Informatik 1

Einführung

von
Irene Rothe

Zi. B 241

irene.rothe@h-brs.de

Motivation für die Informatik

- Top 10 der Programmiersprachen (IEEE): https://spectrum.ieee.org/computing/software/the-top-programming-languages-2019
- Linus Torvalds: Ich weiß nicht, wie ich es erklären soll, was mich am Programmieren so fasziniert, aber ich werde es versuchen. Für jemanden, der programmiert, ist es das Interessanteste auf der Welt. Es ist ein Spiel, bei dem du deine eigenen Regeln aufstellen kannst, und bei dem am Ende das herauskommt, was du daraus machst. Der Reiz besteht dann, dass der Computer das tut, was du ihm sagst. Unbeirrbar. Für immer. Ohne ein Wort der Klage. Du kannst den Computer dazu bringen, das er tut, was du willst, aber du musst herausfinden, wie. Programmieren ist eine Übung der Kreativität.
- Alan Turing (Gründer der Informatik): if thoughts (that is, information) can be broken up into simple constructs and algorithmic steps, then machines can add, subtract or rearrange them as our brains do.

Was ist Informatik?

Was ist Informatik?

https://www.youtube.com/watch?v=y80yQEQENZ0

Informatik =

Lösen von Problemen mit dem Rechner

Was braucht man dafür?

→ Algorithmus

Hauptziel

Sie schreiben Code, der

- niemanden umbringt,
- niemanden verletzt und
- niemanden in den Wahnsinn treibt.

Und Sie bestehen die Klausur!

Beispiele?

- Flugzeugabsturz
- Zu viel Röntgenstrahlung
- Server reagiert nicht
- VW ruiniert?

Mein Teaching Style

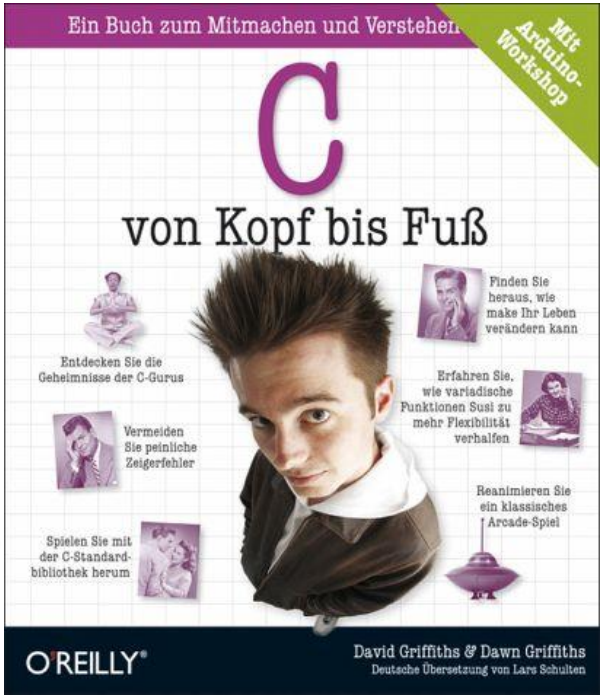
ICH BIN ANSPRECHBAR für so gut wie alles!

Ich wäre gern: Ihr Coach, Ihr Motivator, Ihr Unterstützer

Meine Methode: learning by doing, Lernen aus Fehlern

Literatur

- Der Klassiker:
  Kernighan/Ritchie:
  Programmieren in C
  Hanser
- Griffiths&Griffiths:
  C von Kopf bis Fuß
  O'Reilly



- "Schrödinger programmiert …" von Galileo Computing
- Kurz und knapp, liegt gut vor einem auf dem Tisch:
  "Programmieren in C" RRZN-Handbuch → erhältlich in der Bibliothek für 3,70 €
- "C-Howto" Elias Fischer

Hilfen: Foren

Bekanntestes Forum: **Stackoverflow**

Beim Suchen:

- Etwas hartnäckig sein, bei harten Problemen kann das auch mal Tage dauern
- Immer Freunde, Kommilitonen oder Dozenten wie mich fragen, alles andere wäre Zeitverschwendung

