Programmierung in C

Ablauf

Grundsätzliches

- ihr programmiert auf euren eigenen Rechnern mit UNIX
- das heißt: Linux, macOS, oder Windows mit WSL
- wir helfen euch heute beim einrichten, falls es noch nicht läuft
- ihr könnt falls nötig auch die Poolrechner nutzen

Ablauf

Grundsätzliches

- ihr programmiert auf euren eigenen Rechnern mit UNIX
- das heißt: Linux, macOS, oder Windows mit WSL
- wir helfen euch heute beim einrichten, falls es noch nicht läuft
- ihr könnt falls nötig auch die Poolrechner nutzen
- zunächst: jeden Tag ein Übungszettel
- aus jedem Zettel müssen drei Aufgaben bearbeitet werden

Ablauf

Grundsätzliches

- ihr programmiert auf euren eigenen Rechnern mit UNIX
- das heißt: Linux, macOS, oder Windows mit WSL
- wir helfen euch heute beim einrichten, falls es noch nicht läuft
- ihr könnt falls nötig auch die Poolrechner nutzen
- zunächst: jeden Tag ein Übungszettel
- aus jedem Zettel müssen drei Aufgaben bearbeitet werden
- ab nächster Woche: freies Programmieren in Kleingruppen
- Jede Gruppe entwickelt ein selbstgewähltes Programm
- und stellt es am Ende vor (Präsentationsprüfung)

Ablauf

Wichtige Eckdaten

jeden Tag	10:00 – 12:00 c. t. Vorlesung (hier im SR 055)
jeden Tag	13:00 – 18:00 c. t. Programmierpraktikum (Poolräume)
Fr 21.03.	10:00 – 15:00 Präsentationsprüfung (je 10 min pro Person)
Mo 24.03.	10:00 – 18:00 Präsentationsprüfung (falls nötig)

Ablauf – Gliederung der Vorlesung

1. Woche Grundlagen der Sprache
2. Woche Werkzeuge und Techniken
   - Versionsverwaltung (git)
   - wichtige Werkzeuge (make, gdb, clang-sanitizer)
   - einfache Datenstrukturen
   - nützliche Bibliothek (SDL)
3. Woche Weiterführende Themen
   - Objektorientierte Programmierung
   - Arbeiten mit Bibliotheken
   - Techniken zur strukturierten Programmierung

Ablauf – Was braucht ihr zum Bestehen?

- Prüfungszulassung: 3 Aufgaben auf jedem Zettel bearbeitet
- in Kleingruppe (~3 Leute) ein Projekt entwickelt
- Abgabe des Projektes als git-Repository am *Do 20.03. 18:00 Uhr*
- Projekt in der Prüfung gemeinsam vorstellen
- Präsentationsprüfung am Fr 21.03. oder Mo 24.03.

  2 Personen 15 min Präsentation, 5 min Fragen
  3 Personen 20 min Präsentation, 10 min Fragen
  4 Personen 25 min Präsentation, 15 min Fragen

Ablauf – Akademische Integrität

- der Code in eurem Projekt muss von euch selbst geschrieben sein
- finde ich geklauten Code in eurem Projekt, fallt ihr durch

Ablauf – Akademische Integrität

- der Code in eurem Projekt muss von euch selbst geschrieben sein
- finde ich geklauten Code in eurem Projekt, fallt ihr durch
- ChatGPT darf gefragt werden, aber nicht für euch programmieren
- Dokumentation oder Stack-Overflow-Antworten lesen liefert aber meist bessere Ergebnisse
- könnt ihr ihr euren Code nicht erklären ist das sehr schlecht

Ablauf – Akademische Integrität

- der Code in eurem Projekt muss von euch selbst geschrieben sein
- finde ich geklauten Code in eurem Projekt, fallt ihr durch
- ChatGPT darf gefragt werden, aber nicht für euch programmieren
- Dokumentation oder Stack-Overflow-Antworten lesen liefert aber meist bessere Ergebnisse
- könnt ihr ihr euren Code nicht erklären ist das sehr schlecht
- die Anwesenheit wird nicht kontrolliert,
- aber wer nicht kommt, wird es schwer haben zu bestehen

Die Sprache C

The C Programming Language:

- 1972 erschienen
- Prozedural-strukturierte Programmiersprache (imperativ!)
- Für Betriebssystementwicklung entworfen
- Nahezu auf jedem Gerät verfügbar

Die Sprache C

Wo ist C überall drin?

