Was ist Informatik?

- Informatik = Information + Automatik
  - Die Informatik ist die Wissenschaft von der maschinellen Informationsverarbeitung
- Computer Science
  - die Wissenschaft vom Rechnen (und von Rechnern)

→ Informationsverarbeitung wird letzendlich auf mathematische Grundlagen zurückgeführt.

Analoge Systeme

- Analoge Messung:
  - Darstellung von Information auf einer stufenlosen Skala
  - kann (im Prinzip) beliebig feine Unterschiede darstellen
  - E.g. Thermometer, Geschwindigkeitsanzeige, Plattennadel,...
- Analoge Steuerung:
  - Umsetzung von gewünschten Werten auf einer stufenlosen Skala
  - E.g. Temperatur-Regler, Gas-Pedal, Lautsprecher, ...

Digitale Systeme

- Digitale Messung:
  - Darstellung von Informationseinheiten auf einer Skala mit fixen Stufen (z.B. einer endlichen Zahlenmenge)
  - kann nur endlich viele Zustände darstellen
  - E.g. Ein/Aus, Zählen, ...
- Digitale Steuerung:
  - Umsetzung von gewünschten Werten auf einer Skala mit fixen Stufen
  - E.g. Brenner ein/ausschalten, Gang-Schaltung, ...

Automation

- Information (Meßwerte und Sollwerte) durch automatische Regler in Beziehung zu setzen
  - e.g., Thermostat, Tempomat, Automatik-Getriebe...
- Der Zusammenhang zwischen Meßwerten (Eingabe E) und Sollwerten (Ausgabe A) kann als mathematische Funktion f gedacht werden

$$A = f(E)$$

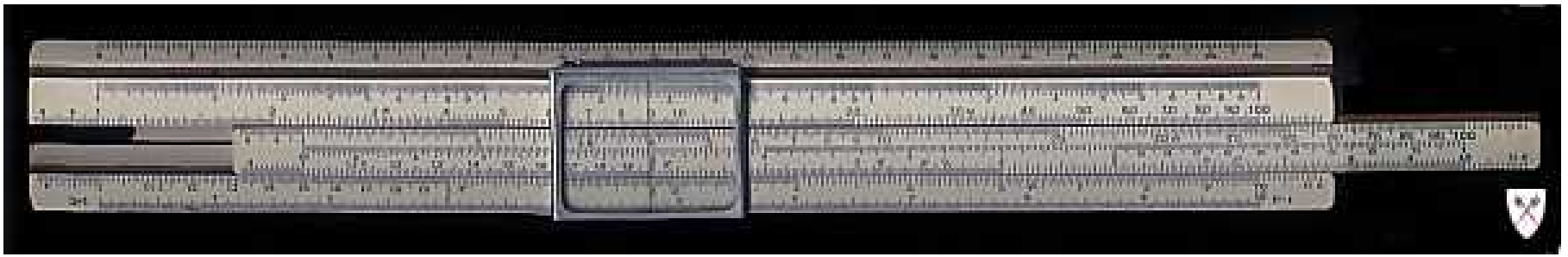
- Die Wissenschaft von solchen Steuer- und Regelkreisen nennt man auch Kybernetik

Rechenmaschinen

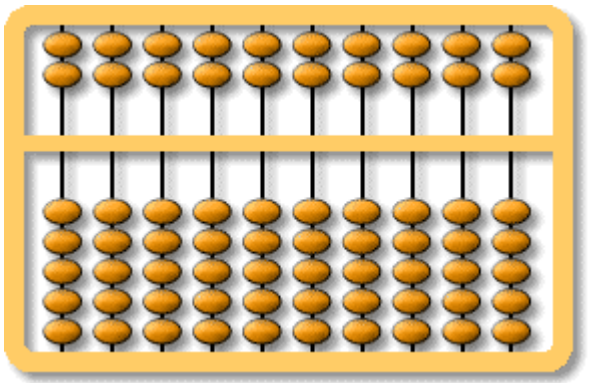
- Analoge Regler sind üblicherweise von der konkreten Problemstellung abhängig
  - Ein Gangschaltung funktioniert anders als ein Thermostat
  - obwohl die grundlegenden Funktionen gleich bzw. sehr ähnlich sind
    - Berechnung des Sollwerts aus dem Ist-Wert
- Wünschenswert sind universell einsetzbare Hilfsmittel um den funktionalen Zusammenhang zwischen Ein- und Ausgabe zu modellieren
  - keine Information über das konkrete Problem
  - Funktionalität zur Berechnung einiger weniger elementarer "Grundrechnungsarten"
  - Aufgabe muß in solche Elementarschritte zerlegt werden