Weitere Hilfen

- Nachdenken über Ordnung auf Rechner oder Stick, z.B. können wenige Unterordner besser sein als viele, Nachdenken über sinnvolle Dateinamen (nicht *Vorlesung1* – diese Info nutzt einem i.R. später wenig, besser *VorlesungMitOrdnungstipps*)
- Backups regelmäßig machen
- *readme.txt* Dateien anlegen mit Infos, die man immer wieder braucht und/oder vergisst
- Emails, bei denen man auf eine Antwort wartet, an sich selbst zur Erinnerung schicken
- Arbeitsplan anlegen
- Dokument in Lea: ProgrammierenAlleinzuHause: Schritt für Schritt Anleitung
- Benennen von Bottlenecks (www.decodingthedisciplines.org), Formulieren von Fragen

Feedback

Bitte melden Sie sich,

- wenn etwas Falsches in meinen Folien steht oder ich etwas Falsches drauf gesprochen habe
- wenn Sie Probleme haben
- wenn Sie eine Idee haben, wie ich etwas besser machen kann

Feedback bzgl. meines Umgangs mit Lea aus den letzten Jahren, woran ich mich halte, so lange nicht neue überzeugendere Idee kommen:

- Keine Texte, lieber Dokumente
- Foren sind gut
- Nicht zu viele Unterordner
- Rund-emails werden gerne gelesen
- Umfragen kommen gut an, wenn sie anonym sind
- Sortierung ist sehr wichtig in Lea
- Beim Hochladen eines Dokuments, dass es schon gab, dazu schreiben, was neu ist, z.B. „Fehler auf Folie…wurde berichtigt

Themengebiete der Informatik

Themengebiete der Informatik

Informatik			
Technische	Praktische	Theoretische	Angewandte
Hardware-Komponenten	Algorithmen Datenstruktren	Automatentheorie	Computergrafik
Schaltnetze, Prozessoren	Programmier-sprachen	Komplexitätstheorie	Datenbanken
Mikroprogrammie-rung	Betriebssysteme	Berechenbarkeit	Künstliche Intelligenz
Rechnerorgani-sation/-architektur	Mensch-Maschine-Kommunikation	Formale Sprachen	Digitale Signalverar-beitung
Rechnernetze	Verteilte Systeme	Formale Semantik	Simulation & Modellierung

Themengebiete der Informatik 1 und 2

Informatik			
Technische	Praktische	Theoretische	Angewandte
Hardware-Komponenten	Algorithmen Datenstruktren	Automatentheorie	Computergrafik
Schaltnetze, Prozessoren	Programmier-sprachen	Komplexitätstheorie	Datenbanken
Mikroprogrammie-rung	Betriebssysteme	Berechenbarkeit	Künstliche Intelligenz
Rechnerorgani-sation/-architektur	Mensch-Maschine-Kommunikation	Formale Sprachen	Digitale Signalverar-beitung
Rechnernetze	Verteilte Systeme	Formale Semantik	Simulation & Modellierung

Bemerkung: zu allen rosa Felder, werden Sie etwas in dieser Vorlesung erfahren

5 Beispiele für Anwendungen

- Skype
- Bildverarbeitung
- Einkaufen im Internet
- Kryptografie
- Linux

Compiler für C89

Freie Compiler:

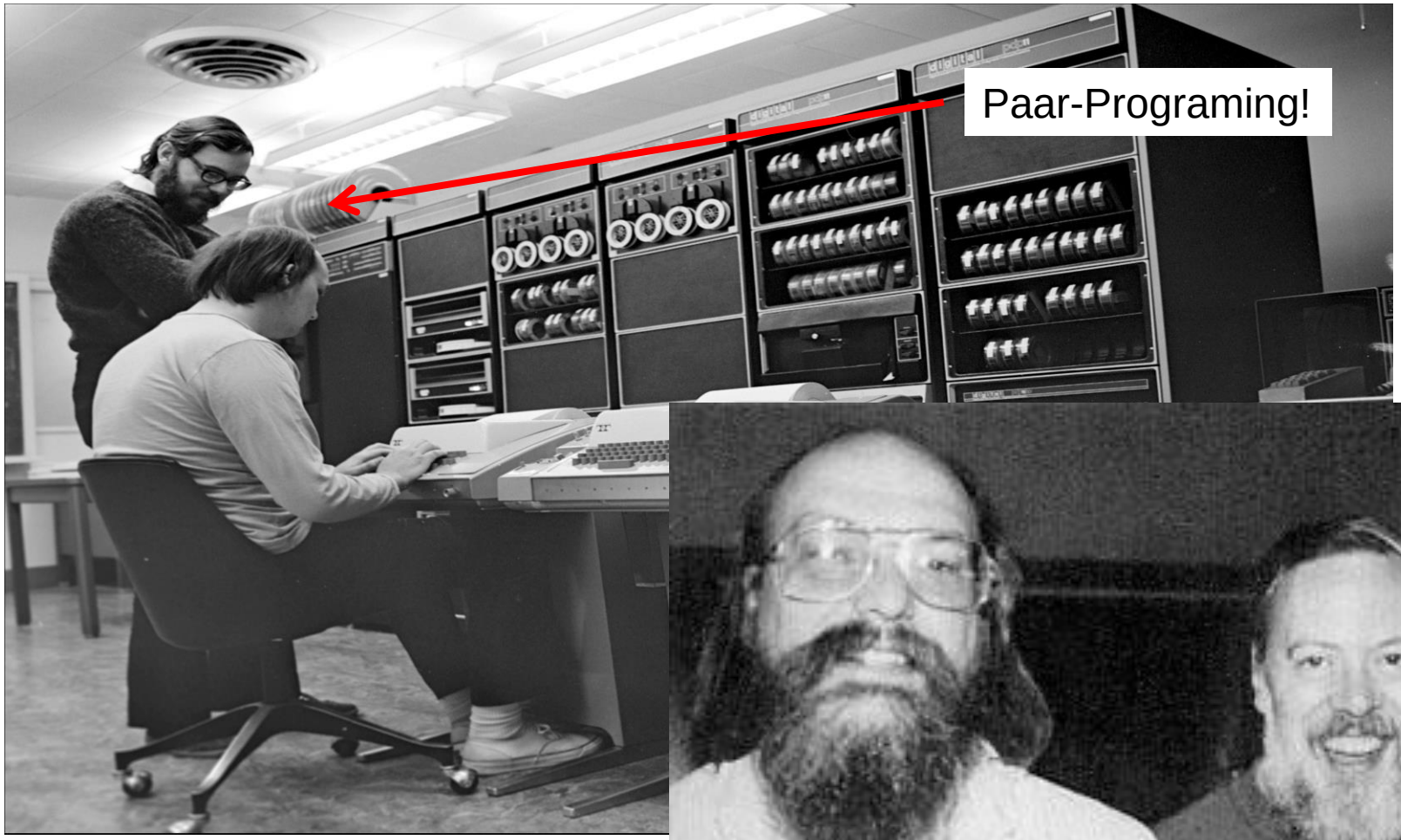
- **Dev-C++** (existiert portable – kann also auf Stick kopiert warden, und man kann überall auch ohne Internet programmieren)
- Code::Blocks, siehe https://www.youtube.com/watch?v=clx8HthehKs

Coco startet mit der C Programmierung Teil 1 - Der Compiler

- gcc + Eclipse
- Online: https://www.onlinegdb.com
- Online: https://www.jdoodle.com/online-java-compiler
- jeder beliebige andere C-Compiler

Achtung: Egal womit Sie arbeiten, legen Sie keine Projekte an, sondern wählen Sie immer: File→Neu→SourceFile

