Programmieren in C

SS 2021

Vorlesung 1, Dienstag 20. April 2019

(kickoff, admin, ein erstes Programm)

Prof. Dr. Peter Thiemann
Programmiersprachen
Institut für Informatik
Universität Freiburg
(nach Folienvorlagen von Prof. Dr. Hannah Bast)

Die heutige Vorlesung

UNI FREIBURG

- Organisatorisches
  - Ablauf — Vorlesungen, Übungen, Projekt
  - Prüfungstechnisches — Punkte, Note, ECTS & Aufwand
  - Art der Vorlesung — Voraussetzungen, Lernziel, Stil
- Inhalt
  - Erstes Programm — Approximation von Pi
  - mit allem Drumherum — Kompilieren, Unit Test, clang-format Makefile, Git, CI

Übungsblatt 1: Arbeiten mit der Infrastruktur

Ablauf 1/4

UNI FREIBURG

- Vorlesungen
  - Jeden Dienstag von 10-12 Uhr online
    - Nicht am 25.5. (Pfingstpause)
    - Insgesamt 13 Termine
    - Livestream und Aufzeichnung
  - Assistent der Vorlesung: Hannes Saffrich
  - Kursmaterialien auf der Webseite:
    - Aufzeichnungen (Links), Folien, Übungsblätter, Code aus der Vorlesung + evtl. zusätzliche Hinweise, Musterlösungen

Ablauf 2/4

UNI FREIBURG

- Übungsblätter
  - Die Übungen sind der wichtigste Teil der Veranstaltung.
  - Jede Woche ein Übungsblatt, insgesamt 11
  - Abgabe bis zur nächsten Vorlesung
  - **Selber** machen!

    Sie können gerne zusammen über die Übungsblätter nachdenken, diskutieren, etc. … aber die Programme müssen Sie zu **100%** selber schreiben.

    Auch die Übernahme von Teilen gilt als Täuschungsversuch.

    Siehe https://de.wikipedia.org/wiki/Plagiat

Ablauf 3/4

UNI FREIBURG

- Das Projekt
  - Etwas größere Programmieraufgabe zum Abschluss…
  - Umfang ca. 4 Übungsblätter
  - Weniger Vorgaben als bei den Übungsblättern
  - Beginn zwei Wochen vor Vorlesungsende

Ablauf 4/4

UNI FREIBURG

- Kommunikation
  - Wöchentlich Feedback zu Ihren Abgaben

    Von Ihrer*m Tutor*in über gitea, siehe Tutorial
  - Für Fragen aller Art gibt es ein **Forum**

    Siehe Link auf der Webseite
  - Meine Sprechstunde dienstags 12-13 in Zoom
    - https://uni-freiburg.zoom.us/j/82759282295
    - Meeting ID: 827 5928 2295
    - Passcode: 1Question
  - Im Tutorat 1-1 Treffen vereinbaren

Prüfungstechnisches 1/3

UNI FREIBURG

- Punkte
  - Maximal 16 pro Übungsblatt, davon 2 für Erfahrungen.
    Insgesamt maximal 176 Punkte für Ü1 – Ü11
  - Für das Projekt gibt es maximal 80 Punkte
  - Macht insgesamt 256 Punkte

Prüfungstechnisches 2/3

UNI FREIBURG

- Gesamtnote (Studienleistung)
  - Die ergibt sich linear aus der Gesamtpunktzahl am Ende

    ab 128: 4.0; ab 140: 3.7; ab 152: 3.3
    ab 164: 3.0; ab 176: 2.7; ab 184: 2.3
    ab 196: 2.0; ab 208: 1.7; ab 220: 1.3
    ab 232: 1.0

  - **Außerdem:** Zum Bestehen müssen mindestens 88 Punkte in den Übungen (Ü1 – Ü11) und mindestens 40 Punkte im Projekt erreicht werden.
  - **Außerdem²:** Sie müssen sich mindestens einmal mit Ihrem Tutor / Ihrer Tutorin treffen, dazu mehr in einer der späteren Vorlesungen.