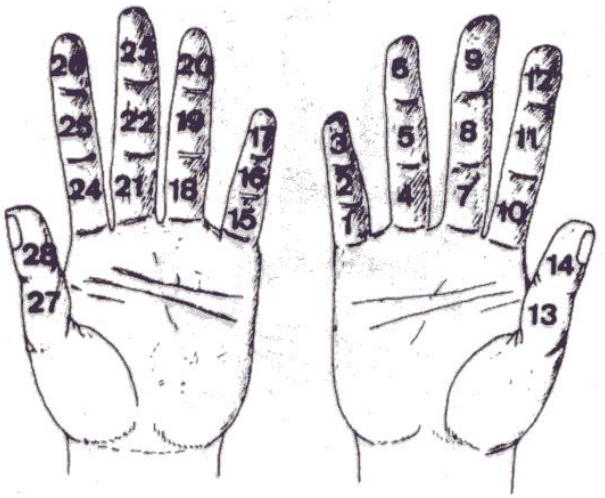
Rechenmaschinen

- analoge Rechenmaschine



- digitale Rechenmaschinen (lat. digitus = Finger)





Digitalisierung

- Der Durchbruch der digitalen Rechner kam durch die Einsicht, daß sich jegliche Form der Information in digitaler Form darstellen läßt
  - Zeichen / Buchstaben (e.g., ASCII code, Unicode, ...)
  - ganze Zahlen (Zeichen mit Ordnungsrelation)
  - reelle Zahlen (ganze Zahl + Information wo das Komma ist)
  - Texte (Aufeinanderfolge von Buchstaben)
  - Bilder (drei Farbwerte für eine endliche Anzahl von Punkten)
  - Multimedia (jpeg, MP3, mpeg,...)
  - ...
- Analoge Geräte sind daher zunehmend im Verschwinden, digitale Computer im Vormarsch
  - Digitale Meßinstrumente, Plattenspieler / CD, Digital Video, ...

Allgemeines Rechner-Modell

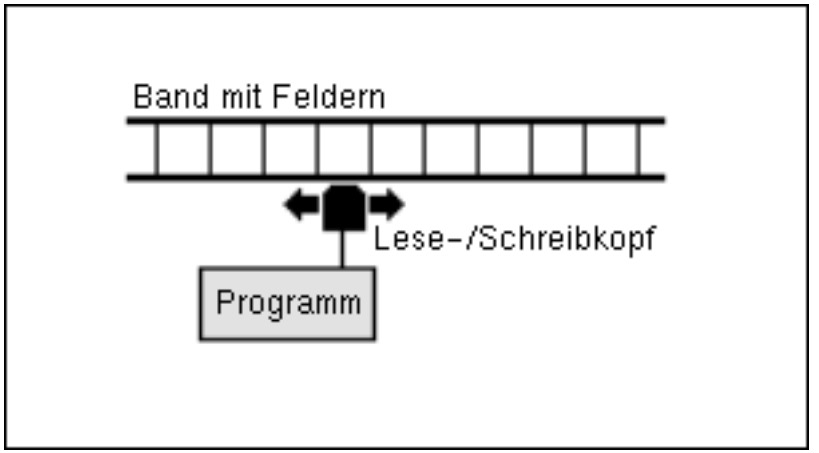
- Eingabe E:
  - Daten, für die die Berechnung durchgeführt werden soll
- Programm f:
  - Anweisungen, wie die Berechnung für beliebige Eingabewerte durchgeführt werden soll
- Ausgabe A:
  - Ergebnis der Berechnung