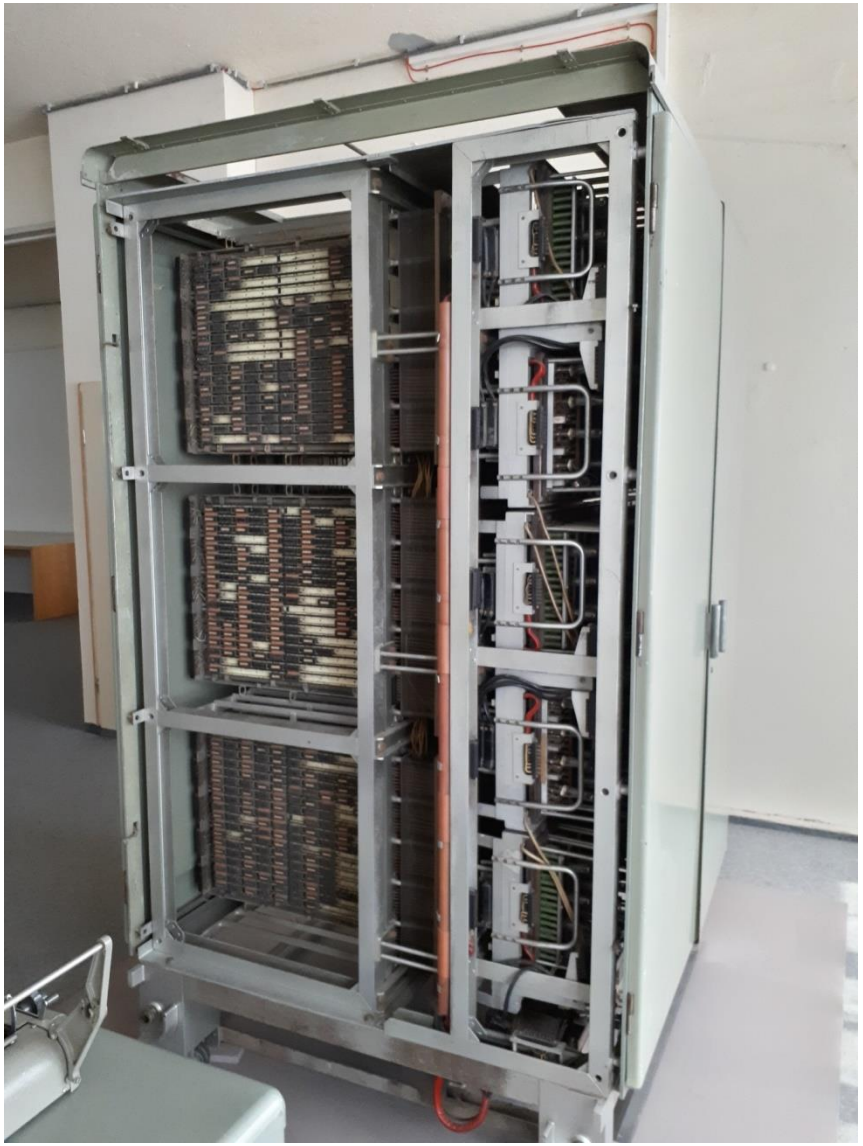
Ken Thompson and Dennis Ritchie



(http://commons.wikimedia.org/wiki/File:Ken_Thompson_%28sitting%29_and_De
Urheber: Peter Hamer; Lizenz: CC BY-SA 2.0)

Erster Rechner der DDR: ZRA

- von VEB Carl Zeiss
- Halbleiter-Dioden und Elektronenröhren
- Speicher: 4096 48-Bit-Worte
- Eingabe: Lochkartenleser
- Ausgabe: Druckwerk
- Platz: 6 × 8 m²
- Strukturell: von-Neumann-Architektur (gemeinsamer Programm- und Datenspeicher)
- acht Schnellspeicher
- Befehle der CPU: wie heute (Holen und Abspeichern von Daten im Hauptspeicher und den Prozessorregistern, Grundrechenarten, Fest- und Gleitkommazahlen, logische Operationen: Konjunktion und Disjunktion sowie Schiebeoperationen)
- 120 FLOPS (2005 3 Milliarden Flops)



Informatik: 2 Semester für Ingenieure

→Zum Lösen von Problemen mit dem Rechner braucht man
Programmierfähigkeiten (nur mit Übung möglich): Was ist Programmieren?
→Was ist ein Flussdiagramm?
Programmiersprache C:

- → Elementare Datentypen
- → Deklaration/Initialisierung
- → Kontrollstrukturen**:** if/else, while, for
- → Funktionen
- → Felder (Strings)
- → Zeiger
- → struct
- → Speicheranforderung: malloc
- → Listen
- → Bitmanipulation

→ Wie löst der Rechner unsere Probleme? → mit **Dualdarstellung** von Zeichen und Zahlen und mit Hilfe von **Algorithmen**
→ Ein Beispiel für ein Problem: **Kryptografie**
→ Sind Rechner auch Menschen? → **Künstliche Intelligenz**
→ Für alle Probleme gibt es viele Algorithmen. Welcher ist der Beste? → **Aufwand** von Algorithmen

Wozu muss ich Programmieren lernen?

