Praxis des Programmierens

SS 2008

Mi, Do 16-18 im RTK
(Vorlesung/Übung gemischt)

Prof. Dr. Stefan Funke
Lehrstuhl für Informatik
Universität Greifswald

stefan.funke@uni-greifswald.de

Vorlesungsinfos/Slides:
http://www.math-inf...de/informatik/COURSES/PraxProg08.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

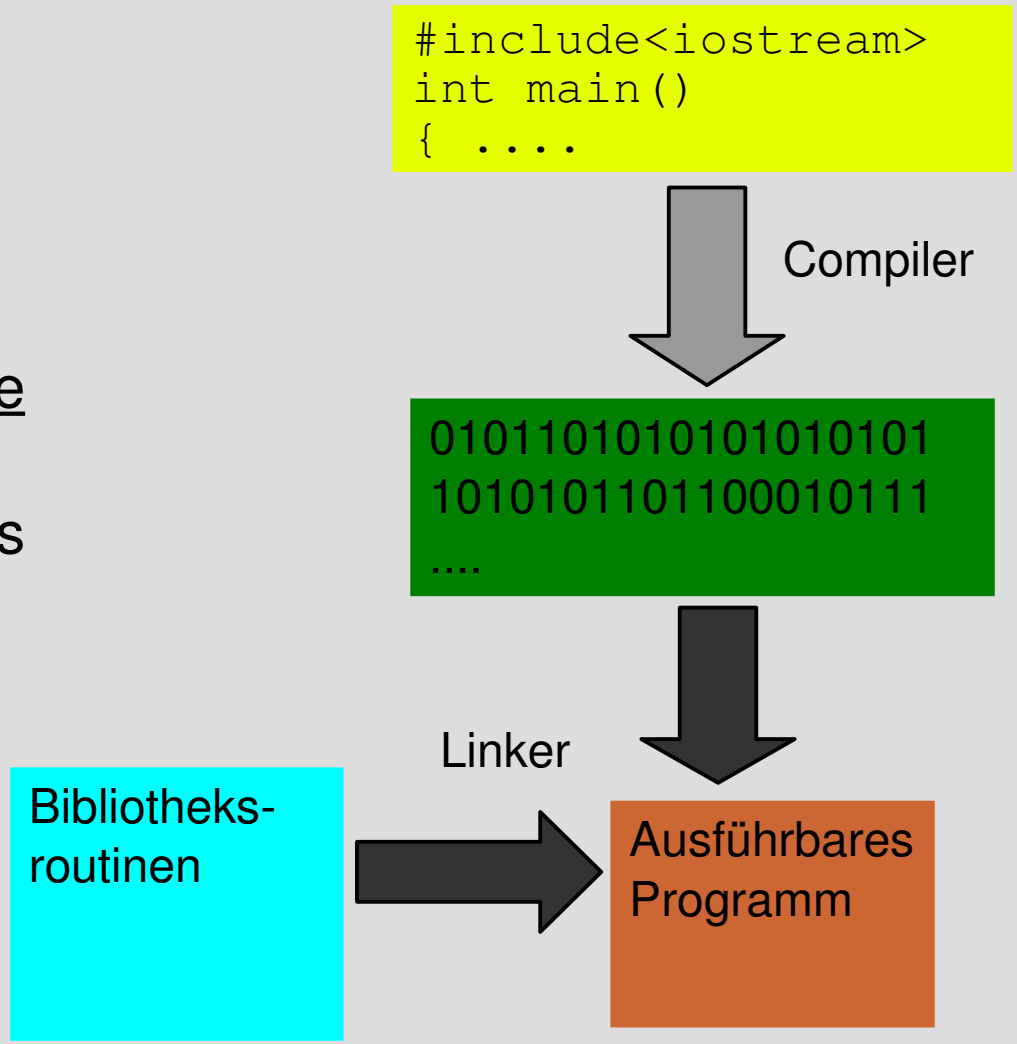
Ein Beispiel für ein C(++) Programm

```
// 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`

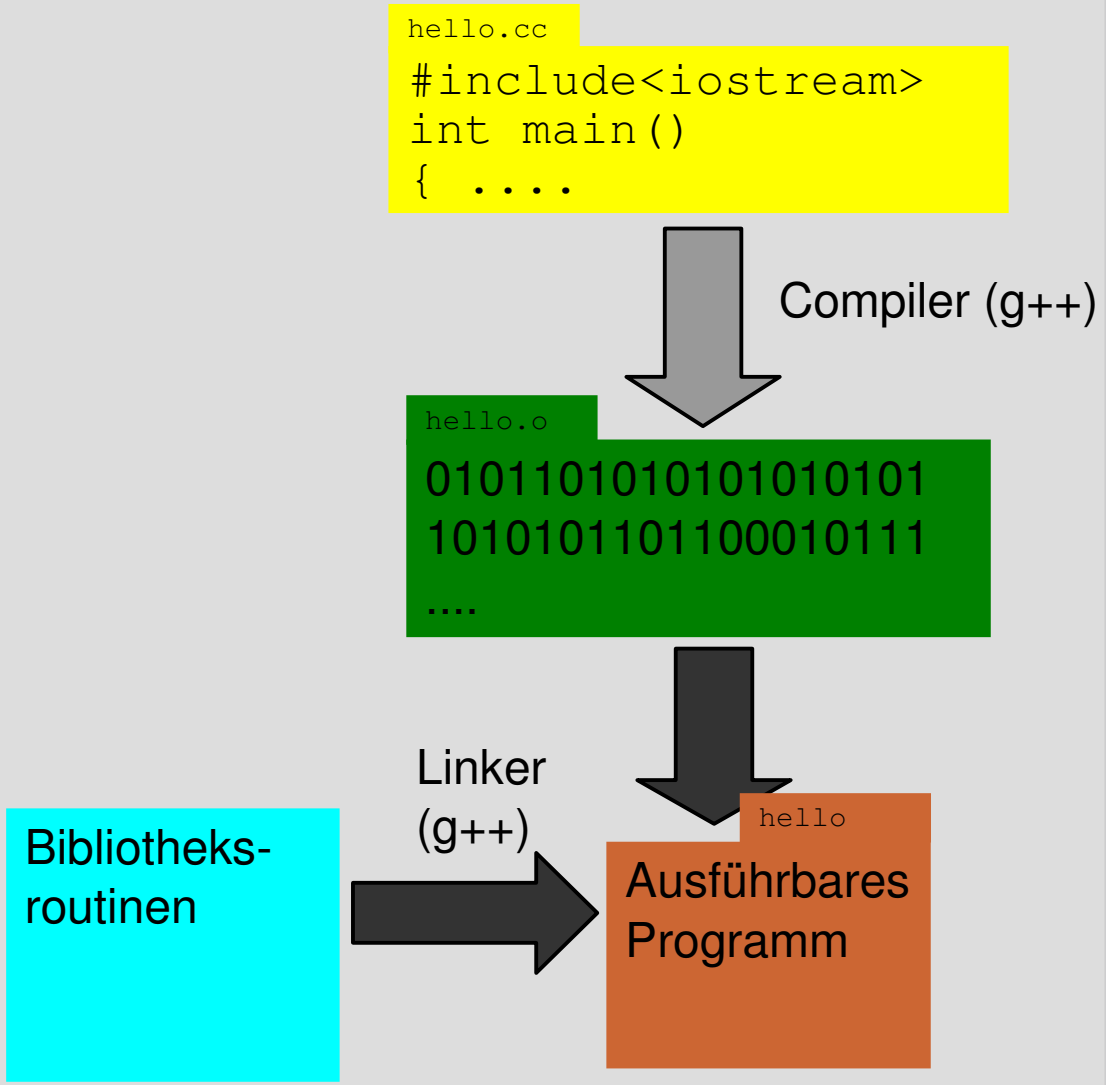
erzeugt Objectfile `hello.o`

Starten des Linkers mit

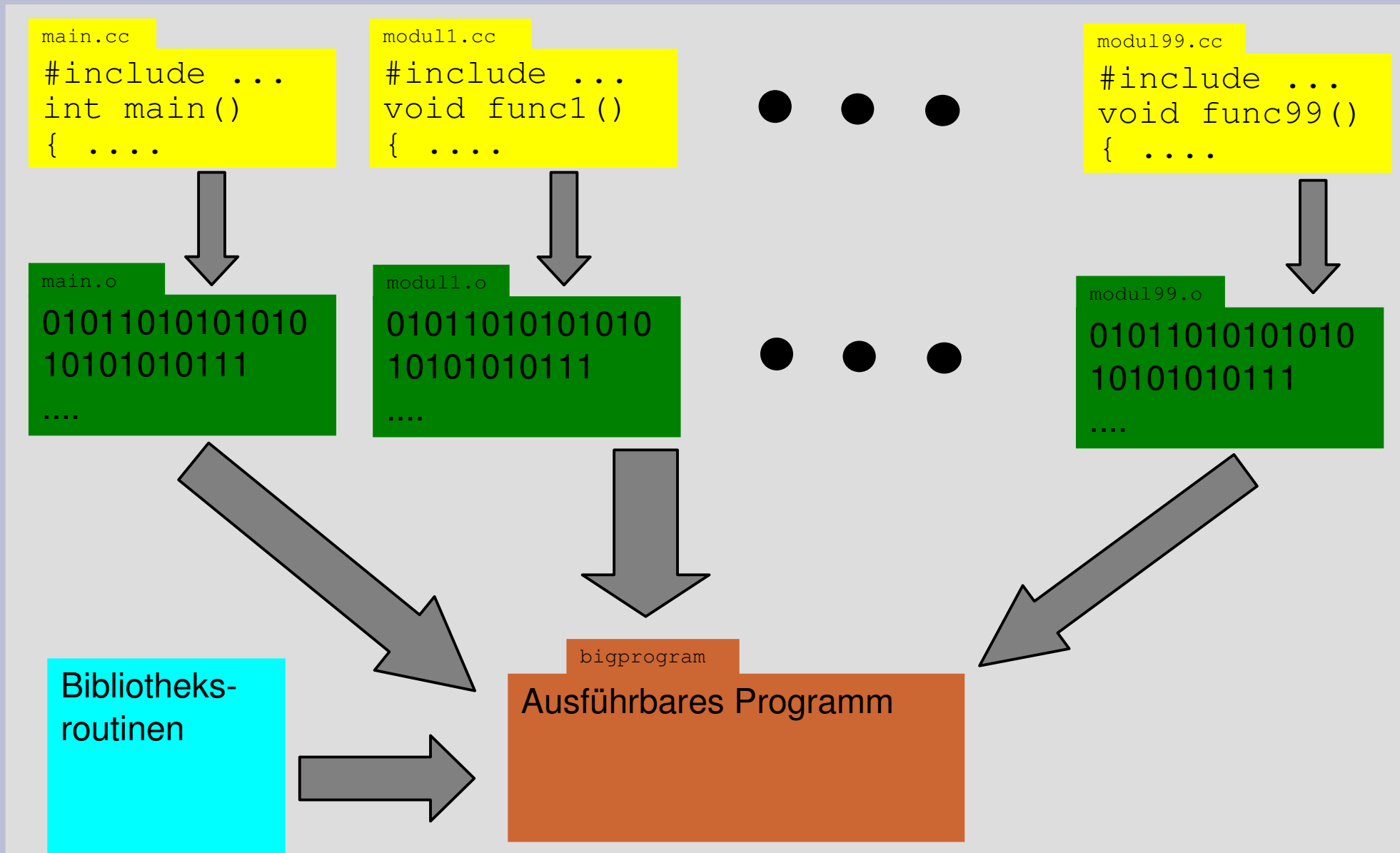
