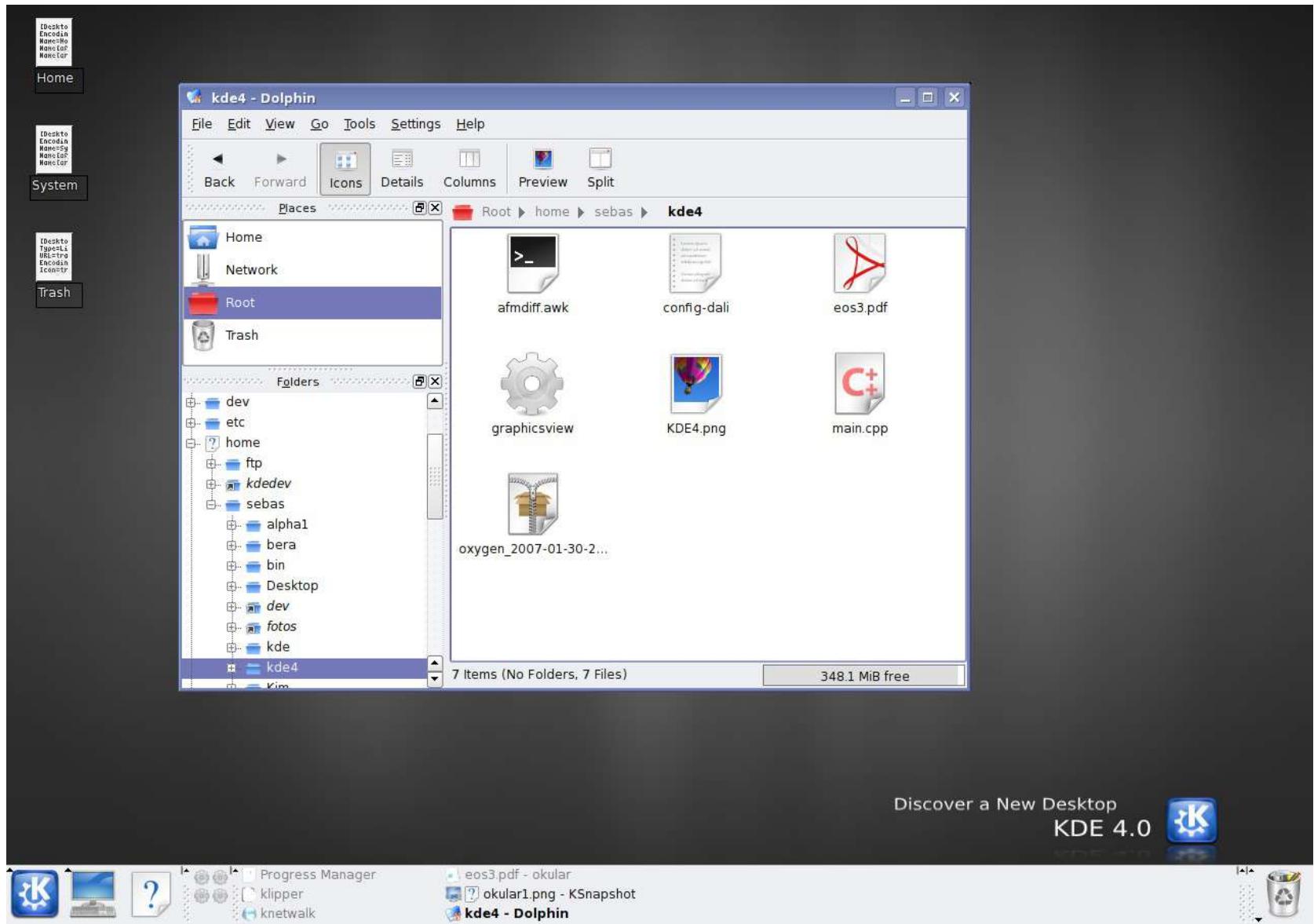
Применение Qt



Применение Qt



Применение Qt



Возможности Qt

- Поддержку 2D и 3D графики (OpenGL, QT 3D)
- Возможность интернационализации
- Использование форматов JSON и XML
- STL-совместимую библиотеку контейнеров
- Поддержку стандартных протоколов ввода-вывода.
- Классы для работы с сетью
- Поддержку программирования баз данных
- И.др.

Модули QT

- **Qt Core** — основной модуль, который содержит все базовые средства Qt . На его основе построены все другие модули. Каждая программа созданная с использованием Qt , использует этот модуль;
- **Qt Network** — модуль для работы с сетевыми средствами;
- **Qt Gui** — модуль поддержки графического вывода на экран. В Qt4 он также содержит набор виджетов для создания графического интерфейса пользователя. В Qt5 виджеты вынесены в отдельный модуль;
- **Qt Widgets** — модуль, который содержит набор виджетов для создания графического интерфейса пользователя (Qt5);
- **Qt WebKit** — средства работы с Веб;
- **Qt WebKit Widgets** — виджеты для работы с Веб (Qt5);
- **Qt Multimedia** — средства работы с мультимедийными устройствами и файлами;
- **Qt Multimedia Widgets** — виджеты для работы с мультимедийными устройствами и файлами (Qt5);
- **Qt Sql** — средства работы с базами данных;
- **Qt Qml** — поддержка декларативного языка QML для разработки динамических визуальных интерфейсов (Qt5);
- **Qt Quick** — поддержка создания динамических визуальных интерфейсов (Qt5);
- **Qt Quick Controls** — использование технологии QtQuick для создания традиционного для рабочих столов графического интерфейса (Qt5);
- **Qt Quick Layouts** — компоновка для элементов QtQuick (Qt5).

Модули QT

для включения модуля QtWidgets
нужно добавить в программу строку
`#include <QtWidgets>`

<https://doc.qt.io/archives/qt-5.10/qtmodules.html>

Пространство имен QT

Qt::red

using namespace Qt;

```
1 #include <QCoreApplication>
2
3 #include <QTCodec>
4 #include <iostream>
5
6 using namespace std; //Подключение стандартной библиотеки функций
7
8 int main(int argc, char *argv[])
9 {
```

Литература по Qt

- Шлее М. «Qt 5.10. Профессиональное программирование на C++». БХВ-Петербург, 2018.

https://qt-book.com/shlee_qt_5_10_professionalnoe_programmirovaniye_na_cpp.pdf

- Алан Купер, Дэвид Кронин, Роберт Рейман. «Алан Купер об интерфейсе. Основы проектирования взаимодействия»

<https://disk.yandex.ru/i/-qIVgEn4j2d6Bw>

- Герберт Шилдт. «Полный справочник по C++». 2015г.

https://sharpened.ucoz.ru/lib/polnyj_spravochnik_po_c-gerbert_shildt-2006.pdf

Интеграция справки по Qt

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>

namespace Ui {
    class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

protected:
    Ui::MainWindow *ui;

private:
    // ...
};

#endif // MAINWINDOW_H
```

QMainWindow Class

Qt 5.3 ▶ Qt Widgets ▶ C++ Classes ▶ QMainWindow

Qt 5.3.2 Reference Documentation

The QMainWindow class provides a main application window. [More...](#)

Header: #include <QMainWindow>

qmake: QT += widgets

Inherits: QWidget

- List of all members, including inherited members

Public Types

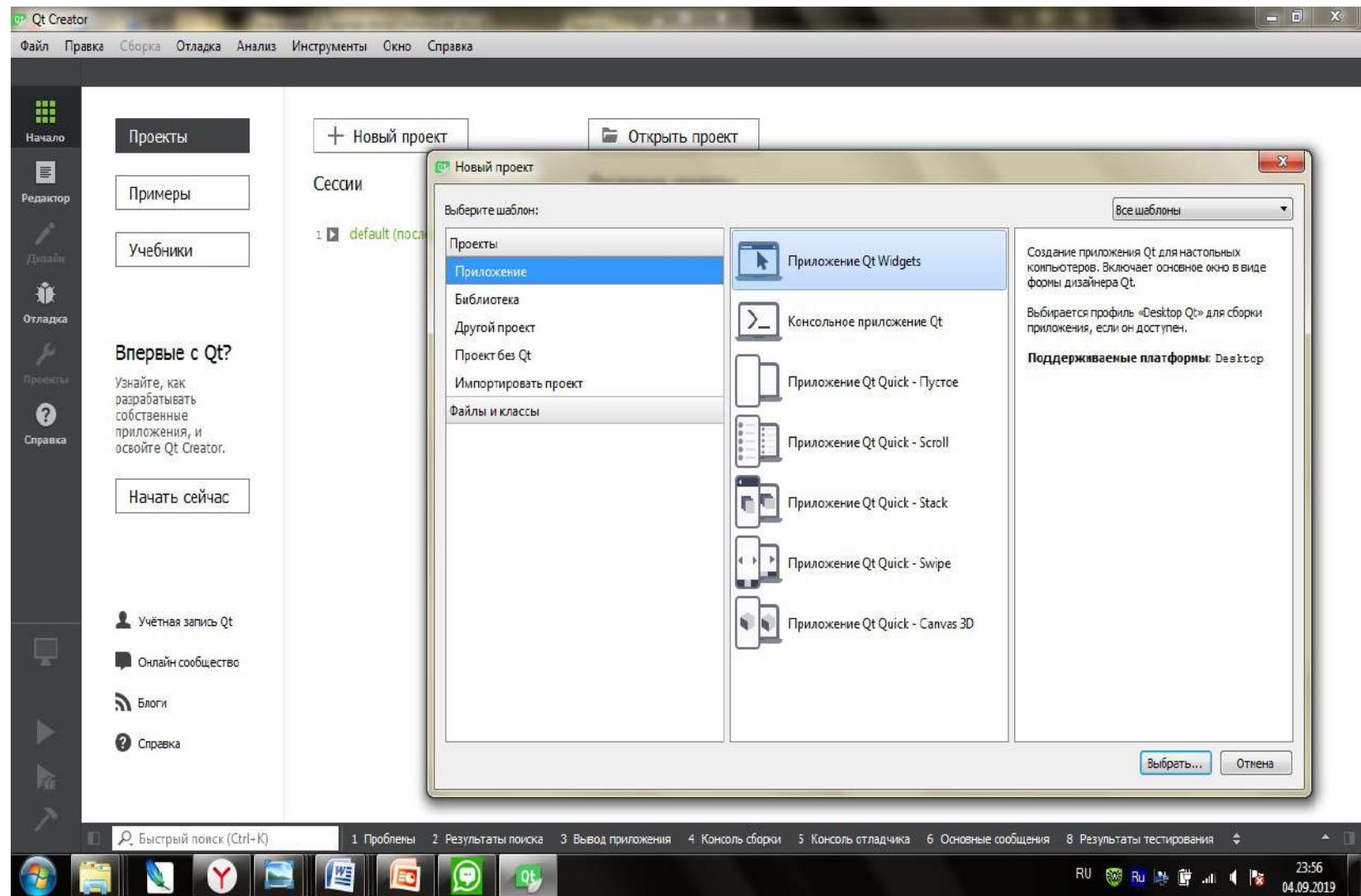
enum DockOption {
 AnimatedDocks,
 AllowNestedDocks,
 AllowTabbedDocks,
 ForceTabbedDocks,
 VerticalTabs }

Contents

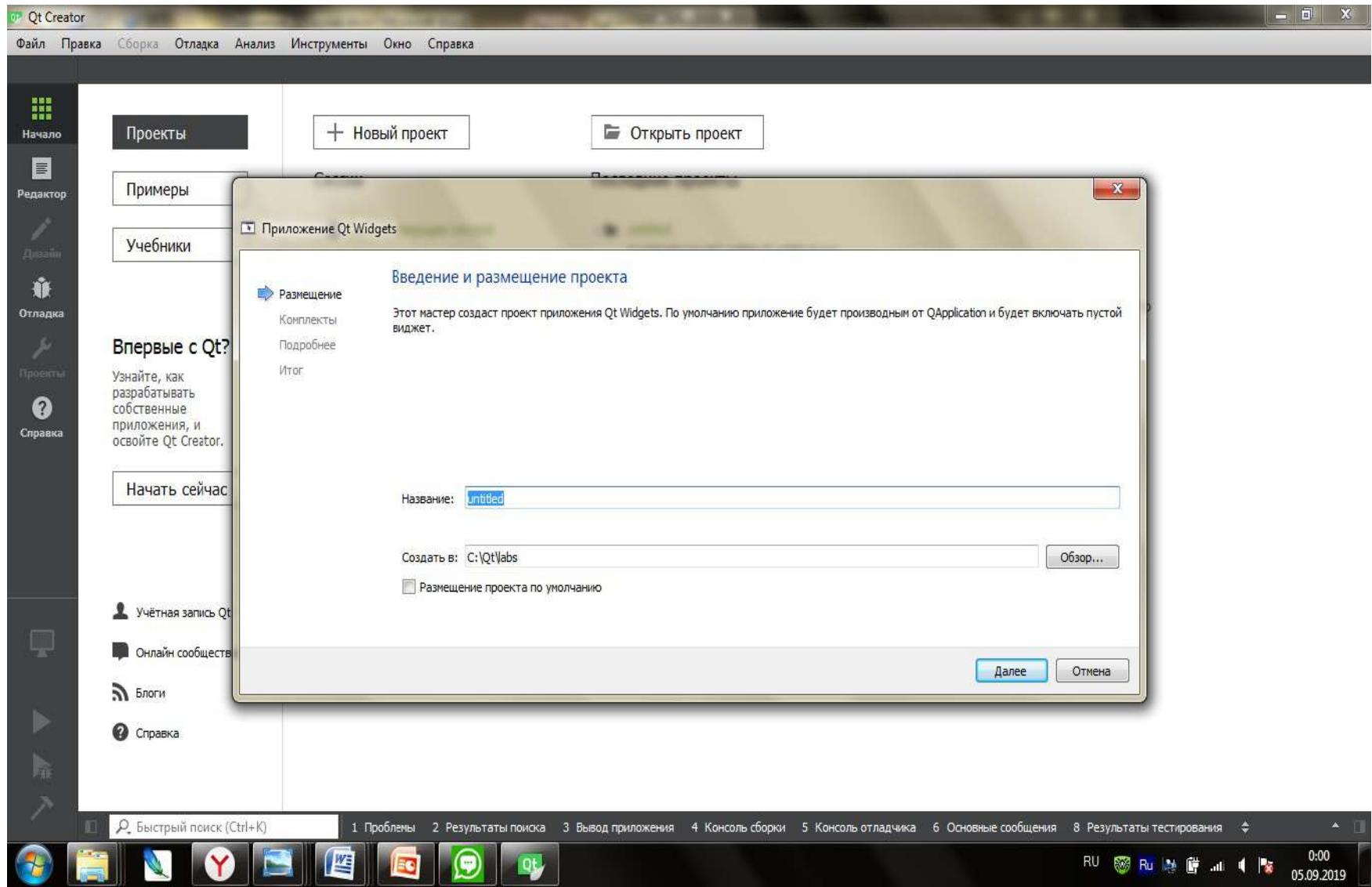
- Public Types
- Properties
- Public Functions
- Public Slots
- Signals
- Protected Functions
- Detailed Description
- Qt Main Window Framework
- Creating Main Window Components
- Creating Menus
- Creating Toolbars
- Creating Dock Widgets
- The Status Bar



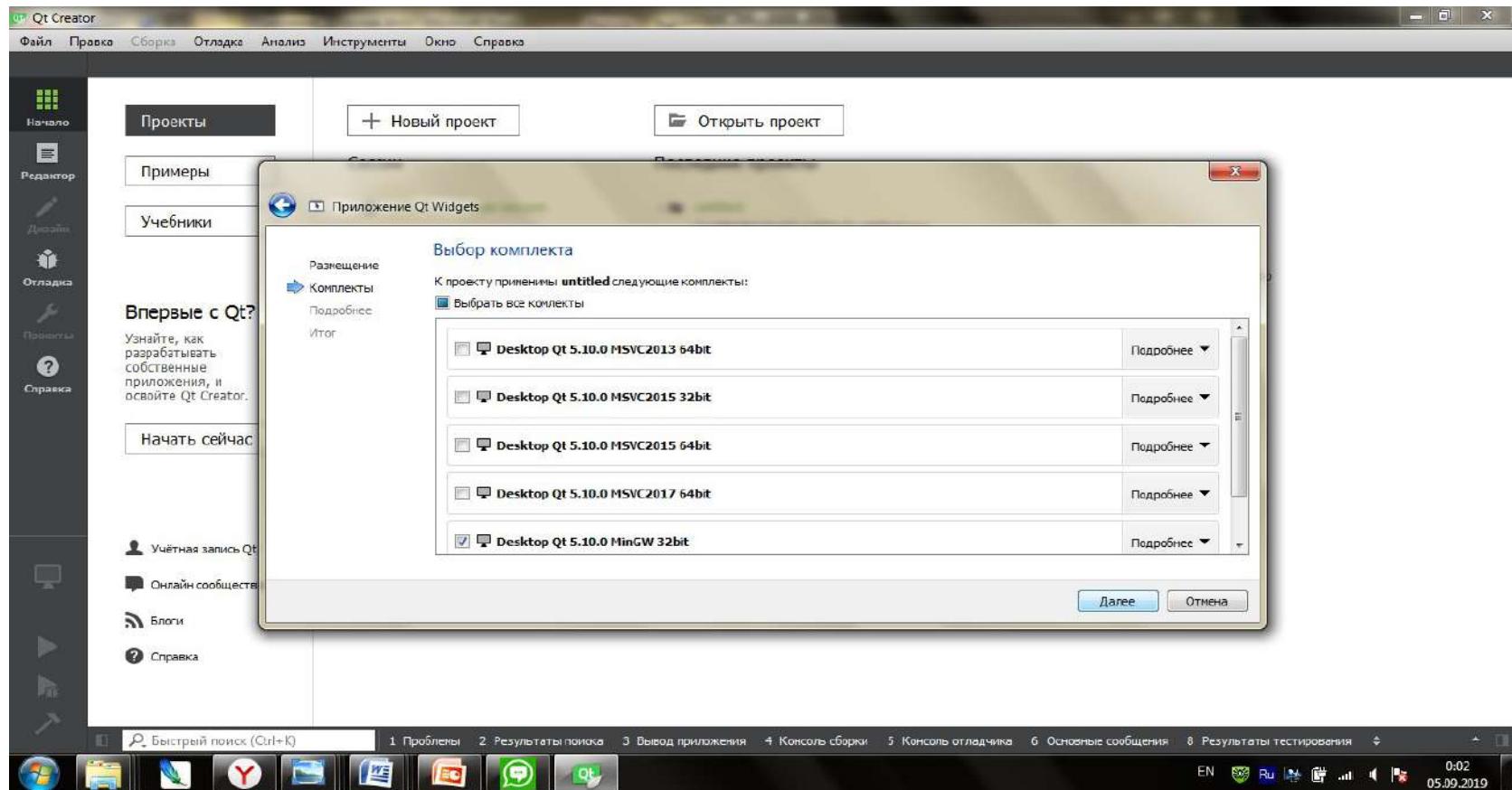
Создание приложения в QtCreator



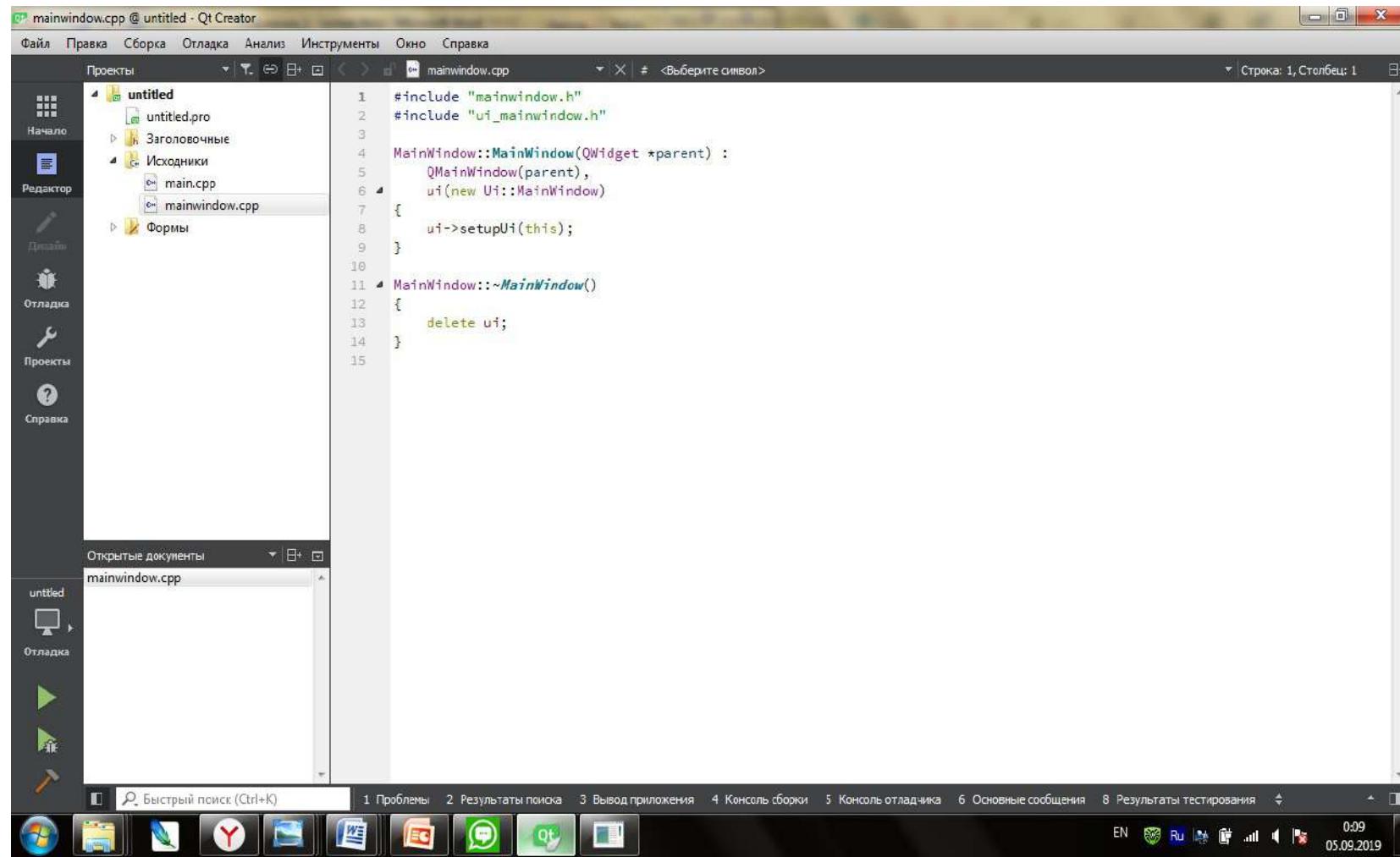
Создание приложения в QtCreator



Создание приложения в QtCreator



Режимы в QtCreator



Панели вывода. Панель проблемы

The screenshot shows a Qt-based IDE interface. On the left, there's a sidebar titled "Открытые документы" (Open documents) containing files: main.cpp, mainwindow.cpp, mainwindow.h, and mainwindow.ui. The main area displays a portion of a C++ code snippet:

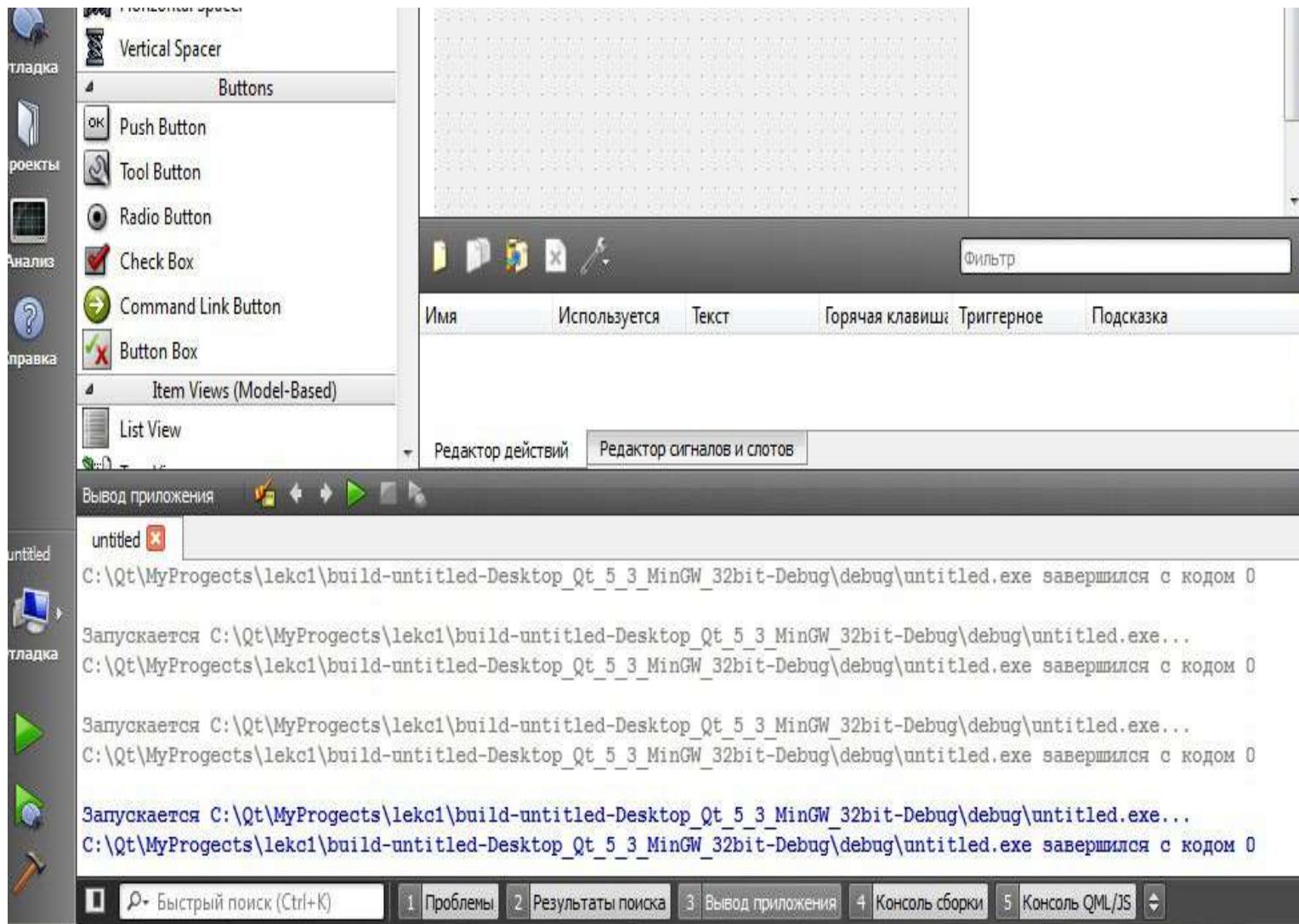
```
11     return a.exec();  
12 }  
13
```

Below the code editor is the "Problems" panel, which lists four errors:

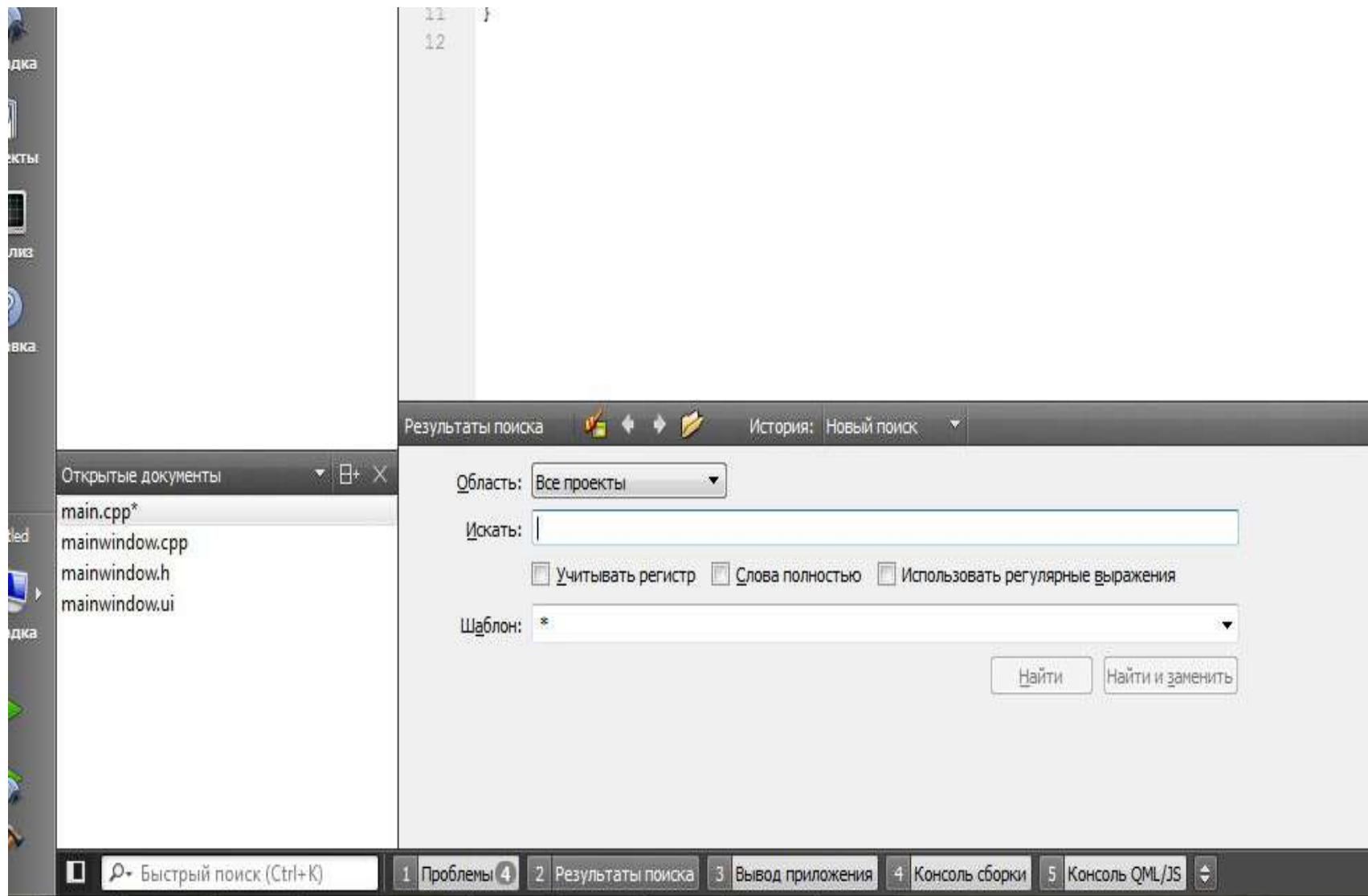
- stray '\320' in program
реп°
- stray '\270' in program
- stray '\320' in program
- stray '\260' in program

The status bar at the bottom contains several tabs: Быстрый поиск (Ctrl+K), Проблемы (4), Результаты поиска, Вывод приложения, Консоль сборки, Консоль QML/JS, and a dropdown menu.

Панели вывода. Вывод приложения



Панели вывода. Результат поиска



Панели вывода. Консоль сборки

The screenshot shows the Qt Creator interface with the 'Build Console' panel open. The console window displays error messages from the build process:

```
..\untitled\main.cpp:10:1: error: stray '\270' in program
..\untitled\main.cpp:10:1: error: stray '\320' in program
..\untitled\main.cpp:10:1: error: stray '\260' in program
Makefile.Debug:347: recipe for target 'debug/main.o' failed
mingw32-make[1]: *** [debug/main.o] Error 1
mingw32-make[1]: Leaving directory 'C:/Qt/MyProjects/lekc1/build-untitled-Desktop_Qt_5_3_Mi
Makefile:34: recipe for target 'debug' failed
mingw32-make: *** [debug] Error 2
21:12:28: Процесс «C:\Qt\Tools\mingw482_32\bin\mingw32-make.exe» завершился с кодом 2.
Ошибка при сборке/установке проекта untitled (комплект: Desktop Qt 5.3 MinGW 32bit)
Во время выполнения этапа «Сборка»
21:12:28: Прошло времени: 00:04.
```

The 'Open documents' sidebar on the left lists files: main.cpp*, mainwindow.cpp, mainwindow.h, and mainwindow.ui. The bottom navigation bar includes tabs for 'Problems' (4), 'Search results' (2), 'Application output' (3), 'Build console' (4), and 'QML/JS console' (5).

Режим дизайна

Файл Правка Сборка Отладка Анализ Инструменты Окно Справка

Проекты mainwindow.cpp <Нет символов>

Qt Начало Редактор Дизайн Отладка Проекты

untitled
untitled.pro
Заголовочные
Исходники
main.cpp
mainwindow.cpp
Формы
mainwindow.ui

```
1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3
4 MainWindow::MainWindow(QWidget *parent) :
5     QMainWindow(parent),
6     ui(new Ui::MainWindow)
7 {
8     ui->setupUi(this);
9 }
10
11 mainwindow::~MainWindow()
12 {
13     delete ui;
14 }
```

Режим дизайна

mainwindow.ui - untitled - Qt Creator

Файл Правка Сборка Отладка Анализ Инструменты Окно Справка

mainwindow.ui

Фильтр

Лэйауты

- Vertical Layout
- Horizontal Layout
- Grid Layout
- Form Layout

Спейсеры

- Horizontal Spacer
- Vertical Spacer

Кнопки

- Push Button
- Tool Button
- Radio Button
- Check Box
- Command Link Button
- Button Box

Предметные виды (Модель-ориентированные)

- List View
- Tree View
- Table View

Предметные виды (Предмет-ориентированные)

- List Widget
- Tree Widget
- Table Widget

Контейнеры

- Group Box
- Scroll Area

Начало

Редактор

Дизайн

Отладка

Проекты

Анализ

Справка

mainwindow : QMainWindow

Свойство Значение

(QObject) objectName MainWindow

(QWidget) windowModality NonModal

enabled checked

geometry [0, 0, 400, 300]

X 0

Y 0

Ширина 400

Высота 300

sizePolicy [Preferred, Preferred]

Горизонтальная ... Preferred

Вертикальная по... Preferred

Горизонтальное ... 0

Вертикальное ра... 0

minimumSize 0 x 0

maximumSize 16777215 x 16777215

sizeIncrement 0 x 0

baseSize 0 x 0

palette Унаследовано

font [MS Shell Dlg]

cursor Arrow

mouseTracking checked

Фильтр

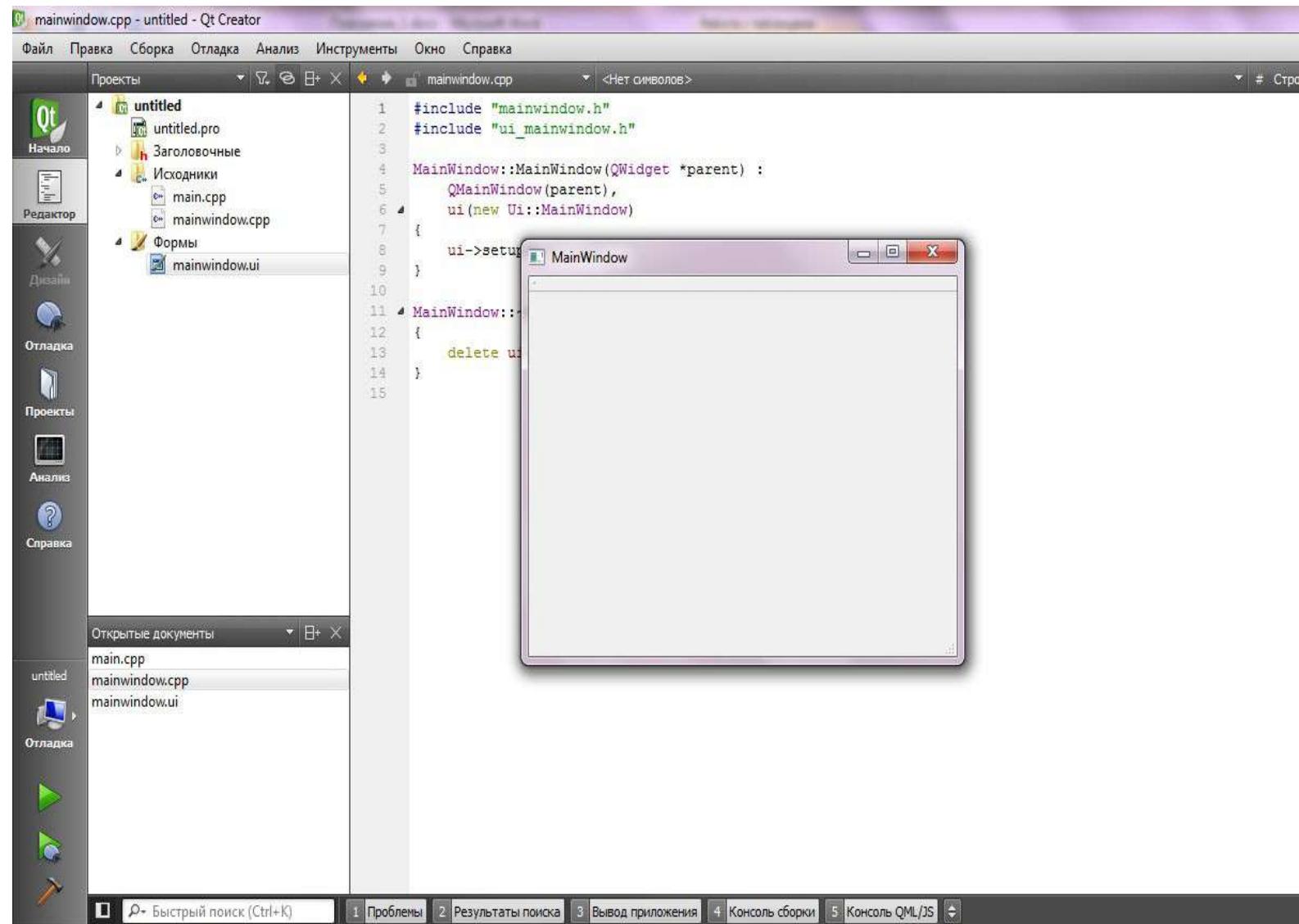
Имя Используется Текст Горячая клавиша Триггерное Подсказка

Редактор действий Редактор сигналов и слотов

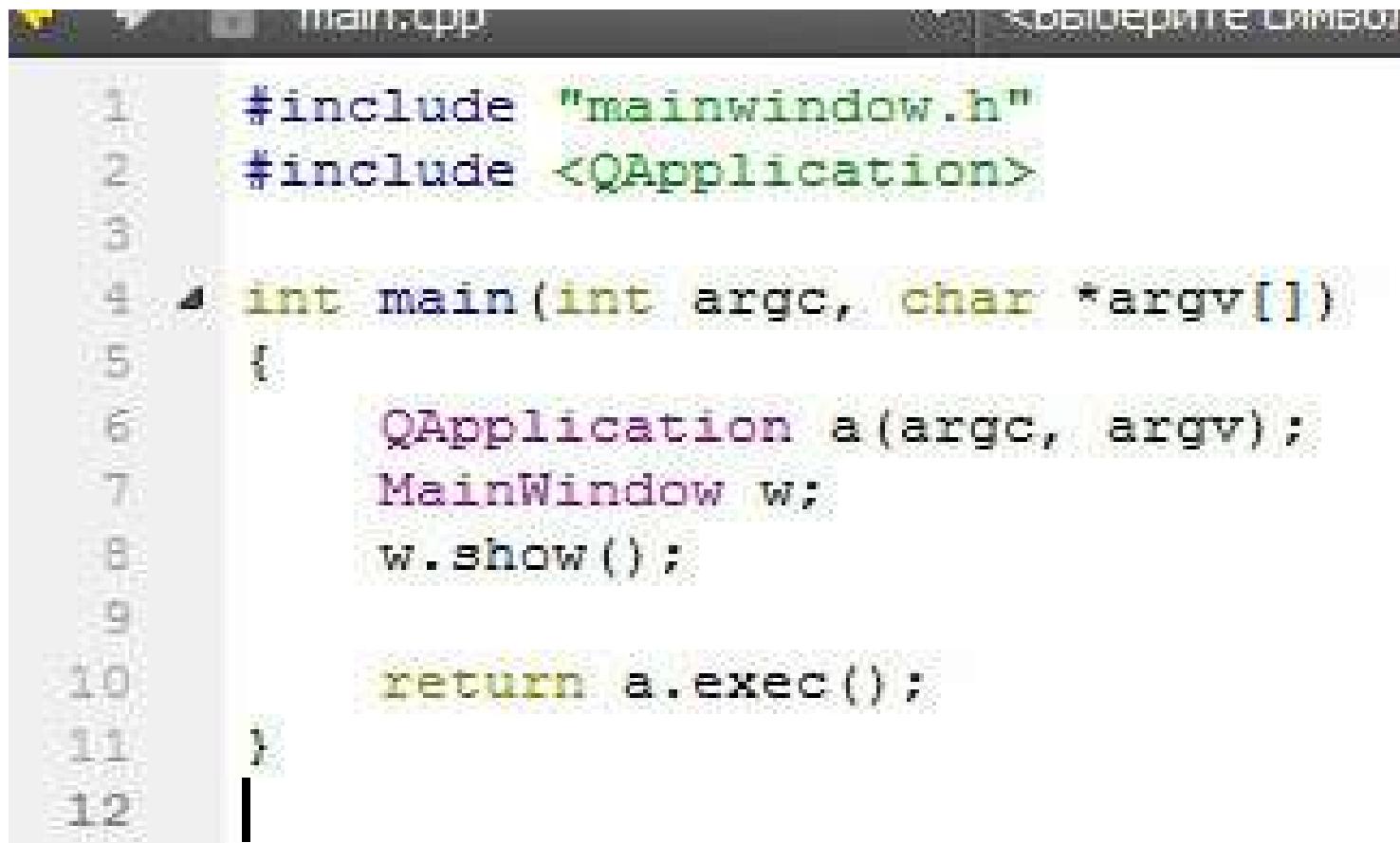
Быстрый поиск (Ctrl+K)

1 Проблемы 2 Результаты поиска 3 Вывод приложения 4 Консоль сборки 5 Консоль QML/JS

Главное окно



main.cpp



A screenshot of a code editor window titled "main.cpp". The window shows C++ code for a Qt application. The code includes #include directives for "mainwindow.h" and <QApplication>, defines the main function with argc and argv parameters, creates a QApplication object named a, creates a MainWindow object named w, shows the window, and finally executes the application using a.exec(). The code is numbered from 1 to 12.

```
1 #include "mainwindow.h"
2 #include <QApplication>
3
4 int main(int argc, char *argv[])
5 {
6     QApplication a(argc, argv);
7     MainWindow w;
8     w.show();
9
10    return a.exec();
11 }
12
```

mainwindow.cpp

```
mainwindow.cpp <Выберите символ>
1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3
4
5
6
7
8
9
10
11 MainWindow::MainWindow(QWidget *parent) :
12     QMainWindow(parent),
13     ui(new Ui::MainWindow)
14 {
15     ui->setupUi(this);
16 }
17
18 MainWindow::~MainWindow()
19 {
20     delete ui;
21 }
```

Заголовок окна

```
#include "mainwindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.setWindowTitle("Qt Application");
    return a.exec();
}
```

Заголовок окна

```
#include "mainwindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.setWindowTitle();           void setWindowTitle(const QString &)
    w.show();

    return a.exec();
}
```

Заголовок окна

The screenshot shows a Qt application interface. At the top, there is a menu bar with Russian labels: Файл (File), Инструменты (Tools), Окно (Window), and Справка (Help). Below the menu is a toolbar with icons for file operations. The main area contains a code editor showing a C++ file named 'main.cpp'. The code defines a main function that includes 'mainwindow.h', uses the QApplication class, creates a MainWindow object, and shows it. The code editor has line numbers from 1 to 13. To the right of the code editor is a window titled 'Программа' (Program) which is a blank white rectangle, representing the main window of the application.

```
#include "mainwindow.h"
#include <QApplication>

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

Создание меню

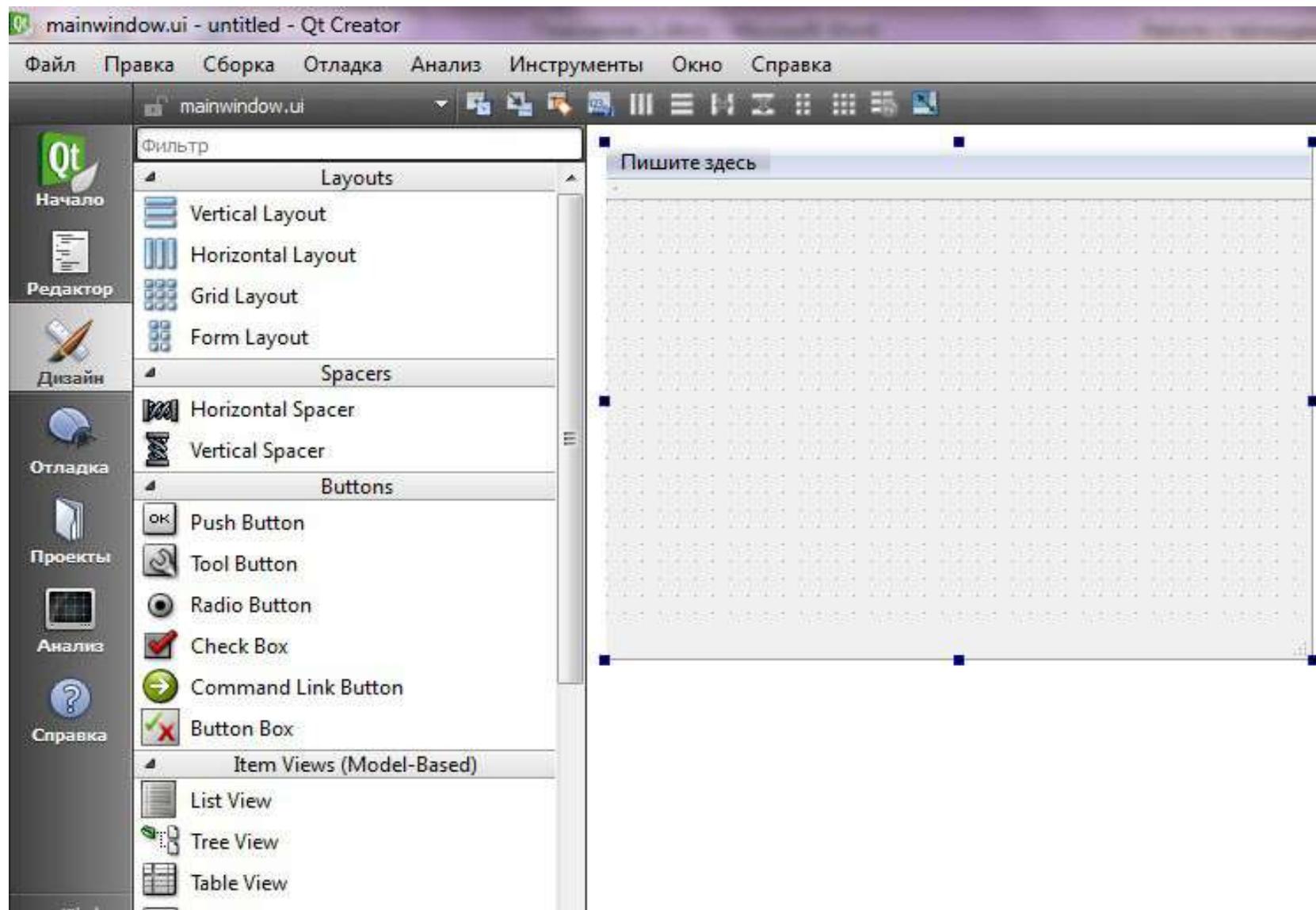
- Меню верхнего уровня
- Всплывающие меню
- Отрывное меню
- Контекстное меню

QMenu

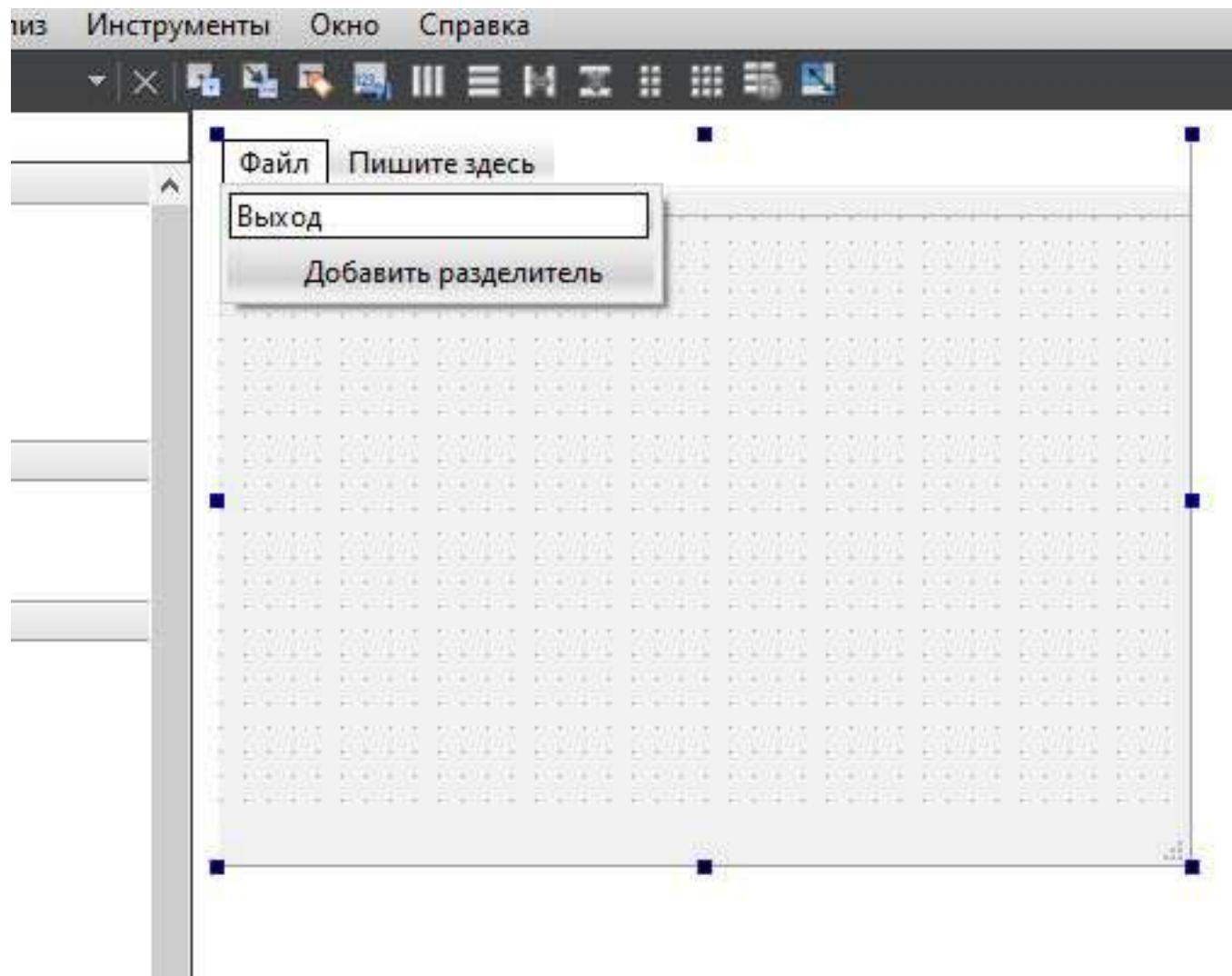
QAction

hovered()

Меню верхнего уровня



Создание меню



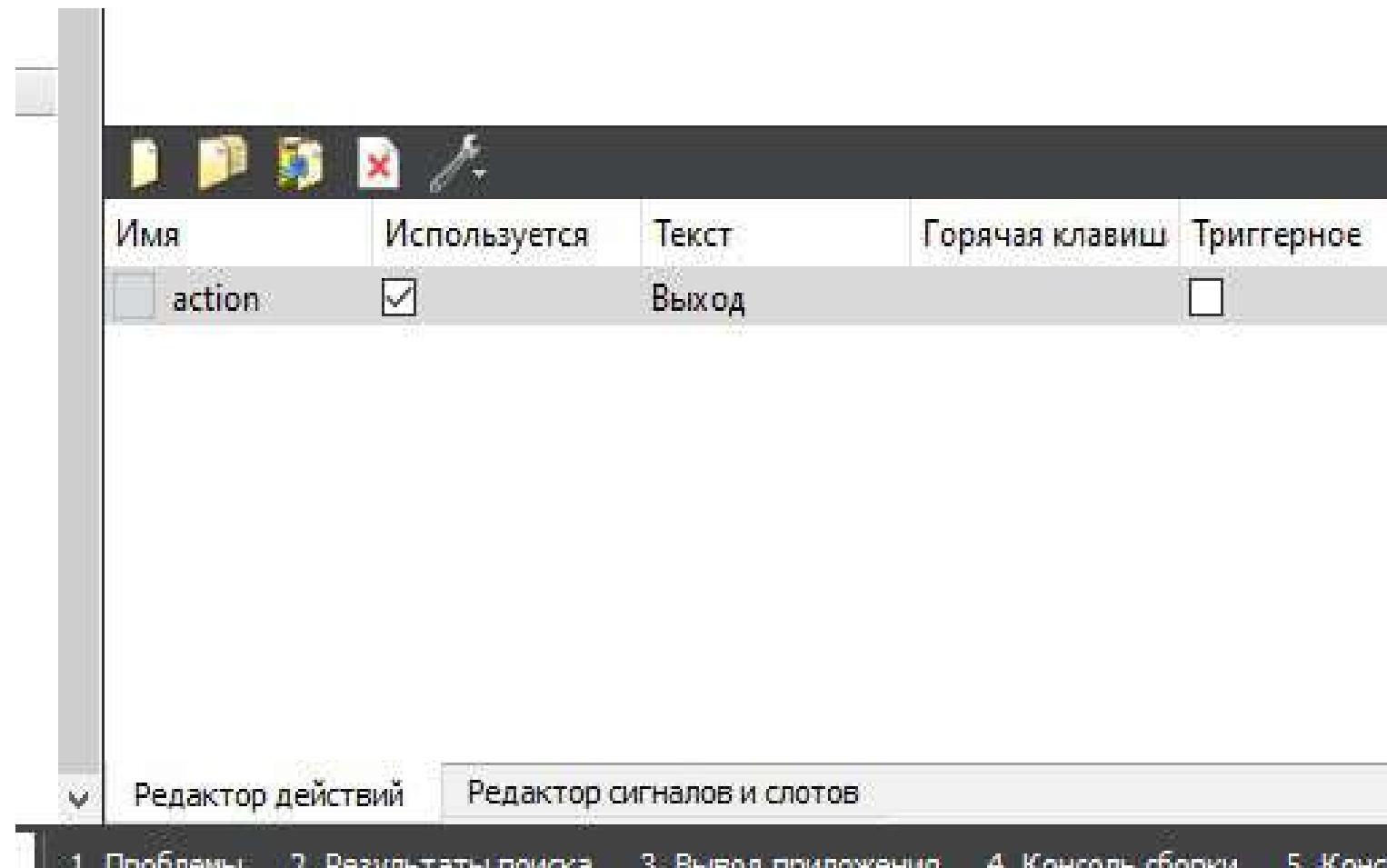
Свойства пунктов меню

The screenshot shows the Qt Designer Properties panel for a QAction object named "action". The properties listed are:

Свойство	Значение
objectName	action
checkable	<input type="checkbox"/>
checked	<input type="checkbox"/>
enabled	<input checked="" type="checkbox"/>
icon	
text	Выход
iconText	Выход
toolTip	Выход
statusTip	
whatsThis	
font	A [MS Shell Di...
shortcut	
shortcutContext	WindowShortcut
autoRepeat	<input checked="" type="checkbox"/>
visible	<input checked="" type="checkbox"/>
menuRole	TextHeuristicRole
iconVisibleInMenu	<input checked="" type="checkbox"/>
shortcutVisibleInCont...	<input type="checkbox"/>
priority	NormalPriority

At the bottom of the panel, there is a message: "Зайдите в раздел \"Параметры\"".

