A practical guide to C++

Janis Fehr
fehr@informatik.uni-freiburg.de

SommerCampus2004

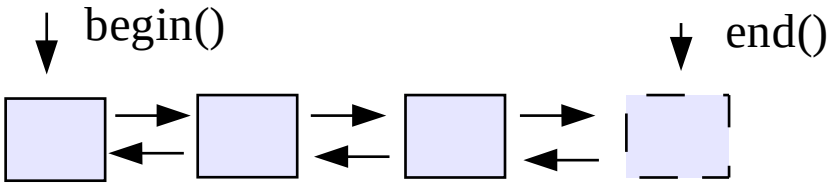
Kurstag 4: Agenda

- Wiederholung Kurstag 3
- Die Qt Library
    - der Qt Präprozessor
    - Signals und Slots
    - Widgets
- Projekt

Wiederholung :Standard Container

Vector

Dynamischer Array:



Einfügen: z.B.: `myVector.push_back(item)`

Element Adressierung: `myVector[i]`

Algorithmen: `myVector.sort()`

Größe ändern: `myVector.resize(newsize)`

Weitere Container Klassen:

List, Stack, Map, Queue, Set ... alles was man aus Info 2 kennt

Wiederholung: Funktions Templates

```
template<typename DataType>
DataType min(DataType a, DataType b)
{
   if(a<b) return a;
   return b;
}

...

int main()
{
   int g,j;
   int l = min<int>(a,b);
   ...
}
```

Compiler

```
int min(int a, int b)
{
   if(a<b) return a;
   return b;
}
```

Die Qt - Library

- Umfangreiche, Platform unabhängige Lib
    - Gui
    - Event orientierte Programmierung
    - Netzwerk
    - Bilddarstellung
- www.trolltech.com
- Freie Version unter GPL
    - kommerzielle Versionen
- Ressourcen: Qt Reference

''Hello Qt !''

```
#include <qapplication.h>
#include <qpushbutton.h>

int main(int argc, char** argv)
{
   QApplication a(argc, argv);
   QPushButton hello(''Hello Qt!,0);
   hello.resize(100,3);

   a.setMainWidget(&hello);
   hello.show;

   return a.exec();
}
```

Include Qt Lib Elemente

Instanziere Qt Basisklasse

Instanziere Button

Setze Hauptwidget

Zeige Widget

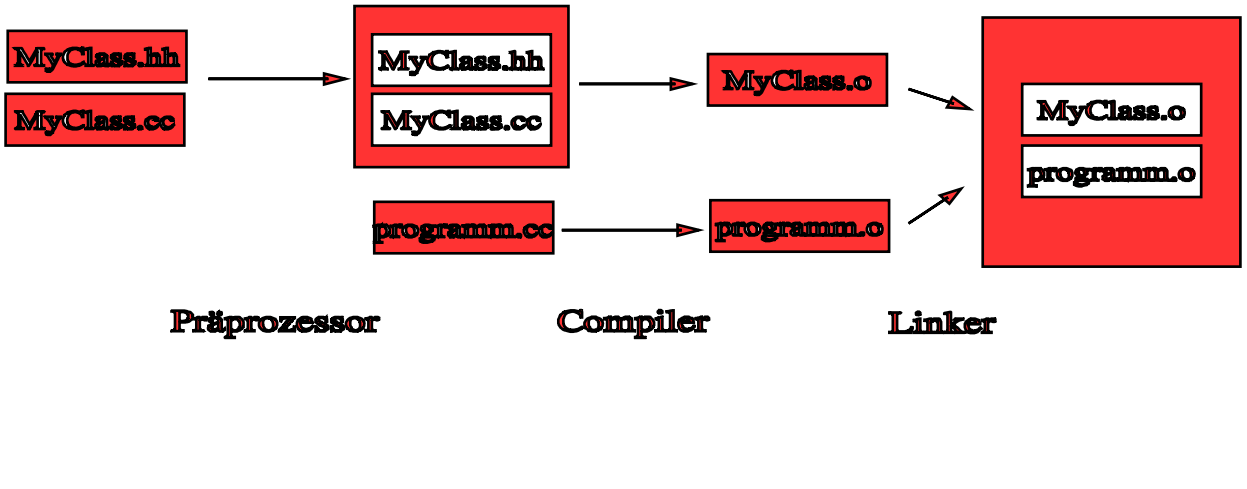
Widget Rückgabe werte

Der Qt Präprozessor

Da Qt C++ um Funktionen erweitert, die über die "normalen" Fähigkeiten von C++ hinaus gehen, reicht es nicht einfach nur die QtLib zu includen.

Lösung: Qt Präprozessor MOC

Zur Erinnerung:



Der MOC ersetz Teile der Qt Schlüsselwörter durch Code

Makefile generierung mit qmake

- Erzeuge Automatisch das richtige Makefile
- qmake benötigt nur wenige Parameter
    - hello_qt.pro
- `qmake -o Makefile hello_qt.pro`
- `make`

hello_qt.pro:

```
SOURCES = hello_qt.cc
HEADERS =
CONFIG += qt warn_on release
```

Signals und Slots

- Event basierte Programmierung
- Signals:
  - werden bei einem Event ausgesendet
  - können Parameter mit sich führen
- Slots
  - fangen Signals auf
  - "normale" Methoden

```
class A
{
        public:
                ...
        signals:
                void mysignal(int a);
        ...
```

```
class B
{
        public:
                ...
        public slots:
                void myslot(int a);
        ...
```

Hauptprogramm:

```
QObject::connect(A,SIGNAL(mysignal(int),B,SLOT(myslot(int));
```

Signals und Slots von Qt Objecten

Natürlich haben die Qt Objekt, wie z.B.: der Qbutton standard Signals und Slots:

- QButton::clicked()
- Qslider::valueChanged()
- ...

Siehe Qt Referenz ...

Bilder mit der Qt darstellen

- Klasse QImage (Pixel basierte Bilder)
- Definiere Farbwerte mit QRgb
- ...

Siehe qt_image ...

Project:

- Jetzt fügen wir die Teile zusammen:
  - lest beim starten des Programms den Bildnamen und den Text ein
    - Siehe Parameterübergabe, 1. Kurstag
  - Wandelt die ASCII Zeichen des Textes in Binärkode um
    - Der Container Map ist dafür sehr gut geeignet
  - Streamt die Bilddaten vom File in ein Bild-Objekt
    - Bild-Objekt vom 3. Kurstag
  - Schreibt den Binärkode an die LSB der Pixel
    - Pixel Klasse vom 2. Kurstag
  - Speichert das Bild ab
  - Bonus1: lasst Euch das Bild mit Qt vor und nach dem Einfügen anzeigen
  - Bonus2: man will die versteckten Daten auch wieder bekommen :-)