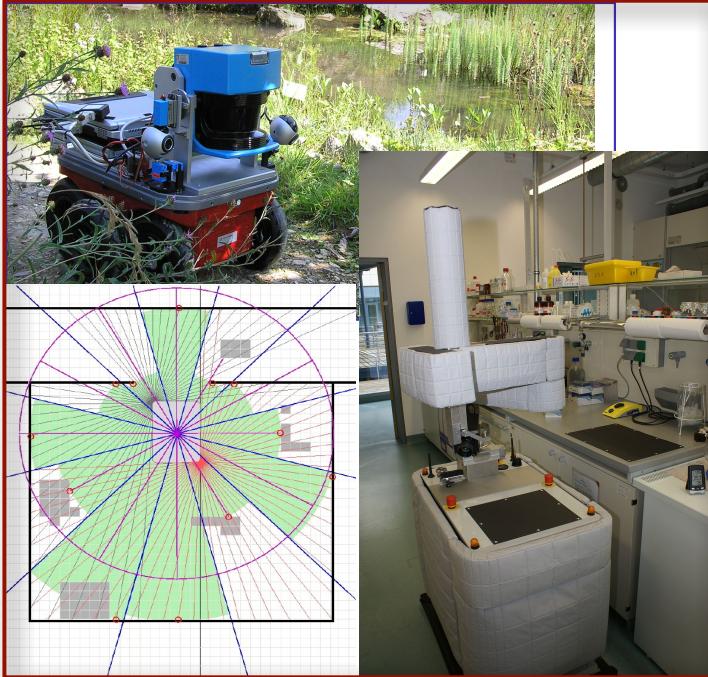


Einführung in die Programmiersprache C++

Thomas Wiemann
Institut für Informatik
AG Wissensbasierte Systeme

Forschungsthemen AG KBS



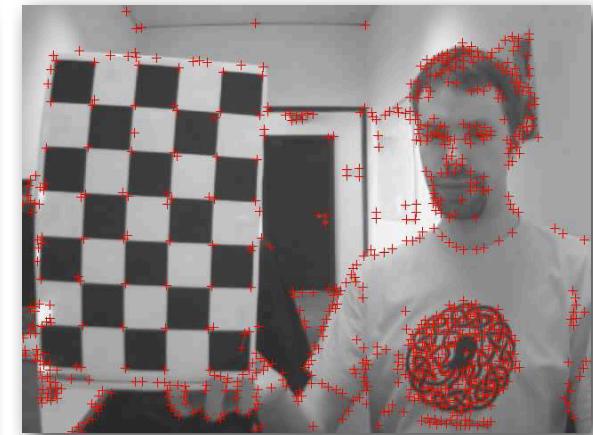
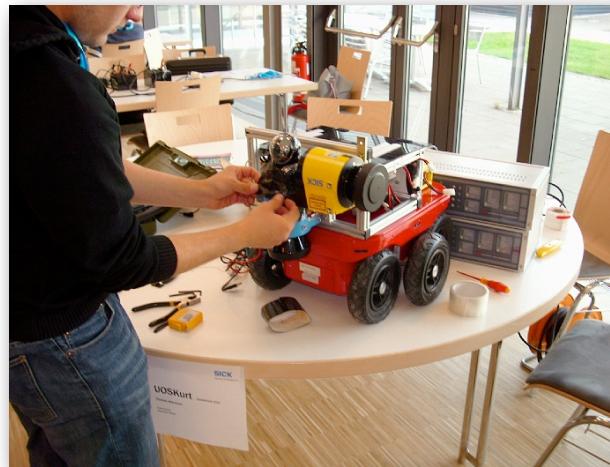
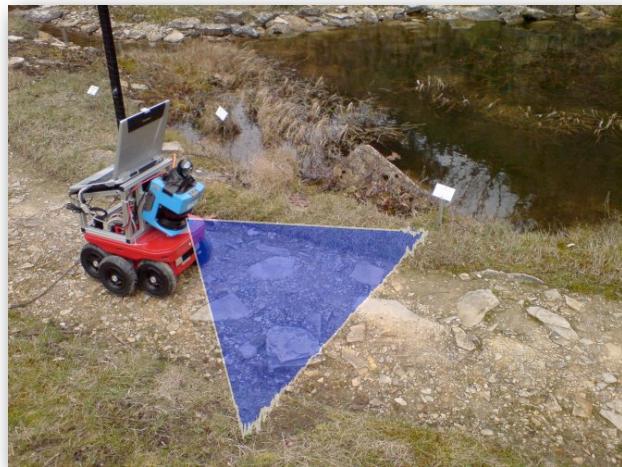
- Steuerung von mobilen Robotern („Embedded Systems“)
- 3D Umgebungserfassung und Modellierung
- Semantische Karten basierend auf Umgebungsmodellen



www.inf.uos.de/kbs

Praktika AG KBS

- ▶ Praktikum WS 2011 / 2012
- ▶ Kleinere Projekte an Kurt-Robotern
- ▶ Bei Interesse: SICK Robot Day 2012
- ▶ KI, Robotik, WBS (Prof. Hertzberg)



C++?

Days 1 - 10

Teach yourself variables, constants, arrays, strings, expressions, statements, functions,....



Days 11 - 21

Teach yourself program flow, pointers, references, classes, objects, inheritance, polymorphism,



Days 22 - 697

Do a lot of recreational programming. Have fun hacking but remember to learn from your mistakes.



Days 698 - 3648

Interact with other programmers. Work on programming projects together. Learn from them.



Days 3649 - 7781

Teach yourself advanced theoretical physics and formulate a consistent theory of quantum gravity.



Days 7782 - 14611

Teach yourself biochemistry, molecular biology, genetics,....



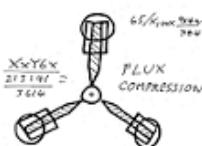
Day 14611

Use knowledge of biology to make an age-reversing potion.



Day 14611

Use knowledge of physics to build flux capacitor and go back in time to day 21.



Day 21

Replace younger self.



As far as I know, this
is the easiest way to

"Teach Yourself C++ in 21 Days".

Ziele der Veranstaltung

- ▶ Wichtig:
 - Systematisches Erlernen von C++
 - Als Informatiker sollte man mehr als nur eine Programmiersprache können!
- ▶ Wichtiger:
 - C / C++ versteckt weniger Systemarchitektur als Java usw.
 - Man muss sich direkter mit dem System auseinander setzen
- ▶ Am Wichtigsten:
 - Praktische Erfahrung in Problemorientierter Programmierung
 - Praktische Erfahrung Software-Design
 - Anwendung dessen was man in Info X gelernt hat
- ▶ Aber:
 - Zeitaufwand kann hoch werden
 - Man muss bereit sein, sich mit den Problemen auseinder zu setzen

Organisatorisches (1) - Vorlesungsteil

- ▶ 15 Vorlesungen á 60 min
- ▶ Inhalt
 - Einführung in die Programmierung mit C (erster Teil)
 - Einführung in die Programmierung mit C++ (zweiter Teil)
 - C++ für Fortgeschrittene (dritter Teil)
- ▶ Voraussetzungen
 - Informatik A, Informatik B
 - UNIX / Linux / Cygwin (kein offizieller Support für Windows!)
 - Interesse am Programmieren
- ▶ Ziele
 - Sicherer Umgang mit C++ lernen
 - Software Engineering Praxis
 - Wie schreibt man Code, der einfach zu verstehen ist?
 - Wie findet man Bugs
 - Wie wird Code wartbar und erweiterbar?

Organisatorisches (2) - Vorlesungsteil

- ▶ Materialien zur Vorlesung werden in StudIP bereit gestellt:
 - Vorlesungsfolien (kein ausformuliertes Skript)
 - Beispielprogramme aus der Vorlesung
 - Übungszettel plus Unterlagen
- ▶ Übungen
 - Jede Woche gibt es einen Zettel, der bearbeitet werden muss
 - Zettel Freitags via StudIP / Fragerunde Montags nach der Vorlesung
 - Tutoren kontrollieren die Leistungen (**Raum 145**)
 - Tutoren machen für jeden Zettel einen Bewertungsvorschlag
 - Die endgültigen Noten werden allerdings von **mir** vergeben!!!
 - Fragen / Kommentare zu den Übungen im StudIP-Forum!
 - Durchschnittsnote der Übungen ergibt die Endnote (**ein F frei**)
 - ➡ **keine Klausur am Ende**
 - ➡ **Anmeldung in OPIUM bis spätestens 31.10!!! (ausnahmslos)**

Organisatorisches (3) - Vorlesungsteil

► Ablauf der Übungen

- Abgabe der Übungen an mich via E-Mail
 - Adresse: cpp@informatik.uni-osnabrueck.de
 - bis spätestens **Montag 0:00 Uhr**
 - d.h. in der Nacht von Sonntag auf Montag
 - Betreff: „**Gruppe A|B|C vollständiger Name**“
 - Anhänge nicht packen!
- Abgeben muss jeder **einzel**n
- Testate erfolgen aber in Zweiergruppen
- Testzuteilung: Listen hängen in 5. Stock AVZ

Nachfragen wegen
Matrikelnummern

► Fragen, Anregungen, Kritik:

- Jederzeit! (in der Woche immer, keine Garantie am Wochenende!!!)
- E-Mail (twiemann@uos.de oder **cpp**-Adresse)
- Vorbeikommen: 509a
- Telefon: 2438, ICQ: 284-384-188

Organisatorisches (4) - Praktikumsteil

- ▶ Die Veranstaltung gibt insgesamt 6 Credits
- ▶ Jeweils 3 für Vorlesung und Praktikum
- ▶ Beides kann nur zusammen als Modul belegt werden (kein Splitting in 3 + 3 Credits möglich!)
- ▶ Endnote Durchschnitt aus Praktikums und Vorlesungsnote!
- ▶ Organisation Praktikum:
 - Zeitraum: Direkt in Anschluss an die Vorlesung, Dauer x Wochen
 - Im Praktikum wird ein mittelkleines Entwicklungsprojekt in Gruppen bearbeitet (Gruppengröße wird noch festgelegt)
 - Jede Gruppe erarbeitet eine Lösung und Dokumentiert Ihr Projekt
 - Jede Gruppe macht eine kleine Abschlusspräsentation
 - Projektlösung als ganzes wird benotet

**Themenvorstellung, Zeitraum und alles weitere in einer
Vorbesprechung (Terminabsprache via StudIP)**

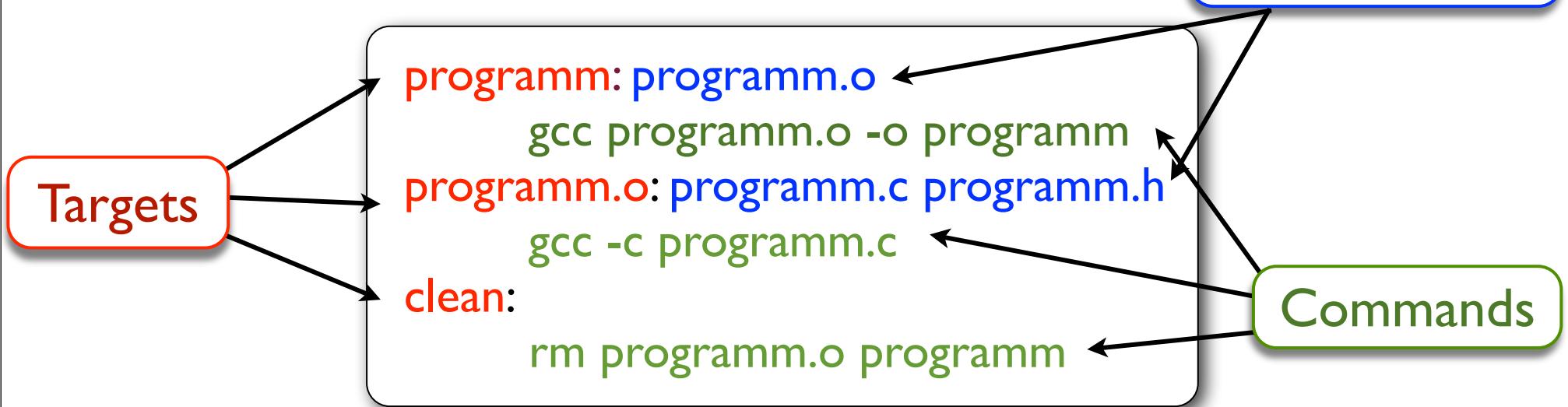
Unterlagen

- ▶ Es gibt kein „offizielles“ Buch zu Vorlesung
 - Der relevante Stoff wird in der Verlesung besprochen
 - Übungen vertiefen und bringen Beispiele über die Vorlesung hinaus
 - ➔ **Selbststudium erforderlich!**
- ▶ „Standardwerk“
 - Bjarne Stroustrup, „*The C++ Programming Language*“
 - Vom Designer der Programmiersprache
 - Gute Referenz, schlechtes Buch zum lernen
- ▶ Gute Bücher zum Lernen:
 - Scott Meyers, „*Effective C++*“, „*More Effective C++*“
 - Herb Sutter, „*Exceptional C++*“, „*More Exceptional C++*“
 - Kerningham / Ritchie „*Programmieren in C*“
 - für den ersten Teil der Vorlesung, nicht die erste Auflage
- ▶ Referenzen
 - Unix man pages für C-Funktionen
 - www.cplusplus.com für C++

Das make-Tool

- ▶ ... ist sehr nützlich wenn man größere Projekte übersetzen will!
- ▶ ... wird später in den Übungen verwendet!

Dependencies



- ▶ Wenn `programm.c` oder `programm.h` sich ändert...
 - ist `programm.o` nicht mehr aktuell
 - `programm.o` wird neu übersetzt
 - nun ist `programm` nicht mehr aktuell
 - `programm` wird übersetzt

Programmdokumentation mit doxygen

- ▶ Mit doxygen lassen sich schnell gute Dokumentationen erzeugen
- ▶ Syntax ähnlich zu javadoc

```
/**  
 * @brief Subscribes to a player server and returns the a client proxy pointer  
 *  
 * If the given player server is running but no fitting PlayerServer instance  
 * is found in @ref m_playerServers, a new one will created.  
 *  
 * @param ip      IP of a running player server  
 * @param port    Connection port on the player server  
 * @param interface Interface ID  
 * @param index    Interface index  
 *  
 * @return Pointer to the corresponding ClientProxy  
 */  
static ClientProxy* SubscribeProxy(string ip, int port, int interface, int index);
```

Beispiel

Gliederung

1. Einführung in C

1.1 Historisches

1.2 Struktur eines C-Programms

1.3 Sprachelemente

1.4 Zeiger

1.5 Benutzerdefinierte Datentypen

1.6 Weitere Sprachelemente

2. Einführung in C++

3. C++ für Fortgeschrittene

4. Weitere Themen rund um C++

Einführung in C - Historisches

- ▶ Geschichte / Ursprung
 - Martin Richards entwickelt 1966 die Sprache BCPL
 - BCPL beeinflusst 1970 die von Ken Thompson entwickelte Sprache B. BCPL und B sind typenlose Sprachen und haben große Nähe zu Assembler
 - Dennis Ritchie von den Bell Labs arbeitete an UNIX
 - 1972 Programmiersprache C
 - UNIX wurde zu 95% in C geschrieben
- ▶ Pro
 - Low-level Programmierung
 - Geschwindigkeits- und Speichereffizient
 - Portabel
- ▶ Contra
 - Unsicher
 - Niedriger Abstraktionslevel

Variablen in BCPL sind Verweise auf eine Speicherzelle, die je nach verwendeten Operanden interpretiert werden (Wertzuweisung, Sprung in Funktion etc)

Gliederung

1. Einführung in C

1.1 Historisches

1.2 Struktur eines C-Programms

1.2.1 Hello World

1.2.2 Quellen / Objektcode / Linken

1.2.3 Der C-Präprozessor

1.3 Sprachelemente

1.4 Zeiger

1.5 Benutzerdefinierte Datentypen

1.6 Weitere Sprachelemente

C - Hello World

- ▶ Erzeuge eine Datei `hello.c`
z.B. mit `emacs`

```
#include <stdio.h>
int main(void)
{
    printf("hello, world!\n");
    return 0;
}
```

- ▶ Übersetze sie mit einem C-Compiler (z.B. `gcc`)
- ▶ Juchu!!
- ▶ Der Compiler übersetzt das Programm in ein Object-File
- ▶ Der Linker führt Object-Files und benötigte Bibliotheken zu einem ausführbaren Programm zusammen
- ▶ Hier macht `gcc` beides in einem Schritt

```
% gcc hello.c -o hello%
hello
hello, world!
%
```