Ingenieure sind immer an allem Schuld.

Boing 737 Max

Aktuelles Beispiel: Absturz in Indonesien 2018, Absturz in Äthopien 2019

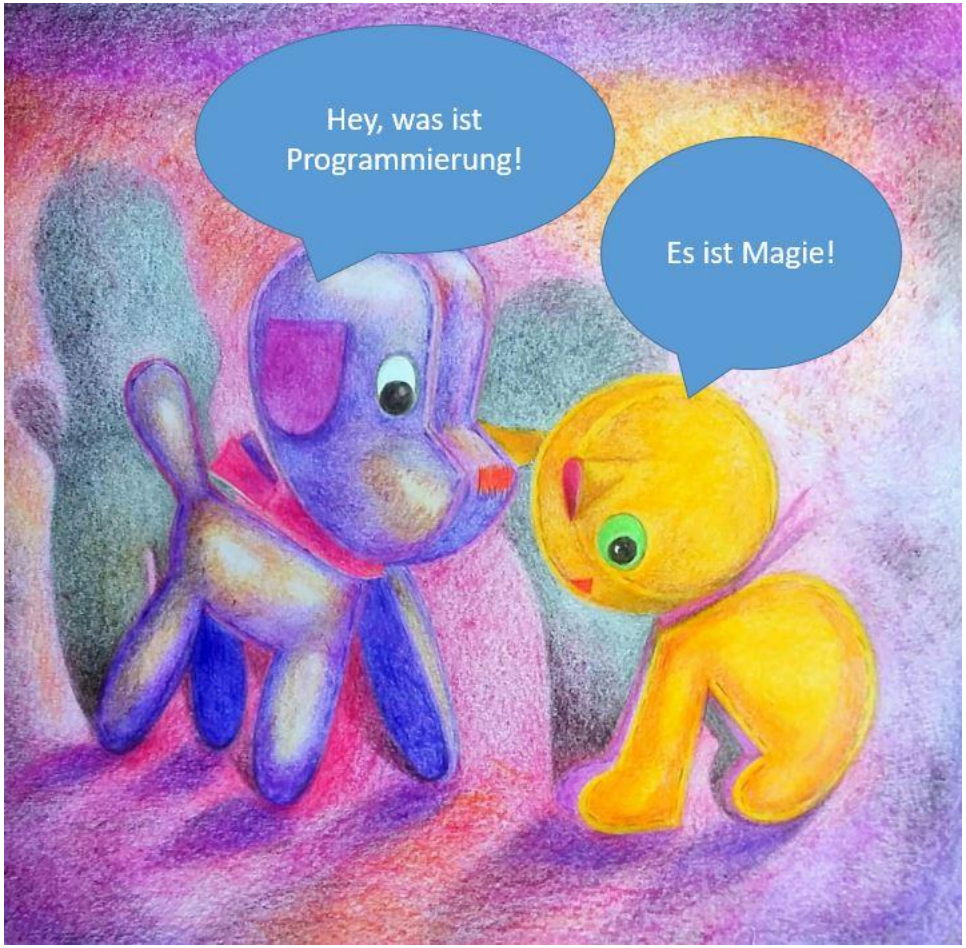
```
immer=1;
while(immer==1){
  nasenhoehe=messenNasenhoehe();
  if (nasenhoehe>…){
     anschaltenMCAS();//senkt Nase nach unten
  }
  else {
     //alles super
}
```

Wozu muss ich Programmieren lernen? → Zum Gedichte schreiben: Perl Poetry

```
#!/usr/bin/perl

APPEAL:

listen (please, please);

open yourself, wide;
    join (you, me),
connect (us,together),

tell me.

do something if distressed;

    @dawn, dance;
    @evening, sing;
    read (books,$poems,stories) until
peaceful;
    study if able;
```

```
  write me if-you-please;

sort your feelings, reset goals, seek (friends, family,
anyone);

        do*not*die (like this)
        if sin abounds;

keys (hidden), open (locks, doors), tell secrets;
do not, I-beg-you, close them, yet.

                                accept (yourself, changes),
                                bind (grief, despair);

require truth, goodness if-you-will, each moment;

select (always), length(of-days)

# listen (a perl poem)
# Sharon Hopkins
# rev. June 19, 1995
```

Was ist Programmieren?



