Rechtlicher Hinweis

Legal notice

Informatik I (B.Sc. Physik)

Grundlagen: Erstellen eines C++-Programms

Dr. Paul Bodesheim
(Paul.Bodesheim@uni-jena.de)

FRIEDRICH-SCHILLER-
UNIVERSITÄT
JENA

Fakultät für Mathematik und Informatik
Lehrstuhl für Digitale Bildverarbeitung

SoSe 2020

Inhalt

Inhalt

Erlernen einer neuen Sprache

Der Rechner muss verstehen, was Sie programmieren.

Syntax 句法

- aus dem Griech.: *syn* - zusammen, *taxis* - Ordnung, Reihenfolge
- Regeln für die Kombination aus elementaren Zeichen zu Wörtern 基本字符与单词结合的规则
- Zusammenführen von Wörtern und Wortgruppen zu Sätzen 将单词和单词组汇编成句子
- Satzlehre als Teilgebiet der Grammatik (regelmäßige Muster, Satzstruktur)
- Programmiersprache: Regeln für gültige Anweisungen / Befehle (Struktureigenschaften)
  编程语言：有效权限/命令的规则

Semantik 语义学

- Bedeutungslehre, Bedeutung der Zeichen
- Natürliche Sprache: Bedeutung eines Satzes / einer Aussage
- Programmiersprache: Bedeutung einer Anweisung / eines Befehls

意义理论，标志意义
自然语言：句子/陈述的含义
编程语言：指令的含义

Beispiel: “x=5” bzw. “x=5;” und nicht “x5=” oder “=x5”
(Bedeutung: x nimmt den Wert 5 an)

Die notwendigen Schritte beim Programmieren in C++

编写程序代码（指令，命令）
没有语法错误（语法正确）
）编译器检查（稍后对此进行详细介绍）

1. Programmcode schreiben (Anweisungen, Befehle)
   - ohne Syntaxfehler (syntaktisch fehlerfrei)
     ⇒ überprüft der Compiler (später mehr dazu)
2. Programmcode übersetzen
   - Überführung in Maschinensprache und Erstellen eines ausführbaren Programms
     ⇒ ebenfalls durch Compiler (später mehr dazu)
3. Programm starten und testen

2翻译程序代码
转换为机器语言并创建可执行程序
）也由编译器提供（稍后会对此进行详细介绍）

3启动并测试程序

Inhalt

Einfaches Beispiel: “Hallo Welt!”

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Hallo Welt!" << endl;

    return 0;
}
```

“Hallo Welt!” im Detail

使用#include集成软件库
关键字：包含#
向编译器通知程序中使用的组件
iostream代表C ++中输入和输出的标准形式
（输入输出流，输入输出流）

#include <iostream>

- Mittels **#include** werden Software-Bibliotheken eingebunden
- Schlüsselwort: **include** mit vorangestelltem #
- Compiler über Komponenten informieren, die im Programm verwendet werden
- **iostream** steht für die Standard-Form der Ein- und Ausgabe in C++ (Input-Output-Stream, Eingabe-Ausgabe-Strom)

using namespace std;

- Verwendeten Namensraum angebgen: std für standard

  ⇒ Vereinfachter Zugriff auf Standard-Funktionen
- Schlüsselwörter: **using**, **namespace**

指定使用的名称空间：标准的std
）简化对标准功能的访问
关键字：using，名称空间

int main()

主要功能的开始，程序的开始
关键字：main
int是要返回的数据类型（整数），用作错误消息（更精确地稍后）
没有参数传递给主函数，因此\（）“（不是这样）

- Beginn der Hauptfunktion, Startpunkt des Programms
- Schlüsselwort: **main**
- **int** ist Datentyp für Rückgabe (ganze Zahl), dient zur Fehlermeldung (später genauer)
- Keine Parameterübergabe an Hauptfunktion, daher “()” (muss nicht so sein)

Einfaches Beispiel: “Hallo Welt!”

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Hallo Welt!" << endl;

    return 0;
}
```

“Hallo Welt!” im Detail (2)