Редактирование меню



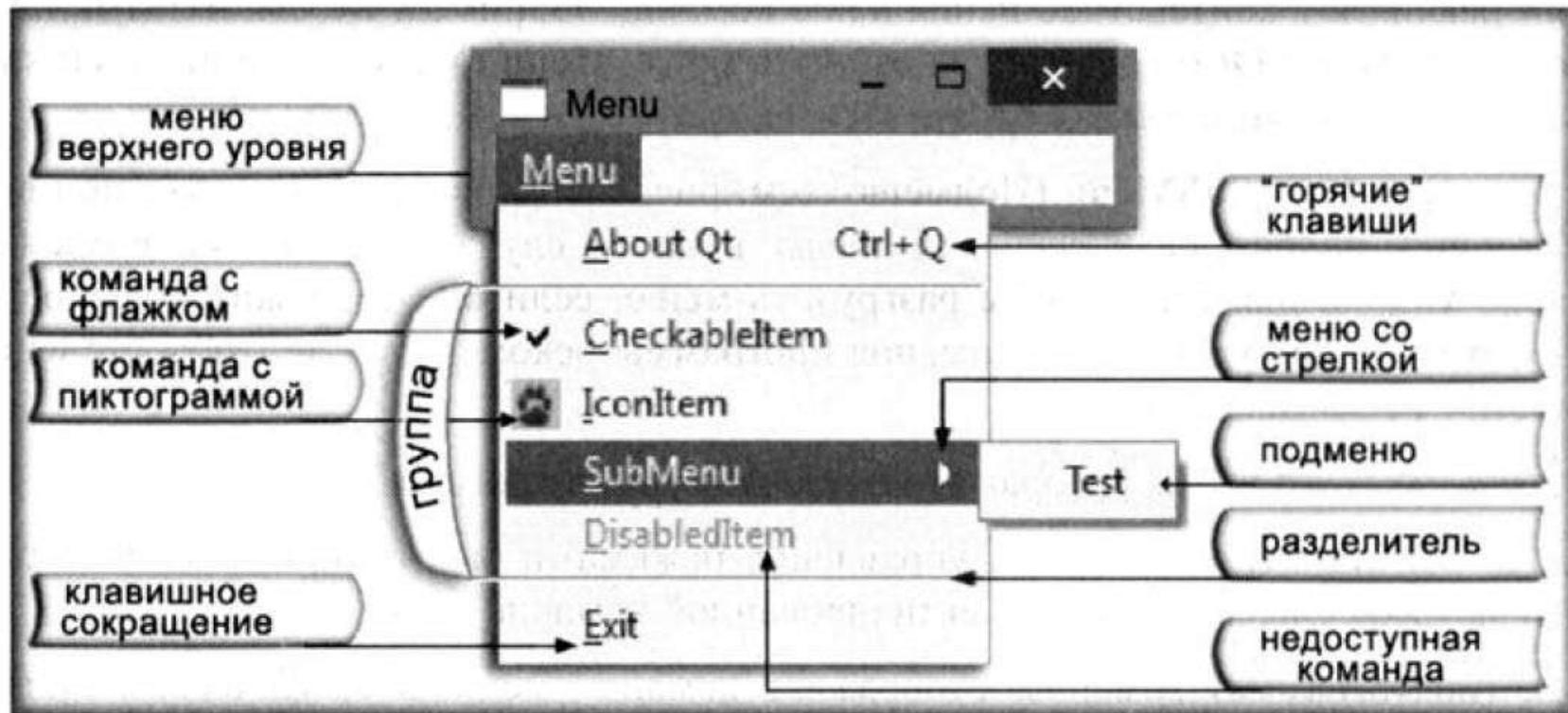
Редактирование меню

Отправитель	Сигнал	Получатель	Слот
action	triggered()	MainWindow	close()

Редактор действий

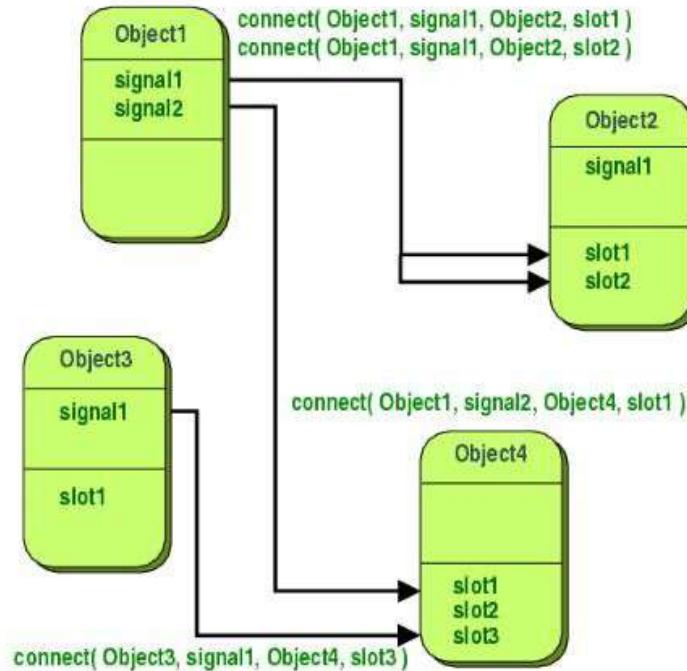
Редактор сигналов и слотов

«Анатомия» меню



Механизм сигналов и слотов

Сигналы и слоты в Qt



Преимущества механизма сигналов и слотов

- каждый класс, унаследованный от QObject, может иметь любое количество сигналов и слотов;
- сообщения, посылаемые посредством сигналов, могут иметь множество аргументов любого типа;
- сигнал можно соединять с различным количеством слотов. Отправляемый сигнал поступит ко всем подсоединенными слотам;
- слот может принимать сообщения от многих сигналов, принадлежащих разным объектам;

Преимущества механизма сигналов и слотов

- соединение сигналов и слотов можно производить в любой точке приложения;
- сигналы и слоты являются механизмами, обеспечивающими связь между объектами. Более того, эта связь может выполняться между объектами, которые находятся в различных потоках;
- при уничтожении объекта происходит автоматическое разъединение всех сигнально-слотовых связей. Это гарантирует, что сигналы не будут отправляться к несуществующим объектам;

Недостатки механизма сигналов и слотов

- сигналы и слоты не являются частью языка C++,;
- отправка сигналов происходит немного медленнее, чем обычный вызов функции, который осуществляется при использовании механизма функций обратного вызова;
- существует необходимость в наследовании класса QObject;
- в процессе компиляции не производится никаких проверок.

Сигналы



Слоты

Public

Private

Protected

Private slots:

Protected slots:

Public slots:

Виджеты, сигналы, слоты

```
1 #include <QApplication>
2 #include <QLabel>
3 int main(int argc, char *argv[])
4 {
5     QApplication app(argc, argv);
6     QLabel *label = new QLabel("Hello, Qt!");
7     label->show();
8     return app.exec();
9 }
```

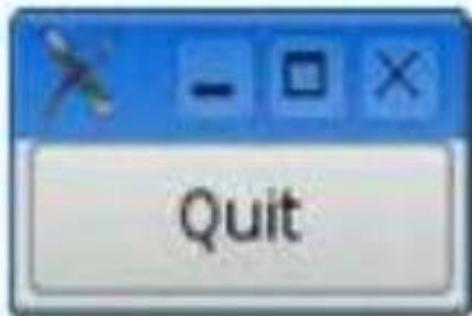
Виджеты, сигналы, слоты

```
bool QObject::connect(  
    const QObject *sender, //источник события  
    const char *signal, // сигнал  
    const QObject *receiver, //объект-приемник  
    const char *method, //функция-обработчик  
    Qt::ConnectionType type = Qt::AutoConnection  
) const
```

Виджеты, сигналы, слоты

```
QObject::connect(  
    scrollBar, // источник события  
    SIGNAL, // сигнал  
    label, // объект-приемник сигнала  
    SLOT(setNum(int)));// функция-обработчик
```

Виджеты, сигналы, слоты



```
1 #include<QApplication>
2 #include<QPushButton.h>
3 int main (int argc, char *argv[])
4 {
5     QApplication app(argc, argv);
6     QPushButton *button =new QPushButton("Quit");
7     QObject::connect(button, SIGNAL(clicked()),
8                       &app, SLOT(quit()));
9     button->show();
10    return app.exec();
11 }
```

Виджеты, сигналы, слоты

```
1 #include<QApplication>
2 #include<QHBoxLayout>
3 #include<QSlider>
4 #include<QSpinBox>
5 int main (int argc, char *argv[])
6 {
7     QApplication app(argc, argv);
8     QWidget *window = new QWidget;
9     window->setWindowTitle("Enter Your Age");
10    QSpinBox *spinBox = new QSpinBox;
11    QSlider *slider =new QSlider(Qt::Horizontal);
12    SpinBox->setRange(0, 130);
13    Slider->setRange(0, 130);
```

Виджеты, сигналы, слоты

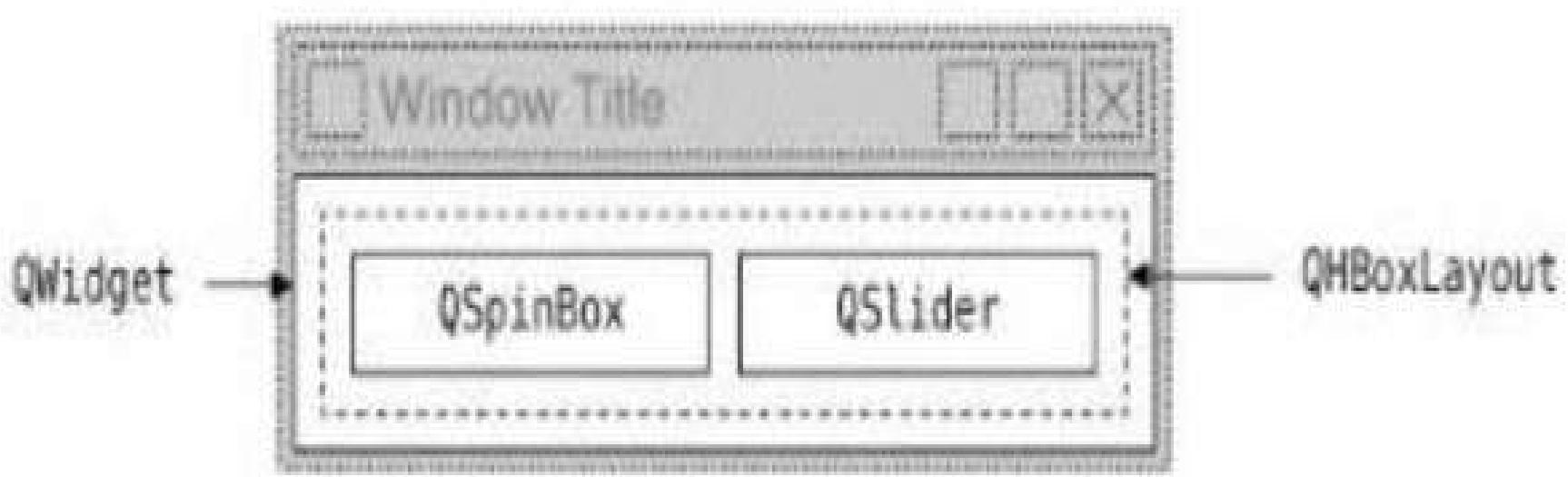
```
14 QObject::connect(spinBox, SIGNAL(valueChanged(int)),
15                     slider, SLOT(setValue(int)) );
16 QObject::connect(slider, SIGNAL(valueChanged(int)),
17                     spinBox, SLOT(setValue(int)) );
18 spinBox->setValue(35);
19 QHBoxLayout *layout = new QHBoxLayout;
20 layout->addWidget(spinBox);
21 layout->addWidget(slider);
22 window->setLayout(layout);
23 window->show();
24 return app.exec();
25 }
```

Менеджер компоновки



- **QHBoxLayout**
- **QVBoxLayout**
- **QGridLayout**

Менеджер компоновки



Человеко- машинное взаимодействие

Лекция 5

Мерзлякова Екатерина Юрьевна

к.т.н. доцент ПМиК

Использование каскадных стилей документа

CSS

***.qss**

`QApplication::setStyleSheet()`

`QWidget::setStyleSheet()`

`a.setStyleSheet("описание стиля")`

```
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;

    QFile file("D:\\q\\styl\\s1.qss");
    file.open(QFile::ReadOnly);
    QString strCSS = QLatin1String(file.readAll());
    a.setStyleSheet(strCSS);
    w.show();
    return a.exec();
}
```



Использование каскадных стилей документа

селектор {свойство: значение}

```
QPushButton {color: blue}
```

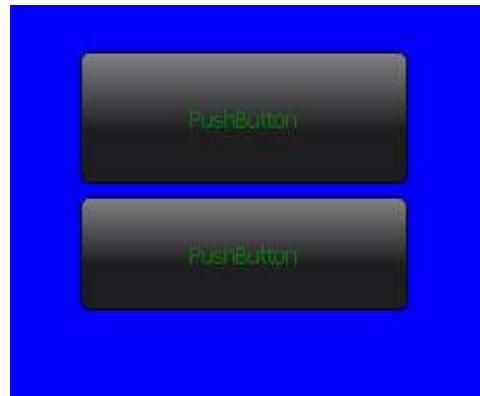
```
QLabel {  
    color: black;  
    background-color: red;  
}
```

```
QLabel {  
    color: rgb(255,0,0);  
    background-color:  
    #FFFFFF;  
}
```



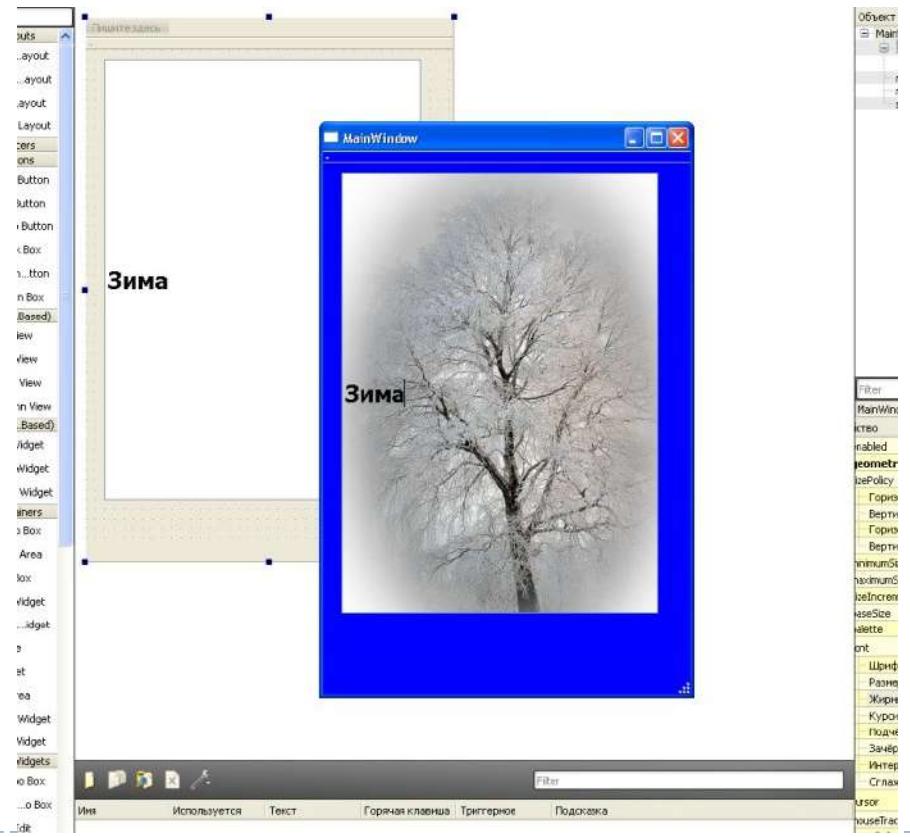
Градиенты

```
QPushButton {  
color: green;  
border: 1px solid black;  
border-radius: 5px;  
background: qlineargradient(x1:0, y1:1, x2:0, y2: 0,  
stop:1 rgb(133,133,135),  
stop:0.4 rgb(31,31,33));  
}
```



Фон виджета

```
QLineEdit {  
background-image: url(D:/q/1.png);  
}  
QMainWindow {  
background-color: blue;  
}
```



Директивы

```
QLineEdit, QLabel, QPushButton {color:  
red}
```

```
.PushButton {color: red}
```

```
QLabel#MyLabel
```



Применение стиля к собственному классу

```
Namespase My {  
Class My : public QWidget {....}; }
```

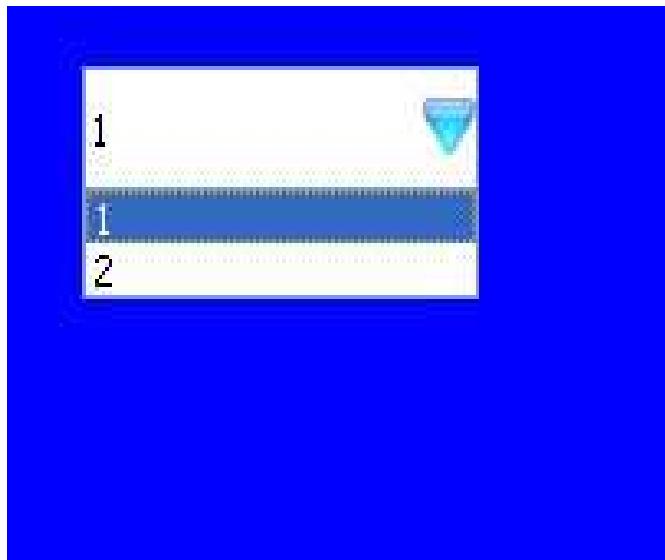
```
/* .qss */  
My {  
    color: red;  
    background-color: blue  
}
```

:/* КОММЕНТ */



Изменение подэлементов

```
QComboBox::drop-down {image: url(D:/q/styl/3.png) }
```

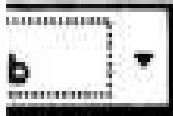


Подэлементы

Подэлемент	Описание	Возможные виды
::down-arrow	Стрелка вниз. Имеется, например, у виджета выпадающего списка и у счетчика	
::down-button	Кнопка вниз. Имеется у виджета счетчика	
::drop-down	Стрелка виджета выпадающего списка	
::indicator	Индикатор кнопки флажка или переключателя, а также группировки кнопок	 
::item	Элемент меню, строки состояния	 Стр.



Подэлементы

<code>::menu-indicator</code>	Индикатор меню кнопки нажатия, обычно это стрелка	
Подэлемент	Описание	Возможные виды
<code>::title</code>	Надпись группы	Title
<code>::up-arrow</code>	Стрелка вверх. Имеется у виджета счетчика	
<code>::up-button</code>	Кнопка вверх. Имеется у виджета счетчика	



Управление состояниями

```
QPushButton: hover {color: red}
```

```
QLineEdit: hover {color: red}
```

Обозначение	Описание
:checked	Активировано
:closed	Виджет находится в закрытом либо свернутом состоянии
:disabled	Виджет недоступен
:enabled	Виджет доступен
:focus	Виджет находится в фокусе ввода
:hover	Указатель мыши находится над виджетом



Управление состояниями

```
QPushButton: hover {color: red}
```

```
QLineEdit: hover {color: red}
```

:indeterminate	Кнопка находится в промежуточном неопределенном состоянии
:off	Выключено (для виджетов, которые могут быть в фиксированном состоянии нажато/не нажато)
:on	Включено (для виджетов, которые могут быть в фиксированном состоянии нажато/не нажато)
:open	Виджет находится в открытом или развернутом состоянии
:pressed	Виджет был нажат мышью
:unchecked	Деактивировано

Объединения состояний

QCheckBox:hover:checked {color: red}

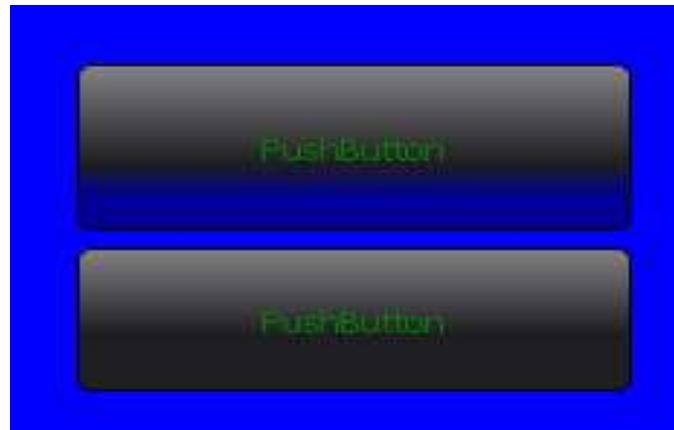
QCheckBox:hover, QCheckBox:checked {color: red}

QLineEdit:!hover {color: red}



Эффект подсвечивания

```
QPushButton:hover {  
background: qlineargradient(x1:0, y1:1, x2:0, y2:0,  
    stop:1 rgb(133,133,135),  
    stop:0.4 rgb(31,31,33),  
    stop:0.2 rgb(0,0,150));  
}
```



Эффект нажатия кнопки

```
QPushButton:pressed {  
background: qlineargradient(x1:0, y1:1, x2:0, y2:0,  
    stop:0 rgba(1,1,5,80),  
    stop:0.6 rgba(18,18,212,80),  
    stop:0.5 rgba(142,142,245,80));  
}
```



2D графика. QPainter

Контекст рисования QPaintDevice можно представить себе как поверхность для вывода графики.

QPaintDevice – это основной абстрактный класс для всех классов объектов, которые можно рисовать.



QPainter QWidget::paintEvent()

```
class MyWindow : public QMainWindow {  
protected:  
    virtual void paintEvent(QPaintEvent* e) {  
        QMainWindow::paintEvent(e);  
  
        QPainter p(this);  
        p.setPen(QPen(Qt::red, 2, Qt::DotLine));  
        p.drawLine(0, 0, 100, 100);  
    }  
};
```



QPainter QWidget::paintEvent()

```
16. 
17. int main(int argc, char *argv[])
18. {
19.     QApplication a(argc, argv);
20.     MyWindow w;
21.     w.show();
22.     return a.exec();
23. }
```



QPainter QWidget::paintEvent()

```
...
void MyWidget::paintEvent(QPaintEvent*)
{
    QPainter painter1;
    QPainter painter2;

    painter1.begin(this);
    //Команды рисования
    painter1.end();

    painter2.begin(this);
    //Команды рисования
    painter2.end();
}
```



QPainter QWidget::paintEvent()

```
3 ...
4 void MyWidget::paintEvent(QPaintEvent*)
5 {
6     QPainter painter;
7
8     painter.begin(this); //Контекст виджета
9     //Команды рисования
10    painter.end();
11
12    QPixmap pix(rect());
13    painter.begin(&pix); //контекст растрового изображения
14    //Команды рисования
15    painter.end();
16 }
17 ...
18
```

оставить старые настройки без изменений - QPainter::save()
окончании работы – восстановить с помощью метода
QPainter::restore().



QPainter Перо

Установить новое перо можно с помощью метода `QPainter ::setPen()`, передав в него объект класса `QPen`. Можно передавать и предопределенные стили пера, указанные в

	Толщина линий			
	1	2	3	4
NoPen				
SolidLine	---	----	---	----
DashLine	— — — —	-----	— — — —	— — — —
DotLine	· · · · ·	· · · · ·	· · · · ·	· · · · ·
DashDotLine	— - — - —	--- · ---	— - — - —	— - — - —
DashDotDotLine	— - - - -	--- · · · ·	— - - - -	— - - - -

`QPen::setWidth()`.



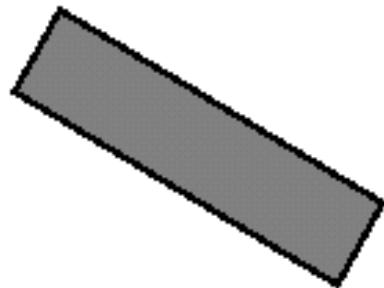
QPainter Перо

```
QPainter painter(this);
painter.setPen(QPen(Qt::red, 3, Qt::DashLine));
```

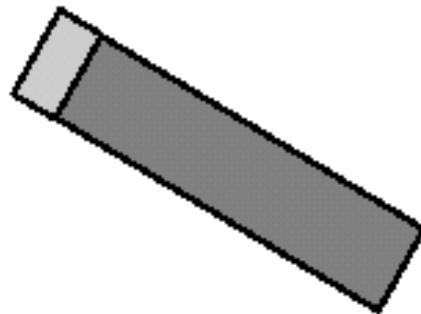


QPainter

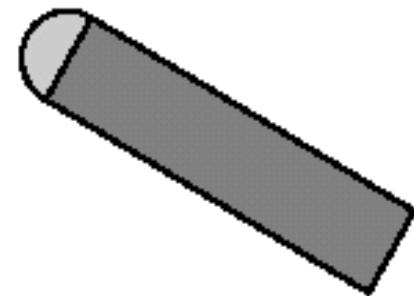
Перо



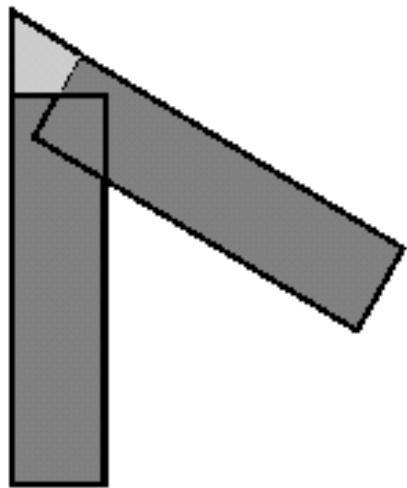
FlatCap



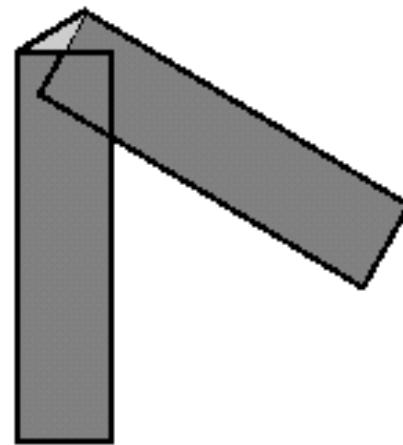
SquareCap



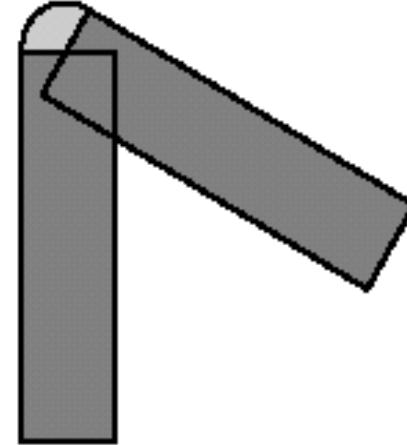
RoundCap



MiterJoin



BevelJoin

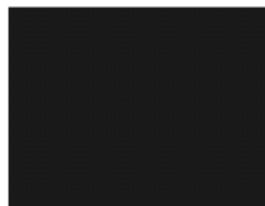


RoundJoin



QPainter Кисть

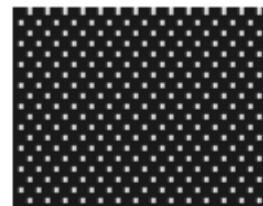
Установить кисть можно методом setBrush() , передав в него объект класса QBrush или один из пре-



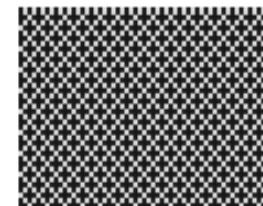
SolidPattern



Dense1Pattern



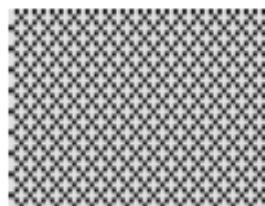
Dense2Pattern



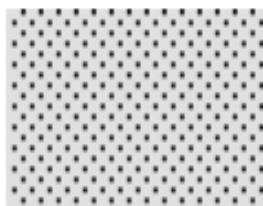
Dense3Pattern



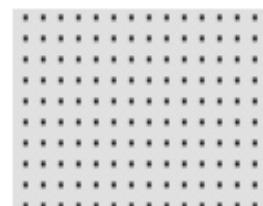
Dense4Pattern



Dense5Pattern



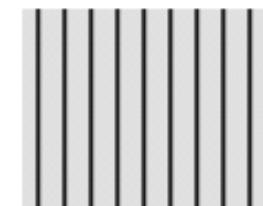
Dense6Pattern



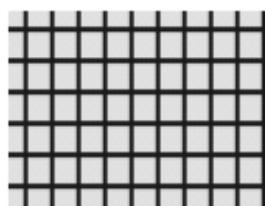
Dense7Pattern



HorPattern



VerPattern



CrossPattern



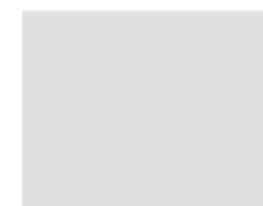
BDiagPattern



FDiagPattern



DiagCross-



NoBrush



QPainter Кисть

```
QPainter painter(this);
painter.setBrush(QBrush(Qt::red, Qt::HorPattern));
```



QPainter Кисть

```
void MyWidget::paintEvent(QPaintEvent*)  
{  
    QPainter painter(this);  
    QPixmap pix(":/Alina.jpg");  
    painter.setBrush(QBrush(Qt::black, pix));  
    painter.drawEllipse(0,0,300,150);  
}
```

QPainter Рисование

Например, вызовом метода `drawRect()` можно нарисовать прямоугольник. И так далее:

(x_1, y_1)

(x_2, y_2)

`drawLine()`

p_2

p_1

p_3

p_4

p_2

p_1

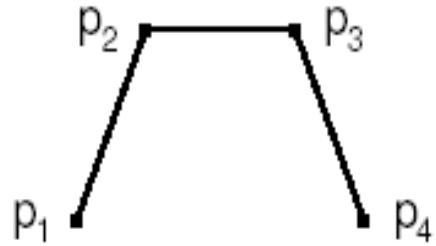
p_3

p_4

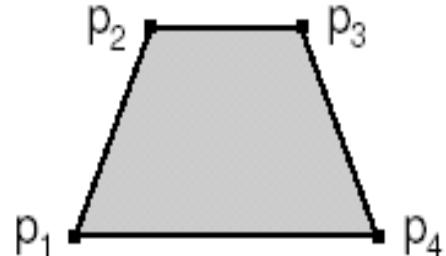
`drawLineSegments()`



`drawCubicBezier()`



`drawPolyline()`

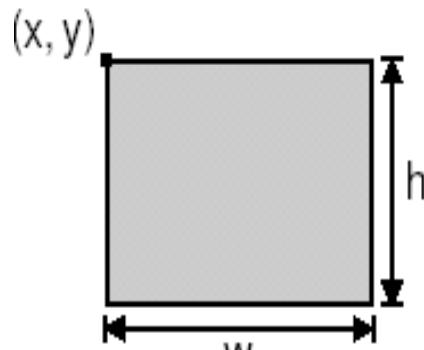


`drawPolygon()`

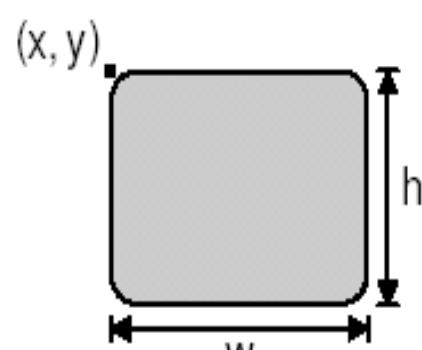


QPainter Рисование

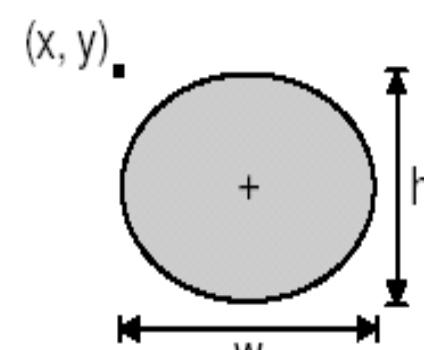
Например, вызовом метода `drawRect()` можно нарисовать прямоугольник. И так далее:



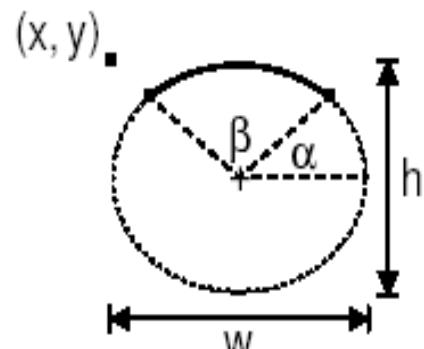
`drawRect()`



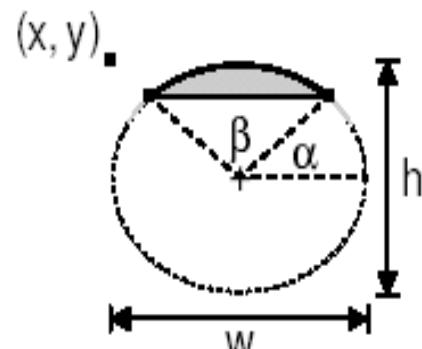
`drawRoundRect()`



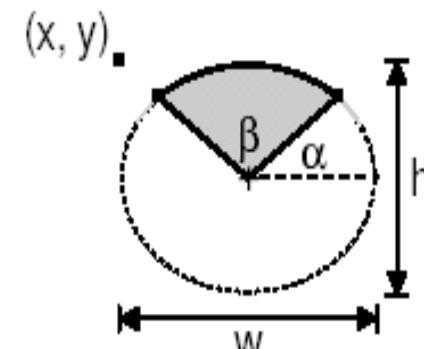
`drawEllipse()`



`drawArc()`



`drawChord()`



`drawPie()`



QPainter Рисование

QPainter::setBrush() нужно установить значение стиля кисти QBrush::NoBrush. QPainter::setPen() установить стиль пера QPen::NoPen. В метод рисования передаются координаты верхнего левого угла, ширина и высота.

```
QPainter p(this);
p.setPen(QPen(Qt::black, 3, Qt::DashDotLine));
p.setBrush(QBrush(Qt::green, Qt::SolidPattern));
p.drawEllipse(20, 20, 100, 60);
```



QPainter Рисование

```
class MyButton : public QPushButton {  
protected:  
    virtual void paintEvent(QPaintEvent* e) {  
        QPushButton::paintEvent(e);  
        QPainter p(this);  
        p.setPen(QPen(Qt::black, 1, Qt::SolidLine));  
        p.setBrush(QBrush(Qt::green, Qt::SolidPattern))  
        p.drawEllipse(10, 7, 10, 6);  
    }  
}
```

QPainter Рисование

```
6
7
8 int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MyButton b;
    b.show();
    return a.exec();
}
```



QPainter Техника сглаживания

Режим сглаживания распространяется на отображение текста и геометрических фигур. Его можно включить в объекте класса QPainter при помощи метода setRenderHint():

```
painter.setRenderHint(QPainter::Antialiasing, true);
```



Контекстно-независимое представление растровых изображений

Данные растрового изображения помещаются в обычный массив, что дает возможность эффективного обращения к каждому из пикселов в отдельности, а также позволяет эффективно производить операции записи и считывания файлов растровых изображений.



Класс QImage

Этот класс унаследован от класса контекста рисования QPaintDevice

Данные растрового изображения хранятся в объектах данного класса построчно, и в каждой строке пиксели расположены слева направо.

scanLine() адрес строки, номер которой соответствует значению, переданному в этот метод. Номер должен быть от 0 до $h-1$, где h - высота изображения.



Класс QImage

`Qimage img(320, 240, QImage::Format_RGB32) //32- глубина цвета.`

<code>QImage::Format_Invalid</code>	Растровое изображение недействительно
<code>QImage::Format_Mono</code>	Каждый пиксель представлен одним битом. Биты укомплектованы в байте таким образом, что первый является старшим разрядом
<code>QImage::Format_MonoLSB</code>	Каждый пиксель представлен одним битом. Биты укомплектованы в байте таким образом, что первый является младшим разрядом
<code>QImage::Format_Index8</code>	данные представляют собой 8-битные индексы цветовой палитры;



Класс QImage

`Qimage img(320, 240, QImage::Format_RGB32) //32- глубина цвета.`

<code>QImage::Format_RGB32</code>	каждый пиксель представлен 32 битами (интенсивности красного, зелёного и синего, плюс значение альфа-канала, всегда равное 0xFF, то есть, прозрачность не поддерживается);
<code>QImage::Format_ARGB32</code>	каждый пиксель представлен 32 битами
<code>QImage::Format_ARGB32_Premultiplied</code>	Практически идентичен формату ARGB32, но код оптимизирован для использования объекта QImage в качестве контекста рисования



Класс QImage

```
QImage img (“D:\\I.jpg”);
```

```
QImage img;  
img.load(“D:\\I.jpg”);
```

Первым параметром передается имя файла, вторым формат. Формат файла обозначается строкой типа `unsigned char*`, принимающей одно из следующих значений: `GIF`, `BMP`, `JPG`, `XMP`, `XBM` или `PNG`. Если формат во втором параметре не указать, то класс `QImage` попытается распознать его самостоятельно.



Класс **QImage**

```
QImage img(320, 240, 32, QColor::blue);
img.save("I.jpg","JPG",85);
```

С помощью метода **save()** можно сохранять растровое изображение из объекта класса **QImage** в файл. Первым параметром передается имя файла, вторым –формат для сохранения, третьим параметром можно передать значение качества. Оно варьируется в диапазоне от 0 до 100.



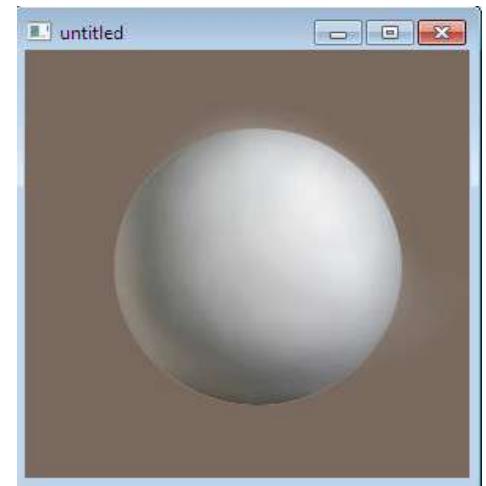
QPainter::drawImage().

```
3 #include <QPainter>
4 #include <QDebug>
5
6
7 class MyWindow: public QMainWindow{
8 protected:
9     virtual void paintEvent(QPaintEvent * e) {
10         QMainWindow::paintEvent(e);
11         QPainter painter(this);
12         QImage img("E:\\1.jpg");
13         painter.drawImage(0,0,img);
14     }
15 }
```



QPainter::drawImage().

```
6  
7  
8 int main(int argc, char *argv[])  
9 {  
0     QApplication a(argc, argv);  
1     MyWindow w;  
2     QImage img("E:\\1.jpg");  
3     w.setGeometry(50, 50, img.width(), img.height());  
4     w.show();  
5     return a.exec();  
6 }
```



QPainter::drawImage()



Контекстно-независимое представление растровых изображений

```
fill();
```

```
QImage img("E:\\1.jpg");
QRgb value;
value = qRgb(122, 163, 39);
img.fill(value);
painter.drawImage(0, 0, img);
```



Контекстно-независимое представление растровых изображений

pixel(x,y)

```
QRgb rgb = img.pixel(250,100);
```

```
setPixel(x,y,rgb);
```

```
QRgb rgb = qRgb(200,100,0);  
img.setPixel(20,50,rgb);
```



Контекстно-независимое представление растровых изображений

```
painter.drawImage(0,0,img,60,60,100,50);
```



invertPixels()

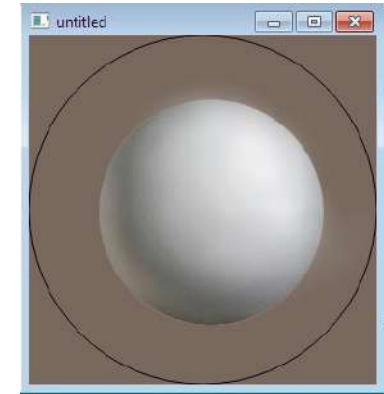
scaled()

mirrored()



Контекстно-независимое представление растровых изображений

```
class MyWindow: public QMainWindow{  
protected:  
    virtual void paintEvent(QPaintEvent * e) {  
        QMainWindow::paintEvent(e);  
        QImage img("E:\\1.jpg");  
        QPainter painter;  
        painter.begin(&img);  
        painter.initFrom(this);  
        painter.setRenderHint(QPainter::Antialiasing,true);  
    }  
};
```



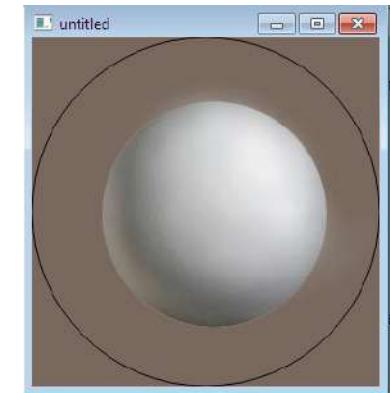
Контекстно-независимое представление растровых изображений

```
painter.drawEllipse(0, 0, img.width(), img.height());  
painter.end();
```

```
painter.begin(this);  
painter.drawImage(0, 0, img);  
painter.end();
```

```
}
```

```
;
```



Контекстно-зависимое представление растровых изображений

QPixmap

```
QPixmap pix(300,300);
```

```
QPixmap::defaultDepth()
```

```
QPixmap pix("E:\\1.jpg");
```

```
load()    save()
```



Контекстно-зависимое представление растровых изображений

QPainter::drawPixmap()

```
QPainter painter(this);
QPixmap pix("D:\\q\\pict\\untitled\\1.jpg");
painter.drawPixmap(0,0,pix);
```



```
QPainter painter(this);
QPixmap pix("D:\\q\\pict\\untitled\\1.jpg");
QRect r(0,0, pix.width(), pix.height()/3);
painter.drawPixmap(r,pix);
```



Контекстно-зависимое представление растровых изображений

QPixmapCache

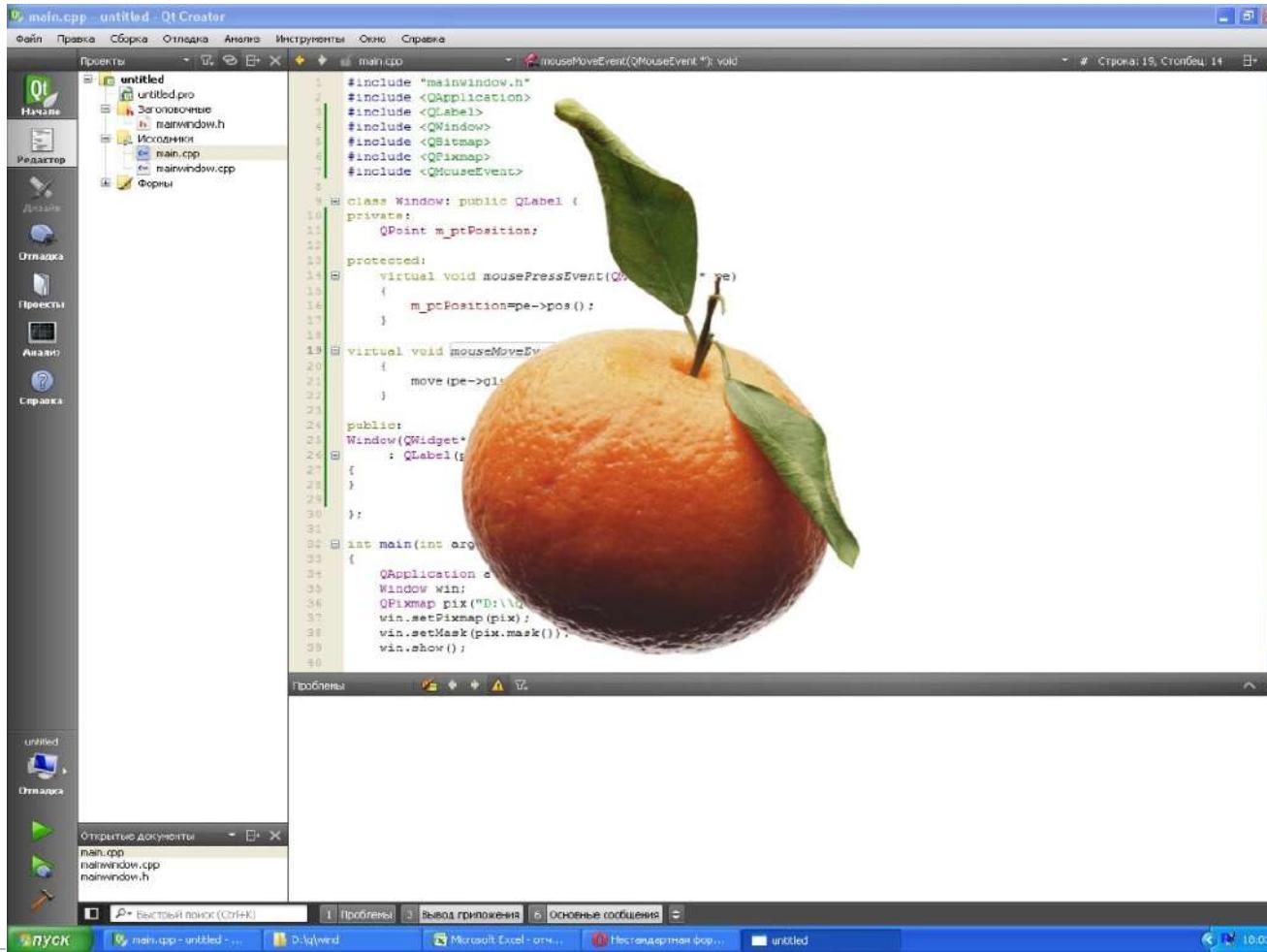
insert()

find()



Контекстно-зависимое представление растровых изображений

setMask()



Контекстно-зависимое представление растровых изображений

```
30
31 int main(int argc, char *argv[])
32 {
33     QApplication a(argc, argv);
34     Window win;
35     QPixmap pix("D:\\q\\wind\\3.png");
36     win.setPixmap(pix);
37     win.setMask(pix.mask());
38     win.show();
39
40     return a.exec();
```



```
3 class Window: public QLabel {  
4     private:  
5         QPoint m_ptPosition;  
  
6     protected:  
7         virtual void mousePressEvent(QMouseEvent* pe)  
8         {  
9             m_ptPosition=pe->pos();  
10        }  
11}
```



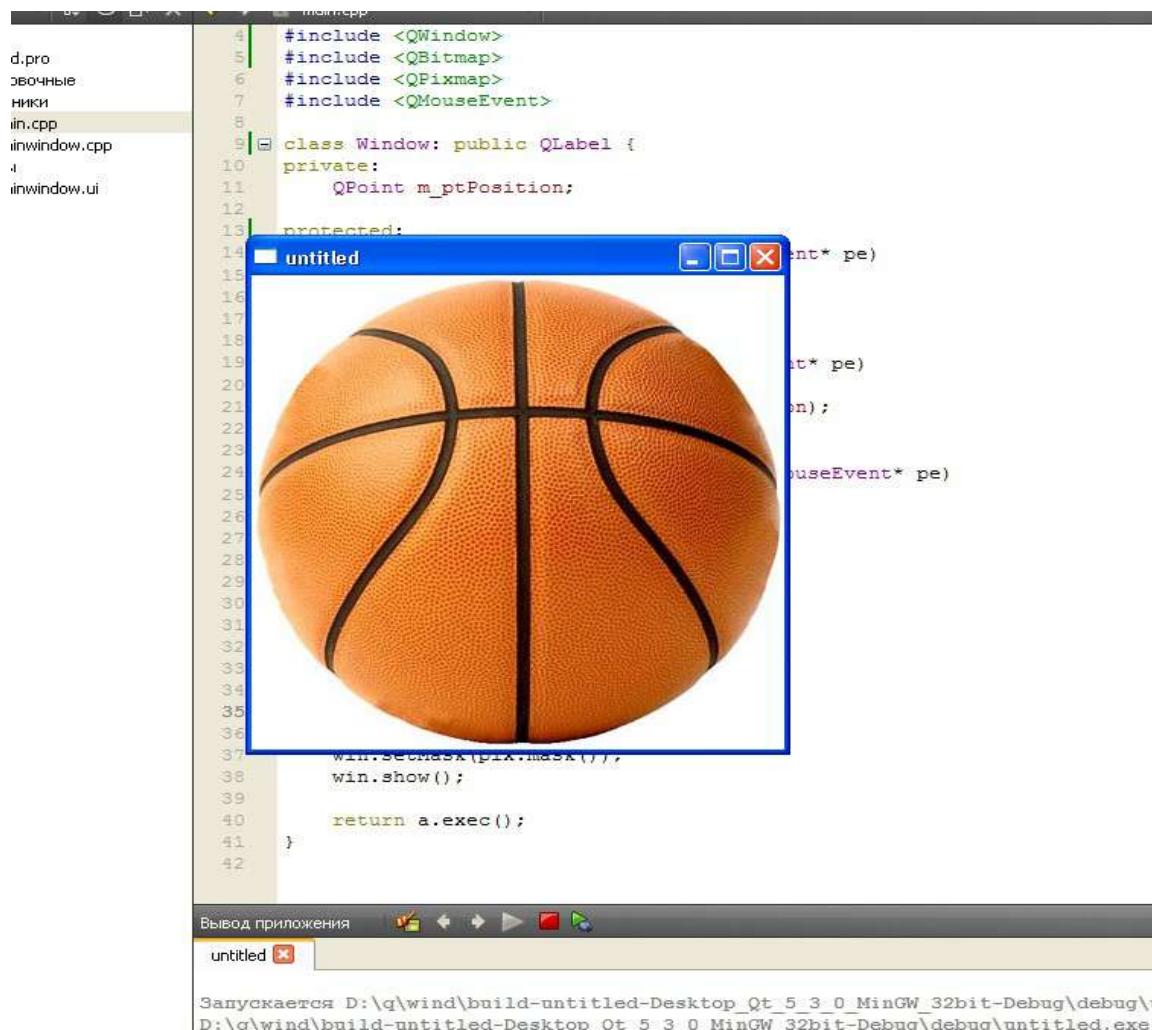
```
virtual void mouseMoveEvent(QMouseEvent* pe)
{
    move(pe->globalPos() - m_ptPosition);
}

virtual void mouseDoubleClickEvent(QMouseEvent* pe)
{
    exit(0);
}

};
```



Контекстно-зависимое представление растровых изображений



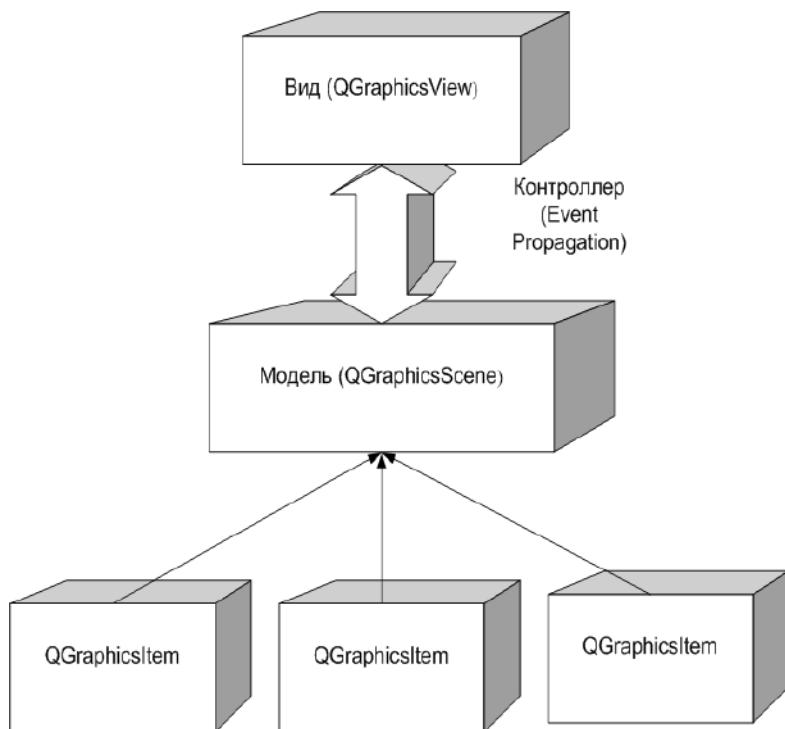
Контекстно-зависимое представление растровых изображений

```
public:  
Window(QWidget* pwgt = 0)  
    : QLabel(pwgt, Qt::FramelessWindowHint |  
Qt::Window)  
{}
```



Графическое представление

Графическое представление – это инструмент для управления и взаимодействия с большим количеством элементов двумерных изображений, включая их визуальное увеличение/уменьшение и поворот. Кроме того, оно берет на себя также и обнаружение столкновений.



