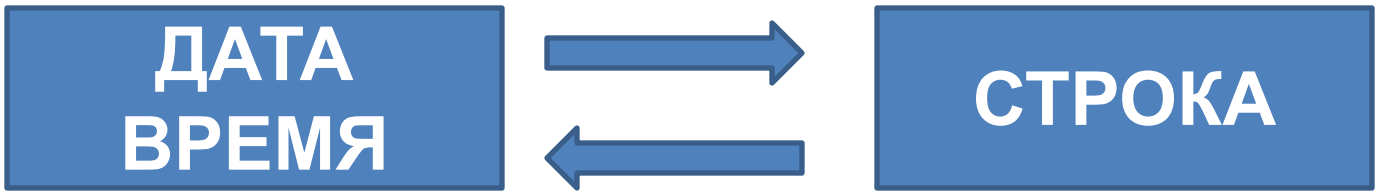
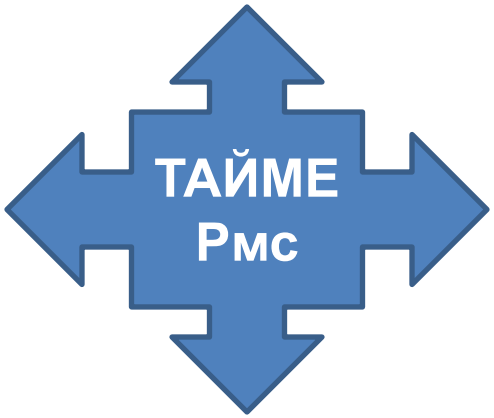
ДАТА И ВРЕМЯ В QT

- **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; • возвращает целое значение
month()	• возвращает месяца в диапазоне от 1 до 12 (1 – январь)
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

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.ddddddd"); //str = "2014.
Июль.суббота"
```

ДАТА И ВРЕМЯ В QT: КЛАСС ДАТЫ

- ***addDays()***
- ***addMonths()***
- ***addYears()***

получить 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
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);
```

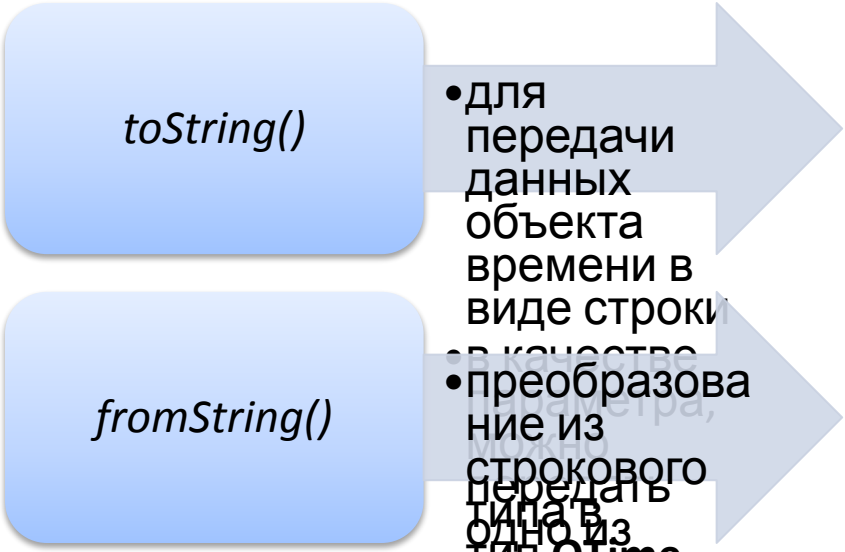
Или

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

ДАТА И ВРЕМЯ В QT: КЛАСС ВРЕМЕНИ

- *hour()* — возвращает положительное целое значение в диапазоне от 0 до 23;
- *minute()* — возвращает целое значение в диапазоне от 0 до 59;
- *second()* — возвращает целое значение в диапазоне от 0 до 59;
- *msec()* — возвращает положительное целое значение в диапазоне от 0 до 999, представляющее собой миллисекунды.

ДАТА И ВРЕМЯ В QT: КЛАСС ВРЕМЕНИ



- для передачи данных объекта времени в виде строки
- преобразование из строкового типа в тип QTime
- в первом параметре нужно передать одно из значений форматов.
- в качестве параметра можно передать одно из форматов времени или задать свой собственный формата

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 объект времени, добавив или отняв от существующего
- addMSecs() — Возвращает объект времени, добавив или отняв от существующего
- currentTime() — Возвращает текущее
- start() — Начинает отсчет времени
- elapsed() — Возвращает количество миллисекунд, прошло с момента начала отсчета

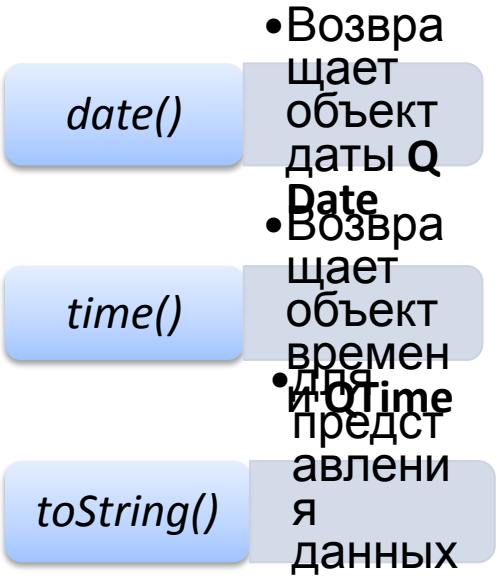
ДАТА И ВРЕМЯ В QT: КЛАСС ВРЕМЕНИ

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



ДАТА И ВРЕМЯ В QT: ТАЙМЕР

QTime

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

ДАТА И ВРЕМЯ В QT: ТАЙМЕР

Qappiication *processEvents()*

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

ДАТА И ВРЕМЯ В QT: ТАЙМЕР

ТАЙМЕР

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

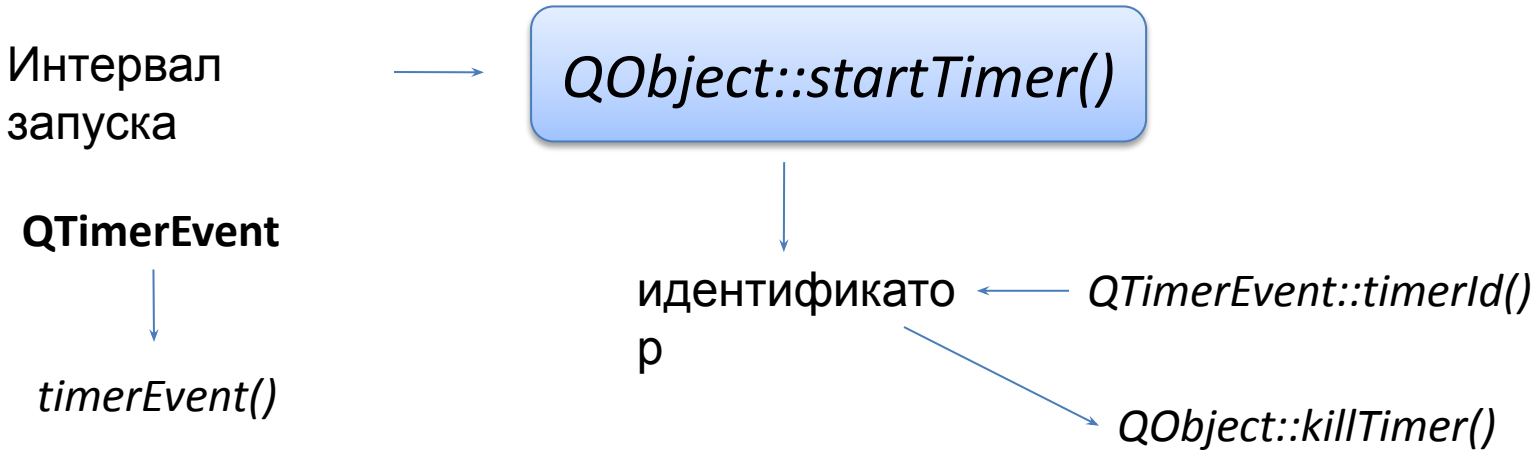
Интервал запуска (firing interval) – это период между событиями таймера.

сигнальное состояние

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
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 возможностям • для каждого потока имеющего Цикл сообщений (event loop). Для запуска цикла сообщений в потоке нужно
в мультипоточном программировании	и разных компьютеров

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(); }
```

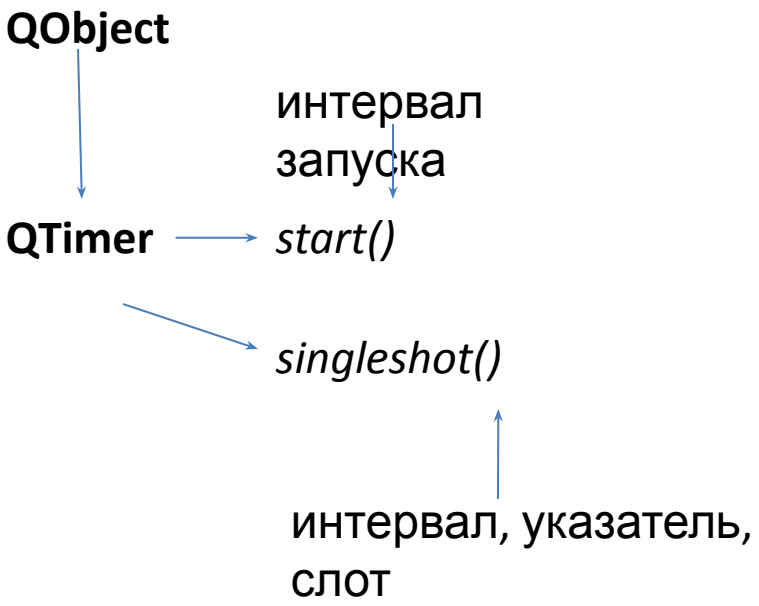
ДАТА И ВРЕМЯ В QT: ТАЙМЕР

```
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 : "");
    }

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

```
#include <QtGui>
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();
}

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()`

`start()`

`stop()`

`timerId()`

QObject::timerEvent().