`g++ hello.o -o hello`

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

Remote-Login (nur beim allerersten Mal)

- Sie werden alle auf meinem Server arbeiten (IP-Adresse: 141.53.34.115)
- Sie müssen sich zunächst auf dem Server einloggen:
  `ssh username@IP-Adresse`
  ssh baut eine verschlüsselte Verbindung zum Server auf
- Ändern Sie dann als aller erstes Ihr Passwort mittels :
  `passwd`
  und merken Sie sich Username und neues Passwort!
- Loggen Sie sich wieder aus:
  `exit`

Remote Login (jedes Mal)

- Einloggen auf dem Server (TNR=Ihre Teilnehmernummer):
  `ssh -C -L 5904:localhost:5900+TNR username@IP-Adresse`
- starten Sie einen virtuellen Desktop auf dem Server
  `vncserver -geometry 900x600 -depth 24 :TNR`
  ggf. müssen Sie ein Passwort angeben
  (das müssen Sie später nur wiederholen, wenn der Server neu gebootet wurde, andernfalls kommt eine Fehlermeldung, die Sie ignorieren können)
- öffnen Sie eine neue Shell auf Ihrem **lokalen Rechner**
- zeigen Sie den virtuellen Desktop auf Ihrem lokalen Rechner an:
  `vncviewer :4`
- ab jetzt arbeiten Sie auf dem virtuellen Desktop, der auf dem Server läuft!
- Öffnen Sie eine Shell auf dem Server

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
```

