

# Linguagens de Programação

Prof. Miguel Elias Mitre Campista

<http://www.gta.ufrj.br/~miguel>

# Parte V

Interface Gráfica usando Qt4

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## História do Qt

- Publicado para uso em maio de 1995
- Desenvolvido por Haavard Nord e Eirik Chambe-Eng
  - Empresa norueguesa Trolltech
- **Objetivo inicial:**
  - Desenvolvimento de interface gráfica para UNIX, Macintosh e Windows
  - Interface gráfica para um programa de base de dados em C++
    - **Portanto, o sistema para desenvolvimento da interface gráfica deveria ser orientado a objetos**
- **Por que o nome Qt?**
  - Q é bonito na fonte do emacs do Haavard e t é de toolkit...

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Primeiro Exemplo

```
#include <QApplication>
#include <QLabel>

int main (int argc, char *argv []) {
    QApplication app (argc, argv);
    QLabel *label = new QLabel ("Hello Qt!");
    label->show ();
    return app.exec ();
}
```

Linguagens de Programação – DEL-Poli/UFRJ

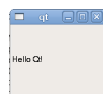
Prof. Miguel Campista

## Primeiro Exemplo

```
#include <QApplication>
#include <QLabel>

int main (int argc, char *argv []) {
    QApplication app (argc, argv);
    QLabel *label = new QLabel ("Hello Qt!");
    label->show ();
    return app.exec ();
}
```

```
shell$> qmake -project
shell$> qmake -makefile
shell$> make
```



Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Primeiras Classes

- **QApplication**
  - Classe para gerenciar recursos da aplicação
- **QLabel**
  - Classe que cria um *widget* para inserir string
    - **Widget → Window + gadget**
    - Elemento visual em uma interface de usuário
      - » Ex.: botões, menus, barras de rolagem e quadros

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Primeiras Classes

- A maioria das aplicações usam as classes `QMainWindow` ou `QDialog` como a janela da aplicação
  - Entretanto, o Qt pode usar qualquer widget como janela
    - O Exemplo 1 usa o widget `Label` como janela, exibida ao executar o método `show`
- Ações dos usuários criam eventos (ou mensagens)
  - Respondidos pelo programa
    - Ex. clique de mouse (evento de pressionar e/ou soltar)
- Programas convencionais diferem de aplicações com interfaces
  - Requerem entradas que são processadas e geram resultados

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Compilação

- `qmake -project`
  - Cria um arquivo de projeto (\*.pro) independente da plataforma
- `qmake arquivo.pro`
  - Cria um makefile específico para a plataforma do arquivo do projeto
- `make`
  - Constrói o programa

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Segundo Exemplo

```
#include <QApplication>
#include <QLabel>

int main (int argc, char *argv [] ) {
    QApplication app (argc, argv);
    QLabel *label = new QLabel ("<h2><i>Hello</i>" **<font color=red> Qt!</font></h2>*");
    label->show ();
    return app.exec ();
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Segundo Exemplo

```
#include <QApplication>
#include <QLabel>

int main (int argc, char *argv [] ) {
    QApplication app (argc, argv);
    QLabel *label = new QLabel ("<h2><i>Hello</i>" **<font color=red> Qt!</font></h2>*");
    label->show ();
    return app.exec ();
}
```

O rótulo pode ser formatado em estilo HTML

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Segundo Exemplo

```
#include <QApplication>
#include <QLabel>

int main (int argc, char *argv [] ) {
    QApplication app (argc, argv);
    QLabel *label = new QLabel ("<h2><i>Hello</i>" **<font color=red> Qt!</font></h2>*");
    label->show ();
    return app.exec ();
}
```



Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Criação de Conexões

- Base dos programas envolvendo o Qt
  - Widgets do Qt emitem sinais (**SIGNAL**)
    - Indicam que uma ação de usuário ou uma mudança de estado ocorreu
  - Sinais podem estar conectados a uma função (**SLOT**)
    - Quando o sinal é emitido, uma função específica é automaticamente chamada para tratar o sinal

A programação usando o Qt é baseada em eventos sinalizados a partir de **SIGNALS** e tratados através de **SLOTS**. Todas as classes que herdam de `QObject` e `QWidget` podem usar **SIGNALS** e **SLOTS**

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Criação de Conexões

- Signals e slots são tipos seguros
  - A assinatura do signal tem que corresponder à assinatura do slot receptor
    - Mesmos parâmetros na mesma sequência
- Signals e slots são programados para:
  - Compilador perceber tipos de assinaturas não correspondentes
  - Classe que implementar o signal não se preocupar com o slot que irá tratá-lo
    - Deve-se garantir apenas que para conectá-los, eles devem ter assinaturas correspondentes
  - Classe que implementar o slot não se preocupar se possui algum signal conectado

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Criação de Conexões

- Slots são como funções membro de C++
  - Podem ser virtuais e sobrecarregadas
  - Podem ser públicas, protegidas e privadas
  - Podem ser invocadas como qualquer outra função membro e seus parâmetros podem ser de qualquer tipo
- A diferença, porém...
  - é que os slots podem ser conectados a um signal, e são chamados automaticamente toda vez que um signal é emitido
  - Signals, por sua vez, são emitidos sempre que há uma mudança de estado

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Criação de Conexões

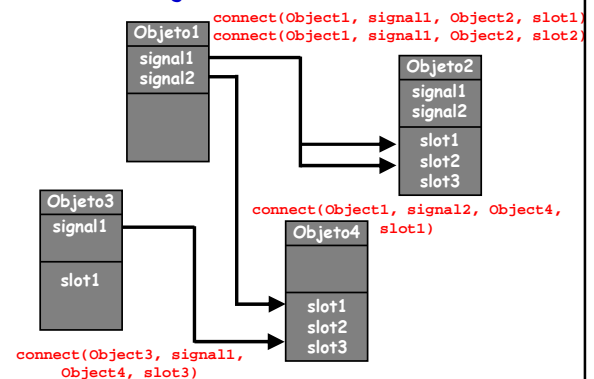
- sender e receiver são ponteiros para QObjects
- signal e slot são assinaturas de funções sem os nomes dos parâmetros
  - Macros SIGNAL () e SLOT () convertem seus argumentos em uma string
- Chamada da função:

```
connect (sender, SIGNAL (signal), receiver, SLOT (slot));
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Criação de Conexões



## Diferentes Tipos de Conexões

- Um signal conectado a diferentes slots:
  - connect (slider, SIGNAL (valueChanged (int)), spinBox, SLOT (setValue (int)));
  - connect (slider, SIGNAL (valueChanged (int)), this, SLOT (updateStatBar (int)));
    - Slots são invocados, um após o outro, em uma ordem não especificada
- Diferentes signals conectados ao mesmo slot:
  - connect (lcd, SIGNAL (overflow ()), this, SLOT (handleMathError ()));
  - connect (calculator, SIGNAL (divisionByZero ()), this, SLOT (handleMathError ()));
    - Quando um dos signals é emitido, o slot é chamado

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Diferentes Tipos de Conexões

- Um signal pode ser conectado a um outro signal:
  - connect (lineEdit, SIGNAL (textChanged (const QString &)), this, SIGNAL (const QString &));
    - Nesse caso, a emissão do primeiro signal implica na emissão do segundo
- Conexões podem ser removidas
  - disconnect (lcd, SIGNAL (overflow ()), this, SLOT (handleMathError ()));
    - A desconexão é raramente usada porque o Qt remove automaticamente conexões que envolvem objetos que já não existem mais

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Terceiro Exemplo

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

int main (int argc, char *argv [] ) {
    QApplication app (argc, argv);
    QPushButton *button = new QPushButton ("Quit");
    QObject::connect (button, SIGNAL (clicked ()), &app, SLOT (quit ()));
    button->show ();
    return app.exec ();
}
```

## Terceiro Exemplo

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

int main (int argc, char *argv [] ) {
    QApplication app (argc, argv);
    QPushButton *button = new QPushButton ("Quit");
    QObject::connect (button, SIGNAL (clicked ()), &app, SLOT (quit ()));
    button->show ();
    return app.exec ();
}
```



## Inserção de Widgets

- Uma widget pode ser:
  - Filha de uma outra widget
    - Widget de barra de rolagem é filha da widget de janela
  - Pai de uma outra widget
    - Widget de janela é pai da widget de barra de rolagem
- Widget de nível hierárquico superior
  - Não possui widget pai
- Subclasses de classes hierarquicamente superiores recebem como parâmetro uma QWidget \*
- Especifica a widget pai

## Quarto Exemplo

```
#include <QApplication>
#include <QBoxLayout>
#include <QSlider>
#include <QSpinBox>

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

    QWidget *window = new QWidget;
    window->setWindowTitle ("Enter Your Age");

    QSpinBox *spinBox = new QSpinBox;
    QSlider *slider = new QSlider (Qt::Horizontal);
    spinBox->setRange (0, 130);
    slider->setRange (0, 130);

    QObject::connect (spinBox, SIGNAL (valueChanged (int)), slider, SLOT (setValue (int)));
    QObject::connect (slider, SIGNAL (valueChanged (int)), spinBox, SLOT (setValue (int)));

    spinBox -> setValue (35);

    QBoxLayout *layout = new QBoxLayout;
    layout->addWidget (spinBox);
    layout->addWidget (slider);
    window->setLayout (layout);

    window->show ();

    return app.exec ();
}
```

## Quarto Exemplo

```
#include <QApplication>
#include <QBoxLayout>
#include <QSlider>
#include <QSpinBox>

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

    QWidget *window = new QWidget;
    window->setWindowTitle ("Enter Your Age");

    QSpinBox *spinBox = new QSpinBox;
    QSlider *slider = new QSlider (Qt::Horizontal);
    spinBox->setRange (0, 130);
    slider->setRange (0, 130);

    QObject::connect (spinBox, SIGNAL (valueChanged (int)), slider, SLOT (setValue (int)));
    QObject::connect (slider, SIGNAL (valueChanged (int)), spinBox, SLOT (setValue (int)));

    spinBox -> setValue (35);

    QBoxLayout *layout = new QBoxLayout;
    layout->addWidget (spinBox);
    layout->addWidget (slider);
    window->setLayout (layout);

    window->show ();

    return app.exec ();
}
```

Widget pai

Widgets filhas

## Quarto Exemplo

Não é necessário especificar a widget pai, pois o layout está sendo instalado em uma widget. Essa widget é considerada implicitamente como pai. Se fosse necessário, as chamadas seriam:

```
QSpinBox *spinBox = new QSpinBox (window);
QSlider *slider = new QSlider (Qt::Horizontal, window);
```

```
QWidget *window = new QWidget;
window->setWindowTitle ("Enter Your Age");

QSpinBox *spinBox = new QSpinBox;
QSlider *slider = new QSlider (Qt::Horizontal);
spinBox->setRange (0, 130);
slider->setRange (0, 130);

QObject::connect (spinBox, SIGNAL (valueChanged (int)), slider, SLOT (setValue (int)));
QObject::connect (slider, SIGNAL (valueChanged (int)), spinBox, SLOT (setValue (int)));

spinBox -> setValue (35);

QBoxLayout *layout = new QBoxLayout;
layout->addWidget (spinBox);
layout->addWidget (slider);
window->setLayout (layout);

window->show ();

return app.exec ();
```