Was ist Programmieren?

Aus Futurama (Matt Groening): Wohnungssuche in Neu-New York“

Ein Beispiel (Kühlschrank-Programm)

```
#include <stdio.h>
int main () {
    int bierflaschenanzahl = 50;
    while (bierflaschenanzahl > 0 ) {
      printf("Kuehlschrank auf.\n");
      printf("Flasche rausnehmen und trinken.\n");
      printf("Kuehlschrank zu.\n");
      bierflaschenanzahl = bierflaschenanzahl-1;
      printf("Es stehen noch %i leckere Bierchen im
            Kuehlschrank!\n",bierflaschenanzahl);
    }
    printf("Bier ist alle!\n");
    return 0;
}
```

Programmieren

Was ist Programmieren?

→ Schreiben von Programmen

Was ist ein Programm?

→ etwas Virtuelles, das nützliche Dinge für uns erledigt auf einem Rechner

Was muss man tun und für Charakterzüge haben, um ein Programm schreiben zu können?

→ eine Sprache erlernen

→ Problem verstehen

→ algorithmisches Denken

→ Hartnäckigkeit und Geduld

Algorithmisch denken

Es gibt 3 Möglichkeiten, aus denen ein Algorithmus aufgebaut ist (plus einige Dinge rings rum):

- Mach was - Dinge
- Mach was wieder und wieder - Dinge
- Mach was, falls –Dinge

Um algorithmisch zu denken, muss man:

- in einzelnen Schritten (unbedingt endlich viele Schritte!) denken
- einzelne Schritte präzise und eindeutig erklären (so dass keine andere Auslegung möglich ist)

Am besten nur noch wie folgt denken:

- tue solange wie ...
- wenn ..., dann ... ansonsten ...
- tue von ... bis ...und allem dazwischen

Dafür gibt es Strukturen: Sprach- und Datenstrukturen

ACHTUNG: ein wirkliche Methode, wie man einen Algorithmus findet, gibt es nicht. Leider!

Übung

Geben Sie jeweils ein Beispiel aus Ihrem Leben an:

1. Was machen Sie mehr als einmal? Wann hören Sie auf damit?
2. Was tun Sie abhängig von einer bestimmten Bedingung? Was tun Sie, wenn diese Bedingung nicht eintritt?

Beispielproblem

1. Stellen Sie sich vor, Ihnen werden nacheinander 20 Zahlen (also nicht alle gleichzeitig) gezeigt.
2. Denken Sie, dass Sie am Ende die größte Zahl benennen könnten?
3. Wie haben Sie das gemacht?
4. Schreiben Sie als Liste auf, wie Sie das gemacht haben. Versuchen Sie jeden einzelnen Schritt aufzuschreiben so präzise wie möglich und immer auf eine neue Zeile.
5. Sie haben Ihren ersten eigenen **Algorithmus** erstellt. Übrigens ist dies kein so einfacher Algorithmus. Es ist nicht klar, warum das eigentlich jeder kann. Wer hat Ihnen das beigebracht? Man lernt im Laufe seines Lebens eine Menge Dinge, deren man sich gar nicht bewusst ist. In der Informatik kann man einige dieser versteckten Fähigkeiten nutzen. Zwei weitere Dinge, die eigentlich jeder kann sind: Sortieren von Büchern und Suchen nach einem Buch in der Bibliothek.
6. Den Algorithmus für die **Suche nach der größten Zahl unter 20** zu programmieren, ist die viel einfachere Aufgabe, als die Idee für den Algorithmus selbst zu haben.
7. Hier lernen wir, wie man einen Algorithmus von Deutsch in eine Programmiersprache umschreibt. Dies ist am Anfang nicht so einfach, aber jede neue Sprache ist am Anfang ungewohnt und neu.
8. Der Algorithmus formuliert in einer Programmiersprachen (in unserem Fall wird das C sein), wird dann von einem Programm, genannt **Compiler** oder Übersetzungsprogramm, in Maschinensprache umgewandelt. Wenn Sie Fehler in der Sprache gemacht haben (eine Art Rechtschreibfehler), kann der Compiler Ihr Programm nicht übersetzen.
9. Das Programm in Maschinensprache kann dann Ihr Rechner ausführen.
10. Man muss dann noch testen, ob das Programm so abläuft, wie man das wollte. Man sollte das Programm mit verschiedenen Eingaben ausprobieren.

Apropo: Testen

- Flugzeugunglück in Barcelona
- Bug Bounty

Wie finden wir einen Algorithmus?

- Suche nach der HOLZHAMMERLÖSUNG (einfachste Lösung)
- Algorithmus in irgendeiner Sprache implementieren
- Suche nach einem besseren = effizienteren Algorithmus (hierbei kann der Holzhammeralgorithmus gut zum Testen dienen)
  - effizient in der Zeit
  - Algorithmus arbeitet gut mit tatsächlichen Daten, zufälligen Daten und extremen Daten

Programmiersprache

- übernimmt die Rolle eines Vermittlers zwischen Mensch und Maschine.
- bietet Möglichkeit zur Eingabe und Ausgabe von Daten
- Problem: endlicher Speicher

Programmiersprache C

- darin wurde UNIX programmiert
- designed for experienced programmers with the power to do whatever needs to be done
- this includes the power to hang yourself by invalid pointer references and array-out-of-bound violations
- Am Ende von 2 Semestern wissen Sie, was die 2 obigen Sätze bedeuten.

Programmieren

- Erlernen einer Sprache mit **Syntax** (Wörtern, Punkt, Komma, ...) und **Semantik** (Bedeutung)
- Kleiner Unterschied: in der Regel sprechen wir diese Sprache nicht mit einem Menschen
- Programm besteht aus
  - einem Anfang,
  - einem Ende und
  - dem Algorithmus dazwischen.
- Alles hat einen Anfang: In der Programmiersprache C sieht der Anfang wie folgt aus:

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

- Und das Ende wie folgt:

```
    return 0;
}
```

Das erste Programm (leicht erweitert)

```
#include <stdio.h>

int main() {
   printf("Hallo Welt!\n");
   printf("Hallo Studierende");
   return 0;
}
```

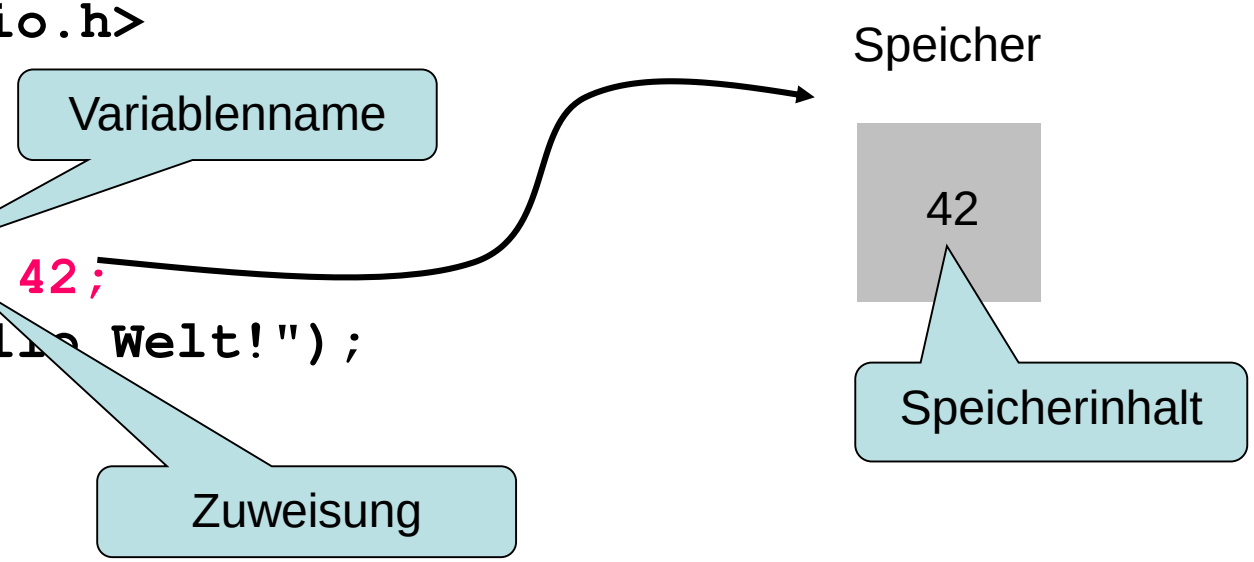
Wozu ist dieses erste Programm nützlich?