- Prozessor:
  - führt die Berechnung durch, in dem es das Programm auf die Eingabe anwendet

Turing Maschine

- theoretisches Rechner-Modell
  - entworfen von Alan Turing
- unendlich langes Speicherband mit unendlich vielen Feldern zur Speicherung genau eines Zeichens
  - auf diesem Band stehen die Eingabe-Daten
  - und am Ende das Ergebnis der Berechnung
- steuerbarer Schreib- und Lesekopf

- Elementare Anweisungen:
  - schreibe ein Zeichen
  - lese ein Zeichen
  - bewege Dich einen Schritt nach links/rechts



Turing Maschine (2)

- ein Prozessor mit endlich vielen internen Zuständen
  - darunter einige ausgezeichnete “Endzustände”
- eine Schalttafel (Programm), das die Maschine steuert
  - abhängig vom internen Zustand und vom Zeichen, das sich gerade am Band befindet
    - ändert der Prozessor seinen Zustand
    - wird ein Kommando an den Schreib- Lesekopf gesandt
  - bis ein Zustand erreicht ist, der als “Endzustand” charakterisiert ist
    - dann befindet sich die Ausgabe auf dem Band

Universelle Turing-Maschine

- Man kann für jedes (?) Rechen-Problem eine eigene Turing-Maschine bauen
  - passendes Programm entwerfen
  - Eingabe auf das Band kodieren
  - Ausgabe vom Band dekodieren
- Es läßt sich zeigen, daß man eine Universelle Turing-Maschine bauen kann, die
  - ein generisches Programm hat
  - zu Beginn eine Kodierung des eigentlichen Programms vom Band liest
  - und den Ablauf dieses Programms simuliert

Der Computer als Universelle Rechenmaschine

- Turing-Maschinen sind nur theoretische Modelle
  - obwohl es Simulationen am Web gibt (z.B. http://www.ifi.unizh.ch/groups/richter/achatz/)
- Aber die Grundbestandteile modernen Computer sind die gleichen
  - Speicherband = Memory
  - Prozessor = CPU

  → von Neumann Rechner-Modell

  → Ein moderner Computer kann im Prinzip nicht mehr (oder weniger) als eine Turing-Maschine!
- Die entscheidende Einsicht ist, daß sich Programme in gleicher Weise darstellen lassen wie Daten
  - dadurch wird eine "universelle" Rechenmaschine möglich

Was ist berechenbar?

- Church-Turing These

> Alles, was in einem intuitiven Sinn berechenbar ist, kann von einem Computer berechnet werden

  - unbeweisbare, aber allgemein akzeptierte Vermutung
- Berechenbar in einem intuitiven Sinn
  - Keine strenge Formalisierung.
  - Idee:
    - alles was ein Mensch im Prinzip mit Papier und Bleistift bewerkstelligen kann
    - durch Befolgung einer festen Abfolge von Anwendungen
    - ohne daß dabei eine besondere Intelligenz-Leistung von nöten wäre
    - Faustregeln, mechanische Berechnungen
  - daher unbeweisbare Behauptung

Intelligente Computer

- Computer können mittlerweile Aufgaben erledigen, für deren Lösung man “Intelligenz” vermuten würde
  - Schach spielen
  - automatische Übersetzung von Texten
  - Planung komplexer Produktionsabläufe
  - Interpetation von Musik
  - u.v.m.
- Diese Probleme lassen sich alle “berechnen”
  - für viele dieser Probleme (z.B. Schach) dachte man, daß das nicht möglich sei
- Lassen sich alle menschlichen Verstandesleistungen mit dem Computer nachvollziehen?
  - Intuition, Kreativität, Intelligenz,...

