Praxis des Programmierens

SS 2009

Di 14-16, Mi, Do 10-12 im RTK
(Vorlesung/Übung gemischt)

Dipl.-Math. Steffen Brasch — sbrasch AT uni-greifswald.de
Prof. Dr. Stefan Funke — stefan.funke AT uni-greifswald.de
Apotheker Stephan Struckmann — struckma AT uni-greifswald.de

Vorlesungsinfos/Slides:
http://www.math-inf...de/informatik/COURSES/PraxProg09/PraxProg09.html

Inhalte

- Anstoß, das Programmieren zu lernen
- ... und zwar nicht nur anhand kleinerer Programm-fragmente sondern anhand eines größeren Projektes
- Sie lernen Linux und die GNU g++ Standardwerkzeuge kennen

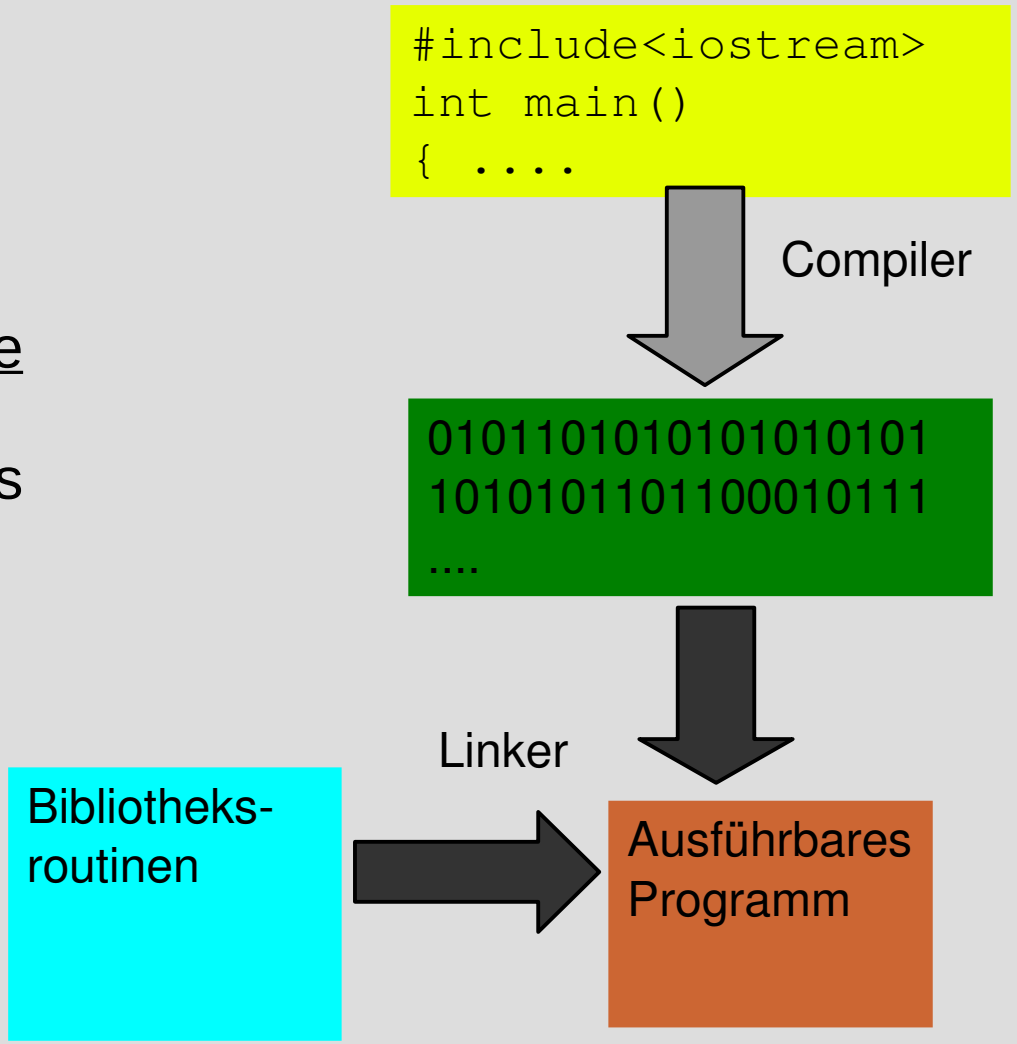
Ein Beispiel für ein C(++) Programm

```
// einbinden von nützlichen Funktionen
#include<iostream>

// die Hauptfunktion
int main()
{
   std::cout<<”Hello World”<<std::endl;
}
```

Was geschieht mit einem solchen Programm?

- Der Rechner kann dieses Sourcefile nicht direkt ausführen
- Es muß zuerst vom Compiler übersetzt werden, d.h. in ein maschinen-lesbares Objectfile gewandelt
- Der Linker verbindet dann das Objectfile mit den notwendigen Bibliotheksroutinen zum Executable



