Qt

QtCore

Ссылки

http://qt-project.org
http://qt-project.org/doc/qt-4.8

Windows: Qt SDK 1.2.1

Linux: yum install make gcc-c++ qt-devel или qt4-devel
или apt-get install libqt4-dev qt-dev-tools build-essential

http://server.aesc.msu.ru

Передача параметров

C:

- все параметры передаются по значению

C++:

- по значению
  int f(int x, point* y);
- по ссылке
  int f(int& x, point& y);
- по константной ссылке
  int f(const int& x, const point& y);

Работа с памятью

C:

- память на стэке:
  point p;
- память в куче:
  point *p = malloc(sizeof(point));
  point *ps = malloc(n * sizeof(point));
  free(p); free(ps);

C++:

- свободная память:
  point *p = new point; point *ps = new point[n];
  delete p; delete[] ps;

Структуры

```
struct point {
    int x, y;
};
struct rect {
    point p1, p2;
    int color;
};
```

Функции-члены

```
struct rect {
    point p1, p2; int color;
    void draw(painter* p) {
        p->drawRect(p1.x, p1.y, p2.x, p2.y, color);
    }
};
rect r;
r.draw(p);
```

Область видимости

```
struct rect {
        void draw(painter* p) { ++x; }
        int timesDrawed() const { return x; }
    private:
        int x;
    public:
        point p1, p2;
}
```

Специальные функции-члены

```
class X {
    X(); /* Стандартный конструктор */
    ~X(); /* Деструктор */
    X(const X&); // Конструктор копирования
    X& operator=(const X&); /* Оператор
присваивания */
};
```

Конструктор

```
struct point {
    point() { x = y = 0; }
    point(int _x, int _y) { x = _x; y = _y; }
    int x, y;
};
point p1;
point *p2 = new point(1, 2);
```

Деструктор

```
struct PointOnScreen {
    PointOnScreen(painter* _p, int _x, int _y) {
        p = _p, x = _x; y = _y;
        p->setPoint(x, y, BLACK);
    }
    ~PointOnScreen(){p->setPoint(x, y,
WHITE);}
private:
    painter *p; int x, y;
};
```

Наследование

```
struct SecondTimer {
    int secsPassed;
    void secPassed() { ++secsPassed; }
};
struct MSecondTimer : SecondTimer {
    int msecsPassed;
    int msecPassed() {
        if (++msecsPassed == 1000) {
            msecsPassed = 0; ++secsPassed; } }
};
```

Полиморфные члены

```
void f(SecondsTimer* t) {
    std::cout << t->secsPassed;
}
MSecondsTimer timer;
f(&timer);
```

Полиморфное поведение

```
struct Figure {
    virtual void draw(painter* p) = 0;
};
struct Rect : Figure {
    void draw(painter* p) { ... }
};
Figure *f = new Rect; f->draw(p);
Rect *r = dynamic_cast<Rect*>(f);
if (r) { ... }
```

Наследование конструктора

```
class Point {
    Point(int _x, int _y): x(_x), y(_y) {}
    int x, y;
};
class Rect {
    Rect(int x1, int x2, int y1, int y2, painter& _p):
        p1(x1,y1), p2(x2, y2), p(_p) {}
    painter& p;
    Point p1, p2;
};
```

Объявление класса в .h-файле

- rect.h:

```
#ifndef RECT_H_INCLUDED
#define RECT_H_INCLUDED
struct Rect {
    int x, y;
    void draw(painter *p);
};
#endif
```

- rect.cpp:

```
#include "rect.h"
void Rect::draw(painter *p) { ... }
```

Шаблоны в C++

```
struct AbstractDeleter {
    virtual void del() = 0;
};
template<class T>
struct Deleter<T*> : AbstractDeleter {
    Deleter(T* _obj) { obj = _obj; }
    void del() { delete obj; }
    T* obj;
};
```

Шаблоны в C++

```
list<AbstractDeleter*> queue;
template<class T>
void queueDelete(const T& object) {
    queue.push(new Deleter<T*>(&object));
}
void deleteAll() {
    for (Deleter* x : queue) { x->del(); delete x; }
    queue.erase();
}
```

Полезные классы Qt

- QString
- QByteArray
- QList<T>
- QVector<T>
- QLinkedList<T>
- QMap<T>, QHash<T>
- QStack<T>, QQueue<T>
- QVariant
- QPair
- Q*Iterator, Q*<T>::iterator

Сигналы и слоты Qt

```
class Rect : public QObject {
    Q_OBJECT
    int x, y;
    void setX(int x);
    public slots:
        void moveRight();
    signals:
        void changed();
};
```

Сигналы и слоты Qt

```
class Button : public QObject { Q_OBJECT
    public:
        void emitClicked();
    signals:
        void clicked();
};
class Scene : public QObject { Q_OBJECT
    public slots:
        void redraw();
};
```

Сигналы и слоты Qt

```
void Rect::setX(int x) {
    this->x = x;
    emit changed();
}
void Rect::moveRight() { setX(x + 1); }
void Button::emitClicked() { emit clicked(); }
void Scene::redraw() { qDebug("redraw"); }
```

Сигналы и слоты Qt

```
int main(int ac, char** av) {
    QCoreApplication app(ac, av);
    Rect r; Button b; Scene s;
    QObject::connect(&b, SIGNAL(clicked()),
                         &r, SLOT(moveRight()));
    QObject::connect(&r, SIGNAL(changed()),
                         &s, SLOT(redraw()));
    b.emitClicked();
}
```

Иерархия объектов

```
QObject::QObject(QObject* parent);
QList<QObject*> QObject::children();

QObject *x = new QObject;
QObject *y = new QObject(x);
delete x;
```

Система сборки Qt

$ qmake-qt4 -project # создаст *.pro
$ qmake-qt4 # создаст Makefile
$ make # скомпилирует программу
$ ./qt-project
redraw

Qt

QtGui

QWidget

QWidget

- QLabel
  - QLabel(QString text, QWidget*)
  - slot setPixmap(QPixmap), setText(QString)
- QPushButton
  - QPushButton(QString text, QWidget*)
  - signal clicked()
  - slot click(), animateClick()
- QLineEdit, QTextEdit
  - QLineEdit(QString contents, QWidget*)
  - signal textChanged(QString), returnPressed()
  - slot setText(QString)
- QCheckBox, QRadioButton,
  - ...

QLayout

- QVBoxLayout, QHBoxLayout
  - addWidget(QWidget*, ...)
  - addLayout(QLayout*, ...)
  - addStretch()
- QGridLayout
  - addWidget(QWidget*, int, int, int, int)
  - addLayout(QLayout*, int, int, int, int)
- QFormLayout
  - addRow({QWidget*|QString}, {QLayout*|QWidget*})
  - addRow({QWidget*|QLayout*})

Форма

```
struct MyForm : QWidget {
    MyForm();
    QLineEdit *fname, *lname;
    public slots: void setName(); void greet();
};
int main(int ac, char** av) {
    QApplication app(ac, av);
    MyForm f; f.show();
    return app.exec(); }
```

Форма

```
MyForm::MyForm(): QWidget(NULL) {
    fname = new QLineEdit; lname = QLineEdit;
    QFormLayout *l = new QFormLayout;
    QLabel *lbl = new QLabel("Enter your name!");
    QPushButton *b = new QPushButton("Submit");
    l->addRow(lbl);
    l->addRow("First name", fname);
    l->addRow("Last name", lname);
    l->addRow(b);
    setLayout(l);
```

Форма

```
    connect(fname, SIGNAL(returnPressed()),
                    lname, SLOT(setFocus()));
    connect(lname, SIGNAL(returnPressed()),
                    b, SIGNAL(clicked()));
    connect(fname, SIGNAL(textChanged(QString)),
                    this, SLOT(setName()));
    connect(lname, SIGNAL(textChanged(QString)),
                    this, SLOT(setName()));
    connect(b, SIGNAL(clicked()),
                    this, SLOT(greet()));
}
```