Artificial Intelligence/ Künstliche Intelligenz

- Philosophische Debatte
  - Hard AI:
    - glaubt, daß sich alle Gedankenprozeße auf das Manipulieren von Symbolen zurückführen lassen i.e., mit digitalen Computern simulieren lassen
  - Soft AI:
    - glaubt, daß subsymbolische Prozesse notwendig sind (z.B. das Verhalten der Neuronen im Gehirn). Bei digitaler Simulation geht zu viel Information verloren
  - Skeptiker:
    - wesentliche Teile des menschlichen Verstands können nicht simuliert werden (Geist, Seele,...)
- Schwerpunkt liegt mittlerweile auf Problemlösungen
    - unter Ausnutzung der Stärken des Computers
    - ohne Modellierung des menschlichen Denkens

Algorithmus

- Algorithmus = “intuitive” Rechenvorschrift
  - Al-Chwarizmi (783-ca.850): Erstes Buch über (algebraische) Rechenvorschriften

- Ein Algorithmus ist ein Verfahren zur schrittweisen Lösung einer Klasse von Problemen
  - deterministisch
    - die Abfolge der Schritte ist für das gleiche Problem immer gleich
  - terminiert in endlicher Zeit
    - irgendwann hat man ein Resultat
  - produziert immer das gewünschte Resultat
    - kann aber auch z.B. eine Näherung sein

Algorithmus: Wiener Schnitzel

Zutaten für 4 Portionen

4 Kalbsschnitzel vom Schlegel mit je 120 - 140 g	Salz
80 g Semmelbrösel	200 g Schweineschmalz
60 g Mehl	2 Eier
1 Zitrone	

1. Die Schnitzel nicht zu dünn klopfen (ca. 4 mm) und die feinen Hauträndern ganz leicht einschneiden.
2. 3 Teller oder flache Schüssel für die Panier vorbereiten. 1 Teller mit Mehl, 1 Teller mit den mit ganz wenig Wasser oder Milch verquirlten Eiern und 1 Teller mit den Semmelbröseln.
3. Schnitzel auf beiden Seiten salzen.
4. Inzwischen in einer großen Pfanne das Schweineschmalz (Butterschmalz ist auch OK) erzhitzen. Mindestens daumendick sollte das Fett in der Pfanne sein.
5. Die Schnitzel im Mehl wenden und leicht abklopfen - das Fleisch soll nur eine hauchdünne Schicht Mehl annehmen.
6. Die bemehlten Schnitzel durch die Eier ziehen, abrinnen lassen und sofort in den Semmelbröseln wenden.
7. Überflüssige Brösel abschütteln. Niemals die Brösel festdrücken - sie dürfen nicht zu fest am Schnitzel kleben!
8. Die panierten Schnitzel sofort im heißen Fett backen!
9. Die Schnitzel müssen genügend Platz haben und im Fett schwimmen.
10. Die Unterseite sollte nach längstens 2 Minuten fertig sein - dann Schnitzel wenden und fertigbacken.
11. Während des Backens die Pfanne wiederholt schütteln sodass das heiße Fett auch über die obere Seite der Schnitzel hinwegspült damit die Panier schön aufgehen kann.
12. Der Backprozess soll sehr rasch vor sich gehen.
13. Das Wiener Schnitzel muss fett-trocken sein! Dafür muss man es gut abtropfen lassen sobald es aus dem Fett genommen wird.