{ ... }

- Inhalt der Hauptfunktion in geschweifte Klammern
- Geschweifte Klammern definieren Blöcke von zusammengehörenden Anweisungen, z.B. für eine Funktion 大括号 定义相关指令的块，例如 功能

cout << “Hallo Welt!” << endl;

- cout: Objekt zur Ausgabe auf Konsole 用于输出到控制台的对象
- <<: schreibt auf Konsole, Verkettungen möglich 写入控制台，可以链接
  (wie Pfeile verstehen, die an Konsole senden) （了解箭头如何发送到控制台）
- “Hallo Welt!”: auszugebende Zeichenkette in “...” 字符
- endl: Zeilenvorschub (Zeilenende, neue Zeile)
  end line 换行

return 0;

- Rückgabe des Wertes 0 (ganze Zahl)
  ⇒ kein Fehler aufgetreten!
- Schlüsselwort: **return**
- Beenden des Programms

Einfaches Beispiel: “Hallo Welt!”

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Hallo Welt!" << endl;

    return 0;
}
```

Überblick: Bestandteile eines Programms

- Optional: Präambel zum Einbinden von Funktionalitäten und Festlegen von Namensräumen
- Hauptfunktion: main

  (später: weitere Funktionen, optional)
- Anweisungen, mit Semikolon abschließen
- Blöcke von Anweisungen in geschweiften Klammern

可选：集成功能和定义名称空间的序言
主要功能：主要
（后来：附加功能，可选）
以分号结尾的说明
花括号中的指令块

分号

大括号

Weitere mögliche Bestandteile im Verlauf der Vorlesung!

Grundlagen von C++ aus “Hallo Welt!”

指令和声明以分号（;）结尾
注意关键字的拼写（通常用小写）及预定义的结构，
例如“ cout ”或“ endl ”
空白是语言单词之间的分隔

- Anweisungen und Deklarationen enden mit Semikolon (;)
- Schreibweise der Schlüsselwörter beachten (in der Regel klein)

  ⇒ ebenso vordefinierte Konstrukte wie ”cout” oder ”endl”
- *white space* als Trennung zwischen Wörtern der Sprache

white space

- Leerzeichen, Tabulator oder Zeilenvorschub (neue Zeile)
- Beliebige Anzahl als Trennung zwischen Wörtern

  ⇒ Theoretisch kann das ganze Programm in eine Zeile geschrieben werden

  ⇒ Funktioniert technisch, aber: unschön, unleserlich, unübersichtlich

空格，制表符或换行符（新行）
任意数字作为单词之间的分隔
理论上，整个程序可以写在一行上
）从技术上来说有效，但是：没有吸引力，难以理解，不清楚

Inhalt

Konventionen (Code conventions)

Regeln für gutes Programmieren

良好编程的规则
每行一个声明
缩进的指令块（也用于嵌套），花括号一个在另一个下面
评论是有道理的（不是解释说的是什么，而是为什么在其中），并且对于练习很重要-
将遵循更多规则！

- Eine Anweisung pro Zeile
- Blöcke von Anweisungen einrücken (ebenfalls bei Verschachtelungen), geschweifte Klammern untereinander
- Kommentieren ist sinnvoll (nicht erklären was da steht, sondern warum es so da steht) *und wichtig für die Übungsaufgaben*

Weitere Regeln werden folgen!

Kommentare 注释

被程序忽略，不进行语法检查
用作读者的解释

- Werden vom Programm ignoriert, keine Syntax-Prüfung
- Dienen zur Erklärung für den Leser
- Zwei Formen:
  - //: restlicher Inhalt dieser Zeile ist Kommentar (ggf. gesamte Zeile)
  - /* ... */: begrenzter Kommentar mit Anfangs- und Endmarkierung (ggf. über mehrere Zeilen)

两种形式：
// 此行的其余内容为注释（必要时整行）
/ * ... * / 带有开始和结束标记的有限注释（可能跨越几行）

- **Wichtig für die Übungsserien:**
  - Einleitender, mehrzeiliger Kommentarblock am Anfang der Quelltextdatei zur Erläuterung des Vorgehens zur Lösung der Aufgabe
  - Darin kann auch ein grober Überblick über Ihr Programm und dessen Verwendung gegeben werden (mit welchen Eingaben testen, wie interagiert der Nutzer mit dem Programm etc.)
  - Kurze, präzise, möglichst einzeilige Kommentare zur Beschreibung von Anweisungen im Programm

源文本文件开头的介绍性多行注释块，用于解释解决任务的过程

这也可以粗略地概述您的程序及其使用（使用什么输入测试，用户如何与程序交互等）。

简短，准确，可能单行的注释，用于描述程序中的指令

Kommentare in “Hallo Welt!”

```
/*
Ueberblick ueber das Programm ''Hallo Welt!''

Dies ist ein Beispielprogramm, welches einfach
nur eine Ausgabe auf der Konsole erzeugen soll.
*/

#include <iostream> // Ausgaben ermoeglichen

using namespace std; // Namensraum festlegen
/* vereinfachte Verwendung von Standard-Funktionalitaeten:
   cout anstelle von std::cout
   endl anstelle von std::endl
*/

// Start des Programms
int main()
{
    cout << "Hallo Welt!" << endl; // Unsere erste Ausgabe!

    return 0; // kein Fehler
}
```

Funktioniert zwar auch, aber bitte nicht so!

```
#include <iostream>

using namespace std;

int main()
{
    cout << /* Bitte ausgeben: */ "Hallo Welt!" << endl;

    return /* kein Fehler! */ 0;
}
```

Inhalt

Beispiel: “Hallo Welt!”

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Hallo Welt!" << endl;

    return 0;
}
```

Wohin mit dem Quellcode?

- Code = Programmcode = Quellcode = Quelltext
- Speichern in einer einfachen Textdatei mit der Endung ".cpp", z.B. in *hello.cpp*
- Wichtig: Editor verwenden, kein Textverarbeitungsprogramm (MS Word, ...)
- Was passiert als nächstes?

保存为带有扩展名\ .cpp“ 的简单文本文件，例如，在hello.cpp中
重要提示：使用编辑器，不使用文字处理程序 (MS Word，...)

Ein ausführbares Programm erzeugen (Programm bauen)

- Quelltext übersetzen (compilieren) mit C++-Compiler
- Überführen des vom Menschen lesbaren Quellcodes in Maschinensprache (Maschinenprogramm)
- Überprüfung von Syntaxfehlern

- In der Konsole: **c++ hello.cpp -o hello**
- Erzeugt bei syntaktisch fehlerfreiem Code die ausführbare Datei **hello** (Fehlermeldung sonst)
- Ausführen des Programms in der Konsole: **./hello**

使用C ++编译器翻译（编译）源代码
将人类可读的源代码转换为机器语言（机器程序）
检查语法错误
在控制台中：c ++ hello.cpp -o hello
如果代码在语法上正确，则它将生成可执行的hello文件（否则为错误消息）
在控制台中执行程序：./hello

Verwenden des Compilers

- **c++ hello.cpp -o hello**
  1. **c++**: Aufruf des C++-Compilers
  2. Name der Quelltextdatei
  3. Setzen des Schalters (flags) -o, steht für output

     $\Rightarrow$ Erlaubt Namensgebung für das Ergebnis des Compiler-Aufrufs
  4. Name des auführbaren Programms (beliebig)

Weitere Schalter / Optionen / Flags für den Compiler

- **-std=c++11**: Verwenden der Spracherweiterung durch den Standard C++11
- **-Wall**: Aktivierung aller Warnungen (Hinweise des Compilers), Warnungen ≠ syntaktische Fehler
- **-O2**: Code optimieren (gleiches Verhalten wie ohne Optimierung, ggf. schneller oder speichereffizienter)
- Beispiel: c++ -std=c++11 -Wall -O2 hello.cpp -o hello
- Üblich, aber nicht zwingend: Schalter nach Name der Quelltextdatei, andere Schalter davor

-std = c ++ 11：标准C ++ 11使用语言扩展
-Wall：激活所有警告（来自编译器的信息），
警告≠语法错误
-O2：优化代码（必要时与不进行优化的行为相同）
更快或更高效的内存）
示例：c ++ -std = c ++ 11 -Wall -O2 hello.cpp -o hello
常见但不是强制性的：在名称之后切换-o
源代码文件，它前面的其他开关

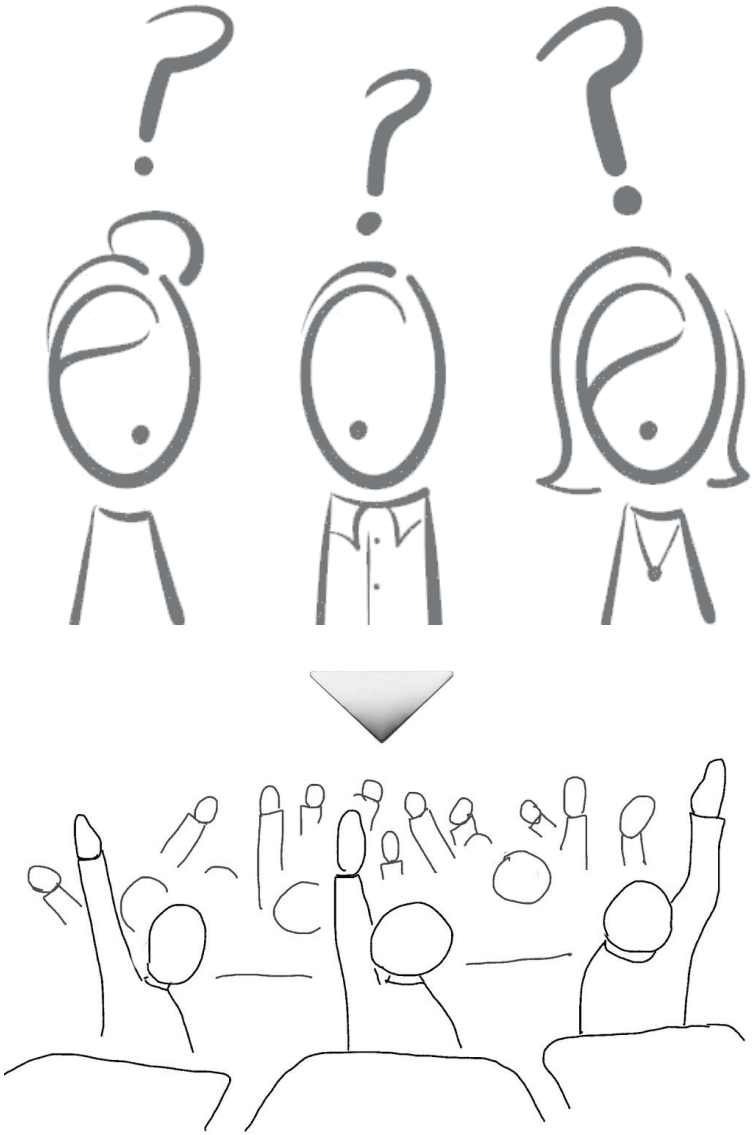
Was ist noch möglich?

- Siehe Manual in Konsole: **man c++**
- GNU compiler collection: gcc bzw. g++

- Automatisierung des Übersetzens mit Makefile
  - Kommando: **make**
  - Beschreibung in einem **Makefile** (eine Datei mit genau diesem Namen)
  - Verwendung in der Konsole: **make hello**
  - Sinnvoll bei umfangreicheren Programmen und Software-Bibliotheken

请参见控制台中的手册：man c ++
GNU编译器集合：gcc或g ++
使用Makefile自动化翻译
命令：make
生成文件中的描述（具有确切名称的文件）
在控制台中使用：打个招呼
对较大的程序和软件库有用

Gibt es Fragen? (Es gibt keine dummen Fragen!)



"Excuse me, is this the Society for Asking Stupid Questions?"