## Quarto Exemplo

```
#include <QApplication>
#include <QBoxLayout>
```

A função `setValue` chama a `valueChanged` do `spinBox` que emite um sinal após ter seu valor alterado. Esse sinal é recebido pela função `setValue` do `slider`

```
QWidget *window = new QWidget;
window->setWindowTitle ("Enter Your Age");

QSpinBox *spinBox = new QSpinBox;
QSlider *slider = new QSlider (Qt::Horizontal);
spinBox->setRange (0, 130);
slider->setRange (0, 130);

QObject::connect (spinBox, SIGNAL (valueChanged (int)), slider, SLOT (setValue (int)));
QObject::connect (slider, SIGNAL (valueChanged (int)), spinBox, SLOT (setValue (int)));

spinBox -> setValue (35);

QHBoxLayout *layout = new QHBoxLayout;
layout->addWidget (spinBox);
layout->addWidget (slider);
```

Em seguida, o `valueChanged` do `slider` emite um sinal que não faz efeito no `spinBox` pois o valor já está como desejado

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Quarto Exemplo

```
#include <QApplication>
#include <QBoxLayout>
```

Retorna o controle do programa para a aplicação. O programa entra em um loop aguardando ações (eventos ou mensagens) realizadas pelos usuários

```
QWidget *window = new QWidget;
window->setWindowTitle ("Enter Your Age");

QSpinBox *spinBox = new QSpinBox;
QSlider *slider = new QSlider (Qt::Horizontal);
spinBox->setRange (0, 130);
slider->setRange (0, 130);

QObject::connect (spinBox, SIGNAL (valueChanged (int)), slider, SLOT (setValue (int)));
QObject::connect (slider, SIGNAL (valueChanged (int)), spinBox, SLOT (setValue (int)));

spinBox -> setValue (35);

QHBoxLayout *layout = new QHBoxLayout;
layout->addWidget (spinBox);
layout->addWidget (slider);
window->setLayout (layout);

window->show ();

return app.exec ();
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Quarto Exemplo

```
#include <QApplication>
#include <QBoxLayout>
#include <QSlider>
#include <QSpinBox>
```

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

    QWidget *window = new QWidget;
    window->setWindowTitle ("Enter Your Age");

    QSpinBox *spinBox = new QSpinBox;
    QSlider *slider = new QSlider (Qt::Horizontal);
    spinBox->setRange (0, 130);
    slider->setRange (0, 130);

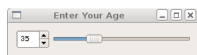
    QObject::connect (spinBox, SIGNAL (valueChanged (int)), slider, SLOT (setValue (int)));
    QObject::connect (slider, SIGNAL (valueChanged (int)), spinBox, SLOT (setValue (int)));

    spinBox -> setValue (35);

    QHBoxLayout *layout = new QHBoxLayout;
    layout->addWidget (spinBox);
    layout->addWidget (slider);
    window->setLayout (layout);

    window->show ();

    return app.exec ();
}
```

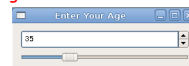


Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Gerente de Layout

- Qt possui três gerentes de layout
  - `QHBoxLayout`
    - Desenha widgets horizontalmente da esquerda para a direita
  - `QVBoxLayout`
    - Desenha widgets verticalmente de cima para baixo
  - `QGridLayout`
    - Desenha widgets em grade



Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Caixas de Diálogo

- Oferecem meios de diálogo entre usuários e aplicações
- Oferecem opções aos usuários
  - Usuários podem escolher ou definir as suas preferências

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Caixas de Diálogo

- Aplicações GUI consistem em:
  - Janela principal, barra de menu e barra de ferramentas
    - Além de dezenas de caixas de diálogo que complementam a janela principal
- Aplicações GUI podem...
  - Responder diretamente às escolhas dos usuários através das ações apropriadas

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Quinto Exemplo

```
#ifndef FINDIALOG_H
#define FINDIALOG_H

#include <QDialog>

class QCheckBox;
class QLabel;
class QLineEdit;
class QPushButton;

class FindDialog : public QDialog {
    /**
     * Macro necessária para todas as classes
     * que definem sinais e slots
     */
    Q_OBJECT

public:
    FindDialog (QWidget *parent = 0);

    // Palavra-chave signals é uma macro
signals:
    /**
     * Declara dois sinais que a caixa de diálogo emite
     * quando o usuário clica o botão Find
     */
    void findNext (const QString &str, Qt::CaseSensitivity cs);
    void findPrevious (const QString &str, Qt::CaseSensitivity cs);
};

#endif
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Quinto Exemplo

```
#ifndef FINDIALOG_H
#define FINDIALOG_H

#include <QDialog>

class QCheckBox;
class QLabel;
class QLineEdit;
class QPushButton;

class FindDialog : public QDialog {
    /**
     * Macro necessária para todas as classes
     * que definem sinais e slots
     */
    Q_OBJECT

public:
    FindDialog (QWidget *parent = 0);

    // Palavra-chave signals é uma macro
signals:
    /**
     * Declara
     * quando o
     */
    void findNext
    void findPrevious
};

#endif
```

Macro necessária para todas as classes que definem sinais e slots

Q\_OBJECT

FindDialog (QWidget \*parent = 0);

Construtor padrão com argumento padrão inicializando ponteiro com zero. Isso indica que a classe não possui pai

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Quinto Exemplo