source `http://helena.ludwig.name/Helenas Kochbuch/wiener_schnitzel.htm`

Algorithmus: Wiener Schnitzel

Zutaten für 4 Portionen

4 Kalbsschnitzel vom Schlegel mit je 120 - 140 g	Salz
80 g Semmelbrösel	200 g Schweineschmalz
60 g Mehl	2 Eier
1 Zitrone	

Eingabe E

Berechnungsvorschrift f

1. Die Schnitzel nicht zu dünn klopfen (ca. 4 mm) und die feinen Hauträndern ganz leicht einschneiden.
2. 3 Teller oder flache Schüssel für die Panier vorbereiten. 1 Teller mit Mehl, 1 Teller mit den mit ganz wenig Wasser oder Milch verquirlten Eiern und 1 Teller mit den Semmelbröseln.
3. Schnitzel auf beiden Seiten salzen.
4. Inzwischen in einer großen Pfanne das Schweineschmalz (Butterschmalz ist auch OK) erzhitzen. Mindestens daumendick sollte das Fett in der Pfanne sein.
5. Die Schnitzel im Mehl wenden und leicht abklopfen - das Fleisch soll nur eine hauchdünne Schicht Mehl annehmen.
6. Die bemehlten Schnitzel durch die Eier ziehen, abrinnen lassen und sofort in den Semmelbröseln wenden.
7. Überflüssige Brösel abschütteln. Niemals die Brösel festdrücken - sie dürfen nicht zu fest am Schnitzel kleben!
8. Die panierten Schnitzel sofort im heißen Fett backen!
9. Die Schnitzel müssen genügend Platz haben und im Fett schwimmen.
10. Die Unterseite sollte nach längstens 2 Minuten fertig sein - dann Schnitzel wenden und fertigbacken.
11. Während des Backens die Pfanne wiederholt schütteln sodass das heiße Fett auch über die obere Seite der Schnitzel hinwegspült damit die Panier schön aufgehen kann.
12. Der Backprozess soll sehr rasch vor sich gehen.
13. Das Wiener Schnitzel muss fett-trocken sein! Dafür muss man es gut abtropfen lassen sobald es aus dem Fett genommen wird.

source `http://helena.ludwig.name/Helenas Kochbuch/wiener_schnitzel.htm`

Kochrezept: Maximum Finden

- Zutaten: 100 Zahlen
- Ergebnis: die größte Zahl
- Rezept:

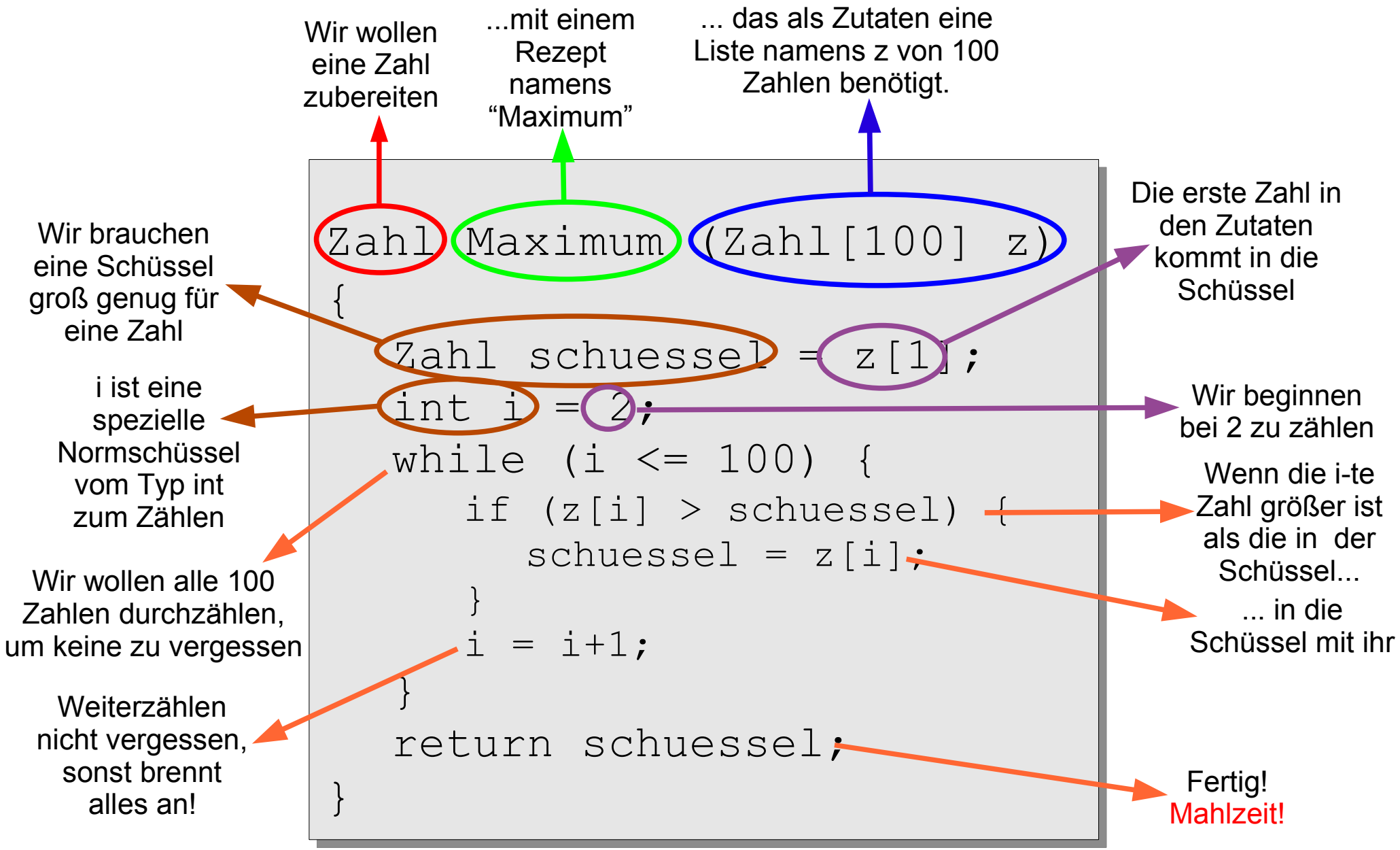
1. Man gebe die erste Zahl in die Schüssel
2. Für jede weitere Zahl
   - Wenn die Zahl größer ist als die in der Schüssel:
     - man nehme die alte Zahl aus der Schüssel und gebe die neue hinein
3. Serviere die Zahl in der Schüssel

