Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

**This is an excerpt of "PPRG – Prozedurale Programmierung" by Prof. Dr. Hans-Peter Hutter, Version 4.1.1 (German language). It contains the first chapter „Grundlagen" (of computer hard- and software) in minimally altered form to be used in the module IT_PROG.**

17 July 2014 / stdm

# 1 Grundlagen

## 1.1 Ziele

- Sie können das Wesen eines Computers erklären.
- Sie können die Arbeitsweise eines Computers anhand des EVA-Schemas abstrahieren.
- Sie können den Unterschied zwischen Daten und Programmen erklären.
- Sie können den Unterschied zwischen einem Programm und einem Prozess in eigenen Worten erklären.
- Sie können das Dateisystem und die wichtigsten Befehle dazu anhand einer Zeichnung erklären.

## 1.2 Was ist ein Computer?

- Hauptaktivitäten eines Computers:
  - Eingabe: Der Computer wartet auf die Eingabe von Daten in irgendeiner Form.
    - Tastatur, Maus
    - Signal von externer Leitung (LAN, Modem, etc.)
    - Zeitsignal von einer Uhr
    - Geld, Waren, etc.
  - Verarbeitung: Er verarbeitet die Eingabedaten (eventuell zusammen mit internen Daten).
    - selbständig (Rechenschieber ist noch kein Computer)
    - mehr als nur Grundoperationen (Einfacher Rechner, Schreibmaschine sind noch keine Computer)
  - Ausgabe: Er gibt Resultat (ebenfalls Daten) aus der Verarbeitung in irgendeiner Form zurück.
    - Bildschirm, Drucker, akustisch
    - Geld, Kaffee, Sandwich

## 1.3 Personal Computer (PC)

- Ein PC ist ein Computer, der für verschiedenste Aufgaben eingesetzt werden kann:
  - Bearbeiten von Texten, Zeichnungen, Bildern, Musik
  - Terminkalender
  - E-Mail
  - Internet-Zugang
  - uam.
- Was braucht es dazu?
  - Eine Komponente für die Datenverarbeitung: Prozessor.
  - Komponenten für Datenspeicherung: Arbeitsspeicher, Harddisk, CD-Rom, DVD, USB-Stick etc.
  - Komponenten für Dateneingabe: Tastatur, Maus, LAN, Modem
  - Komponenten für Datenausgabe: Bildschirm, Drucker, LAN, Modem
  - Programme für die verschiedenen Aufgaben

## 1.4 Arbeitsweise eines Computers

- Obwohl ein Computer die unterschiedlichsten Aufgaben erledigen kann, beruht seine Arbeitsweise auf einem einfachen Prinzip.
- Dieses kann mit folgender Analogie umschrieben werden:

Ein Computer gleicht einem Mann (oder einer Frau) mit einem Taschenrechner und zwei Stapeln von Rechnungen: Auf dem einen sind die unerledigten, auf dem anderen die erledigten. Der Mann nimmt eine Rechnung nach der anderen, und schaut, was er berechnen muss. Dann tippt er die Zahlen auf dem Blatt in den Rechner und gibt nacheinander die auszuführenden Operationen ein. Hin und wieder schreibt er sich ein Zwischenresultat auf einen Zettel. Schliesslich schreibt er das Ergebnis vom Display des Rechners ab und trägt es an der richtigen Stelle im Formular ein.

- Ein Computer setzt dieses Prinzip technisch um. Die Computer wurden im Verlauf der Geschichte stetig optimiert und stellen heute hochkomplexe technische Systeme dar.
- Nichtsdestotrotz kann die Arbeitsweise eines Computers auch heute noch für die meisten Programmieraufgaben mit dem einfachen Schema in Abbildung 1 abstrahiert werden.

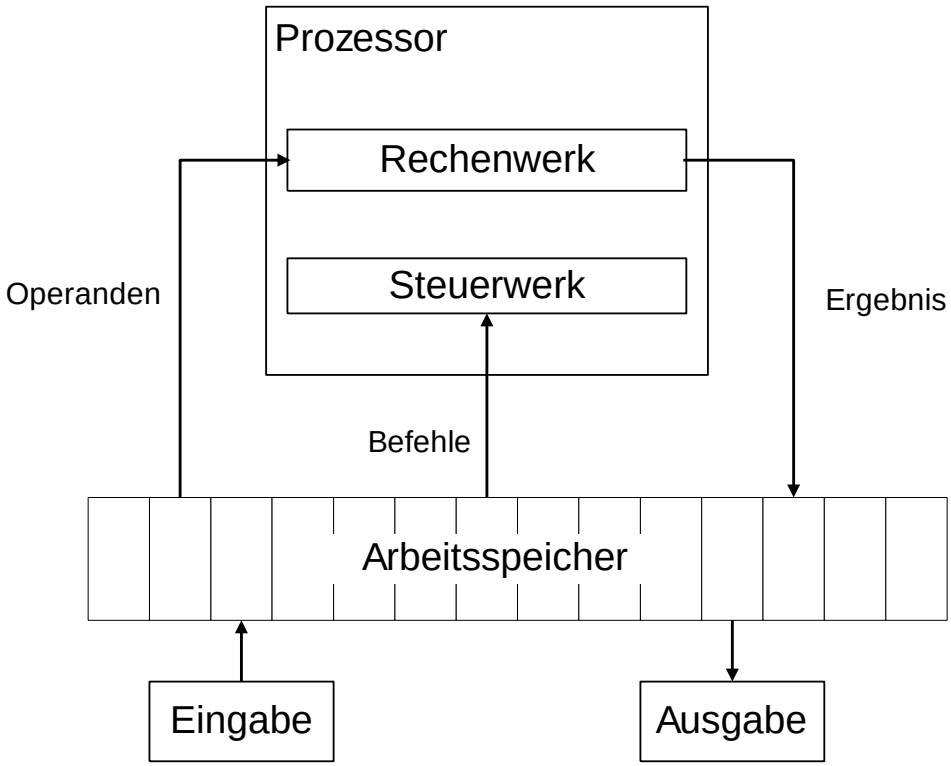


Abbildung 1: EVA-Schema eines Computers

- Das sogenannte EVA-Schema (EVA: Eingabe-Verarbeitung-Ausgabe) besteht aus dem Prozessor, dem Arbeitsspeicher sowie einer Eingabe und einer Ausgabe.
- Die einzelnen Komponenten in diesem Schema haben folgende Aufgaben:
  - Prozessor
    - Der Prozessor ist das Herz des Computers. Er arbeitet eng mit dem Arbeitsspeicher zusammen.
  - Er hat einen fixen Befehlssatz (Grundoperationen, Transportbefehle, Sprünge).
    - Beispiel einer typischen Operation:
      - Der Prozessor holt zwei Zahlen aus dem Arbeitsspeicher, addiert sie, und schreibt das Resultat in den Arbeitsspeicher zurück.
    - Der Prozessor kann in zwei weitere Einheiten unterteilt werden:
      - Rechenwerk
        - Das Rechenwerk entspricht dem Taschenrechner von vorher.
        - Es führt die Grundrechenarten `+ - * /` auf den Eingabewerten (Operanden) aus

und berechnet daraus das Ergebnis.
    - Steuerwerk:
        - Das Steuerwerk entspricht dem Mann, der den Taschenrechner bedient.
        - Es steuert den Ablauf der Berechnungen.
        - Dazu holt es einen Befehl nach dem andern vom Arbeitsspeicher und interpretiert den Befehl.
        - Es sagt dem Rechenwerk, welche Operation als nächstes auszuführen ist und stellt die Operanden für das Rechenwerk bereit.
        - Es sorgt dafür, dass das Ergebnis wieder in den Arbeitsspeicher abgelegt wird.
        - Schliesslich koordiniert es auch das Schreiben und Lesen des Arbeitsspeichers (sowohl vom Prozessor als auch von den E/A-Geräten).
- Arbeitsspeicher
    - Der Arbeitsspeicher speichert alle Daten, die der Prozessor im Moment braucht.
    - Er speichert zudem auch die Befehle (Instruktionen), die dem Prozessor sagen, was er machen muss.
    - Man nennt eine solche Prozessorarchitektur, bei der Befehle und Daten im gleichen Arbeitsspeicher liegen, *Von-Neumann-Struktur*.
    - Der Arbeitsspeicher besteht aus einer grossen Anzahl einzeln ansprechbarer Speicherzellen.
    - Eigenschaften einer Speicherzelle
        - Sie kann entweder ein Datum (Zahl, Zeichen, etc.) oder einen Befehl für den Prozessor enthalten. Sie hat eine fixe Länge 8, 16, 32 oder 64 Binärstellen (Bits)
        - Eine Binärstelle wird als **Bit** (**Binary Digit** oder deutsch Binärziffer) bezeichnet. Ein Bit kann nur zwei mögliche Werte annehmen: 0 oder 1.
        - 8 Bit werden häufig zu einem **Byte** zusammengefasst. Mit einem Byte sind dem-entsprechend 256 (= $2^8$ ) verschiedene Werte darstellbar.
        - **Word** bezeichnet die Anzahl Bits, die der Prozessor gleichzeitig verarbeiten/lesen/speichern kann (früher 8 oder 16, heute meist 32 oder 64 Bits)
    - Jede Speicherzelle hat eine eindeutige Adresse (0..MaxSpeicher), mit der sie angesprochen wird.
    - Der Adressraum, d.h. die maximal adressierbare Anzahl Speicherzellen eines Prozessors sind durch seine Anzahl Adressleitungen, in der Regel 32 (heute sogar 64), gegeben. Er beträgt somit $2^{32}$ = 4G (Giga = Milliarden) adressierbare Speicherplätze.
    - Eine Speicherzelle wird immer als Ganzes gelesen und geschrieben.
    - Wird ein neuer Wert in eine Speicherzelle geschrieben, so geht der alte Wert verloren.
    - Beim Ausschalten gehen alle Daten im Arbeitsspeicher verloren.
    - Das Steuerwerk des Prozessors legt genau fest, wann Daten in den Arbeitsspeicher geschrieben bzw. herausgelesen werden dürfen (synchroner Ablauf). Eine Speicherzelle darf nicht gleichzeitig beschrieben und ausgelesen werden.
    - Dank der synchronen Arbeitsweise des Prozessors ist der Ablauf eines korrekten Programms vollständig reproduzierbar, d.h. es gibt immer das gleiche Resultat.
- Eingabe
    - Die Eingabe bezeichnet Komponenten, die Daten von extern in den Arbeitsspeicher schreiben.
    - Beispiele für solche Komponenten sind die Tastatur, die Harddisk, das Modem etc.
    - In der Analogie im obigen Beispiel entspricht dies dem Stapel *unerledigter* Rechnungen.
- Ausgabe
    - Die Ausgabe bezeichnet Komponenten, die Daten vom Speicher abholen und sie nach aussen geben.

  - Beispiele für Ausgabekomponenten sind der Bildschirm, der Drucker, das Modem etc.
  - In der Analogie im obigen Beispiel entspricht dies dem Stapel mit *erledigten* Rechnungen.

- **Befehlszyklus:**
  Der Prozessor in Abbildung 1 führt immer denselben sogenannten Befehlszyklus aus:
  1. Nächsten Befehl vom Arbeitsspeicher holen und interpretieren
  2. Operanden vom Arbeitsspeicher holen
  3. Berechnung ausführen
  4. Ergebnis in den Arbeitsspeicher ablegen

  Der gesamte Befehlszyklus und alle Lese- und Schreiboperationen auf dem Arbeitsspeicher laufen synchron zum Prozessortakt ab.

## 1.5 Programme

- Woher kommen die Befehle, die dem Prozessor sagen, was er tun muss?
- Die Befehle für den Prozessor stehen in den sogenannten *Programmen.* Sie sagen dem Prozessor, welche Befehle er der Reihe nach ausführen muss.
- Man nennt die Programme häufig auch *Software,* im Gegensatz zur *Hardware* (Computer, Bildschirm, Tastatur, etc.)
- Programm
  - Programme kann man kaufen, kopieren oder auch selber programmieren.
  - Ein Programm ist normalerweise auf HD, CDROM etc. gespeichert.
  - Es muss zuerst in Arbeitsspeicher geladen werden.
  - Sodann muss der Prozessor wissen, wo der 1. Befehl des Programms steht (Einstiegspunkt).
  - Nun führt der Prozessor die Befehle des Programms einen nach dem andern aus.
  - Ein Programm ist also nichts anderes als eine Abfolge (Sequenz) von Prozessorbefehlen, den sogenannten *Maschinenbefehlen* sowie den dazugehörigen Daten.
  - Der Prozessor holt die Befehle der Reihe nach aus dem Arbeitsspeicher und arbeitet einen Befehl nach dem anderen ab.
  - Der Prozessor weiss immer nur gerade, wo der nächste Befehl steht.
  - Der Befehlssatz des Prozessors ist fix vorgegeben. Das Programm bestimmt nur, welche Befehle und in welcher Reihenfolge sie ausgeführt werden (Sprungbefehle).
- Arten von Programmen
  Die folgenden Liste zählt verschiedene Arten von Programmen beispielhaft auf:
  - Monitorprogramm (Startprogramm des Computers)
  - Betriebssystem
    - Verwaltet Prozesse, Dateien, Ein-/Ausgabegeräte
    - Stellt Benutzerschnittstellen zur Verfügung
      - Graphische Benutzeroberfläche
        - interpretiert Benützer-Eingaben über die Tastatur und über die Maus
      - Kommandozeile (Kommando-Interpreter)
        - interpretiert nur Befehle, die der Benützer über die Tastatur eingibt
        - Diese Benützerbefehle werden direkt in die entsprechende Maschinenbefehle umgesetzt und ausgeführt.
  - Hilfsprogramme
    - Treiber für Drucker, Bildschirm, Harddisk etc.
    - Anti-Virus-Programme
  - Standardsoftware

        - Textverarbeitung
        - Grafikprogramme
        - Datenbanken
        - Internet-Browser
        - Mail
        - Tabellenkalkulationsprogramme
        - Mathematik, Statistik
    - Compiler für Programmiersprachen
        - Die Maschinenbefehle sind prozessorabhängig.
        - Ein Programm wird heute nur noch äusserst selten direkt in Maschinenbefehlen geschrieben, da dies sehr mühsam und unübersichtlich ist.
        - Stattdessen werden Programme in einer höheren Programmiersprache geschrieben und dann von sogenannten *Compilern* in die Maschinensprache des Prozessors übersetzt.
        - Höhere Programmiersprachen erlauben die effiziente Entwicklung von Software.
        - Diese Programmiersprachen sind möglichst unabhängig vom Prozessor, auf dem das Programm schliesslich laufen soll.
    - Anwenderprogramme
      Alle Programme, die von den Anwendern geschrieben wurden.
        - Beispiele:
            - Spiele
            - Flugreservationssystem
            - Stundenplan
            - Simulator
- Prozess
    - Ein Prozess ist ein Programm, das gerade vom Prozessor (sequentiell) abgearbeitet wird.
    - Heutige Prozessoren können quasi gleichzeitig mehrere Prozesse abarbeiten.
        - Bsp.: Betriebssystem, Textverarbeitung und Browser laufen (scheinbar) gleichzeitig.
        - Der Prozessor schaltet dazu blitzschnell zwischen den Prozessen hin und her und führt dabei jeweils eine kleine Anzahl Befehle jedes Prozesses aus.
        - Für den Benützer scheinen die Programme dann parallel abzulaufen.
        - Die einzelnen Programme laufen umso langsamer, je mehr Prozesse gleichzeitig laufen.
    - Ein Prozess entspricht einem simulierten Computer (siehe Abbildung 2):
        - Er arbeitet nach dem gleichen EVA-Schema wie der Computer.
        - Jeder Prozess hat einen eigenen Arbeitsspeicherbereich mit Daten und Befehlen (Programm).
        - Der Prozessor arbeitet die Befehle des Prozesses in der gewünschten Reihenfolge ab.
        - Der Prozess besitzt ebenfalls Ein-/Ausgabekomponenten, um Daten aus seinem Arbeitsspeicher nach aussen zu geben oder von aussen in den Arbeitsspeicher zu holen.
        - Prozesse können auch gegenseitig Daten austauschen.
        - Die Prozesse merken normalerweise nichts voneinander, ausser wenn sie Daten austauschen. Dann tritt der eine Prozess als Eingabekomponente des anderen auf und/oder umgekehrt.

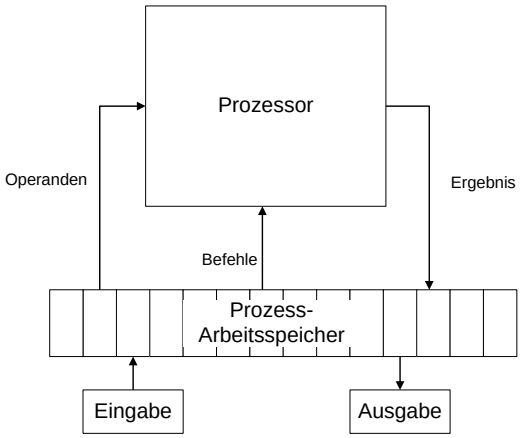

Abbildung 2: EVA-Schema eines Prozesses

## 1.6 Dateien

- Daten, die der Computer erzeugt, werden häufig für die spätere Wiederverwendung auf externen Speichern abgelegt in einer sogenannten Datei.
- Datei (engl. file)
  - ist nichts anderes als sequentiell abgespeicherte binäre Daten (z.B. auf einer Harddisk).
  - Die Interpretation der Daten ist Sache der Programme.
  - Nur das entsprechende Programm kann die Daten in einer Datei richtig interpretieren.
  - Das Lesen (und Schreiben) von Daten von einer Harddisk dauert im Vergleich zum Lesen der Daten aus dem Arbeitsspeicher ca. **1 Million Mal länger**!
- Es existieren verschiedenste Dateien auf einem Computer:
  - Textdateien
  - Programmdateien (Binärdateien)
  - Bilder
  - Sounds
  - Anwendungsspezifische Dateien (Word, Excel, FrameMaker)
- Wie findet Benützer/Computer die Daten wieder?
  - Dateien werden über ihren Namen angesprochen.
  - Der Name einer Datei kann vom Benützer beliebig gewählt und geändert werden.
  - Das Betriebssystem übersetzt den Dateinamen in die entsprechende Zylindernummer, Sektornummer und Lesekopfnummer auf der HD.
- Eine Datei enthält neben den eigentlichen Daten weitere Informationen:
  - Länge der Datei
  - Zugriffsrechte: Lesen, Schreiben, Ausführen
  - Weitere Attribute: Archive, Hidden, Directory
  - Besitzer der Datei
  - Information zur Interpretation der Daten. Diese ist bei Windows in der File-Extension enthalten.
- File-Name-Extension (Dateinamen-Erweiterung):
  - bezeichnet die Zeichenfolge nach dem letzten Punkt (".") im Dateinamen.
  - Sie gibt dem Benutzer und dem Betriebssystem einen Hinweis auf das Programm, mit dem die Datei erstellt worden ist oder gelesen werden kann:
    - .doc Word-Datei
    - .exe Ausführbare Datei (maschinenlesbar)

  - .gif Bitmap-Datei (Bilder)
  - .wav Sound-Datei (Sound = Geräusch, Laut, Ton, Klang)
  - .java Java-Programme (Java-Quellcode)
- Dateisystem (Filesystem)
  - Dateien werden in einem sogenannten Dateisystem (Filesystem) verwaltet
  - Directory oder Folder (Dateiverzeichnis oder Ordner)
    - Mehrere Dateien können in einem Directory zusammengefasst werden.
    - Das Directory selbst ist ebenfalls eine Datei:
      - Es enthält die Information, wo sich die Files dieses Directories auf der HD befinden.
      - Directories können neben normalen Files auch selbst wieder Directories enthalten. Diese heissen Subdirectories (Unterverzeichnisse)
    - Laufwerk (engl. drive)
      - Bezeichnet ein Wurzelverzeichnis auf einem Speichermedium (HD, CDROM, DVD)
      - wird bei Windows mit einem Buchstaben bezeichnet **`A:\, B:\, ..., Z:\`**
  - Das Dateisystem verwaltet die Dateien und Verzeichnisse in einer baumartigen Anordnung siehe Abbildung 3.

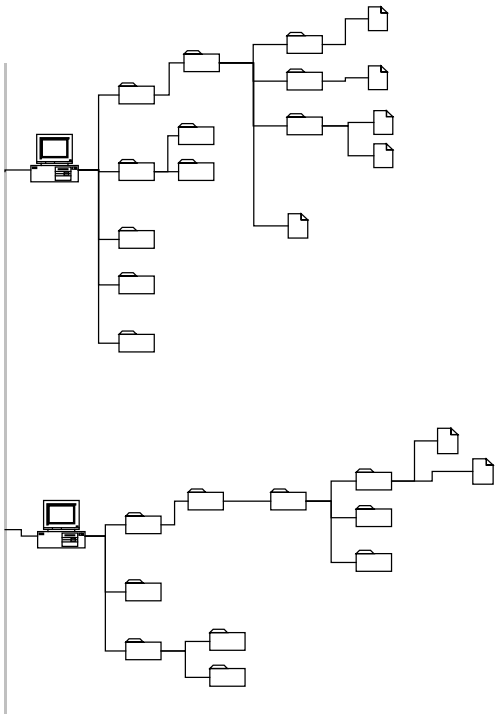

Abbildung 3: Dateisystem

  - Wie spezifiziert man ein File eindeutig:
    - entweder durch die Angabe des Filenamens und des absoluten Pfades
      - ***Laufwerk*`:\`*`dir1`*`\`*`subdir1`*`\`*`subdir2`*`\...\`*`filename`**
      - Bsp.: **`D:\MeineDaten\Finanzen\Steuern.xls`**
    - oder durch die Angabe des relativen Pfades
      - Der relative Pfad bezieht sich auf das aktuelle Verzeichnis (Arbeitsverzeichnis)
      - `.` bezeichnet das aktuelle Verzeichnis
        `..` bezeichnet das nächst übergeordnete Verzeichnis (Parent directory)
      - Beispiel`:`
        Das aktuelle Directory sei`:` **`D:\MeineDaten\Finanzen`**

Mit folgenden Angaben werden die Dateien „Steuern.xls“, Brief1.doc“ und „Adressen.xls“ angesprochen:

```
Steuern.xls
.\Steuern.xls
..\Briefe\Brief1.doc
..\Adressen.xls
```

- Directories auf anderen Rechnern
  - Um Directories auf anderen Rechnern anzusprechen, muss zusätzlich zum lokalen Pfad auf der betreffenden Maschine der Maschinenname selbst angegeben werden.
  - Unter Windows existiert dazu eine universelle Namenskonvention (universal naming convention (UNC)). Dabei wird eine beliebige Datei eindeutig spezifiziert durch die Angabe des Computer gefolgt vom lokalen Pfad auf diesem Computer:

    ***servername**freigabename**dir1**subdir1*\...*filename***
  - Bsp.:
    Die Datei „java.doc“ auf dem Computer „public.zhaw.ch“ im lokalen Verzeichnis „users\staff\huhp “ kann folgendermassen referenziert werden:

    **\\public.zhaw.ch\users\staff\huhp\public\java.doc**
  - Directories auf anderen Rechnern können auch als logisches Laufwerk angebunden und dann angesprochen werden:
    - Bsp.: Das Laufwerk **M:** auf den Praktikums-Rechnern entspricht dem Directory **\\user.zhaw.ch\user$** auf dem Server „mustang“.
- Navigieren im Dateibaum mit der Maus:
  - Doppelklicken auf ein Directory-Symbol öffnet entsprechendes Verzeichnis
  - Mit Windows-Explorer (rechte Maustaste auf **MyComputer**-Icon drücken)
  - Auf der linken Seite wird der Verzeichnisbaum dargestellt.
  - Auf der rechten Seite der Inhalt des ausgewählten Directories.
  - Erzeugen und löschen von Directories
    - Im Directory, in dem ein neues Verzeichnis erzeugt werden soll, rechte Maustaste drücken.
    - Es erscheint eine Auswahl
    - Menupunkt "Neu – Ordner" auswählen
  - Datei verschieben File in neues Directory ziehen:
    **Linke** Maustaste über File-Icon drücken, auf Zielordner hin ziehen, loslassen
  - Datei kopieren:
    - gleich wie bei Datei verschieben, aber mit **rechter** Maustaste
    - dann "Hierher kopieren" auswählen.
- Navigieren von der Kommandozeile aus:
  - **Command Prompt** (bzw. **Eingabeaufforderung**) unter **StartMenu** anwählen.
  - Beachten Sie: Gross-/Kleinschreibung wird unter Windows nicht unterschieden
  - Die zwei wichtigsten Kommandi der Kommandozeile:
    - **cd** (change directory):
      **cd *pfad*** wechselt aktuelles Directory zu Pfad ***pfad***
      (absolut oder relativer Pfad)
      Beispiele:
      **cd d:\MeineDaten** oder **cd ..\Briefe**
    - **dir** (directory):
      **dir *pfad\directoryname*** Inhalt eines Directory anzeigen

**`dir`** ***`pfad\directoryname\filename`*** File anzeigen

Beispiele:
**`dir d:\MeineDaten`** oder **`dir ..\Briefe\Brief1.doc`**

## 1.7 Geschichte des Computers

17. Jh. Erste mechanische Rechenmaschinen mit den 4 Grundrechenarten (Wilhelm Schickard, Blaise Pascal, G. W. Leibnitz)

1805 J.-M. Jacquard erfindet programmierbaren Webstuhl (gelochte Holzplättchen)

1822 Charles Babbage entwirft *Analytical Machine*: Programmierbarer Allzweckrechner, Programm auf Lochkarten, Trennung von Speicher (store) und Rechenwerk (mill). Rechnet im Dezimalsystem. Bereits bedingte Sprünge vorgesehen. Wurde nie gebaut. Nur eine Komponente, die Differenzmaschine, wurde realisiert.

1938 Konrad Zuse (Bauingenieur), Z1: 1. funktionierender programmgesteuerter Rechenautomat. 4 Grundrechenoperationen + Wurzel. Verwendet Dualzahlen. Gleitkommadarstellung. Programm auf 35mm-Film gelocht. Voll mechanisch.

1941 Z3. wie Z1 aber mit elektormechanischen Relais.

1946 J.P. Eckert, J.W. Mauchly: ENIAC(Electronic Numerical Integrator and Computer):
1. voll elektronischer Rechner (18000 Röhren).
John Von Neumann schlägt vor: Programm im Arbeitsspeicher abzulegen.

1949 EDSAC: (Electronic Delay Storage Automatic Computer) erster universeller Digitalrechner mit gespeichertem Programm. Verwendete Ultraschall-Verzögerungsleitungen als Speicher.

Seit 1950 : Industrielle Rechnerproduktion

- Man unterscheidet normalerweise 5 Computer-Generationen:
  - bis Ende 50er:
    Elektronenröhren, Speicher wenige 100 Maschinenwörter, Programmierung in Maschinencode
  - bis Ende 60er:
    Transistoren als Schaltelemente, Ferritkernspeicher (1k), Magnetbänder. Erste Compiler. Programme als Lochkartenstapel.
  - seit Mitte 60er:
    Teilweise integrierte Schaltkreise. Time-Sharing, Ein-/Ausgabe über eine Art Schreibmaschine (Teletypewriter), später Bildschirme mit Tastatur
  - seit Anfang 70er: Mikroprozessoren, 8 Bit, PC
  - seit Anfang 80er: Mehrprozessorsysteme, 16-32-64-Bit-Architekturen, PC-Netzwerke
  - 90er-Jahre:
    Portable Computer und PDAs (Personal Digital Assistant), globale Vernetzung durch Internet

## 1.8 Programmiersprachen

- Parallel zur Entwicklung der Computer wurden unzählige Programmiersprachen entwickelt.
- Programmiersprache
  - ist eine Sprache, um Programme zu schreiben.
  - Wir können damit dem Computer sagen, was er tun soll.
  - Was muss eine Programmiersprache festlegen:
    - Syntax: → Grammatikregeln. Was sind gültige Buchstaben, Wörter, Sätze.
      - "Ich sehen Frau mit das blauen Fahrrad" ist kein gültiger deutscher Satz.

    - Semantik: Was bedeuten die Wörter und Sätze.
        - "Gelbe Ideen schlafen fürchterlich" macht im normalen Sprachgebrauch keinen Sinn.
- Anforderungen an eine Programmiersprache:
    - mächtig: Alles ausdrückbar, was ich dem Computer beibringen will.
    - verständlich: Der Programmierer und Prozessor müssen die Sprache verstehen.
    - eindeutig: Ein bestimmter Satz darf nicht mehrere Bedeutungen haben.
    - kompakt: möglichst wenige Befehle
- Was soll als Programmiersprache verwendet werden?
    - Maschinensprache:
        - war früher die einzige Programmiermöglichkeit
        - ist eindeutig
        - Es kann damit alles beschrieben werden, was der Prozessor kann.
        - Prozessor versteht sie direkt
        - ist schwierig verständlich für Menschen
    - Menschliche Sprache
        - Alles ausdrückbar
        - Menschen verstehen sie gut.
        - Es braucht eine automatische Übersetzung in die Maschinensprache des Prozessors.
        - Problem:
          Die natürliche Sprache ist häufig nicht eindeutig (ein Satz kann verschiedene Bedeutungen haben)
            - Bsp. "Schliessen Sie die Datei mit der Maus".
              Heisst das „Eine Datei mit dem Bild einer Maus“ schliessen, oder eine bestimmte Datei per Mausklick schliessen?
    - Kompromiss:
        - Wir definieren eine eigene künstliche Sprache → Programmiersprache
- Programmiersprachen
    - sind künstliche Sprachen, um mit dem Computer zu kommunizieren.
    - Eine Programmiersprache ist definiert durch:
        - Satz von Symbolen (Buchstaben, Zahlen, Zeichen): Welche Zeichen sind erlaubt?
        - Syntax (Grammatikregeln): Wie ist eine Anweisung aufgebaut?
        - Semantik (Bedeutung): Was bedeutet jede Anweisung?
    - Eine Programmiersprache muss automatisch in die Maschinensprache des Prozessors übersetzt (compiliert) werden können.
    - Programmiersprachen lassen sich, wie Computer, ebenfalls in Generationen einteilen. Es gibt Tausende von Programmiersprachen. Abbildung 4 auf Seite 12 gibt einen kleinen Überblick über die wichtigsten von ihnen.
        - Während die maschinennahen Programmiersprachen (Generation 0 oder 1) mehr oder weniger die Möglichkeiten des Prozessors widerspiegeln verfügen die höheren Programmiersprachen (Generation 2 und höher) über mächtigere Programmierkonstrukte, die das Programmieren einfacher und sicherer machen.
        - Die höheren Programmiersprachen folgen dabei unterschiedlichen Denkmustern (Paradigmen). Beispiele:
            - Prozedurale Programmierung
            - Objektorientierte Programmierung
        - Die verschiedenen Programmiersprachen unterstützen diese Paradigmen mehr oder weniger gut.

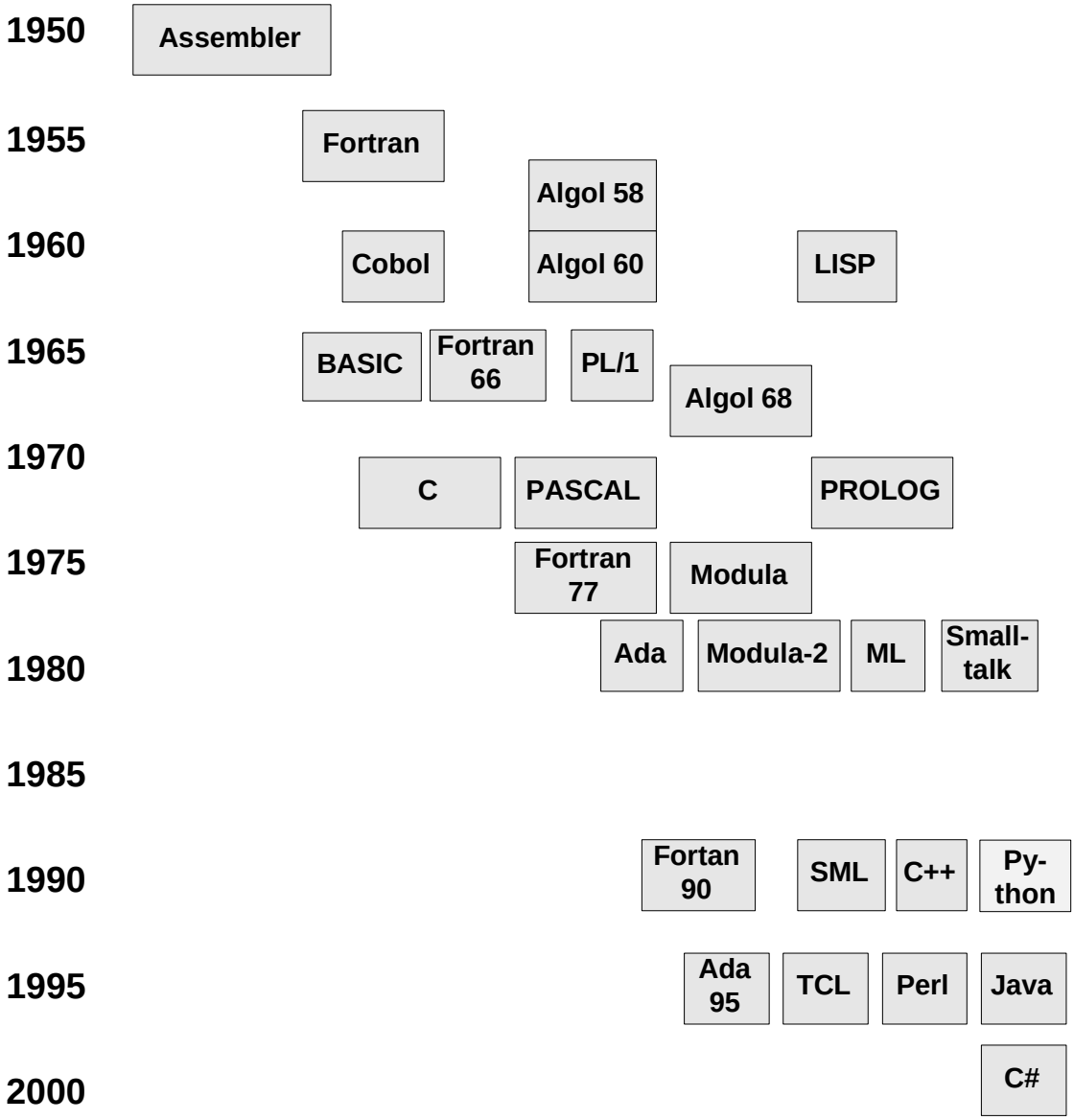


Abbildung 4:Übersicht über die wichtigsten Programmiersprachen