```
#ifndef FINDIALOG_H
#define FINDIALOG_H

#include <QDialog>

class QCheckBox;
class QLabel;
class QLineEdit;
class QPushButton;

class FindDialog : public QDialog {
    /**
     * Macro necessária para todas as classes
     * que definem sinais e slots
     */
    Q_OBJECT

public:
    FindDialog (QWidget *parent = 0);

    // Palavra-chave signals é uma macro
signals:
    /**
     * Declara dois sinais que a caixa de diálogo emite
     * quando o usuário clica o botão Find
     */
    void findNext (const QString &str, Qt::CaseSensitivity cs);
    void findPrevious (const QString &str, Qt::CaseSensitivity cs);
};

#endif
```

A palavra-chave signals é uma macro que o pré-compilador converte em código em C++ antes da compilação

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Quinto Exemplo

```
private slots:
    void findClicked ();
    void enableFindButton (const QString &text);

private:
    QLabel *label;
    QLineEdit *lineEdit;
    QCheckBox *caseCheckBox, *backwardCheckBox;
    QPushButton *findButton, *closeButton;
};

#endif
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Quinto Exemplo

```
private slots:
    void findClicked ();
    void enableFindButton (const QString &text);

private:
    QLabel *label;
    QLineEdit *lineEdit;
    QCheckBox *caseCheckBox, *backwardCheckBox;
    QPushButton *findButton, *closeButton;
};

#endif
```

A palavra-chave slots também é uma macro que o pré-compilador converte em código em C++ antes da compilação

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Quinto Exemplo

```
private slots:
    void findClicked ();
    void enableFindButton (const QString &text);

private:
    QLabel *label;
    QLineEdit *lineEdit;
    QCheckBox *caseCheckBox, *backwardCheckBox;
    QPushButton *findButton, *closeButton;
};

#endif
```

A implementação das funções definidas como slots precisarão acessar as widgets filhas. Portanto, será necessário manter ponteiros para essas classes

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Quinto Exemplo

```
#include <QtGui>
#include "qt-ex05.h"

FindDialog::FindDialog (QWidget *parent) : QDialog (parent) {
    label = new QLabel (tr ("Find @what?"));
    lineEdit = new QLineEdit;
    label->setBuddy (lineEdit);

    caseCheckBox = new QCheckBox (tr ("Match @case"));
    backwardCheckBox = new QCheckBox (tr ("Search @backward"));

    findButton = new QPushButton (tr ("Find"));
    findButton->setDefault (true);
    findButton->setEnabled (false);

    closeButton = new QPushButton (tr ("Close"));

    connect (lineEdit, SIGNAL (textChanged (const QString &)), this, SLOT (enableFindButton (const QString &)));
    connect (findButton, SIGNAL (clicked ()), this, SLOT (findClicked ()));
    connect (closeButton, SIGNAL (clicked ()), this, SLOT (close ()));

    QHBoxLayout *topLeftLayout = new QHBoxLayout;
    topLeftLayout->addWidget (label);
    topLeftLayout->addWidget (lineEdit);

    QVBoxLayout *leftLayout = new QVBoxLayout;
    leftLayout->addLayout (topLeftLayout);
    leftLayout->addWidget (caseCheckBox);
    leftLayout->addWidget (backwardCheckBox);
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Quinto Exemplo

```
#include <QtGui>
#include "qt-ex05.h"

FindDialog::FindDialog (QWidget *parent) {
    label = new QLabel (tr ("Find @what?"));
    lineEdit = new QLineEdit;
    label->setBuddy (lineEdit);

    caseCheckBox = new QCheckBox (tr ("Match @case"));
    backwardCheckBox = new QCheckBox (tr ("Search @backward"));

    findButton = new QPushButton (tr ("Find"));
    findButton->setDefault (true);
    findButton->setEnabled (false);

    closeButton = new QPushButton (tr ("Close"));

    connect (lineEdit, SIGNAL (textChanged (const QString &)), this, SLOT (enableFindButton (const QString &)));
    connect (findButton, SIGNAL (clicked ()), this, SLOT (findClicked ()));
    connect (closeButton, SIGNAL (clicked ()), this, SLOT (close ()));

    QHBoxLayout *topLeftLayout = new QHBoxLayout;
    topLeftLayout->addWidget (label);
    topLeftLayout->addWidget (lineEdit);

    QVBoxLayout *leftLayout = new QVBoxLayout;
    leftLayout->addLayout (topLeftLayout);
    leftLayout->addWidget (caseCheckBox);
    leftLayout->addWidget (backwardCheckBox);
}
```

Biblioteca que inclui diferentes módulos que poderiam ser incluídas como bibliotecas individualmente

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Quinto Exemplo

```
#include <QtGui>
#include "qt-ex05.h"

FindDialog::FindDialog (QWidget *parent) {
    label = new QLabel (tr ("Find @what?"));
    lineEdit = new QLineEdit;
    label->setBuddy (lineEdit);

    caseCheckBox = new QCheckBox (tr ("Match @case"));
    backwardCheckBox = new QCheckBox (tr ("Search @backward"));

    findButton = new QPushButton (tr ("Find"));
    findButton->setDefault (true);
    findButton->setEnabled (false);

    closeButton = new QPushButton (tr ("Close"));

    connect (lineEdit, SIGNAL (textChanged (const QString &)), this, SLOT (enableFindButton (const QString &)));
    connect (findButton, SIGNAL (clicked ()), this, SLOT (findClicked ()));
    connect (closeButton, SIGNAL (clicked ()), this, SLOT (close ()));