Aufgabe 2

- schließen Sie die Ansicht des virtuellen Desktops (Klick rechts oben)
- beenden Sie die ssh-Verbindung (`exit`)
- fahren Sie den Rechner runter
- fahren Sie den Rechner wieder hoch und verbinden Sie sich mit dem Remote-Desktop
- schauen Sie nach, ob noch alles so wie vorher ist
- gehen Sie nach Hause

Virtuellen Desktop anzeigen und Editor starten

- loggen Sie sich wie das letzte Mal gezeigt ein, Sie brauchen dazu Username, Passwort, Teilnehmernummer
- Sie können die Slides auf dem lokalen (!) Rechner in einem Browser anzeigen
- starten Sie auf dem virtuellen Desktop eine Shell und rufen Sie den Editor auf mit

  `kate &`

  (das „&“ startet das Programm im Hintergrund

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 im Editor 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` liest Werte ein

```
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

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

Vektoren/Arrays

- Mittels
  `int A[20];`
  legen Sie einen Vektor/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

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: Beispiel

Call-by-Value:

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

Call-by-Reference:

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

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 Vektoren/Arrays

- Übergeben eines Vektors 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

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

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
- andere Module, die Ihre Sortierroutine benutzen möchten, können sich über das Interface im entsprechenden Header-File informieren (`bbsort.h`), d.h. dieses mittels `#include` einbinden

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

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

Aufgabe

- Schreiben Sie eine Klasse, welche Tankstellen (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 in
  `/home/PROJEKT/MiniGraph.gtxt`
- Format: siehe `README.gtxt`

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 (z.B. Wikipedia)
- 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 in
  /home/PROJEKT/MA.gtxt
  (das sind die Straßendaten von Massachusetts)

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

Qt Tutorial

- Arbeiten Sie sich durch das Tutorial für Qt unter http://doc.trolltech.com/4.3/tutorial.html durch
- falls Sie Ihre eigene Linuxinstallation haben, stellen Sie sicher, dass Sie das Paket libqt4-dev (und seine Abhängigkeiten) installiert haben

HiWi-Stellen zu vergeben

Im Rahmen eines durch den
Ideenwettbewerb 2008
geförderten Projektes
"Ö-Web – LowCost Flottenmanagement"
sind ab ca. Juli 2008 insgesamt
100 Hiwi-Stunden
Programmierarbeit zu vergeben
Aufgabe ist die Programmierung einer Web2.0/Ajax
Anwendung mit dem
Google Web Toolkit
Java-Kenntnisse sind von Vorteil

Aufgabe für Leute die schon wieder nix zu tun haben

- Ihre Implementierung von Dijkstra als auch die Visualisierung läuft (hoffentlich) halbwegs rund auf dem Server/Ihrem PC/Notebook
- Leider wird Ihre Implementierung auf Navigationssystemen oder gar Handys nicht besonders flüssig laufen
- Hauptgrund ist der in Navigationssystemen sehr begrenzte Hauptspeicher