Beispiel anhand der GNU g++ Toolchain

Aufruf des Compilers:
`g++ hello.cc -c`

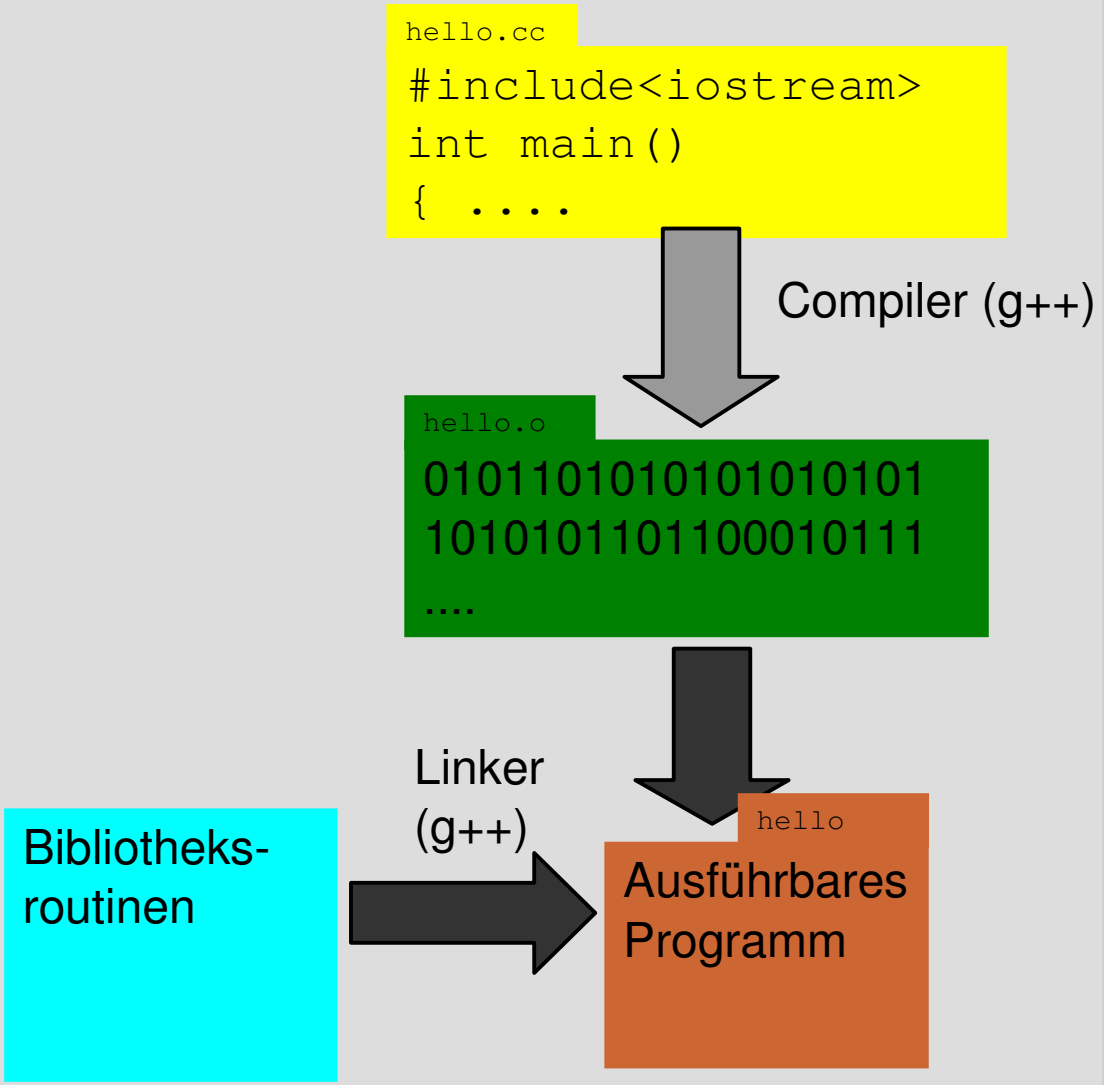
erzeugt Objectfile `hello.o`

Starten des Linkers mit

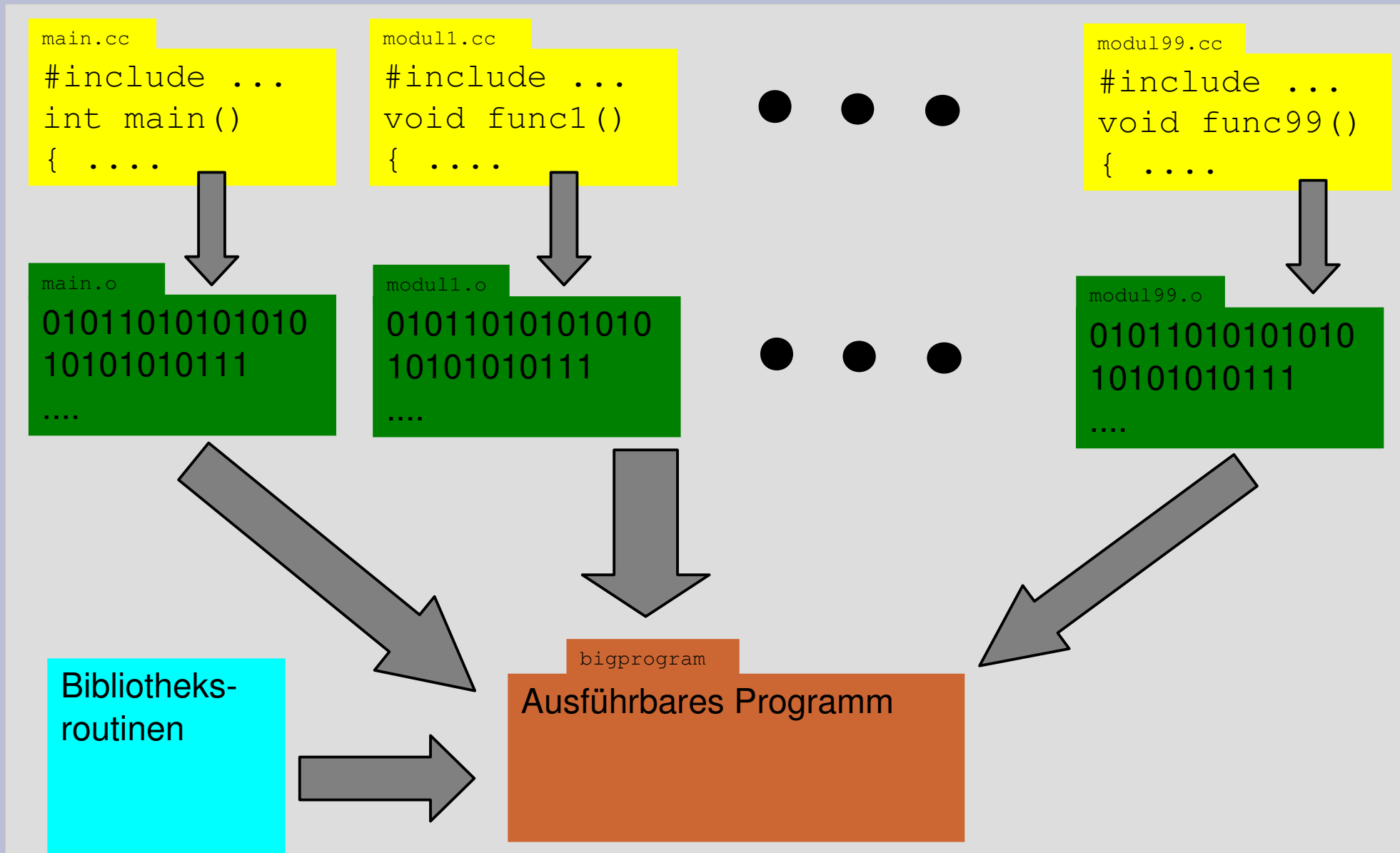
`g++ hello.o -o hello`

bindet notwendige Bibliotheks-routinen hinzu und erzeugt ausführbares Programm `hello`

Starten des Programmes mit
`./hello`



Komplexe Projekte bestehen meist aus vielen Sourcefiles



Erfolgreich größere Program-mierprojekte durchführen

- ... hat sehr viel mit guter Organisation zu tun
- Wie kann ein großes Projekt in kleine Module zerlegt werden?
- Genau definieren, wie Module interagieren
- Ausführlich kommentieren!
- ... den Überblick bewahren!

Unsere Arbeitsumgebung: LINUX

- Freies UNIX Betriebssystem
- Enthält vollständige Entwicklungstoolchain (GNU g++), um C/C++ Programme zu entwickeln
- Kommandozeilenorientiert (es gibt auch IDEs = Integrated Development Environment, z.B. Eclipse); benutzen wir aber zunächst nicht
- Multiuser-Betriebssystem: mehrere Benutzer können gleichzeitig auf einem Rechner arbeiten (Remote-Login)

Start von Linux auf Ihrem lokalen Rechner

- Rechner einschalten :-)
- im Bootmanager „Suse Linux“ auswählen
- Einloggen, Beispiel:
  Username=req13
  Passwort=req13rtk
- Shell starten (Muschel-Icon unten)

Kommandos auf der Shell

ls	Verzeichnisinhalt ausgeben
mkdir	Verzeichnis erstellen
cd	in Verzeichnis wechseln (ohne Argument: in Home-Verz. wechseln)
rmdir	Verzeichnis löschen
mv	Dateien/Verzeichnisse ver-schieben/umbenennen
man	Hilfe zu Kommando aufrufen
kate	Texteditor aufrufen

Aufgabe 1

- Erstellen Sie folgende Verzeichnisstruktur
  ```
  NameVorname/
     Trockenuebungen/
     MeinProjekt/
  ```

- Erzeugen Sie in `NameVorname/` eine Datei `Email.txt`, welche Ihre Email Adresse enthält
- Erzeugen Sie in `NameVorname/Trockenuebung/` eine Datei `Changelog.txt`, in welcher Sie dokumentieren, was sie gerade getan haben, z.B. so: `2008/04/09: Verzeichnisstruktur erstellt`

Ein minimales C/C++ Program

```
// einbinden von nützlichen Funktionen
#include<iostream>

using namespace std;

// die Hauptfunktion

int main()
{
   cout<<”Hello World”<<endl;
}
```

Speichern/Übersetzen des Programms

- Tippen Sie das Programm in einem Editor (z.B. kate) ein und speichern Sie es unter hello.cc
- gehen Sie auf die Shell und kompilieren es:
  `g++ hello.cc -c`
- schauen Sie nach, was der Compiler produziert hat mit:
  `ls`
- Linken Sie das Programm mit
  `g++ hello.o -o hello`
- Programm starten mit
  `./hello`

Variablen und Konstanten in C/ C++

- C/C++ unterscheidet zwischen verschiedenen Arten der Repräsentation von Daten – Datentypen:
  - `int` ganze Zahlen, z.B. 7
  - `float` Gleitkommazahlen, z.B. 4.6
  - `char` Zeichen, z.B. 'a'
  - `void` Spezieller (Nicht-)Typ, der nur für Funktionsdeklarationen gebraucht wird

Ein-/Ausgabe von Daten

- `cout` gibt Zeichenketten/Variablenwerte aus

```
cout<<“Variable x hat Wert
"<<x<<endl;
```

- `endl` erzeugt einen Zeilenvorschub
- `cin` liest Werte ein:

```
int x;
cout<<“Bitte Wert eingeben:  ";
cin>>x;
```

Aufgabe

- Schreiben Sie ein Programm, welches 2 Zahlen einliest, diese addiert und die Summe ausgibt.

for-Schleife

- Syntax:
  for (StartBefehl; Bedingung; SchleifenUpdate)
- Beispiel:

```
for (int i=0; i<20; i++)
{
    ... code ...
}
```

- obige Schleife wird 20 Mal durchlaufen, i hat dabei die Werte i=0,1,2,...,19

while-Loop (eigentl. überflüssig)

- Syntax:

```
while (Bedingung)
   { ..<body>....}
```

- führt den Code in <body> aus, solange die Schleifenbedingung erfüllt ist
- kann ersetzt werden durch

```
for( ;Bedingung; )
   {..<body>....}
```

- umgekehrt können Sie eine for-Schleife auch durch einen while-Loop ersetzen (mit entsprechenden Variablen)

while-Loop (nicht ganz überflüssig)

- alternativer Syntax:

```
do
{...<body>...}
while (Bedingung);
```

- führt den Schleifenbody einmal aus (auf jeden Fall) und wiederholt, solange die Bedingung erfüllt ist

Aufgaben

- Schreiben Sie ein Programm,
  - das die Zahlen von 1 bis 20 aufsteigend ausgibt
  - das die Zahlen von 20 bis 1 absteigend ausgibt
  - die Summe der ersten n (Eingabe) Zahlen berechnet
  - n! berechnet

Arrays

- Mittels

  ```
  int A[20];
  ```

  legen Sie ein Array aus 20 Ganzzahlen an, die Sie mit A[0], A[1], ... A[19] ansprechen können

Aufgabe

- Schreiben Sie ein Programm, das eine Zahl n einliest, dann nach n Zahlen fragt und diese mittels Bubblesort sortiert
- Recherchieren Sie falls nötig Bubblesort im Internet

Funktionen

- Syntax:

```
returntyp funktionsname(argtyp argname, ...)
{
   // „normaler Programmcode“
}
```

- als `returntyp` bzw. `argtyp` können Sie alle elementaren C/C++ Datentypen verwenden, für `returntyp` auch void (dann hat die Funktion keinen Rückgabewert)
- Funktionen müssen **vor** ihrer Verwendung definiert werden
- bei zyklischen Abhängigkeiten können Sie Funktionen „vorab“ ankündigen (Strichpunkt beachten)

```
returntyp funktionsname(argtyp, argtyp, ...);
```

Funktionen: Ergebnisrückgabe

Call-by-Value:

```
int quadrat1(int x)
{
   return x;
}
```

Call-by-Reference:

```
void quadrat2(int &x)
{
   x=x*x;
}
```

Im ersten Fall würden Sie wie folgt das Quadrat von X berechnen und ausgeben:

```
int y=quadrat(x);
cout<<"Das Quadrat von " <<x<<" ist
"<<y<<endl;
```

Im zweiten Fall würde das ergebnis direkt in die Variable x geschrieben, d.h. falls der Originalwert von x noch benötigt wird, muss er vorher gesichert werden:

```
int x_alt=x;
quadrat2(x);
cout<<"Das Quadrat von " <<x_alt<<"
ist "<<x<<endl;
```

Im Allgemeinen ist die Rückgabe eines Ergebnisses "by-value" weniger fehleranfällig.