    QHBoxLayout *topLeftLayout = new QHBoxLayout;
    topLeftLayout->addWidget (label);
    topLeftLayout->addWidget (lineEdit);

    QVBoxLayout *leftLayout = new QVBoxLayout;
    leftLayout->addLayout (topLeftLayout);
    leftLayout->addWidget (caseCheckBox);
    leftLayout->addWidget (backwardCheckBox);
}
```

Função tr: marca a string para possível tradução para outras línguas

& marca a letra usada para possibilitar acesso via teclado (nesse caso Alt+F é o atalho)

Função setBuddy marca o objeto quando o atalho está pressionado

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Quinto Exemplo

```
QVBoxLayout *rightLayout = new QVBoxLayout;
rightLayout->addWidget (findButton);
rightLayout->addWidget (closeButton);
rightLayout->addStretch ();

QHBoxLayout *mainLayout = new QHBoxLayout;
mainLayout->addLayout (leftLayout);
mainLayout->addLayout (rightLayout);
setLayout (mainLayout);

setWindowTitle (tr ("Find"));
setFixedHeight (sizeHint ().height ());
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Quinto Exemplo

```
QVBoxLayout *rightLayout = new QVBoxLayout;
rightLayout->addWidget (findButton);
rightLayout->addWidget (closeButton);
rightLayout->addStretch ();

QHBoxLayout *mainLayout = new QHBoxLayout;
mainLayout->addLayout (leftLayout);
mainLayout->addLayout (rightLayout);
setLayout (mainLayout);

setWindowTitle (tr ("Find"));
setFixedHeight (sizeHint ().height ());
}
```

Fixa a altura da janela

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Quinto Exemplo

```
void FindDialog::findClicked () {
    QString text = lineEdit->text ();
    Qt::CaseSensitivity cs = caseCheckBox->isChecked () ? Qt::CaseSensitive : Qt::CaseInsensitive;
    if (backwardCheckBox->isChecked ())
        emit findPrevious (text, cs);
    else
        emit findNext (text, cs);
}

void FindDialog::enableFindButton (const QString &text) {
    findButton->setEnabled (!text.isEmpty ());
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Quinto Exemplo

Mais macro...

```
void FindDialog::findClicked () {
    QString text = lineEdit->text ();
    Qt::CaseSensitivity cs = caseCheckBox->isChecked () ? Qt::CaseSensitive : Qt::CaseInsensitive;
    if (backgroundCheckBox->isChecked ())
        emit findPrevious (text, cs);
    else
        emit findNext (text, cs);
}

void FindDialog::enableFindButton (const QString &text) {
    findButton->setEnabled (!text.isEmpty ());
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Quinto Exemplo

```
#include <QApplication>
#include "qt-ex05.h"

int main (int argc, char *argv []) {
    QApplication app (argc, argv);
    FindDialog *dialog = new FindDialog;
    dialog->show ();
    return app.exec ();
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Quinto Exemplo

```
#include <QApplication>
#include "qt-ex05.h"

int main (int argc, char *argv []) {
    QApplication app (argc, argv);
    FindDialog *dialog = new FindDialog;
    dialog->show ();
    return app.exec ();
}
```



Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Criação de Caixas de Diálogo

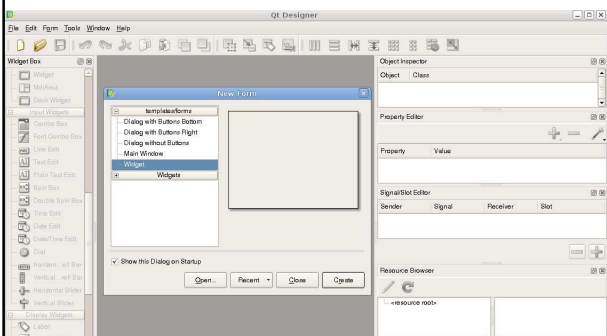
- Pode ser feito através da escrita de programas em C++
- Pode ser feito utilizando ferramentas visuais
  - Qt4 Designer

```
shell$> designer-qt4
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Qt4 Designer

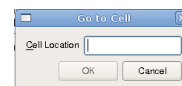


Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Sexto Exemplo

- Como criar uma janela como a janela abaixo com o Qt4 designer?



Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista



## Sexto Exemplo

- Escolher um template da lista
  - Template widget
- Criação das widgets filhas
  - Arrastar da caixa de widgets: dois Push Buttons, um Horizontal Spacer, um Line Edit e um Label
    - O Horizontal Spacer é invisível na forma final

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Sexto Exemplo

- Clique no text label
  - Certifique-se que a propriedade objectName é "label" e mude a propriedade text para "&Cell Location"
- Clique no line editor
  - Certifique-se que a propriedade objectName é "lineEdit"
- Clique no primeiro button
  - Mude a propriedade objectName para "okButton", coloque a propriedade enabled para "false", a text property para "OK" e a propriedade default para "true"

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Sexto Exemplo

- Clique no segundo button
  - Mude a propriedade objectName para "cancelButton" e a text property para "Cancel"
- Clique no background da janela para selecionar a própria janela
  - Mude a propriedade objectName para "GoToCellDialog" e a propriedade windowTitle para "Go to Cell"
- Mude "&Cell Location" para "Cell Location"
  - Clique em Edit→Edit Buddies. Em seguida, clique na label e a arraste até a line editor

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Sexto Exemplo

- Volte para o modo de edição
  - Vá em Edit→Edit Widgets
- Clique no rótulo "Cell Location" e após pressionar o shift, selecione o line editor
  - Clique em Form→Lay Out Horizontally
- Clique no spacer e então pressione o shift enquanto seleciona os dois botões
  - Clique em Form→Lay Out Horizontally

