

Grundlagen der Informatik

Gliederung der Vorlesung

- **Einführung (1)**
- **Grundlagen der Programmierung**
 - a. Algorithmenstrukturen (2)
 - b. Datenstrukturen (3)
- 3. Programmierung mit C**
 - a. Einführung (4)
 - b. Elementare Datentypen (5)
 - c. Kontrollstrukturen (6)
 - d. Komplexe Datenstrukturen (6)
 - e. Funktionen (8)
 - f. Arbeit mit Dateien
- 4. Datenbanken**
 - a. Einführung (10)
 - b. Relationale Datenbanksprache SQL (11)
- 5. Datenkommunikation**
 - a. Einführung (12)
 - b. Sprachen für die Datenkommunikation (13)
- 6. Softwaretechnologie (14)**

Einführung

Aspekte der Informatik

In welcher Form tritt uns die Informatik gegenüber?

- | | |
|---|----------------------------------|
| - gibt Spaß und Freiheit im Privatleben | - gibt Einfluß und Macht |
| - ist nackte Notwendigkeit im Berufsleben | - vernichtet Arbeitsplätze |
| - reorganisiert ganze Arbeitsgebiete | - schafft neue Betätigungsfelder |
| - führt zu einer globalen Vernetzung | - verändert soziale Beziehungen |

Ergebnis: globale Umwälzungen in:

Wissenschaft, Privatsphäre, Industrie, Gesellschaft

Triebkräfte der Entwicklung:

militärischer, wissenschaftlicher Bereich ➡ kommerzieller, privater Bereich

Grundbegriffe

Informatik:

Wissenschafts- und Technologiedisziplin, die sich mit Methoden und Verfahren der automatisierten Verarbeitung von Informationen befaßt.

→ Computer Science

Information:

elementare Kategorie wie Stoff und Energie

→ neues Wissen über einen Sachverhalt

→ Beseitigung von Ungewißheit

- qualitativer Aspekt:
 - Syntax (Struktur, Form)
 - Semantik (Bedeutung, Inhalt)
 - Pragmatik (Wert, Praxis)
- quantitativer Aspekt:
 - Bit (elementare Informationseinheit, entspricht einer Ja-Nein-Entscheidung)
 - Byte (8 Bit, 1 Zeichen), Wort, ...

verwandte Begriffe: Daten, Wissen

Verarbeitung: Eingabe, Speicherung, Umformung, Transport, Ausgabe

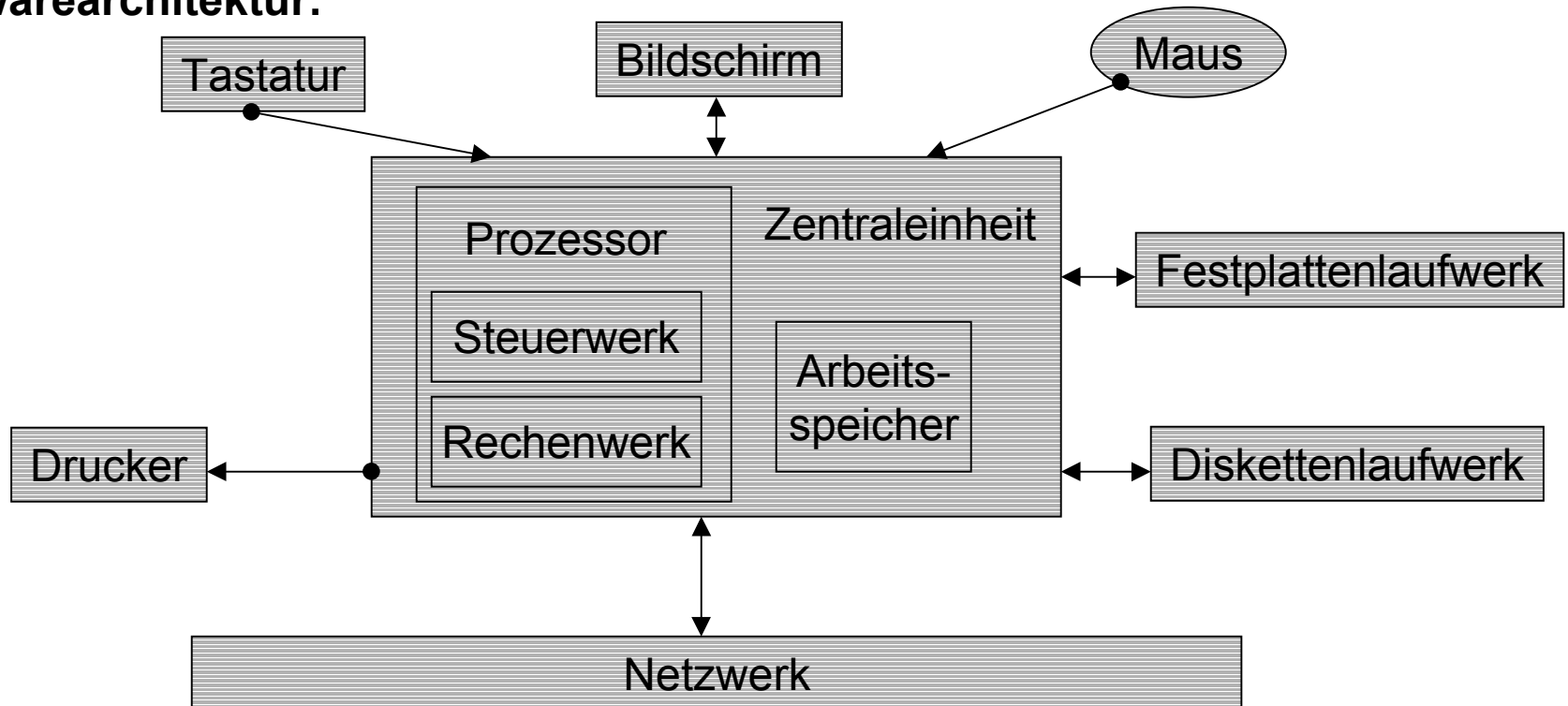
Automatisierung: durch Computer und Programme