Aufgabe

- Schreiben Sie ein Programm, welches zwei (positive ganze) Zahlen x,y einliest und „x hoch y“ berechnet
- Die Potenz sollte dabei in einer Funktion berechnet werden und einmal durch einen Rückgabewert, und einmal per Call-by-Reference zurückgegeben werden

Weiteres zu Arrays

- Übergeben eines Arrays an eine Funktion:

```
void fun(int A[], int n)
{ // Code, der auf A[0],...,A[n-1]
  // operiert
}
```

- Aufruf:

```
{
   int A[20];
   fun(A, 20);
}
```

- Übergabe hier immer als Call-by-Reference!
- Größe des Vektors (hier: 20) muss immer mit übergeben werden

Aufgabe

- Lagern Sie Ihre Bubblesort-Routine in eine eigene Funktion aus, welche als Parameter das zu sortierende Array und die Array-Größe übernimmt

Modularisierung (1)

- oft wollen Sie eine geschriebene Routine (wie z.B. Ihr Bubblesort) auch in anderen Programmen wiederverwenden und nicht immer in das entsprechende Sourcefile einkopieren
- in diesem Fall sollten Sie die Sortierroutine in einer separaten Datei speichern (z.B. `bbsort.cc`), welches Sie für sich übersetzen können
- desweiteren legen Sie ein Header-file (`bbsort.h`) an, sodass andere Module, die Ihre Sortierroutine benutzen möchten, das Interface im entsprechenden Header-File (`bbsort.h`) mittels `#include` einbinden können

Modularisierung (2)

main.cc

```
....
#include“myfun.h“
int main()
{
     int A[100];
     // Zahlen einlesen
     ..
     mysort(A,100);
     // Ausgabe
     ..
}
```

myfun.h

```
void mysort( ... );
```

myfun.cc

```
....
#include“myfun.h“

void mysort(...)
{
     // Implementierung
     // der Funktion
}
```

Übersetzung mit:

```
g++ -c main.cc
g++ -c myfun.cc
g++ main.o myfun.o -o main
```

Modularisierung (3)

- je mehr Module sie haben, desto umständlicher wird das übersetzen
- insbesondere die Übersicht darüber zu behalten, welche Änderungen welche Neuübersetzungen bedürfen, ist schwierig
- wenn Sie z.B. am Interface von mysort() ändern, müssen sowohl myfun.cc als auch main.cc neu übersetzt werden
- wenn Sie nur die Implementierung/Umsetzung von mysort() intern ändern, reicht es, myfun.cc neu zu übersetzen
- mit einem Makefile können Sie über solche Abhängigkeiten den Überblick bewahren und nur die wirklich notwendigen Module neu übersetzen

Makefiles

Syntax:
`<target>:<list of dependencies>`
`<tab><build-command>`
Ein „Target“ wird nur neu erstellt, wenn es älter (Datum/Zeit) als die „Dependencies“ ist.
Es wird per default immer versucht, das erste Target zu „bauen“: `make`
Alternativ kann das Target angegeben werden:

```
make main.o
```

Abhängigkeiten werden rekursiv verfolgt, d.h. falls sich `myfun.h` geändert hat, wird bei Eingabe von make alles neu übersetzt+ gelinkt. Wenn sich nur `myfun.cc` geändert hat, wird nur das übersetzt+gelinkt.
Man speichtert ein Makefile typischerweise unter dem Namen `Makefile`.

Makefile

```
main: main.o myfun.o
	g++ main.o myfun.o -o main

main.o: myfun.h main.cc
	g++ -c main.cc

myfun.o: myfun.cc myfun.h
	g++ -c myfun.cc
```

Aufgabe

- Zerlegen Sie Ihr Sortierprogramm in zwei Module:
  - eines welches die eigentliche Sortierroutine enthält – mit Header-File): `bbsort.cc/bbsort.h`
  - die „Hauptroutine“ (enthält die Funktion main()): `main.cc`
- Erstellen Sie das entsprechende Makefile dazu
- Übersetzen und testen Sie Ihr Programm

Komplexe Datentypen

- oft ist es wünschenswert, mehrere Daten (z.B. Immatrikulationsnummer, Alter, Punktzahl) zu einem komplexeren Objekt zusammenzufassen und dieses wie einen „normalen“ Datentyp behandeln zu können (also wie `int`, `float`, `double`, ...)
- dazu gibt es in C++ die sogenannten Klassen

C++ Klassen

```
class ExamGrade
   {
   public:
      int matrno;
      int grade;
   };
```

- C++ Klassen müssen in einem Header-File deklariert werden und können in einem .cc File wie folgt instantiiert werden:
  `ExamGrade pruefung;`
- Auf die Elemente einer Klasse kann man wie folgt zugreifen:
  `pruefung.matrno=1234567;`
- `public:` bedeutet, dass auf die folgenden Elemente von außen zugegriffen werden kann

C++ Klassen

- außer „Daten“ können Sie auch Funktionen (oder hier „Methoden“ genannt) in eine C++ Klasse packen, welche jedoch nur auf den Daten der Klasse operieren darf:

```
class ExamGrade{
   public:
      int matrno;
      int grade;
      void printMe();
};
```

- Mögliche Anwendung:

```
ExamGrade pruefung;
pruefung.matrno=10;
pruefung.grade=14;
pruefung.printMe();
```

C++ Klassen

- die Methode printMe() müssen Sie jedoch noch entsprechend bereitstellen:

```
void ExamGrade::printMe()
{
   std::cout<<“Matr.nr. "<<matrno<<" hat
     Note "<<grade<<std::endl;
}
```

- die Klassendefinition (`class ExamGrade ...`) gehört typischerweise in ein `.h` File, die obige Implementierung der Methode in das entsprechende `.cc` File, ähnlich der Aufteilung in "Ankündigung" und Implementierung einer Funktion in einem Modul

Einschub: Die Klasse string

- Sie können aus der Standardbibliothek für C++ einen speziellen Datentyp `string` benutzen, welcher Zeichenketten repräsentiert und einfacher als Vektoren vom Typ `char` zu handhaben ist
- Sie müssen dazu das entsprechende Include-File einbinden:
  `#include<string>`
- und können damit Variablen vom Typ string instantiieren, einlesen und auslesen:

```
std::string meinName;
std::cin>>meinName;
std::cout<<"Ich heisse "<<meinName<<std::endl;
```

- Wichtig: ein mittels `cin` eingelesener String kann keine Leerzeichen enthalten
- Wie schon bei `cin`,`cout` und `endl`, können Sie sich das `std::` sparen, wenn Sie zuvor `using namespace std;` eingeben

Aufgabe

- Schreiben Sie eine Klasse, welche eine Tankstelle (mit Adresse und Benzinpreis) repräsentiert und auch zwei Methoden bereitstellt, um eine Instanz dieser Klasse mit entsprechenden Werten zu belegen (z.B. Methoden `::eingabe()` und `::ausgabe()`
- Nutzen Sie diese Klasse in einem Programm, welches nach einer Zahl `n` fragt, und dann zur Eingabe der Daten von `n` Tankstellen auffordert, und diese dann in umgekehrter Reihenfolge wieder ausgibt
- Ihr Programm sollte über die entsprechenden `.h` und `.cc` Dateien verteilt sein und über ein Makefile kompiliert werden

Die Standard C++ Library (STL)

- es gibt eine Standardbibliothek für das Programmieren in C++, die „Standard Template Library“ (STL)
- siehe auch http://www.sgi.com/tech/stl/
- sie stellt eine Vielzahl nützlicher Funktionen bereit, z.B. Routinen zur Sortierung, Dateiein- und ausgabe, etc.

Aufgabe

- Schlagen Sie im STL Manual die Klasse `vector` nach; inwieweit unterscheidet diese sich von einem Array/Vector, wie Sie ihn bislang genutzt haben?
- ... wahrscheinlich werden Sie etwas erschlagen sein von der Fülle der Informationen

Die `vector` Klasse für Dummys

- Sie können einen `vector` prinzipiell fast genauso wie ein Array behandeln, d.h. `A[0]` greift auf das erste Element, `A[1]` auf ....
- wichtige Unterschiede:
  - Größe eines `vector` ist dynamisch (0 anfangs)!
  - einem `vector v` kann ein Element `e` mit `v.push_back(e)` hinzugefügt werden
  - die Größe kann auch explizit durch `v.resize()` angeben werden

Die `vector` Klasse für Dummys

- Sie können einen `vector` von (fast) jeder Klasse/Datentyp erstellen
- Beispiel:

```
#include<vector>
...
vector<int> meineZahlen;
meineZahlen.push_back(7);
meineZahlen.push_back(8);
std::cout<<meineZahlen[0]<<"
"<<meineZahlen[1]<<std::endl;
```

Aufgabe

- Modifizieren Sie Ihr letztes Programm, sodass es die Klasse `vector` benutzt anstatt einfacher Arrays/Vektoren

Dateiein-/ausgabe

- Es ist auch möglich, Daten auf Festplatte zu schreiben bzw. zu lesen
- `#include<fstream>`
  bindet den notwendigen Header ein
- Datei "myFile.txt" zum Lesen öffnen und ersten String lesen:

```
ifstream meineDatei("myFile.txt");
meineDatei>>myString;
meineDatei.close();
```

- Datei "myFile.txt" zum Schreiben öffnen:

```
ofstream meineDatei("myFile.txt");
meineDatei<<"Hallo"<<endl;
meineDatei.close();
```