Графическое представление

Модель данных, сцена, реализована с помощью объекта класса **QGraphicsScene**. Элементами модели данных являются графические примитивы (геометрические фигуры и растровые изображения).

Все графические примитивы реализованы с помощью классов-потомков класса **QGraphicsItem**.

Графическая сцена — это данные, для отображения которых используется вид (**QGraphicView**).



Сцена

Объект сцены – представляет собой контейнер, содержащий в себе объекты, которые созданы от классов, наследующих `QGraphicsItem`. Эти объекты являются данными без графического представления. Элементы добавляются на сцену при помощи метода `addItem()`.

oval (методом `addEllipse()`),
текст (методом `addText()`),
линия (методом `addLine()`)

`items()`

`itemAt()` возвращает самый верхний элемент, находящийся на заданных координатах.

Представление

mainwindow.ui @ untitled - Qt Creator

Правка Сборка Отладка Анализ Инструменты Окно Справка

mainwindow.ui*

Фильтр

- Double Spin Box
- Time Edit
- Date Edit
- Date/Time Edit
- Dial
- Horizontal Scroll Bar
- Vertical Scroll Bar
- Horizontal Slider
- Vertical Slider
- Key Sequence Edit
- Display Widgets
- Label
- Text Browser
- Graphics View
- Calendar Widget

Пишите здесь

Объект Класс

centralWidget	QWidget
graphicsView	QGraphicsView
menuBar	QMenuBar

Фильтр

graphicsView : QGraphicsView

Свойство Значение

QObject	
objectName	graphicsView
QWidget	
enabled	<input checked="" type="checkbox"/>
geometry	[15, 11, 341 x 151]
X	15
Y	11
Ширина	341
Высота	151
sizePolicy	[Expanding, Expanding, 0, 0]
Горизонтальная ...	Expanding
Вертикальная по ...	Expanding

Элемент

- Установка местоположения методом **setPos();**
- Скрытие и показ: методы **hide()** и **show();**
- Установка доступного/недоступного состояния с помощью метода **setEnabled();**
- Трансформация: производится при помощи метода **setTransformation(),** в который передается объект класса, ответственного за трансформации (Класс **QTransform**);
- Перерисовка – **paint().**



Пример

A screenshot of a Qt application window titled "untitled". Inside the window, there is a large green rectangle and a brown house icon with a red roof and blue windows. A red line connects the top-left corner of the green rectangle to the word "move" located near the bottom-left of the house icon. The application window has a standard title bar with minimize, maximize, and close buttons. The background of the application window is white.

Пример

```
{  
    QApplication app(argc, argv);  
    QGraphicsScene scene ( QRectF(-100, -100, 300, 300) );  
    QGraphicsView view(&scene);
```



Пример

```
QGraphicsRectItem* pRectItem = scene.addRect(QRectF(-30, -30, 120, 80),  
                                         QPen(Qt::black),  
                                         QBrush(Qt::green)  
                                         );  
  
pRectItem->setFlags(QGraphicsItem::ItemIsMovable);
```



Пример

```
QGraphicsPixmapItem* pPixmapItem = scene.addPixmap(QPixmap("house.jpg"));
pPixmapItem->setFlags(QGraphicsItem::ItemIsMovable);

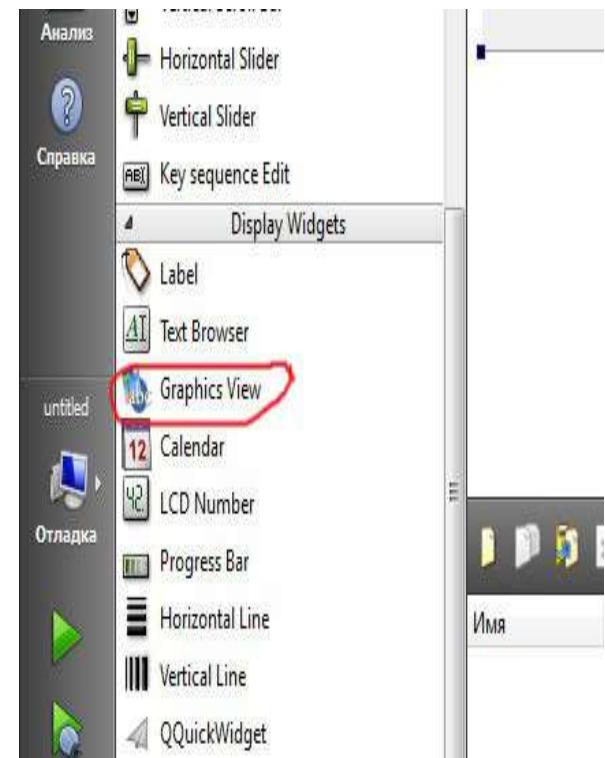
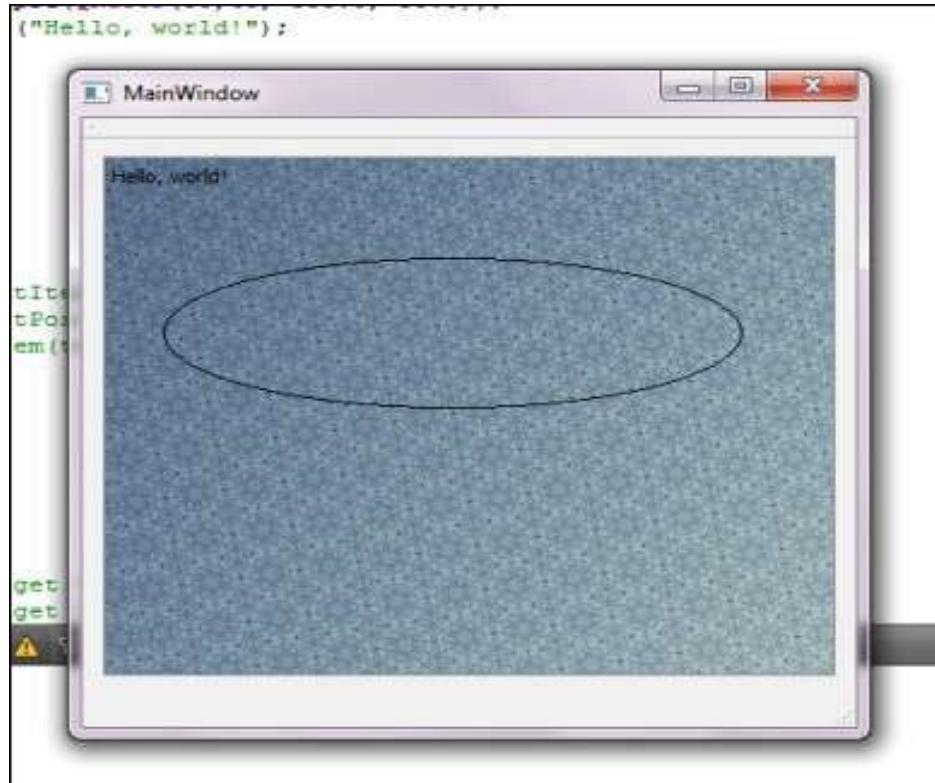
QGraphicsTextItem* pTextItem = scene.addText("move");
pTextItem->setFlags(QGraphicsItem::ItemIsMovable);

QGraphicsLineItem* pLineItem = scene.addLine(QLineF(-10, -10, -80, -80),
                                             QPen(Qt::red,2));
pLineItem->setFlags(QGraphicsItem::ItemIsMovable);

view.show();
return app.exec();
```



Пример



```
QGraphicsScene * scene = new QGraphicsScene;  
ui->graphicsView->setScene(scene);
```



Пример

```
8  
9 MainWindow::MainWindow(QWidget *parent) :  
10     QMainWindow(parent),  
11     ui(new Ui::MainWindow)  
12 {  
13     ui->setupUi(this);  
14     QGraphicsScene * scene = new QGraphicsScene;  
15     ui->graphicsView->setScene(scene);  
16     scene->addPixmap(QPixmap("C:\\Qt\\MyProjects\\scene\\1.jpg"));  
17     scene->addEllipse(QRectF(30,60, 300.0, 90.0));  
18     scene->addText("Hello, world!");
```

Пример

addPixmap(const QPixmap &*pixmap*)

addEllipse(qreal *x*, qreal *y*, qreal *w*, qreal *h*, const QPen & *pen* = QPen(), constQBrush & *brush* = QBrush())

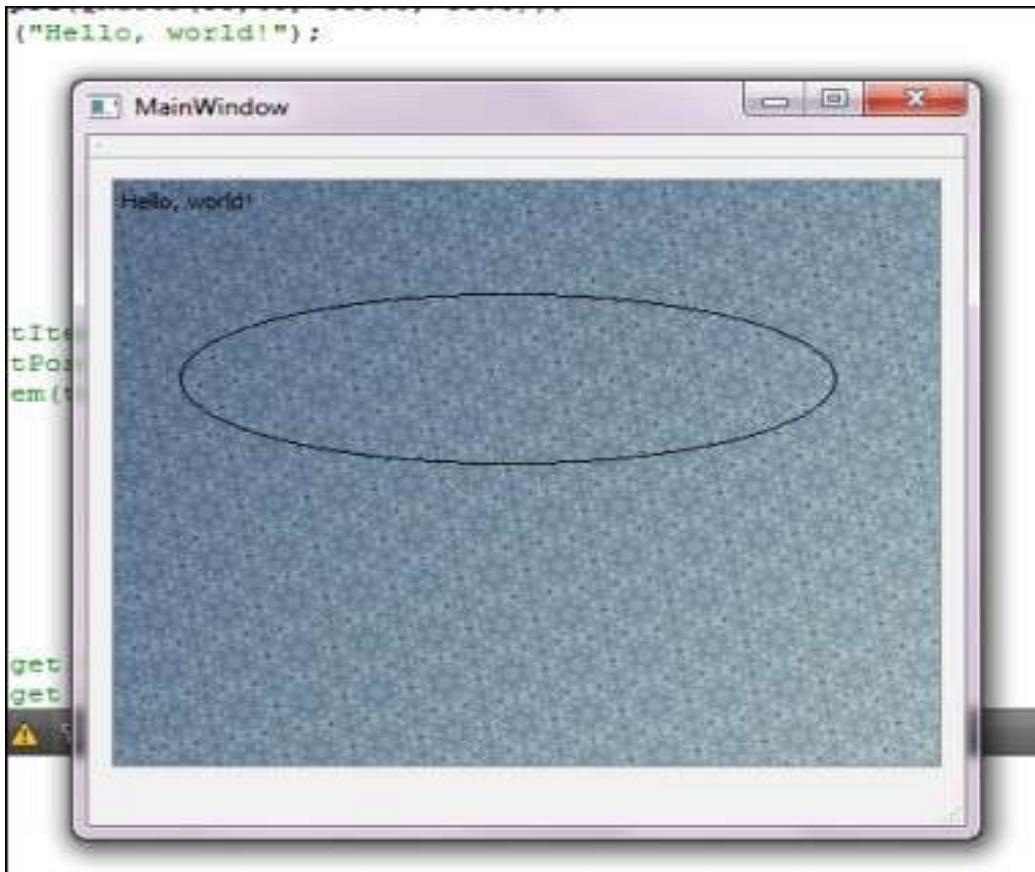
QRectF(qreal *x*, qreal *y*, qreal *width*, qreal *height*)

определяет прямоугольник (QRectF) внутри которого расположен наш элемент. Этот метод нужен для того, чтобы графическая сцена могла вовремя понять, что элемент необходимо перерисовать;

addText(const QString & *text*, const QFont & *font* = QFont())



Пример



Координаты, которые мы указали при добавлении эллипса, являются **координатами модели**, а не графического окна. При отображении модели объектом QGraphicsView они будут автоматически переведены в **координаты окна** QGraphicsView.

Пример

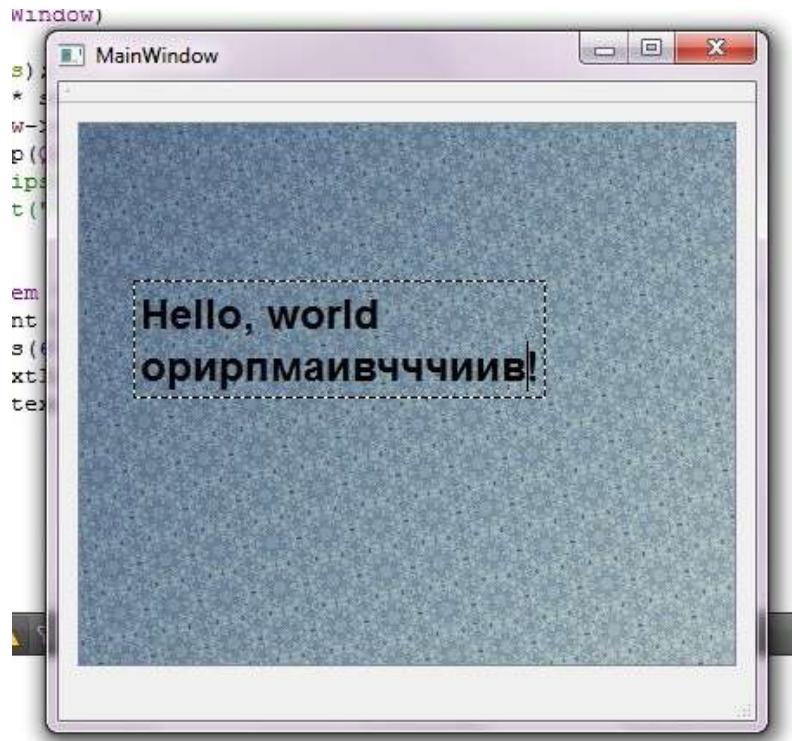
```
QGraphicsTextItem *textItem = new QGraphicsTextItem("Hello, world!");
textItem->setFont(QFont("Times", 18, QFont::Bold));
textItem->setPos(67, 90);
scene->addItem(textItem);
```



Пример

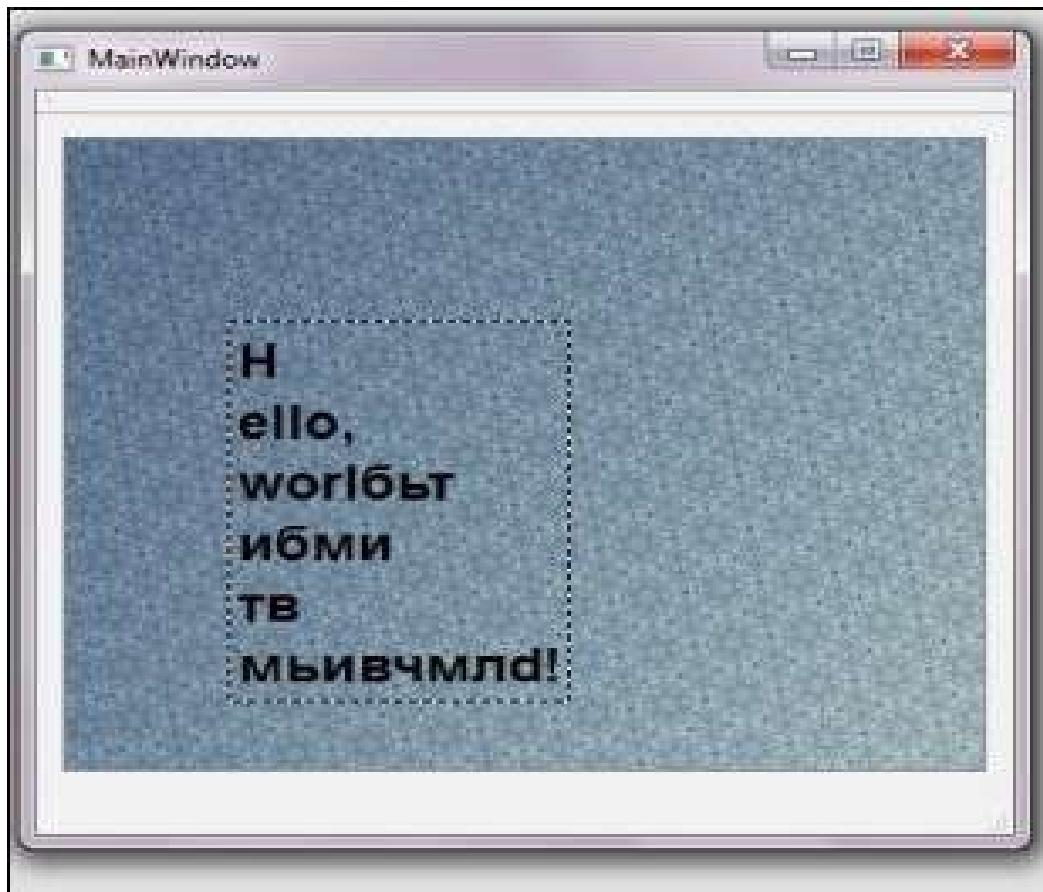


Пример



`textItem->setTextInteractionFlags(Qt::TextEditable);`

Пример



```
textItem->setTextWidth(10);
```



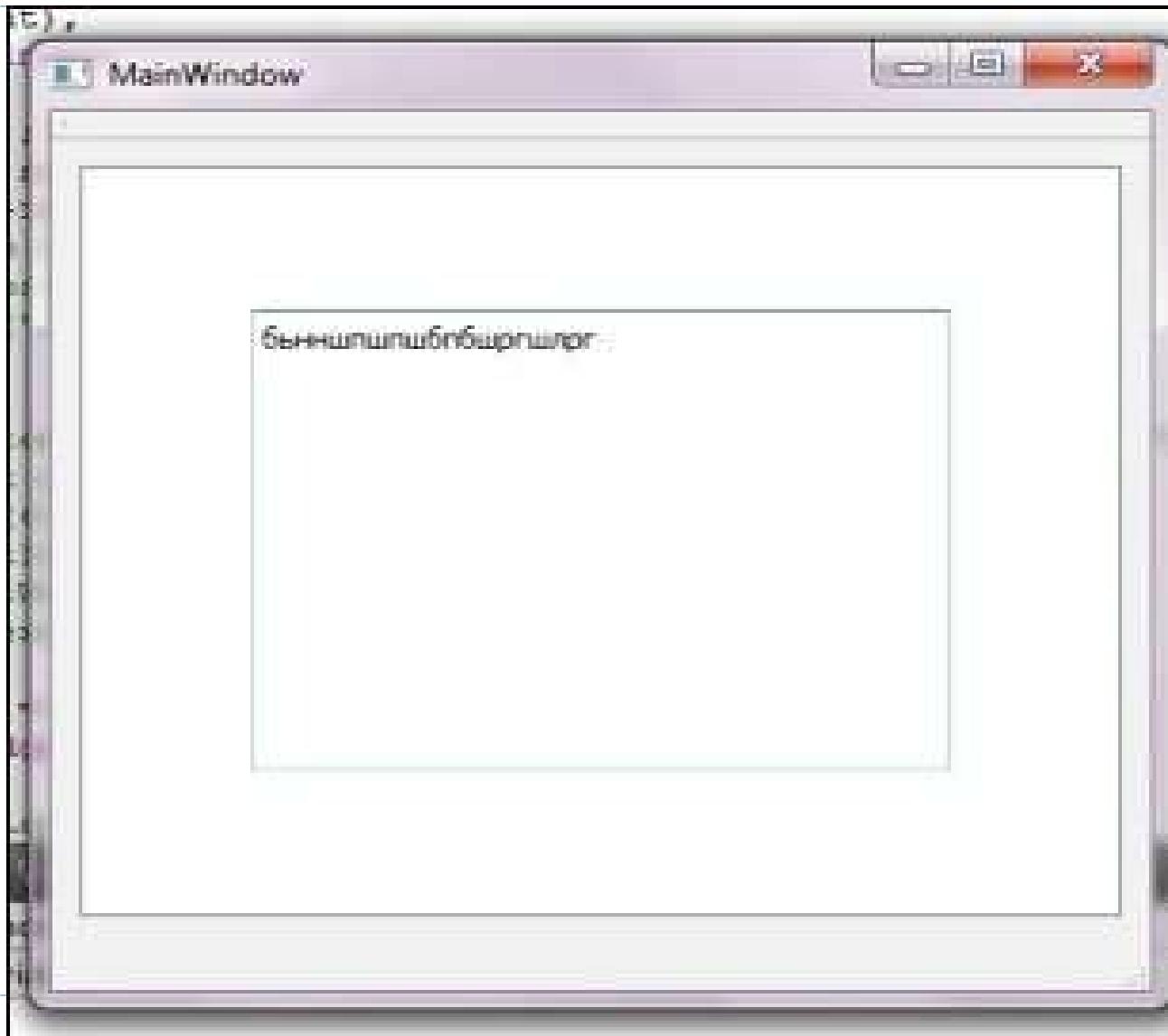
Добавление виджета

addWidget(QWidget * *widget*,Qt::WindowFlags *wFlags* = 0)

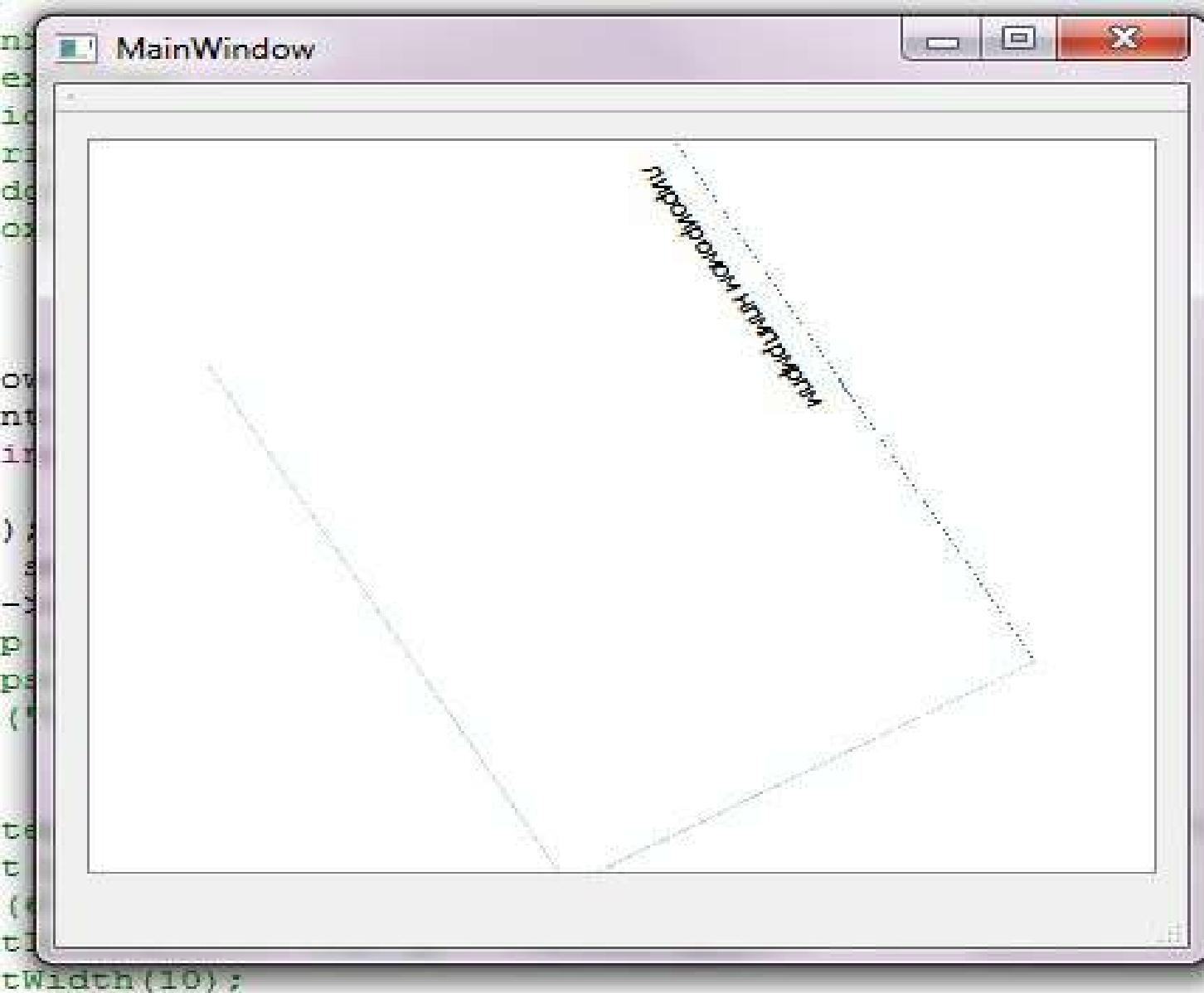
```
QTextEdit *tEdit = new QTextEdit;  
QGraphicsProxyWidget *proxy = scene->addWidget(tEdit);
```



Добавление виджета



Добавление виджета



Трансформация виджета

```
QTextEdit *tEdit = new QTextEdit;
QGraphicsProxyWidget *proxy = scene->addWidget(tEdit);

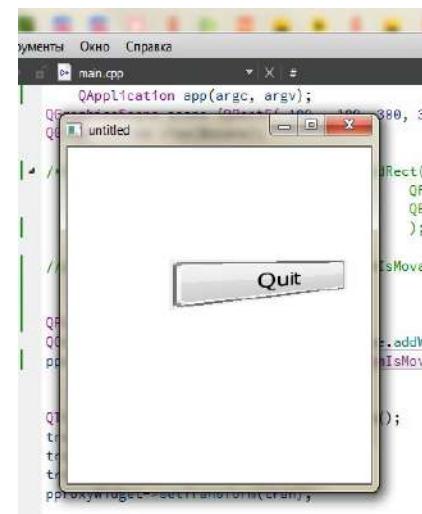
QTransform transform = proxy->transform();
transform.translate(50., 30.);
transform.rotate(60.0);
proxy->setTransform(transform);
```



Добавление и трансформация виджета

```
QPushButton cmd("Quit");
QGraphicsProxyWidget* pproxyWidget = scene.addWidget(&cmd);

QTransform tran = pproxyWidget->transform();
tran.rotate(-45,Qt::YAxis);
tran.rotate(-45,Qt::YAxis);
tran.scale(8,2);
pproxyWidget->setTransform(tran);
```



Создание игры

```
9  class GameScene : public QGraphicsScene  
10 {  
11     Q_OBJECT  
12 public:  
13     GameScene(QObject *parent = 0);  
14 protected:  
15     virtual void keyPressEvent(QKeyEvent * keyEvent);  
16 private:  
17     QGraphicsPixmapItem * Cat;  
18     QGraphicsPixmapItem * Dog;  
19     QGraphicsPixmapItem * home;  
20     int R;
```



Создание игры

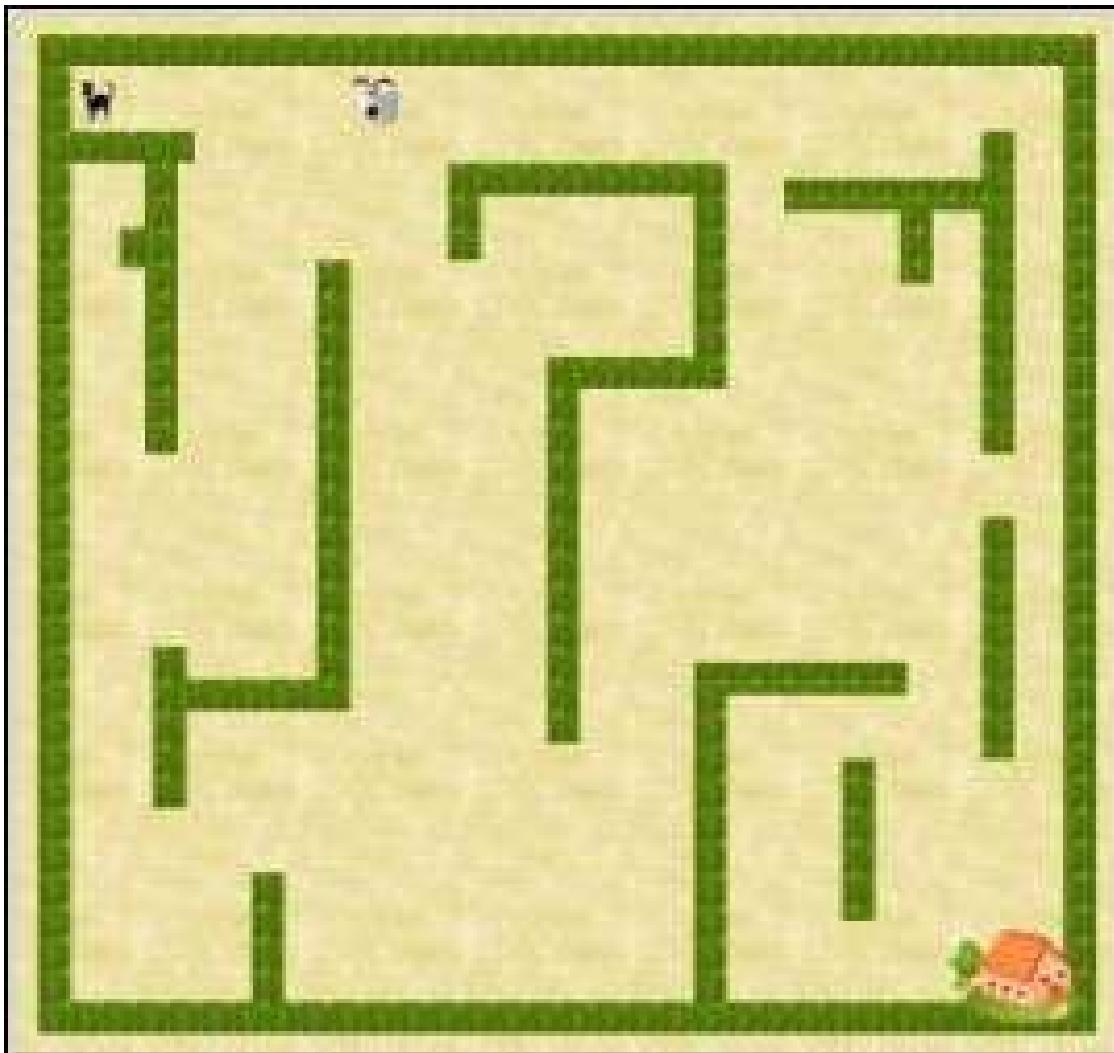
```
void makeWalls();  
void The_End();  
void Win();  
QGraphicsItem * itemCollidesWith(QGraphicsItem * item);  
  
public slots:  
    void DogGo();  
    void Rand_dog();  
};
```



Создание игры

```
9  
10 MainWindow::MainWindow(QWidget *parent) :  
11     QMainWindow(parent),  
12     ui(new Ui::MainWindow)  
13 {  
14     ui->setupUi(this);  
15     GameScene * scene = new GameScene;  
16     ui->graphicsView->setScene(scene);  
17 }  
18  
19 }
```

Создание игры



Game_Scene.cpp

```
2
3 GameScene::GameScene(QObject *parent) : QGraphicsScene(parent)
4 {
5     makeWalls();
6     Cat = addPixmap(QPixmap("C:\\Qt\\MyProjects\\Labirint\\untitled\\black_cat.png"));
7     QTransform tran = Cat->transform();
8     tran.translate(25.5, 25.5);
9     Cat->setTransform(tran);
10    Cat->setData(0, "Cat");
```



Движение собаки

```
34     setBackgroundBrush(QBrush(QColor(255,255,255), QPixmap("C:\\Qt\\MyProject\\Icons\\background.jpg"));
35
36     qrand(QTime(0,0,0).secsTo(QTime::currentTime()));
37     Rand_dog();
38     QTimer * timer = new QTimer(this);
39     connect(timer, SIGNAL(timeout()), this, SLOT(DogGo()));
40     timer->start(100);
41
42     QTimer * timer2 = new QTimer(this);
43     connect(timer2, SIGNAL(timeout()), this, SLOT(Rand_dog()));
44     timer2->start(5000);
45
46 }
```



makeWalls()

```
85 void GameScene::makeWalls()
86 {
87     float walls[23][4] = {{0, 0, 20, 620},
88                           {20, 0, 620, 20},
89                           {625, 0, 20, 620},
90                           {20, 700, 620, 20},
91                           * * * * *
92
93                           25), 10, 150, 10),
94                           (400, 80, 20, 120),
95                           (525, 105, 20, 50)};
96
97     QBrush brush(QColor(255, 255, 255), QPixmap("C:\\Qt\\MyProjects\\Labirint\\untitled\\grass.jpg"));
98     QPen pen(Qt::NoPen);
99     for (int i = 0; i < 23; i++) {
100         QGraphicsItem * item = addRect(QRectF(walls[i][0], walls[i][1], walls[i][2], walls[i][3]), pen, brush);
101         item->setData(0, "Wall");
102     }
103 }
```

keyPressEvent(),

```
48
49 void GameScene::keyPressEvent(QKeyEvent * keyEvent)
50 {
51     QPointF np;
52     np.setX(0);
53     np.setY(0);
54     switch (keyEvent->key()) {
55         case Qt::Key_Left:
56             np.setX(-10);
57             break;
58         case Qt::Key_Right:
59             np.setX(10);
60             break;
```

keyPressEvent(),

```
    np.setX(10);
    break;
    case Qt::Key_Up:
        np.setY(-10);
        break;
    case Qt::Key_Down:
        np.setY(10);
        break;
}
QTransform tran = Cat->transform();
tran.translate(np.x(), np.y());
Cat->setTransform(tran);
```

keyPressEvent(),

```
QGraphicsItem * obstacle = itemCollidesWith(Cat);
if (obstacle) {
    if (obstacle->data(0) == "Wall") {
        tran.translate(-np.x(), -np.y());
        Cat->setTransform(tran);
    }
    else
        if (obstacle->data(0) == "Dog") The_End();
            else if (obstacle->data(0) == "Home") Win();
}
}
```



itemCollidesWith()

В основе функции `itemCollidesWith()` лежит метод **`collidingItems()`** класса `QGraphicsScene`. Этот метод возвращает список примитивов, находящихся в состоянии столкновения с тем примитивом, который был передан методу в качестве параметра (под столкновением понимается частичное или полное перекрытие примитивов в системе координат сцены).



itemCollidesWith()

```
QGraphicsItem * GameScene::itemCollidesWith(QGraphicsItem * item)
{
    QList<QGraphicsItem *> collisions = collidingItems(item);
    foreach (QGraphicsItem * it, collisions) {
        if (it == item)
            continue;
        return it;
    }
    return NULL;
}
```

Координата Z определяет, какой из примитивов будет виден, если несколько примитивов частично или полностью перекрываются. Кроме того от значения третьей координаты зависит порядок, в котором располагаются примитивы в списке, возвращаемом методом collidingItems()

DogGo()

```
140 void GameScene::DogGo()
141 {
142     QPointF np;
143     np.setX(0);
144     np.setY(0);
145
146     switch (R)
147     {
148         case 1 : np.setX(-10); break; //left
149         case 2 : np.setX(+10); break; //right
150         case 3 : np.setY(-10); break; //top
151         case 4 : np.setY(+10); break; //bot
152     }
153
154     qDebug() << R;
155     QTransform tran = Dog->transform();
156     tran = Dog->transform();
157     tran.translate(np.x(), np.y());
158     Dog->setTransform(tran);
159 }
```

DogGo()

```
QGraphicsItem *obstacle = itemCollidesWith(Dog);
tran = Dog->transform();
if (obstacle) {
    if ((obstacle->data(0) == "Wall") || ((obstacle->data(0) == "Home")))
        tran.translate(-np.x(), -np.y());
    Dog->setTransform(tran);
    Rand_dog();
}
else
    if (obstacle->data(0) == "Cat") The_End();
```



Человеко-машинное взаимодействие

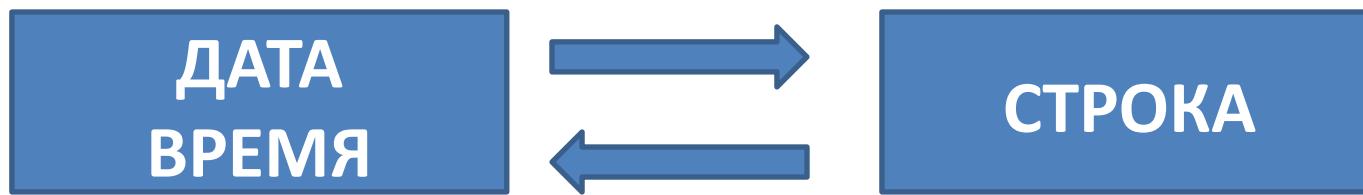
Лекция 6

Мерзлякова Екатерина Юрьевна

к.т.н. доцент ПМИК

ДАТА И ВРЕМЯ В QT

- **QDate, QTime и QDateTime**



ДАТА И ВРЕМЯ В QT



Внешние прерывания — это прерывания, вызываемые асинхронными событиями, например, устройствами ввода/вывода или самим устройством таймера.

Если программа занята интенсивными вычислениями, то события таймера могут быть обработаны по окончании процесса вычисления. При выходе из приложения таймеры автоматически уничтожаются.

ДАТА И ВРЕМЯ В QT: КЛАСС ДАТЫ

QDate

год

месяц

день

создадим объект, который будет содержать
дату 15 октября 2014:

QDate date(2014, 10, 15);

QDate date;

date.setDate(2014, 10, 15);

ДАТА И ВРЕМЯ В QT: КЛАСС ДАТЫ

`year()`

•возвращает целый год в диапазоне от 1752 до 8000;

`month()`

•возвращает целое значение месяца в диапазоне от 1 до 12 (с января по декабрь);

`day()`

•возвращает день месяца в диапазоне от 1 до 31.

`daysInMonth()`

•узнать количество дней в месяце

`daysInYear ()`

•количество дней в году

`toString()`

•получить текстовое представление даты

`dayOfWeek()`

• получить день недели

ДАТА И ВРЕМЯ В QT: КЛАСС ДАТЫ

Как задать собственный формат времени

```
QDate date(2014, 10, 15);  
QString str;  
str = date.toString("d.M.yy");  
//str - "3.7.14"  
  
str = date.toString("dd/MM/yy");  
//str - "03/07/14"  
  
str = date.toString("yyyy.MMM.ddd");  
//str = "2014.июл.Суб"  
  
str =  
date.toString("yyyy.MMMM.dddddd"); //str =  
"2014.Июль.суббота"
```

ДАТА И ВРЕМЯ В QT: КЛАСС ДАТЫ

addDays ()

addMonths ()

получить ~~измененную~~ измененную дату, добавив или отняв от нее дни/месяца/года

```
QDate date(2007, 1, 3);  
QDate date2 = date.addDays(-7);  
QString str = date2.toString("dd/MM/yy");  
//str ="27/12/06"
```

ДАТА И ВРЕМЯ В QT: КЛАСС ДАТЫ

fromString()

- обратное преобразование из строкового типа к типу **QDate**

currentDate()

- получение текущей даты

daysTo()

- узнать разницу в днях между двумя датами

Пример: определить количество дней от текущей даты до Нового года:

```
QDate dateToday = QDate::currentDate();
QDate dateNewYear(dateToday.year(), 12, 31);
qDebug() << "Осталось " <<
dateToday.daysTo(dateNewYear) <<
" дней до Нового года";
```

ДАТА И ВРЕМЯ В QT: КЛАСС ДАТЫ

Сравнение объектов дат с помощью операторов
==, !=, <, <=, > и >=.

Например:

```
QDate date1(2007, 1, 3);  
QDate date2(2007, 1, 5);  
bool b = (date1 == date2); //b =  
false
```

ДАТА И ВРЕМЯ В QT: КЛАСС ВРЕМЕНИ

QTime

часы

минуты

секунды (0)

миллисекунды (0)

Операции сравнения

`==`, `!=`, `<`, `<=`, `>` или `>=`

Точность – миллисекунды

Ограничение 24-часовым
интервалом

```
QTime time(20, 4);
```

Или

```
QTime time; time.setHMS  
(20, 4, 23, 3);
```

ДАТА И ВРЕМЯ В QT: КЛАСС ВРЕМЕНИ

hour()

- возвращает положительные значения часа в диапазоне от 0 до 23;

minute()

- возвращает целое значение, обозначающее минуты, в диапазоне от 0 до 59;

second()

- возвращает целое значение, обозначающее секунды, в диапазоне от 0 до 59;

msec()

- возвращает целое значение в диапазоне от 0 до 999, представляющее собой миллисекунды.

ДАТА И ВРЕМЯ В QT: КЛАСС ВРЕМЕНИ

toString()

- для передачи данных объекта времени в виде строки
- в качестве параметра, можно передать одно из форматов времени или задать свой собственный

fromString()

- преобразование из строкового типа в тип **QTime**
- в первом параметре метода нужно передать одно из значений форматов.

Например:

```
QTime time(20, 4, 23, 3);  
QString str;  
str = time.toString("hh:mm:ss.zzz");  
//str = "20:04:23.003"  
str = time.toString("h:m:s ap");  
//str = "8:4:23 pm"
```

ДАТА И ВРЕМЯ В QT: КЛАСС ВРЕМЕНИ

addSecs()

- Возвращает измененный объект времени, добавив или отняв от существующего объекта переданные значения секунд.

addMSecs()

- Возвращает измененный объект времени, добавив или отняв от существующего объекта переданные значения миллисекунд.

currentTime ()

- Возвращает текущее время

start()

- Начинает отсчет времени

elapsed()

- сколько времени прошло с момента начала отсчета

ДАТА И ВРЕМЯ В QT: КЛАСС ВРЕМЕНИ

Пример вычисления времени работы функции *test()*:

```
QTime time;  
time.start();  
test();  
qDebug() << "Время работы функции  
test() равно"  
    << time.elapsed()  
    << "миллисекунд"  
    << endl;
```

ДАТА И ВРЕМЯ В QT: КЛАСС ВРЕМЕНИ

QDateTime

Дата

Время

date()

- Возвращает объект даты **QDate**

time()

- Возвращает объект времени **QTime**

toString()

- для представления данных в виде строки.

ДАТА И ВРЕМЯ В QT: ТАЙМЕР

QTime

```
QTime time;  
time.start();  
for(;time.elapsed() < 1000;) {  
}  
function();
```

ДАТА И ВРЕМЯ В QT: ТАЙМЕР

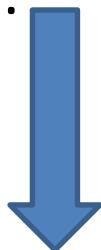
QApplication *processEvents()*

```
QTime timer;  
timer.start () ;  
for (;timer.elapsed () < 1000;) {  
    qApp->processEvents (0);  
}
```

ДАТА И ВРЕМЯ В QT: ТАЙМЕР

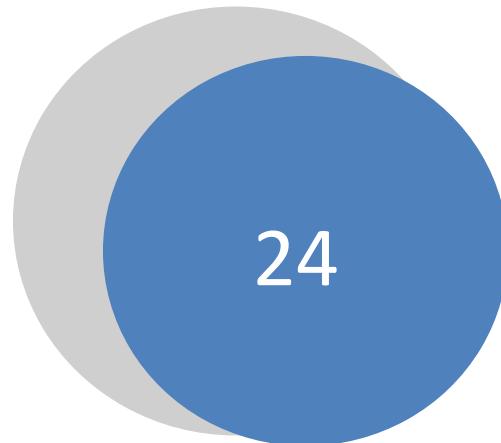
ТАЙМЕР

Интервал запуска (firing interval) – это период между событиями таймера.



сигнальное состояние

События таймера происходят асинхронно и не прерывают обработку других событий, выполняемых в том же потоке.



ДАТА И ВРЕМЯ В QT: ПРИМЕНЕНИЕ ТАЙМЕРА

в текстовом
редакторе

- для автоматического сохранения файлов

в качестве
альтернативы
многопоточности

- разбив программу на части, каждая из которых будет выполняться при наступлении события таймера

для отображения
информации о
состоянии данных

- данных, изменяющихся с течением времени.

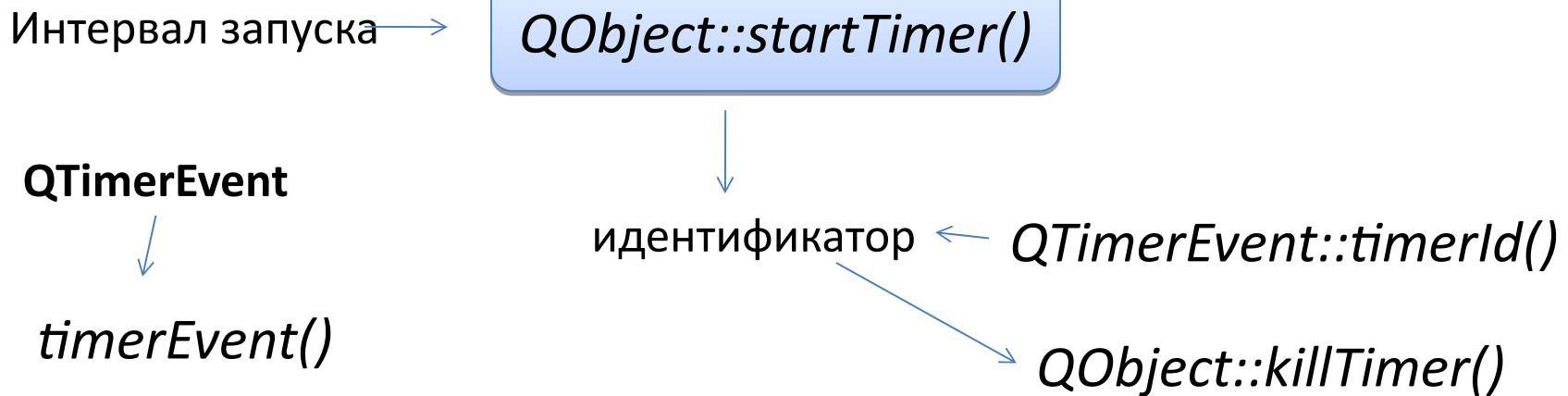
для исполнения
программ в режиме
реального времени.

- для избежания разногласий, связанных с мощностью
и возможностями разных компьютеров

в мультипоточном
программировании

- для каждого потока, имеющего цикл сообщений
(event loop). Для запуска цикла сообщений в потоке
нужно вызвать метод `QThread::exec()`.

ДАТА И ВРЕМЯ В QT: ТАЙМЕР



```
int main (int argc, char** argv) {  
    QApplication app (argc, argv);  
    BlinkLabel lbl("<FONT COLOR =  
RED><CENTER>Blink</CENTER></FONT>");  
    lbl.show();  
    return app.exec(); }
```

ДАТА И ВРЕМЯ В QT: ТАЙМЕР

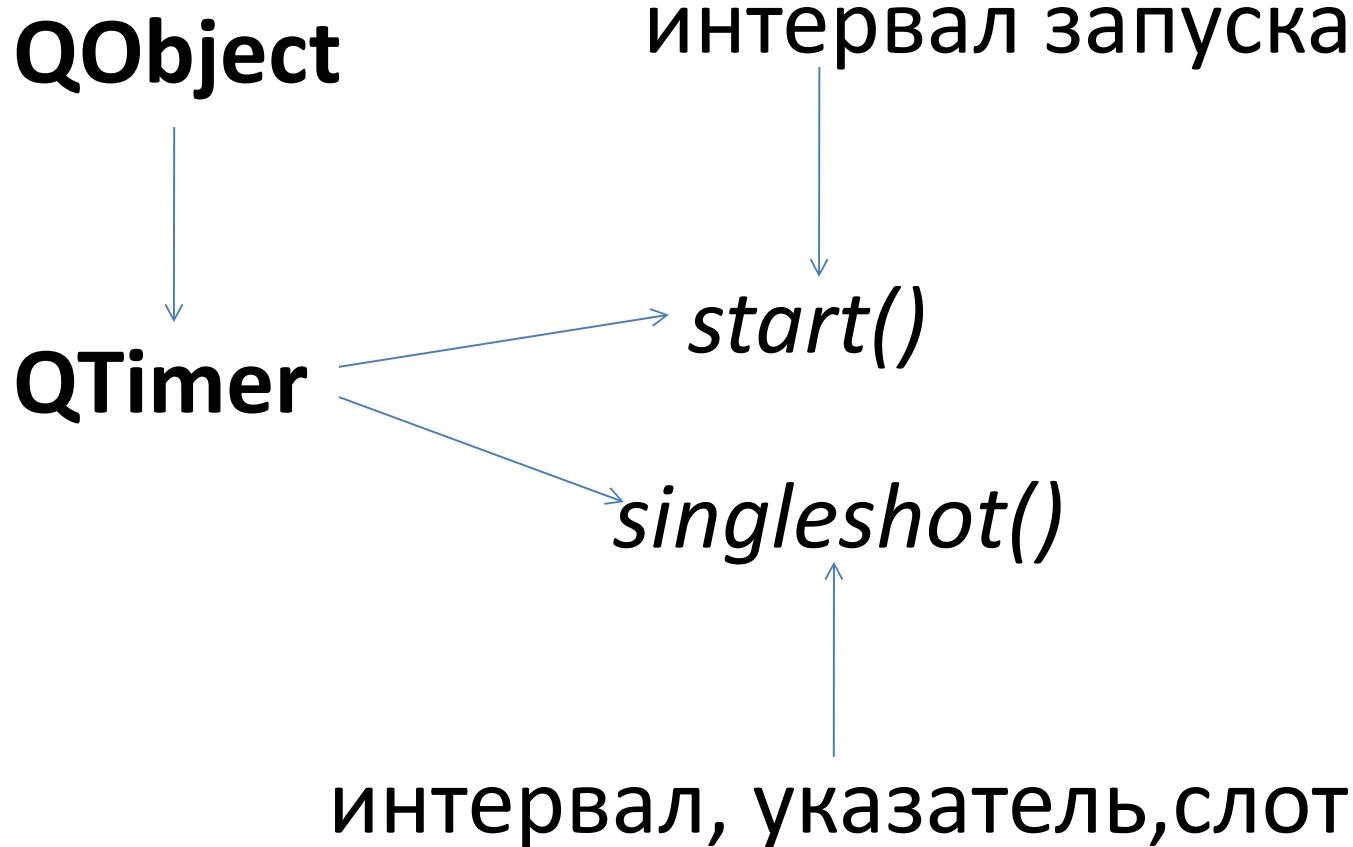
```
class BlinkLabel : public QLabel {
private:
    bool    m_bBlink;
    QString m_strText;

protected:
    virtual void timerEvent(QTimerEvent*)
    {
        m_bBlink = !m_bBlink;
        setText(m_bBlink ? m_strText : "");
    }
}
```

ДАТА И ВРЕМЯ В QT: ТАЙМЕР

```
public:  
    BlinkLabel(const QString& strText,  
               int          nInterval = 200,  
               QWidget*     pwgt      = 0  
    )  
    : QLabel(strText, pwgt)  
    , m_bBlink(true)  
    , m_strText(strText)  
    {  
        startTimer(nInterval);  
    }  
};
```

ДАТА И ВРЕМЯ В QT: КЛАСС QTIMER



ДАТА И ВРЕМЯ В QT: КЛАСС QTIMER

```
int main(int argc, char** argv) {  
  
    QApplication app(argc, argv);  
    MyProgram myProgram;  
    QTimer::singleShot(5 * 60 * 1000,  
                      &app, SLOT(quit()));  
    myProgram.show();  
    return app.exec(); }
```

timeout() *setInterval()* *isActive()* *stop()*

ДАТА И ВРЕМЯ В QT: КЛАСС QTIMER

```
class Clock : public QLabel {  
Q_OBJECT  
  
public:  
Clock(QWidget* pwgt = 0) : QLabel(pwgt)  
{  
    QTimer* ptimer = new QTimer(this);  
    connect(ptimer, SIGNAL(timeout()),  
            SLOT(slotUpdateDateTime()));  
    ptimer->start(500);  
    slotUpdateDateTime();  
}
```

ДАТА И ВРЕМЯ В QT: КЛАСС QTIMER

```
public slots:  
void slotUpdateDateTime()  
{ QString str =  
QDateTime::currentDateTime().toString  
    (Qt::SystemLocaleDate);  
setText("<H2><CENTER>" + str +  
    "</CENTER></H2>");  
}  
}; #endif //_Clock_h_
```

ДАТА И ВРЕМЯ В QT: КЛАСС QTIMER

QBasicTimer

isActive()

QObject::timerEvent().

start()

stop()

timerId()

Qt и SQL. Программирование баз данных

Первичный ключ

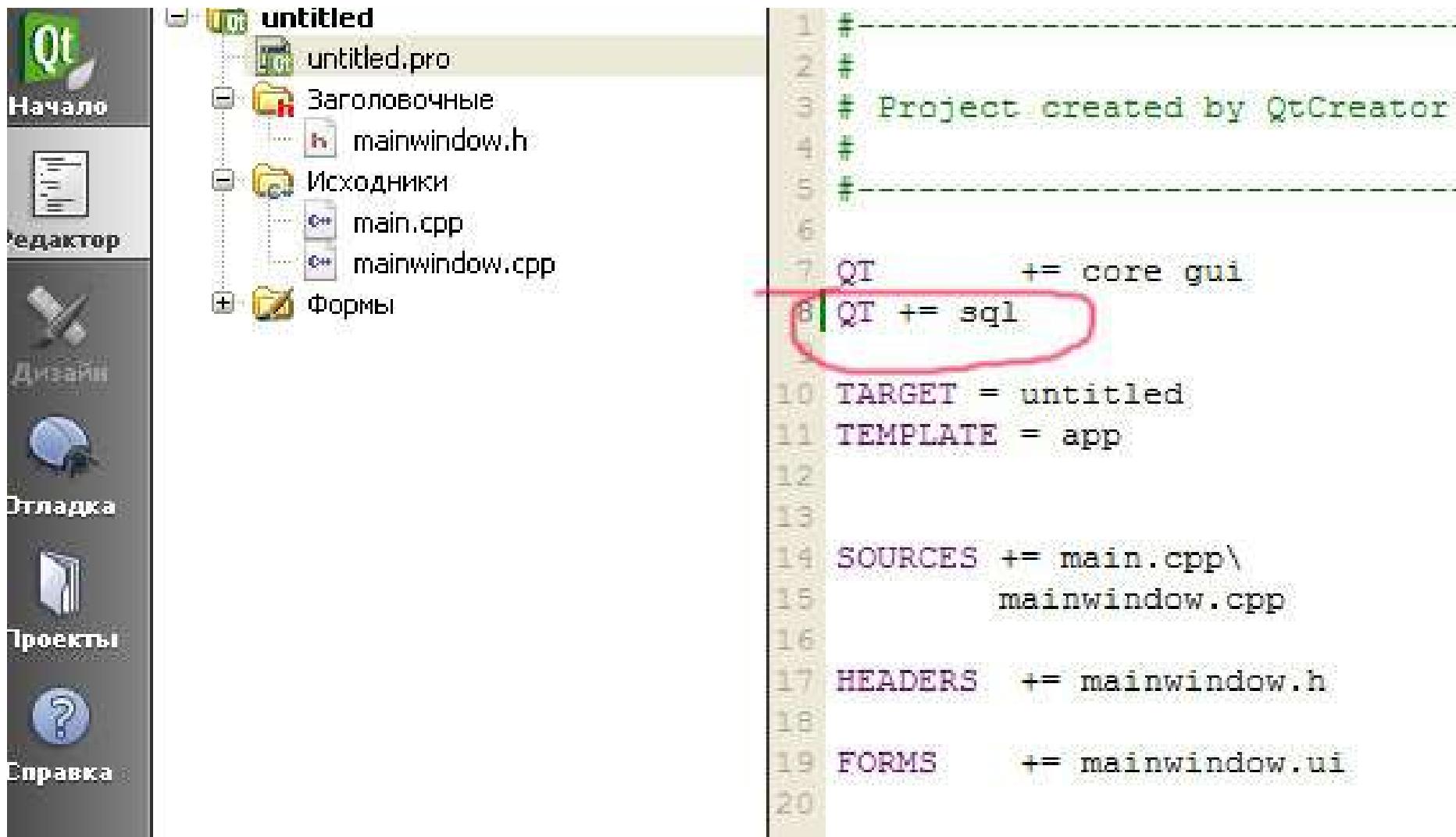


Строка
(запись)

Ячейка (поле)

Number	Name	Phone	Email
1	Piggy	+834378756	piggy@mega.de
2	Kermit	+427856438	kermit@mega.de
3	Gonzo	+988468267	gonzo@mega.de

Qt и SQL. Программирование баз данных



Qt Creator interface showing a project named "untitled". The left sidebar includes icons for Начало, Редактор, Дизайн, Отладка, Проекты, and Справка. The Projects tab shows the "untitled" project with files: untitled.pro, Заголовочные (mainwindow.h), Исходники (main.cpp, mainwindow.cpp), and Формы.

```
1 #
2 #
3 # Project created by QtCreator
4 #
5 #
6 #
7 QT      += core gui
8 QT      += sql
9 #
10 TARGET = untitled
11 TEMPLATE = app
12
13
14 SOURCES += main.cpp \
15             mainwindow.cpp
16
17 HEADERS  += mainwindow.h
18
19 FORMS    += mainwindow.ui
20
```

Qt и SQL. Создание таблицы

```
#include <QtSql>
```

```
CREATE TABLE addressbook (
    number INTEGER PRIMARY KEY NOT NULL,
    name VARCHAR(15),
    phone VARCHAR(12),
    email VARCHAR(15)
);
```

Qt и SQL. Добавление данных

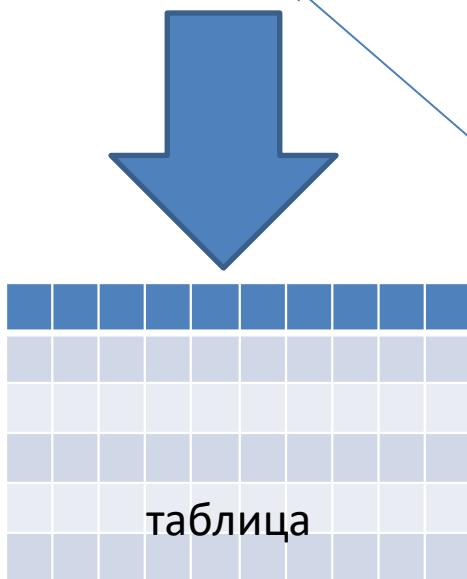
**INSERT INTO
VALUES**

```
INSERT INTO addressbook (number, name, phone, email)  
VALUES(1, 'Piggy', '+49 631322187', 'piggy@mega.de');
```

```
INSERT INTO addressbook (number, name, phone, email)  
VALUES(2, 'Kermit', '+49 631322181', 'kermit@mega.de');
```

Qt и SQL. Выборка данных

SELECT... FROM... WHERE ...