Hardware

Gesamtheit aller materiellen, physischen, technischen Geräte zur Verarbeitung von Informationen (Computer, Bildschirm, Tastatur, Maus, Drucker, Scanner, Plotter, Mikrophon, Lautsprecher)

Computer: univerelles technisches Arbeitsmittel für die Verarbeitung von Informationen

Hardwarearchitektur:



Organisationsprinzipien:

- Taktgesteuerte automatische Arbeit
- Adressierter Speicher für binär codierte Daten und Programme
- Programme bestehen aus Befehlen über Daten
- Sequentielle Abarbeitung von Befehlen, veränderbar durch Bedingungen und Sprünge

Heutige Computerklassen:

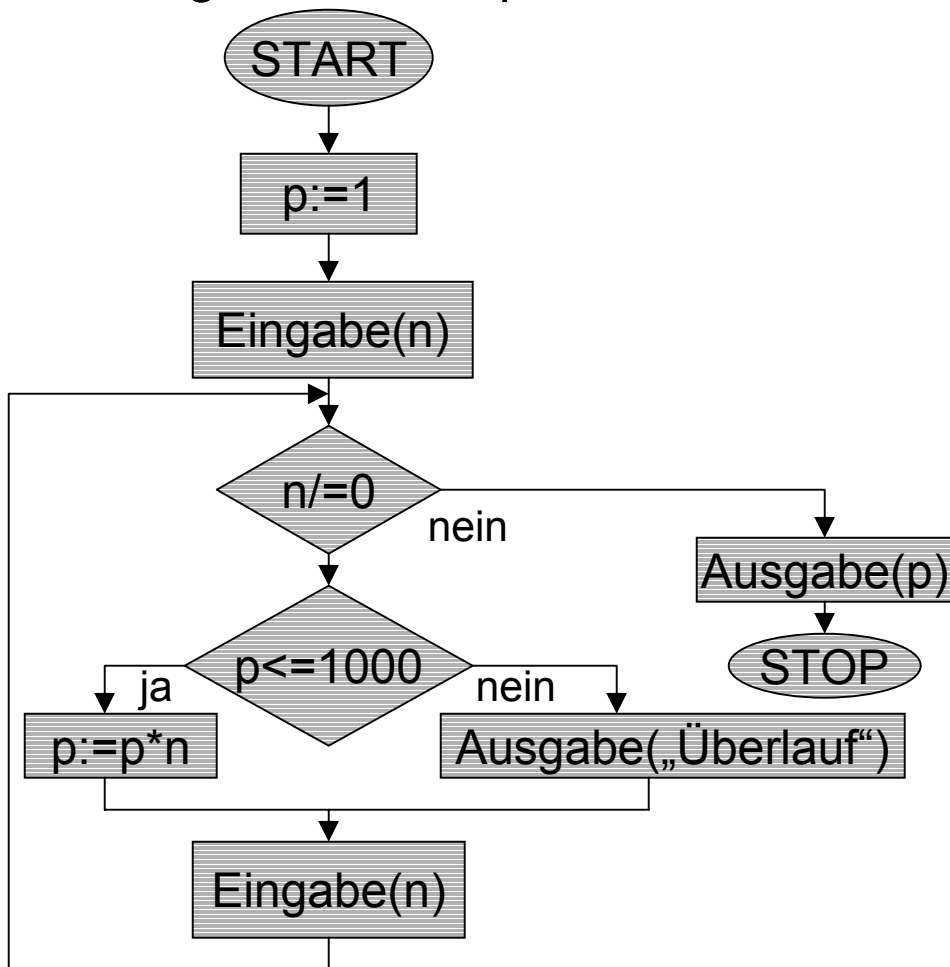
- - Personalcomputer (PC):
weit verbreiteter Einzelplatzrechner für Privatpersonen, Verwaltungen, ...
- - Workstation (WS):
professionelle vernetzte Arbeitsplätze für Programmierer, Entwickler, Manager, ...
- - Großcomputer (Mainframe):
leistungsstarke Zentralrechner für rechenzentren, Großbetriebe, ...
- - Supercomputer:
hochleistungsfähige rechner mit Spezialarchitekturen für komplizierte, zeitaufwendige, wissenschaftlich-technische Berechnungen
- - Einbaucomputer:
spezielle kleine Rechner zur direkten Einbettung in andere technische Systeme