Aufgabe

- Erweitern Sie Ihr Tankstellenprogramm, sodass Sie Ihre Eingabe auch in eine Datei speichern können bzw. davon laden
- achten Sie darauf, Ihren Code über die entsprechenden .h und .cc Dateien zu verteilen und ein passendes Makefile
- **führen Sie mir Ihr Ergebnis vor**

Nützliche Tips (1)

- Schreiben Sie sobald wie möglich ein Makefile (!)
- gewöhnen Sie sich ein konsistenten Stil in Sachen Einrückungen an (z.B. 2x <Space> pro Programmblock)

```
if (hallo)
{
  for(int i=0; i<n; i++)
    counter++;
}
```

Nützliche Tips (2)

- Benutzen Sie wenn möglich aussagekräftige und sinnvolle Variablennamen (Ausnahme evtl. Schleifenvariablen)
- Legen Sie fest, ob Sie Variablen oder Typen groß oder klein schreiben, z.B. alle Typ- bzw. Klassennamen mit Großbuchstaben beginnen, alle Variablen mit Kleinbuchstaben

Projektskizze (1)

- Ziel des Projektes ist es einen kleinen „Routenplaner“ zu implementieren der es erlaubt
  - Straßenkarten mit der Maus visuell zu navigieren
  - Orte/Knoten auf der visualisierten Straßenkarte auszuwählen
  - kürzeste Wege zwischen Orten zu berechnen

Projektskizze (2)

- Sie bekommen dafür Straßendaten (der USA) zur Verfügung gestellt, welche sowohl mit Koordinaten (Knoten) als auch Reisezeiten (Kanten) behaftet sind

Projektskizze (3)

- Die grundlegende Datenstruktur des Routenplaners ist die Repräsentation des Straßennetzwerks
- Diese Repräsentation sollte so gewählt sein, dass sowohl der Zugriff auf die Daten zwecks Visualisierung als auch zur Berechnung der kürzesten Wege so effizient wie möglich erfolgen kann
- desweiteren ist auf Platzeffizienz zu achten, da große Straßennetzwerke wie das der USA ca. 20 Mio Knoten und 60 Mio Kanten enthalten

Projektskizze (4)

- Vorgehensweise
  - Entwurf einer Datenstruktur, welche die Straßendaten speichert
  - Implementierung eines Algorithmus zur Berechnung kürzester Wege in Graphen (Dijkstra's Algorithmus)
  - Implementierung der Visualisierungs-/Interaktionskomponente

Aufgabe

- Entwerfen Sie eine Datenstruktur zur Repräsentierung der Straßendaten inkl. Ein-/Ausgabeoperatoren
- Die Repräsentation sollte insbesondere den effizienten Zugriff auf die zu einem Knoten adjazenten Knoten erlauben
- Testen Sie Ihre Datenstruktur anhand der Beispieldaten auf der Webseite

Vorschlag einer Repräsentation

- Knoten sind durchnummeriert (0, 1, ...)
- wir speichern in einem Vektor `coordList` die Koordinaten jedes Knotens
- die Kanten sind durchnummeriert (0,1, ...) und zwar geordnet nach Quellknoten
- wir speichern in einem Vektor `edgeList` für jede Kante ihren Zielknoten und ihr Gewicht (Reisezeit)
- in einem zusätzlichen Vektor `edgeOffsets` wird für jeden Knoten ein Verweis auf die erste ausgehende Kante gespeichert

Aufgabe

- Wieviel Speicher verbraucht diese/Ihre Repräsentation (bzgl. #Kanten und #Knoten)?
- Wie kann man damit die zu einem Knoten i adjazenten Knoten ansprechen?

Aufgabe: Dijkstras Algorithmus

- schlagen Sie im Internet Dijkstras Algorithmus nach
- implementieren Sie ihn für Ihre Repräsentation des Straßennetzwerks
- wahrscheinlich brauchen Sie dazu den Datentyp `priority_queue`, aus der C++ STL (schlagen Sie ihn nach!)
- wie "verpacken" Sie den Algorithmus in eine Klasse?
- Diskutieren Sie Ihren "Verpackungsentwurf" mit mir, bevor Sie mit der Implementierung beginnen

Aufgabe: Dijkstras Algorithmus (2)

- Wenn Sie Ihre Implementierung auf dem Spielzeuggraphen getestet haben, probieren Sie es mit den Daten von Massachusetts, die Sie auf der Webseite finden

Aufgabe: Dijkstras Algorithmus (3)

- Sie finden in /home/PROJEKT/ meine Implementierung, welche Sie mit

  ./demo-dijkstra MA.gtxt <start> <ziel>

  aufrufen können
- das Programm berechnet:
  - Distanz <start> nach <ziel> **inkl**. Initialisierung des Distanzvektors (**ohne** Abbruch bei erreichen von <ziel>)
  - Distanz <start+1> nach <ziel+1> **inkl**. Reinitialisierung der zuvor umgesetzten Distanzwerte (**+** Abbruch bei erreichen von <ziel>)
  - Distanz <start+2> nach <ziel+2> **inkl**. Reinitialisierung nur der zuvor umgesetzten Distanzwerte (**+** Abbruch bei erreichen von <ziel>)
  - Distanz <start+3> nach <ziel+3> **inkl**. Reinitialisierung nur der zuvor umgesetzten Distanzwerte (**ohne** Abbruch bei erreichen von <ziel>)

Aufgabe: Dijkstras Algorithmus (4)

- Vergleichen Sie die Laufzeiten mit Ihrer Implementierung; nutzen Sie zur Bestimmung der Laufzeiten Ihrer Routinen die Klasse Timer in
  /home/PROJEKT/Timer.h
  Relevante Methoden sind ::start(), ::stop() und ::secs()
- Versuchen Sie, Laufzeitunterschiede zu erklären

Aufgabe: Dijkstras Algorithmus (5)

- Fügen Sie eine Routine hinzu, die Ihnen auch die Ausgabe eines berechneten kürzesten Weges erlaubt
- Sprechen Sie mit mir Ihre aktuelle Implementierung durch, arbeiten Sie eventuelles Feedback ein, bevor Sie mit dem GUI beginnen

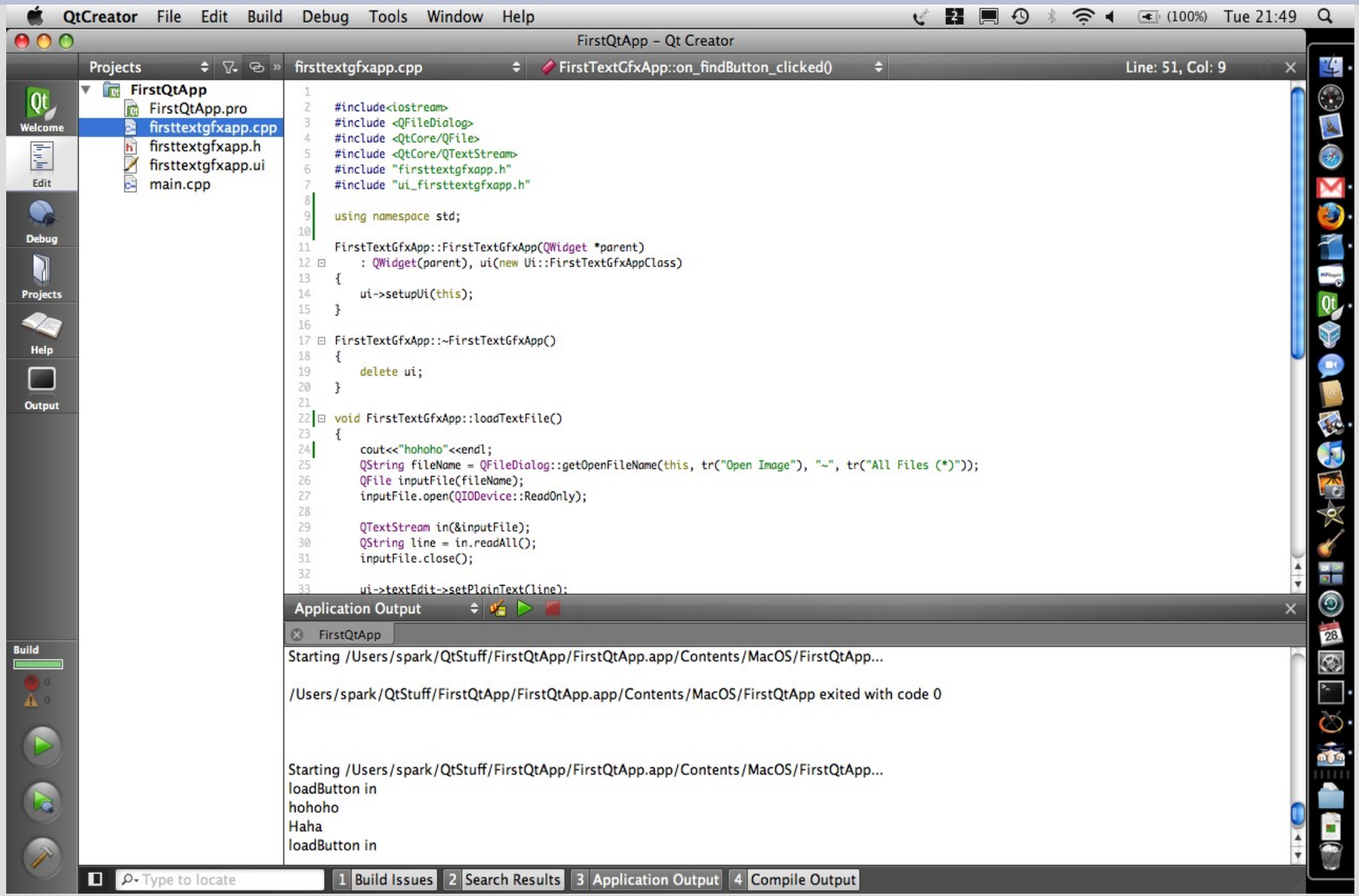
Graphical User Interface (GUI)