Оператор выборки, по условиям

Таблица для запроса

Проекция – выбор столбцов
(если * - все столбцы)

Qt и SQL. Выборка данных

SELECT... FROM... WHERE ...

SELECT email

FROM addressbook

WHERE name = 'Piggy';

Qt и SQL. Изменение данных

UPDATE ... SET ... WHERE ...

Строки по условию

Столбцы для изменения

Таблица

UPDATE addressbook

SET email = 'piggy@supermega.de'

WHERE name = 'Piggy';

Qt и SQL. Удаление данных

DELETE FROM ... WHERE ...

Критерий
Таблица

```
DELETE FROM addressbook  
WHERE name = 'Piggy';
```

Классы модуля QSql

Уровень драйверов: QSqlDriver, QSqlDriverCreator<T*>,
QSqlDriverCreatorBase, QSqlDriverPlugin и QSqlResult

Программный уровень:

QSqlDatabase, QSqlQuery, QSqlError, QSqlField,
QSqlIndex и QSqlRecord

Уровень пользовательского интерфейса:

QSqlQueryModel, QSqlTableModel и
QSqlRelationalTableModel

Драйверы БД

Идентификатор	описание
Идентификатор	Описание
QOCI	Доступ к базам данных Oracle через Oracle Call Interface . Поддерживаются версии 7, 8 и 9
QODBC	ODBC (Open Database Connectivity) , открытый интерфейс доступа к базе данных) — стандартный ODBC -драйвер для Microsoft SQL Server , IBM DB2 , Sybase SQL , iODBC и других баз данных
QMYSQL	MySQL — самый популярный в настоящее время бесплатный менеджер базы данных
QTDS	Sybase Adaptive Server
QPSQL	Базы данных PosgreSQL с поддержкой SQL92/SQL3
QSQLITE2	SQLite версии 2
QSQLITE	SQLite версии 3
QIBASE	Borland InterBase
QOCI	Oracle Call Interface

<http://www.sqlite.org>.

Драйверы БД



Соединение с базой данных

QSqIDatabase::addDatabase()

имя базы данных — передается в метод [QSqIDatabase::setDatabaseName\(\)](#);

имя пользователя, желающего к ней подключиться, — передается в метод [QSqIDatabase::setUserName\(\)](#);

имя компьютера, на котором размещена база данных, — передается в метод [QSqIDatabase::setHostName\(\)](#);

пароль — передается в метод [QSqIDatabase::setPassword\(\)](#).

Соединение с базой данных

```
9  
10 int main(int argc, char *argv[])  
11 {  
12     QApplication app(argc, argv);  
13  
14     QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");  
15     db.setDatabaseName("MyBase1");  
16     db.setUserName("user");  
17     db.setHostName("localhost");  
18     db.setPassword("password");
```



Соединение с базой данных обычно идентифицируется **по имени соединения**, а не по имени базы данных. Вы можете иметь множество соединений с одной и той же базой данных.

Открытие базы данных

```
19 if (!db.open()) {  
20     qDebug() << "Cannot open database:" << db.lastError();  
21     return false; }  
22 else qDebug() << "The Base.";
```

- *QSqlDatabase::lastError()* возвращает объект класса **QSqlError**



Если у вас возникла необходимость получить строку с ошибкой, то нужно вызвать из объекта класса **QSqlError** метод **text()**.

Метаинформация

```
QStringList lst = db.tables();
Foreach (QString str, lst) {
qDebug() << "Table:" << str;
}
```

Запросы

```
24  
25 QSqlQuery query;  
26     QString str = "CREATE TABLE Mybase ("  
27             "number INTEGER PRIMARY KEY NOT NULL, "  
28             "name    VARCHAR(15), "  
29             "phone   VARCHAR(12), "  
30             "email   VARCHAR(15) "  
31         ")";  
32  
33     if (!query.exec(str)) {  
34         qDebug() << "Unable to create a table";  
35     }
```

Запросы

QSqlQuery - возможность навигации

SELECT



next() – перемещаться на следующую строку данных

previous() – перемещаться на предыдущую строку данных

first() – установить первую строку данных

last() – установить последнюю строку данных

seek() – установить строку данных по указанному целочисленному индексу в его параметре.

size() – получить количество строк данных

Вставка данных

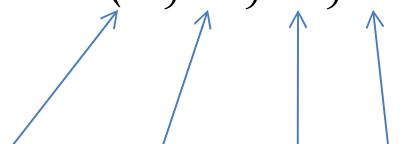
```
query.prepare("INSERT INTO addressbook  
    (number, name, phone, email)  
VALUES(:number, :name, :phone, :email);";  
  
query.bindValue (:number", "1");  
query.bindValue(":name", "Piggy");  
query.bindValue(":phone", " + 49 631322187");  
query.bindValue(":email", "piggy@mega.de");
```

Вставка данных

```
query.prepare("INSERT INTO addressbook  
(number, name, phone, email) VALUES(?, ?, ?, ?);");
```

```
query.bindValue("1");  
query.bindValue("Piggy");  
query.bindValue("+ 49 631322187");  
query.bindValue("piggy@mega.de");
```

Безымянные параметры



Вставка данных

```
36 //Adding some information
37 QString strF =
38     "INSERT INTO MyBase (number, name, phone, email) "
39     "VALUES (%1, '%2', '%3', '%4');"
40
41 str = strF.arg("1")
42         .arg("Andrew")
43         .arg("+49 631322187")
44         .arg("andrew@mega.de");
45 if (!query.exec(str)) {
46     qDebug() << "Unable to do insert opeation";
47 }
```

Вставка данных

```
47
48
49     str = strF.arg("2")
50             .arg("Alex")
51             .arg("+49 631322181")
52             .arg("alex@mega.de");
53     if (!query.exec(str)) {
54         qDebug() << "Unable to do insert operation";
55     }

```

```
56
57     if (!query.exec("SELECT * FROM MyBase;")) {
58         qDebug() << "Unable to execute query - exiting";
59         return 1;
60     }
61
62     //Reading of the data
63     QSqlRecord rec      = query.record();
64     int          nNumber = 0;
65     QString      strName;
66     QString      strPhone;
67     QString      strEmail;
68 }
```

```
69     while (query.next()) {
70         nNumber = query.value(rec.indexOf("number")).toInt();
71         strName = query.value(rec.indexOf("name")).toString();
72         strPhone = query.value(rec.indexOf("phone")).toString();
73         strEmail = query.value(rec.indexOf("email")).toString();
74
75         qDebug() << nNumber << " " << strName << ";\t"
76                     << strPhone << ";\t" << strEmail;
77     }
78
79     return app.exec();
80 }
81
```

```
20             qDebug() << "Cannot open database:" << db.lastError  
21             return false; }  
22     else qDebug() << "The Base.";  
23  
24     QSqlQuery query;  
25     QString str = "CREATE TABLE Mybase ( "  
26                                         "number INTEGER PRIMARY KEY NOT NU
```

Консоль приложения

untitled

Запускается D:\Kat\basa\untitled-build-desktop\debug\untitled.exe...

The Base.

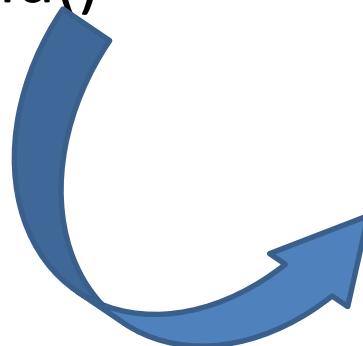
1	"Andrew" ;	"+49 631322187" ;	"andrew@mega.de"
2	"Alex" ;	"+49 631322181" ;	"alex@mega.de"

QSqlRecord

`QSqlRecord::count()`

`QSqlRecord::fieldName()`

`QSqlRecord::field()`



QSqlField

`QSqlField::name()`

`QSqlField::type()`

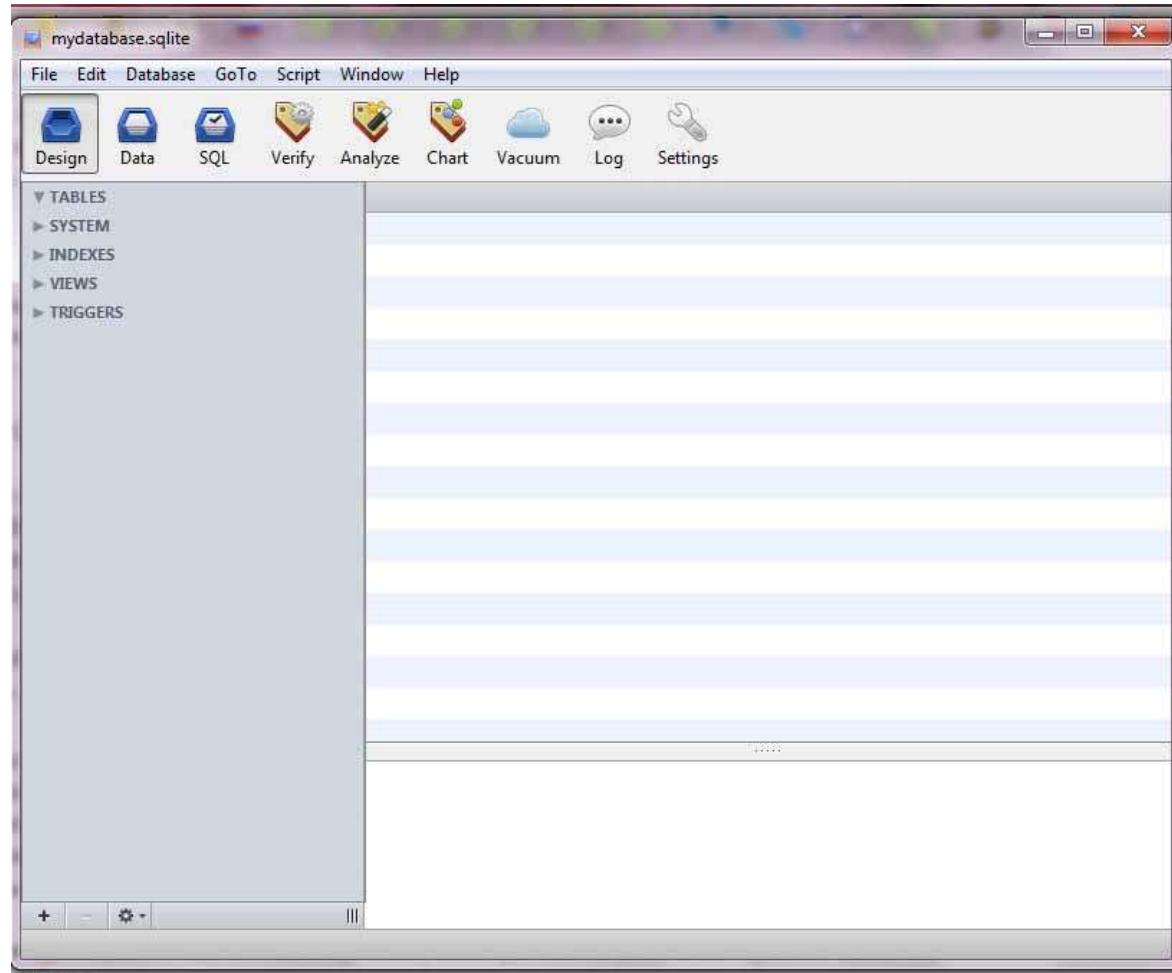
`QSqlField::length()`

`QSqlField::value()`

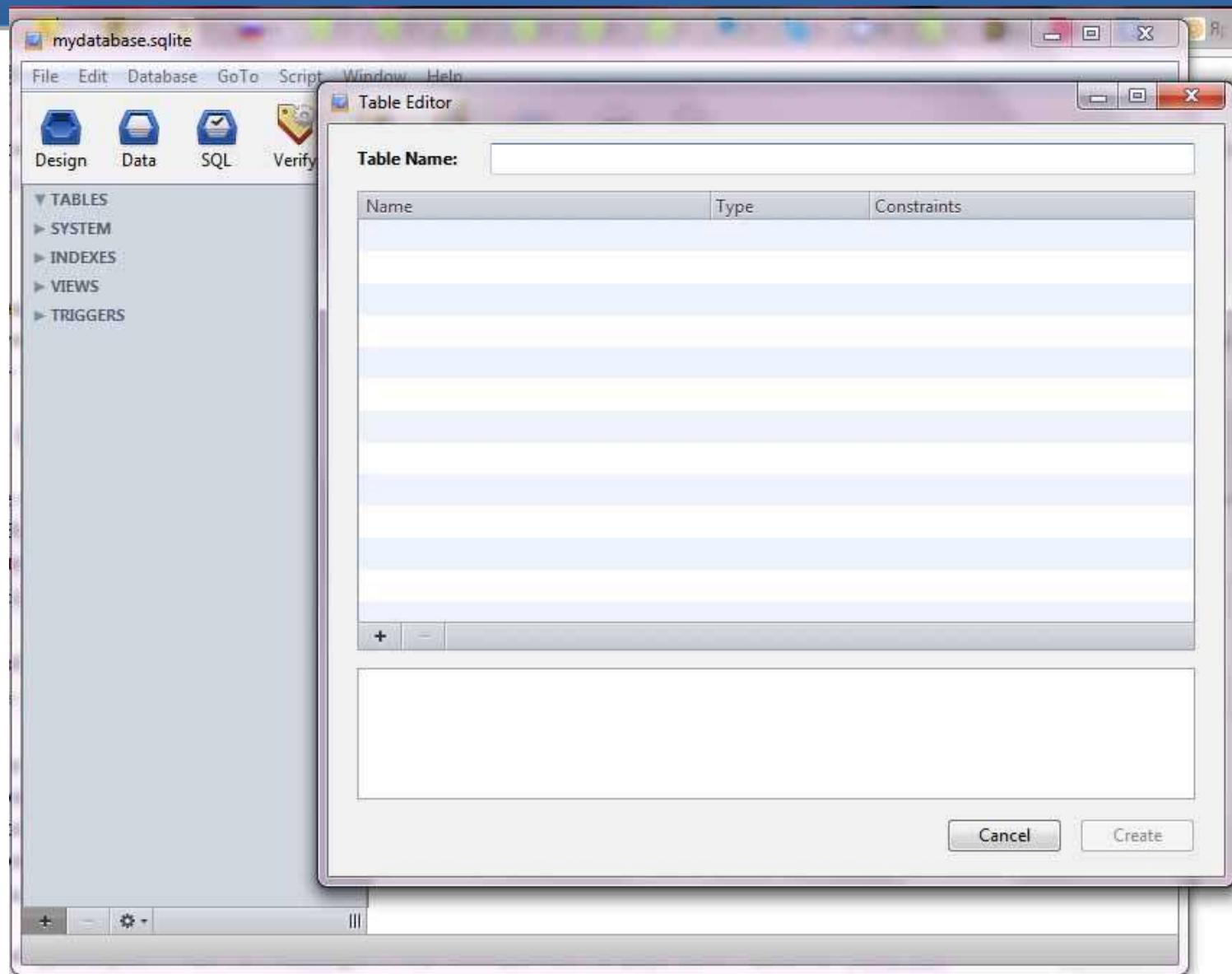
`QSqlRecord::contains()`

SQLiteManager

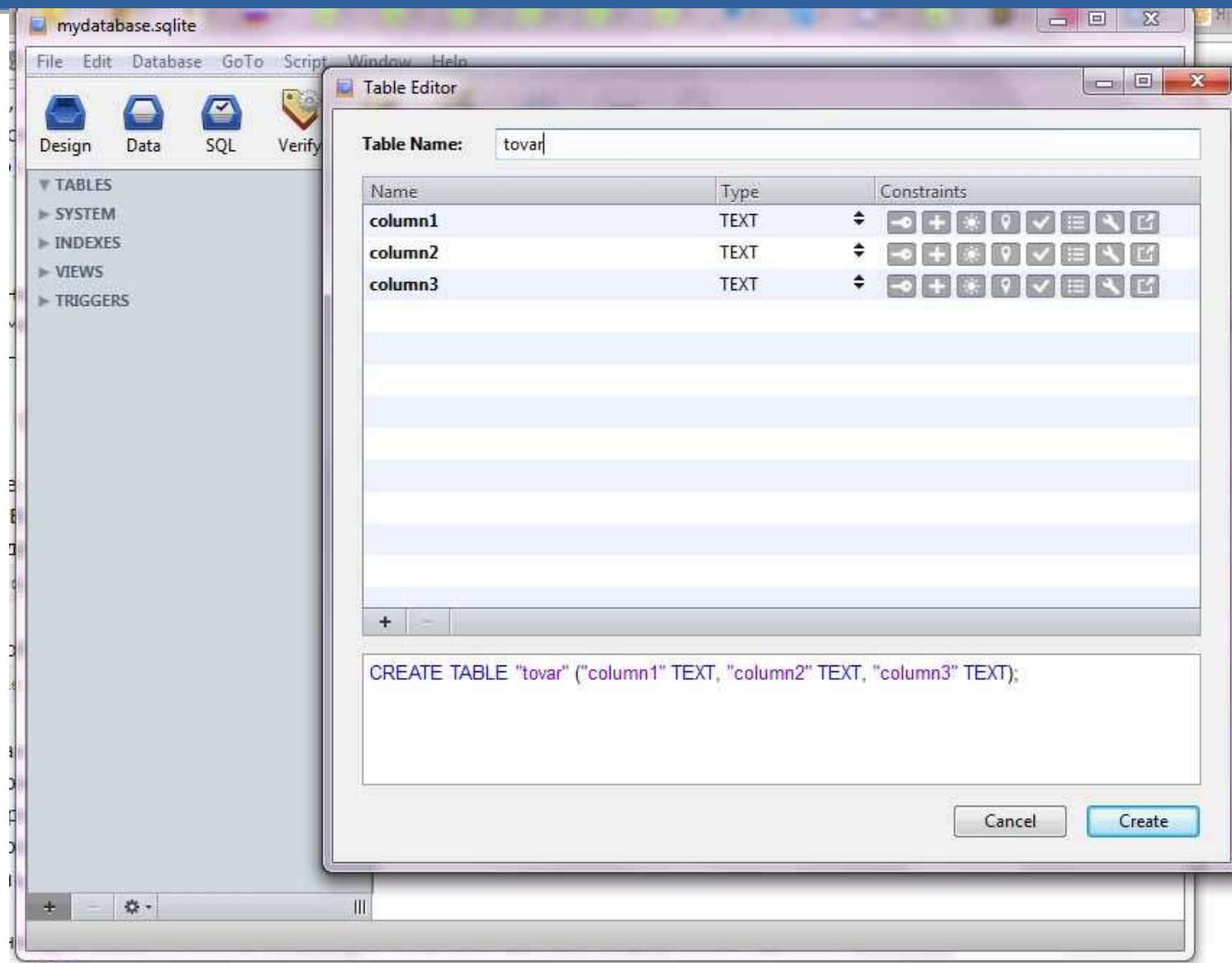
<http://www.sqlabs.net/sqlitemanager.php>



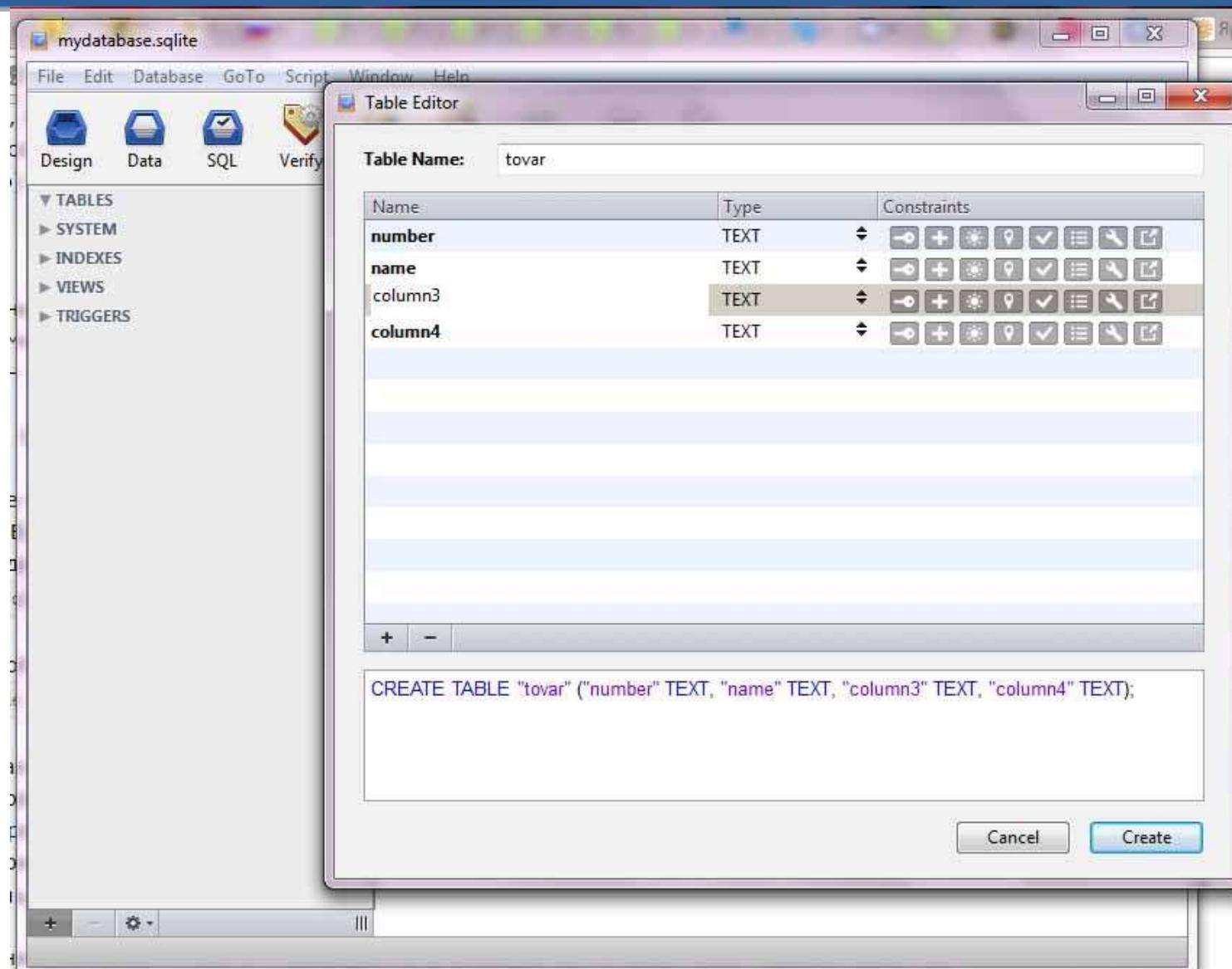
SQLiteManager



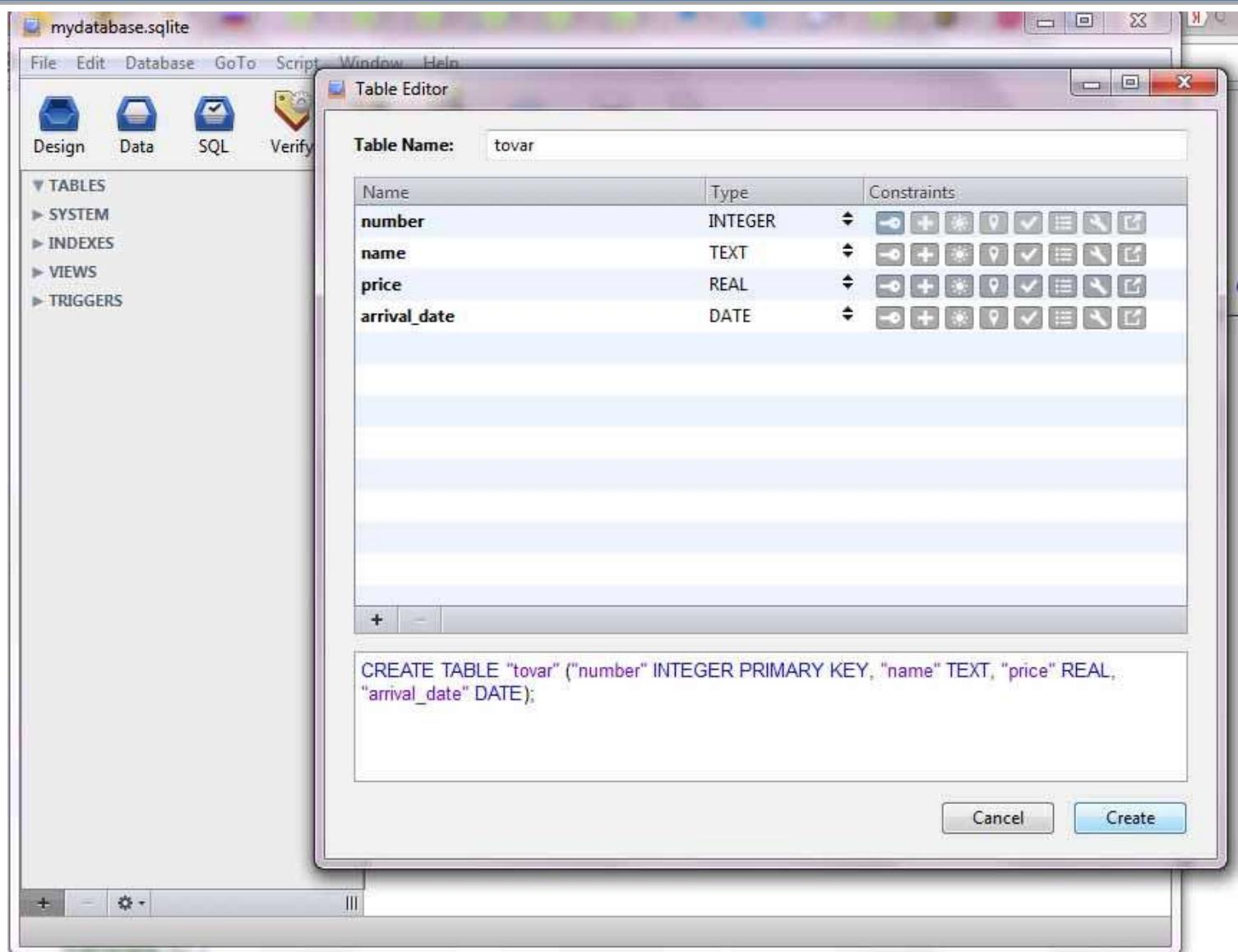
SQLiteManager



SQLiteManager



SQLiteManager



SQLiteManager и QTCreator

```
9 int main(int argc, char *argv[])
10 {
11     QApplication a(argc, argv);
12
13     QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");
14     db.setDatabaseName("C:\\\\Qt\\\\MyProjects\\\\base_5lekc\\\\mydatabase.sqlite");
15     if (!db.open()) {
16         qDebug() << "Не открывается" << db.lastError();
17         return false;
18     } else qDebug() << "Открылась";
19     return a.exec();
20 }
21
```



untitled

Запускается C:\\Qt\\\\MyProjects\\\\lekc5-1\\\\build-untitled-Desktop_Qt_5_3_MinGW_32bit-Debug
Открылась

Концепция интервью

- модель запроса
- табличная модель
- реляционная модель.



QSqlQueryModel

```
QTableView      view;
QSqlQueryModel model;
model.setQuery("SELECT phone, email "
               "FROM addressbook "
               "WHERE name = 'Piggy';"
               );

if (model.lastError().isValid()) {
    qDebug() << model.lastError();
}

view.setModel(&model);
view.show();
```

QSqlTableModel

	number	name	phone	email
1	1	Andrew	+49 631322187	andrew@mega.de
2	2	Alex	+49 631322181	alex@mega.de

QSqlTableModel

```
24 |     QTableView      view;
25 |     QSqlTableModel model;
26 |
27 |     model.setTable("Mybase");
28 |     model.select();
29 |     model.setEditStrategy(QSqlTableModel::OnFieldChange);
30 |
31 |     view.setModel(&model);
32 |     view.show();
```

QSqlTableModel

setEditStrategy()

onRowChange

onFieldChange

OnManualSubmit

submitAll()
revertAll()

Прохождение по строкам модели

```
for (int nRow =0; nRow < model.rowCount(); ++nRow) {  
    QSqlRecord rec = model.record(nRow);  
    int nNumber = record.value("number").toInt();  
    QString strName = record.value("name").toString();  
    ...  
}
```

Фильтрация и сортировка

```
model.setFilter("name = 'Piggy'");  
model.setSort(0, Qt::DescendingOrder);  
model.select();
```

Вставка новых записей

```
model.insertRows(0, 1);
model.setData(model.index(0, 0), 4);
model.setData(model.index(0, 1), "Sam");
model.setData(model.index(0, 2), "+49 63145476576");
model.setData(model.index(0, 3), "sam@mega.de");
if (!model.submitAll()) {
    qDebug() << "Insertion error!";
}
```

Удаление записей

```
model.setFilter("name = 'Piggy'");  
model.select();  
model.removeRows(0, model.rowCount());  
model.submitAll();
```

Реляционная модель QSqlRelationalTableModel

relationModel()

relation()

setRelation()

Реляционная модель QSqlRelationalTableModel

```
CREATE TABLE status (
    number INTEGER PRIMARY KEY NOT NULL,
    married VARCHAR(5) "
)
;
```

Внесем в нее данные:

```
INSERT INTO status (number, married) VALUES(1, 'YES');
INSERT INTO status (number, married) VALUES(2, 'NO');
```

Реляционная модель QSqlRelationalTableModel

	Married	name	phone	email
1	YES	Piggy	+8578635563	piggy@mega.de
2	NO	Kermit	+2786432435	kermit@mega.de

QSqlRelationalTableModel

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

    if (!createConnection()) {
        return -1;
    }

    QTableView view;
    QSqlRelationalTableModel model;

    model.setTable("addressbook");
    model.setRelation(0, QSqlRelation("status", "number", "married"));
    model.select();

    view.setModel(&model);
    view.show();