Dieses Programm ist gut, um zu wissen,

- dass ein Compiler auf unserem Rechner installiert ist,
- dass der Compiler funktioniert,
- dass wir mindestens ein Programm schon mal hinbekommen haben haben.

Ganzzahlen (integer)

```
#include <stdio.h>

int main(){

    int zahl = 42;
    printf("Hallo Welt!");
    return 0;
}
```



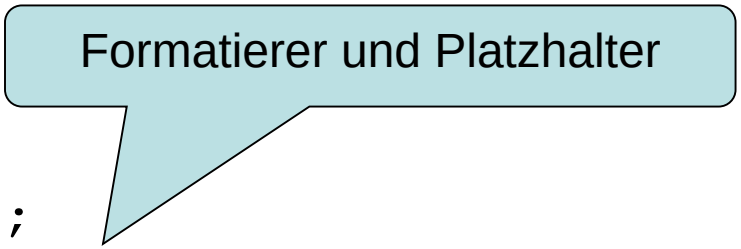
Zuweisung: Schreiben eines Wertes in den Speicher unseres Rechners. Solange obiges Programm ausgeführt wird, können wir den Speicherplatz mit dem Namen **`zahl`** ansprechen.

Achtung: Zuweisungsoperator = ist kein mathematisches Gleichheitszeichen! Der Ablauf einer Zuweisung ist wie folgt:

1. Erst wird alles ausgeführt rechts vom = und das von links nach rechts.
2. Schreiben des (eventuell berechneten) Wertes auf den Speicherplatz, der links vom = benannt wird.

Ganzzahlausgabe

```
#include <stdio.h>

int main() {

    int zahl = 42;
    printf("Zahl = %i",zahl);
    return 0;
}
```



Die Ausgabe auf dem Rechnerbildschirm sieht wie folgt aus: Zahl = 42

Funktion aus der Bibliothek **`stdio.h`** zur Ausgabe auf dem Bildschirm: **`printf()`**

Bemerkung: Formatierer i wie integer

Ganzzahleingabe

```
#include <stdio.h>

int main(){

    int zahl = 42;
    scanf("%i",&zahl);
    printf("Zahl = %i",zahl);
    return 0;
}
```

Speicheradresse, wo die Zahl hingeschrieben werden soll.

Funktion aus der Bibliothek **`stdio.h`** zur Eingabe: **`scanf()`**

Übung: Fehlerprogramm

Finden Sie Syntaxfehler:

```
1: #insert <stdio.h>

2: int program(){

3: int zahl1 = 15;
4: int zahl1 = 0;

5:  print("Zahl   = %i \n",zahl1);
6:  scanf("%h",zahl2);

7:  printf("Zahl   = %i \n",zahl2)
8:  printf("Summme = %I \n",{zahl1+zahl2});
9:  return 0;
}
```

Typische Anfangsfehler

Hilfe bei der Fehlersuche: https://www.youtube.com/watch?v=Ji6bMBDuqKU

Coco startet mit der C Programmierung Teil 2 - Fehler beim Programmieren

Ingenieure sind

Informatiker, die keine Angst vor Strom haben.

Begriffe

- **Hauptprogramm**: `int main (){return 0;}`

Achtung: **`main`** darf es nur genau einmal pro Programm geben!

- **Anweisung**: alles, wonach ein Semikolon steht
- Schlüsselwörter: **`int, return, include`**