- Nach Abschluss des algorithmischen Teils des Routenplaners möchten wir auch eine grafische Benutzeroberfläche dazu entwickeln
- dazu werden wir die Klassenbibliothek Qt benutzen, siehe auch
  http://trolltech.com/products/qt

QtCreator

- Trolltech – der Hersteller der Qt Bibliothek – bietet eine IDE (Integrated Development Environment), welche das Erstellen von größeren Programmprojekten erleichtert
- Finden Sie im Internet QtCreator 1.1, laden Sie es herunter und installieren Sie es gemäß den Instruktionen **Hinweis**: Für die Rechner im RTK brauchen Sie die 64bit-Linux-Version, falls Sie daheim Linux installiert haben, höchstwahrscheinlich die 32bit-Linux-Version
- QtCreator ist für verschiedene Plattformen erhältlich, die Screenshots wurden auf der MacOS X Version erstellt

QtCreator

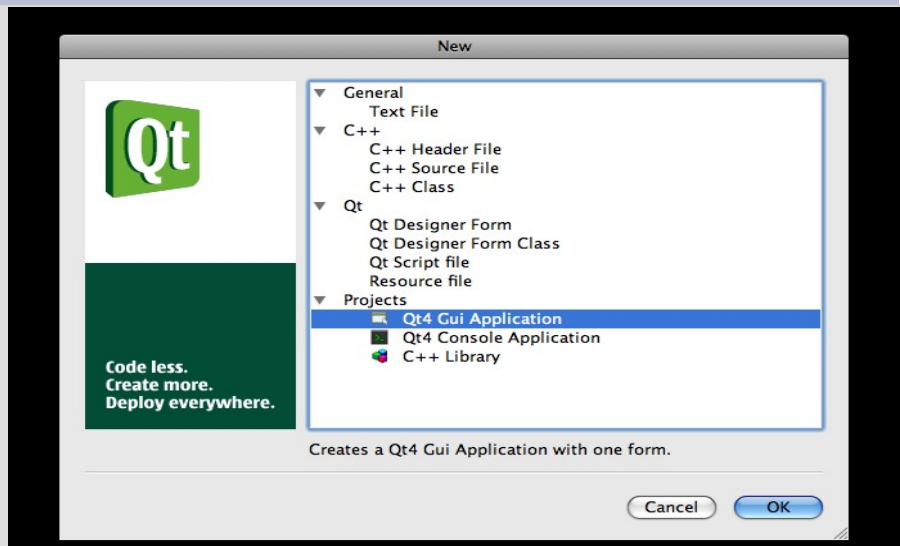

QtCreator File Edit Build Debug Tools Window Help
FirstQtApp – Qt Creator
Projects
firsttextgfxapp.cpp
FirstTextGfxApp::on_findButton_clicked()
Line: 51, Col: 9
FirstQtApp
FirstQtApp.pro
firsttextgfxapp.cpp
firsttextgfxapp.h
firsttextgfxapp.ui
main.cpp
Welcome
Edit
Debug
Projects
Help
Output
Build
#include<iostream>
#include <QFileDialog>
#include <QtCore/QFile>
#include <QtCore/QTextStream>
#include "firsttextgfxapp.h"
#include "ui_firsttextgfxapp.h"
using namespace std;
FirstTextGfxApp::FirstTextGfxApp(QWidget *parent)
: QWidget(parent), ui(new Ui::FirstTextGfxAppClass)
{
ui->setupUi(this);
}
FirstTextGfxApp::~FirstTextGfxApp()
{
delete ui;
}
void FirstTextGfxApp::loadTextFile()
{
cout<<"hohoho"<<endl;
QString fileName = QFileDialog::getOpenFileName(this, tr("Open Image"), "~", tr("All Files (*)"));
QFile inputFile(fileName);
inputFile.open(QIODevice::ReadOnly);
QTextStream in(&inputFile);
QString line = in.readAll();
inputFile.close();
ui->textEdit->setPlainText(line);
Application Output
FirstQtApp
Starting /Users/spark/QtStuff/FirstQtApp/FirstQtApp.app/Contents/MacOS/FirstQtApp...
/Users/spark/QtStuff/FirstQtApp/FirstQtApp.app/Contents/MacOS/FirstQtApp exited with code 0
Starting /Users/spark/QtStuff/FirstQtApp/FirstQtApp.app/Contents/MacOS/FirstQtApp...
loadButton in
hohoho
Haha
loadButton in
Type to locate
1 Build Issues
2 Search Results
3 Application Output
4 Compile Output

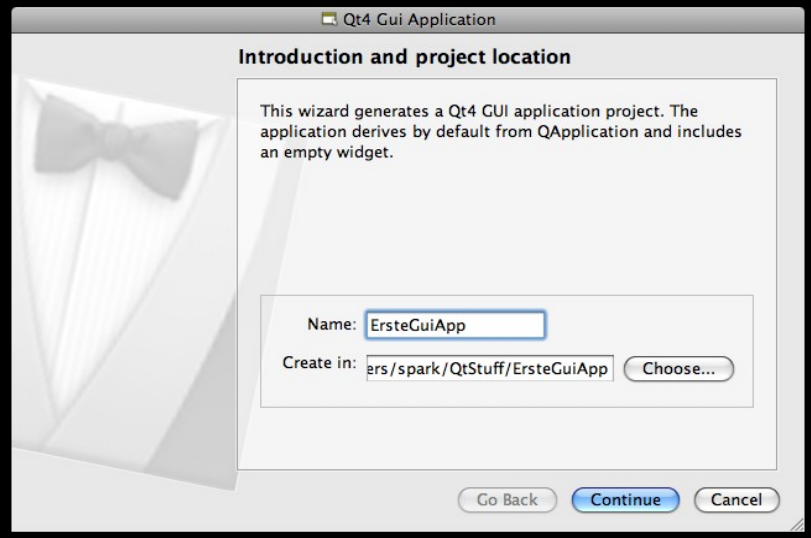
QtCreator

- Wir werden im Folgenden Schritt-für-Schritt eine kleine Demo-Anwendung mit Qt erstellen, welche einige der Elemente eines Qt-Programms illustriert
- Danach sollten Sie – unter Zuhilfenahme der Dokumentation von Qt – in der Lage sein, ein einfaches GUI für den Routenplaner

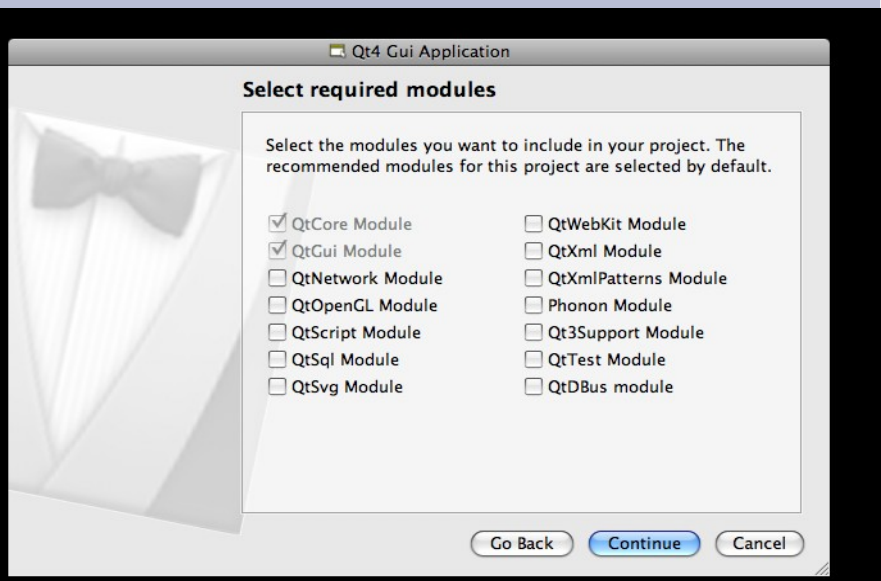
Erstellen eines GUI-Skeletts

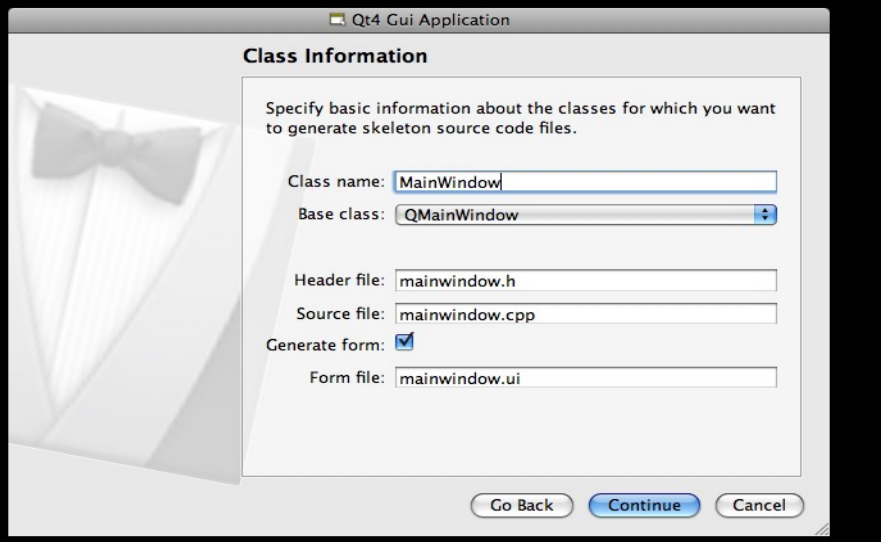
- Erzeugen Sie ein neues Projekt durch File->New
- Wählen Sie dann Qt4 "Gui Application"
- Denken Sie sich einen schönen Namen für Ihre erste Qt-Anwendung aus (z.B. "ErsteGuiApp")
- Legen Sie ein Projektverzeichnis fest (am Besten ein neues erstellen)