Algorithmus & Programm

- Algorithmus:
  - abstrakte Definition einer Vorgehensweise
  - verläßt sich auf “universelle” Grundbausteine
- Programm:
  - konkrete Umsetzung eines Algorithmus
  - in einer bestimmten Umgebung von Grundbausteinen (Küche/Programmiersprache)
  - je nach Umgebung kann die Umsetzung verschieden effizient erfolgen (z.B. Gasherd/E-herd/Mikrowelle, Kochzeile/Betriebsküche bzw. Java/C/Lisp/Prolog)

Programm: Maximum Finden

```
Zahl Maximum (Zahl[100] z)
{
   Zahl schuessel = z[1];
   int i = 2;
   while (i <= 100) {
      if (z[i] > schuessel) {
         schuessel = z[i];
      }
      i = i+1;
   }
   return schuessel;
}
```


Wir wollen eine Zahl zubereiten
...mit einem Rezept namens “Maximum”
... das als Zutaten eine Liste namens z von 100 Zahlen benötigt.
Zahl Maximum (Zahl[100] z)
{
Zahl schuessel = z[1];
int i = 2;
while (i <= 100) {
if (z[i] > schuessel) {
schuessel = z[i];
}
i = i+1;
}
return schuessel;
}
Die erste Zahl in den Zutaten kommt in die Schüssel
Wir brauchen eine Schüssel groß genug für eine Zahl
i ist eine spezielle Normschüssel vom Typ int zum Zählen
Wir beginnen bei 2 zu zählen
Wenn die i-te Zahl größer ist als die in der Schüssel...
... in die Schüssel mit ihr
Wir wollen alle 100 Zahlen durchzählen, um keine zu vergessen
Weiterzählen nicht vergessen, sonst brennt alles an!
Fertig!
Mahlzeit!