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Sexto Exemplo

- Clique no background para remover qualquer seleção
  - Clique em Form→Lay Out Vertically
- Clique em Form→Adjust Size
  - Redimensiona a janela para o tamanho desejado
- Salve a caixa de diálogo como gotoCellDialog.ui em um diretório chamado gotoCell

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Sexto Exemplo

- Em seguida, crie a função principal fora do Qt4 designer:

```
#include <QApplication>
#include <QDialog>

#include "ui_gotoCellDialog.h"

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

    Ui::GotoCellDialog ui;
    QDialog *dialog = new QDialog;
    ui.setupUi (dialog);
    dialog->show ();

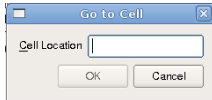
    return app.exec ();
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Sexto Exemplo

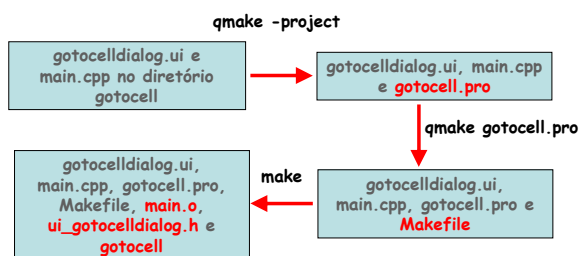
```
shell$> qmake -project
shell$> qmake arquivo.pro
shell$> make
shell$> programa
```



## Compilador da Interface do Usuário

- Qt4 designer gera arquivo da interface do usuário
  - Arquivo \*.ui
- Compilador qmake detecta o arquivo \*.ui
  - Além disso, qmake cria um Makefile apropriado para invocar o Compilador da Interface do Usuário (uic)
    - uic → *User Interface Compiler*
  - O uic converte o arquivo \*.ui em C++
    - Coloca o resultado em um arquivo \*.h
      - Contém a definição da classe relacionada com a interface criada
      - Contém a função `setupUi` que inicializa a interface

## Voltando ao Sexto Exemplo...



ui\_gotocelldialog.h possui definição da classe `Ui::GoToCellDialog` e da função `setupUi`

## Voltando ao Sexto Exemplo...

- A classe `Ui::GoToCellDialog` tem a seguinte forma...

```
class Ui::GoToCellDialog {
public:
    QLabel *label;
    QLineEdit *lineEdit;
    QSpacerItem *spacerItem;
    QPushButton *okButton;
    QPushButton *cancelButton;
    ...
    void setupUi (QWidget *widget) {...}
};
```

Classe não possui classe base. Passa-se, então, um objeto da classe `QDialog` para a função `setupUi` como na *main* do Sexto Exemplo...

## Voltando ao Sexto Exemplo...

- Até o momento, a interface existe mas...
  - Botões OK e Cancel não fazem nada
  - Editor de texto aceita qualquer coisa

➔ Programação de uma nova classe para implementar essas funcionalidades...

- Classe herda de `QDialog` e `Ui::GoToCellDialog`
  - Por padrão, classe filha se chama `GoToCellDialog`
    - Nome igual ao da classe criada pelo uic, mas **sem** o prefixo `Ui::`

## Voltando ao Sexto Exemplo...

```
#ifndef GOTOCELLDIALOG_H
#define GOTOCELLDIALOG_H

#include <QDialog>
#include "ui_gotocelldialog.h"

class GoToCellDialog : public QDialog, public Ui::GoToCellDialog {
    Q_OBJECT
public:
    GoToCellDialog (QWidget *parent = 0);
private slots:
    void on_lineEdit_textChanged ();
};

#endif
```

## Voltando ao Sexto Exemplo...

```
#include <QtGui>
#include "gotocelldialog.h"

GoToCellDialog::GoToCellDialog (QWidget *parent) : QDialog (parent) {
    setupUi (this);
}

QRegExp regExp ("[A-Za-z][1-9][0-9]{0,2}");
lineEdit->setValidator (new QRegExpValidator (regExp, this));

connect (okButton, SIGNAL (clicked ()), this, SLOT (accept ()));
connect (cancelButton, SIGNAL (clicked ()), this, SLOT (reject ()));

void GoToCellDialog::on_lineEdit_textChanged () {
    okButton->setEnabled (lineEdit->hasAcceptableInput ());
}
```

## Voltando ao Sexto Exemplo...

```
#include <QtGui>
#include "gotocelldialog.h"

GoToCellDialog::GoToCellDialog (QWidget *parent) : QDialog (parent) {
    setupUi (this);
}

QRegExp regExp ("[A-Za-z][1-9][0-9]{0,2}");
lineEdit->setValidator (new QRegExpValidator (regExp, this));

connect (okButton, SIGNAL (clicked ()), this, SLOT (accept ()));
connect (cancelButton, SIGNAL (clicked ()), this, SLOT (reject ()));

void GoToCellDialog::on_lineEdit_textChanged () {
    okButton->setEnabled (lineEdit->hasAcceptableInput ());
}
```

Chama a função `setupUi` herdada, definida na classe `Ui::GoToCellDialog`, para criar a interface

## Voltando ao Sexto Exemplo...