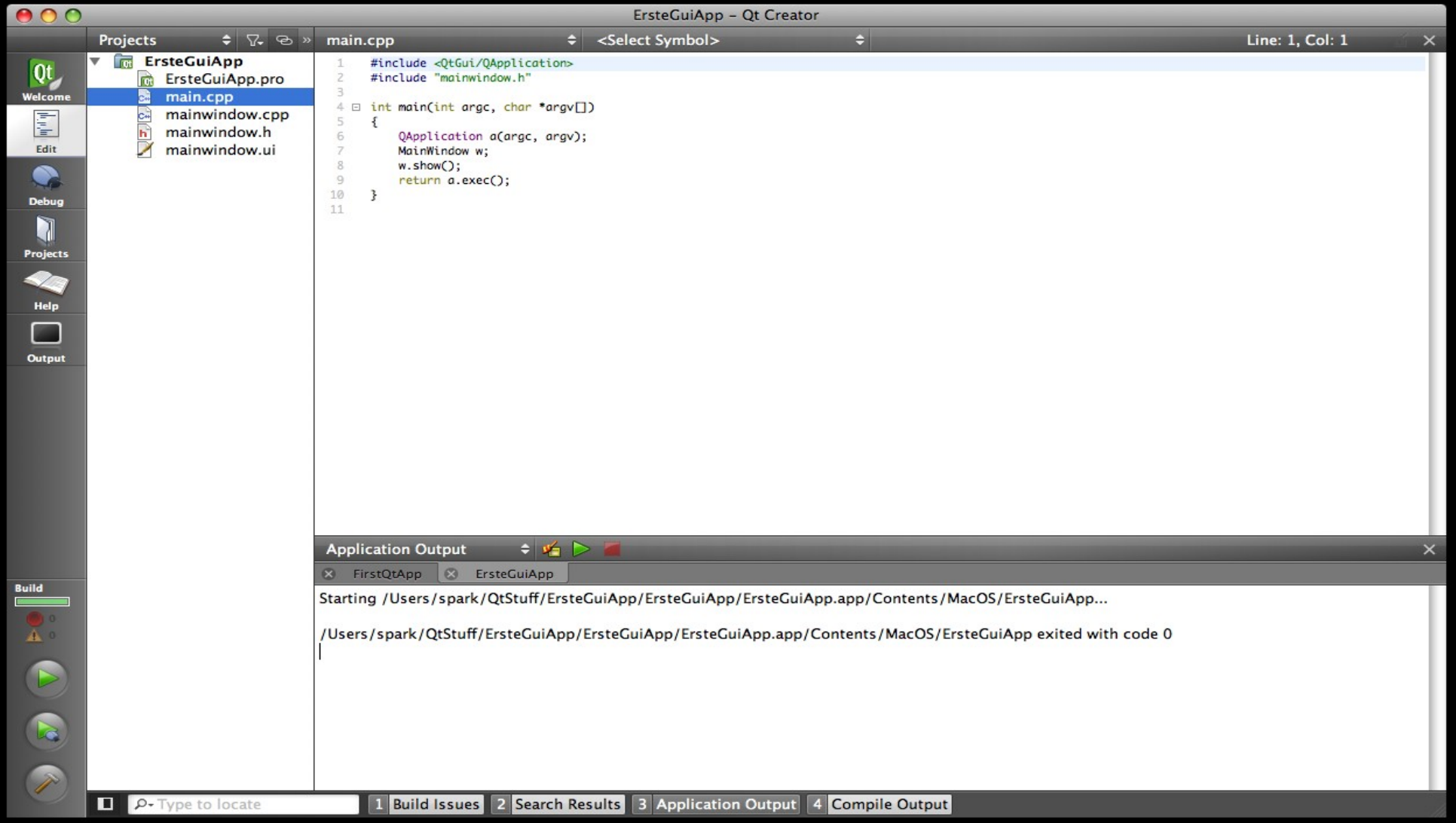


- da wir keine besonderen Funktionalitäten benötigen, müssen keine weiteren Module im folgenden Dialog ausgewählt werden
- als Name der Klasse, welche das "Hauptfenster" repräsentiert, kann beibehalten werden





QtCreator - Hauptansicht



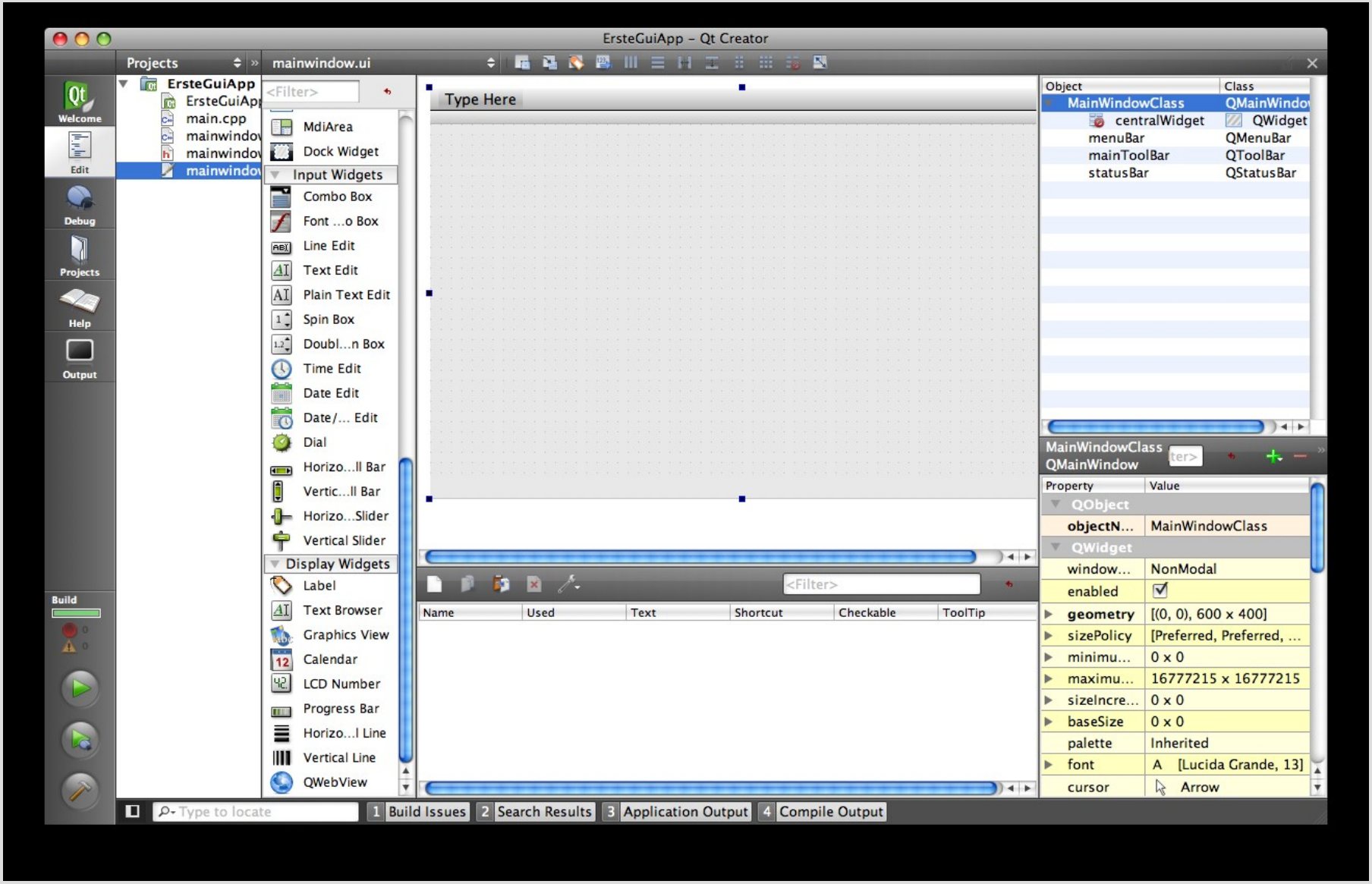
QtCreator

- QtCreator hat für Sie automatisch schon ein Grundgerüst für eine Qt-Anwendung angelegt, welche Sie durch Build->Run auch übersetzen und starten können
- auf der linken Seite sehen die alle Dateien, die in Ihrem Projekt enthalten sind; schauen Sie sich die Dateien im einzelnen an

Hinzufügen von GUI Elementen

- als erstes möchten wir die Struktur unseres GUIs entwerfen; unser erstes Programm soll folgende GUI-Elemente enthalten:
  - eine Textzeile, wo wir eine Zeile Text eingeben können
  - eine Grafikbox, in die wir malen können
  - eine Textbox, in den ein längerer Text eingegeben/geladen werden kann
  - ein Knopf, mit dem wir eine Datei auswählen können, die in die Textbox geladen wird
  - ein Knopf, der nach dem Text der Textzeile im Text der Textbox sucht und gleichzeitig etwas in die Grafikbox malt
- klicken Sie hierzu auf die .ui-Datei Ihres Projektes