Hauptspeicherbelegung

- Schätzen Sie überschlagsmäßig ab, wieviel Speicher Ihre Darstellung des deutschen Straßengraphen (ca. 4 Mio. Knoten, 8 Mio. Kanten) benötigt (meine vorgeschlagene Repräsentation braucht etwas mehr als 100 MB); darin sind Daten wie Straßennamen etc. noch nicht enthalten
- Im Folgenden werden wir zwei Ansätze betrachten, die diese große Hauptspeicherbelegung reduzieren könnten
- Wählen Sie **einen** davon aus und setzen ihn um

Ansatz 1: Datenkompression

- Beim Betrachten der Daten ist relativ viel Redundanz zu beobachten, Beispiele:
  - Aufeinanderfolgende Knoten haben typischerweise sehr ähnliche Koordinaten; dennoch werden die Koordinaten immer vollständig gespeichert
  - Die ausgehenden Kanten eines Knoten enden meist in Knoten mit KnotenID "in der Nähe", dennoch wird immer die vollständige KnotenID gespeichert

Ansatz 1: Datenkompression

- Ein Ansatz ist es, diese Redundanzen zu minimieren, indem wir:
  - Nicht für jeden Knoten seine vollständigen Koordinaten speichern, sondern nur Delta-Werte, d.h. wenn Knoten i Koordinaten (x_i, y_i) und Knoten (i+1) Koordinaten (x_(i+1), y_(i+1)) hat, speichern wir für Knoten (i+1) nur (x_(i+1)-x_i, y_(i+1)-y_i)
  - Genauso können wir für die Zielknoten einer Kante "Delta-Werte", d.h. Die ID-Differenz zum Startknoten speichern

Ansatz 1: Datenkompression

- Mögliche Vorgehensweise:
  - Approximieren Sie zunächst die float-Koordinaten der Knoten durch Brüche
  - Berechnen Sie sowohl Delta-Werte der Koordinaten als auch Delta-Werte der Zielknoten
  - Schauen Sie sich an, wie die Verteilung der Delta-Werte aussieht
  - Konstruieren Sie einen Huffman-Code für die Deltawerte und benutzen Sie diesen zur Speicherung derselben
  - Passen Sie Ihre Graphdatenstruktur und ggf. auch Ihre Dijkstraimplementierung an
  - Evaluieren Sie die Laufzeiten im Vgl. zur unkomprimierten Version

Ansatz 1: Datenkompression

- Ein Problem, das Sie u.a. haben werden:
  - Die mit einem Knoten/einer Kante assoziierten Daten werden durch den Huffman-Code nicht mehr fixe Größe haben; dadurch wird es schwierig, z.B. die Koordinaten von Knoten i oder den Zielknoten von Kante j direkt zu addressieren
  - Möglichkeit der Auflösung: "Gruppieren" Sie Knoten bzw. Kanten immer in z.B. 8er Gruppen, speichern Sie nur für die erste Kante/den ersten Knoten die vollständige(n) Zielknotendarstellung/Koordinaten, für die folgenden 7 nur die entsprechenden Deltas
  - Experimentieren Sie mit der Gruppengröße und vergleichen Sie den Laufzeitverlust

Ansatz 2: Externspeicher

- Eine weitere Art und Weise, die Hauptspeicherbelegung zu reduzieren, ist es, den Graph "extern" auf der Festplatte zu speichern, und nur bei Bedarf die notwendigen Informationen von dort in den Hauptspeicher zu transferieren
- Hinweis: Sie können mittels ifstream::seek() einen beliebige Position in einer geöffneten Datei suchen und dann von dort beginnend Daten einlesen
- Modifizieren Sie Ihre Graphrepräsentation entsprechend und vergleichen Sie die Laufzeiten der "externen Graphspeicherung" mit der Version Ihres Programmes, welches alle Daten im Hauptspeicher vorhält
- Erklären Sie die Laufzeitunterschiede