```
#include <QtGui>
#include "gotocelldialog.h"

GoToCellDialog::GoToCellDialog (QWidget *parent) : QDialog (parent) {
    setupUi (this);
}

QRegExp regExp ("[A-Za-z][1-9][0-9]{0,2}");
lineEdit->setValidator (new QRegExpValidator (regExp, this));

connect (okButton, SIGNAL (clicked ()), this, SLOT (accept ()));
connect (cancelButton, SIGNAL (clicked ()), this, SLOT (reject ()));

void GoToCellDialog::on_lineEdit_textChanged () {
    okButton->setEnabled (lineEdit->hasAcceptableInput ());
}
```

Função `setupUi` conecta automaticamente qualquer slot que segue o padrão `on_objectName_signalName()` com o sinal do `objectName` definido no `signalName()` correspondente:

```
connect (lineEdit, SIGNAL (textChanged (const QString &)), this, SLOT (on_lineEdit_textChanged ());
```

## Voltando ao Sexto Exemplo...

```
#include <QtGui>
#include "gotocelldialog.h"

GoToCellDialog::GoToCellDialog (QWidget *parent) : QDialog (parent) {
    setupUi (this);
}

QRegExp regExp ("[A-Za-z][1-9][0-9]{0,2}");
lineEdit->setValidator (new QRegExpValidator (regExp, this));

connect (okButton, SIGNAL (clicked ()), this, SLOT (accept ()));
connect (cancelButton, SIGNAL (clicked ()), this, SLOT (reject ()));

void GoToCellDialog::on_lineEdit_textChanged () {
    okButton->setEnabled (lineEdit->hasAcceptableInput ());
}
```

Usa um validador para restringir as possíveis entradas. Qt possui três validadores disponíveis (`QIntValidator`, `QDoubleValidator` e `QRegExpValidator`). O `QRegExpValidator` usa a expressão regular "[A-Za-z][1-9][0-9]{0,2}" que permite uma letra maiúscula ou minúscula seguida de um dígito no intervalo [1,9] e zero, um ou dois ({0,2}) dígitos no intervalo [0-9].

## Voltando ao Sexto Exemplo...

```
#include <QtGui>
#include "gotocelldialog.h"

GoToCellDialog::GoToCellDialog (QWidget *parent) : QDialog (parent) {
    setupUi (this);
    lineEdit->setValidator (new QRegExpValidator (regExp, this));
}

connect (okButton, SIGNAL (clicked ()), this, SLOT (accept ()));
connect (cancelButton, SIGNAL (clicked ()), this, SLOT (reject ()));

void GoToCellDialog::on_lineEdit_textChanged () {
    okButton->setEnabled (lineEdit->hasAcceptableInput ());
}
```

Ao passar o ponteiro `GoToCellDialog` para o construtor de `QRegExpValidator`, o validador se torna filho do objeto `GoToCellDialog`

## Voltando ao Sexto Exemplo...

```
#include <QtGui>
#include "gotocelldialog.h"

GoToCellDialog::GoToCellDialog (QWidget *parent) : QDialog (parent) {
    setupUi (this);
}

QRegExp regExp ("[A-Za-z][1-9][0-9]{0,2}");
lineEdit->setValidator (new QRegExpValidator (regExp, this));

connect (okButton, SIGNAL (clicked ()), this, SLOT (accept ()));
connect (cancelButton, SIGNAL (clicked ()), this, SLOT (reject ()));

void GoToCellDialog::on_lineEdit_textChanged () {
    okButton->setEnabled (lineEdit->hasAcceptableInput ());
}
```

Se o botão `OK` for clicado, o atributo `QDialog::Accepted` recebe valor 1. Se o botão `Cancel` for clicado, o atributo `QDialog::Accepted` recebe valor 0. Resultado pode ser usado para saber se o usuário clicou `OK` ou não

## Voltando ao Sexto Exemplo...

```
#include <QtGui>
#include "gotocelldialog.h"

GoToCellDialog::GoToCellDialog(QWidget *parent) : QDialog(parent) {
    setUpUi (this);

    QRegExp regExp ("[A-Za-z]{1-9}[0-9]{0,2}");
    QLineEdit::setValidator (new QRegExpValidator (regExp, this));

    connect (okButton, SIGNAL (clicked ()), this, SLOT (accept ()));
    connect (cancelButton, SIGNAL (clicked ()), this, SLOT (reject ()));
}

void GoToCellDialog::on_lineEdit_textChanged () {
    okButton->setEnabled (!lineEdit->hasAcceptableInput ());
}
```

Ao alterar o texto, o botão OK se torna habilitado se a entrada estiver de acordo com os requisitos definidos como aceitáveis

## Relacionamento Pai-Filho

- Implementado na classe QObject
- Pai adiciona o filho em uma lista de filhos
  - Quando o pai é deletado, ele percorre a lista de filhos deletando cada um deles
    - Os filhos deletam seus filhos e assim por diante...
  - Quando o filho é deletado antes do pai, ele é simplesmente removido da lista do pai
- Simplifica o gerenciamento da memória
  - Reduz a possibilidade de vazamento de memória
    - Só é necessário deletar objetos criados dinamicamente que não possuem pai
- Filhos são exibidos dentro da área do pai
  - Remover o pai significa também remover o filho da tela

## Reescrevendo a Função Principal do Sexto Exemplo

```
#include <QApplication>
#include <QDialog>

#include "gotocelldialog.h"

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

    GoToCellDialog *dialog = new GoToCellDialog;
    dialog->show ();

    return app.exec ();
}
```

## Reescrevendo a Função Principal do Sexto Exemplo

```
#include <QApplication>
#include <QDialog>

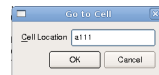
#include "gotocelldialog.h"

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

    GoToCellDialog *dialog = new GoToCellDialog;
    dialog->show ();

    return app.exec ();
}
```

```
shell$> qmake -project
shell$> qmake -makefile
shell$> make
shell$> programa
```



## Alterando o Sexto Exemplo...

- Uso do QDialogButtonBox
  - Apresentam os botões em um formato correto independente do sistema operacional
- Alterações (Usando o Qt4 designer):
  - Clique na janela para remover o layout
    - Vá em Form → Break Layout
  - Remova as widget filhas
    - Botões OK e Cancel, o espaço horizontal e layout horizontal
  - Arraste o Button Box para a janela
  - Clique na janela
    - Vá em Form → Lay Out Vertically

## Alterando o Sexto Exemplo...

- Como duas widgets foram removidas e uma foi inserida...
  - É necessário mudar o código da classe GoToCellDialog
    - As alterações são feitas no arquivo gotocelldialog.cpp

## Alterando o Sexto Exemplo...