Anwendungsbeispiele

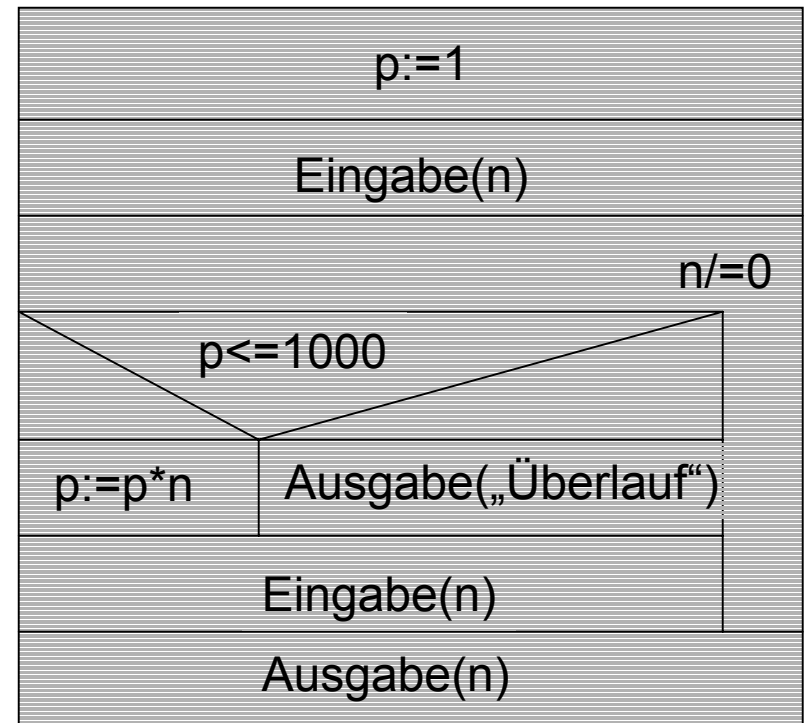
Produktbildung:

Eine Folge von Zahlen soll eingegeben und miteinander multipliziert werden. Wird eine Null eingegeben, wird das Produkt der bisherigen Zahlen ausgegeben. Ist das Produkt größer 1000, erfolgt statt der Multiplikation eine Fehlerausschrift.

Programmablaufplan:



Struktogramm:



Programmiersprache:

dient der Beschreibung von Strukturen über Daten und Anweisungen und damit als Bindeglied zwischen Mensch und Computer

Klassen von Programmiersprachen:

- - Maschinorientierte Sprachen
Maschinencodes (1GL), Assemblersprachen (2GL)
- - Problemorientierte Sprachen:
prozedurale (3G), bildschirmorientierte (4GL), deklative (5GL) Sprachen
funktionale, objektorientierte Sprachen

Computersystem:

= Hardware + Software

Hardware speichert Daten und führt Anweisungen aus, die in Software codiert sind

Firmware:

fest in Schaltkreisen realisierte Daten und Programme

- zur Erweiterung des Funktionsumfangs der Hardware
- zur Effizienzsteigerung der Software

Historische Entwicklung

Natürliche Vorgeschichte:

- Informationsverarbeitung in Lebewesen: Genetik, Neurologie, Psychologie, Sprache
- Informationsverarbeitung in sozialen Strukturen: Gruppe, Herse, Familie, Staat

Kulturelle Vorgeschichte:

- Informationsdarstellung und Codierung
 - 4000 v.u.Z.: älteste Zahlzeichen der Welt
 - 300 v.u.Z.: Griechenland: Axiome, Sätze, Regeln, Algorithmen
 - 700: Dezimalsystem in indien
 - 1600: Dualsystem in Deutschland
- Erleichterung der Rechenarbeit
 - 1500: Rechenbücher, Tafelwerke
- Mechanische Rechenmaschinen
 - 1100 v.u.Z.: Abakus in Ostasien
 - 1600: Rechner für Grundrechenarten
- Automaten und Steuerungen
 - 1700: Uhren, Musikautomaten
 - 1800: Reglungen, Lochkarten
- Analogrechner
 - mechanisch: Fliehkraftregler, Differentialgetriebe
 - elektrisch: wissenschaftlich-technische Rechner

Elektronische Digitalrechner:

- 
- technische Voraussetzungen
 - 1906: Elektronenröhre
 - 1947: Transistor
 - 1961: Integrierter Schaltkreis
 - theoretische Voraussetzungen
 - 1833: Prinzip des programmgesteuerten Rechners (Babbage)
 - 1855: Boolesche Algebra (Boole)
 - 1931: Entscheidbarkeit (Gödel)
 - 1936: Turing-Maschine (Turing)
 - 1944: Von-Neumann-Computer (von Neumann)
 - erste Realisierungen
 - 1941: Relais ZUSE Z3 (Zuse)
 - 1944: Relais MARK I (Aiken)
 - 1951: Elektronenröhren ENIAC (Eckert)
 - 1955: Transistoren TRADIC (Bell. Lab.)
 - 1965: integrierte Schaltkreise IBM/360 (IBM)

Computergenerationen:



0. Generation:

1941; Relais;

Maschinencode, militärischer Einsatz

1. Generation:

1951; Elektronenröhre; Magnettrommel; Lochkarte; Schreibmaschine; Assemblersprache; wissenschaftlich-technischer Einsatz

2. Generation:

1960; Transistor; Ferritkern; Magnetband; Lochkarte; Drucker; problemorientierte Sprache, einfaches Betriebssystem; beginnender kommerzieller Einsatz

3. Generation:

1965; integrierter Schaltkreis (SSI); Mikroferritkern; Wechselplatte; Bildschirm; Datenfernübertragung; universelle Sprache; komplexes Betriebssystem; kommerzieller Einsatz

4. Generation:

1975; höher integrierter Schaltkreis (LSI); Spektrum an peripheren Geräten; komfortable Basis- und Anwendungssoftware; Vernetzung; komfortable Nutzeroberfläche; bildschirmorientierte Sprache; privater Einsatz

5. Generation:

1985; höchstintegrierter Schaltkreis (VLSI); natürliche, deklarative Sprache; Multimedia; Wissensverarbeitung; Künstliche Intelligenz; weltweite Vernetzung; Informationsdienste

Teilgebiete der Informatik

- Theoretische Informatik:

Modellierung und Untersuchung grundlegender informationeller Strukturen und Prozesse mit mathematischen Mitteln (Automaten-, formale Sprach-, Informations-, Algorithmentheorie, Logik, Algebra, ...)

- Technische Informatik:

Struktureller und funktioneller Aufbau von Rechnersystemen (Schaltkreise, Prozessoren, Computer, Netze, periphere Geräte, ...)

- Praktische Informatik:

Mittel, Methoden und Werkzeuge zum Entwurf und zur Umsetzung von Problemlösungen auf dem Computer (Betriebssysteme, Compiler, Tools, Programmierungs-, Softwaretechnik, ...)

- Angewandte Informatik:

Aufbereitung, Modellierung und Umsetzung von computergestützten Lösungen für spezifische Anwendungsgebiete (Wirtschaft, Medizin, Bau, Kommunikation, Büro, ...)