- viele modernen Betriebssysteme (Linux, macOS, Windows, FreeBSD, Solaris, ...)
- eingebettete Programme (Autocomputer, Haushaltsgeräte, Entertainment, ...)
- viele wichtige Softwarebibliotheken (FFmpeg, SQLite, cURL, zlib, glibc, ...)
- graphische Frameworks (Gnome, XFCE, SDL, ...)
- viele klassische Videospiele (Doom, Super Mario 64, Pokémon (Gen 3/4), ...)
- Serversoftware (Apache, nginx, PostgreSQL, dovecot, mongoDB, ...)
- Interpreter (Python, node.js, Ruby, lua, sh, awk, ...)
- ...eigentlich überall

Programmausführung

- In für den Computer verständliche Sprache übersetzen (kompilieren) mit

```
$ cc -o programmname programmname.c
```

- Ausführen auf der Konsole: `$ ./programmname`

```c
#include <stdio.h>

int main(void)
{
  printf("Hello World\n");
  return (0);
}
```

Was haben wir da geschrieben?

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5   printf("Hello World\n");
6   return (0);
7 }
```

- Zeile 1: Möchten Funktionalität für Ein-/Ausgabe benutzen
- Zeile 3–7: Definition `main()`
  - Zeile 3: Signatur: Rückgabewert und Parameter
  - Zeile 4, 7: Beginn und Ende des Funktionsblockes
  - Zeile 5: Funktionsaufruf, gibt Argument auf der Konsole aus
  - Zeile 6: Beenden Funktion, geben 0 zurück

Warum genau so?

Wir schreiben in C, in einer „Hosted“ Umgebung

- D. h. wir haben die C-Standardbibliothek zur Verfügung
- Diese stellt uns `stdio.h` mit `printf()` zur Verfügung
- Außerdem schreibt dies uns vor, dass der *Programmeintrittspunkt* `main()` heißt und auch seine Signatur
- Um zu verstehen, warum das vorgeschrieben sein muss, müssen wir uns überlegen, wie unser Programm überhaupt gestartet wird

Was ist ein Programm?

- Ein Betriebssystem, Firmware eines Microkontrollers, ...
- Wir schreiben hier (im weiteren Sinne) *Anwendungsprogramme*
- D. h. es gibt einen Nutzer, der den Rechner bedient und euer Programm startet
- Dazu ist ein Betriebssystem notwendig
- Unsere Programme haben (vorerst) keine graphische Oberfläche, sondern werden im Terminal ausgeführt

Programmumgebung: Allgemein

- Ein Programm besteht grob aus einer Menge an Funktionen
- Die Funktion mit der die Programmausführung beginnt, ist der *Programmeintrittspunkt* `main()`
- Nur sie wird vom Betriebssystem aufgerufen, aber sie kann andere Funktionen innerhalb des Programmes ausführen
- ... oder vorgefertigte Funktionen, als Teil von „Bibliotheken“

Eigene Programme schreiben

- Wir schreiben erstmal nur die Funktion `main()` und schreiben keine neuen anderen Funktionen
- Stattdessen beschäftigen wir uns mit einfachen Anweisungen und Variablen
- Um Variablen zu benutzen, muss man sie dem Compiler bekannt machen (*deklarieren*)
- Sowohl Anweisungen als auch Deklarationen werden mit Semikoli beendet.

```
1 float pi, daumen;          // (Fliess-)Kommazahlen
2 daumen = 13.372;
3 pi = 3.141;
4 int antwort = pi*daumen;   // ~42
```

Formatierte Ausgabe

- Um berechnete Ergebnisse auszugeben, müssen sie in eine menschenlesbare Form umgewandelt werden
- `printf()` aus `stdio.h` ist eine mächtige Funktion, die das kann
- Dazu wird als erstes Argument ein sog. Formatstring übergeben, dieser bestimmt *wie* ausgegeben wird
- Bspw.: `printf("i ist: %d\n", i);`
  - Das `"%d"` ist ein Platzhalter, der das nächste Argument als Dezimalzahl interpretiert ausgibt
  - Das `"\n"` ist eine spezielle Escapesequenz welche die Zeile beendet (newline)
  - Der Rest in dem String wird direkt so ausgegeben
  - Für $i = 42$ also: „`i ist: 42`“

Weiterführende Quellen

- Diverse Autoren.
  Wikibooks
  https://en.wikibooks.org/wiki/C_Programming
- Jens Gustedt.
  Modern C
  http://icube-icps.unistra.fr/index.php/File:ModernC.pdf
- C Standards Committee
  C11 Standard Draft
  http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1570.pdf
- OpenGroup
  POSIX 2008 Standard
  http://pubs.opengroup.org/onlinepubs/9699919799/