```
#include <QtGui>
#include "gotocelldialog.h"

GoToCellDialog::GoToCellDialog (QWidget *parent) : QDialog (parent) {
    setupUi (this);

    buttonBox->button (QDialogButtonBox::Ok)->setEnabled (false);

    QRegExp regExp ("^[A-Za-z][1-9][0-9]{0,2}");
    lineEdit->setValidator (new QRegExpValidator (regExp, this));
    connect (buttonBox, SIGNAL (accepted ()), this, SLOT (accept ()));
    connect (buttonBox, SIGNAL (rejected ()), this, SLOT (reject ()));
}

void GoToCellDialog::on_lineEdit_textChanged () {
    buttonBox->button (QDialogButtonBox::Ok)->setEnabled (lineEdit->hasAcceptableInput ());
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Alterando o Sexto Exemplo...

```
#include <QtGui>
#include "gotocelldialog.h"

GoToCellDialog::GoToCellDialog (QWidget *parent) : QDialog (parent) {
    setupUi (this);

    buttonBox->button (QDialogButtonBox::Ok)->setEnabled (false);

    QRegExp regExp ("^[A-Za-z][1-9][0-9]{0,2}");
    lineEdit->setValidator (new QRegExpValidator (regExp, this));
    connect (buttonBox, SIGNAL (accepted ()), this, SLOT (accept ()));
    connect (buttonBox, SIGNAL (rejected ()), this, SLOT (reject ()));
}

void GoToCellDialog::on_lineEdit_textChanged () {
    buttonBox->button (QDialogButtonBox::Ok)->setEnabled (lineEdit->hasAcceptableInput ());
}
```

Botão Ok é desabilitado inicialmente. Esse procedimento não pode ser feito no Qt4 Designer com o Button Box

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Alterando o Sexto Exemplo...

```
#include <QtGui>
#include "gotocelldialog.h"

GoToCellDialog::GoToCellDialog (QWidget *parent) : QDialog (parent) {
    setupUi (this);

    buttonBox->button (QDialogButtonBox::Ok)->setEnabled (false);

    QRegExp regExp ("^[A-Za-z][1-9][0-9]{0,2}");
    lineEdit->setValidator (new QRegExpValidator (regExp, this));
    connect (buttonBox, SIGNAL (accepted ()), this, SLOT (accept ()));
    connect (buttonBox, SIGNAL (rejected ()), this, SLOT (reject ()));
}

void GoToCellDialog::on_lineEdit_textChanged () {
    buttonBox->button (QDialogButtonBox::Ok)->setEnabled (lineEdit->hasAcceptableInput ());
}
```

Objeto buttonBox, nome padrão do Qt4 designer, é usado para refazer as conexões

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Alterando o Sexto Exemplo...

```
#include <QtGui>
#include "gotocelldialog.h"

GoToCellDialog::GoToCellDialog (QWidget *parent) : QDialog (parent) {
    setupUi (this);

    buttonBox->button (QDialogButtonBox::Ok)->setEnabled (false);

    QRegExp regExp ("^[A-Za-z][1-9][0-9]{0,2}");
    lineEdit->setValidator (new QRegExpValidator (regExp, this));
    connect (buttonBox, SIGNAL (accepted ()), this, SLOT (accept ()));
    connect (buttonBox, SIGNAL (rejected ()), this, SLOT (reject ()));
}

void GoToCellDialog::on_lineEdit_textChanged () {
    buttonBox->button (QDialogButtonBox::Ok)->setEnabled (lineEdit->hasAcceptableInput ());
}
```

É necessário definir qual dos dois botões, Ok ou Cancel, se quer habilitar. Na primeira versão os botões eram objetos separados

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Alterando o Sexto Exemplo...

```
#include <QtGui>
#include "gotocelldialog.h"

GoToCellDialog::GoToCellDialog (QWidget *parent) : QDialog (parent) {
    setupUi (this);

    buttonBox->button (QDialogButtonBox::Ok)->setEnabled (false);

    QRegExp regExp ("^[A-Za-z][1-9][0-9]{0,2}");
    lineEdit->setValidator (new QRegExpValidator (regExp, this));
    connect (buttonBox, SIGNAL (accepted ()), this, SLOT (accept ()));
    connect (buttonBox, SIGNAL (rejected ()), this, SLOT (reject ()));
}

void GoToCellDialog::on_lineEdit_textChanged () {
    buttonBox->button (QDialogButtonBox::Ok)->setEnabled (lineEdit->hasAcceptableInput ());
}
```

```
shell$> qmake -project
shell$> qmake arquivo.pro
shell$> make
shell$> programa
```



Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Leitura Recomendada

- Jasmin Blanchette e Mark Summerfield, "C++ GUI Programming with Qt4", 2nd edition, Editora Prentice Hall, 2008
- Trolltech, "Qt Reference Documentation (Open Source Edition)", 2005
  - Disponível em : <http://doc.qt.nokia.com/4.0/>

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista