Skript Informatik

Was ist Informatik ?

- Informatik ist die Wissenschaft von der systematischen Verarbeitung von Informationen, insbesondere deren automatisierte Verarbeitung mit Hilfe von elektronischen Rechenanlagen.
- Die Informatik beschäftigt sich mit:
  - dem Aufbau, der Arbeitsweise und dem Konstruktionsprinzip von elektronischen Rechenanlagen und
  - den Strukturen, Eigenschaften und Beschreibungsmöglichkeiten von Informationen und deren Weiterverarbeitung.

Algorithmus

- Ein Algorithmus ist eine endliche Folge genau definierter Vorschriften zur Lösung einer Aufgabenstellung.
- Ein Algorithmus ist ein Verfahren, welches in einem endlichen Text niedergelegt werden muss und effektiv durch eine Maschine (Computer) ausgeführt werden kann.
- Algorithmen wandeln Eingabedaten in Ausgabedaten um.

Programm

- Programme sind in einer Programmiersprache formulierte Algorithmen.
- Im Gegensatz zu Algorithmen sind Programme in einer eindeutigen Programmiersprache verfasst.
- Ein und derselbe Algorithmus kann in verschiedenen Programmiersprachen formuliert werden.

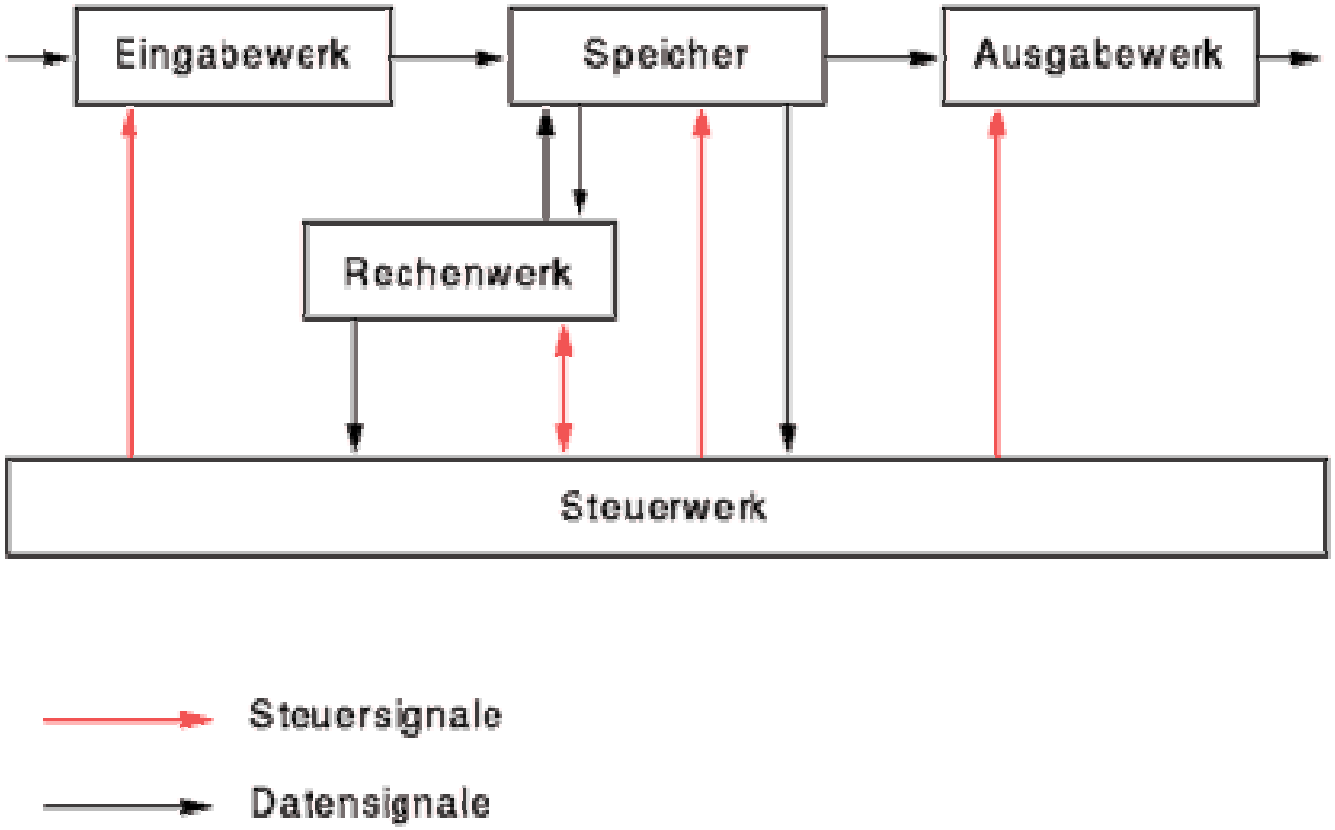
Softwareentwicklung

- Unter Softwareentwicklung versteht man die systematische Konstruktion von Programmen zur Lösung eines in der realen Welt gestellten Problems.

Computer

- Von wenigen Ausnahmen abgesehen, liegt modernen Computern das Prinzip des Von-Neumann-Rechners zugrunde.
- Der Rechner besteht aus fünf Funktionseinheiten:
  - Steuerwerk
  - Rechenwerk
  - Speicher
  - Eingabewerk
  - Ausgabewerk
- Diese Funktionseinheiten sind durch ein Bussystem verbunden.
- Von-Neumann-Rechner folgen dem EVA-Prinzip: Eingabe - Verarbeitung - Ausgabe

Von-Neumann-Rechner



Zentraleinheit

- Steuerwerk, Rechenwerk und Speicher fasst man auch unter der Bezeichnung „Zentraleinheit" zusammen.
- Steuerwerk und Rechenwerk sind meist in einem Chip, dem Prozessor (CPU = Central Processing Unit) untergebracht.
- Der Prozessor ist mit den anderen Bestandteilen des Rechners über einen Bus verbunden.

Steuerwerk

- Das Steuerwerk steuert und koordiniert alle Aktionen, die in der Zentraleinheit ablaufen.
- Das Steuerwerk hat folgende Aufgaben:
  - Befehle in der richtigen Reihenfolge aus dem Speicher laden
  - Befehle decodieren
  - Befehle interpretieren
  - Versorgung der Funktionseinheiten mit Steuersignalen

Rechenwerk

- Im Rechenwerk werden die arithmetischen und logischen Verknüpfungen ausgeführt (ALU: Arithmetic Logical Unit).
- Die Operanden werden hierbei vom Steuerwerk geliefert.
- Das Rechenwerk wird mit den für die richtige Durchführung der Operation notwendigen Steuersignalen versorgt.

Speicher

- Im Speicher werden:
  - Programme
  - Daten
  - Zwischenergebnisse
  - Endergebnisse

  abgelegt
- Jede Speicherzelle hat eine Adresse.

Bus

- Ein Rechner ist modular aufgebaut.
- Größere Funktionseinheiten sind meist in Form von einzelnen Chips realisiert.
- Neben den oben erwähnten Funktionseinheiten gibt es u.a. Chips für die Bildschirmsteuerung, die externe Kommunikation, die Kontrolle von Peripheriegeräten.

Bus

- Die notwendigen Verbindungen zwischen den einzelnen Elementen werden von sogenannten Bussen übernommen.
- Als Beispiele sind der Datenbus und der Adressbus zu nennen.
- Der Datenbus transportiert Daten.
- Über den Adressbus wird festgelegt, welche Speicherzelle gelesen oder beschrieben werden soll.

Eingabe- und Ausgabewerk

- Geräte, die an dem Ein- und Ausgabewerk angeschlossen sind, werden als Peripherie bezeichnet.
- Beispiele für die Peripherie sind: Tastatur, Maus, Bildschirm, Drucker, Scanner, DVD-Laufwerk.

- Keine Peripheriegeräte hingegen sind: Konventionelle Festplatten, SSD, USB-Sticks. Sie zählen zu den Datenspeichern.

Programmiersprachen

- Man unterscheidet 5 Generationen von Programmiersprachen:
  - 1. Generation: Maschinensprache
  - 2. Generation: Assemblersprache
  - 3. Generation: Höhere Programmiersprache
  - 4. Generation: Nichtprozedurale Sprache
  - 5. Generation: Wissens- und Objektorientierte Sprache

Maschinensprache

- Die Maschinensprache entspricht dem Befehlsvorrat einer speziellen CPU.
- In Maschinensprache geschriebene Programme sind nur auf einem CPU-Typ lauffähig.

Maschinensprache Beispiel

0010	0010	0011	1100
0001	0110	0001	0101
0000	0000	0000	0101
0010	0000	0011	1100
0110	0100	0000	1110
0000	0000	0000	0011
1101	0000	1000	0001

Assemblersprache

- Der Vorrat an symbolischen Befehlen entspricht dem der Maschinensprache.
- Einem Maschinenbefehl ist jeweils ein symbolischer Befehl zugeordnet.
- In Assembler geschriebene Programme sind nur auf einem CPU-Typ lauffähig.

Assemblersprache

- Mit Assembler-Programmen kann man, unter Ausnutzung aller Möglichkeiten der CPU, sehr effiziente Programme erstellen.
- Assemblerprogramme werden mit speziellen Übersetzern in Maschinensprache übersetzt.

```
MOVE.L #222222, D0
MOVE.L #333333, D1
ADD.L D1, D0
```

Höhere Programmiersprachen

- Für diese Sprachen gibt es eine Vielzahl von Bezeichnungen.
- Sie sind weitgehend maschinenunabhängig, genormt und oft für einen speziellen Anwendungsbereich konzipiert.
- Zur Problemlösung werden Algorithmen formuliert, d.h. die Abfolge, in der die Daten bearbeitet werden, wird definiert.

Höhere Programmiersprachen

- Hier trennte sich auch die Entwicklung zwischen Sprachen für kaufmännische Anwendungen, welche meist große Datenbestände verwalten müssen und wissenschaftlich-technische Anwendungen, bei denen die effiziente Programmierung komplexer mathematischer Algorithmen im Vordergrund steht.

Höhere Programmiersprachen

- Als Urahn der kaufmännisch orientierten Programmiersprachen ist COBOL (Common Business Oriented Language, ab 1959) zu sehen; die älteste Programmiersprache für wissenschaftlich-technische Anwendungen ist FORTRAN (FORmula TRANslator, ab 1954).
- Weitere Beispiele sind PL/1, PASCAL, BASIC, ALGOL und C.

```
a = 222222;
b = 333333;
c = a + b;
```

Nichtprozedurale Sprachen

- Die Definition dieser Sprachen und der im nächsten Unterkapitel beschriebenen Sprachen ist fließend.
- Eine wichtige Eigenschaft ist die Nichtprozeduralität.
- Dem Rechner muss nicht mehr länger mitgeteilt werden, wie er das Problem zu lösen hat, sondern nur noch, was zu geschehen hat.

Nichtprozedurale Sprachen

- Verwendung finden sie überwiegend im Bereich von Dateiverwaltungs- und Datenbanksystemen.
- Deshalb werden diese Sprachen auch als datenorientierte Sprachen bezeichnet.
- Am bekanntesten ist SQL für die relationale Datenmanipulation.

Nichtprozedurale Sprachen

- Die Sprachen der 4. Generation haben die Produktivität in der Programmentwicklung erheblich erhöht.
- Allerdings nehmen sie sehr stark Hardwareressourcen in Anspruch.
- Viele dieser Sprachen sind proprietär, d.h. sie gehören zu einer bestimmten Rechnerplattform, und damit ist man vom jeweiligen Hersteller abhängig.

Objektorientierte und Wissensbasierte Sprachen

- Schwierig ist die Klassifikation dieser Sprachen.
- Sie werden teilweise auch als 5. Generation bezeichnet.
- Zu bedenken ist jedoch, dass es sich hierbei um die verschiedensten Arten von Programmiersprachen handelt.

Objektorientierte und Wissensbasierte Sprachen

- Objektorientierte Sprachen beruhen auf einem vollkommen anderen Programmierprinzip.
- Die Daten, die ein Programm verarbeitet und die Methoden, mit denen die Daten verarbeitet werden, werden zu sogenannten Objekten zusammengefasst, die dann miteinander kommunizieren.
- Eine wichtige Eigenschaft ist die Möglichkeit der Vererbung von Eigenschaften eines Objektes auf ein anderes.
- Beispiele sind C++ und Java.

Objektorientierte und Wissensbasierte Sprachen

- Wissensbasierte Sprachen, auch KI-Sprachen (KI = Künstliche Intelligenz) sind Sprachen zur Entwicklung von Expertensystemen.
- Beispiele sind LISP und PROLOG.

Übersetzer

- Ein grundlegendes Problem aller Programmiersprachen ist die Übersetzung des sogenannten Quellprogramms in Maschinensprache.
- Hierbei werden drei Konzepte unterschieden:
  - Assembler
  - Compiler
  - Interpreter

Assembler

- Assembler setzen Befehle eins zu eins aus der maschinenorientierten Sprache in die Maschinensprache um.

Compiler

- Compiler übersetzen ein in einer höheren Sprache geschriebenes Programm in Maschinensprache.
- Dabei wird die Quelldatei auf syntaktische Fehler geprüft.
- Nach erfolgreicher Übersetzung und dem Link-Vorgang kann das Programm gestartet werden.
- Bei jeder Änderung muss das gesamte Programm neu übersetzt werden.

Interpreter

- Interpreter gehen Zeile für Zeile durch den Quelltext.
- Eine Zeile wird hierbei auf syntaktische Korrektheit geprüft und dann sofort ausgeführt.
- Danach wird die nächste Zeile bearbeitet.
- Im Gegensatz zum Compiler erstellt der Interpreter kein eigenes, lauffähiges Programm.
- Von daher muss z.B. bei der Weitergabe immer sichergestellt sein, dass der Empfänger auch über den Interpreter verfügt.

Entwicklungsprozess für C++Programme

- Der **Editor** unterstützt den Programmierer bei der Eingabe der Quelldatei z.B. *Beispiel.cpp*.
- Mit dem **Präprozessor** wird die Quelldatei für den Übersetzungsvorgang vorbereitet.
- Der **Compiler** übersetzt die Quelldatei in Maschinensprache. Als Resultat des erfolgreichen Übersetzungsvorgangs erhält man das Objektprogramm *Beispiel.obj*.

Entwicklungsprozess für C++Programme

- Der **Linker** fügt dem Objektprogramm andere Objektprogramme, z.B. aus Programm-Bibliotheken hinzu. Es resultiert ein ausführliches Maschinenprogramm *Beispiel.exe*.
- Der **Lader** transferiert das ausführbare Maschinenprogramm in den Hauptspeicher und startet es.
- Der **Debugger** wird zur Analyse des Quellprogramms und für Tests eingesetzt.

C++ Historie

- 1966 Martin Richards entwickelte **BCPL**, assemblernahe Sprache
- 1967 **Simula 67**, Programmiersprache für mathematisch technische Probleme und Simulation, erste objektorientierte Konzepte
- 1970 Ken Thompson entwickelte **B**, eine assemblernahe Sprache; Unix Betriebssystem für PDP 7
- 1972 Dennis Ritchie (Bell Labs) entwirft **C**; rechnerunabhängige Sprache zur Portierung von Unix auf PDP 11

C++ Historie

- 1978 Kernighan und Ritchie geben das erste Referenzbuch zu **C** heraus
- 1979 Start der Arbeiten an **C** mit Klassen
- 1983 erster Entwurf von **C++** durch Bjarne Stroustrup; Bell-Labs in den USA
- 1985 Bjarne Stroustrup veröffentlicht das Buch: The **C++** Programming Language

C++ Historie

- 1987 Veröffentlichung des ersten reinen **C++** Compilers für PCs und Workstations
- 1988 American National Standards Institute (kurz: ANSI) veröffentlicht den ersten **C** Standard
- 1990 erste **C++** Version von Borland (heute: Inprise)
- 1992 erste **C++** Version von Microsoft
- 1998 Veröffentlichung des ANSI **C++** Standards ISO/IEC 14882