Prüfungstechnisches 3/3

UNI FREIBURG

- ECTS Punkte und Aufwand
  - Informatik / ESE / MST / BOK: 6 ECTS Punkte
  - Insgesamt 6 x 30 = 180 Arbeitsstunden
  - Davon 120 Stunden für die ersten 11 Vorlesungen + Übungen
    - also etwa 10 Stunden Arbeit / Woche
    - also etwa 8 Stunden pro Übungsblatt
  - Bleiben 60 Stunden für das Projekt und die dazugehörigen (letzten beiden) Vorlesungen + Übungen

    Wer schon Vorkenntnisse hat, wird weniger Arbeit haben.

Art der Vorlesung 1/6

UNI FREIBURG

- Voraussetzungen
  - Sie wissen, wie Programmieren "im Prinzip" abläuft, z.B.:
    - die Zahlen von 1 bis 10 ausgeben
    - berechnen, ob eine gegebene Zahl prim ist
    - Space shuttle flight system… 
  - Verständnis einiger Grundkonzepte
    - Variablen, Konditionale, Schleifen, Funktionen
- Wenn Ihnen das alles gar nichts sagt, …
  - dann wird es mehr Arbeit als die ECTS Punkte aussagen!

Art der Vorlesung 2/6

UNI FREIBURG

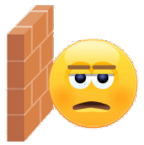
- Lernziel
  - Programmieren in C nach „best practice"
    … im Umfang von 500 – 1000 Zeilen
  - Gute Struktur, gute Namen, gute Dokumentation
  - Verwendung von Standardtools und –techniken:
    - Unit Tests — assert, unity
    - Styler — clang-format, clang-tidy
    - Build System — make und CI
    - Versionskontrolle — Git

Art der Vorlesung 3/6

UNI FREIBURG

- Die Vorlesung ist eher **breit** als **tief**
  - Die Aufgaben sind eher einfach.
  - Die Konstrukte und Konzepte auch.
  - Sie sollen vor allem lernen **guten Code** zu schreiben.
  - Das können viele Leute, die es sich selber beibringen oder beigebracht haben, nicht.
  - Mit der richtigen Herangehensweise macht Programmieren viel mehr Spaß.

  Sonst wird viel Zeit mit Debuggen verschwendet, die besser ins Programmieren investiert wäre.



Art der Vorlesung 4/6

UNI FREIBURG

- Art der Vermittlung

  - Live Programmierung; no magic
    - Dokumentation jeweils letzte Folie jeder Vorlesung
    - Google und stackoverflow sind deine Freunde
  - Ich werde vor allem immer das erklären, was Sie auch gerade brauchen (für das jeweilige Übungsblatt)
- Fragen, Fragen, Fragen
  - Selber ausprobieren, aber nicht zu lange, dann fragen!

    In der Vorlesung, in der Übung oder über das Forum

    Hilfreich: Versuchen Sie jemand anderem das Problem zu erklären

Art der Vorlesung 5/6

UNI FREIBURG

- Ansatz "low-level"
  - Kommandozeile (Unix Shell), Texteditor, Makefile
  - Aufwändige Entwicklungsumgebungen / IDEs (z.B. Eclipse oder NetBeans) sind was für Fortgeschrittene, die den "low level" beherrschen.

  Wer sich auskennt, kann eine IDE benutzen. Abgaben müssen "low-level" ins gitea hochgeladen werden (insbesondere mit Makefile)

  Das ist ggf. mehr Arbeit

Art der Vorlesung 6/6

UNI FREIBURG

- Systemanforderungen
  - C-Compiler: gcc, mindestens Version 9.3.0
  - Buildtool: make
  - Versionskontrolle: git
  - Installation auf vielen Linuxen (ubuntu, debian, …):
    - apt-get install gcc make git
  - Wer unbedingt Windows verwenden will:

    Windows 10: hat eine integrierte Linux-Shell (Ubuntu)

    Windows 8.0: Cygwin (auf eigene Gefahr)

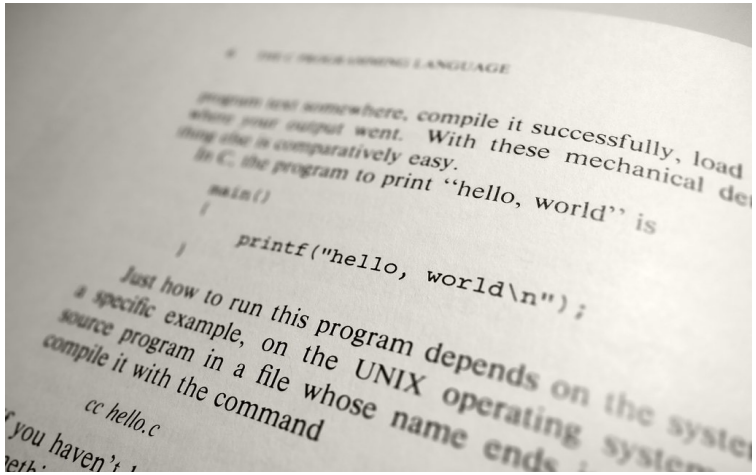
    Windows 3.0: **sie haben da irgendwas verpasst**

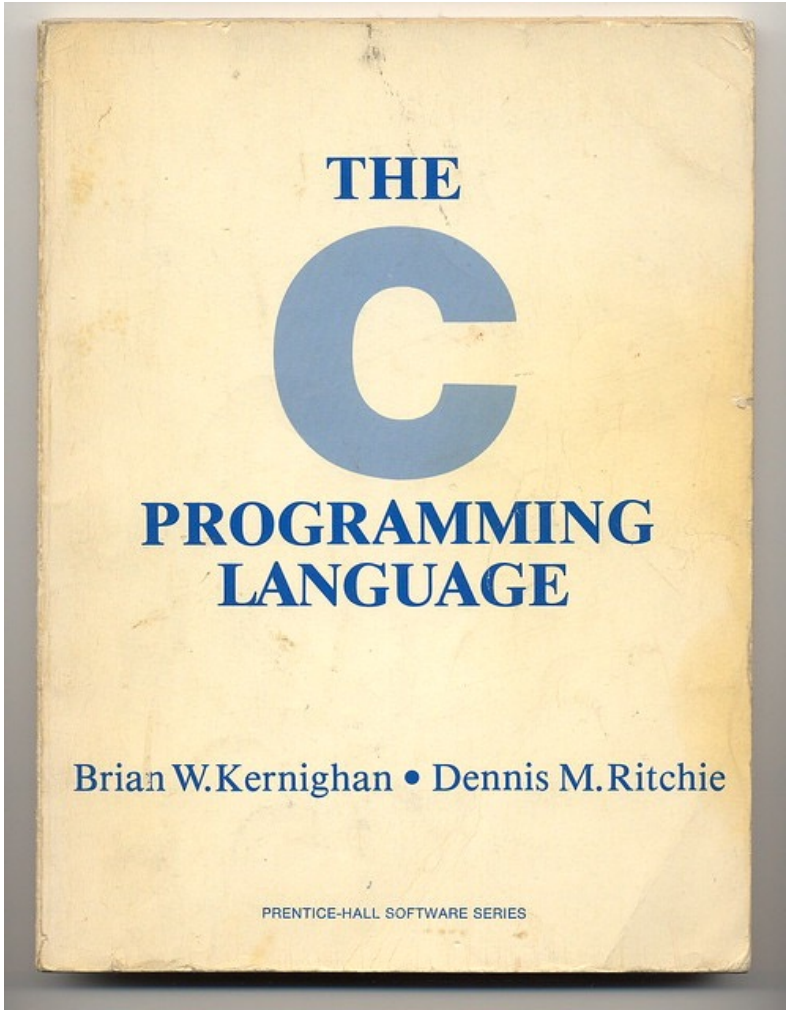
Die Sprache C

UNI FREIBURG

- Nachfolger der Sprachen BCPL [Richards 1967] und B [Johnson 1973], Vorgänger von D
- Kernighan und Ritchie ”in the early 1970s”
- Implementierungssprache von Unix (anstelle von Assembler)
- Unix-Dienstprogramme in C implementiert
- Low-level: wir lernen C, damit wir nicht Assembler programmieren müssen
- Neben Kontrollstrukturen muss in C auch der Speicher großteils selbst verwaltet werden -> Gefahr!
- Geschichtliches The Development of the C Language
- http://www.bell-labs.com/usr/dmr/www/chist.html

Kernighan & Ritchie



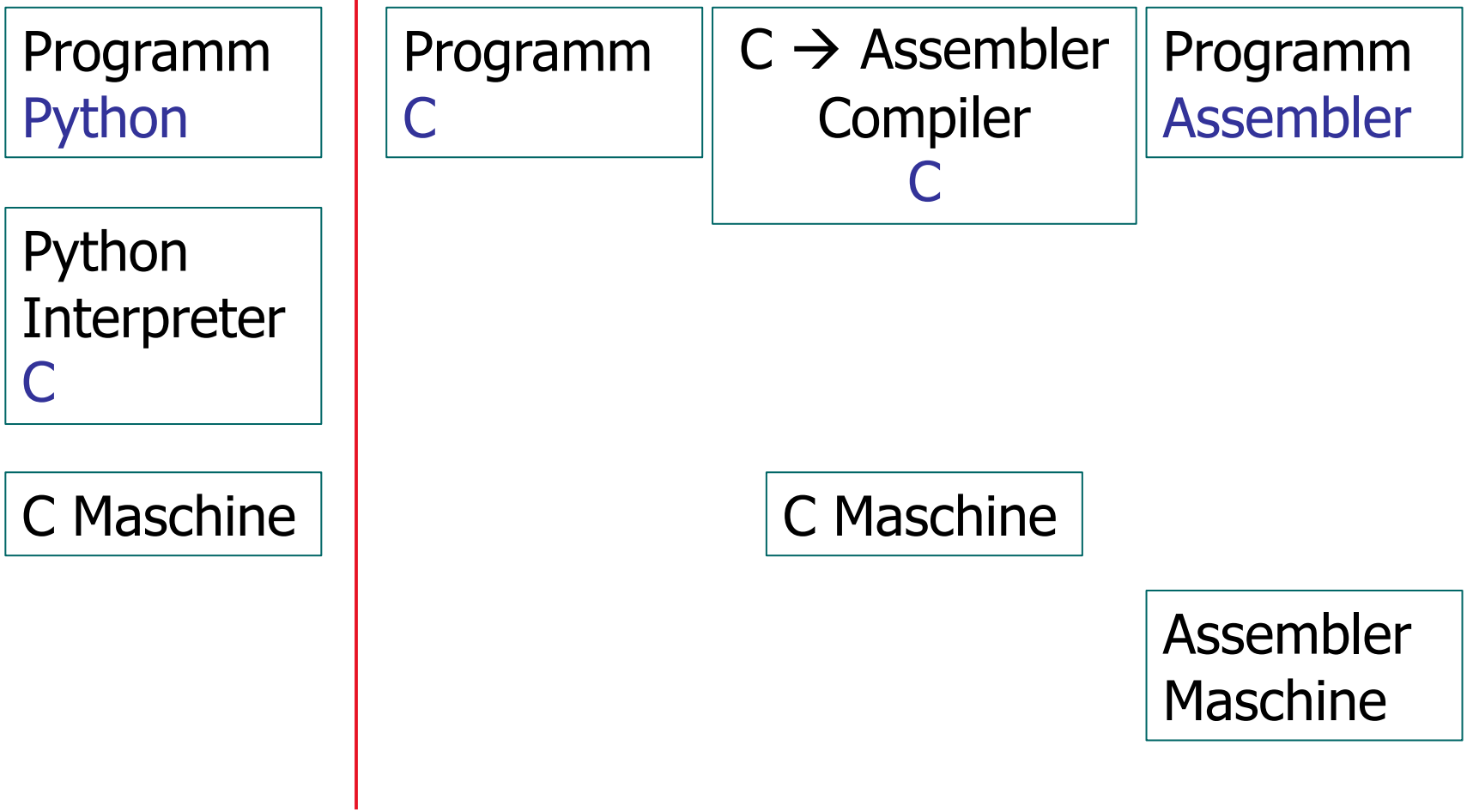


Die Sprache C

UNI FREIBURG

- Compiliert, nicht interpretiert!
- Was bedeutet das?

Interpreter vs Compiler



Erstes Programm 1/12

UNI FREIBURG

- Wir schauen uns ein einfaches Problem an.
- Wir werden es **live** in ein Programm umsetzen.
- Es kommt weniger auf das Problem an, als auf das ganze **Drumherum.**
- Genau wie bei der Bearbeitung des Übungsblatts.

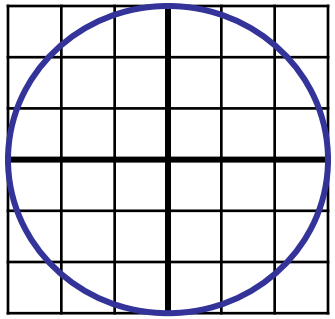
Erstes Programm 2/12

UNI FREIBURG

- Approximieren der Zahl **Pi** über die Kreismethode

  Berechne die Anzahl k aller ganzzahligen Gitterpunkte (x, y) mit $|x|, |y| \leq n$, die innerhalb des Kreises mit Radius n um den Ursprung liegen (also $x^2 + y^2 \leq n^2$).

  Für $n \to \infty$ entspricht der Anteil dieser Gitterpunkte k/n^2 dem Verhältnis der Fläche des Kreises ($\pi \cdot n^2$) zur Fläche des Gitters ($4n^2$) … also $\pi / 4$.



Erstes Programm 3/12

UNI FREIBURG

- Version 1
  - Eine Textdatei für den gesamten Programmcode

    ApproximationOfPi.c
  - Sie können dafür einen beliebigen Texteditor verwenden

Erstes Programm 4/12

UNI FREIBURG

- Kompilieren
  - Wir benutzen den Gnu C Compiler, kurz **gcc**

    gcc –o ApproximationOfPi ApproximationOfPi.c

    Ich benutze in der Vorlesung die gcc **Version 9.3.0** …

    Ältere oder experimentelle Versionen auf eigene Gefahr!
  - Der Befehl oben erzeugt Maschinencode, der wie folgt ausgeführt werden kann

    ./ApproximationOfPi

    Ohne die –o Option würde das ausführbare Programm einfach "a.out" heißen, …

Erstes Programm 5/12

UNI FREIBURG

- Version 2
  - Wir schreiben jetzt zwei Dateien (warum sehen wir gleich)

    ApproximationOfPi.c
    ApproximationOfPiMain.c
  - Die erste Datei enthält nur die Funktion.
  - Die zweite Datei enthält das restliche Programm und liest zu Beginn die erste Datei ein

    ```
    #include "./ApproximationOfPi.c"
    ```

    In Vorlesung 2 werden wir sehen, dass das nicht optimal ist, aber für heute (und Ü1) ist das völlig in Ordnung.

Erstes Programm 6/12

UNI FREIBURG

- Unit Tests
  - Wir schreiben jetzt noch eine dritte Datei

    ApproximationOfPiTest.c
  - Diese enthält Testfunktionen, sowie eine main Funktion, die die gewünschten Tests aufruft.

```
#include “unity/unity.h“
#include "./ApproximationOfPi.c“

/* Testfunktionen (weggelassen) */

int main(void) {
  UNITY_BEGIN();

  RUN_TEST(…);
  return UNITY_END();
}
```