Форма

```
void MyForm::setName() {
    lbl->setText(QString("Hello, %1 %2!")
                 .arg(fname->text()).arg(lname->text()));
}
void MyForm::greet() {
    if (QMessageBox::information(this, "Greeter",
lbl->text(), QMessageBox::Ok | QMessageBox::Close) ==
QMessageBox::Close)
    QApplication::instance()->quit();
}
```

Рисование

- QPaintDevice
  - QPixmap
    - Оптимизирован для отображения на экране
  - QImage
    - Оптимизирован для изменения пикселей
  - QPicture
    - Запоминает последовательность рисований
- QPainter
  - Обеспечивает рисование всех примитивов
- QPen (границы), QBrush (заливка), QFont
- QColor
  - typedef uint32_t QRgb;
  - QRgb qRgb(int r, int g, int b);

QImage

- Дорогие функции
  - QColor QImage::pixel(int x, int y)
  - QColor QImage::setPixel(int x, int y, int color)
- Дешевые функции
  - QRgb* QImage::scanLine(int i)
  - QRgb* QImage::bits();

QImage

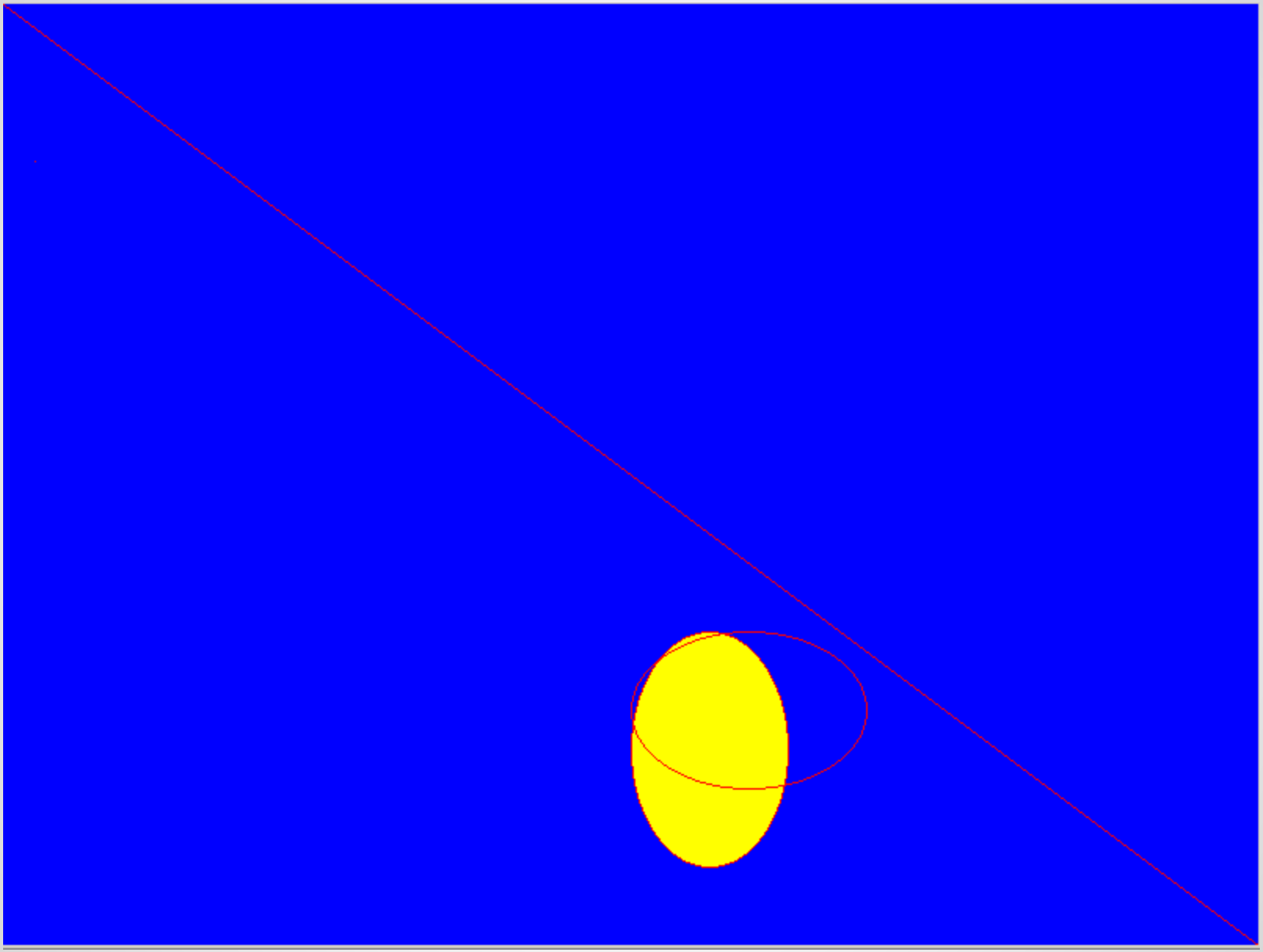
```
QImage img(800, 600, Image::
Format_RGB32);
img.fill(QColor(255,255,0));
for (int i = 0; i < 600; ++i) {
    QRgb *line = (QRgb*)img.scanLine(i);
    for (int j = 0; j < 800; ++j) {
        line[j] = qRgb(i / 600.0 * 255,
                       (1 - j / 800.0) * 255, 0);
    }
}
```

QImage

```
QLabel lbl;
lbl.setPixmap(QPixmap::fromImage(img));
lbl.show();
```

QPainter

```
QPixmap pic; QPainter p; p.begin(&pic);
p.setBrush(QBrush(QColor(255,255,0)));
p.setPen(QColor(255,0,0));
p.drawLine(QPoint(0,0),QPoint(800,600));
p.drawPoint(20,100);
p.drawEllipse(400, 400, 100, 150);
p.setBrush(Qt::NoBrush);
p.drawEllipse(400, 400, 150, 100);
p.end();
```



QEvent

Qt

QtNetwork

Ссылка

http://doc.crossplatform.ru/qt/4.7.x/

Socket

- Две конечных точки: клиент и сервер
- Потоковая передача (TCP)
  - Клиент открывает соединение к серверу, используя заранее известный идентификатор
  - Записанное в сокет на одном конце читается на другом
- Передача сообщений (UDP)
  - Клиент отправляет сообщение серверу
  - Сервер получает это сообщение

Блокирующая работа с сетью

```
write(sock1, "GET / HTTP/1.0\r\n\r\n");
/* Ожидание записи в соединение */
read(sock1, buf);
/* Ожидание чтения из соединения */
```

Неблокирующая работа с сетью

```
for (;;) {
    (event, sock) = poll(sock1, sock2, ..., sockN);
    /* ожидание любого события */
    if (event == READY_WRITE) {
        write(sock, "GET / HTTP/1.0\r\n\r\n");
    } else if (event == READY_READ) {
        read(sock, buf);...
    }
}
```

processEvents()

```
void DifficultCalculation() [slot] {
    for (int step = 0; step < 1e9; ++step) {
        /* Сложные вычисления */
        QApplication::instance()-
>processEvents();
    }
}
```

QEventLoop

```
QEventLoop loop;
connect(button, SIGNAL(clicked()),
            &loop, SLOT(quit()));
loop.exec();
qDebug() << "Button was pressed!";
```

QTimer

- QTimer()
- start(int msec)
- signal timeout()
- void setSingleShot(bool)

QIODevice

- qint64 read(void* data, qint64 size);
- QByteArray readAll();
- void write(void* data, qint64 size);
- void write(const QByteArray& data);
- void close();
- signal readyRead();
- signal readChannelFinished();
- Дети
  - QBuffer
  - QAbstractSocket
  - QFile
  - QNetworkReply

QAbstractSocket

- void connectToHost(QString h, qint16 p)
- signal connected()
- signal error(QAbstactSocket::SocketError)
- signal disconnected()

QTcpSocket

http://server.aesc.msu/qt/fortune/main.cpp

QTcpServer

- bool listen(QHostAddress, quint16)
- bool hasPendingConnections()
- QTcpSocket *nextPendingConnection()
- signal newConnection()

QTcpServer

http://server.aesc.msu/qt/fortune-srv/main.cpp

QDataStream

- QDataStream(QIODevice*)
- setVersion(int)
- operator>>(...)
- operator<<(...)