Grundlegendes Problem der Programmierung

- Der Mensch ist gescheit, aber unpräzise
  - kann schwierige Probleme lösen
  - kann aber oft den Lösungsvorgang nicht exakt beschreiben
- Ein Computer ist dumm, aber genau
  - im Prinzip kann er nur bis zwei zählen, sonst nichts.
  - aber das kann er fehlerlos und schnell

- Programmiersprachen versuchen, einen Kompromiss zu finden, der es
  - Menschen erlaubt, sich flexibel auszudrücken
  - Computern erlaubt, die Sprache eindeutig in logische Ausdrücke zu übersetzen

Maschinensprache

- Dient direkt zur Steuerung des Prozessors
  - hängt von der Architektur des Prozessors ab, d.h. jeder Rechner hat seine eigene Maschinensprache
- Grundlegende Maschinen-Befehle lassen sich in folgende Kategorien unterteilen:
  - **Arithmetische Operationen:** Führen Berechnungen durch
  - **Speicheroperationen:** Übertragen Daten zwischen Prozessorregistern und Speicher
  - **Vergleichsoperationen:** Vergleich von Werten
  - **Steueroperationen:** Verzweigungen, die den Ablauf des Programms beeinflussen
- Befehle und Argumente als Binärstrings kodiert
  - Folgen von 0 und 1
  - können vom Rechner direkt interpretiert und ausgeführt werden

Assembler-Code

- Ersetzt die Binärstrings der Maschinen-Befehle durch mnemonische Codes
  - Übersetzung durch einfache Tabelle
  - Dadurch Programmierung (ein wenig) leichter
- Beispiel
  - Der Maschinen-Befehl 10110000 01100001
    in der Maschinensprache von x86-Prozessoren (Intel)
  - entspricht dem Assemblerbefehl mov al, 61h
  - und bedeutet, dass der hexadezimale Wert 61 (97 dezimal) ins Register ‚al' geladen werden soll
- maschinennahe Programmierung dennoch mühselig
  - große konzeptuelle Distanz zwischen Algorithmus und Implementierung

Höhere Programmiersprachen

- Mittler zwischen Mensch und Maschine:
  - Mensch übersetzt Algorithmen in Programmiersprache
  - Computer übersetzt Programmiersprache in Maschinensprache
    - Compiler: übersetzt das Programm vor der Ausführung
    - Interpreter: “dolmetscht” das Programm während der Ausführung
- Syntax:
  - Definition der zulässigen Worte bzw. Sätze in einem Programm (z.B. `int, while, =, 1, 2, ..`)
- Semantik:
  - Definition der Bedeutung der Syntax
  - Programmbibliotheken (Libraries):
    - Sammlungen vordefinierte Routinen, die von einem Programm eingebunden werden können (z.B. “Wasser kochen”)
    - vereinfachen und verbessern Programmierung durch Wiederverwendung

Typen von Programmiersprachen

- Prozedurale (Imperative) Programmiersprachen
  - Das Programm ist eine Abfolge von Befehlen
  - z.B. Fortran, Basic, Cobol, C, Pascal, Modula, etc.
- Funktionale Programmiersprachen
  - Das Programm wird als eine Verschachtelung mathematischer Funktionen verstanden (ein Programm ist eine Abbildung Eingabe-Daten auf Ausgabe-Daten)
  - z.B. Lisp, ML, Miranda, Haskell
- Logische Programmiersprachen
  - Ein Programm ist ein logischer Ausdruck
  - z.B. Prolog

Objekt-Orientierte Programmierung

- im Zentrum des Designs stehen nicht mehr Algorithmen, sondern Datenstrukturen
  - Daten und Methoden zu ihrer Behandlung werden in sogenannte Klassen zusammengefaßt
  - fördert modulare Programmierung und damit Wiederverwendbarkeit von Code bzw. Code-Teilen
- e.g., SmallTalk, C++, Java, C#, etc.