Erstes Programm 7/12

UNI FREIBURG

- Überprüfung des Stils
  - Bisher haben wir "nur" versucht, unser Programm zum Laufen zu bringen
  - Selbstverständlich sollte unser Code auch schön und für anderen Menschen gut lesbar sein
  - Es gibt extra Programme dafür, wir benutzen **clang-format** und **clang-tidy**
  - Um damit den Stil aller .cpp Dateien zu überprüfen, können wir einfach schreiben

    clang-tidy *.c
  - Liefert sehr detaillierte und in aller Regel selbsterklärende Fehlermeldungen zum Stil des Codes

Erstes Programm 8/12

UNI FREIBURG

- Makefile und make
  - Woher wissen andere, wie sie unseren Code kompilieren, testen, oder seinen Stil überprüfen können?
  - Wir schreiben eine Datei **Makefile** mit folgender Syntax

```
<target>:
        <Befehl 1>
        <Befehl 2>
        ...
```

Achtung: jede der Befehlszeilen muss mit einem TAB anfangen!

  - **make <target>** in dem Verzeichnis, in dem diese Datei steht, bewirkt die Ausführung der entsprechenden Befehle

make kann noch viel, viel mehr… mehr dazu in den nächsten Vorlesungen

Erstes Programm 9/12

UNI FREIBURG

- Gitea
  - Git repository für Ihre Abgaben und Korrekturen
  - **Einloggen dort mit dem Uni-Account und Passwort**
  - Sie kriegen dann automatisch Zugang zu

    Git, CI, Forum … siehe nächste drei Folien

Erstes Programm 10/12

UNI FREIBURG

- Git
  - Jeder hat dort ein Verzeichnis mit vordefiniertem Inhalt (u.a. Unterverzeichnisse für die einzelnen Übungsblätter)
  - Sie bekommen Ihre Arbeitskopie dieses Verzeichnisses mit
    - git clone <URL>
  - In Ihrer Arbeitskopie können Sie Unterordner und Dateien hinzufügen und editieren, aber bitte nichts an der vorgegebenen Struktur ändern!
    - git add <file name> fügt eine Datei erstmals hinzu
    - git commit <file name> fixiert die Änderungen lokal
    - git push lädt die Änderungen zu uns hoch
    - git pull lädt die Kommentare, Punkte, etc herunter

Erstes Programm 11/12

UNI FREIBURG

- CI (Continuous Integration)
  - Mit CI können Sie (und wir) überprüfen, ob der hochgeladene Code wie gewünscht funktioniert.
  - Nach jedem Push führt das CI-Werkzeug folgenden Befehle aus

make clean	Löscht Nebenprodukte
make compile	Kompiliert ihren Code
make test	Führt die Unit Tests aus
make checkstyle	Prüft den Stil Ihres Codes

  - Sie bekommen dann angezeigt, ob es funktioniert hat, und auf Wunsch auch die komplette Ausgabe
  - Voraussetzung: Makefile muss vorhanden sein!

Erstes Programm 12/12

UNI FREIBURG

- Forum/Chat
  - Machen Sie bitte regen Gebrauch davon und haben Sie **keine Hemmungen**, Fragen zu stellen

    Insbesondere wenn Sie bei einem Fehler mit eigenem Nachdenken, Google, stackoverflow etc nicht weiterkommen

    In C können Kleinigkeiten zu stundenlangem Suchen führen, was sehr frustrierend ist
  - Geben Sie sich aber gleichzeitig Mühe, Ihre Fragen möglichst konkret und genau zu stellen.

Hinweise für das Ü1 1/1

UNI FREIBURG

- Arbeitsaufwand
  - Sie haben **eine Woche** Zeit.
  - Die meiste Arbeit wird das Drumherum sein.
  - Fangen Sie **rechtzeitig** an und fragen Sie auf dem Forum, wenn Sie nicht weiterkommen.
  - Fragestunden im Rahmen der Tutorate

Literatur / Links

- Git
  - https://git-scm.com/
  - Kurze Einführung dazu siehe Webseite/Tutorat
- C
  - https://en.cppreference.com/w/c
  - Am Anfang nur elementare Sachen