    return app.exec();
}
```

Человеко-машинное взаимодействие

Лекция 11

Мерзлякова Екатерина Юрьевна

к.т.н. доцент ПМИК

Жизненный цикл тестирования

- жизненный цикл тестирования — это **последовательность действий, проводимых в процессе тестирования, с помощью которых гарантируется качество программного обеспечения и его соответствие требованиям**



Виды тестирования

- 1. Функциональные.
- 2. Нефункциональные
- 3. Связанные с изменениями.

Функциональные виды тестирования.

Функциональные тесты.

- 1. Функциональные.
- 2. Нефункциональные
- 3. Связанные с изменениями.

Функциональные виды тестирования.

Тестирование безопасности.

- 1. Конфиденциальность.
- 2. Целостность.
- 3. Доступность.

Нефункциональные виды тестирования.

Тестирование производительности

- Измерение времени выполнения выбранных операций при определенных интенсивностях выполнения этих операций.
- Определение количества пользователей, одновременно работающих с приложением.
- Определение границ приемлемой производительности при увеличении нагрузки (при увеличении интенсивности выполнения этих операций).
- Исследование производительности на высоких, предельных, стрессовых нагрузках.

Нефункциональные виды тестирования.

Стрессовое тестирование



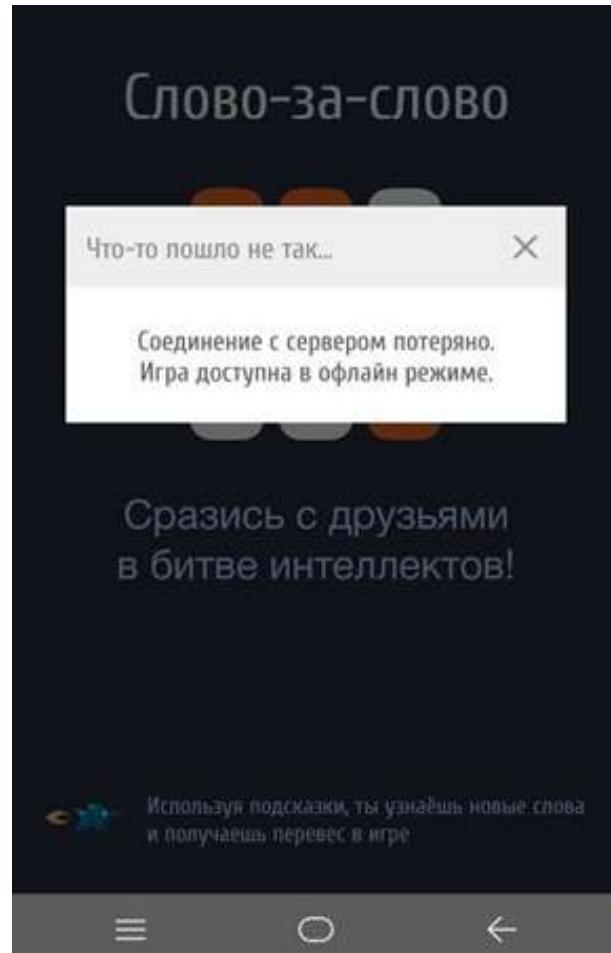
Нефункциональные виды тестирования.

Объемное тестирование

- Измерение времени выполнения выбранных операций при определенных интенсивностях выполнения этих операций.
- Может производиться определение количества пользователей, одновременно работающих с приложением.

Нефункциональные виды тестирования.

Тестирование стабильности или надежности



Нефункциональные виды тестирования.

Тестирование Установки



Нефункциональные виды тестирования.

Тестирование Установки

- Процесс получения списка файлов проводится **до**, а проверка файлов — **после** установки программы;
- Проверка корректности регистрации библиотек и служебных записей;
- Корректность регистрации расширений для новых файлов инсталлированной веб-программы внутри операционной системы;
- Проверять учетную запись пользователя и моментально информировать о потенциально возможных проблемах с пользовательскими правами;
- Проверка на возможные конфликты в отношении системных доступов к общим файловым ресурсам при одновременной установке сразу нескольких программ.

Нефункциональные виды тестирования.

Тестирование Установки

- Тестирование на обратную совместимость – при выполнении обновлений, все ранее инсталлированные файлы и папки должны правильно открываться и выполнять свои непосредственные системные «обязанности»;
- Если продукт запущен, клиент должен получить специальное уведомление о том, что выполнить обновление продукта не получится, поскольку программа уже функционирует.

Нефункциональные виды тестирования.

Тестирование Установки

- Процедура удаления продукта в процессе выполнения инсталляции (удалять можно как библиотеки в системном каталоге, так и регистрационные записи внутри системного реестра);
- Тестирование проверки удаления реестров, ссылки на системные библиотеки;
- Тест на сохранность данных при взаимодействии с продуктом.
- Проверка поведения инсталлятора, при условии, что каталог продукта закрыт для процесса удаления по правам доступа;
- Если пользователь не авторизирован для деинсталляции продукта, он должен получить специальное уведомление и процесс удаления должен быть остановлен.

Нефункциональные виды тестирования.

Тестирование удобства пользования

- **Производительность, эффективность (efficiency)** - сколько времени и шагов понадобится пользователю для завершения основных задач приложения.
- **Правильность (accuracy)** - сколько ошибок сделал пользователь во время работы с приложением .
- **Активизация в памяти (recall)** – как много пользователь помнит о работе приложения после приостановки работы с ним на длительный период времени.
- **Эмоциональная реакция (emotional response)** – как пользователь себя чувствует после завершения задачи - растерян, испытал стресс? Порекомендует ли пользователь систему своим друзьям?

Нефункциональные виды тестирования.

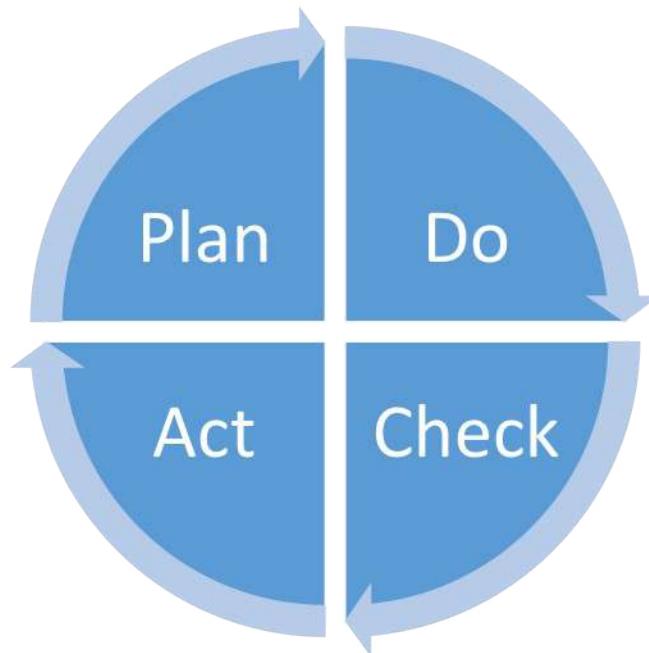
Тестирование удобства пользования

- Готовый продукт - тестирование черного ящика (black box testing),
- Интерфейсы приложения (API) - тестирование белого ящика (white box testing).

Нефункциональные виды тестирования.

Тестирование удобства пользования

- «пока-йока» или fail-safe, «защита от дурака»
- цикл Демминга (Plan-Do-Check-Act) :



Нефункциональные виды тестирования.

Тестирование удобства пользования

- Тестирование пользовательского интерфейса = Тестирование удобства пользования
- Тестирование удобства пользования можно провести без участия эксперта



Нефункциональные виды тестирования.

Тестирование на отказ и восстановление

- Целью данного вида тестирования является проверка систем восстановления (или дублирующих основной функционал систем), которые, в случае возникновения сбоев, обеспечат сохранность и целостность данных тестируемого продукта.

Нефункциональные виды тестирования.

Тестирование на отказ и восстановление

- Отказ электричества на компьютере-сервере.
- Отказ электричества на компьютере-клиенте.
- Незавершенные циклы обработки данных (прерывание работы фильтров данных, прерывание синхронизации).
- Объявление или внесение в массивы данных невозможных или ошибочных элементов.
- Отказ носителей данных.

Нефункциональные виды тестирования.

Тестирование на отказ и восстановление

- Симулировать внезапный отказ электричества на компьютере (обесточить компьютер).
- Симулировать потерю связи с сетью (выключить сетевой кабель, обесточить сетевое устройство).
- Симулировать отказ носителей (обесточить внешний носитель данных).
- Симулировать ситуацию наличия в системе неверных данных (специальный тестовый набор или база данных).

Нефункциональные виды тестирования.

Конфигурационное тестирование

- Проект по профилированию работы системы.

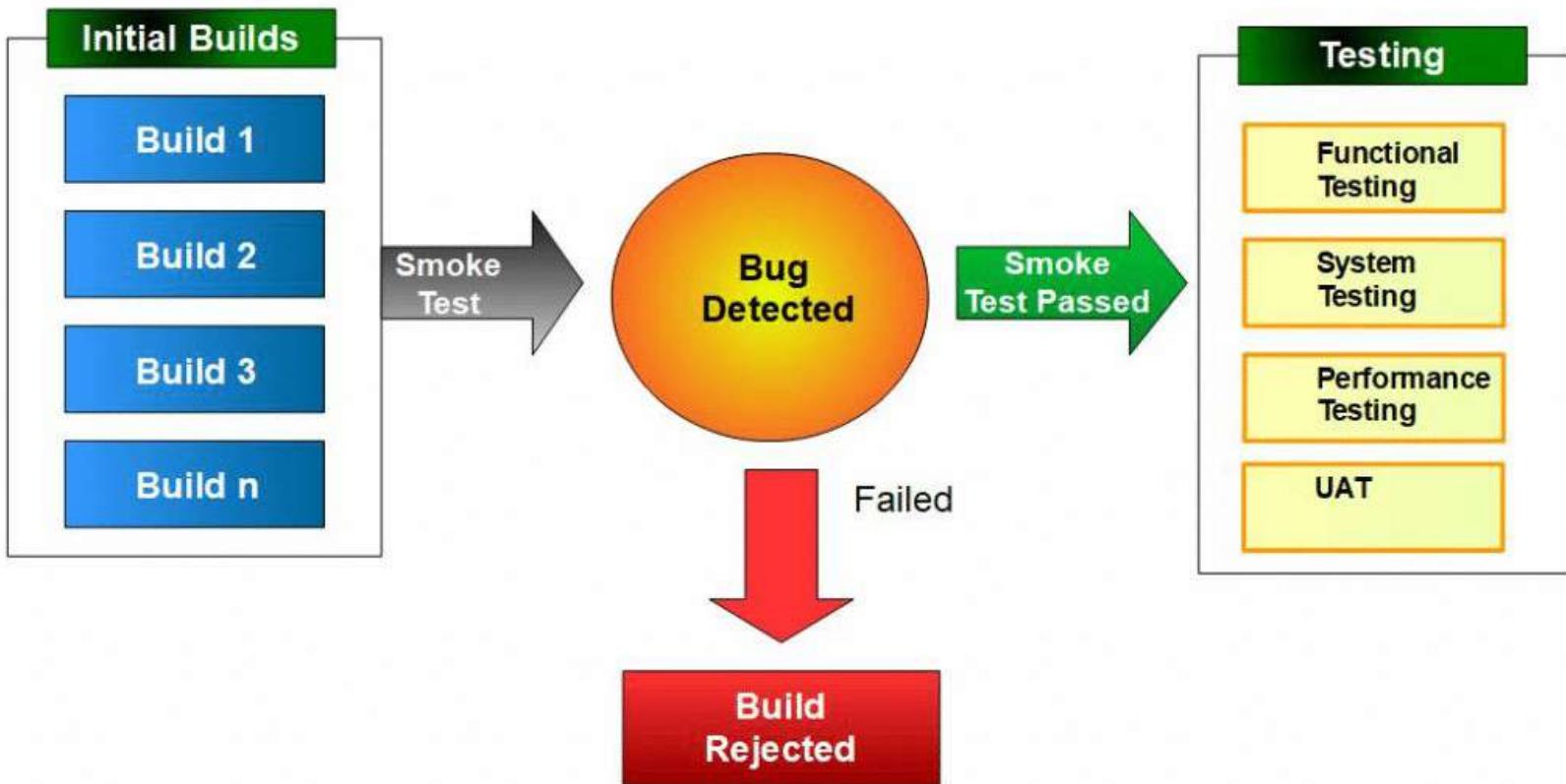
Цель Тестирования: определить оптимальную конфигурацию оборудования, обеспечивающую требуемые характеристики производительности и времени реакции тестируемой системы.

- Проект по миграции системы с одной платформы на другую.

Цель Тестирования: проверить объект тестирования на совместимость с объявленным в спецификации оборудованием, операционными системами и программными продуктами третьих фирм.

Связанные с изменениями виды тестирования.

Дымовое тестирование



Связанные с изменениями виды тестирования.

Регрессионное тестирование

- Регрессия багов (**Bug regression**) - попытка доказать, что исправленная ошибка на самом деле не исправлена.
- Регрессия старых багов (**Old bugs regression**) - попытка доказать, что недавнее изменение кода или данных сломало исправление старых ошибок, т.е. старые баги стали снова воспроизводиться.
- Регрессия побочного эффекта (**Side effect regression**) - попытка доказать, что недавнее изменение кода или данных сломало другие части разрабатываемого приложения.

Связанные с изменениями виды тестирования.

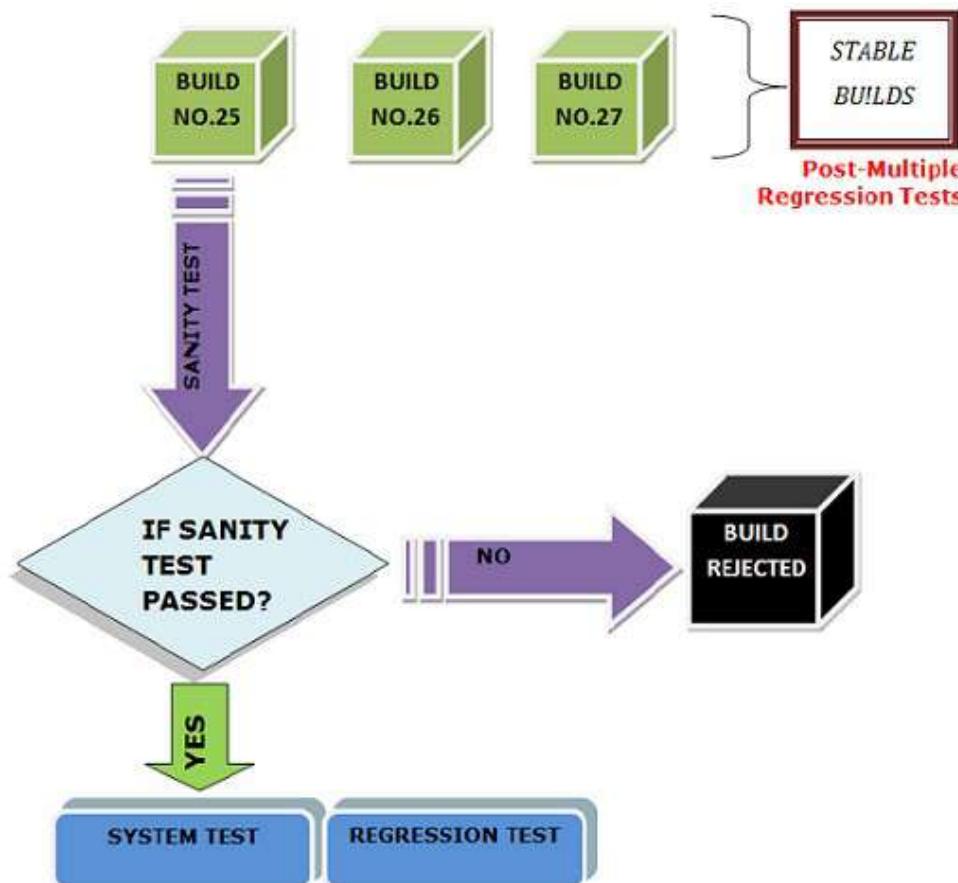
Тестирование сборки



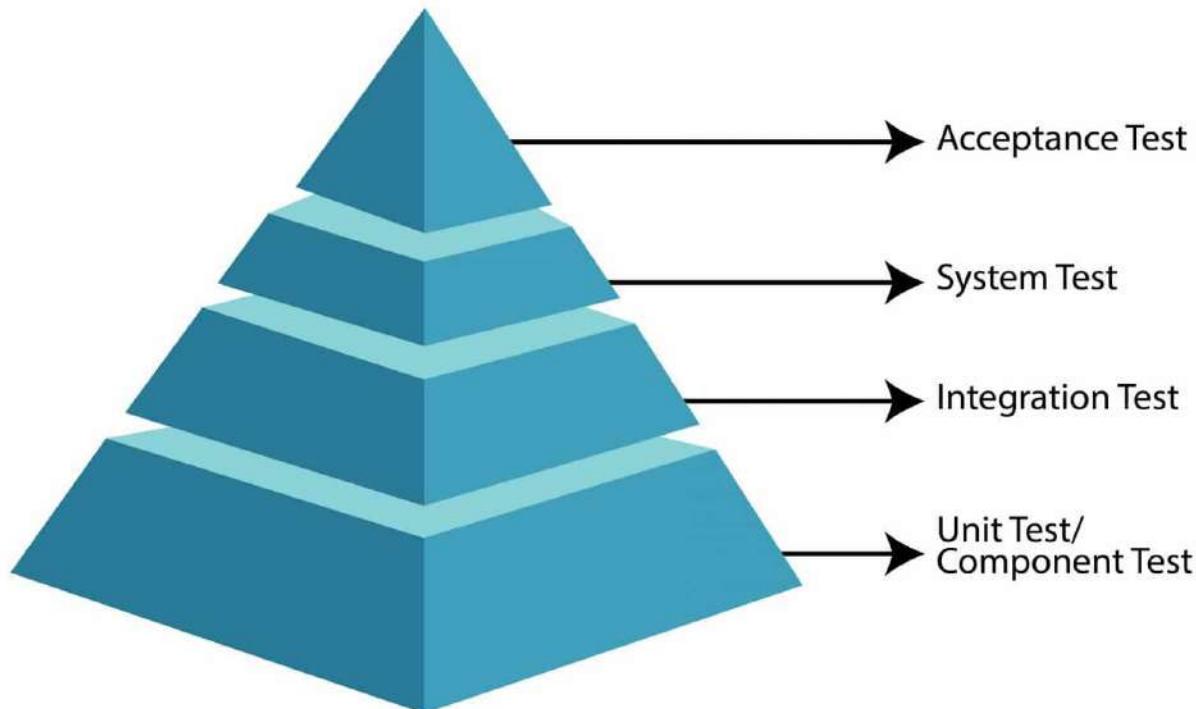
- Тестовые примеры проверки сборки должны охватывать все важные функциональные возможности программного обеспечения.

Связанные с изменениями виды тестирования.

Санитарное тестирование



Уровни Тестирования. Компонентное или Модульное тестирование



разработка от тестирования (test-driven development)
подход тестирования вначале (test first approach)

Уровни Тестирования. Интеграционное тестирование

- **Компонентный интеграционный уровень**
(Component Integration testing) проверяется взаимодействие между компонентами системы после проведения компонентного тестирования.
- **Системный интеграционный уровень**
(System Integration Testing) - проверяется взаимодействие между разными системами после проведения системного тестирования.

Уровни Тестирования. Интеграционное тестирование

- **Снизу вверх** (Bottom Up Integration):

Все низкоуровневые модули, процедуры или функции собираются воедино и затем тестируются. После чего собирается следующий уровень модулей для проведения интеграционного тестирования.

- **Сверху вниз** (Top Down Integration):

Сначала тестируются все высокоуровневые модули, затем постепенно, один за другим, добавляются низкоуровневые. Все модули более низкого уровня симулируются заглушками с аналогичной функциональностью, затем, по мере готовности, они заменяются реальными активными компонентами.

- **Большой взрыв** ("Big Bang" Integration):

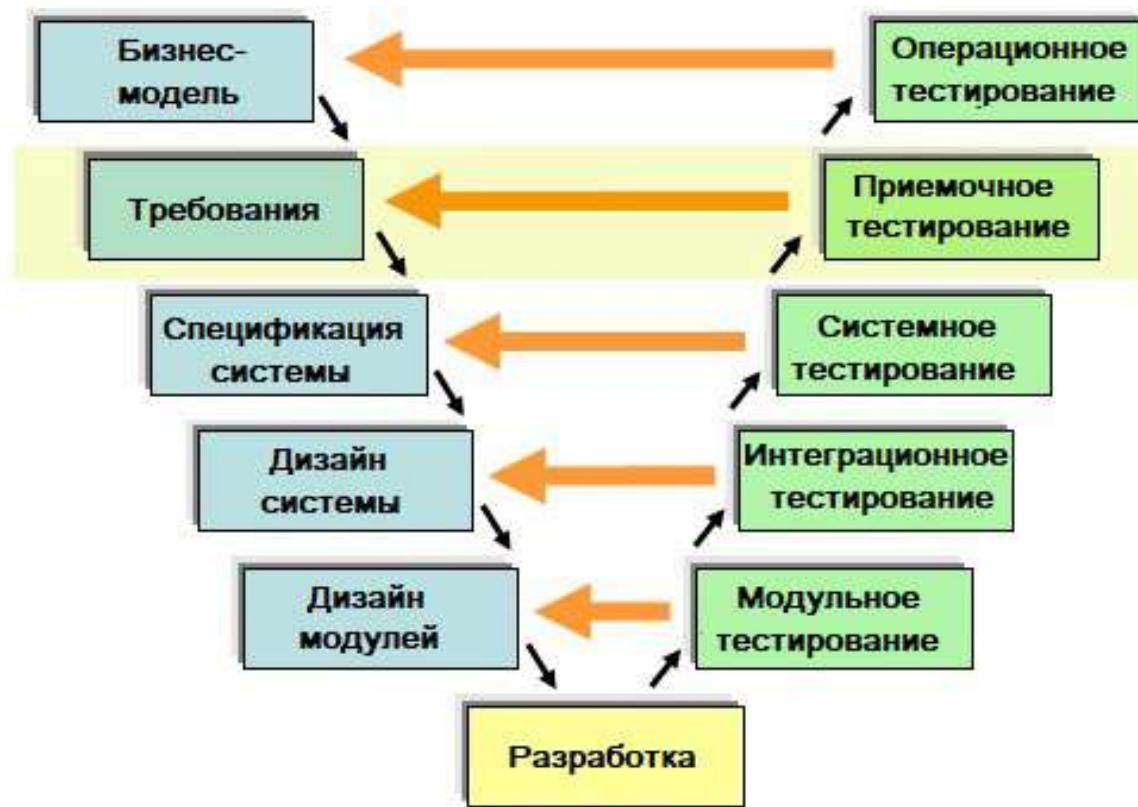
Все (или практически все) разработанные модули собираются вместе в виде законченной системы или ее основной части, а затем проводится интеграционное тестирование.

Уровни Тестирования.

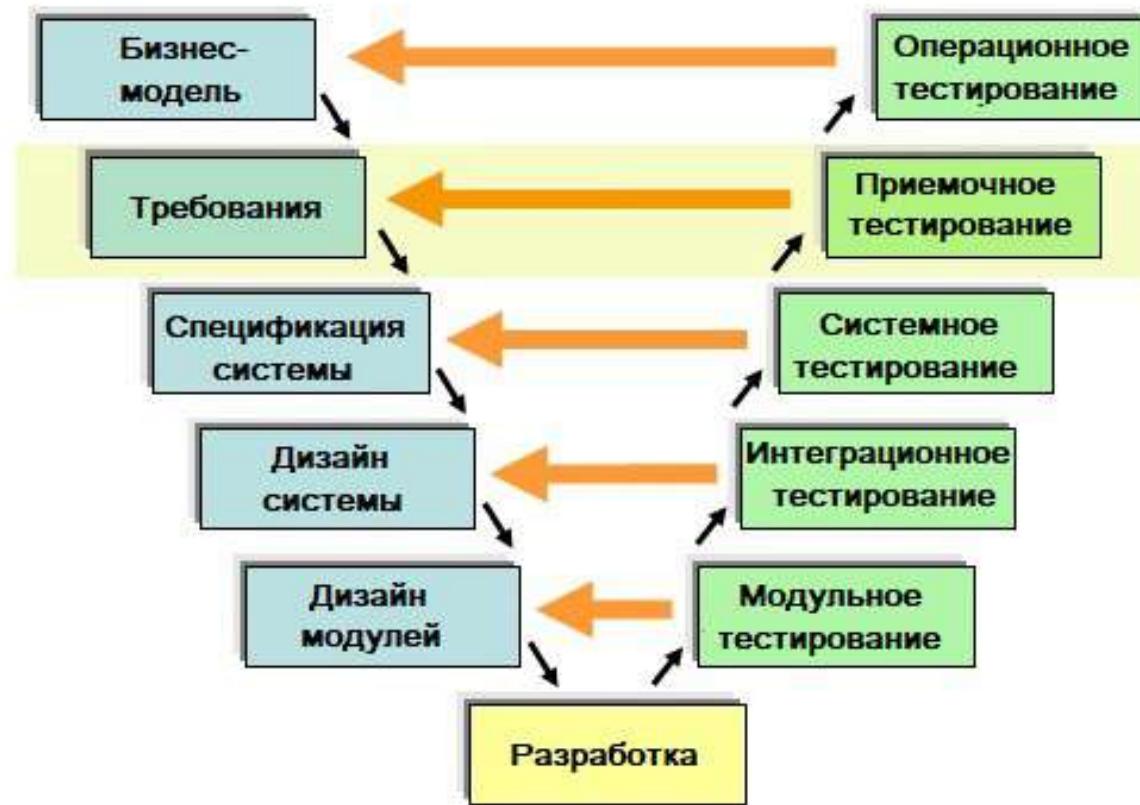
Системное тестирование

- **на базе требований** (requirements based) - для каждого требования пишутся тестовые случаи (test cases), проверяющие выполнение данного требования.
- **на базе случаев использования** (use case based) - на основе представления о способах использования продукта создаются случаи использования системы (Use Cases).

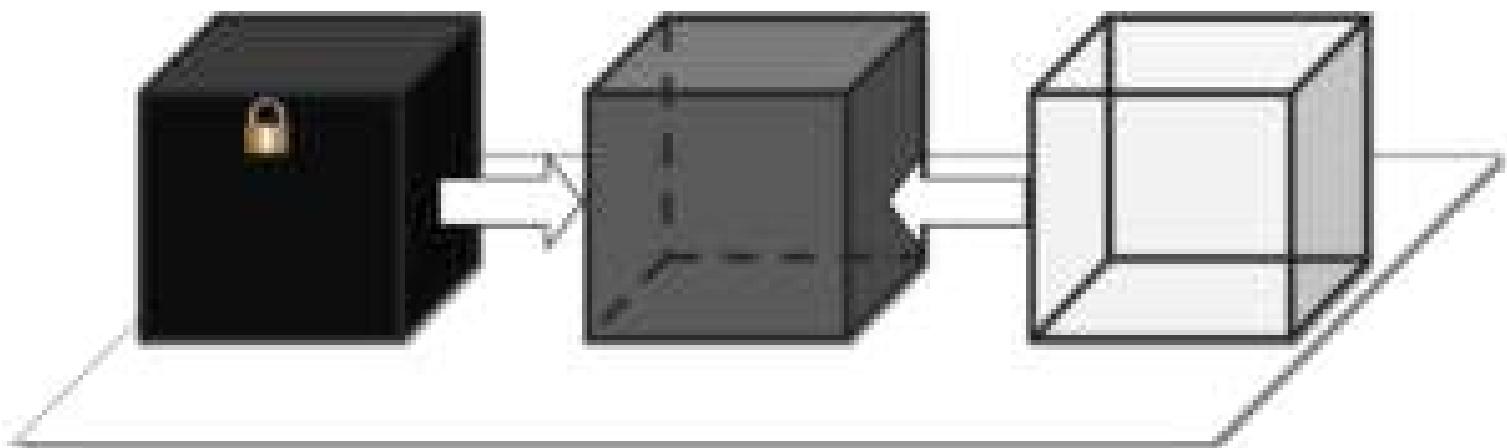
Уровни Тестирования. Приемочное тестирование



Уровни Тестирования. Операционное тестирование



Типы тестирования



Black Box



Black Box. Преимущества

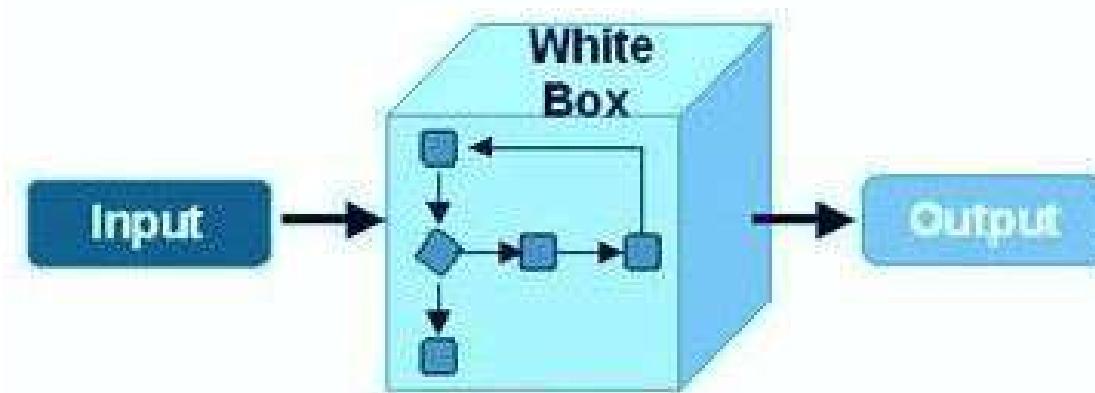
- 1. тестирование производится с позиции конечного пользователя и может помочь обнаружить неточности и противоречия в спецификации;
- 2. тестировщику нет необходимости знать языки программирования и углубляться в особенности реализации программы;
- 3. тестирование может производиться специалистами, независимыми от отдела разработки, что помогает избежать предвзятого отношения;
- 4. можно начинать писать тест-кейсы, как только готова спецификация.

Black Box. Недостатки

- 1. тестируется только очень ограниченное количество путей выполнения программы;
- 2. без четкой спецификации (а это скорее реальность на многих проектах) достаточно трудно составить эффективные тест-кейсы;
- 3. некоторые тесты могут оказаться избыточными, если они уже были проведены разработчиком на уровне модульного тестирования.

White Box

- тестирование, основанное на анализе внутренней структуры компонента или системы;
- тест-дизайн - составление тест-кейсов на основе анализа внутреннего устройства системы или кода



White Box. Преимущества

1. тестирование может производиться на ранних этапах: нет необходимости ждать создания пользовательского интерфейса;
2. можно провести более тщательное тестирование с покрытием большого количества путей выполнения программы.

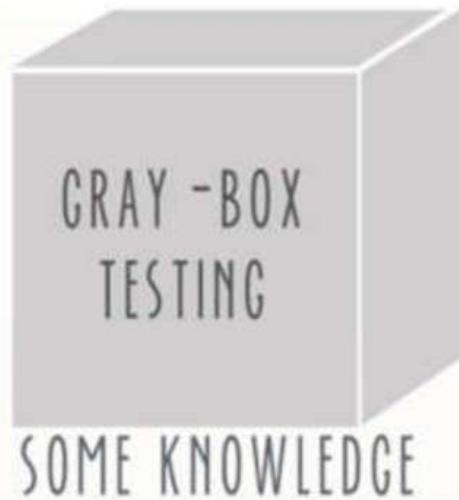
White Box. Недостатки

1. для выполнения тестирования белого ящика необходимо большое количество специальных знаний;
2. при использовании автоматизации тестирования на этом уровне поддержка тестовых скриптов может оказаться достаточно накладной, если программа часто изменяется.

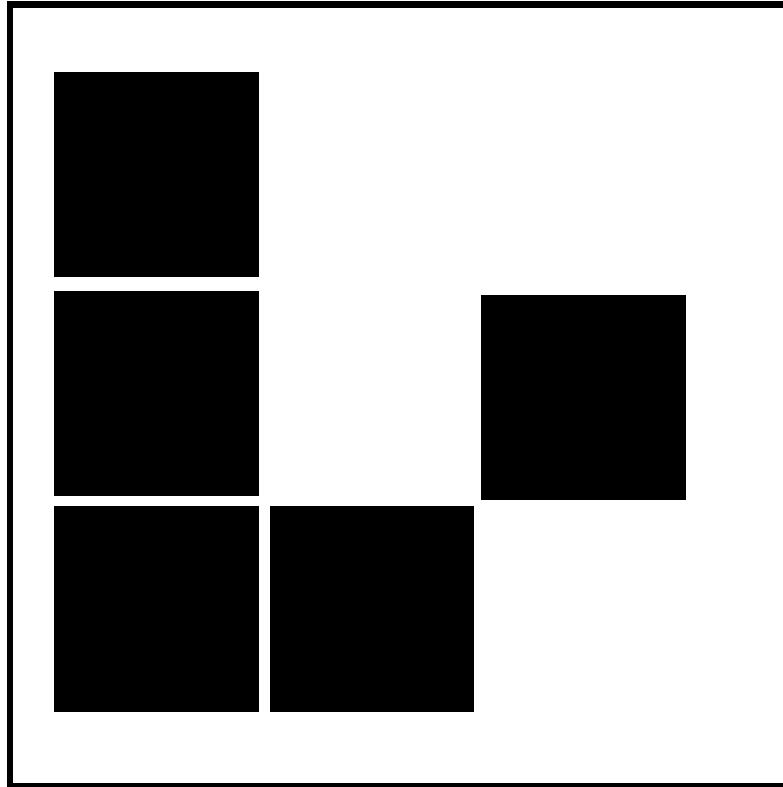
Сравнение Black Box и White Box

Критерий	Black Box	White Box
<i>Определение</i>	тестирование, как функциональное, так и нефункциональное, не предполагающее знания внутреннего устройства компонента или системы	тестирование, основанное на анализе внутренней структуры компонента или системы
<i>Уровни, к которым применима техника</i>	В основном: <ul style="list-style-type: none">Приемочное тестированиеСистемное тестирование	В основном: <ul style="list-style-type: none">Юнит-тестированиеИнтеграционное тестирование
<i>Кто выполняет</i>	Как правило, тестировщики	Как правило, разработчики
<i>Знание программирования</i>	Не нужно	Необходимо
<i>Знание реализации</i>	Не нужно	Необходимо
<i>Основа для тест-кейсов</i>	Спецификация, требования	Проектная документация

Grey Box

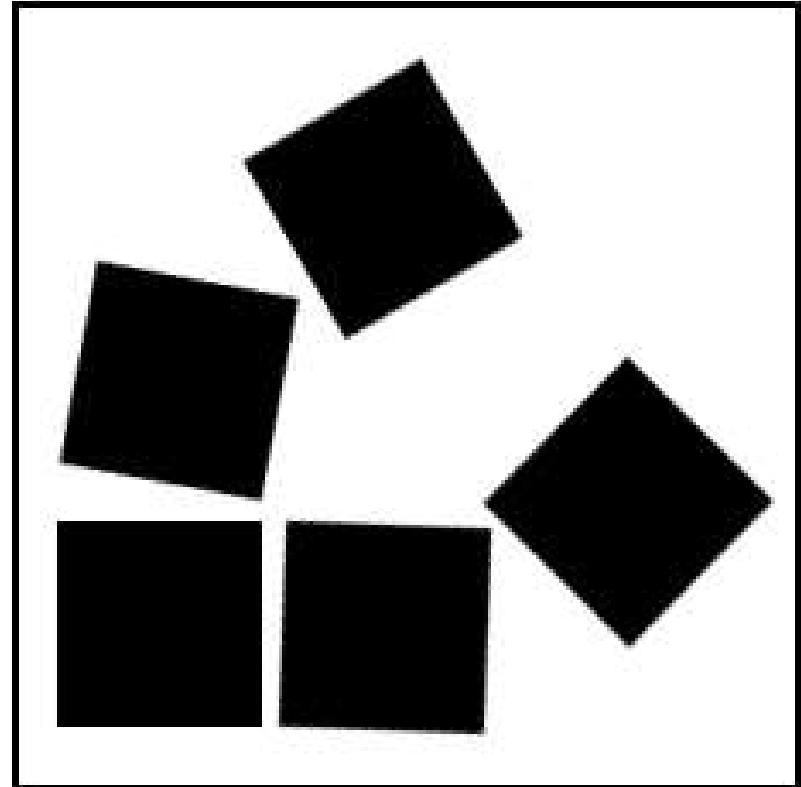


Статическое и динамическое тестирование

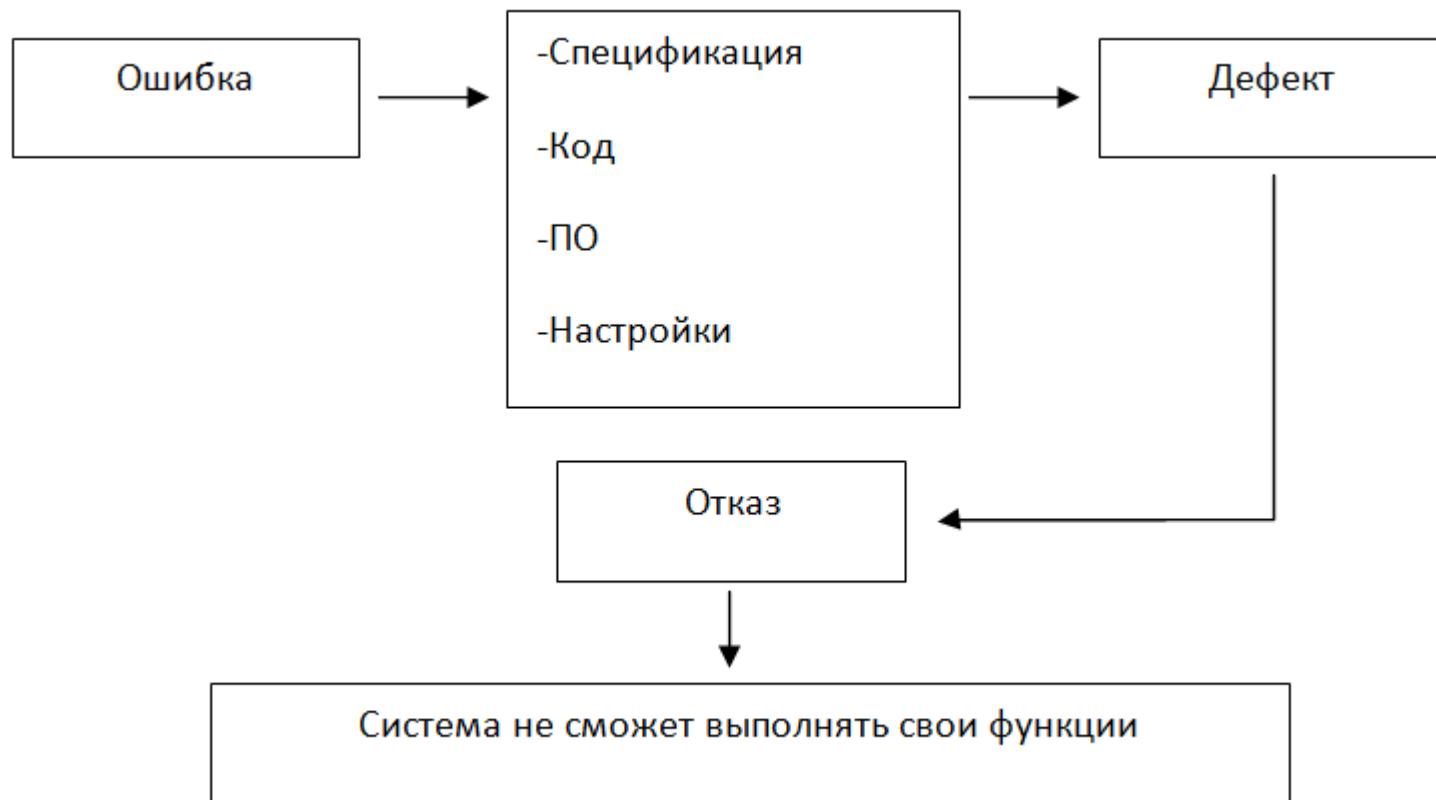


Статическое

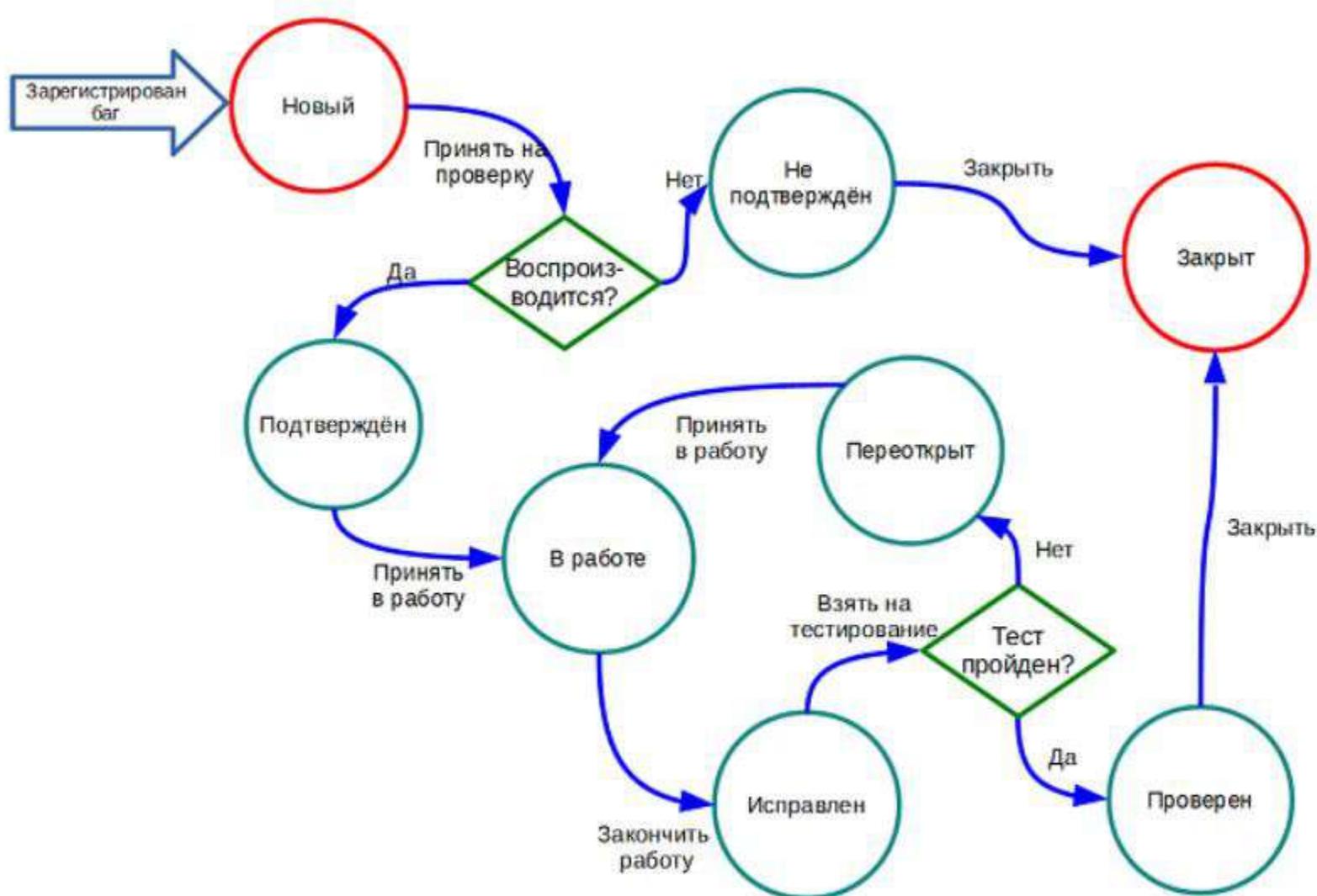
Динамическое



Отчёты о дефектах



Жизненный цикл «бага»



Использование данных отчета о дефекте

- Что было не так?
- Когда была создана эта проблема? Какое именно действие при разработке явилось ее источником? Это была проблема в требованиях? Проектировании системы? Коде? Тестировании?
- Когда проблема была выявлена? Сколько она существовала до того, как мы ее обнаружили?
- Каким образом была найдена эта проблема?
- Можно ли было обнаружить ее раньше?
- Сколько стоило устранение этой проблемы?
- Какого рода была эта проблема?

Атрибуты отчёта о дефекте

- Идентификатор (**identifier**) представляет собой уникальное значение, позволяющее однозначно отличить один отчёт о дефекте от другого и используемое во всевозможных ссылках.

Атрибуты отчёта о дефекте

- **Краткое описание (summary)** должно в предельно лаконичной форме давать исчерпывающий ответ на вопросы «Что произошло?», «Где это произошло?», «При каких условиях это произошло?»
- **Подробное описание (description)** представляет в развернутом виде необходимую информацию о дефекте, а также (обязательно!) описание фактического результата, ожидаемого результата и ссылку на требование (если это возможно).

Атрибуты отчёта о дефекте

➤ Шаги по воспроизведению (steps to reproduce)

описывают действия, которые необходимо выполнить для воспроизведения дефекта.

➤ Воспроизводимость (reproducibility)

показывает, при каждом ли прохождении по шагам воспроизведения дефекта удаётся вызвать его проявление. Это поле принимает всего два значения: всегда (always) или иногда (sometimes).

Атрибуты отчёта о дефекте

➤ **Важность (severity)** показывает степень ущерба, который наносится проекту существованием дефекта.

Критическая

Высокая

Средняя

Низкая

Атрибуты отчёта о дефекте

- Срочность (priority) показывает, как быстро дефект должен быть устранён.
 - ❑ Наивысшая
 - ❑ Высокая
 - ❑ Обычная
 - ❑ Низкая

Атрибуты отчёта о дефекте

- **Фактический результат (actual result)** - результат, полученный после прохождения шагов к воспроизведению.
- **Ожидаемый результат (expected result)** - описывает ожидаемое поведение ПО после прохождения шагов к воспроизведению.

Атрибуты отчёта о дефекте

- **Симптом (symptom)** — позволяет классифицировать дефекты по их типичному проявлению.
 - Косметический дефект
 - Повреждение/потеря данных
 - Проблема в документации
 - Некорректная операция
 - Проблема инсталляции
 - Ошибка локализации
 - Нереализованная функциональность
 - Проблема масштабируемости
 - Низкая производительность
 - Крах системы
 - Неожиданное поведение
 - Недружественное поведение
 - Расхождение с требованиями
 - Предложение по улучшению

Атрибуты отчёта о дефекте

- **Комментарий (comments, additional info)** — может содержать любые полезные для понимания и исправления дефекта данные.
- **Приложения (attachments)** — список прикреплённых к отчёту о дефекте приложений (копий экрана, вызывающих сбой файлов и т.д.)

Свойства качественных отчётов о дефектах

- ✓ Тщательное заполнение всех полей точной и корректной информацией.
- ✓ Правильный технический язык.
- ✓ Специфичность описания шагов.
- ✓ Отсутствие лишних действий и/или их длинных описаний.
- ✓ Отсутствие дубликатов.
- ✓ Очевидность и понятность.
- ✓ Прослеживаемость.
- ✓ Отдельные отчёты для каждого нового дефекта.
- ✓ Соответствие принятым шаблонам оформления и традициям.

Человеко-машинное взаимодействие

Лекция 10

Мерзлякова Екатерина Юрьевна

к.т.н. доцент ПМИК

Обеспечение качества и тестирование ПО

- Тестирование требований
- Уточнение требований

Анализ требований
(Requirement Analysis)

- Определение цели тестирования
- Составление тест-плана
- Определение трудоёмкости тестирования
- Составление расписания тестирования

Планирование
тестирования
(Test Planning)

- Составление тест-кейсов, чек-листов
- Создание тестовых данных

Разработка тестов
(Test Development)

- Выполнение тестов
- Создание отчётов по багам (баг-репорт)

Выполнение тестов
(Test Development)

- Составление отчёта по тестированию
- Определение состояния ПО
- Оповещение заинтересованных лиц о результатах тестирования

Оценка результатов
тестирования
(Test Reporting)

Обеспечение качества ПО

- **Обеспечение качества (Quality Assurance - QA)**
 - это совокупность мероприятий, охватывающих все технологические этапы разработки, выпуска и эксплуатации программного обеспечения (ПО) информационных систем, предпринимаемых на разных стадиях жизненного цикла ПО для обеспечения требуемого уровня качества выпускаемого продукта.

Набор стандартов ISO 9126



Верификация и валидация

- **Верификация (verification)** - это процесс оценки системы или её компонентов с целью определения, удовлетворяют ли результаты текущего этапа разработки условиям, сформированным в начале этого этапа. Т.е. выполняются ли наши цели, сроки, задачи по разработке проекта, которые были определены в начале текущей фазы.
- **Валидация (validation)** - это определение соответствия разрабатываемого ПО ожиданиям и потребностям пользователя, требованиям к системе.

Верификация и валидация

№	Верификация	Валидация
1.	Делаем ли мы продукт правильно ?	Делаем ли мы правильный продукт?
2.	Реализована ли вся функциональность?	Правильно ли реализована функциональность?
3.	Верификация происходит раньше и включает проверку правильности написания документации, кода и т.д.	Валидация происходит после верификации и, как правило, отвечает за оценку продукта в целом.
4.	Производится разработчиками	Производится тестировщиками
5.	Включает статический анализ – инспектирование кода, сравнение требований и т.п.	Включает динамический анализ – выполнение программы для сравнения ее реальной работы с установленными требованиями
6.	Основывается на объективной оценке соответствия реализованных функций	Субъективный процесс, включающий личную оценку качества работы ПО

Откуда берутся ошибки в ПО?

- **Ошибка (error)** – это действие человека, которое порождает неправильный результат.
- **Дефект, Баг (Defect, Bug)** – недостаток компонента или системы, который может привести к отказу определенной функциональности. Дефект, обнаруженный во время исполнения программы, может вызвать отказ отдельного компонента или всей системы.
- **Сбой (failure)** – несоответствие фактического результата (actual result) работы компонента или системы ожидаемому результату (expected result).

Причины появления дефектов в программном коде

- 1. Недостаток или отсутствие общения в команде.
- 2. Сложность программного обеспечения.
- 3. Изменения требований.
- 4. Плохо документированный код.
- 5. Средства разработки ПО.

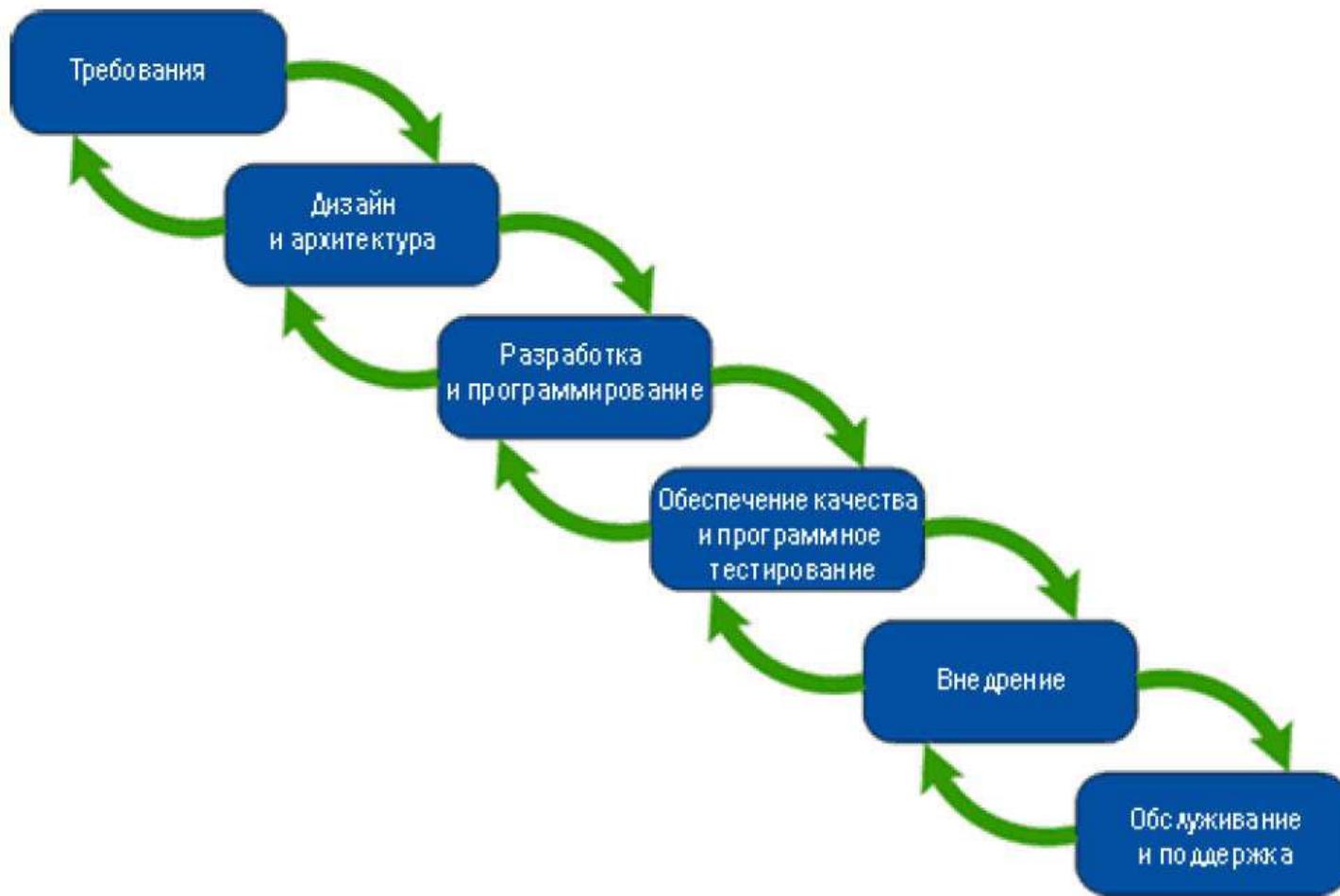
Принципы тестирования

1. Тестирование показывает наличие дефектов.
2. Исчерпывающее тестирование невозможно.
3. Раннее тестирование.
4. Скопление дефектов.
5. Парадокс пестицида.
6. Тестирование зависит от контекста.
7. Заблуждение об отсутствии ошибок.

Принципы тестирования

- тестирование должно производиться независимыми специалистами;
- привлекайте лучших профессионалов;
- тестируйте как позитивные, так и негативные сценарии;
- не допускайте изменений в программе в процессе тестирования;
- указывайте ожидаемый результат выполнения тестов.

Жизненный цикл ПО



ISO / IEC / IEEE 12207: 2017

Systems and Software development — software lifecycle processes

Общая информация

Статус :  Опубликовано

Дата публикации : 2017-11

Издание : 1

Количество страниц : 145

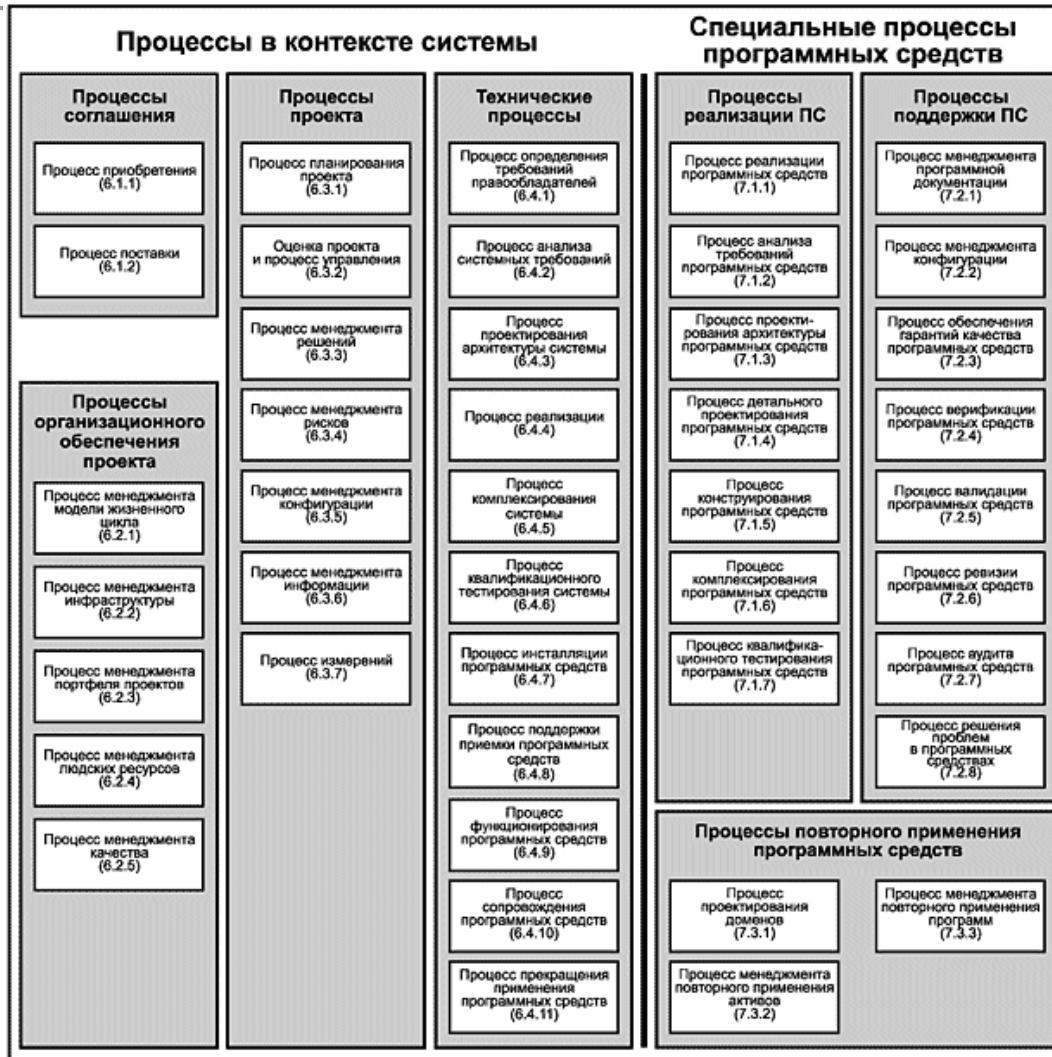
Технический комитет : ISO / IEC JTC 1 / SC 7 Разработка программного обеспечения и систем

ICS : 35.080 Программное обеспечение

ГОСТ Р ИСО/МЭК 12207-2010
НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ
Информационная технология
Системная и программная инженерия
ПРОЦЕССЫ ЖИЗНЕННОГО ЦИКЛА ПРОГРАММНЫХ СРЕДСТВ

1. Устанавливает общую структуру процессов жизненного цикла программных средств, на которую можно ориентироваться в программной индустрии.
2. Определяет процессы, виды деятельности и задачи, которые используются при приобретении программного продукта или услуги, а также при поставке, разработке, применении по назначению, сопровождении и прекращении применения программных продуктов.
3. Устанавливает процесс, который может использоваться при определении, управлении и совершенствовании процессов жизненного цикла программных средств
4. Разработан для сторон, приобретающих системы, программные продукты и услуги, а также для поставщиков, разработчиков, операторов, сопровожденцев, менеджеров (в том числе, менеджеров по качеству) и пользователей программных продуктов.

ГОСТ Р ИСО/МЭК 12207-2010
НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ
Информационная технология
Системная и программная инженерия
ПРОЦЕССЫ ЖИЗНЕННОГО ЦИКЛА ПРОГРАММНЫХ СРЕДСТВ

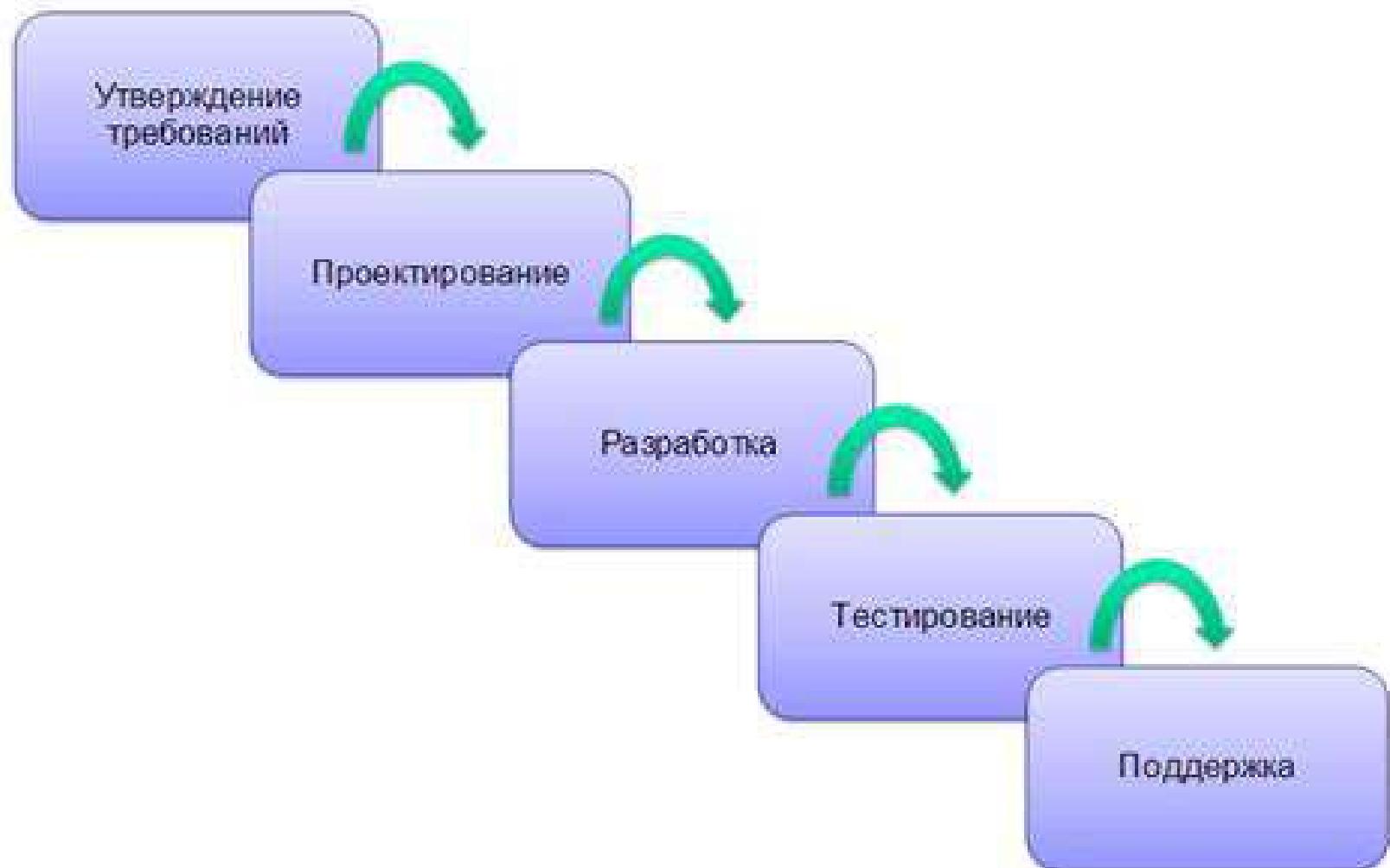


Группы процессов жизненного цикла

Модели разработки ПО

- Каскадная
- v-образная
- Итерационная
- Инкрементальная
- Спиральная
- Гибкая

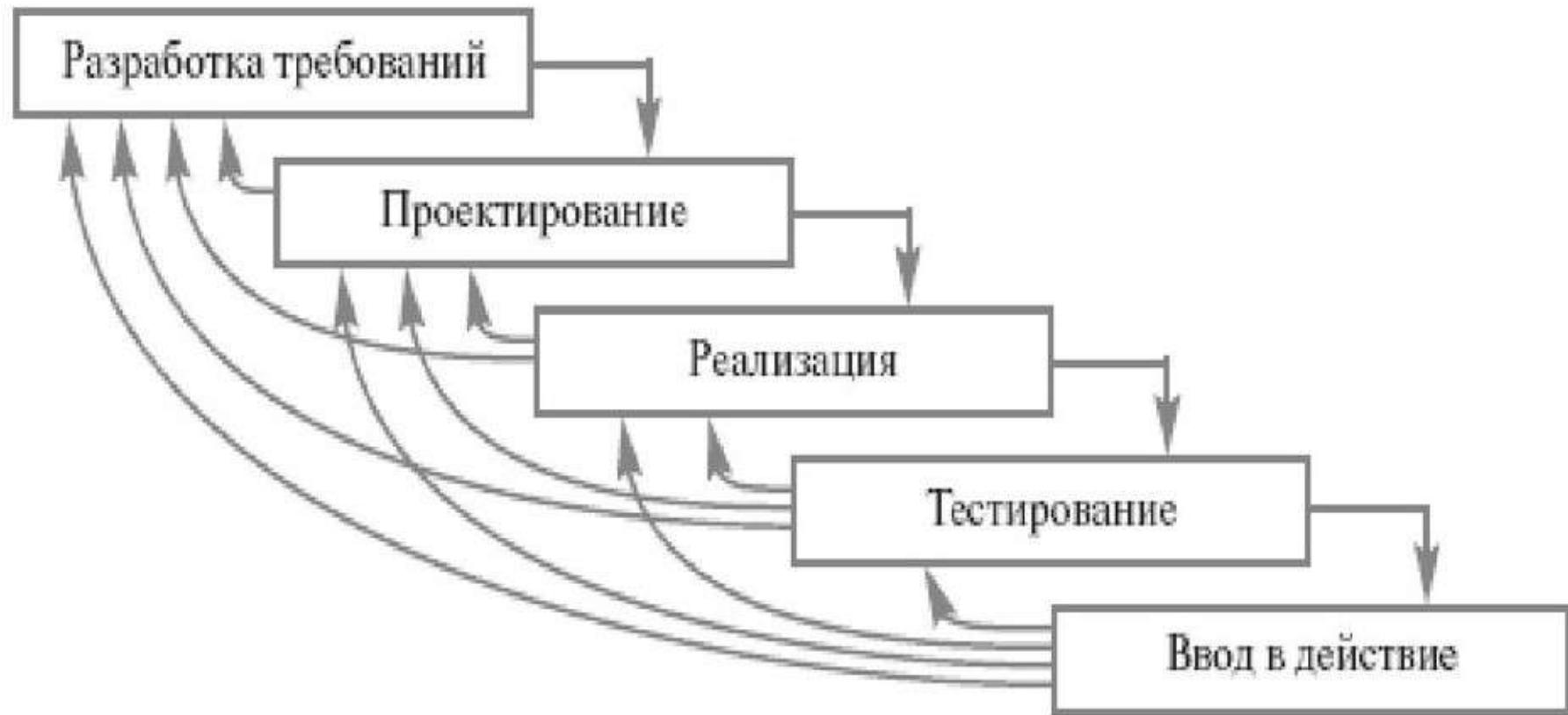
Каскадная модель (waterfall)



Особенности каскадной модели

- высокий уровень формализации процессов;
- большое количество документации;
- жесткая последовательность этапов жизненного цикла без возможности возврата на предыдущий этап.

Итеративные или инкрементальные модели



Спиральная модель



V - модель



Agile «гибкая» модель

МОДЕЛИ



Методологии



12 принципов Agile

1. Регулярно и как можно раньше удовлетворять потребности заказчика, предоставляя ему программное обеспечение.
2. Учитывать, что требования могут измениться на любом этапе разработки.
3. Выпускать версии готовой программы как можно чаще.
4. Ежедневно вместе работать над проектом — разработчикам и заказчикам.
5. Поручить работу мотивированным профессионалам.
6. Общаться напрямую.
7. Считать главным показателем прогресса работающий продукт.
8. Поддерживать постоянный ритм работы.
9. Уделять пристальное внимание техническому совершенству и качеству проектирования.
10. Минимизировать лишнюю работу.
11. Стремиться к самоорганизующейся команде.
12. Всем участникам команды — постоянно искать способы повышать эффективность работы.

Человеко-машинное взаимодействие

Лекция 8

Мерзлякова Екатерина Юрьевна
к.т.н. доцент ПМИК

Система помощи

 Help: Catalog Help

[Home](#) [Back](#) [Close](#)

Catalog Help



Introduction

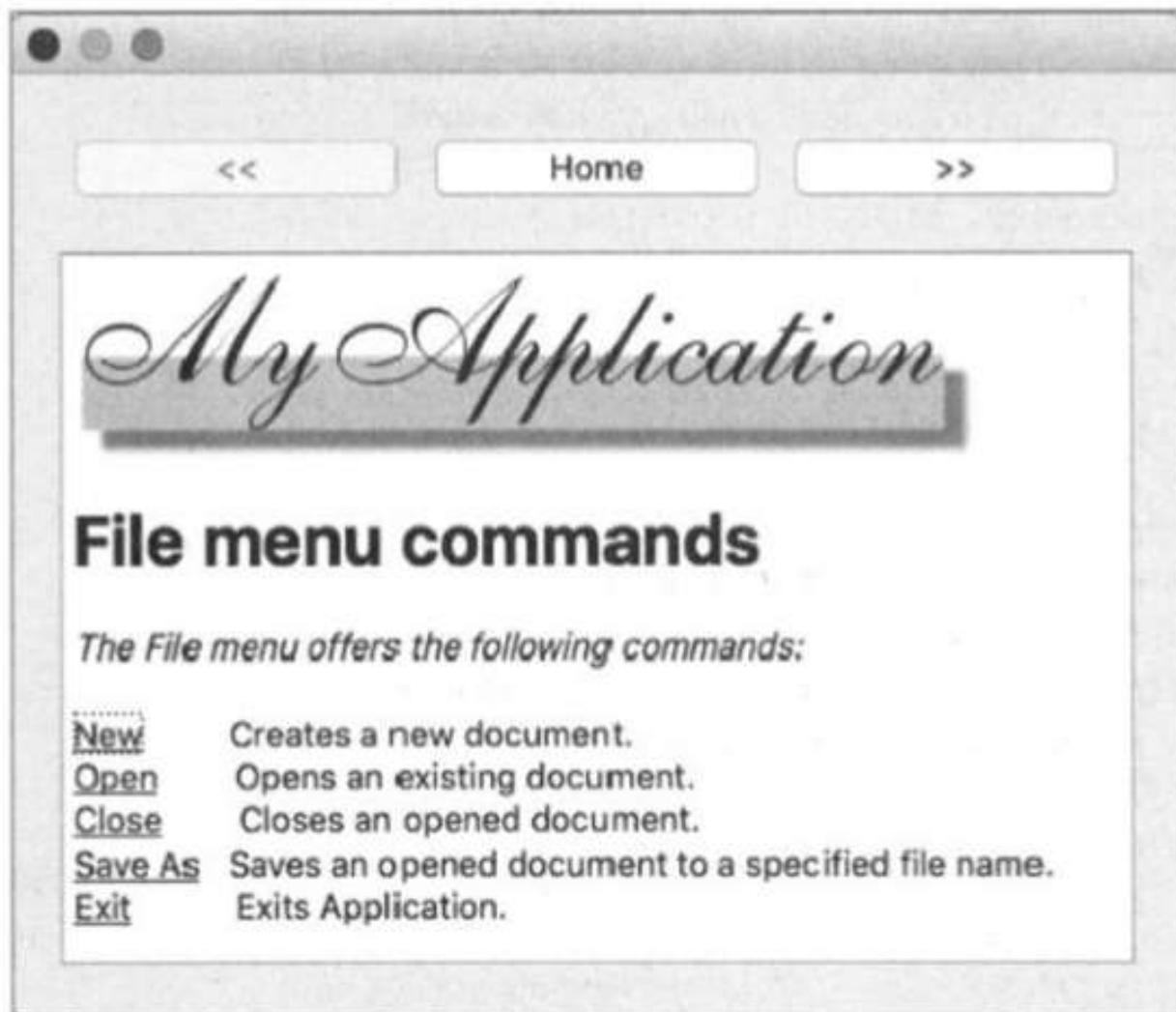
The Catalog application is used to maintain a bibliographic database of books, articles, thesis and other reference material used by researchers.

The application has facilities for adding, editing and deleting catalog entries, and for searching for particular entries.

Contents

1. [Adding New Entries](#)
2. [Editing Existing Entries](#)
3. [Deleting Entries](#)
4. [Searching for Entries](#)

QTextBrowser



QTextBrowser

```
#include < QApplication>
#include "HelpBrowser.h"

// -----
int main (int argc, char** argv)
{
    QApplication app(argc, argv);
    HelpBrowser helpBrowser(":/", "index.htm");

    helpBrowser.resize(450, 350);
    helpBrowser.show();

    return app.exec();
}
```

QTextBrowser

```
#include <QtWidgets>

// =====
class HelpBrowser : public QWidget {
    Q_OBJECT

public:
    HelpBrowser(const QString& strPath,
                const QString& strFileName,
                QWidget* pwgt      = 0
                ) : QWidget(pwgt)
    {
        QPushButton* pcmdBack     = new QPushButton("<<");
        QPushButton* pcmdHome     = new QPushButton("Home");
        QPushButton* pcmdForward = new QPushButton(">>");
        QTextBrowser* ptxtBrowser = new QTextBrowser;
```

QTextBrowser

```
connect (pcmdBack, SIGNAL(clicked()),  
         ptxtBrowser, SLOT(backward()) )  
    );  
  
connect (pcmdHome, SIGNAL(clicked()),  
         ptxtBrowser, SLOT(home()) )  
    );  
  
connect (pcmdForward, SIGNAL(clicked()),  
         ptxtBrowser, SLOT(forward()) )  
    );
```

QTextBrowser

```
//  
connect(ptxtBrowser, SIGNAL(backwardAvailable(bool)),  
       pcmdBack, SLOT(setEnabled(bool))  
);  
connect(ptxtBrowser, SIGNAL(forwardAvailable(bool)),  
       pcmdForward, SLOT(setEnabled(bool))  
);
```

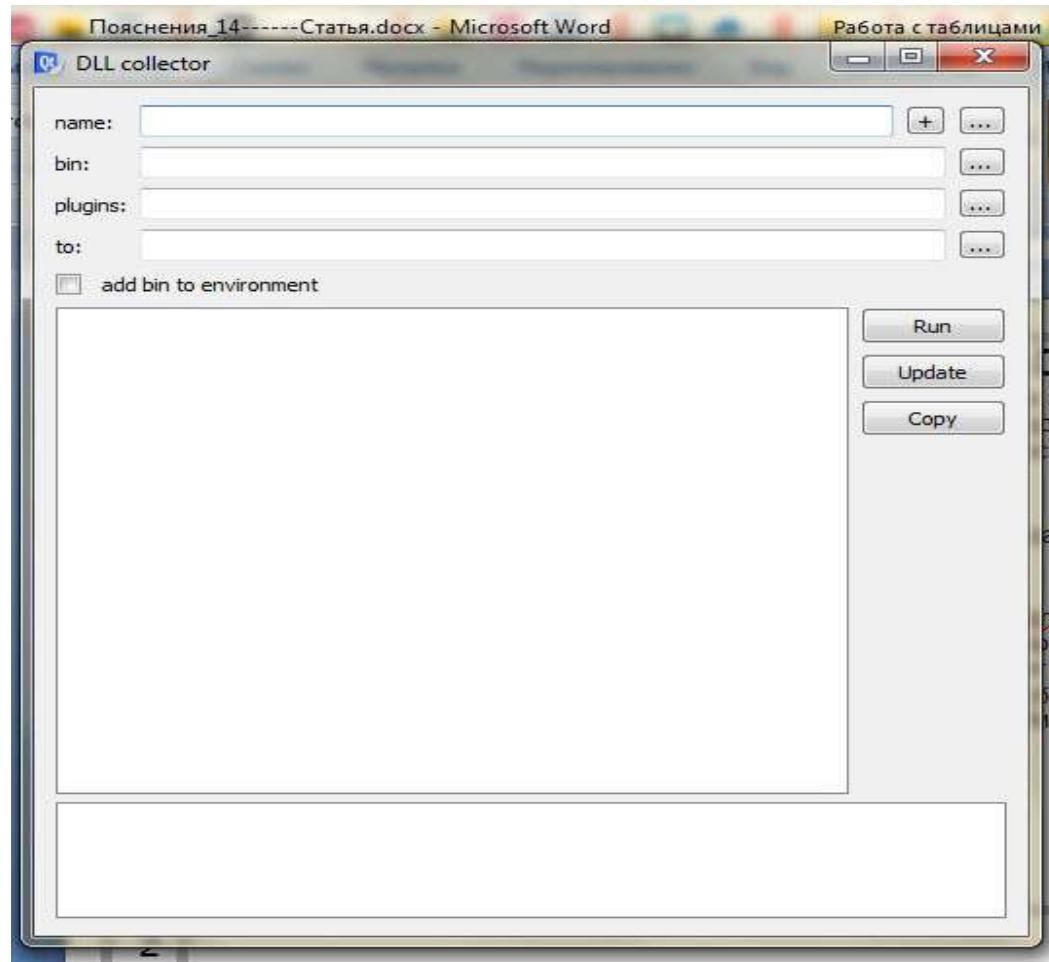
QTextBrowser

```
ptxtBrowser->setSearchPaths (QStringList() << strPath);  
ptxtBrowser->setSource (QString(strFileName)) ;
```

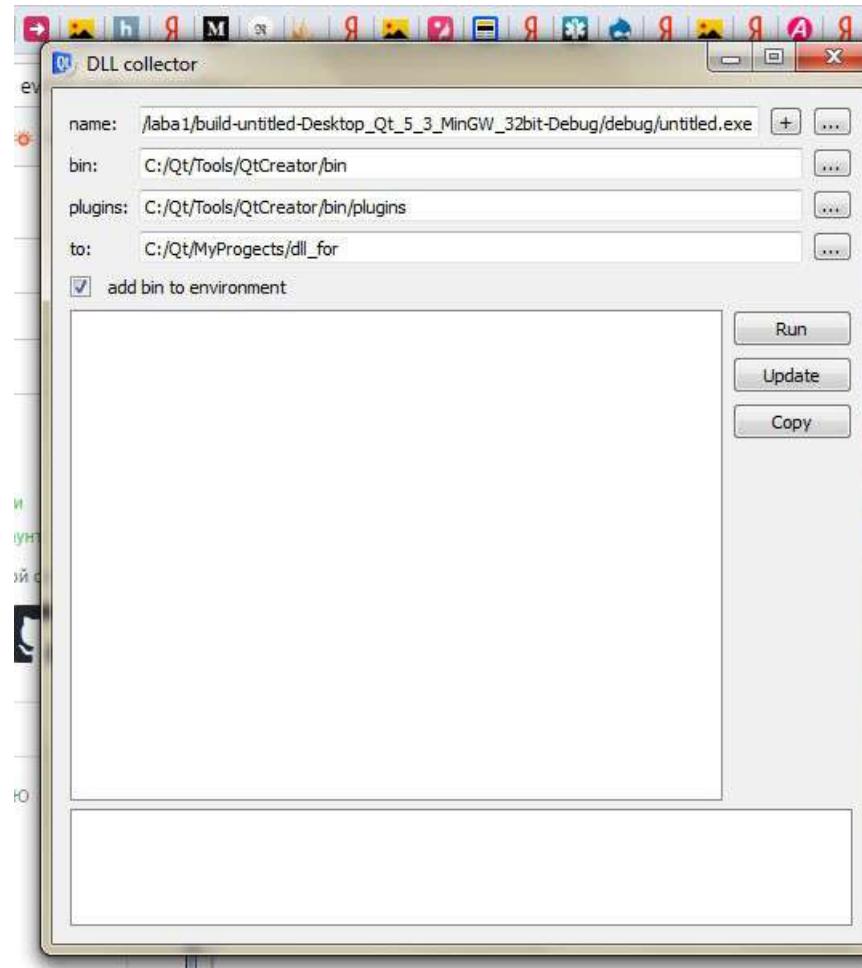
QTextBrowser

```
//Layout setup  
QVBoxLayout* pvbxLayout = new QVBoxLayout;  
QHBoxLayout* phbxLayout = new QHBoxLayout;  
phbxLayout->addWidget (pCmdBack) ;  
phbxLayout->addWidget (pCmdHome) ;  
phbxLayout->addWidget (pCmdForward) ;  
pvbxLayout->addLayout (phbxLayout) ;  
pvbxLayout->addWidget (pTxtBrowser) ;  
setLayout (pvbxLayout) ;  
};  
};
```

Инсталлятор для Qt-приложения. утилита DLL Collector



Как работает DLL Collector

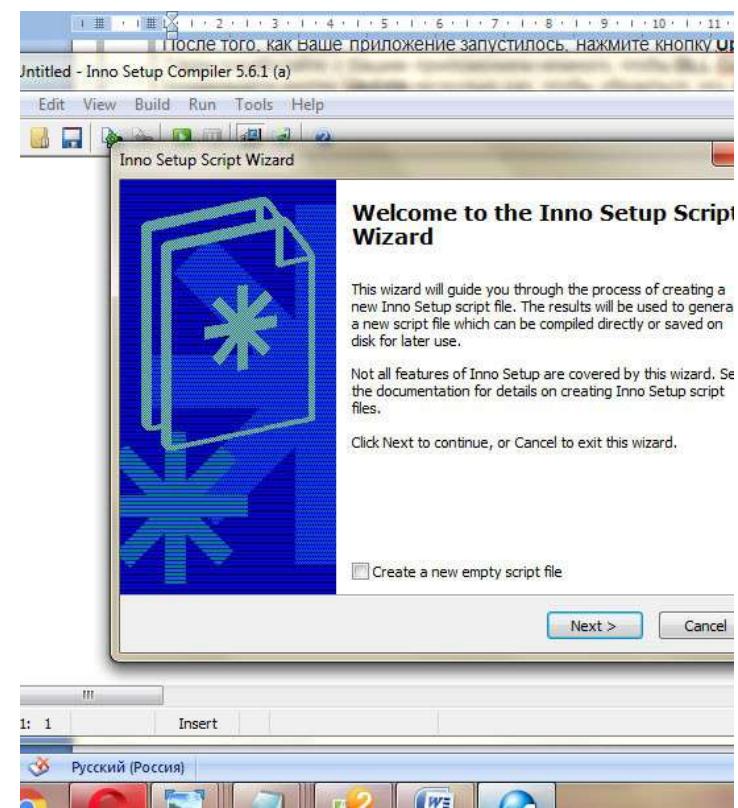
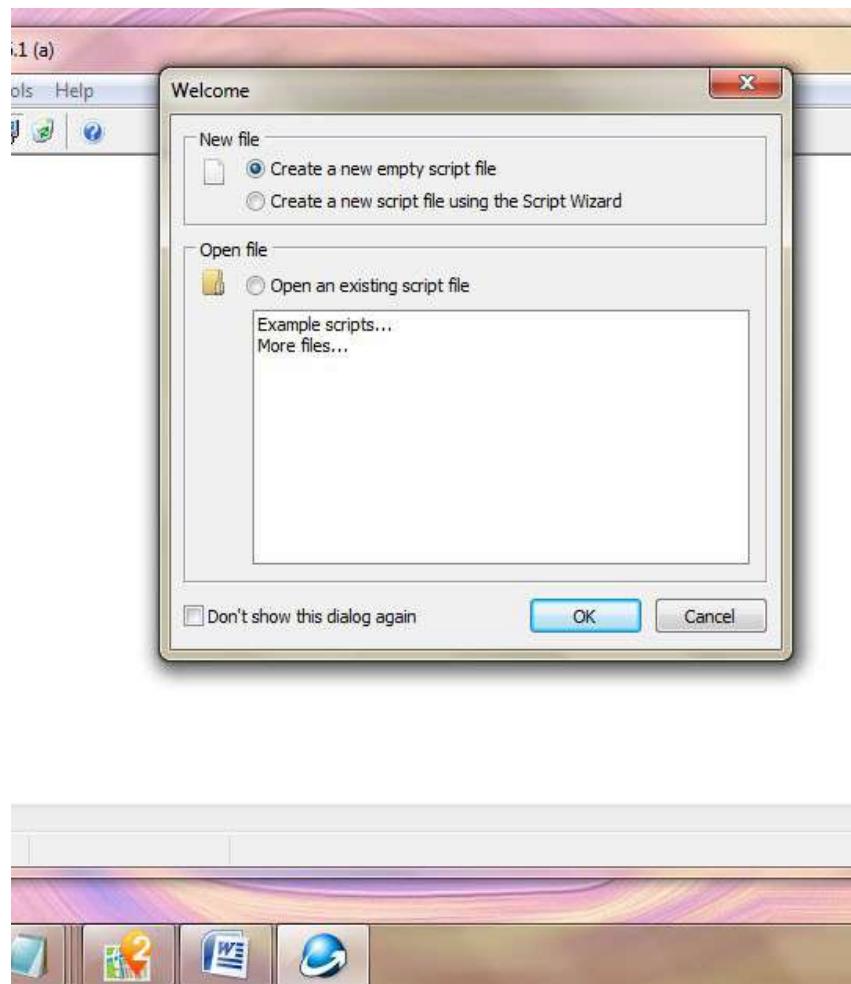


Как работает DLL Collector

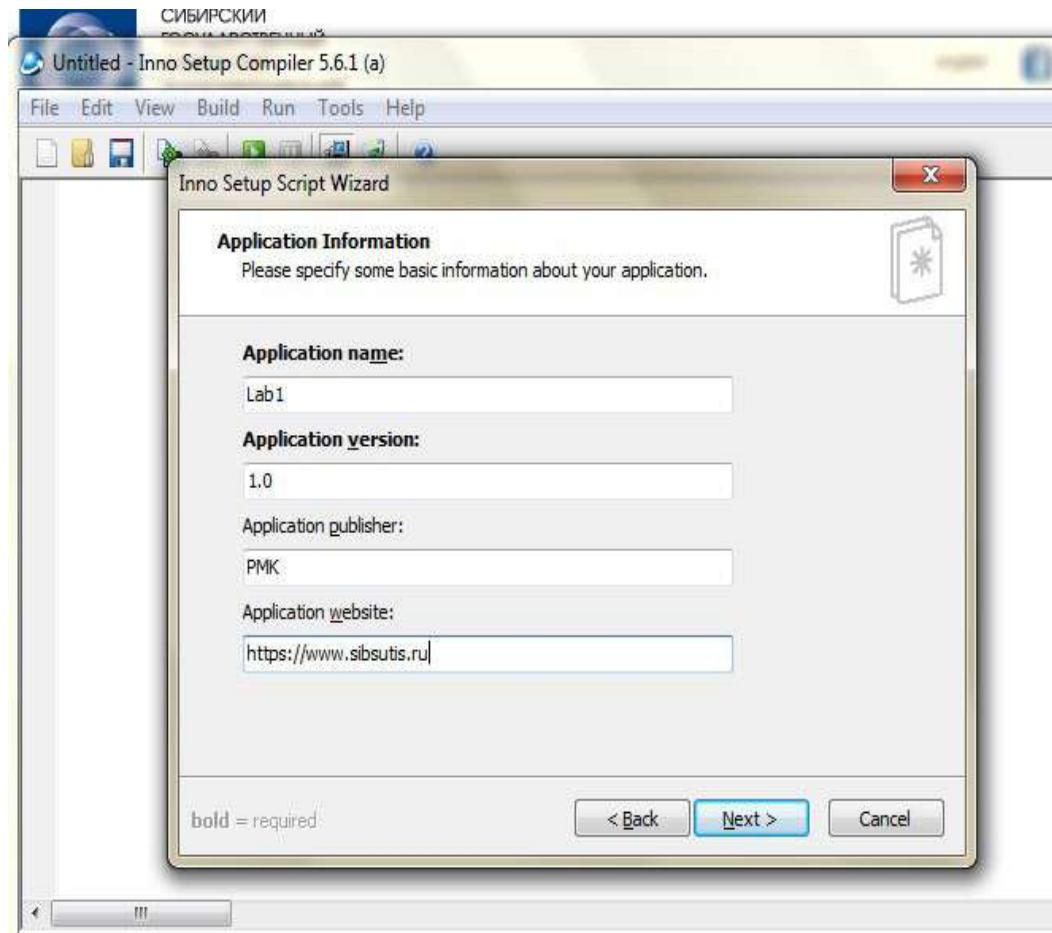
- 1) запускаем Ваше приложение нажатием кнопки **Run**.
- 2) нажмите кнопку **Update**. Высветится список плагинов и библиотек. Также поработайте с Вашим приложением немного, чтобы **DLL Collector** наверняка обнаружил все зависимости и понажимайте кнопку **Update** несколько раз, чтобы убедиться, что все зависимости найдены.
- 3)После того, как библиотеки найдены, нажмите кнопку **Copy**, и все библиотеки будут скопированы.

<http://www.cyberforum.ru/blogs/131347/blog2457.html>

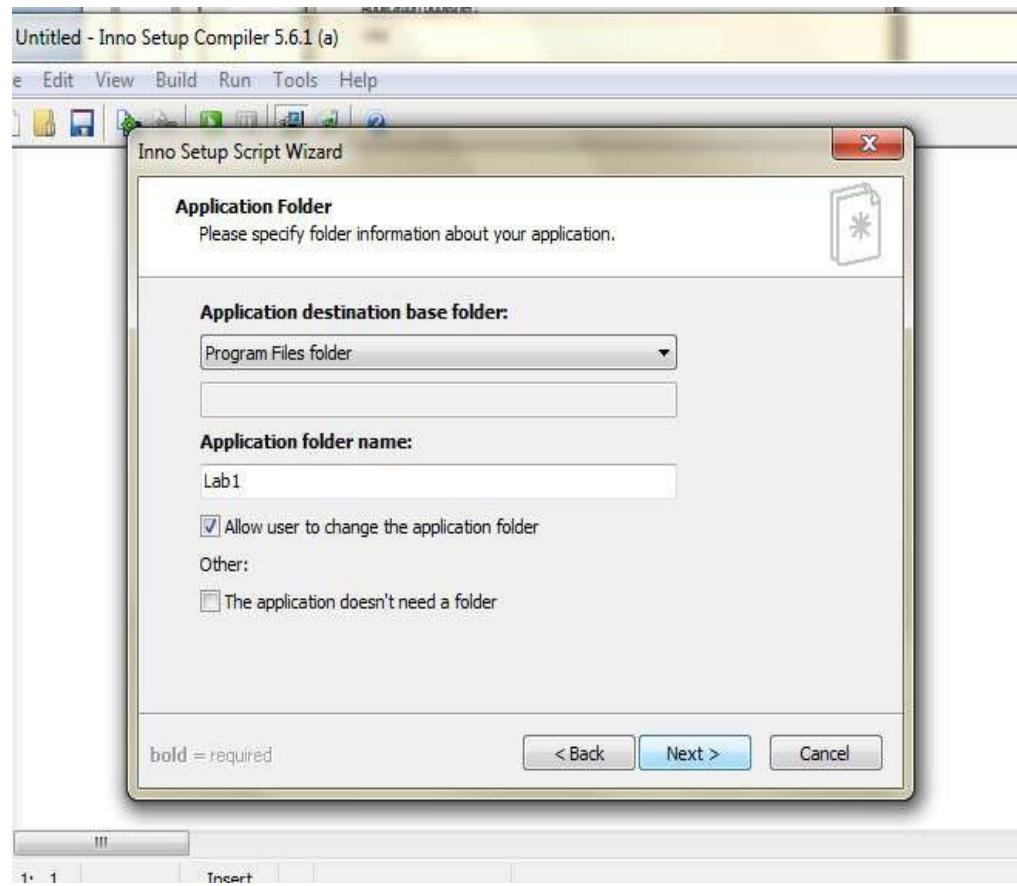
Утилита InnoSetup



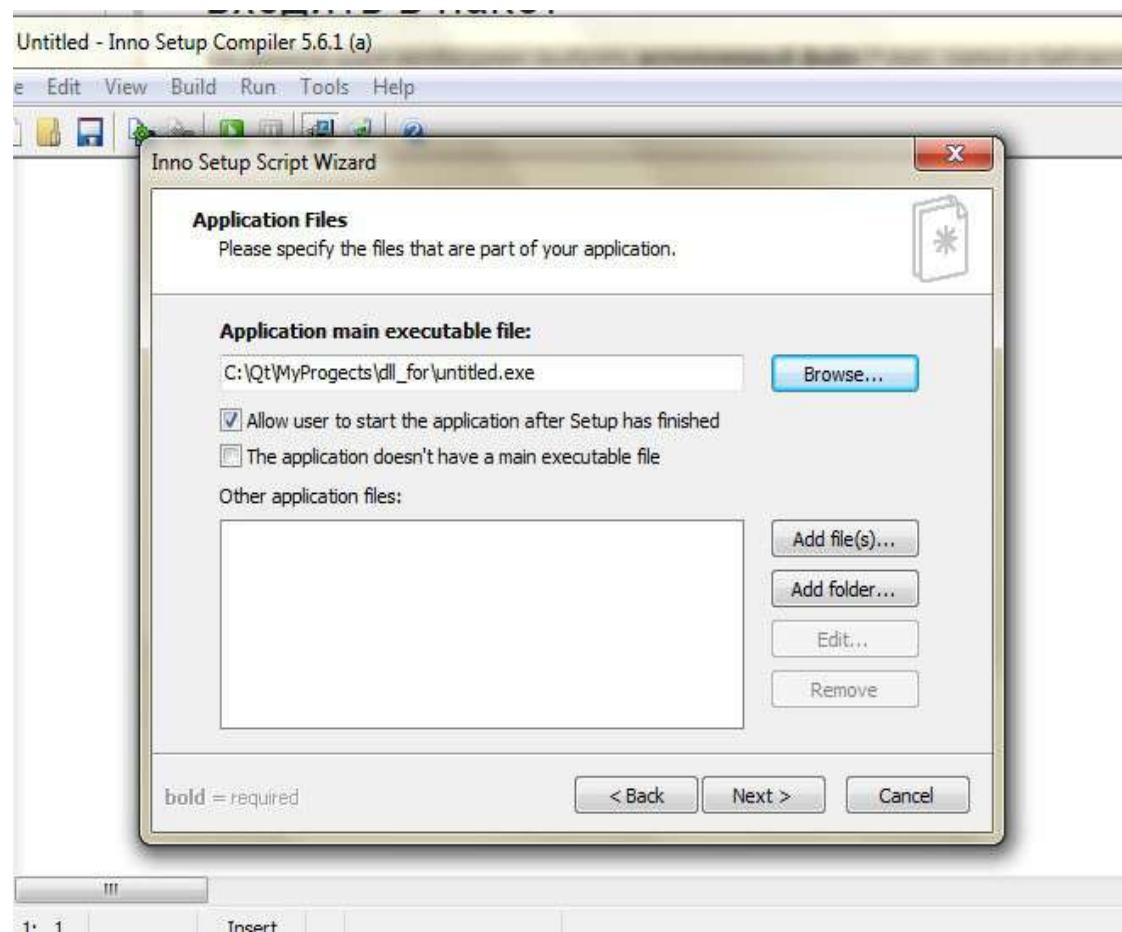
Утилита InnoSetup



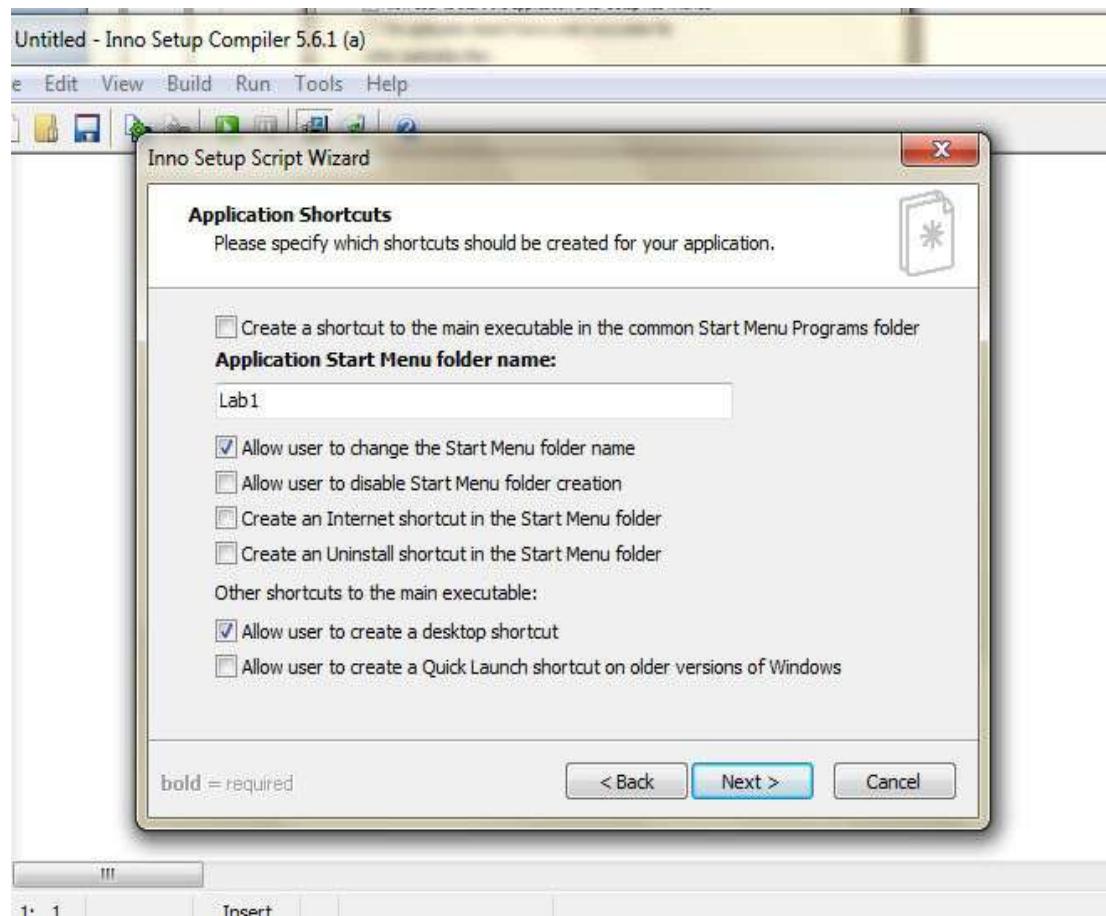
Утилита InnoSetup



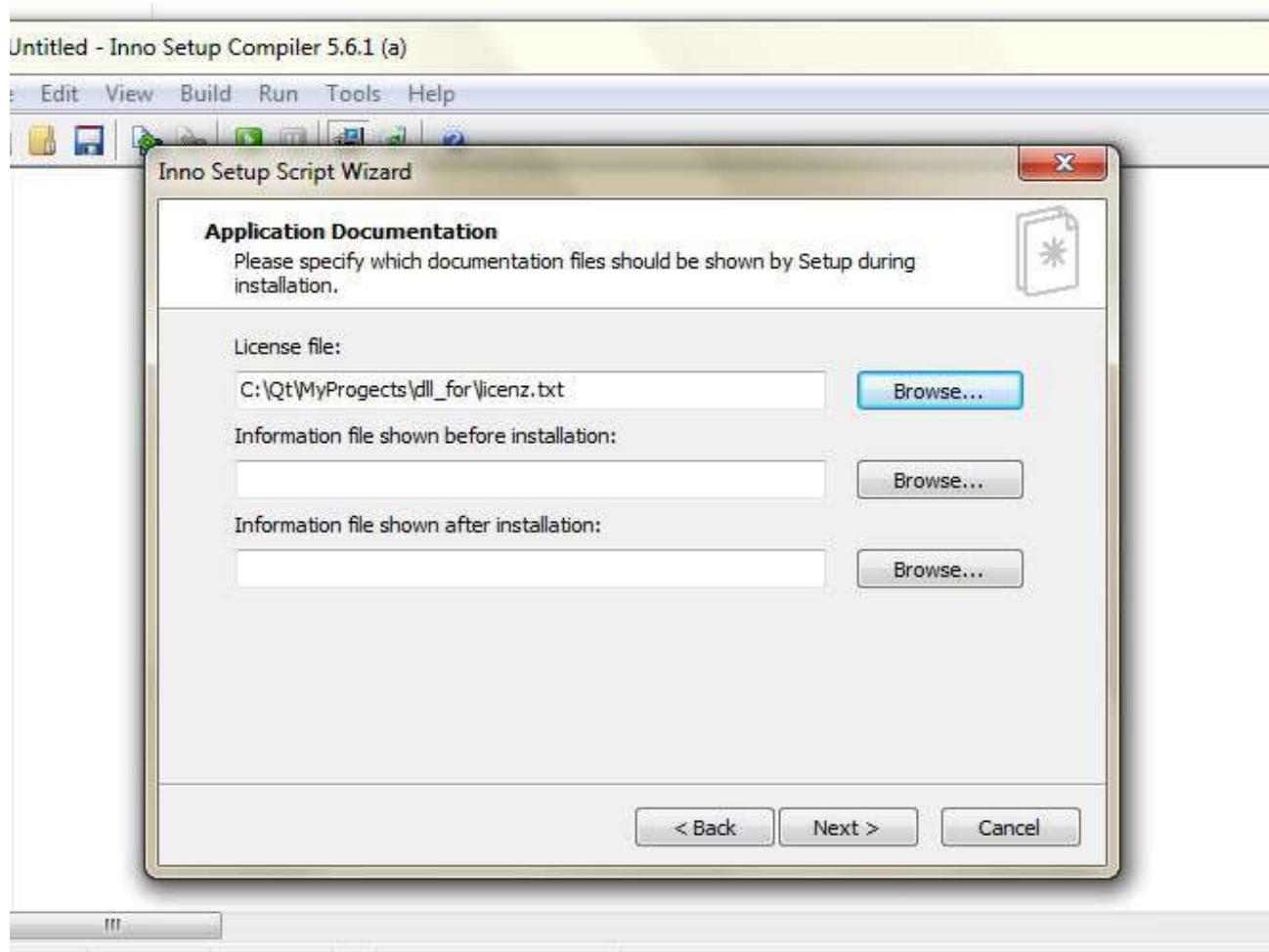
Утилита InnoSetup



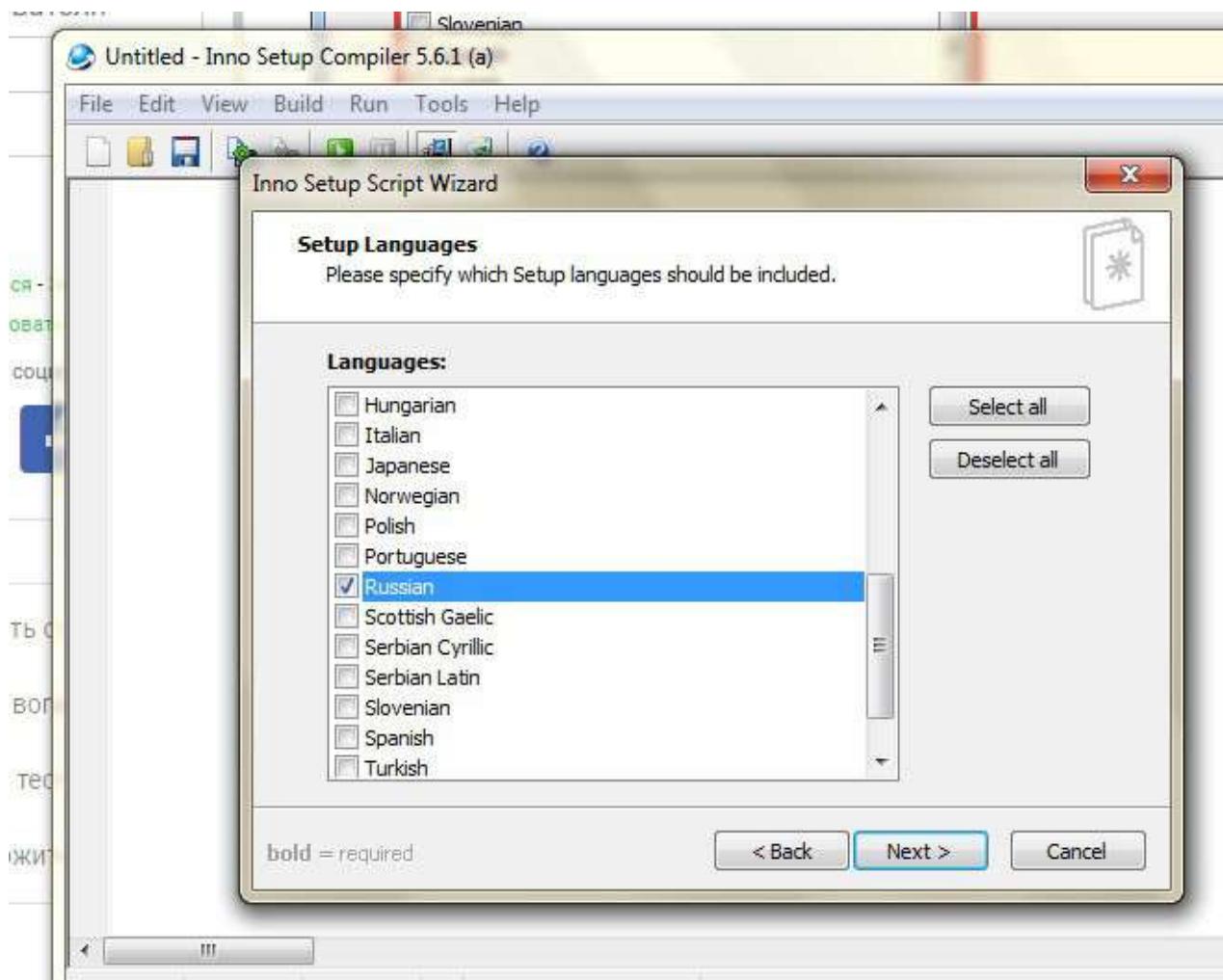
Утилита InnoSetup



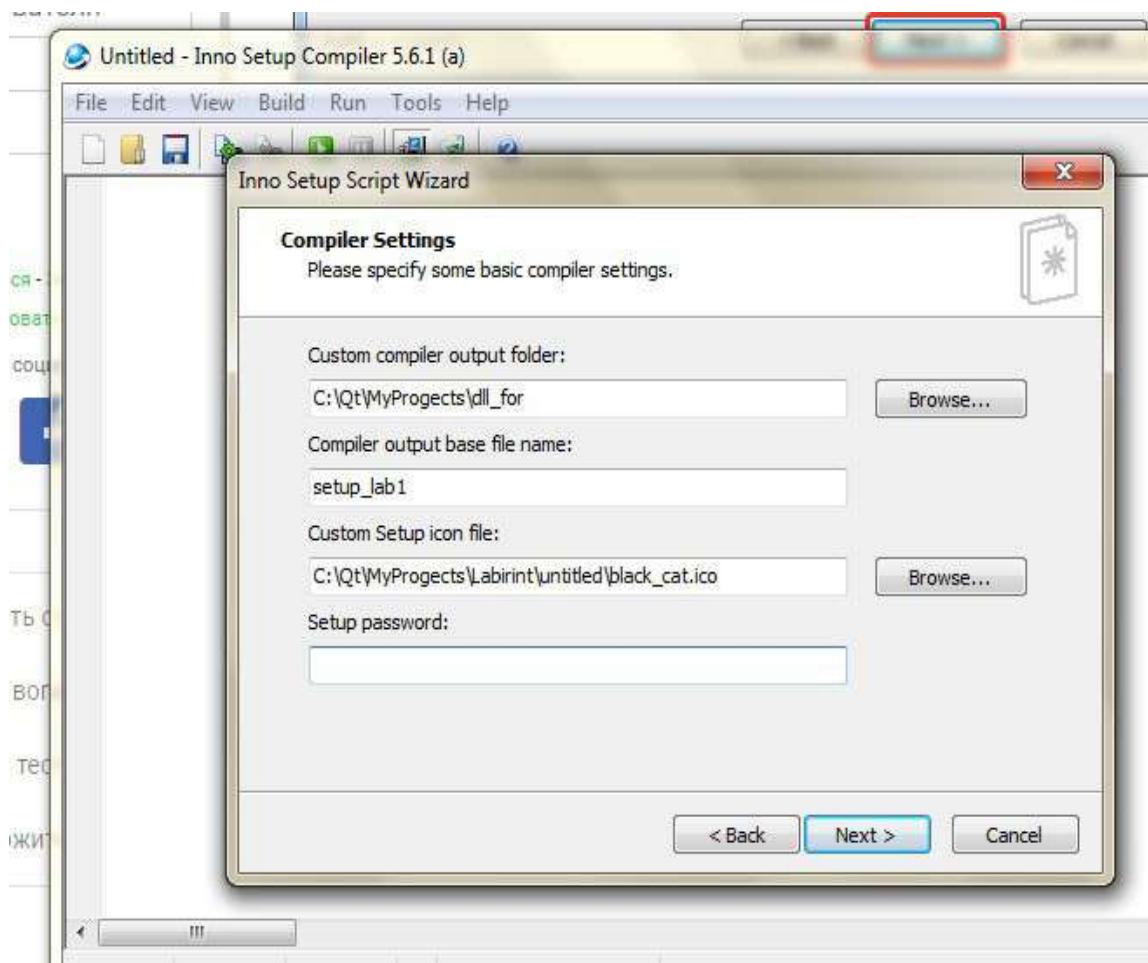
Утилита InnoSetup



Утилита InnoSetup



Утилита InnoSetup



Утилита InnoSetup

The screenshot shows the Inno Setup Compiler 5.6.1 application window. At the top, there's a file browser with two entries: 'licenz.txt' and 'setup_lab1.exe'. Below the browser is the main window title bar: 'Untitled - Inno Setup Compiler 5.6.1 (a)'. The menu bar includes File, Edit, View, Build, Run, Tools, and Help. The toolbar contains icons for opening files, saving, printing, and running scripts. The main code editor area displays the following Inno Setup script:

```
; Script generated by the Inno Setup Script Wizard.  
; SEE THE DOCUMENTATION FOR DETAILS ON CREATING INNO SETUP SCRIPT  
  
#define MyAppName "Lab1"  
#define MyAppVersion "1.0"  
#define MyAppPublisher "PMK"  
#define MyAppURL "https://www.sibsutis.ru"  
#define MyAppExeName "untitled.exe"  
  
[Setup]  
; NOTE: The value of AppId uniquely identifies this application.  
; Do not use the same AppId value in installers for other applications.  
; (To generate a new GUID, click Tools | Generate GUID inside the  
AppId={{86722434-5154-4BC5-8D62-4CAA413A18E2}  
AppName=#MyAppName}  
  
Parsing [Icons] section, line 44  
Parsing [Run] section, line 47  
Parsing [Files] section, line 39  
Creating setup files  
    Updating icons (SETUP.EXE)  
    Compressing: C:\Qt\MyProjects\dll_for\untitled.exe  
    Compressing Setup program executable  
    Updating version info  
  