Die Wissenschaft Informatik

- Im deutschen Sprachraum unterscheidet man traditionell 4 Untergebiete
  - Technische Informatik
  - Praktische Informatik
  - Theoretische Informatik
  - Angewandte Informatik

Technische Informatik

- Beschäftigt sich mit den Geräten zur Informationsverarbeitung (Hardware)
  - Bauteile:
    - Prozessoren, Speicher, ..
  - Rechnerarchitekturen:
    - Welche Bauteile braucht man?
    - Wie setzt man die Bauteile zusammen?
    - logische Grundlagen des Rechnerbaus
  - Peripheriegeräte:
    - Drucker, CD, DVD, Bildschirme, ...
- Grenzen zur Elektrotechnik sind fließend

Praktische Informatik

- beschäftigt sich mit der den Programmen (Software), die für die Funktion des Computers notwendig sind
  - Betriebssysteme
  - Algorithmen und Datenstrukturen
  - Programmiersprachen, Compiler
  - Datenbanksysteme
  - Softwaretechnik
  - Rechnernetzwerke
- schlägt die Brücke zwischen Hardware und den Anwendungen, die auf dem Computer laufen sollen
  - nicht zu verwechseln mit “Angewandter Informatik”

Diese Vorlesung ist hauptsächlich über Praktische Informatik!

Theoretische Informatik

- beschäftigt sich mit theoretischen Grundlagen der Informatik
  - Automatentheorie, formale Sprachen
  - Algorithmentheorie
  - Berechenbarkeit
  - Komplexität
- Grenzen zwischen Theorie und Praxis sind fließend, z.B.
  - Implementierungen von theoretischen Automaten-Modellen bilden die Grundlage aller Compiler
  - Komplexitätsabschätzungen der Algorithmen sind auch für Anwendungen enorm wichtig

Angewandte Informatik

- beschäftigt sich mit dem Einsatz von Computersystemen in verschiedensten Anwendungsgebieten
- Die Hauptschwerpunkte sind
  - Mensch-Maschine Kommunikation
  - Schnittstellengestaltung
  - Benutzeroberflächen (GUIs)
  - Ergonomie
  - System-Design und Evaluierung
- zahlreiche Anwendungsgebiete
  - Office-Anwendungen
  - Multimedia
  - Entertainment
  - u.v.m.

Die Grenzen der Informatik

- Diese Unterteilung wird zunehmend problematisch
  - Grenzen zwischen den Teilgebieten sind fließend
  - wird dem Wachstum des Gebiets nicht mehr gerecht
  - Grenzen zu anderen Wissenschaften sind oft fließend
- Insbesondere in der Angewandten Informatik haben sich zunehmend Schwerpunkte entwickelt, die man mittlerweile oft als eigene Richtungen ansieht
  - Datentechnik
  - Wirtschaftsinformatik
  - Medizinische Informatik
  - Bioinformatik
  - Robotik
  - Cognitive Science und Artificial Intelligence
  - u.v.m.

Der Darmstädter Weg

nach Grundausbildung Aufteilung in 8 zukunftsorientierte Bereiche:

- Computational Engineering
  - Modellierung und Simulation, virtuelle Welten, Robotik, Hochleistungsrechnen
- Computer Microsystems
  - Mikroelektronische, Eingebettete Systeme, HW/SW-Systeme, Echtzeitsysteme, ...
- Data and Knowledge Engineering
  - Datenbanksysteme, Wissensbasierte Systeme, Data Warehouses, Data Mining, Maschinelles Lernen, ...

- Foundations of Computing
  - Algorithmen, ...
- Human Computer Systems
  - Mensch/Maschine Schnittstelle, Graphische DV, multimodale Systeme, e-Learning, ...
- Net Centric Systems
  - Rechnernetzwerke, Verteilte Systeme, Ubiquitous Computing, ...
- Software Engineering
  - Entwurfstechniken, -methoden und -werkzeuge, Organisation komplexer Software-Systeme, ...
- Trusted Systems
  - Sicherheit, Zuverlässigkeit, Kryptographie, Authentifizierung, ...

Grobplan für den Rest der Vorlesung

- Einführung ins Programmieren mit KarelJ
- Grundlagen der Informatik
- Grundlegende Konzepte der Programmierung
- Einführung in Java
- Objekt-Orientierte Programmierung in Java
- Gängige Java-Libraries