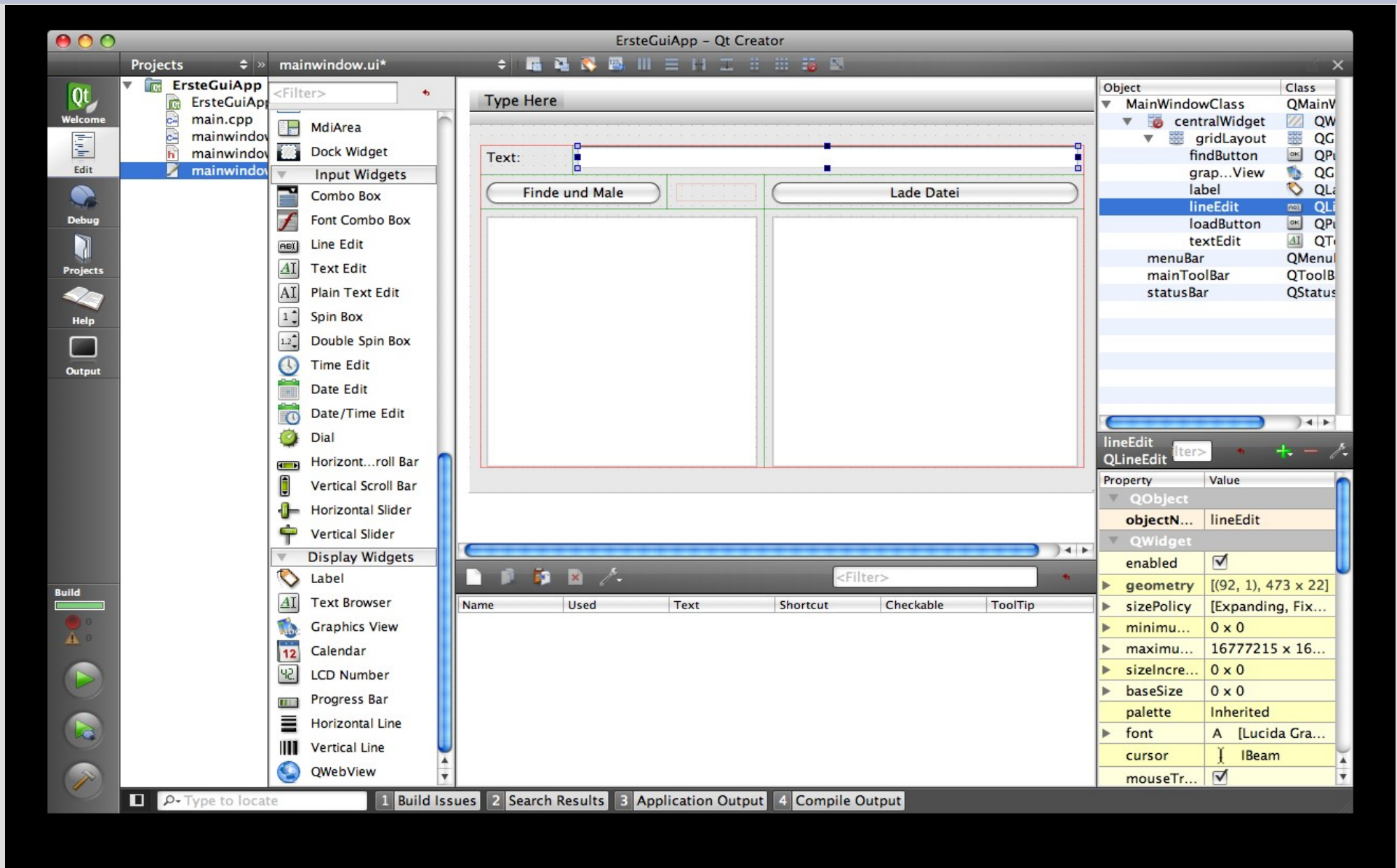
QtCreator – QtDesigner



QtCreator – QtDesigner

- der sich öffnende QtDesigner erlaubt es, per Maus verschiedene GUI-Elemente hinzuzufügen, die Bedienung sollte sich Ihnen mit der Hilfefunktion erschließen
- erstellen Sie das Userinterface, indem Sie zunächst alle gewünschten "Widgets" platzieren und diese dann in einem "Gridlayout" anordnen (→ Hilfefunktion!)
- versuchen Sie das folgende Ergebnis zu bekommen:

QtCreator – QtDesigner

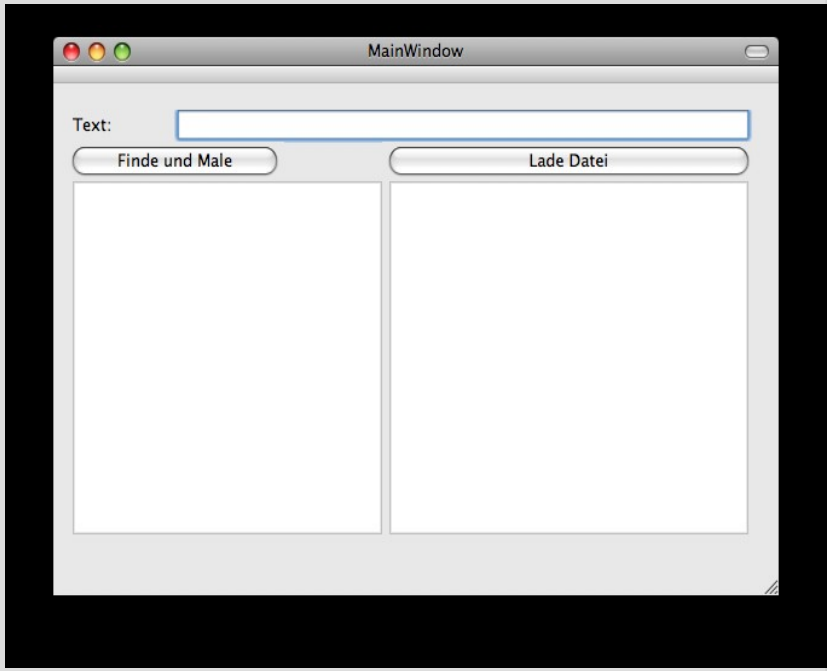


QtCreator – QtDesigner

- achten Sie darauf, dass die entsprechenden "Widgets" (so heißen die GUI Elemente) folgende Namen (Fenster rechts oben) haben:
  - "Lade Datei"-Knopf soll "loadButton" heißen
  - "Finde und Male"-Knopf soll "findButton" heißen
  - Graphics View soll "graphicsView" heißen
  - die einzeilige Texteingabe soll "lineEdit" heißen
  - das mehrzeilige Textfeld soll "textEdit" heißen
- übersetzen Sie Ihre Projekt nochmal und lassen es laufen

QtCreator

- als Ergebnis sollten Sie folgendes Fenster auf dem Bildschirm sehen
- zu diesem Zeitpunkt hat die Oberfläche jedoch keine Funktionalität, d.h. Anklicken der Knöpfe bewirkt nichts
- das wollen wir im Folgenden ändern



Eventhandler

- um das GUI mit Funktionalität zu füllen müssen Sie in Ihrer Klasse mainWindow für jedes GUI-Ereignis(Event) eine entsprechende Methode schreiben
- Beispiel: die Aktion, die bei Drücken des Knopfes "loadButton" ausgeführt werden soll, müssen Sie eine Methode

  ```
  void on_loadButton_clicked()
  ```

  der Klasse mainWindow hinzufügen; die muss in der Sektion

  ```
  private slots:
  ```

  der .h-Datei erfolgen

Eventhandler

- fügen Sie einen entsprechenden Eintrag der mainWindow.h Datei hinzu und nutzen Sie folgende Implementierung in der entsprechenden .cpp Datei:

```
QString fileName = QFileDialog::getOpenFileName(this,
      tr("Open Image"), "~", tr("All Files (*)"));
QFile inputFile(fileName);
inputFile.open(QIODevice::ReadOnly);

QTextStream in(&inputFile);
QString line = in.readAll();
inputFile.close();

ui->textEdit->setPlainText(line);
QTextCursor cursor = ui->textEdit->textCursor();
```

- obiger Code startet einen Filerequester und lädt die gewählte Datei in das große Textfeld Ihres GUIs
- Schlagen Sie die dabei verwandten Klassen in der Qt Dokumentation nach, fügen Sie die dafür notwendigen #include-Direktiven in der .cpp Datei ein