*** Finished. [0:56:43, 00:01,685 elapsed]
```

Утилита InnoSetup

	Дата изменения	Тип	Размер
КЦИЯ№8.pptx	03.04.2015 23:30	Презентация Mic...	1 298 КБ
КЦИЯ№9.pptx	10.04.2015 20:27	Презентация Mic...	3 288 КБ
КЦИЯ№10.pptx			
КЦИЯ№11.pptx			
КЦИЯ№12.pptx			
КЦИЯ№13.pptx			
ентиация Mic...			
т и SQL.ppt			
ложение (1)			
LiteManager4Setu...			
нок JPEG (19)			
11130084838_1.jp...			
еномитбсвб9м31...			
InAYjD8no.jpg			
l_1_max.jpg			
7.jpg			
'6807306_yvdgev9...			
ernovik.jpg			
-kit-psd_36-2020...			
g			
titled-1.jpg	30.10.2018 0:59	Рисунок JPEG	118 КБ
акива1.jpg	03.04.2015 14:16	Рисунок JPEG	74 КБ

Установка — Lab1, версия 1.0

Лицензионное Соглашение

Пожалуйста, прочтите следующую важную информацию перед тем, как продолжить.

Пожалуйста, прочтите следующее Лицензионное Соглашение. Вы должны принять условия этого соглашения перед тем, как продолжить.

Лицензионное соглашение

Я принимаю условия соглашения
 Я не принимаю условия соглашения

Далее > **Отмена**

Анализ юзабилити сайта



Анализ юзабилити сайта



Анализ юзабилити сайта

The screenshot shows the homepage of OZON.ru. At the top, there's a banner with a deal for an electric toothbrush: "-50% Электрическая зубная щётка Philips Sonicare 2 490 ₽". Below the banner are three main categories: "Техника по-крупному" (large kitchen appliances), "С книгой в руках" (books), and "Обучение с увлечением" (learning). Further down, there's a section for "Бытовая техника" (household appliances) and a "Скидки на электронику" (discounts on electronics) section featuring items like a fitness tracker, a speaker, and a smartphone.

https://www.ozon.ru/

Бесплатная доставка. Заказывайте в приложении

Москва + 1083 пункта выдачи заказов Условия доставки Условия оплаты Помощь

Город доставки ваших покупок — Москва? Да, все верно Нет, изменить город

+7 495 730-57-57

My OZON Заказы Избранное Корзина

Все разделы Все акции Электроника Бытовая техника Дом и сад Детские товары Книги Одежда, обувь и аксессуары Красота и здоровье Спорт и отдых

ooo! И другие крутые предложения Смотреть все

-50% Электрическая зубная щётка Philips Sonicare 2 490 ₽ 4 980 ₽

Бесплатная доставка
Заказывайте в приложении

PHILIPS -50% На технику Philips

Техника по-крупному

Техника для кухни Техника для уборки Уход за одеждой Техника для красоты Крупная техника

С книгой в руках

Бестселлеры Путеводители Школьная программа Бизнес-литература

Бытовая техника

Для кухни Для дома Для красоты Климатическая техни...

Обучение с увлечением

Методики Развитие речи Развитие моторики Сюжетные игры Развитие внимания

Доставка по всей России 20 способов оплаты Подарочные сертификаты Мобильное приложение OZON Гид

Скидки на электронику

JBL

Что отпугивает

Регистрация

Имя:

Фамилия:

*Логин (мин. 3 символа):

Пароль:

*Подтверждение пароля:

*E-Mail:

 my@mai.ru

Пароль должен быть не менее 6 символов длиной.

*Обязательные поля

[Вернуться на главную страницу](#)

✓ Регистрируясь, вы выражаете согласие с [политикой конфиденциальности](#) нашей компании.

Самые распространенные ошибки

The screenshot shows the homepage of the Logos Group website. At the top, there is a blue header bar with the text "Самые распространенные ошибки". Below this, the main content area features the company logo "ЛОГОС" (Logos) and the text "группа компаний" (Group of Companies). To the right, there is a banner with the text "25 лет" (25 years) and "мы работаем для Вас!" (We work for you!). Further right is a phone icon followed by the phone number "+7 (383) 353-14-40" and the text "г. Новосибирск" (Novosibirsk). There is also a button labeled "ЗАДАТЬ ВОПРОС" (Ask a question). Below the header, there is a navigation menu with links: "О КОМПАНИИ", "СЕТЬ МАГАЗИНОВ", "ИНТЕРНЕТ-МАГАЗИН", "КЛУБ РУКОДЕЛИЯ", "КАРЬЕРА", "КОНТАКТЫ", and a search icon. A search bar with the placeholder "Найти" (Find) and a pink "НАЙТИ" (Find) button are located below the menu. The main title "Интернет-магазин" (Internet-Marketplace) is displayed prominently.

Интернет-магазин

Главная > Интернет-магазин

Ткани оптом >

В оптовом интернет-магазине представлены различные виды тканей, меха, кожа, а также швейная фурнитура и бытовая швейная техника.

Швейное оборудование >

В интернет-магазине бытового швейного оборудования представлены столы для швейных машин, оверлоки, швейные механические и электро-механические машины, машины для рукоделия и аксессуары, манекены, гладильные доски, оборудования для отпаривания одежды и парогенераторы.

Рукоделие >

Удобно

ПЕРСОНАЛЬНЫЙ МЕНЕДЖЕР

ШИРОКИЙ ВЫБОР АССОРТИМЕНТА

КАЧЕСТВЕННЫЕ ТКАНИ И ФУНКЦИЯ

15 МИНУТ С ПОСЛЕДНЕГО ЗАКАЗА

СЕГОДНЯ ОБРАБОТАНО 1001 МЕТРОВ ТКАНИ

НОВЫЕ ПОСТУПЛЕНИЯ

Все категории | Одежные материалы | Дополнительные

Штапель купон цветы 248.00 ₽

Плательная жаккард 252.00 ₽

Софти стрейч горох 238.00 ₽

Штапель горох 248.00 ₽

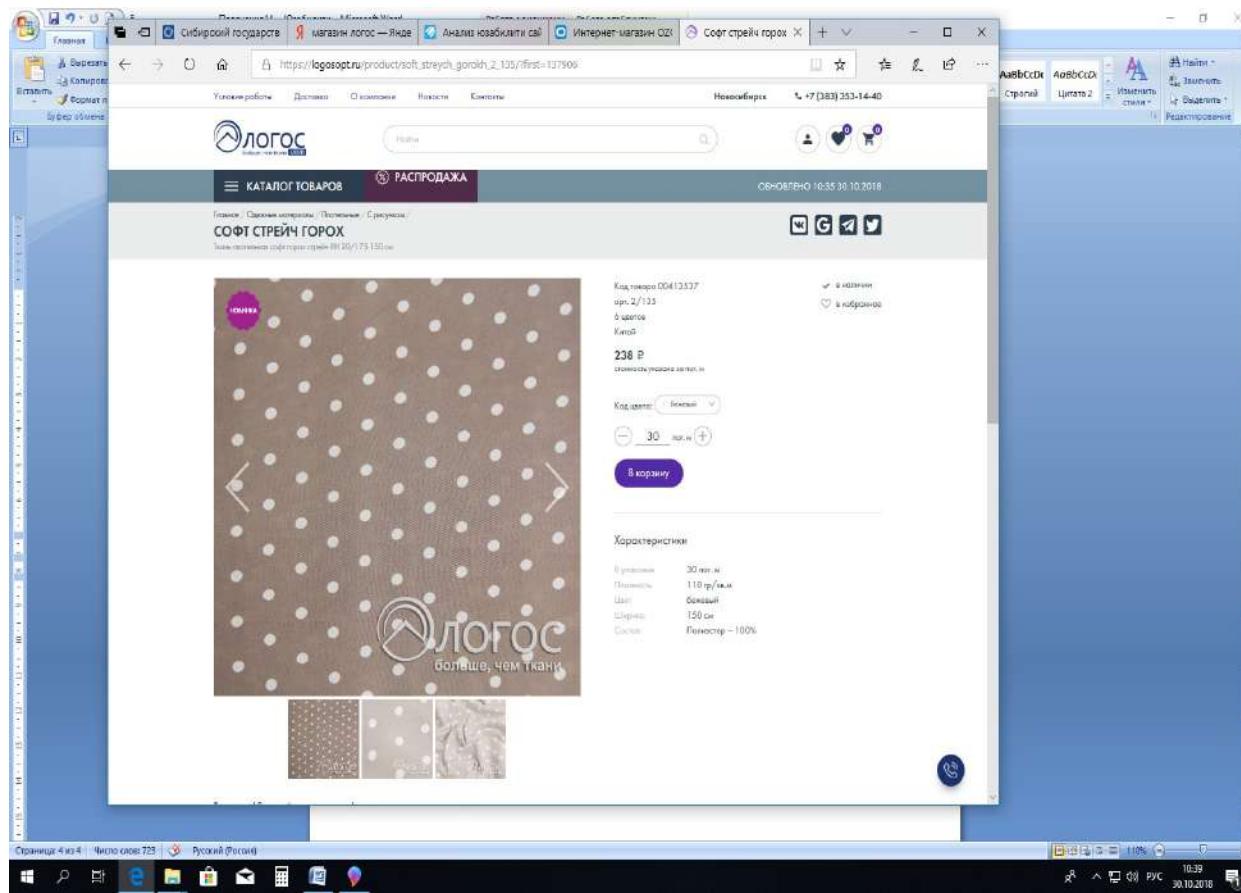
Сорочка с рисунком 298.00 ₽

Гобордин

Фаттин

Трикотаж

Просто



Ошибка

Не удобно

The screenshot shows the homepage of the Leonardo.ru website. At the top, there is a large banner with a floral pattern. Below it, the main navigation bar includes links for Главная, Доставка, Оплата, Контакты, and a dropdown menu for Новосибирск. The phone number 8 (800) 700-51-27 is prominently displayed, along with working hours from 9:00 to 21:00 and a note about free delivery. A "Корзина" (Cart) button shows 0 items and 0 rubles. The central part of the page features a large image of people at a craft fair, with text encouraging visitors to attend master classes at Leonardo stores. On the left, there is a sidebar with sections for "Ближайшие фестивали" (Upcoming festivals) listing events in Moscow, St. Petersburg, and Kursk; "Ищем мастеров" (Looking for masters) for hobbyists; and "Открытие новых магазинов" (Opening of new stores). The main content area has sections for "Новости" (News) and "Конкурсы" (Competitions), each with several items listed. At the bottom, there is a footer with a link to the shop page and a "Сообщите нам о проблеме" (Tell us about the problem) button.

Не удобно

Главная Доставка Оплата Контакты

Новосибирск | Вход Регистрация

Адреса магазинов
Расписание мастер-классов

8 (800) 700-51-27
ежедневно с 9.00 до 21.00, звонок по РФ бесплатный

Корзина
0 товаров | 0 руб.

Каталог товаров

краски

Интернет-магазин

Оформить заказ

blaze - для тех КТО В ТЕМЕ
БЫТОВЫЕ ШВЕЙНЫЕ МАШИНЫ

О нас

Наши магазины
Наличие и цены
Вакансии
Новости
Производители
Контакты

Ближайшие фестивали

1 - 5 ноября
Москва МЕГА Белая Дача

7 - 9 декабря
Санкт-Петербург ТРЦ РИО

1 - 3 февраля
Курск ТРЦ Европа/Central Park

20 - 24 марта
Москва ТЦ уточняется

5 - 7 апреля
Санкт-Петербург ТРК Балканы Nova

Помощь

Активация дисконтной карты

Новости

23.10.2018 Регулярные мастер-классы в Рязани

10.10.2018 В Москве на Каширском шоссе открылся флагман «Леонардо»

15.10.2018 Билеты на фестивальные мастер-классы в МЕГЕ Белая Дача

15.10.2018 Теперь в московском районе Коньково есть свой большой «Леонардо»

ИЩЕМ МАСТЕРОВ

Главная > Интернет-магазин

Поиск по товарам

Разделы каталога, товары которых попали в поиск:

Товары для художников / Краски и контуры
Товары для художников / Краски и контуры / Пальчиковые краски
Товары для художников / Краски и контуры / Краски по ткани
Товары для художников / Краски и контуры / Краски в аэрозольных упаковках
Декорирование / Витраж / Краски по стеклу и керамике
Рукоделие и творчество / Батик / Краски для батика
Детское творчество / Материалы для творчества / Пальчиковые краски
Товары для художников / Краски и контуры / Краски по стеклу и керамике
Товары, найденные по запросу "краски":

Выводить по: 30

Краска акриловая "Love2Art" матовая АСР-60 60 мл 02 кофе с молоком	Краски акриловые "ТАИР" художественные "Деколор" 50 мл Белое серебро 512100	Краски масляные "Сонет" № 1 46 мл Белые цинковые 2604100
232 ⁰⁰ руб	229 ⁰⁰ руб	155 ⁰⁰ руб
Посмотреть	Посмотреть	Посмотреть

Отправьте нам сообщение

Анализ юзабилити сайта



Улучшение юзабилити посредством фокус-группы



Google. Analytics

Google Analytics

Reporting

Customization Admin

calmahome@gmail.com

http://read-able.com - http://read-able.... The Readability Test Tool

Day Week Month

Find reports & more

Pageviews vs. Select a metric

Pageviews

2,000 4,000

Mar 22 Mar 29 Apr 5

Primary Dimension: Page Page Title Other

Plot Rows Secondary dimension: Custom User ID Sort Type: Default advanced

Page	Custom User ID	Pageviews	Unique Pageviews	Avg. Time on Page	Entrances	Bounce Rate	% Exit
1. /index.html	abc-123-xyz	98 (4.23%)	25 (2.93%)	00:01:27	491	34.42%	19.0%
2. /index.html	def-789-uvw	97 (4.19%)	21 (2.46%)	00:01:24	21	4.28%	19.05%
3. /index.html	abc-789-rst	82 (3.54%)	18 (2.11%)	00:01:25	18	3.67%	38.89%
4. /index.html	abc-456-uvw	80 (3.46%)	24 (2.81%)	00:01:40	21	4.28%	33.33%
5. /index.html	ghi-123-xyz	73 (3.15%)	18 (2.11%)	00:01:09	15	3.05%	33.33%
6. /index.html	def-456-xyz	72 (3.11%)	15 (1.76%)	00:00:45	14	2.65%	14.29%

PII Viewer for Google Analytics 0.0.2 by @dvdmspn OFF ON

Display mapping: display_name

Сервис "Яндекс.Метрика"

Вебмастер Метрика Виджеты Рекламная сеть Директ Поиск для сайта API еще

Войти Помощь

Яндекс

Вход

пароль

запомнить меня

Войти

зарегистрироваться

Метрика

Баннер: Веб-аналитика для повышения эффективности вашего сайта

Яндекс.Метрика — бесплатный инструмент для повышения конверсии сайта. Наблюдайте за ключевыми показателями эффективности сайта, анализируйте поведение посетителей, оценивайте отдачу от рекламных кампаний.

Получить счётчик

Демо-доступ без регистрации

Обзор сервиса

График: Количество посетителей

Легенда:

- Прямые запросы
- Переходы с поисковых систем
- Переходы из ссылок на сайте
- Переходы из рекламы

Статистика: Трекеры, Визиты, Просмотры

Сводка

Ключевые параметры о жизни сайта на одном экране.

Актуальные данные

Современно узнавайте об изменениях посещаемости вашего сайта.

Цели

Фиксируйте совершение целевых действий и оценивайте конверсию.

Конструктор отчетов

Создавайте собственные информационные презентации по любым параметрам.

Новости

12 мая 2011 г.
Запущен API Яндекс.Метрики.

20 апреля 2011 г.
Яндекс.Метрика считает отказы по-новому.

21 апреля 2011 г.
Баннер в Яндекс.Метрике: смотрите, что делают посетители на сайте.

Архив новостей

Оценивайте конверсию по рекламе с помощью специальных меток.

Присоединяйтесь к клубу Яндекс.Метрики

Usability Hub

UsabilityHub

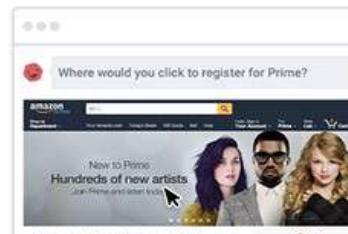
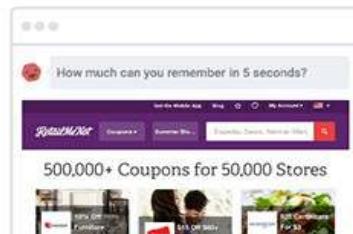
Tests PRO Plan Docs Customers Blog Register Get paid to test Sign In

Settle design debates with data

Remote user testing to help you make confident design decisions.

Get started now

The UsabilityHub suite of tests.



Usability Hub Usambilla



Optimal Workshop



Ручные способы анализа юзабилити сайта



Юзабилити структуры проекта и его навигации



Анализ сайта

Репрезентативная задача 1: Заказ детализации на период с 14 октября по 21 октября. С сайта мегафон, личный кабинет.

Личный кабинет

361,94 ₽

+7 000 Мерзлякова Екатерин...

Счёт

Доступно на сегодня
361,94 ₽

Бонусные баллы
0 ⚡

Мой номер
+7 000

Пополнить счёт	Обещанный платёж	Расходы, пополнения и детализация
		✓
Перевести деньги	Управление автоплатежами	Потратить бонусы
Виртуальная карта «Счёт телефона»	Персональные предложения	
	1	

Анализ сайта

ГЛАВНАЯ СТРАНИЦА → РАСХОДЫ, ПОПОЛНЕНИЯ И ДЕТАЛИЗАЦИЯ → ЗАКАЗ ДЕТАЛИЗАЦИИ



Детализация - это подробный отчёт о стоимости звонков, сообщений, интернет-сессий и других услуг связи. Закажите детализацию, чтобы посмотреть, как расходовались деньги на вашем номере за интересующий период времени.

Начало периода

Конец периода

Формат отчёта

Электронная почта

Заказать

Анализ сайта

 Личный кабинет

 361,94 ₽

 17 ос
Мерзлякова Екатерин...

ГЛАВНАЯ СТРАНИЦА → РАСХОДЫ, ПОПОЛНЕНИЯ И ДЕТАЛИЗАЦИЯ → ЗАКАЗ ДЕТАЛИЗАЦИИ → ЗАКАЗАТЬ ДЕТАЛИЗАЦИЮ РАСХОДОВ?

Заказать детализацию расходов?

Стоимость услуги: 12 ₽

Адрес доставки: Kate 129, г. Краснодар

Период детализации: 14 октября 2018 г. – 21 октября 2018 г.

Формат детализации: HTML

Заказать

Отмена

Анализ сайта

Заказ детализации

Операция успешно завершена



Личный кабинет



Детализация - это подробный отчёт о стоимости звонков, сообщений, интернет-сессий и других услуг связи. Закажите детализацию, чтобы посмотреть, как расходовались деньги на вашем номере за интересующий период времени.

Начало периода

Конец периода

Формат отчёта

Электронная почта

Заказать

Анализ сайта

 Личный кабинет

 361,94 ₽

 +7 923 245 5562 
Мерзлякова Екатерин...

ГЛАВНАЯ СТРАНИЦА → РАСХОДЫ, ПОПОЛНЕНИЯ И ДЕТАЛИЗАЦИЯ → ЗАКАЗ ДЕТАЛИЗАЦИИ

Заказ детализации

Детализация - это подробный отчёт о стоимости звонков, сообщений, интернет-сессий и других услуг связи. Закажите детализацию, чтобы посмотреть, как расходовались деньги на вашем номере за интересующий период времени.

Начало периода

Конец периода

Формат отчёта

Электронная почта

 Заказать

Анализ сайта

Репрезентативная задача 2: Посмотреть информацию по тарифам.

 **Личный кабинет**  361,94 ₽ 

Счёт

Доступно на сегодня
361,94 ₽

Бонусные баллы
0 

Мой номер


Пополнить счёт 	Обещанный платёж 	Расходы, пополнения и детализация 
Перевести деньги 	Управление автоплатежами 	Потратить бонусы 0 
Виртуальная карта «Счёт телефона» 	Персональные предложения 1 	

Анализ сайта

Личный кабинет

361,94 ₽

Мерзлякова Екатерин...

361,94 ₽



Бонусные баллы

0 ⚡

Мой номер



Перевести
деньги



Управление
автоплатежами



Потратить бонусы

0



Виртуальная карта
«Счёт телефона»



Персональные предложения

1



Услуги

Остатки по
пакетам услуг

Тариф

Услуги и опции

Анализ сайта

 Личный кабинет

 827,04 ₽

 +7 929 ...
Мерзлякова Екатерин...

ГЛАВНАЯ СТРАНИЦА → СПИСОК ТАРИФОВ → ВСЁ ПРОСТО 16

Тариф

Всё просто 16

Однаковая стоимость звонков, SMS и интернета дома и в поездках по России

[Сменить тариф](#)

[Настроить опции](#)

Абонентская плата

0 ₽

Звонки на все местные номера

1.2 ₽ за минуту

[Подробнее](#)

Анализ сайта



(₽) 827,

- ✓ Услуга подключена 27 апреля 2017 г. в 08:07

Управляйте входящими звонками, не прерывая текущий разговор.
Стоимость удержания вызова в подробном описании.

[Подробное описание >](#)

0 ₽

[Отключить](#)

С абонентской платой

Название услуги и описание

Абон. плата:

Интернет XS

- ✓ Услуга подключена 6 марта 2018 г. в 00:06

Опция для работы с e-mail, общения в социальных сетях и просмотра страниц.

[Подробное описание >](#)

Сначала 150 ₽ за
30 дней,
Потом 5 ₽ в день

[Отключить](#)

Анализ сайта



Объём трафика

/0 Мб в сутки

150 ₽ в месяц

Абонентская плата в первый месяц

[Подробнее](#)

Абонентская плата со второго месяца

5 ₽ в сутки

Территория действия

Домашний регион

Устройства, на которых работает опция

Смартфон

[Подключить](#)

Поиск



Личный кабинет



Как проверить остатки
пакетов?

Проверить остатки пакетов
трафика можно в [Личном кабинете](#)
или в приложении [Мегафон](#)

Анализ сайта

Частным клиентам Корпоративным клиентам Инвесторам и прессе Интернет-магазин 

Новосибирская ... ▾

Ваш регион — Новосибирская область? Да [Изменить регион](#)

 МЕГАФОН

Поиск 

 Личный кабинет

Тарифы **Интернет** Услуги и опции Роуминг Акции Оплата Устройства Поддержка

ВСЕ ИНТЕРНЕТ-ОПЦИИ → ИНТЕРНЕТ XS

Интернет XS

Объём трафика

70 МБ в сутки

Как проверить остатки пакетов?

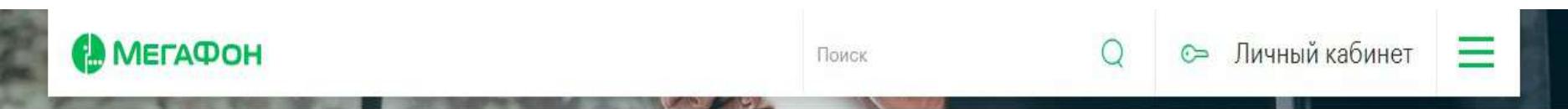
Абонентская плата в первый месяц

150 ₽ в месяц

Проверить остатки пакетов трафика можно в [Личном кабинете](#) или в приложении Мегафон

[Подробнее](#)

Анализ сайта



Тарифы «Включайся!»

Все возможности для смартфонов

Тарифы «Тёплый приём»

Для выгодных звонков в страны СНГ

Другие тарифы

Для общения с абонентами МегаФона

Переходи на НОЛЬ

Безлимитные звонки на местные номера
МегаФона

Без абонентской платы

[Купить SIM-карту](#)

[Перейти на тариф
абонентам МегаФона](#)

Специальные

*Заинтересовал тариф «Посекундный», выбираю для просмотра.

Анализ сайта



ТАРИФЫ → ДРУГИЕ ТАРИФЫ → ПОСЕКУНДНЫЙ

Поиск



Личный кабинет



Посекундный

Единая стоимость на номера любых операторов и поsekундная оплата

Абонентская плата

Звонки на номера Домашнего региона

2,9 ₽ за минуту

[Подробнее](#)



Включайся! Пиши

Безлимитные мессенджеры и пакет сообщений для долгих переписок.

Подробнее о тарифе

Междугородные звонки

Как узнать свой тариф?

Наберите бесплатную команду

*105*3#

Анализ сайта



Поиск



Личный кабинет



Тарифы «Включайся!»

Все возможности для смартфонов

Тарифы «Тёплый приём»

Для выгодных звонков в страны СНГ

Другие тарифы

Я звоню:



Редко | Часто

Я использую:



Мессенджеры | Соцсети и мессенджеры

Я смотрю и слушаю:



Музыку | Видео

Включайся!
Общайся
Акция

[Подробнее о тарифе](#)

1000 минут

Безлимитный интернет

на всё, что хочешь

Бесплатно



ЛитРес

Одна книга каждый месяц и скидка 20% на первую покупку

450 ₽ в месяц

[Купить SIM-карту](#)

Только для новых
абонентов

Анализ сайта



Поиск



Личный кабинет



Я звоню:



Редко | Часто

Я использую:



Мессенджеры | Соцсети и мессенджеры

Я смотрю и слушаю:



Музыку | Видео

Включайся! Слушай

АКЦИЯ

Бесплатная подписка
на BOOM

[Подробнее о тарифе](#)

600 минут
12 ГБ

Неограниченный трафик
на музыку и мессенджеры



Бесплатно

BOOM
Музыка в ВК без ограничений

ЛитРес

Одна книга каждый месяц и скидка 20% на первую покупку

300 ₽ в месяц

[Купить SIM-карту](#)

[Перейти на тариф
абонентам Мегафона](#)

Включайся!

1000 минут

Безлимитный интернет

450 ₽ в месяц

Анализ сайта

The screenshot shows the top navigation bar of the Megafon website. It features the Megafon logo, a search bar with a magnifying glass icon, a 'Личный кабинет' (Personal Cabinet) button with a key icon, and a menu icon represented by three horizontal lines. Below the header, there's a large green promotional banner. On the left of the banner, it says '300₽ в месяц' and has a purple button labeled 'Купить SIM-карту'. To the right, there's a link 'Перейти на тариф абонентам Мегафона' and a circular image showing a person holding a purple balloon.

В абонентскую плату включено

Мобильный интернет

12 ГБ

Интернет на мессенджеры

WhatsApp, Viber, Facebook Messenger, Snapchat,
eMotion, ТамТам

Интернет на музыку

VOOM, Яндекс.Музыка, Звук, ВКонтакте Музыка, Apple
Music, Radio Record и др. радиостанции

Безлимитно



Безлимитные мессенджеры
на тарифах Включайся!

Продолжайте общаться в поездках
по миру всего за 99 ₽ в сутки.

Подключайте опцию Соцсети

Пользуйтесь популярными

Человеко-машинное взаимодействие

Лекция 3

Мерзлякова Екатерина Юрьевна
к.т.н. доцент ПМИК

Проблемно-центрированная разработка интерфейса

- анализ задач и пользователей;
- выбор репрезентативных задач;
- заимствование;
- черновое описание дизайна;
- обдумывание дизайна;
- создание макета или прототипа;
- тестирование дизайна с пользователями;
- итерирование;
- реализация;
- отслеживание эксплуатации;
- изменение дизайна.

Анализ задач и пользователе



Кто и зачем собирается использовать разрабатываемую систему?

- уровень знаний пользователя
- область деятельности
- общие характеристики

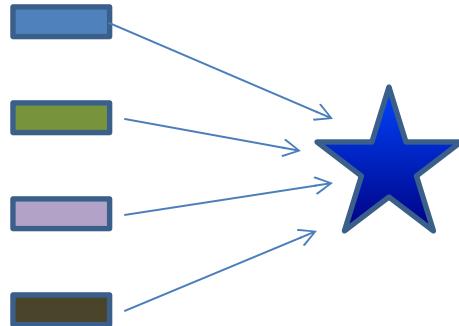
- Система должна запрашивать от пользователя информацию в порядке, который кажется ему естественным,
- Система должна давать возможность простой коррекции ошибок при вводе данных,
- Выбранные для организации интерфейса аппаратные средства должны вписываться в рабочую среду пользователя и быть эргономичными для его действий.

Выбор репрезентативных задач



- задачи, которые пользователи описали разработчикам
- реальные задачи, с которыми сталкиваются пользователи
- задачи должны достаточно полно покрывать всю функциональность системы
- смесь простых и более сложных задач

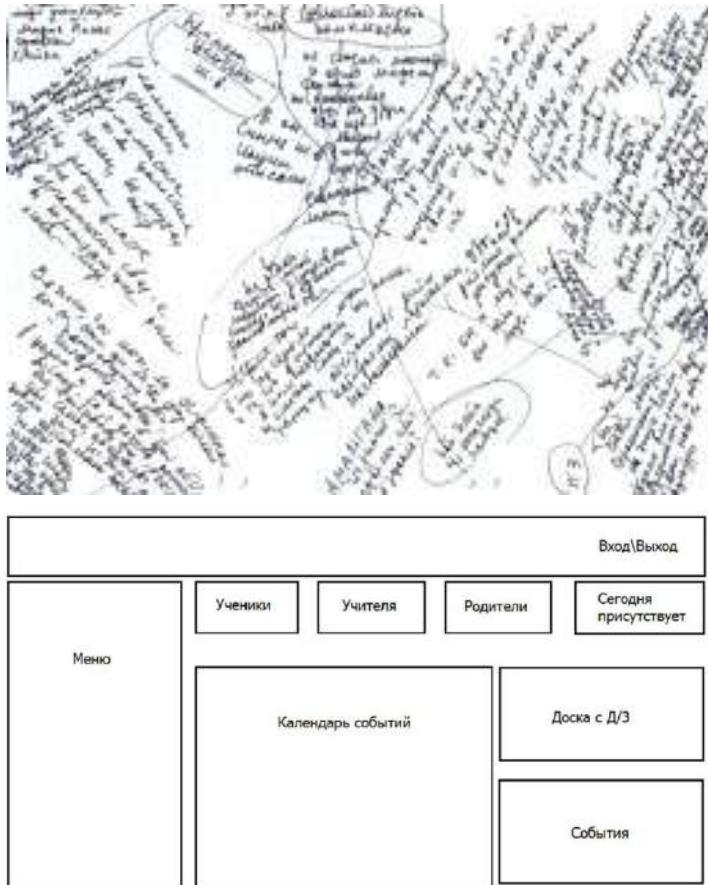
Заимствование



найти существующие интерфейсы, с помощью которых пользователи могут выполнить требуемую работу, и затем строить идеи новой системы на их базе

- Какие программы ваши пользователи используют сейчас?
- Найти существующие интерфейсы, с помощью которых пользователи могут выполнить требуемую работу, и затем строить идеи новой системы на их базе
- Чаще всего наилучшим вариантом будет придерживаться правилам старой системы

Черновое описание дизайн



Черновое (грубое) описание разрабатываемой вами системы должно быть положено на бумагу (обязательно). Это позволяет задуматься о многих вещах. Но это описание не следует оформлять в виде компьютерной программы (пока), даже если вы умеете пользоваться какими-либо системами автоматизации разработки. Такие системы вынуждают вас прикрепляться к конкретным решениям, которые ещё слишком рано делать.

Обдумывание дизайн



- ✓ Стоимость построения законченного пользовательского интерфейса и его тестирование с достаточным количеством пользователей для выявления всех проблем очень высока.
- ✓ Существует несколько структурных подходов, которые можно использовать, чтобы исследовать сильные и слабые стороны интерфейса до его программного воплощения.

CWT - позволяет находить места в дизайне, где пользователь может делать ошибки.

GOMS - оценка трудоёмкости выполнения задач по времени и выявление задач, требующих слишком много шагов.

Создание макета или прототипа



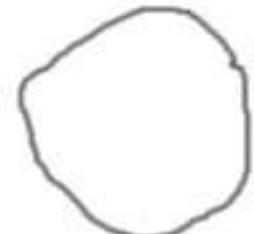
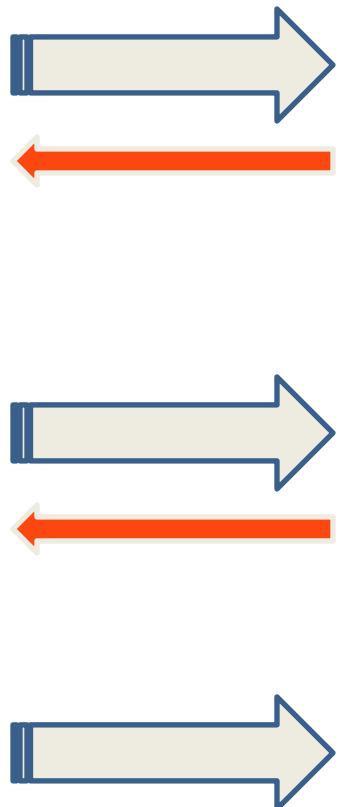
Тестирование дизайна пользователями



«думать вслух»



Итерирование



Тестирование
проанализировать
результаты тестирования,
соизмеряя **стоимость**
корректировок с
серьёзностью возникших
проблем, затем доработать
интерфейс и протестировать
его снова.



цель тестирования состоит не
в том, чтобы доказать
правильность интерфейса, а в
том, чтобы улучшить его

Построение интерфейса систем



пользовательский интерфейс часто занимает более половины кода коммерческого продукта

Отслеживание дизайна систем



Программный
продукт

команда разработчиков не должна быть изолирована от всей остальной деятельности, связанной с функционированием системы

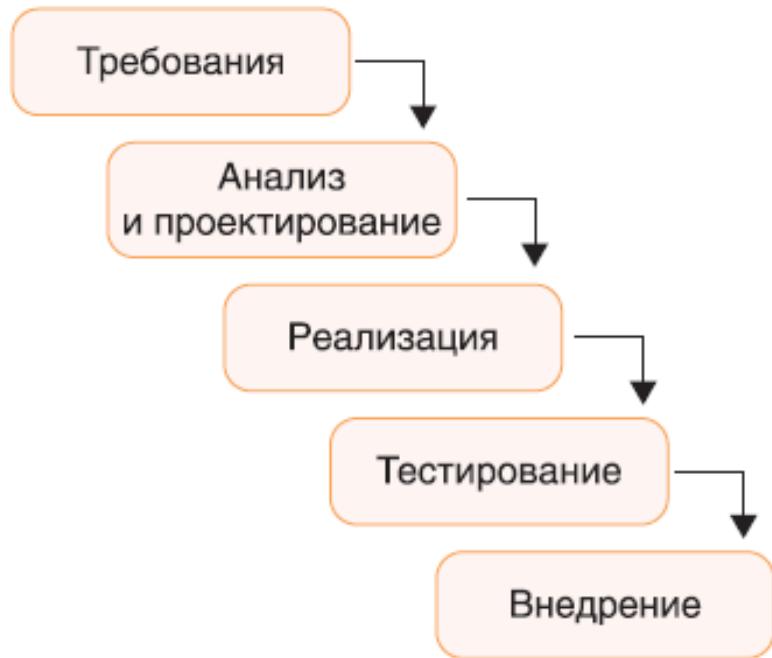
Изменение дизайна



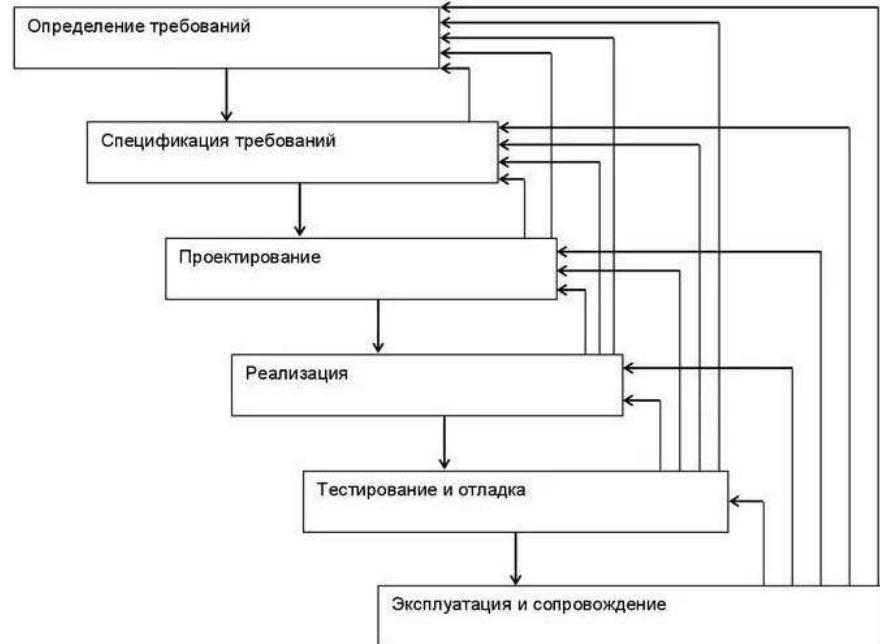
Независимо от того, насколько удачно система была спроектирована первоначально, с большой вероятностью она будет **терять адекватность** с течением лет. Меняются и задачи и пользователи. **Приёмы работы** меняются из-за нового оборудования и программных продуктов. Пользователи приобретают **новые навыки** и ожидаемые реакции. Разработчики должны стоять вровень с этими изменениями, не только отслеживая состояние той **рабочей среды**, для которой была предназначена их система, но и **развитие** всего общества, технологий и методов, потребностей.

Управление процессом разработки

Метод водопада



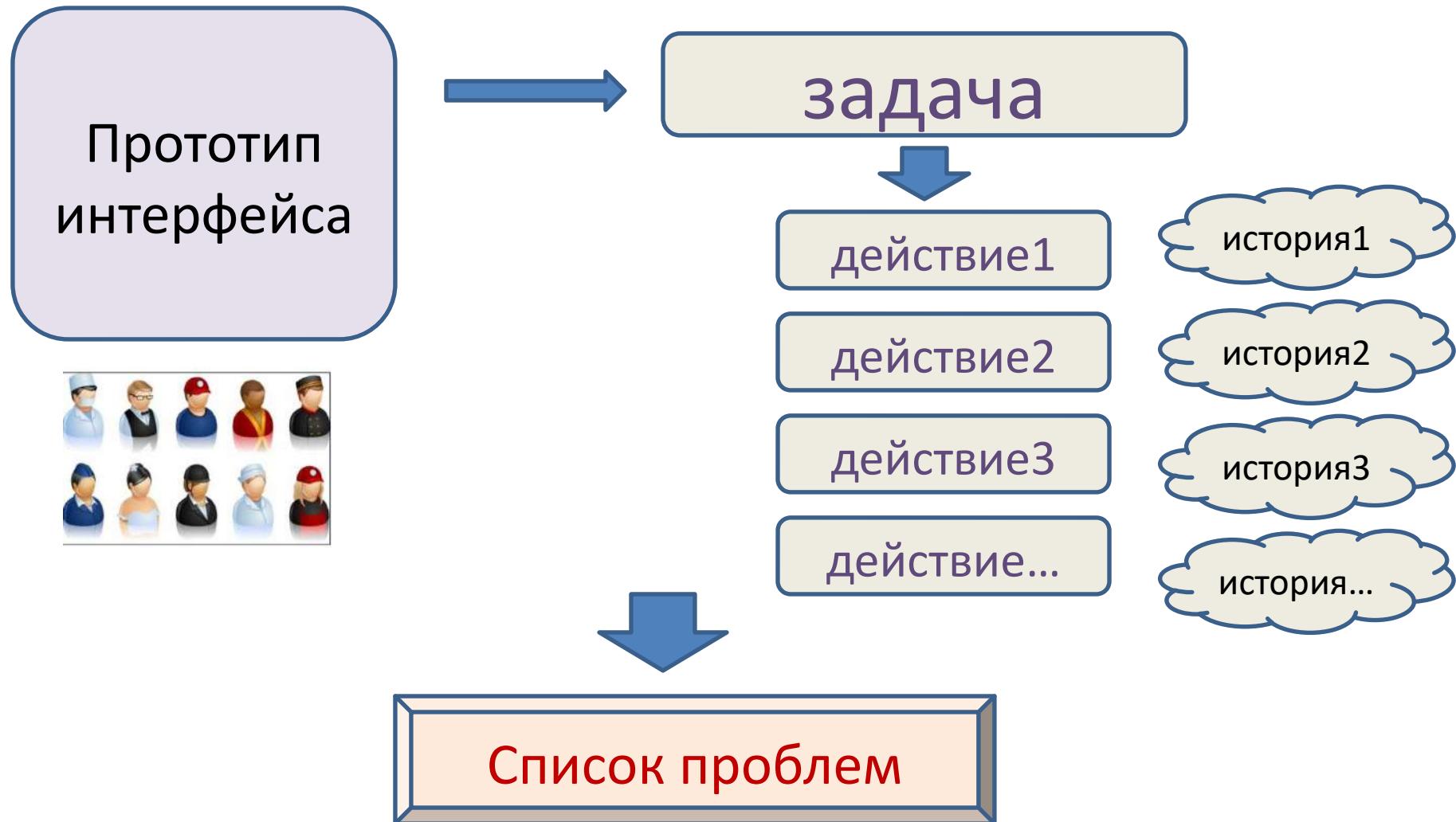
Проблемно-центрированный подход



Требования practicalnost

- ✓ Требования practicalности – это целевые значения для таких характеристик, как скорость выполнения репрезентативных задач и допустимое количество ошибок.
- ✓ Эти показатели могут использоваться, чтобы мотивировать разработчиков и обосновывать решения по распределению ресурсов.
- ✓ Целевые значения могут быть выбраны так, чтобы побить конкурентов или обеспечить функциональные нужды для хорошо определённых задач.

CWT-анализ интерфейс



CWT-анализ интерфейс

 CDCOPY V4.950 - © by Markus Barth '97-'00 mbarth2193@aol.com - □ X

File Edit Function Help

CD-ID : ec111610

ASPI
HL-DT-STRW/DVD GCC-4320B1.00 1:0:0

#	Time	Filesize	Track name
1	5:05:73	51 MB	Track 1
2	3:49:63	38 MB	Track 2
3	3:18:64	33 MB	Track 3
4	4:15:49	43 MB	Track 4
5	7:02:00	70 MB	Track 5
6	3:29:47	35 MB	Track 6
7	1:31:71	15 MB	Track 7
8	4:17:04	43 MB	Track 8
9	6:54:20	69 MB	Track 9
10	1:57:14	19 MB	Track 10
11	2:21:27	23 MB	Track 11
12	7:27:00	75 MB	Track 12

Space
Free : 1538 MB
Selected: 33 MB

Alert at end

WAV



Progress

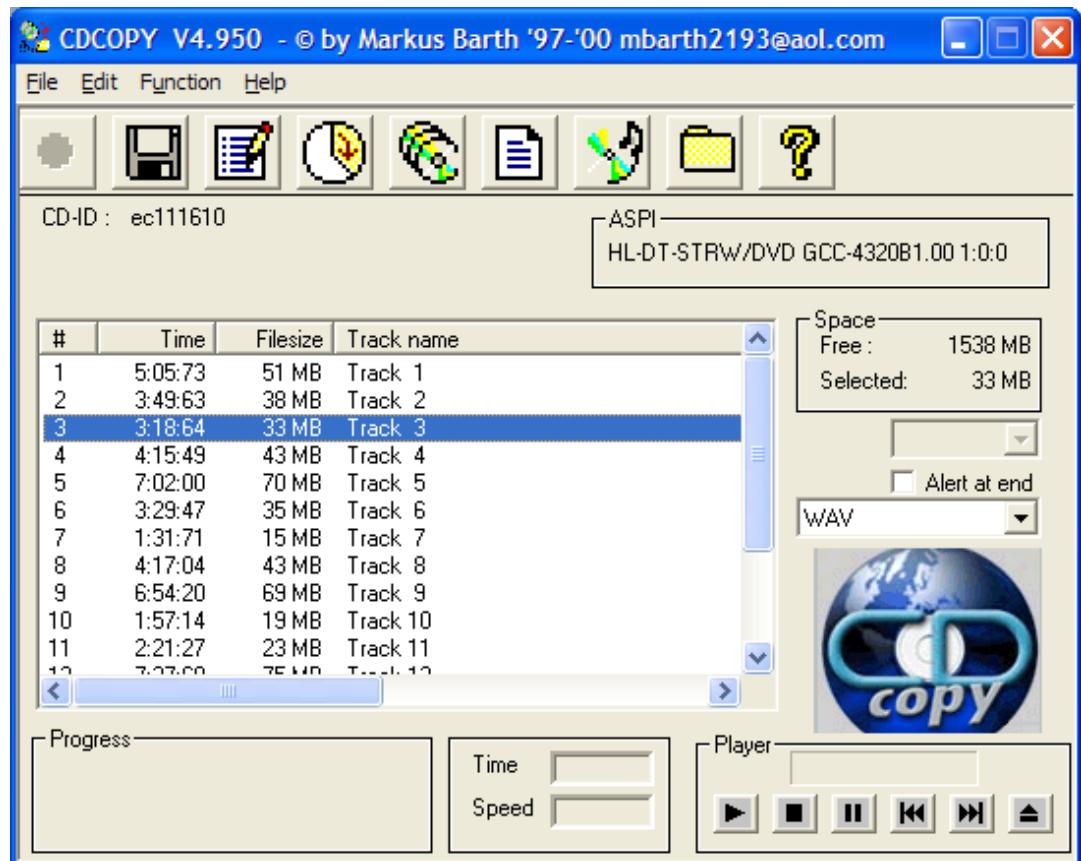
Time
Speed

Player



CWT-анализ интерфейс

1. Загрузить аудио диск в устройство чтения компакт-дисков компьютера;
2. Запустить программу CDCopy;
3. В появившемся списке треков выбрать трек № 3;
4. В списке форматов файлов выбрать MP3(MPEG 1 Lay. 3);
5. Нажать кнопку "Start copying".



Что дает SWT-анализ интерфейс

- Он может поставить под сомнение ваши первоначальные и не вполне обоснованные предположения о том, как мыслит пользователь (вначале поставить диск, а потом запустить программу, или наоборот? кнопка с изображением дискеты означает копирование или сохранение?).
- Он может выявлять элементы управления, которые очевидны для разработчика, но могут быть скрыты от пользователя (список форматов выходного файла).
- Он может выявлять затруднения с надписями и подсказками (неудачное предупреждение "No media present").
- Он может обнаруживать неадекватную обратную связь, что может заставить пользователя сомневаться в результате и повторять всё с начала, хотя всё было сделано правильно (отсутствие индикации пути, по которому сохраняется файл).
- Он может показывать недостатки в текущем описании интерфейса (слова "Start copying" вместо графического изображения кнопки в руководстве пользователя).



Рекомендации при выполнении СУТ-анализа

- ✓ Будут ли пользователи пытаться произвести тот или иной эффект, который даёт действие?
- ✓ Видят ли пользователи элемент управления (кнопку, меню, переключатель и т.д.) для осуществления действия?
- ✓ Если пользователи нашли элемент управления, поймут ли они, что он производит тот эффект, который им нужен?
- ✓ После того как действие сделано, будет ли понятен пользователям тот отклик, который они получают, чтобы перейти к следующему действию с уверенностью?

Операции в GOMS – это элементарные действия, которые нельзя разложить на более мелкие.

Шаг метода: нажатие на кнопку

- визуально определить местонахождение кнопки (мыслительная операция);
- навести на кнопку указатель мыши (внешняя операция);
- щелкнуть кнопкой мыши (внешняя операция).

K – нажатие клавиши;

B – клик кнопкой мыши;

P – наведение указателя мыши;

R – ожидание ответной реакции компьютера;

H – перенос руки с клавиатуры на мышь или наоборот;

D – проведение с помощью мыши прямой линии (например, выделение или прокрутка текста);

M – мыслительная подготовка (к осуществлению одной из перечисленных операций).

K 0.2 с

B 0.2 с

P 1.1 с

R 0.25 с (или больше)

H 0.4 с

D 2 с (или больше)

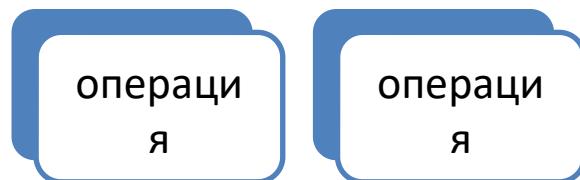
M 1.35 с



(репрезентативная задача)



(список действий)



(элементарные действия пользователя)

Пример: Возьмём в качестве исследуемой программы Microsoft Word 2003. Пусть поставлена **цель**: напечатать (вставить в текст) уравнение $x^2 + x - 1 = 0$ с помощью редактора формул (перед печатью уравнения набирался текст).

Для выполнения цели сформулируем три
подцели:

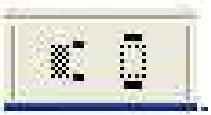
1. Вызвать редактор формул через меню "Вставка | Объект... | Microsoft Equation 3.0".
2. Напечатать формулу в редакторе.
3. Выйти из редактора и подготовиться к продолжению набора текста.

Теперь опишем методы для каждой подцели:

1. Вызвать редактор формул через меню "Вставка | Объект... | Microsoft Equation 3.0".
 - 1.1. Войти в меню "Вставка"
 - 1.2. Выбрать пункт "Объект..."
 - 1.3. Выбрать объект "Microsoft Equation 3.0" путём скроллинга списка типов объектов.

2. Напечатать формулу в редакторе.

2.1. Напечатать символ x .

2.2. В окне "Формула" выбрать объект 

2.3. В выпадающем меню объектов выбрать 

2.4. Напечатать символ 2 .

2.5. Нажать клавишу \rightarrow .

2.6. Напечатать " $+x - 1 = 0$ ".

3. Выйти из редактора и подготовиться к продолжению набора текста.

3.1. Нажать клавишу Esc.

3.2. Нажать клавишу End (иногда при выходе из редактора формула остаётся выделенной, поэтому необходимо снять выделение).

Теперь распишем каждый метод с точностью до операции (повторим для наглядности название подцелей и методов):

1. Вызвать редактор формул через меню "Вставка | Объект... | Microsoft Equation 3.0".

1.1. Войти в меню "Вставка"

H (переместить руку на мышь, т.к. мы набирали текст)

P (переместить указатель мыши)

B (клик мыши)

1.2. Выбрать пункт "Объект..."

PB

1.3. Выбрать объект "Microsoft Equation 3.0" путём скроллинга списка типов объектов.

PB (перемещение указателя и фиксация мыши на элементе управления скроллинга)

D(3.0) (скроллинг вниз и поиск нужной строки, экспериментальная оценка времени)

PBV (установка указателя и двойной щелчок)

R(0.8) (ожидание запуска редактора формул)

2. Напечатать формулу в редакторе.

2.1. Напечатать символ x.

H (перемещение руки на клавиатуру)

K (печать x)

2.2. В окне "Формула" выбрать объект



H (перемещение руки на мышь)

PB

2.3. В выпадающем меню объектов выбрать

PB



2.4. Напечатать символ 2.

H

K

2.5. Нажать клавишу →.

K

2.6. Напечатать " $+x - 1 = 0$ ".

7K (нажать Shift, печатать + (отпустить Shift), печатать оставшиеся символы)

3. Выйти из редактора и подготовиться к продолжению набора текста.

3.1. Нажать клавишу Esc.

K

R(0.8) (ожидание выхода из редактора формул и переход в текстовый режим)

3.2. Нажать клавишу End.

K

$HPBPBPBD(3.0)PBBR(0.8)HKHPBPBHKK7KKR(0.8)K$

$MHPBPBPBD(3.0)PBBR(0.8)$ МНК $MHPBPB$ МНК MK $M7K$ $MKR(0.8)$ MK

$$8M = 10.8, 4H = 1.6, 6P = 6.6, 7B = 1.4, D(3.0) = 3.0, 2R(0.8) = 1.6, 12K = 2.4.$$

Общий итог: 27.4 с.

Пример: Возьмём в качестве исследуемой программы текстовый редактор. Пусть поставлена **цель**: напечатать (вставить в текст) уравнение $x^2 + x - 1 = 0$ с помощью редактора формул (перед печатью уравнения набирался текст).

8М, 10Н, 3Р, 6В, 20К.

Оценка среднего времени решения задачи составляет 23.3 с.

MHPBPBPBD(3.0)PBBR(0.8) 10.95 c.

MHPBR(0.8) 3.85 c.

Время нажатия на клавишу:

К = 0.2 с (естественной последовательности)

К = 0.5 с (случайной последовательности)

К = 0.75 с (сложные коды)

извлечения простого элемента знания из долговременной памяти
1.2 с

извлечения простого элемента знания из краткосрочной памяти
0.6 с

Золотые правила построения интерфейсов

Правила Нильсена-Молиха (Nielsen, Molich)

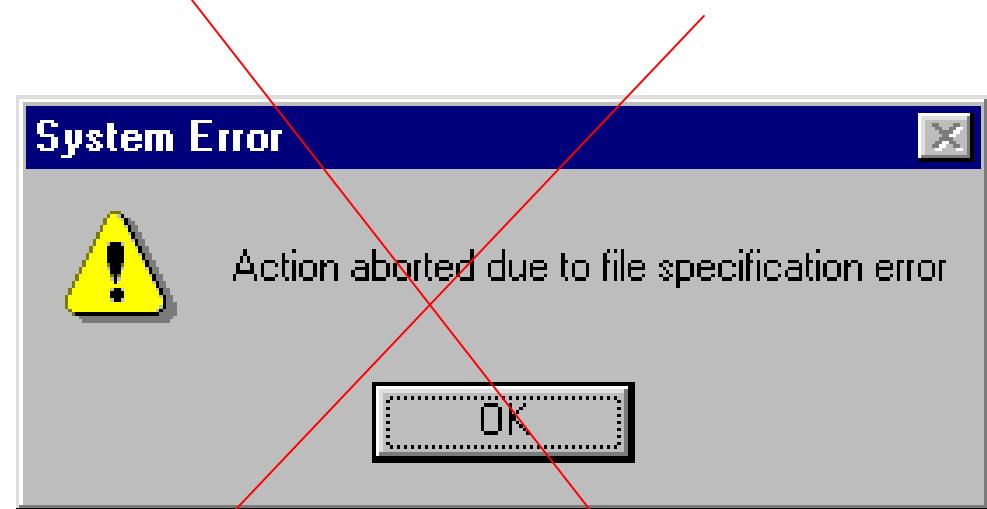
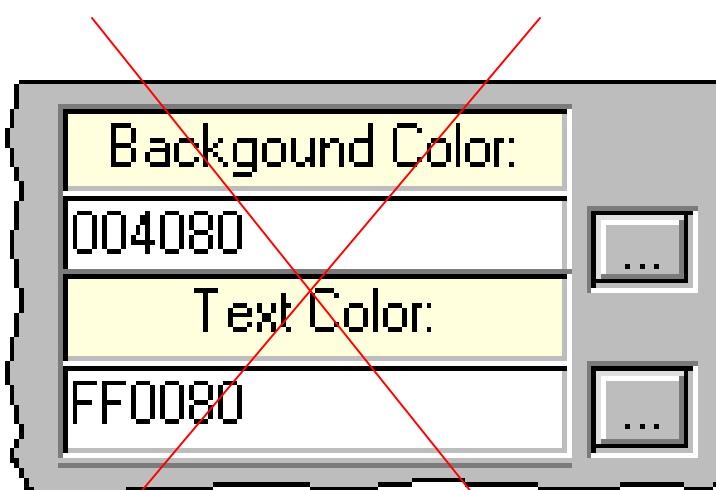
1. Простой и естественный диалог:

- не должно присутствовать не относящейся к теме или редко используемой информации ;
- "лучше меньше да лучше";
- информация, которая выводится на экран, должна появляться в порядке, соответствующем ожиданиям пользователя

Правила Нильсена-Молиха (Nielsen, Molich)

2. Говорите на языке пользователя:

- используйте слова и понятия из мира пользователя;
- не используйте специфических инженерных терминов.

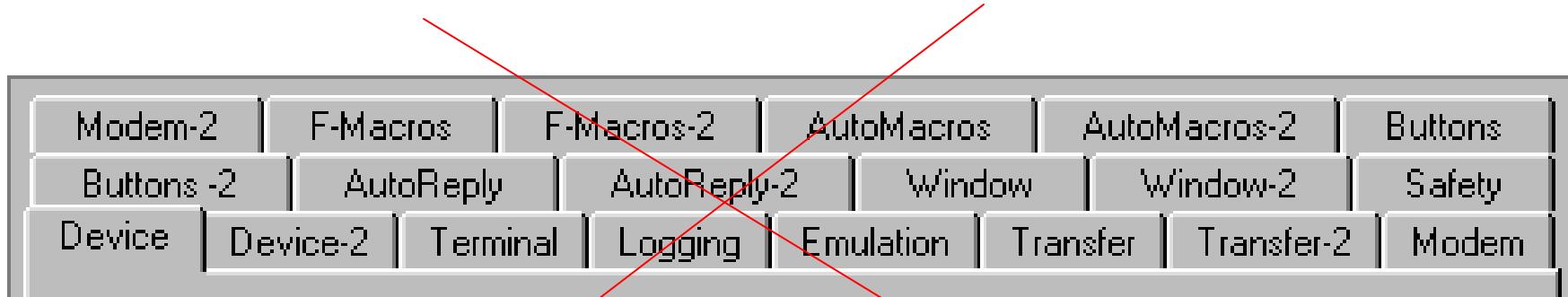


Правила Нильсена-Молиха

(Nielsen, Molich)

3. Минимизируйте загрузку памяти пользователя:

- не заставляйте пользователя помнить вещи от одного действия к следующему;
- оставляйте информацию на экране до тех пор, пока она не перестанет быть нужной;
- хорошим стилем считается делать только один ряд закладок.



Правила Нильсена-Молиха (Nielsen, Molich)

4. Будьте последовательны:

- у пользователей должна быть возможность изучить действия в одной части системы и применить их снова, чтобы получить похожие результаты в других местах.

5. Обеспечьте обратную связь:

- дайте пользователю возможность видеть, какой эффект оказывают его действия на систему.

Правила Нильсена-Молиха (Nielsen, Molich)

6. Обеспечьте хорошо обозначенные выходы:

- Если пользователь попадает в часть системы, которая его не интересует, у него всегда должна быть возможность быстро выйти оттуда, ничего не повредив.

