6ª Lista de Exercícios
Engenharia de Software I

Qt: uma biblioteca multi-plataforma para o desenvolvimento interfaces gráficas com o usuário

- Desenvolvimento de Interfaces Gráficas com o Usuário (GUI) –

Prof. Tiago Garcia de Senna Carneiro
Departamento de Computação
Universidade Federal de Ouro Preto

Nesta lista de exercícios você deverá:

1. **Responder ao questionário que compõe esta lista de exercícios.**
2. **Implementar todos os exercícios que a acompanham.**

Questões teóricas

1) A maioria das bibliotecas para desenvolvimento de GUI utiliza o mecanismo de "callback functions" para tratar as interações do usuário com os *widgets* presentes na interface.
    a. Explique como é o funcionamento desse mecanismo.
    b. Aponte suas principais desvantagens.

2) A biblioteca Qt implementa o mecanismo "Signals & Slots" para associar interações realizados pelo usuário com código que deve trata-las. Por exemplo, o mecanismo pode ser utilizado para associar o código que aborta a aplicação ao botão "sair" que aparece na tela.
    a. O que é um *signal*?
    b. O que é um *slot*?
    c. Explique como esse mecanismo funciona.
    d. Aponte as vantagens desse mecanismo sobre o mecanismo de "callback".

3) *Signals* são emitidos por um objeto para sinalizar que seu estado interno (atributos) foi alterado. Responda:
    a. Que tipos de objetos podem emitir sinais definidos em uma classe?
    b. A quantos *slots* um único sinal pode ser associados?
    c. Quando um *signal* é emitido, o *slot* a ele conectado é imediatamente executado. O que acontece quando são utilizados conexões com fila (*queued conections*)?

4) *Slots* podem disparados por qualquer *signal*. Este fato, de alguma maneira, fere os conceitos da orientação por objetos? Como?

5) As palavras reservadas *emmit*, *slot* e *signal* não pertecem à linguagem C++. Entretanto elas são utilizadas para a que o programador possa definir seus próprios *widgets* cujos *signals* e *slots* são por ele projetados e definidos. Veja a trecho de código abaixo, onde um *widget* chamado "MyClass" é definido e responda. Quais são os passos para se compilar essa classe?

```
class MyClass : public QObject
 {
     Q_OBJECT

 public:
     MyClass(QObject *parent = 0);
     ~MyClass();

 signals:
     void mySignal();

 public slots:
     void mySlot();
 };
```

Questões práticas

Acompanha esta lista um diretório de códigos contenho o arquivo "mainQt2.cpp". Ele contém vários aplicativos que você pode executar alternando a diretiva de pré-compilação que inicia o arquivo entre: "#define DEMO1", #define DEMO2", #define DEMO3" e assim por diante. Cada aplicativo corresponde ao tutorial de mesmo número fornecido junto com a biblioteca Qt. Você deve resolver as seguintes questões:

1) Com a aplicação DEMO1 em execução, tente alterar o tamanho da janela. Como o botão se comporta?

2) Execute a aplicação DEMO2 e entenda seu funcionamento. A *widget* QPushButton herda vários *signals* da *widget* QAbstractButton, entre eles "*pressed()" e "clicked()"*. Altere o *signal* conectado ao *slot "quit()"* da aplicação entre esses dois signals. Explique como o comportamento da aplicação mudou?

3) Com a aplicação DEMO3 em execução, tente alterar o tamanho da janela. Como o botão se comporta se comparado ao funcionamento da aplicação DEMO1?

4) Com a aplicação DEMO4 em execução, tente alterar o tamanho da janela. O que aconteceu? Por quê?

5) Na função "main()" da aplicação DEMO4, tente criar outro objeto da classe "MyWidget" e exibi-lo. O que aconteceu?

6) Execute a aplicação DEMO5 e entenda seu funcionamento. Tente mudar o número de dígitos utilizados pelo *Visor LCD* (originalmente igual a 2) e a largura da faixa utilizada pela *Barra de Rolagem* (originalmente definida entre 0 e 99). A interface pode deixar de funcionar? Quando?

7) Na aplicação DEMO5, substitua a *widget* "QSlider" pela *widget* "QSpinBox". Qual a vantagem dessa abordagem? Quais alterações você precisou fazer no código?

8) Ainda na aplicação DEMO5, faça com que a aplicação seja finalizada quando o visor LCD provocar sofrer um *overflow*. Quais alterações você precisou fazer no código?

9) Altere o aplicação DEMO6 para que cada Barra de Rolagem (*slider*) seja iniciada com um valor aleatório.

10) Sobre a aplicação DEMO7:
    a. Você não conseguirá compilar essa aplicação utilizando diretamente o ambiente de desenvolvimento Dev-C++, porque ela utiliza o meta objeto MyLCD que é definido no arquivo "MyLCD.h" através do uso das palavras reservadas emit, slot e signal.
    b. Perceba que o método ou *signal* "MyLCD::valueChanged(int)" foi declarado mas não foi definido. Este fato certamente irá gerar um erro de ligação no compilador C++.
    c. Na linha de comando (cmd), vá para o diretório onde estão os arquivos desse projeto. Lembre-se de atualizar a diretiva de pré-compilação para "#define DEMO7" no arquivo "mainQt2.cpp". Depois, execute o comando "moc MyLCD.h" e tente entender o código gerado pelo compilador de meta-objetos. Repare que o método "MyLCD::valueChanged(int)" foi gerado automaticamente por esse compilador.
    d. Agora, você precisará utilizar o compilador C++ na linha de comando, da mesma maneira que faria no Linux:
        i. Você deve utilizar o aplicativo "qmake" para gerar um arquivo de projeto, cuja extensão é ".pro". Este arquivo lista para o compilador C++ todos os arquivos "*.h" e "*.cpp" que compõe seu projeto. Digite o comando: "qmake –project". Veja se o arquivo de extesão "*.pro" foi gerado, você pode abri-lo usando o "bloco de notas".
        ii. Então, você deve executar o comando "qmake" para que o "makefile" do seu projeto seja gerado. O "makefile" contém instruções para o compilador e ligador C++. Ele informa quais arquivos "*.h" das bibliotecas em uso devem ser compilados e sua localização. O "makefile" também informa quais arquivos objetos ("*.lib" ou "*.dll") precisam ser ligados à sua aplicação. Digite: "qmake". Veja se o arquivo de nome "Makefile" foi gerado. Você pode abri-lo com o bloco de notas.
        iii. Finalmente, execute o "make" para compilar seu programa. Digite: "make". Seu aplicativo será gerado dentro do subdiretório "debug". Execute o aplicativo.

Referências

1. *Qt 4.2 Whitepaper* em (http://www.trolltech.com/products/qt/learnmore/whitepapers/whitepapers)

2. Signal and Slots em (C:/Qt/4.2.2/doc/html/signalsandslots.html)

3. Tutoriais para Qt em (C:/Qt/4.2.2/doc/html/examples.html#qt-tutorial)

4. Usando o compilador de meta objetos "moc" em (C:/Qt/4.2.2/doc/html/moc.html)