LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

mobile and distributed systems group

Grundlagen C und C++

Einheit 01: Organisation & Einführung

Lorenz Schauer
Lehrstuhl für Mobile und Verteilte Systeme

Heutige Agenda

Organisatorisches

- Ziele, Aufbau und Inhalte
- Zielgruppe, Vergütung, Webseite
- Kontakt

Einführung in C und C++

- Motivation
- Einführung in C
- Grundlagen
  - Aufbau, Ein- und Ausgabe, Schleifen

Programmieren mit C

- Installation
- Der Weg zum ersten Programm
  - Hallo Welt!
- Übungen in C

Lernziele

- Motivation zur Programmierung mit C und C++
- Installation der notwendigen Programme
- Erste C-Programme schreiben und ausführen können

Organisatorisches zum Kurs

- **Ziele**:
  - Grundlagen zu C und C++ kennenlernen und verstehen
    - Keine Vorkenntnisse notwendig!
    - **Aber**: Grundsätzliches Verständnis fürs Programmieren empfohlen!
  - Praktische Einarbeitung in die Programmierung mit C und C++
    - Kleinere Programme sollen selbstständig entwickelt und ausgeführt werden können
  - Vermittlung von notwenigen theoretische Grundlagen und Basis-Konzepte der Programmierung mit C und C++

- **Aufbau**:
  - Mischung aus Vorlesung und praktischen Programmiereinheiten
  - Kleinere Programmieraufgaben müssen während der Veranstaltung selbstständig gelöst werden (ggf. mit Hilfestellung)
    - Bitte bringen Sie daher auch immer Ihr eigenes Gerät (Laptop) mit!

Kursorganisation

- **Zielgruppe:**
  - Studenten der Informatik bzw. mit Nebenfach Informatik, die ihr Wissen in dem angebotenen Thema vertiefen wollen.

- **Vergütung**:
  - Der Kurs stellt ein freiwilliges Zusatzangebot zur Verbesserung der Lehre dar
  - Keine Vergütung!

- **Ort und Zeit:**
  - Insgesamt 5 Einheiten:
    - Vom 23.11. bis einschl. 21.12.2015 immer Montags, von 18.00 -20.00 Uhr c.t.
    - Hauptgebäude (Geschwister-Scholl-Platz 1), Raum: M010

- **Webseite**:
  - http://www.mobile.ifi.lmu.de/lehrveranstaltungen/grundlagen-c-und-c/
  - Keine Anmeldung notwendig

Kursinhalte (C)

- Insgesamt 5 Einheiten zu folgenden Themen:
  - 2 Einheiten zu C
  - 3 Einheiten zu C++

  - 23.11. (Heute):
    - Organisatorisches, Einführung, Installation der Tools, Hello World in C
    - Grundlagen in C
      - Nutzereingaben, Kontrollstrukturen, …
      - Übungsaufgaben

  - 30.11.:
    - Weitere Grundlagen in C
      - Typedef, Enum, Struct, Pointer, Arrays
    - Auf dem Weg zur Objektorientierung
      - Heap und Stack
      - Vorbereitung auf C++

Kursinhalte (C++)

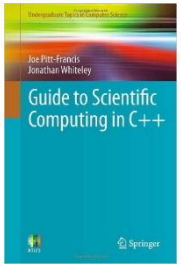
- 07.12.:
  - Grundlagen C++
    - Erstes Programm
    - Variablen, Kontrollstrukturen
    - Übungen
- 14.12.:
  - File input und output
  - Speicherverwaltung und Pointer
  - Übungen
- 21.12.:
  - Objektorientierung mit C++
    - Einführung in Klassen
    - Übungen

Kontakt

- **Veranstalter**:
  - Lorenz Schauer (Wiss. Mitarbeiter)
    - Büro:
      - Lehrstuhl für Mobile und Verteilte Systeme
        Oettingenstraße 67, Raum U160
    - Sprechstunde:
      - Montags, 10 - 12 Uhr
      - Donnerstags, 14.00 - 16.00 Uhr
    - Kontakt:
      - Mail: lorenz.schauer@ifi.lmu.de
      - Tel.: 089-2180-9157
      - Web: http://www.mobile.ifi.lmu.de/team/lorenz-schauer/

Literaturhinweise (Kostenlos)

Elias Fischer:
C-HowTo – Programmieren in C
Deutschsprachiges Online-Tutorial, auch als kostenloser Download unter: http://www.c-howto.de/
Print-Version als Buch (kostenpflichtig) bestellbar



Joe Pitt-Francis:
Guide to Scientific Computing in C++
(Undergraduate Topics in Computer Science)
Kostenlos für LMU Studenten über E-Book-Programm der UB:
http://ebooks.ub.uni-muenchen.de/29027/

Unzählige kostenlose online Tutorials mit Erklärungen und Übungen zu C bzw. C++ in deutsch und englisch vorhanden. Just google it…

Beispiele:

- http://c.learncodethehardway.org/book/
- http://www.online-tutorials.net/c-c%2B%2B-c/c%2B%2B-tutorial-teil-1/tutorials-t-1-58.html

Teil 1: Einführung in C und C++

Motivation: Warum Programmieren?

Grundproblem:

Menschen besitzen komplexe Sprache (Subjekt – Prädikat – Objekt). Das Verstehen von Maschinencode ist umständlich

0011 1100 0001 1101 1100 1000
0110 0011 1010 1011 1001 1100
0010 0101 0011 1110 0110 0011
0011 01……

???

Maschinen besitzen eine sehr primitive Sprache (Befehlsfolge) und können nur Maschinensprache verstehen:

0011 1100 0001 1101 1100 1000
0110 0011 1010 1011 1001 1100
0010 0101 0011 1110 0110 0011
0011 01……

Motivation: Warum Programmieren?

Lösung:

- Menschen nutzen einen anderen Befehlssatz als Sprache L1 (Bsp.: C++, Java, Python, usw.)
- Maschinen verarbeiten eine Übersetzung (Compiler) bzw. Interpretation (Interpreter) von L1, hier als L0 bezeichnet.

- **Compiler**:
  - Vollständige Übersetzung des Programms von L1 zu L0
  - Quellprogramm in L1 wird verworfen
  - Zielprogramm in L0 wird in Speicher geladen und ausgeführt

- **Interpreter**:
  - Jede L1 Anweisung wird analysiert, dekodiert und unmittelbar in L0 ausgeführt
  - Quellcode + Interpreter auf ausführender Maschine benötigt
  - I.d.R. langsamer als kompilierte Programme

Warum C?

Gründe für die Programmiersprache C

- Geschichtliches:
  - C wurde 1978 von Kernighan und Ritchie veröffentlicht, um Betriebssysteme unabhängig von ihrer Hardware schreiben zu können
    - Bsp.: UNIX ist in C geschrieben
  - Alles sollte möglich sein, was auch in Assembler möglich ist, nur eben auf eine maschinenunabhängige Art
  - C verbreitete sich sehr schnell und wurde ständig weiter entwickelt
    - Standardisierung durch ANSI bzw. ISO
    - C11 wurde am 8. Dezember 2011 veröffentlicht
  - Ist heute eine der meist verbreitetsten Sprachen

- Verwendung:
  - Systemprogrammierung (Betriebssysteme, eingebettete Systeme)
  - Anwendungen
  - Compiler, Programmbibliotheken
  - Unterstützung (Zwischencode) für höhere Programmiersprachen

Warum C?

Eigenschaften von C

- Ist einfach aufgebaut und extrem vielseitig
  - Geringe Menge an Schlüsselwörtern
  - Ermöglicht direkte Speicherzugriffe und hardwarenahe Konstrukte
  - Hohe Ausführungsgeschwindigkeit und geringe Codegröße

- Kaum Einschränkungen bei Speicherzugriffen
  - Mächtig, aber auch sicherheitskritisch
    - Bsp.: Buffer-Overlow
  - Compiler kann nur eingeschränkt bei Fehlersuche helfen

- Wenn man C kann, sind Programmiersprachen wie C++, Java, Perl oder PHP einfacher zu erlernen

Warum C++

Gründe für die Programmiersprache C++

- Geschichtliches:
  - 1979 von Bjarne Stroustrup als Erweiterung der Programmiersprache C entwickelt.
  - Eine der ersten Erweiterungen war ein Klassenkonzept mit Datenkapselung
  - 1983 wurde *C with Classes* in C++ umbenannt.
  - 1985 erschien die erste Version und 1988 die Version 2.0 von C++
    - Erweiterungen: Mehrfachvererbung, Abstrakte Klassen, …
  - ISO Standardisiert. Heute: C++14 (Seit Januar 2015)
  - Zählt zu den verbreitetsten Programmiersprachen

- Verwendung
  - Systemprogrammierung
    - Betriebssysteme, eingebettete Systeme, Treiber, virtuelle Maschinen, …
  - Anwendungsprogrammierung
    - Anwendungen mit hohen Forderungen an Effizienz

Warum C++?

Eigenschaften von C++

- Erweitert die Sprache C um objektorientierte Programmiereigenschaften
  - Weitere Datentypen, Klassen, Mehrfachvererbung, Templates, usw.
  - C++ Standardbibliothek

- Ermöglicht effiziente und maschinennahe Programmierung
- Aber auch eine Programmierung auf hohem Abstraktionsniveau
- Sehr vielseitig einsetzbar
- Breites Leistungsspektrum
- Komplex und daher evtl. lange Einarbeitungszeit

Teil 2: Einführung in C/C++

Installation und Hello World!

Installation – Überblick

Alles was wir brauchen:

- C bzw. C++ Compiler
- Texteditor bzw. IDE (Integrated Development Environment)

C/C++ Compiler:

- Der in Microsoft Visual C++ enthaltene Compiler ist einer der verbreitetsten für Windows.
- Der g++ aus der GNU Compiler Collection (GCC)
  - ist quelloffen und frei verfügbar
  - verfügbar für Unix, Linux, Mac OS X, Windows und AmigaOS
  - existiert seit 1987 und ist somit einer der ältesten C++-Compiler
- Weitere, wie:
  - Intel C++ Compiler
  - Comeau C++
  - …

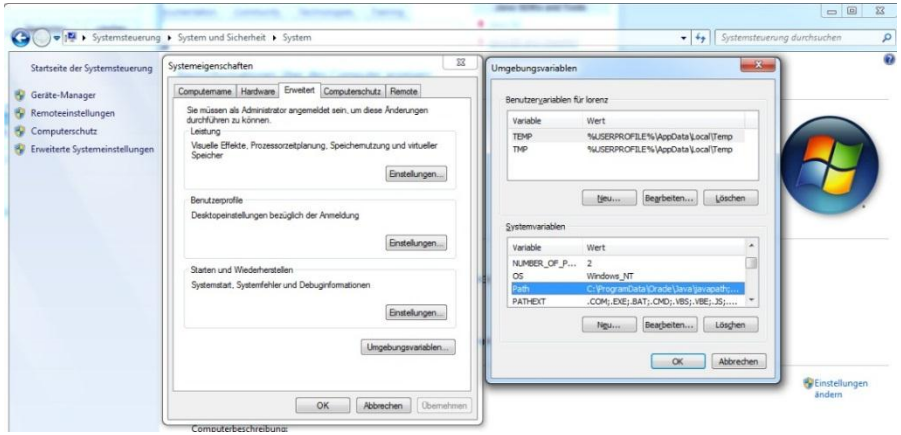
Installation (Für Windows)

GCC unter Windows

- Einfache Installation über MinGW (Minimalist GNU for Windows)
  - http://www.mingw.org/
- Mindestens *MinGW base tools* installieren

Schritt 2: Ausführungspfad setzten

- Systemsteuerung -> System und Sicherheit -> System -> Erweiterte Systemeinstellungen auswählen -> Unter dem Reiter „Erweitert“ die Schaltfläche Umgebungsvariablen anklicken
- Pfad des MinGW/bin eintragen. Bsp.:
  - C:\Program Files\MinGW\bin
  - Trennung mit ;



Installation (Für Linux bzw. Mac)

Installationshinweise für OS X unter:

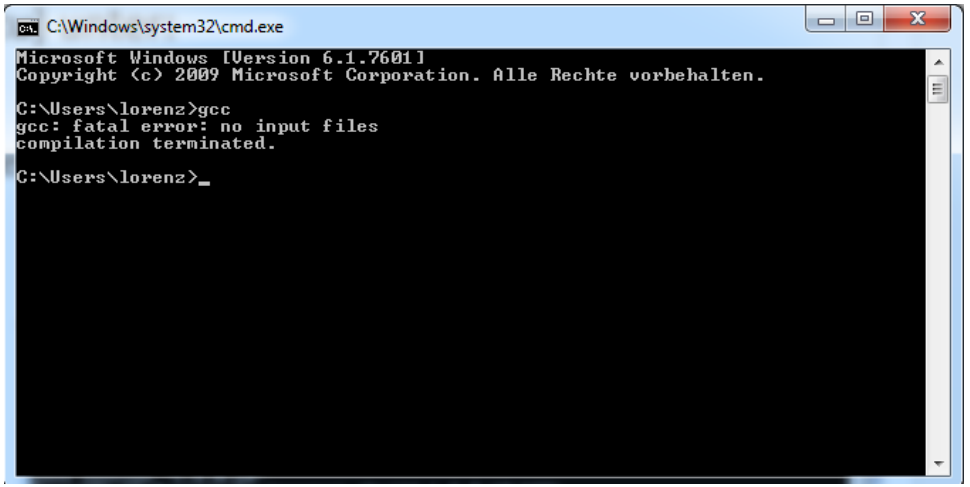
- http://wiki.freitagsrunde.org/GCC_unter_Mac_OS
- Entweder mit XCode, oder mit Brew (Paketmanager): *brew install gcc4.8*

Installationshinweise für Linux (Ubuntu) unter:

- https://wiki.ubuntuusers.de/GCC
- Entweder bereits installiert oder per *sudo apt-get install build-essential* nachinstallieren

Schritt 3: Installation testen

- In Konsole eingeben: gcc
- Ergebnis:
  „gcc: fatal error: no input files“
  „compilation terminated“



Der Weg zum ersten Programm

3 Schritte sind zu durchlaufen:

- Erstellen des Quellcodes
- Kompilieren mittels *gcc programm.c –o programm*
- Starten des Programms
  - Windows: Ausführen der .exe
  - Linux: Kompilat ausführbar machen mit *chmod +x programm* und dann ausführen

Erstellen des Quellcodes

Der Quellcode kann mit jedem beliebigen Texteditor erzeugt werden.

- Bsp.: Notepad++ (Windows https://notepad-plus-plus.org/download/v6.8.3.html)
- Bsp.: Geany (für Linux)

- C-Datei anlegen:
  - Bsp.: MeinProgramm.c
  - Bei Windows: Dateiendung einblenden, um .c statt .txt zu erzeugen

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(){

    puts("Willkommen beim lustigen Zahlenraten\n");

    srand(time(NULL));
    int meineZahl = rand() % 10;

    //printf("Die zahl ist: %d\n",meineZahl);

    int erraten = 0;

    while(!erraten){

        printf("Bitte geben Sie eine Zahl zwischen 0 und 9 ein: ");
        fflush(stdout);

        int a;
        scanf("%d",&a);

        if (a == meineZahl){
            printf("Super!");
            erraten = 1;
        }

        else
            printf("Schade! Bitte nochmal versuchen!\n");
```

Entwicklungsumgebungen

Einfache Texteditoren sind ausreichend aber umständlich.

- Daher: Unterstützung beim Programmieren durch IDEs

Eine IDE (*Integrated Development Environment*) bietet i.d.R.:

- Texteditor
- Compiler bzw. Interpreter
- Linker
- Debugger
- Quelltextformatierungsfunktion

Vorteile:

- bietet viel mehr Features und Funktionen als einfache Editoren.
  - Syntax-Highlighting, Auto-Verfollständigung, Such-Funktionen, usw.
- spart viel Zeit bei der Programmierung.
- erkennt Tipp- sowie Syntaxfehler.
- ermöglicht Projektmanagement und Teamarbeit.

Kostenlose C/C++ Entwicklungsumgebung:

- Eclipse IDE for C/C++ Developers:
  - https://eclipse.org/downloads/

- Oder als Eclipse-Plugin für eine bestehende Version
  - http://www.eclipse.org/cdt/downloads.php

- Microsoft Visual Studio
  - Kostenlos für Studenten der LMU über MSDNAA
  - Informationen bei der RBG: http://www.rz.ifi.lmu.de/Dienste/MSDNAA/
  - Sehr umfangreich und für verschiedene Sprachen einsetzbar

Hello World!

```
// Dieser Code sollte nun bei allen laufen

#include<stdio.h>

int main() {

   // Gibt Servus Welt aus…
   printf(“Servus Welt!");

   return 0;

}
```

Teil 3: Grundlagen C

Aufbau, Ein- und Ausgabe, Kontrollstrukturen

Grundsätzlicher Aufbau eines C-Programms


```
// Für den Präprozessor: Einbinden von Definitionsdateien

#include <stdio.h>      // Definitionen für Ein/Ausgabe
#include <conio.h>      // DOS console input/output (nicht ANSI Standard)

// Evtl. Deklarationen bzw. Definitionen:
#define pi 3.14

//Vor der main-Methode sollten verwendeten Funktionen deklariert werden.
int doSomeCrazyStuff(void);

//Definition von globalen Variablen
int zahl;
char zeichen = ‘a‘;

int main(){
   int lokale_variable = 0;
   puts(„Willkommen in der Main-Methode.“);
   lokale_variable = doSomeCrazyStuff();
   return 0;
}

int doSomeCrazyStuff(void){

   //..methoden-rumpf
}
```

C-Variablen

Variablen (Deklarieren und Initialisieren):

- `Datentyp name = Wert;`
  - Bsp.: `int a = 10;`

Konstante Variablen:

- `const datentyp name = Wert;`
  - Bsp.: `const double PI = 3.1415;`

Typumwandlungen:

- Implizit in einem gemischten Ausdruck
  - Bsp.: Ist ein Operator `double`, wird auch der andere in `double` umgewandelt.
- Explizit durch Type-casting:
  - `(typname) Ausdruck;`
  - Bsp.: `d = (double) n/3; // Wobei n vom Typ int ist`

C-Datentypen

Type	Keyword	Bytes	Range
character	**char**	1	-128 .. 127
unsigned character	**unsigned char**	1	0 .. 255
integer	**int**	2	-32 768 .. 32 767
short integer	**short**	2	-32 768 .. 32 767
long integer	**long**	4	-2 147 483 648 .. 2 147 483 647
unsigned integer	**unsigned int**	2	0 .. 65 535
unsigned short integer	**unsigned short**	2	0 .. 65 535
unsigned long integer	**unsigned long**	4	0 .. 4 294 967 295
single-precision floating-point (7 Stellen)	**float**	4	1.17E-38 .. 3.4E38
double-precision floating-point (19 Stellen)	**double**	8	2.2E-308 .. 1.8E308

Ausgabe von Zeichenketten

Formatierte Ausgabe von Zeichenketten mittels `printf()`:

- Aus Definitionsdatei `stdio.h`
- bsp.: `printf(„Servus Welt!“);`
- Gibt int-Wert zurück
  - Anzahl an ausgegebenen Zeichen
  - EOF im Fehlerfall
- Neben der Zeichenketten können Argumente durch Umwandlungsangaben `%` angegeben werden
  - Bsp.: `printf(„Die Zahl lautet: %d“, zahl);`

Zeichen	Argument; Ausgabe als
d,i	**int;**dezimale Zahl
o	**int;** oktale Zahl ohne Vorzeichen
x,X	**int;** hexadezimale Zahl ohne Vorzeichen, mit **abcdef** oder **ABCDEF** für 10, ..., 15.
u	**int;** dezimale Zahl ohne Vorzeichen
c	**int;** einzelnes Zeichen
s	**char *;** aus der Zeichenkette werden Zeichen ausgegeben bis vor **'\0'**, oder so viele Zeichen, wie die Genauigkeit ist.
f	**Gleitkomma;** [-]m.dddddd (die Genauigkeit gibt die Anzahl der Nachkommastellen an)
p	**Zeiger;** Gibt das Eingabeargument als Zeiger in hexadezimaler Form aus (near-Zeiger als YYYY, far-Zeiger als XXXX:YYYY).
%	es wird kein Argument umgewandelt; ein % wird ausgegeben

Eingabe

Formatierte Eingabe mittels scanf():

- Analog zu printf und in stdio.h definiert
- liest Zeichen aus der Standard-Eingabe
- interpretiert sie unter der Kontrolle der Zeichenkette
- legt die Resultate in den restlichen Argumenten ab.
  - Die Argumente müssen alle **Zeiger** sein!

```
//Beispiele für Nutzung von scanf():

#include <stdio.h>
int main(){

   int zahl;

   printf(„Bitte geben Sie eine Zahl ein: “);
   scanf(„%d“,&zahl); // Schreibt Nutzereingabe an die Adresse der Variablen
                      // zahl

}
```

Kontrollstrukturen

Wir besitzen alle gängigen Kontrollstrukturen einer modernen Programmiersprache (Bsp.: Java):

- Auswahlstrukturen: if-else, switch-case
- Wiederholungsstrukturen: for, while, do-while
- Sprungbefehle: break, continue
- Zusätzliche Sprungbefehle: goto (Mit Vorsicht zu behandeln!)

```
// Beispiel für goto-Sprungbefehl:

if (Bedingung) goto ZIEL;
//...

ZIEL:
Anweisung1;
Anweisung2;
//….
```

Aufgabe 1

Schreiben Sie ein C-Programm mit dem Namen *ZahlenDreieck.c*, welches folgende Ausgabe auf der Konsole ausgibt:

```
0
0 1
0 1 2
0 1 2 3
0 1 2 3 4
0 1 2 3 4 5
0 1 2 3 4 5 6
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8 9
```

- Verwenden Sie dazu die `while`-Schleife
- Hinweise:
  - Es können auch mehrere `while`-Schleifen genutzt werden
  - Mit dem String „`\n`“ wird ein Zeilenumbruch erzwungen

Aufgabe 2

Schreiben Sie nun ein weiteres C-Programm mit dem Namen *ZahlenRaten.c*, das folgende Aufgaben übernimmt:

- Ihr Programm denkt sich zunächst eine Zahl zwischen 0 und 10 aus
  - Hinweise:
    - rand() liefert eine ganzzahlige Zufallszahl im Bereich von 0 und 2^15 - 1.
    - srand() wird zur Initialisierung des Zufallszahlengenerators benutzt
    - Mit dem Modulo Parameter % können wir den Zufallszahlbereich eingrenzen
- Ihr Programm fordert nun den Nutzer immer wieder auf eine Zahl auf der Konsole einzugeben, bis der Nutzer die vom Programm ausgedachte Zahl korrekt erraten hat!
  - Gibt der Nutzer also eine falsche Zahl ein, meldet das Programm zurück, dass die eingegebene Zahl falsch ist und der Nutzer wird erneut aufgefordert eine Zahl einzugeben
  - Gibt der Nutzer hingegen die vom Programm ausgedachte Zahl ein, meldet das Programm „Herzlichen Glückwunsch“ und beendet die Ausführung