7. Обеспечьте быстрые клавиши и ярлыки:

- элементы быстрого доступа могут помочь опытным пользователям избегать длинных диалогов и информационных сообщений, которые им не нужны.

Правила Нильсена-Молиха (Nielsen, Molich)

6. Хорошие сообщения об ошибках.

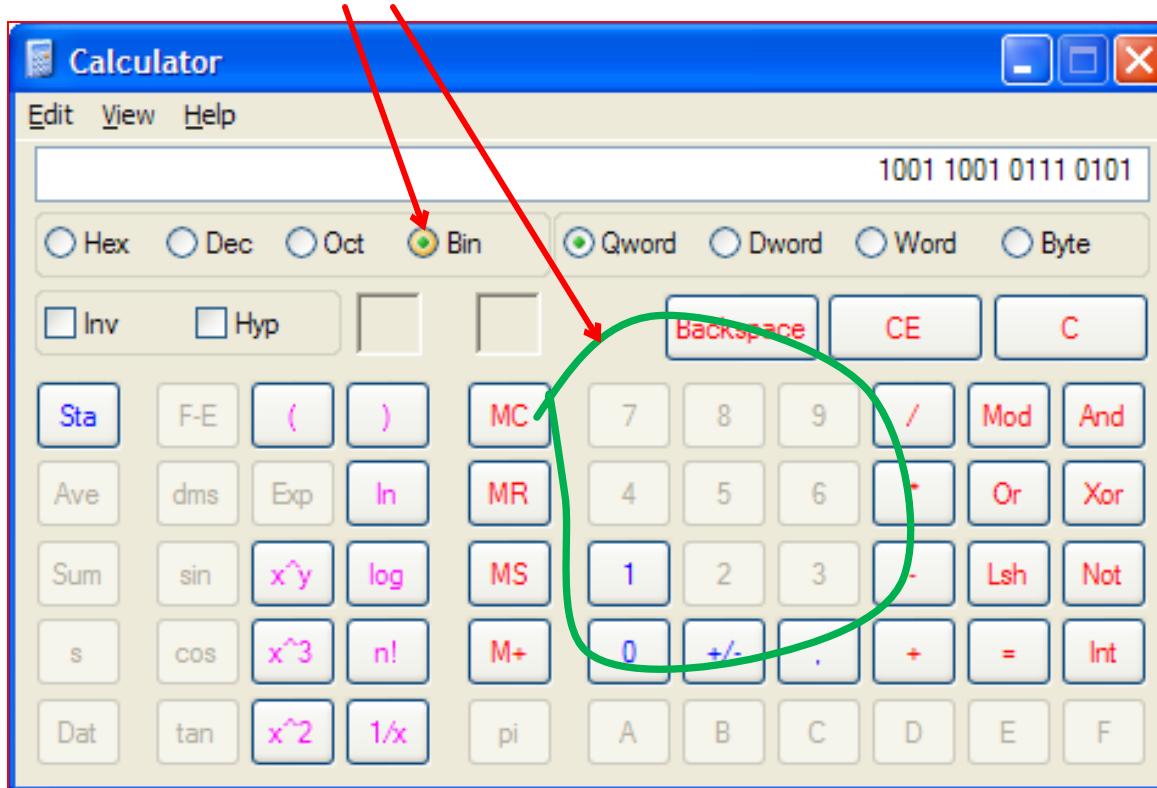
- Хорошее сообщение об ошибке помогает пользователю понять, в чём проблема и как это исправить.



Правила Нильсена-Молиха (Nielsen, Molich)

6. Предотвращайте ошибки

Всегда, когда вы пишете сообщение об ошибке, вы должны спросить себя, можно ли избежать этой ошибки?



Золотые правила построения интерфейсов

Принципы организации графического интерфейса

1. Принцип кластеризации:

- организуйте экран в виде визуально разделённых блоков с похожими элементами управления, предпочтительно с названием для каждого блока.;
- подобные команды должны быть в одном меню: это позволяет им быть визуально близко и идти под одним заголовком;
- команды, относящиеся к некоторой конкретной области функциональности, могут также быть показаны в диалоговых боксах, опять таки в визуально определимых блоках.

Принципы организации графического интерфейса

2. Принцип "видимость отражает полезность" :

- делайте часто используемые элементы управления заметными, видимыми и легко доступными;
- прячьте или сжимайте редко используемые элементы;

Принципы организации графического интерфейса

3. Принцип интеллектуальной последовательности:

- используйте похожие экраны для похожих функций.;
- экраны не должны выглядеть одинаково, если в действительности они должны отражать совершенно другие вещи;
- предупреждение о критической ошибке в системе реального времени должно иметь вид, значительно отличающийся от экрана помощи или информационного сообщения.

Принципы организации графического интерфейса

4. Принцип "цвет как приложение":

- не полагайтесь на цвет как носитель информации;
- используйте цвет умеренно, чтобы лишь акцентировать информацию, передаваемую другими средствами;
- обязательно дайте дополнительные ключи для пользователей, не способных воспринимать изменения цвета;
- помните, что многие пользователи могут, и часто это делают, изменить цвет окон, подсветок и других системных объектов. Стройте ваш продукт так, чтобы он работал с пользователем, а не боролся с ним.

Принципы организации графического интерфейса

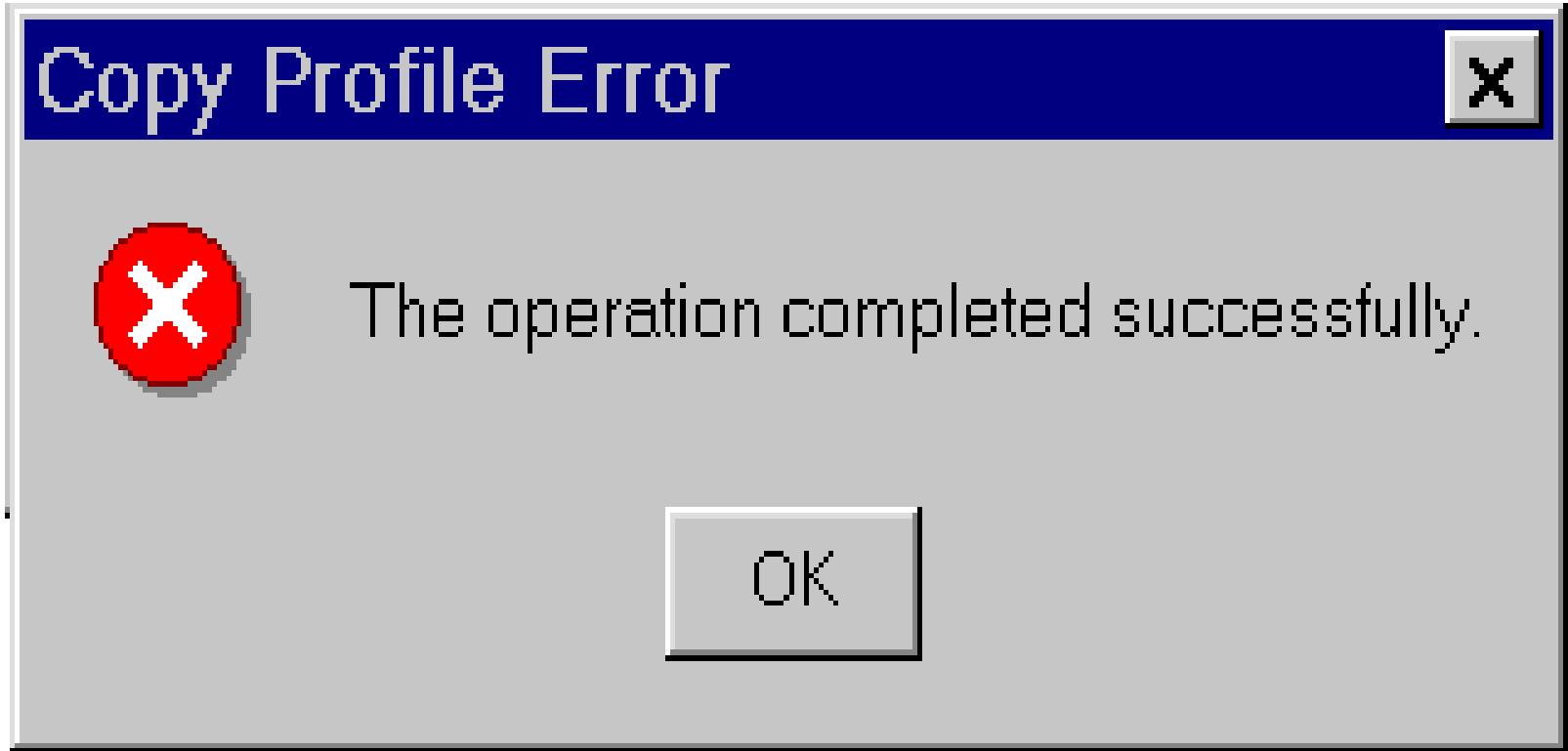
5. Принцип уменьшения беспорядка:

- не помещайте на экран слишком много всего;
- не пытайтесь наделить каждое меню собственным шрифтом или работать с большим набором размеров. Как правило, пользователи заметят не столько различия, сколько беспорядок.

Как НЕ НАДО делать



Как НЕ НАДО делать



Как НЕ НАДО делать

Selected files:

C:\GRAPHICS\GIFCONV\FORM\base9

C:\GRAPHICS\GIFCONV\FORM\base9

C:\GRAPHICS\GIFCONV\FORM\base9

C:\GRAPHICS\GIFCONV\FORM\base9

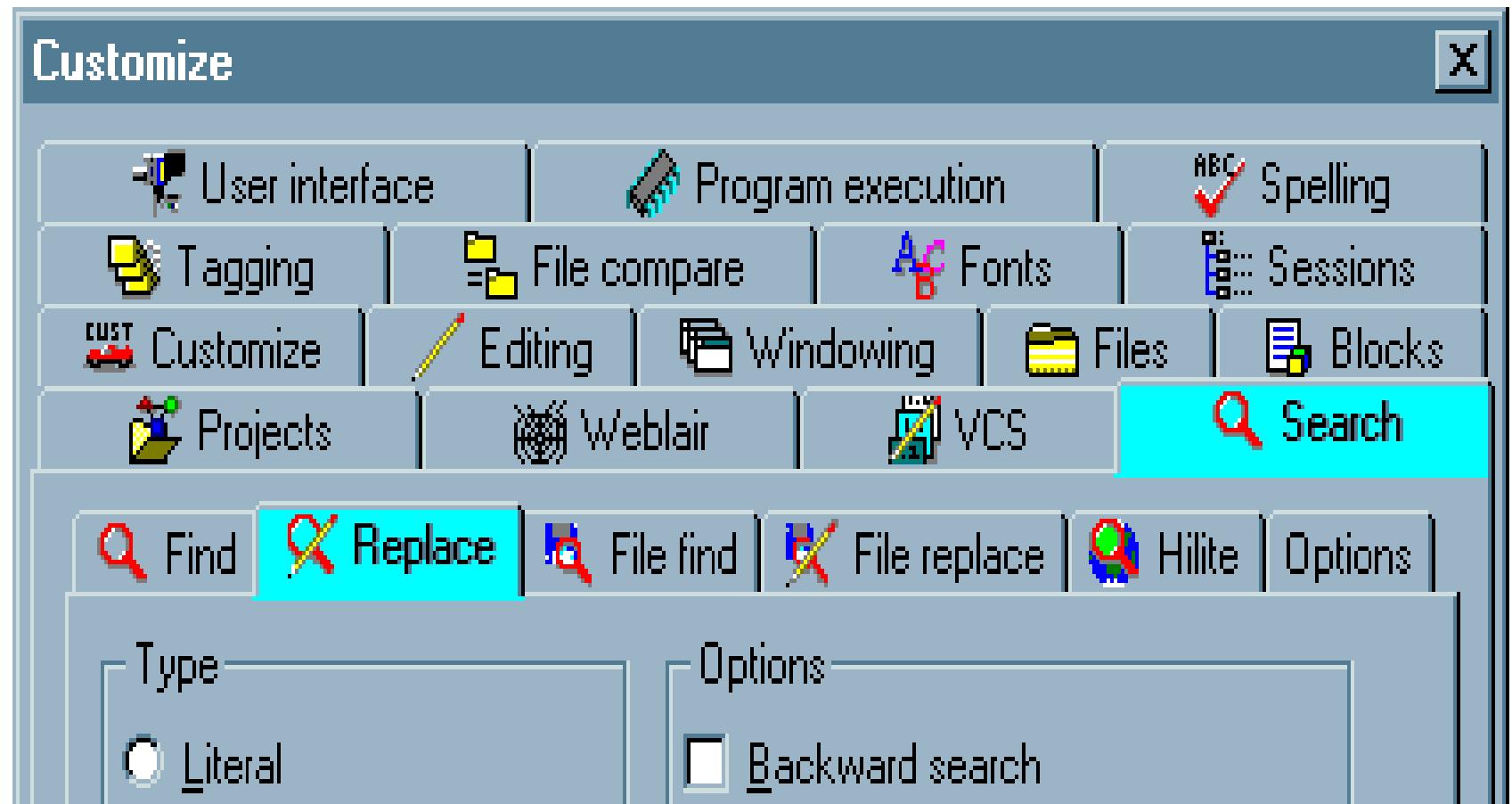
C:\GRAPHICS\GIFCONV\FORM\base9

C:\GRAPHICS\GIFCONV\FORM\base9

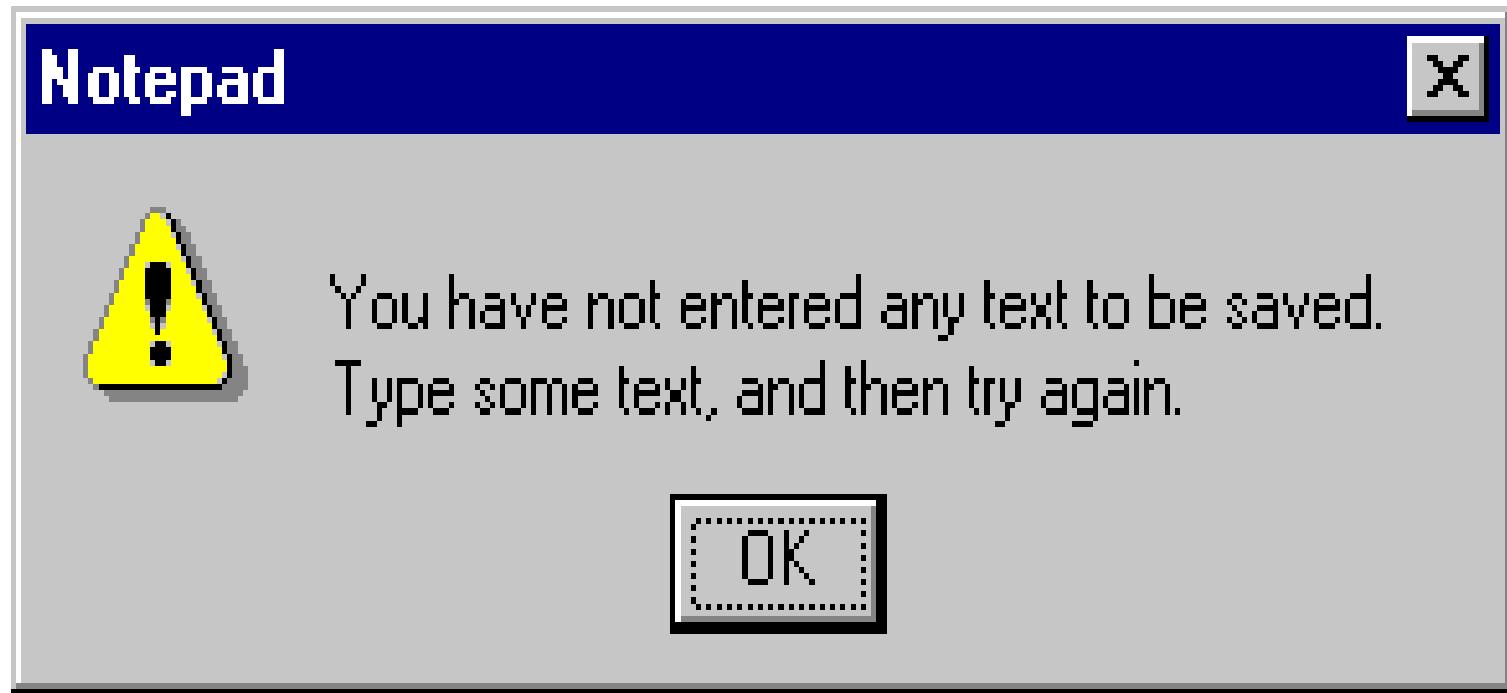
Select

Delete

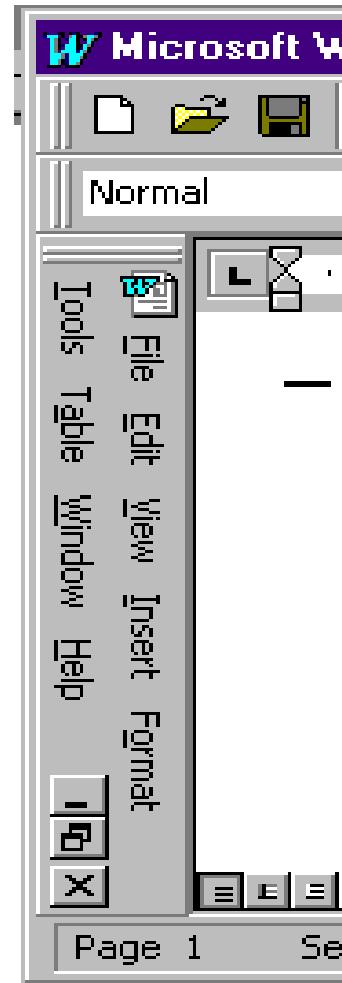
Как НЕ НАДО делать



Как НЕ НАДО делать



Как НЕ НАДО делать



Как НЕ НАДО делать

Whenever your local SIS Administrator sends you an actual software Package, the SIS Package Command Manager will appear (usually at network logon time) displaying the available Package(s). The following screenshots display several options to what you will see when you receive an actual SIS Package.

To start the demonstration, click the "CLICK HERE TO CONTINUE" button of the screen.

Как НЕ НАДО делать

PIRATES Tutorial

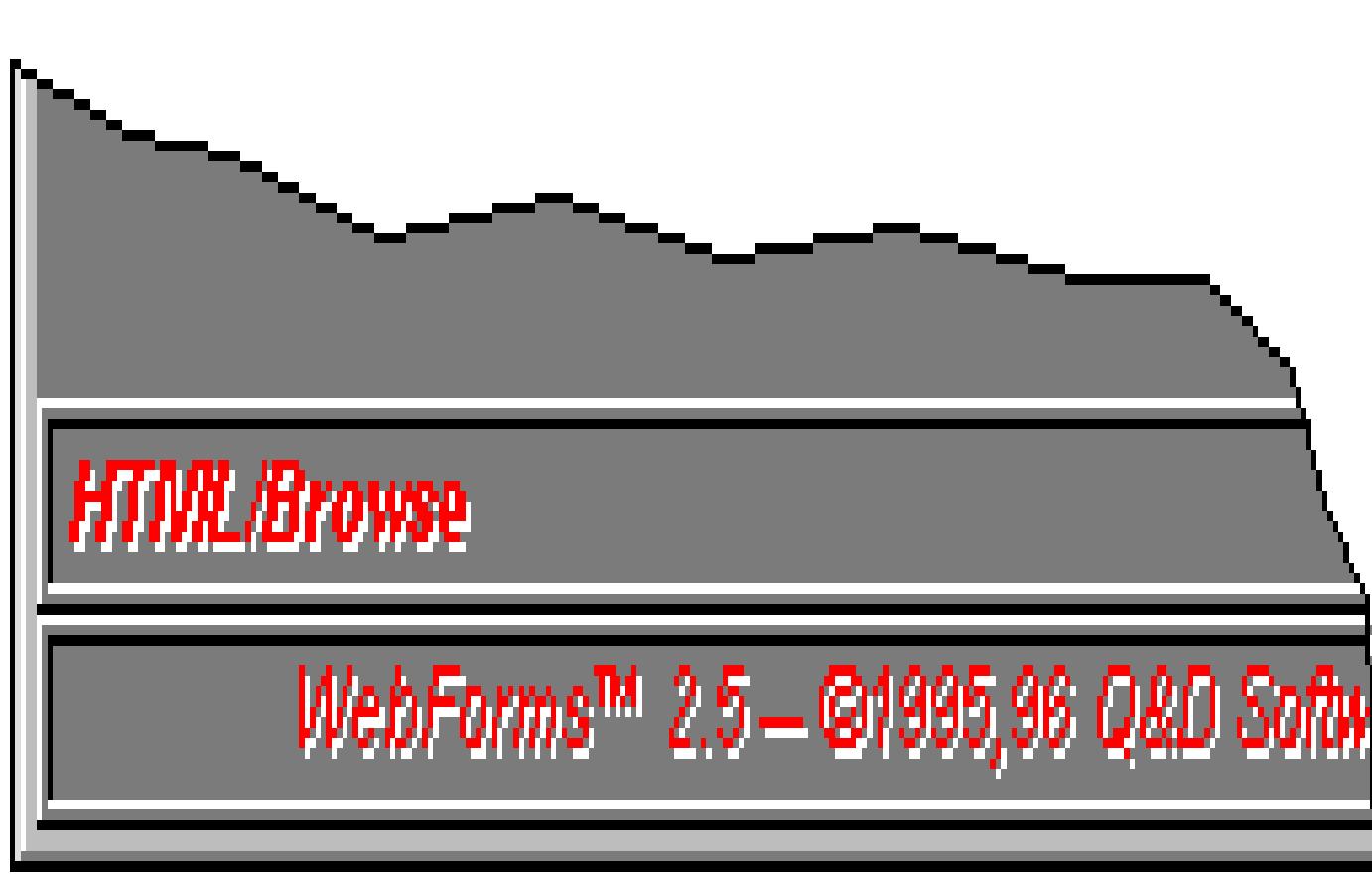
PIRATES is a multiplayer strategy game of piracy and plunder on the high seas. As a ship captain, you may trade with island merchants and make an honest living, or plunder local ships and other players. You will probably do a little of each!

Press arrow to move to next frame -->

Как НЕ НАДО делать

Form Title -- (appears above URL in most browsers and is used by WWW search engines)	Background Color:
Q&D Software Development Order Desk	FFFFBF0
Form Heading -- (appears at top of Web page in bold type)	Text Color:
Q&D Software Development Order Desk	000080
E-Mail responses to (will not appear on status bar)	Background Graphic:
dversch@q-d.com	...
Text to appear in Submit button	Mailto
Send Order	CGI
Clear Form	
Scrolling Status Bar Message (max length = 200 characters)	
WebMania 1.5b with Image Map Wizard is here!!	
<< Prev Tab	Next Tab >>

Как НЕ НАДО делать



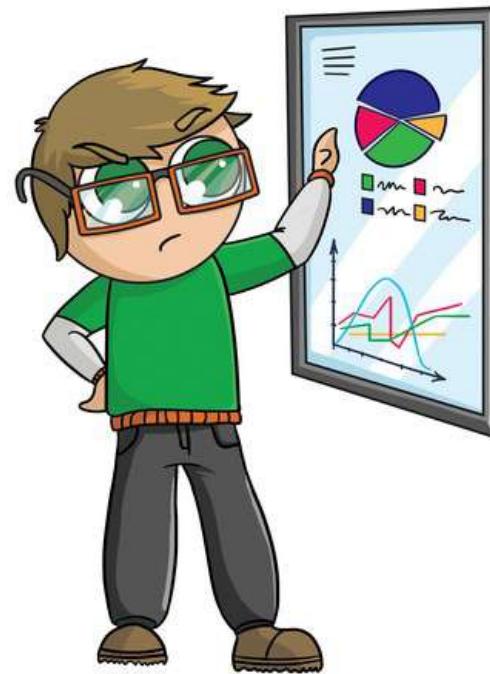
Человеко-машинное взаимодействие

Лекция 12

Мерзлякова Екатерина Юрьевна

к.т.н. доцент ПМИК

Тест-дизайн



Тест-анализ граничит
с аналитикой



Тест-дизайн —
с автоматизацией

Тестовое покрытие

- **Покрытие требований (Requirements Coverage)** - оценка покрытия тестами функциональных и нефункциональных требований к продукту, путем построения матриц трассировки (traceability matrix).
- **Покрытие кода (Code Coverage)** - оценка покрытия исполняемого кода тестами, путем отслеживания непроверенных в процессе тестирования частей программного обеспечения.
- **Тестовое покрытие на базе анализа потока управления** - оценка покрытия основанная на определении путей выполнения кода программного модуля и создания выполняемых тест кейсов для покрытия этих путей.

Техники тест-дизайна

- Эквивалентное Разделение
- Анализ Границых Значений
- Причина / Следствие
- Предугадывание ошибки
- Исчерпывающее тестирование
- Парное тестирование

Использование техники анализа классов эквивалентности

- 1. Необходимо определить класс эквивалентности.
- 2. Затем нужно выбрать одного представителя от каждого класса.
- 3. Нужно выполнить тесты. На этом шаге мы выполняем тесты от каждого класса эквивалентности.

Использование техники анализа классов эквивалентности

Пример: функция подсчета комиссии при отмене бронирования авиабилетов. Размер комиссии зависит от времени до вылета, когда совершена отмена



Использование техники анализа классов эквивалентности

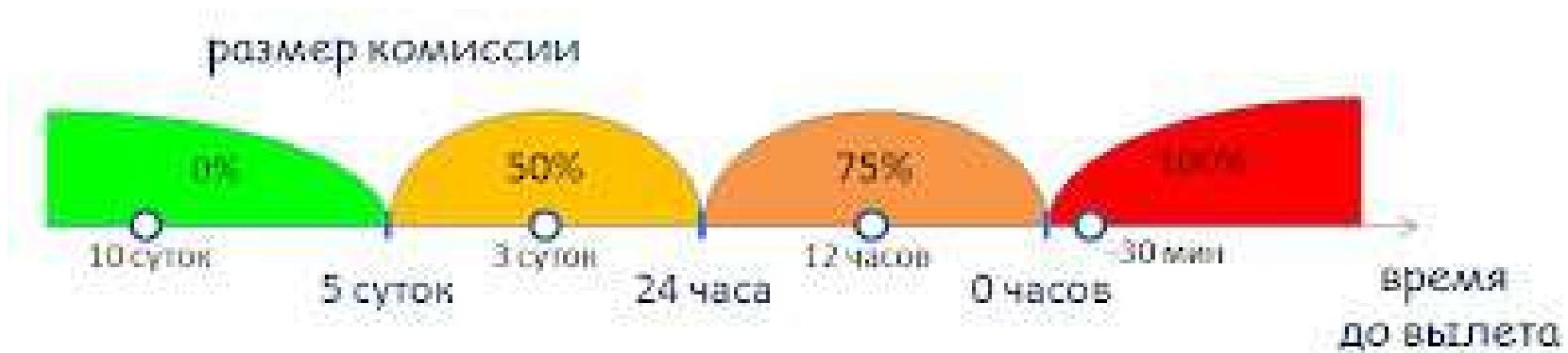
1. Определим классы эквивалентности:

1 класс: время до вылета > 5 суток.

2 класс: 24 часа < время до вылета < 5 суток.

3 класс: 0 часов < время до вылета < 24 часа.

4 класс: время до вылета < 0 часов (вылет уже состоялся).



Использование техники анализа классов эквивалентности

2. Выберем представителя от каждого класса:

- время до вылета = 10 суток (тест из 1-го класса).
- время до вылета = 3 суток (тест из 2-го класса).
- время до вылета = 12 часов (тест из 3-го класса).
- время до вылета = -30 мин (тест из 4-го класса).



Использование техники анализа классов эквивалентности

3. Выполним тесты:

- Отменим бронь за 10 суток до вылета и проверим, что комиссия составила 0%.
- Отменим бронь за 3 суток до вылета и проверим, что комиссия составила 50%.
- Отменим бронь за 12 часов до вылета и проверим, что комиссия составила 75%.
- Отменим бронь через 30 мин после вылета и проверим, что комиссия составила 100%.



Плюсы и минусы техники анализа классов эквивалентности



Заметное сокращение времени и улучшение структурированности тестирования.



При неправильном использовании техники, мы рискуем потерять баги.

Использование техники анализа границых значений

1. Выделить классы эквивалентности.
2. Определить граничные значения этих классов.
3. Понять, к какому классу будет относиться каждая граница.
4. Для каждой границы провести тесты по проверке значения до границы, на границе, и сразу после границы.

Использование техники анализа границых значений

Пример: функция подсчета комиссии при отмене бронирования авиабилетов. Размер комиссии зависит от времени до вылета, когда совершена отмена:



Использование техники анализа границых значений

1. Выделим классы эквивалентности:

- 1 класс: время до вылета > 5 суток.
- 2 класс: 24 часа < время до вылета < 5 суток.
- 3 класс: 0 часов < время до вылета < 24 час.
- 4 класс: время до вылета < 0 часов (вылет уже состоялся).



Использование техники анализа границых значений

2. Определим границы:

- 5 суток.
- 24 часа.
- 0 часов



Использование техники анализа границых значений

**3. Определим, к какому классу относятся
границы:**

5 суток – ко 2-му классу.

24 часа – к о2-му классу.

0 часов – к 4-му классу.



Использование техники анализа границых значений

4. Протестируем значения на границах, до и после них:

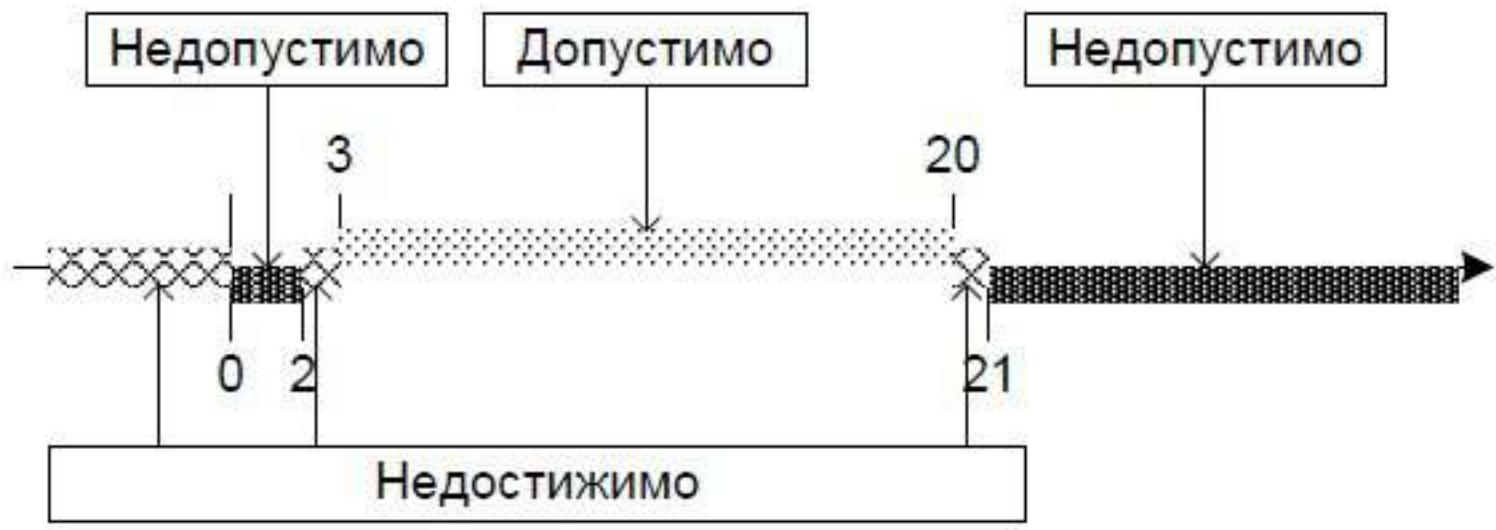
1. Отменим бронь **за 5 суток + 1 секунду** до вылета (или как можно ближе к границе, но слева от нее) и проверим, что комиссия равна 0%.
2. Отменим бронь **ровно за 5 суток** до вылета и проверим, что комиссия равна 50%.
3. Отменим бронь **за 5 суток – 1 секунду** до вылета и проверим, что комиссия равна 50%.
4. Отменим бронь **за 24 часа + 1 секунду** до вылета и проверим, что комиссия равна 50%.
5. Отменим бронь **ровно за 24 часа** до вылета и проверим, что комиссия равна 50%.
6. Отменим бронь **за 24 часа - 1 секунду** до вылета и проверим, что комиссия равна 75%.
7. Отменим бронь **за 1 секунду** до вылета и проверим, что комиссия равна 75%.
8. Отменим бронь **ровно во время вылета** и проверим, что комиссия равна 100%.
9. Отменим бронь **спустя 1 секунду** после вылета и проверим, что комиссия равна 100%.

Плюсы и минусы техники анализа границных значений

- Эта техника добавляет в технику анализа классов эквивалентности ориентированность на конкретный тип ошибок.
- техника граничных значений ориентирована на обнаружение конкретной проблемы – возникновения ошибок на границах классов эквивалентности.
- эффективность техники анализа граничных значений зависит от правильности ее использования.

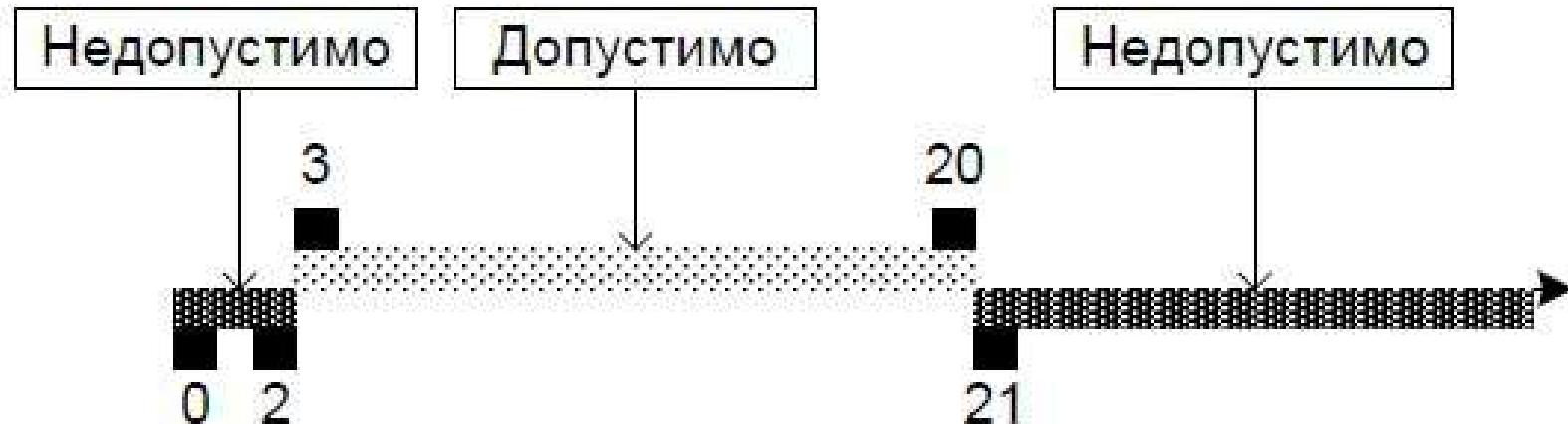


Пример использования техники тестирования на основе классов эквивалентности и граничных условий



Итоговое разбиение на классы эквивалентности значений длины имени пользователя

- $[0, 2]$ — недопустимая длина;
- $[3, 20]$ — допустимая длина;
- $[21, \infty]$ — недопустимая длина.

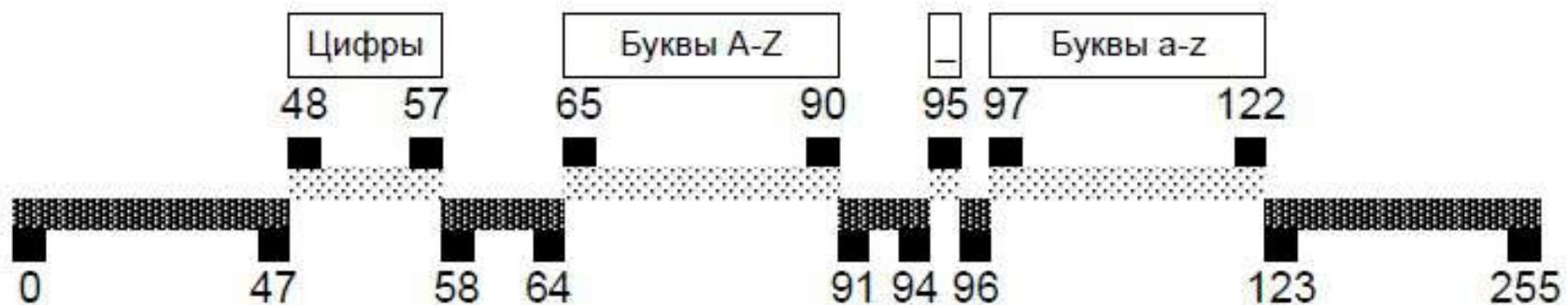


Значения входных данных для тест-кейсов

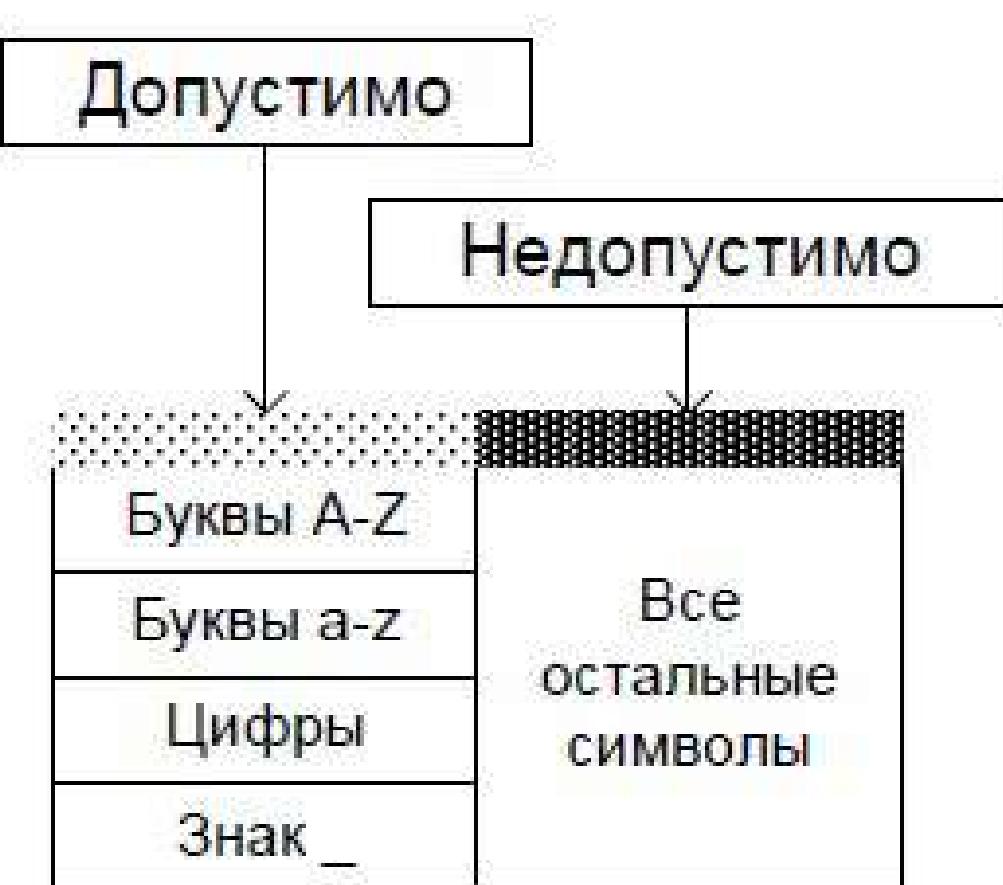
(реакция на длину имени пользователя)

	Позитивные тест-кейсы		Негативные тест-кейсы		
Значение	AAA	123_zzzzzzzzzzzzzz	AA	Пустая строка	1234_zzzzzzzzzzzzzz
Пояснение	Строка минимальной допустимой длины	Строка максимальной допустимой длины	Строка недопустимой длины по нижней границе	Строка недопустимой длины, учтена для надёжности	Строка недопустимой длины по верхней границе

Неудачный способ поиска классов эквивалентности для наборов допустимых и недопустимых символов



Классы эквивалентных допустимых и недопустимых символов



Значения всех входных данных для тест-кейсов

	Позитивные тест-кейсы		Негативные тест-кейсы			
Значение	AAA	123_zzzzzzzzzzzzzzz	AA	Пустая строка	1234_zzzzzzzzzzzzzzz	#\$%
Пояснение	Строка минимальной допустимой длины	Строка максимальной допустимой длины	Строка недопустимой длины по нижней границе	Строка недопустимой длины, учтена для надёжности	Строка недопустимой длины по верхней границе	Строка допустимой длины, недопустимые символы

Тестирование мобильных приложений

- Учесть все модели устройств
- Тестировать самую старую, самую новую ОС и браузеры
- Провести тест-кейсы
- Проверить удобство обновлений
- Проверить работу при слабом Wi-Fi
- Проверить взаимодействие с интерфейсом

Тестирование мобильных приложений. Размер экрана и touch-интерфейс

1. Все элементы должны быть такого размера, чтобы пользователь мог однозначно попасть по ним.
2. Отсутствие пустых экранов в приложении – пользователь не должен оказываться в ситуации, в которой не очевидно, что сейчас происходит и что делать.
3. Следует проверять многократное быстрое нажатие на кнопку – часто при этом может случиться падение приложения. Также следует проверять мультитач – нажатие на несколько кнопок одновременно.
4. Следует проверять наличие или отсутствие «нативных» жестов (pinch-to-zoom, doubletap) – если, например, поддерживается зум части приложения, то должен использоваться жест по умолчанию. А если нет необходимости выделять картинку, то по даблтапу она не должна выделяться.

Тестирование мобильных приложений.

Ресурсы устройства

1. Утечки памяти - проявляется на окнах с большим количеством информации (длинные списки как пример), во время задач с длительным workflow (когда пользователь долго не выходит из приложения), при некорректно работающем кэшировании изображений.
2. Обработка ситуаций нехватки памяти для функционирования ОС, когда приложение активно или работает в фоне.
3. Недостаток места для установки или работы приложения.
4. Отсутствие в некоторых устройствах поддерживаемых приложением функций (3G, SD-карта).
5. Установка или перенос приложения на карту SD.

Тестирование мобильных приложений. Разрешения экрана и версии ОС

1. Ретина и обычные экраны.
2. Адаптация приложения к портретной и альбомной
ориентациям устройства.
3. Версии ОС.
4. Поддержка необходимых медиа-файлов данной моделью
и ОС.
5. Соответствие используемых в приложении view их
смысловому назначению и концепциям платформы.

Тестирование мобильных приложений. Реакция приложения на внешние прерывания

1. Входящие и исходящие SMS, MMS, звонки, оповещения других приложений.
2. Выключение устройства, изъятие аккумулятора, разрядка устройства.
3. Переход в режим ожидания (в том числе и с защитой паролем). Смена ориентации устройства в режиме ожидания.
4. Отключение и подключение провода.
5. Отключение и включение сети, Bluetooth, авиарежима, GPS.
6. Потеря связи с сервером или прокси (подключение есть, но пакеты не доходят).
7. Отключение и подключение SD-карты, дополнительных устройств вроде физической клавиатуры или гарнитуры.
8. Зарядка устройства, работа с физической клавиатурой.

Тестирование мобильных приложений. Платный контент внутри приложения

1. Соответствие цены и содержимого, заявленного в приложении.
2. Восстановление покупки (обновление приложения).

Тестирование мобильных приложений. Интернационализация

1. Проверка корректности перевода.
2. Проверка того, что все надписи входят в соответствующие формы, кнопки и т.п.
3. Проверка форматов дат, разделителей в числах, специфических особенностей локализации (вроде пробела перед знаком вопроса во французской, верхних индексов “о” и “а”, в порядковых числительных в испанской и других нетривиальных моментах).

Тестирование мобильных приложений. Обновления

1. Убедиться, что поддерживаются те же версии ОС, что и предыдущая версия (если новая версия приложения использует новые возможности ОС, то для старых поддерживаемых версий ОС необходимо создание урезанной версии приложения).
2. Проверка адекватного обновления (сохраняются все данные пользователя и т. п.).

Тестирование мобильных приложений. Постоянная обратная связь с пользователем

1. У всех нажимаемых элементов должно быть нажатое состояние (отклик на действие). В Android-приложениях у элементов может быть ещё одно состояние – focused.
2. Реакция кнопок на нажатие. Скорость отклика элементов должна быть достаточно высокой. Желательно использовать для проверки этого пункта самые слабые устройства среди поддерживаемых.
3. Сообщения при загрузке контента или прогресс-бар.
4. Сообщения при ошибке доступа к сети, GPS.
5. Наличие понятных сообщений при попытке удалить важную информацию.
6. Наличие экрана или сообщения при окончании процесса или игры.
7. Наличие и синхронность звуков или вибрации с уведомлениями и другими событиями на экране.

Тестирование мобильных приложений. Жесты.

Tap



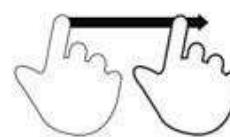
Briefly touch surface with fingertip

Double tap



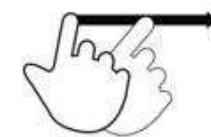
Rapidly touch surface twice with fingertip

Drag



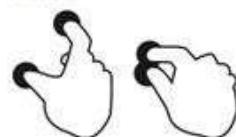
Move fingertip over surface without losing contact

Flick



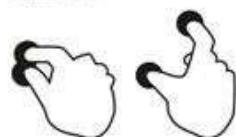
Quickly brush surface with fingertip

Pinch



Touch surface with two fingers and bring them closer together

Spread



Touch surface with two fingers and move them apart

Press



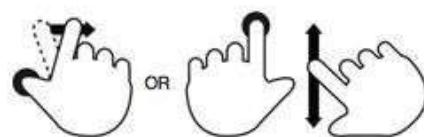
Touch surface for extended period of time

Press and tap



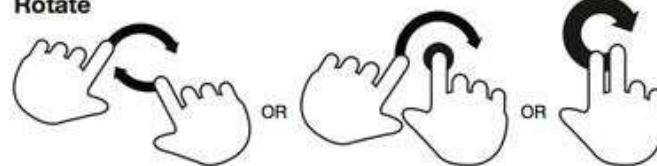
Press surface with one finger and briefly touch surface with second finger

Press and drag



Press surface with one finger and move second finger over surface without losing contact

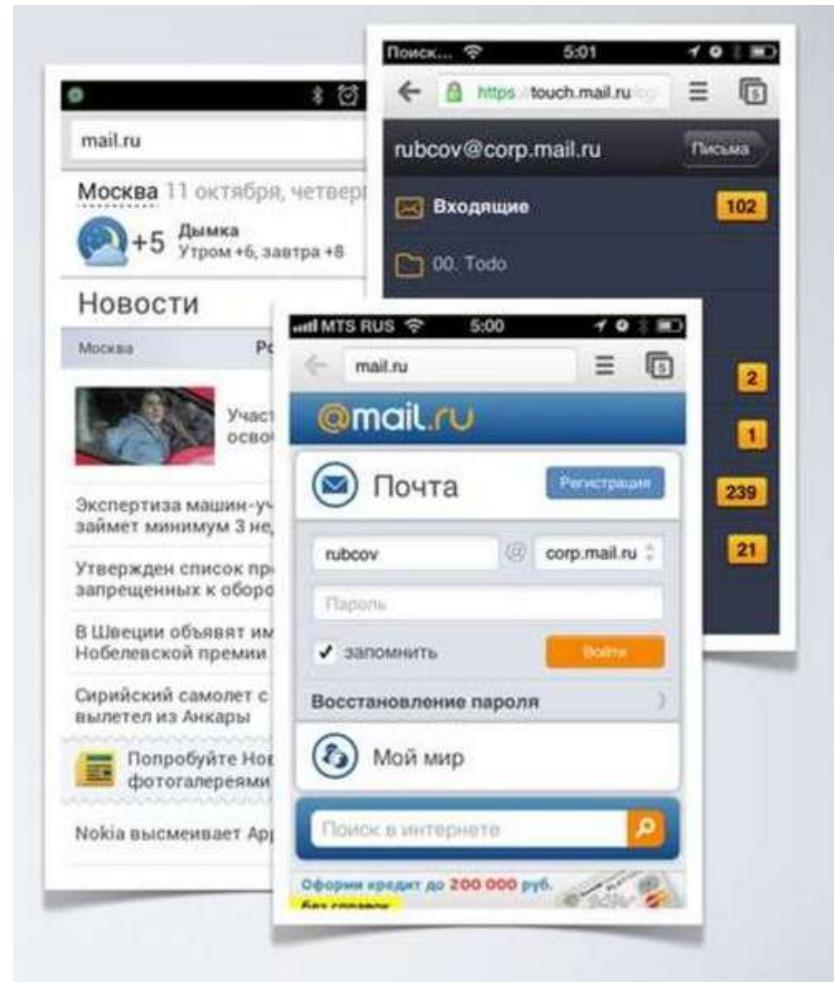
Rotate



Touch surface with two fingers and move them in a clockwise or counterclockwise direction

Типы мобильных приложений

Мобильные веб-приложения



Типы мобильных приложений

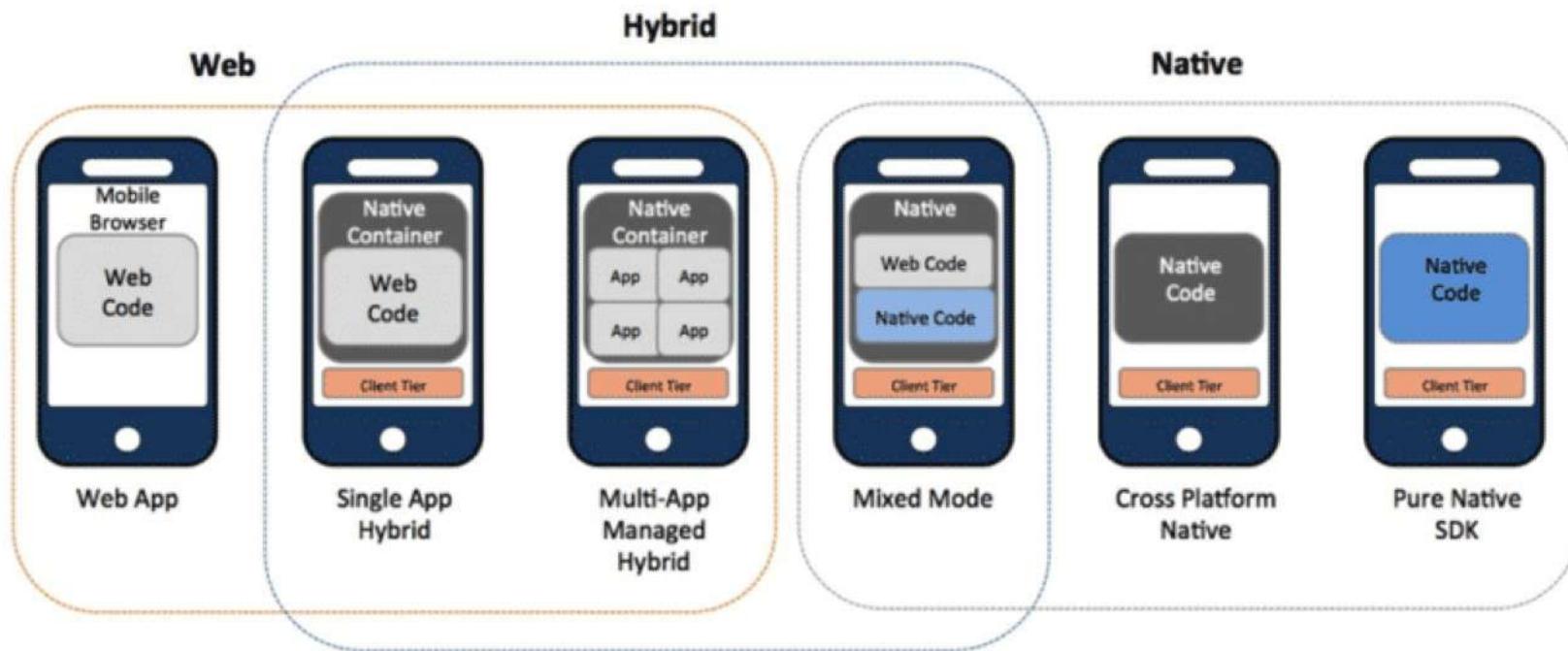
Нативные приложения

NATIVE



Типы мобильных приложений

Гибридные приложения



Инструменты для тестирования мобильных приложений

1. Эмуляторы устройств
2. DevTools
3. Сервисы TestFlight и Beta
4. Снiffeры



Стандарты, регламентирующие процесс тестирования

- наличие целей тестирования на каждом уровне тестирования;
- любому процессу разработки соответствует свой процесс тестирования;
- возможность рецензирования документации тестировщиками начиная с самых первых версий;
- анализ требований и написание тестов начинается одновременно с деятельностью разработчиков или раньше.

Стандарты, регламентирующие процесс тестирования

IEEE 12207/ISO/IEC 12207-2008 Software Life Cycle Processes – описывает жизненный цикл программного обеспечения и место различных процессов в нём;

ISO/IEC 9126-1:2001 Software Engineering – Software Product Quality – описывает характеристики качества программных продуктов;

IEEE 829-1998 Standard for Software Test Documentation – описывает виды документов, служащих для подготовки тестов;

IEEE 1008-1987 (R1993, R2002) Standard for Software Unit Testing – описывает организацию модульного тестирования;

ISO/IEC 12119:1994 Information Technology. Software Packages – Quality Requirements and Testing (аналог AS/NZS 4366:1996 и ГОСТ Р-2000, также принят IEEE под номером IEEE 1465-1998) – описывает требования к процедурам тестирования программных систем;

ISO/IEC 20000:2005. Процессы, сертификация ISO 20000 – описывает требования к управлению и обслуживанию IT-сервисов.

IEEE 12207/ISO/IEC 12207-2008 Software Life Cycle Processes

Под жизненным циклом программного обеспечения (ЖЦ ПО) понимается цикл создания и развития ПО, который начинается с момента принятия решения о необходимости создания программного продукта и завершается в момент изъятия программного продукта из эксплуатации.

Группировка процессов жизненного цикла ПО согласно IEEE 12207

Фактор	Атрибуты
Процессы соглашения	<ul style="list-style-type: none">•Поставка;•приобретение
Процессы организационного обеспечения проекта	<ul style="list-style-type: none">•Процесс менеджмента модели жизненного цикла;•процесс менеджмента инфраструктуры;•процесс менеджмента портфеля проектов;•процесс менеджмента людских ресурсов;•процесс менеджмента качества

Группировка процессов жизненного цикла ПО согласно IEEE 12207

Фактор	Атрибуты
Процессы проекта	<ul style="list-style-type: none">•Процессы менеджмента проекта:<ul style="list-style-type: none">✓ процесс планирования проекта;✓ процесс управления и оценки проекта;•процессы поддержки проекта:<ul style="list-style-type: none">✓ процесс менеджмента решений;✓ процесс менеджмента рисков;✓ процесс менеджмента конфигурации;✓ процесс менеджмента информации;✓ процесс измерений

Группировка процессов жизненного цикла ПО согласно IEEE 12207

Фактор	Атрибуты
Технические процессы	<ul style="list-style-type: none">• Определение требований правообладателей;• анализ системных требований;• проектирование архитектуры системы;• процесс реализации;• процесс комплексирования системы;• процесс квалификационного тестирования системы;• процесс инсталляции программных средств;• процесс поддержки приемки программных средств;• процесс функционирования программных средств;• процесс сопровождения программных средств;• процесс изъятия из обращения программных средств

Группировка процессов жизненного цикла ПО согласно IEEE 12207

Фактор	Атрибуты
Процессы реализации программных средств	<ul style="list-style-type: none">•Процесс анализа требований к программным средствам;•процесс проектирования архитектуры программных средств;•процесс детального проектирования программных средств;•процесс конструирования программных средств;•процесс комплексирования программных средств;•процесс квалификационного тестирования программных средств

Группировка процессов жизненного цикла ПО согласно IEEE 12207

Фактор	Атрибуты
Процессы поддержки программных средств	<ul style="list-style-type: none">•Процесс менеджмента документации программных средств;•процесс менеджмента конфигурации программных средств;•процесс обеспечения гарантии качества программных средств;•процесс верификации программных средств;•процесс валидации программных средств;•процесс ревизии программных средств;•процесс аудита программных средств;•процесс решения проблем в программных средствах

Группировка процессов жизненного цикла ПО согласно IEEE 12207

Фактор	Атрибуты
Процессы повторного применения программных средств	<ul style="list-style-type: none">• Процесс проектирования доменов;• процесс менеджмента повторного применения активов;• процесс менеджмента повторного применения программ

ISO/IEC 9126-1:2001 Software Engineering – Software Product Quality

- **точку зрения разработчиков**, которые воспринимают внутреннее качество ПО;
- **точку зрения руководства и аттестации ПО** на соответствие сформулированным к нему требованиям, в ходе которой определяется внешнее качество ПО;
- **точку зрения пользователей**, ощущающих качество ПО при использовании.

ISO/IEC 9126-1:2001 Software Engineering – Software Product Quality



Факторы и атрибуты внешнего и внутреннего качества ПО согласно ISO 9126

Функциональность (functionality) – способность ПО в определенных условиях решать задачи, нужные пользователям. Определяет, что именно делает ПО

1. *Функциональная пригодность (suitability)* – способность решать нужный набор задач
2. *Точность (accuracy)* – способность выдавать нужные результаты
3. *Способность к взаимодействию, совместимость (interoperability)* – способность взаимодействовать с нужным набором других систем
4. *Соответствие стандартам и правилам (compliance)* – соответствие ПО имеющимся стандартам, нормативным и законодательным актам, другим регулирующим нормам
5. *Защищенность (security)* – способность предотвращать неавторизованный и неразрешенный доступ к данным, коммуникациям и другим элементам ПО

Факторы и атрибуты внешнего и внутреннего качества ПО согласно ISO 9126

Надежность (reliability) – способность ПО поддерживать определенную работоспособность в заданных условиях

1. *Зрелость, завершенность (maturity)* – величина, обратная частоте отказов ПО. Определяется средним временем работы без сбоев и величиной, обратной вероятности возникновения отказа за данный период времени
2. *Устойчивость к отказам (fault tolerance)* – способность поддерживать заданный уровень работоспособности при отказах и нарушениях правил взаимодействия с окружением
3. *Способность к восстановлению (recoverability)* – способность восстанавливать определенный уровень работоспособности и целостность данных после отказа в рамках заданных времени и ресурсов
4. *Соответствие стандартам надежности (reliability compliance)*

Факторы и атрибуты внешнего и внутреннего качества ПО согласно ISO 9126

Удобство использования (usability), или практичность – способность ПО быть удобным в обучении и использовании, а также привлекательным для пользователей

1. *Понятность (understandability)* – показатель, обратный усилиям, которые затрачиваются пользователями на восприятие основных понятий ПО и осознание способов их использования для решения своих задач
2. *Удобство обучения (learnability)* – показатель, обратный усилиям, затрачиваемым пользователями на обучение работе с ПО
3. *Удобство работы (operability)* – показатель, обратный трудоемкости решения пользователями задач с помощью ПО
4. *Привлекательность (attractiveness)* – способность ПО быть привлекательным для пользователей
5. *Соответствие стандартам удобства использования (usability compliance)*

Факторы и атрибуты внешнего и внутреннего качества ПО согласно ISO 9126

Производительность (efficiency), или эффективность, – способность ПО при заданных условиях обеспечивать необходимую работоспособность по отношению к выделяемым для этого ресурсам

1. *Временная эффективность (time behaviour)* – способность ПО решать определенные задачи за отведенное время
2. *Эффективность использования ресурсов (resource utilisation)* – способность решать нужные задачи с использованием заданных объемов ресурсов определенных видов. Имеются в виду такие ресурсы, как оперативная и долговременная память, сетевые соединения, устройства ввода и вывода и пр.
3. *Соответствие стандартам производительности (efficiency compliance)*

Факторы и атрибуты внешнего и внутреннего качества ПО согласно ISO 9126

Удобство сопровождения (maintainability) – удобство проведения всех видов деятельности, связанных с сопровождением программ

1. Анализируемость (*analyzability*), или удобство проведения анализа – удобство проведения анализа ошибок, дефектов и недостатков, а также удобство анализа необходимости изменений и их возможных последствий
2. Удобство внесения изменений (*changeability*) – показатель, обратный трудозатратам на выполнение необходимых изменений
3. Стабильность (*stability*) – показатель, обратный риску возникновения неожиданных эффектов при внесении необходимых изменений
4. Удобство проверки (*testability*) – показатель, обратный трудозатратам на проведение тестирования и других видов проверки того, что внесенные изменения привели к нужным результатам
5. Соответствие стандартам удобства сопровождения (*maintainability compliance*)

Факторы и атрибуты внешнего и внутреннего качества ПО согласно ISO 9126

Переносимость (portability) – способность ПО сохранять работоспособность при переносе из одного окружения в другое, включая организационные, аппаратные и программные аспекты окружения

1. Адаптируемость (*adaptability*) – способность ПО приспосабливаться к различным окружениям без проведения для этого действий, помимо заранее предусмотренных
2. Удобство установки (*installability*) – способность ПО быть установленным или развернутым в определенном окружении
3. Способность к сосуществованию (*coexistence*) – способность ПО сосуществовать в общем окружении с другими программами, деля с ними ресурсы
4. Удобство замены (*replaceability*) другого ПО данным – возможность применения данного ПО вместо других программных систем для решения тех же задач в определенном окружении.
5. Соответствие стандартам переносимости (*portability compliance*)

Факторы оценки качества ПО с точки зрения пользователей согласно ISO 9126

Фактор	Описание
Эффективность (effectiveness)	Способность решать задачи пользователей с необходимой точностью при использовании в заданном контексте
Продуктивность (productivity)	Способность предоставлять определенные результаты в рамках ожидаемых затрат ресурсов
Безопасность (safety)	Способность обеспечивать необходимо низкий уровень риска нанесения ущерба жизни и здоровью людей, бизнесу, собственности или окружающей среде
Удовлетворение пользователей (satisfaction)	Способность приносить удовлетворение пользователям при использовании в заданном контексте

IEEE 829-1998 Standard for Software Test Documentation

Артефакт	Описание
Тестовый план (test plan)	Основной документ, связывающий разработку тестов и тестирование с задачами проекта. Определяет необходимые виды тестирования, техники, проверяемую функциональность, критерии оценки полноты тестов и критерии выхода, а также график тестирования
Тестовые сценарии (test case)	Сценарии проведения отдельных тестов. Каждый тестовый сценарий предназначен для проверки определенных свойств некоторых компонентов системы в определенной конфигурации

IEEE 829-1998 Standard for Software Test Documentation

Артефакт	Описание
Описания тестовых процедур (test procedure specifications)	Тестовые процедуры могут быть представлены в виде скриптов или программ, автоматизирующих запуск тестовых сценариев (автоматизированное тестирование), или в виде инструкций для человека, следуя которым можно выполнить те же сценарии (ручное тестирование)
Отчеты о нарушениях (test incident reports, или bug reports)	Сценарии проведения отдельных тестов. Каждый тестовый сценарий предназначен для проверки определенных свойств некоторых компонентов системы в определенной конфигурации
Итоговый отчет о тестировании (test summary report)	Отчет, аккумулирующий общую информацию по результатам тестов, включающий достигнутое тестовое покрытие и общую оценку качества компонентов тестируемой системы

IEEE 1008-1987 (R1993, R2002) Standard for Software Unit Testing

- планирование – определение используемых методов тестирования необходимых ресурсов, критериев полноты тестов и критерия выхода;
- определение проверяемых требований и ограничений;
- уточнение, корректировка и детализация планов;
- разработка набора тестов;
- выполнение тестов;
- проверка достижения критерия выхода с последующей оценкой полноты тестирования по специально выбранным критериям;
- оценка затрат ресурсов и качества протестированных модулей.

ISO/IEC 12119:1994 Information Technology. Software Packages – Quality Requirements and Testing

- 1) общие требования к содержанию;
- 2) обозначения и указания;
- 3) функциональные возможности;
- 4) надёжность;
- 5) практичность;
- 6) эффективность;
- 7) сопровождаемость и мобильность

ISO/IEC 20 000:2005. Процессы, сертификация ISO 20 000

1) ISO 20 000-1:2005 «**Information technology – Service management.**
Part 1: Specification»

- а) **процессы предоставления сервисов (Service delivery processes)** – группа процессов управления уровнем сервисов, предоставлением отчётности по предоставлению сервисов, составлением бюджета и статистики затрат;
- б) **процессы управления взаимодействием (Relationship processes)** – описываются вопросы коммуникации между компанией, заказчиком и различными подрядчиками;
- в) **процессы разрешения (Resolution processes)** – управление проблемами и инцидентами;
- г) **процессы контроля (Control processes)** – описываются особенности контроля и управления изменениями в компании;
- д) **процессы управления релизами (Release processes)** – описываются активности, связанные с внедрением новых и уже имеющихся решений.

ISO/IEC 20 000:2005. Процессы, сертификация ISO 20 000

2) ISO 20000-2:2005 «Information technology – Service management.

Part 2: Code of Practice»

- **планирование (PLAN)** – подразумевает ответы на вопросы «что?», «когда?» нужно сделать, а также «кто?» и «с помощью чего?» должен это сделать, т. е. на данном этапе проектируются или корректируются процессы;
- **выполнение (DO)** – включает выполнение запланированных на первом этапе работ;
- **проверка (CHECK)** – собираются метрики, готовится отчётность, согласно которой выявляется, какой результат дало выполнение работ;
- **действие (ACT)** – планы корректируются согласно результатам предыдущего этапа, проводятся требующиеся изменения.
- В стандарте также определяются требования к мерам ответственности руководителей компании, которая занимается разработкой, т. е. предоставляет IT-сервисы, к компетенции персонала и управлению документацией.