Tangierende Gebiete:

Kybernetik, Neurologie, Psychologie, Soziologie, Mathematik, Physik, Biologie

Wichtige Gebiete der Praktischen und angewandten Informatik:

Datenbanken, Künstliche Intelligenz, Multimedia, Informations- und Steuerungssysteme, Modellierung und Simulation, Mensch-Maschine-Schnittstellen

Grundlagen der Programmierung

Algorithmenstrukturen

Ziel der Arbeit mit dem Computer: **Lösung von Problemen**

Problemlösung dargestellt durch

- **Anfangszustand** / Eingaben
- Menge von **Operationen**
- **Endzustand** / Ausgaben

Problemklasse: Menge ähnlicher Probleme

Algorithmierung:

Entwicklung eines **allgemeinen Lösungsplanes** für eine Klasse von Problemstellungen, der festlegt, wie man durch **Ausführung von Operationen** von einem gegebenen **Anfangszustand zu einem Endzustand** gelangt.

Algorithmus

Präzise, eindeutige Beschreibung eines allgemeinen Verfahrens, das unter Verwendung von endlich vielen, effektiv ausführbaren, elementaren Arbeitsschritten (Operationen) zur Lösung einer Klasse von Problemen automatisch abgearbeitet werden kann.

Prozess: Abarbeitung / Ausführung eines Algorithmus.

Prozessor: Einrichtung, die einen Algorithmus ausführt.

Eigenschaften von Algorithmen:

- Statische Finitheit (endliche Beschreibung)
- Dynamische Finitheit (endliche Ressourcen für Ausführung)
- Terminierung (Ausführung endet nach endlich vielen Schritten)
- Determiniertheit (zu jeder Aktion gibt es höchstens eine Folgeaktion)

Theorie: Berechenbarkeit, Turing-Maschine, Automaten, Rekursive Funktionen, Komplexitätstheorie.

Beispiele: Summierung von Zahlen, Bestimmung des ggT, Sortieren von Objekten.

Datenstrukturen, Operationen, Kontrollstrukturen

Elementare Datentypen (Objekte, auf denen Operationen ausgeführt werden):

- Konstanten zur Darstellung von Zahlen: 0; 1; 345; -124; 213412342;
- Benannte Konstante: $\pi = 3,14$;
- Variablen zur Aufnahme (Instanzierung) mit Konstanten; x; y; ggT;

Elementare Operationen (für die folgende Darstellung vorausgesetzt):

- Eingabe von Daten: Eingabe(x), Eingabe(X,Y);
- Arithmetischer Ausdruck: $2 - 4$; $1/a$; $5 * (x + v)$;
- Logischer Ausdruck (Bedingung): $3 < 5$; $5 < 3$; $x > 5$; $a == b$; $c != d$;
- Zuweisung von Daten: $x = 2$; $y = z$; $v = 3 + 5$;
- Ausgabe von Daten: Ausgabe(x); Ausgabe(3); Ausgabe(x, 4, 8)

Anmerkung: im folgenden benutzen wir immer **V** für Verarbeitung!

Elementare Kontrollstrukturen


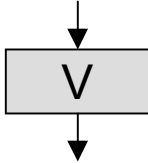

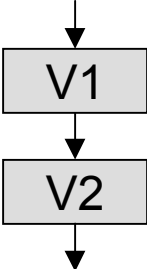
- Folge / Sequenz
- Alternative / Auswahl / Selektion
- Wiederholung / Iteration
- Block / Unterprogramm

Darstellung von Kontrollstrukturen

Struktogramm: block-orientierte Darstellung

Programmablaufplan: graph-orientierte Darstellung

Pseudocode: verbale, „halbformale“ Darstellung

Kontrollstruktur	Struktogramm	Programmablaufplan	Pseudocode
Keine			Verarbeitung V
Folge			Erst Verarbeitung V1 dann Verarbeitung V2

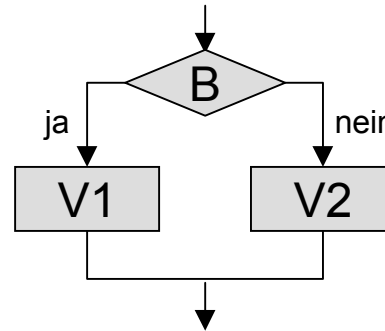
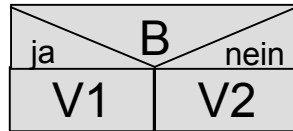
Kontrollstruktur

Struktogramm

Programmablaufplan

Pseudocode

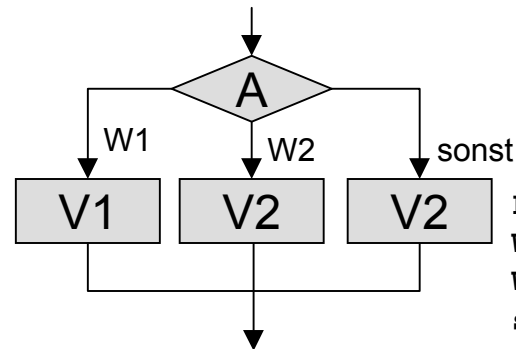
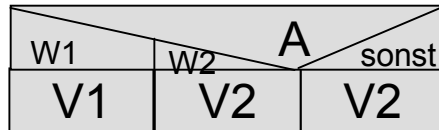
Alternative



Falls Bedingung B:

- wahr: Verarbeitung V1
- falsch: Verarbeitung V2

Mehrfache Alternative



Falls Ausdruck A ergibt:

- W1: Verarbeitung V1
- W2: Verarbeitung V2
- sonst: Verarbeitung V3

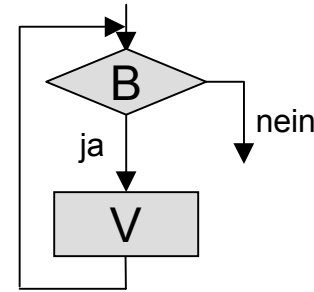
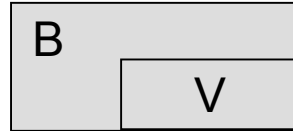
Kontrollstruktur

Struktogramm

Programmablaufplan

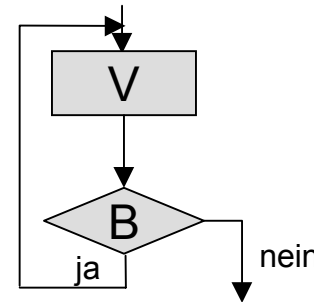
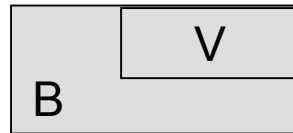
Pseudocode

**Wiederholung
(abweisend)**



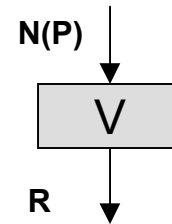
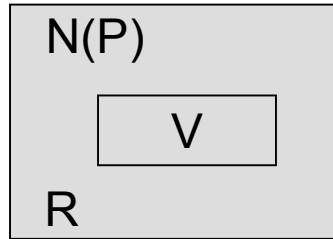
Solange Bedingung B
wiederhole Verarbeitung V

**Wiederholung
(nicht abweisend)**



Wiederhole Verarbeitung V
solange Bedingung B

Definition eines Blocks



Definition Block N mit formalen Parametern P, Rumpf V und Rückgabewert R

Aufruf eines Blocks: $N(P)$; oder $X = N(P)$;

Als Verarbeitung $N(P)$ mit oder ohne Zuweisung. Ersetzen der formalen Parameter P durch aktuelle Parameter; Abarbeiten des Rumpfes V (Parameter P können verwendet oder verändert werden). Rückgabe des Ergebnisses R .

Zusatz (nur Programmablaufplan):

START

Beginn Verarbeitung

STOP

Ende Verarbeitung

M

Verbindung von / zu M

Anwendungsbeispiele

Produktbildung: Eine Folge von Zahlen soll eingegeben und miteinander multipliziert werden. Wird eine Null eingegeben, wird das Produkt der bisherigen Zahlen ausgegeben. Ist das Produkt größer 1000, erfolgt statt der Multiplikation eine Fehlerausgabe.

Struktogramm

Programmablaufplan

Pseudocode

Fakultätsfunktion: Multiplikation aller natürlichen Zahlen von 1 bis n .

Struktogramm

Programmablaufplan

Pseudocode

Fakultätsfunktion: Multiplikation aller natürlichen Zahlen von 1 bis n .

Struktogramm

Programmablaufplan

Pseudocode

Grundlagen der Programmierung

Datenstrukturen

Algorithmen bestehen aus Operationen über Daten/Datenstrukturen

Datenstrukturen:

- sind Operanden in Operationen
- werden gelesen oder verändert
- sind konstant oder variabel
- besitzen elementare oder komplexe Struktur

Datentyp:

Zusammenfassung/Verallgemeinerung ähnlicher Datenstrukturen,
definiert durch einen Wertebereich und die darauf festgelegten Operationen

Konstante:

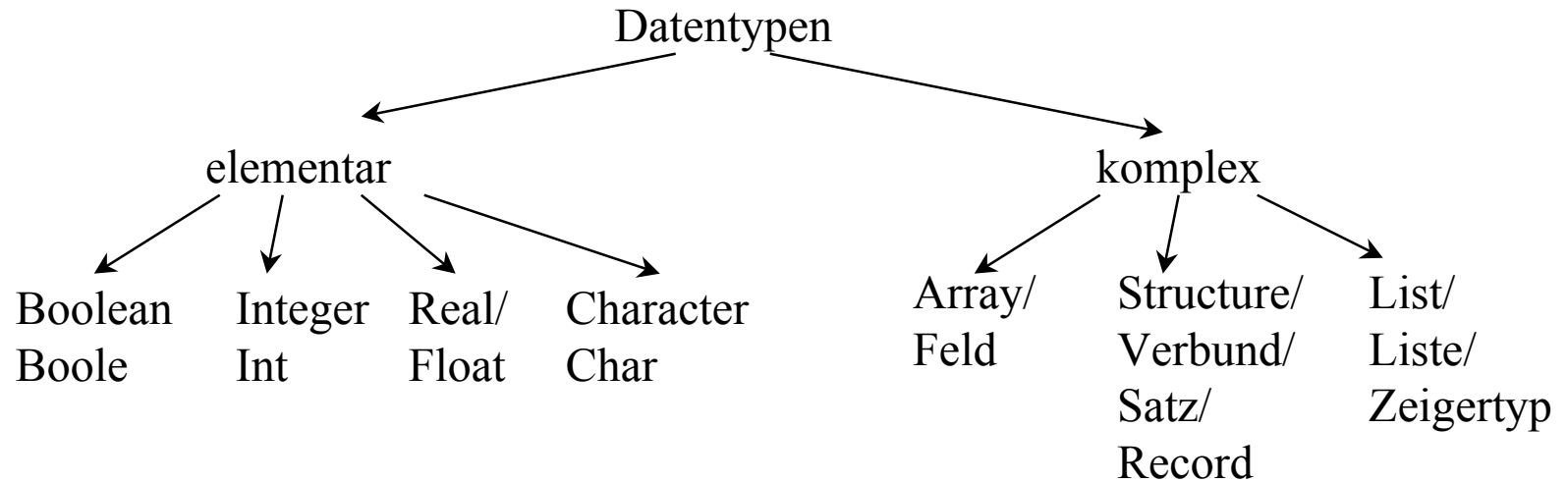
Datenstruktur eines bestimmten Datentyps mit festem, zeitlich unveränderlichem Wert

Variable:

Datenstruktur, die bei der Ausführung eines Algorithmus unterschiedliche Werte im Rahmen eines Datentyps annehmen kann

Folie 2

Datentypen



abstrakter Datentyp (ADT):

Schema zur Realisierung eines Datentyps mit

- festgelegten externen Datenstrukturen und Operationen,
- verborgenen/gekapselten internen Datenstrukturen und Operationen

Folie 3

Elementare Datentypen

Integer: ganze Zahlen

Strukturen: 1; 33; 1422; -35

Operationen: **integer** x;

+; -; *; /; <; >; =; !=

Real: reelle Zahlen

Strukturen: 1.2; 55.3333; -4.4

Operationen: **real** x;

+; -; *; /; <; >; =; !=

wurzel, potenz, logarithmus

Charakter: Zeichen

Strukturen: 'a'; 'A'; 'x'; '+'

Operationen: **charakter** x;

uppercase; lowercase

Boolean: Wahrheitswerte

Strukturen: 1; 0;

Operationen: **boolean** x;

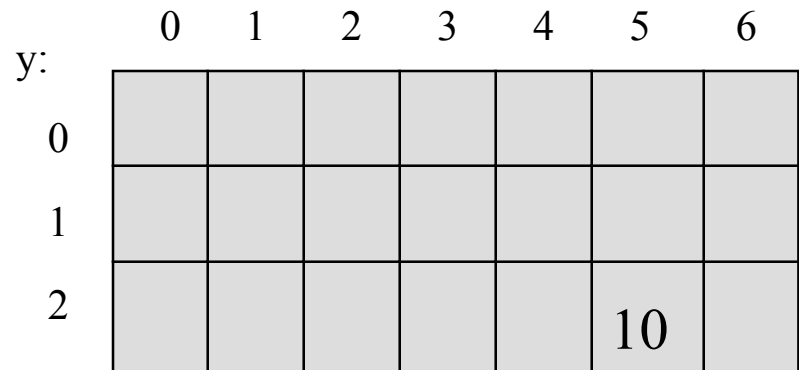
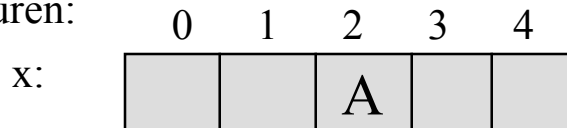
~; ∨; ∧; -->; =; !=