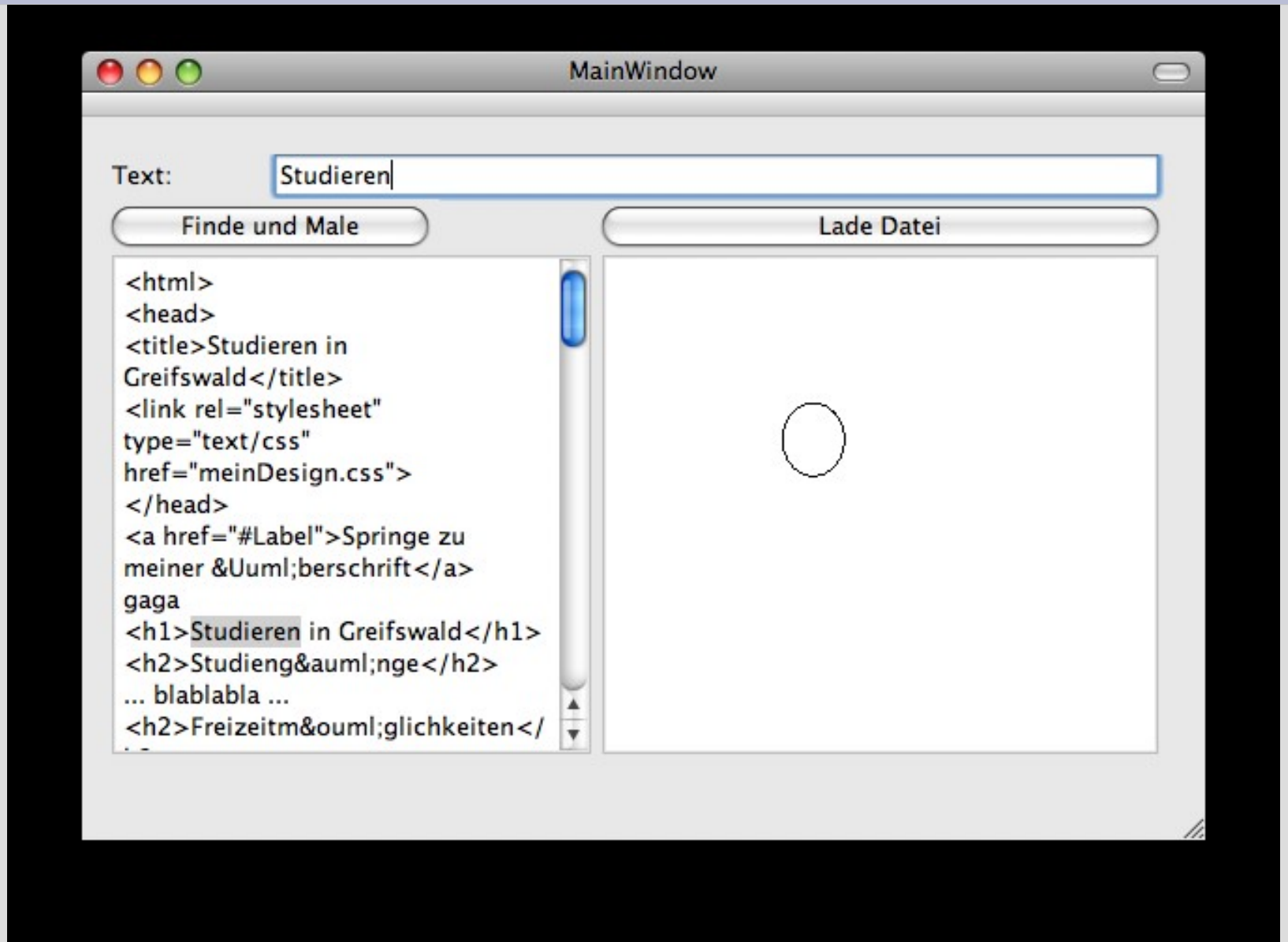
Eventhandler

- analog implementieren Sie einen Eventhandler für den findButton mit folgender Funktionalität:

```
QString searchString = ui->lineEdit->text();
ui->textEdit->find(searchString,
  QTextDocument::FindWholeWords);
myScene.clear();
QGraphicsEllipseItem *ell = myScene.
    addEllipse(random()%100, random()%100,
    random()%20+20, random()%20+30);
ui->graphicsView->setScene(&myScene);
ui->graphicsView->show();
```

- versuchen Sie nachzuschlagen, was diese Zeilen bewirken und welche include-Dateien Sie zur Übersetzung einbinden müssen
- Hinweis: Sie müssen in der Klasse mainWindow eine Variable vom Typ QgraphicsScene anlegen

Ergebnis


MainWindow
Text:
Studieren
Finde und Male
Lade Datei
<html>
<head>
<title>Studieren in
Greifswald</title>
<link rel="stylesheet"
type="text/css"
href="meinDesign.css">
</head>
Springe zu
meiner Überschrift
gaga
<h1>Studieren in Greifswald</h1>
<h2>Studiengänge</h2>
... blablabla ...
<h2>Freizeitmöglichkeiten</

Plan für den Routenplaner

- Im Folgenden soll für Ihre fertiggestellte Implementierung von Dijkstras Algorithmus ein GUI erstellt werden, d.h.
  - Darstellung der Landkarte
  - Auswahl von Start und Ziel von der Landkarte
  - Berechnung des kürzesten Pfades und Visualisierung desselben
- Wir werden dafür Daten aus dem Openstreetmap-Projekt (OSM) nutzen (http://www.openstreetmap.org)

Plan für den Routenplaner

- Eine Möglichkeit, eine Karte darzustellen ist es, basierend auf den Knotenkoordinaten, den Graph in ein Fenster des GUI zu zeichnen; das ist sehr aufwändig und es ist nicht so einfach, eine optisch ansprechende Darstellung zu erreichen
- Alternativ können wir die „Kartenzeichnungen“ nutzen, die OSM bereits von den Kartendaten angefertigt hat; das werden wir machen. Basierend auf diesen Kartenzeichnungen müssen wir dann später nur noch den berechneten Pfad im Fenster einzeichnen.

OSM Tiles/Kacheln

- Das OSM Projekt hat für die Kartendaten bereits Tiles/Kacheln vorberechnet, die aus dem Internet heruntergeladen werden können
- Beispiel: Die Koordinaten (Längen-/Breitengrad) von Wampen sind:
  13.420614309094/54.128108107232
- Mithilfe der folgenden Funktion können Sie die Koordinaten in „Kachelnummern“ umrechnen:
  - int long2tile(double lon, int z)
    { return (int)(floor((lon + 180.0) / 360.0 * pow(2.0, z))); }
  - int lat2tile(double lat, int z)
    { return (int)(floor((1.0 - log(tan(lat * M_PI/180.0) + 1.0 / cos(lat * M_PI/180.0)) / M_PI) / 2.0 * pow(2.0, z))); }

OSM Tiles

- Beispiel:
  Wampen in Zoomstufe 12 hat x-Tile-Koordinate 2200 und y-Tile-Koordinate 1312
- Es kann runtergeladen werden unter
  http://tile.openstreetmap.org/12/2200/1312.png

Aufgabe

- Machen Sie sich mit der Repräsentation der Tiles von Openstreetmap (GoogleMaps funktioniert übrigens ähnlich) vertraut auf http://wiki.openstreetmap.org/wiki/Slippy_map_tilenames
- Finden Sie in der QtCreator-Online-Hilfe heraus, wie Sie Dateien aus dem Internet laden können
- Modifizieren Sie dann Ihr GUI Programm so, dass Sie OSM-Tiles darstellen können, heraus- bzw. hereinzoomen und den dargestellten Ausschnitt verschieben
  - Einfach: legen Sie in QtCreator entsprechende „Buttons“ an, die den Bildschirmausschnitt verschieben und einen „Schieberegler“, der den Zoomlevel ändert
  - Alternativ, aber aufwendiger: Versuchen Sie herauszubekommen, wie Sie die Karte mit der Maus „ziehen“ bzw. den Zoomlevel mit dem Mausrad ändern können

Laden von Dateien aus dem Internet

- Legen Sie Objekte vom Typ QNetworkManager, QNetworkRequest und QNetworkReply an, setzen Sie die gewünschte URL und führen Sie die Anfrage aus:

```
QNetworkAccessManager myManager;
QNetworkRequest myReq;
myReq.setUrl(QUrl(''http://www.google.de''));
QNetworkReply *myReply;
myReply=myManager->get(myReq);
```

- Das Objekt vom Typ QNetworkAccessManager „managt“ alle Verbindungen ins Internet, QNetworkRequest bzw. -Reply sollten selbsterklärend sein
- es bietet sich an, einen QNetworkAccessManager einmal im Programm zu instantiieren und diesen dann wiederzuverwenden (siehe später das Demoprogramm)

Laden von Dateien aus dem Internet

- Um bei Fertigstellung der Internetanfragen benachrichtigt zu werden, müssen Sie entsprechende „Handler“ für das Signal „wir sind fertig“ definieren:

```
connect((const QObject*)myReply, SIGNAL(finished()),
this, SLOT(requestFinished()));
```

- es sollte in Ihrer aktuellen Klasse daher eine Methode vom Typ requestFinished() vorhanden sein
- sobald die Internetanfragen (im Hintergrund) fertig sind (alle Daten da) wird die Methode requestFinished() aufgerufen.
- in Ihrem Fall sollten Sie hier die Darstellung des gerade heruntergeladenen Tiles in eine GraphicsScene anstoßen (siehe Demo-Programm)

Laden von Bilddateien aus dem Internet

- Sobald eine Internetanfrage erfolgreich abgearbeitet wurde, können Sie die Antwort des Servers wie folgt auslesen (falls es sich dabei um ein Bild im PNG-Format – wie bei uns – handelt):

```
QByteArray content=myReply->readAll();
QPixmap myMap;
myMap.loadFromData(content);
```

- die geladene Pixmap können Sie per ::addPixmap() Ihrer QGraphicsScene hinzufügen und ggf. auch verschieben (→ Demoprogramm)

Beispielprogramm

- Schauen Sie sich das Beispielprogramm auf der Veranstaltungswebseite (Netzwerk-Demo)
- Sie können das „Archiv“ (eine gepackte Datei, welche alle Dateien des Projektes enthält) runterladen und mit
  `tar -xjf <Archivname>`
  in einem beliebigen Unterverzeichnis von Ihnen auspacken (oder Sie benutzen ein Archivprogramm mit GUI)
- Sie können dann das Projekt entweder in qtcreator laden (die entsprechende .pro-Datei öffnen) oder auf der Kommandozeile im Verzeichnis
  `qmake`
  bzw.
  `qmake-qt4` (auf Ubuntu, wenn auch Qt3 installiert ist)
  aufrufen

Vorschau

- Als nächstes werde ich Ihnen die OSM-Daten im für Ihre Dijkstra-Implementierung kompatiblen Format zur Verfügung stellen
- Sie können dann darauf basierend kürzeste Wege berechnen und diese zusammen mit den OSM-Kacheln visualisieren