Folie 4

Array (Feld)

Aneinanderreihung von Elementen bekannter Anzahl und gleichen Datentyps mit indexiertem Zugriff über die Position

Strukturen:



Operationen:

Typdefinition: `string1 = array character [5]; matrix1 = array integer [3,7];`

Datendefinition: `string1 x; matrix1 y;`

Zugriff: `a:=x[2]; b:=y[1,6];`

Änderung: `x:= { 'B', 'L', 'A', 'U' }; x[U]:='G'; x[1]:='R'; y[2,5]:=10`

Beispiele:

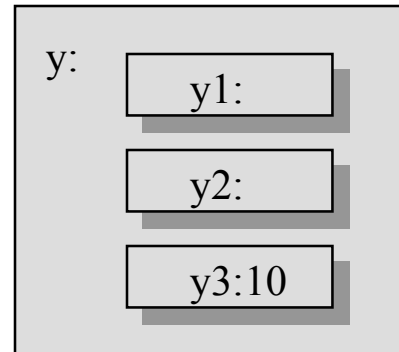
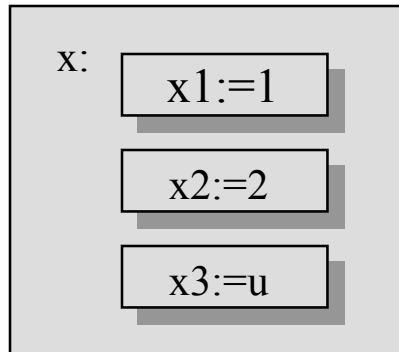
- Änderung der Reihenfolge von Elementen
- Aufbau einer Matrix mit dem kleinen 1x1

Folie 5

Structure (Verbund)

Zusammenfassung einer Anzahl benannter Elemente gleichen oder unterschiedlichen Datentyps mit direktem Zugriff über den Namen

Strukturen:



Operationen:

Typdefinition: `structure1 = structure {integer x1, integer x2, character x3};`
 `structure2 = structure {boolean y1, character y2, integer y3};`

Datendefinition: `structure1 x; structure2 y;`

Zugriff: `a:= x.x2; b:= y.y2;`

Änderung: `x:= {1,2,'u'}; x.x3:= 'U'; y.y3:= 10;`

Beispiele:

- Speicherung einer Adresse mit Name, Vorname, Postleitzahl, Ort, Straße, Nummer
- Verarbeitung von Büchern mit ISBN, Autor, Titel, Jahr, Deskriptor

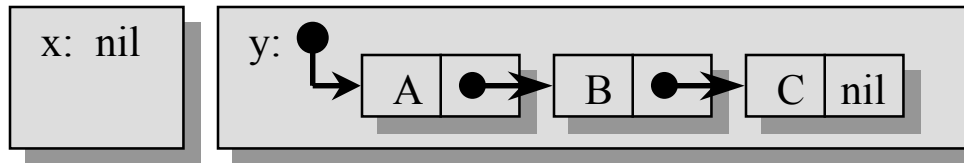
Folie 6

List (Liste)

Verkettete variabel lange Folge von Elementen gleichen Datentyps mit Zugriff über Zeiger

(Realisierung als abstrakter Datentyp)

Strukturen:



Operationen:

Typdefinition: `intlist = list integer; charlist = list charakter`

Datendefinition: `intlist x; charlist y`

Zugriff: `a:= first (y); b:= rest (y);`

Änderung: `x:= nil; x:= {1,2,3,4}; y:= nil;`

`y:= cons ('C', y); y:= cons ('B', y); y:= cons ('A', y); first (y):= 'D';`

Beispiele:

- Verkettung eingegebener Zahlen und Ermittlung deren Anzahl
- Aufbau eines Kellerspeichers

Kombination von Datentypen

Komplexe Datentypen werden aufgebaut aus:

- elementaren Datentypen und/oder
- komplexen Datentypen

Typdefinition:

Feld von Strukturen:	s1 = structure {integer s11, character s12}; a1 = array s1 [100];
Struktur von Feldern:	a2 = array integer [20]; a3 = array character [10]; s2 = structure {a2 s21, a3 s22};
Liste von Feldern:	l1 = list a3; l2 = list a1;

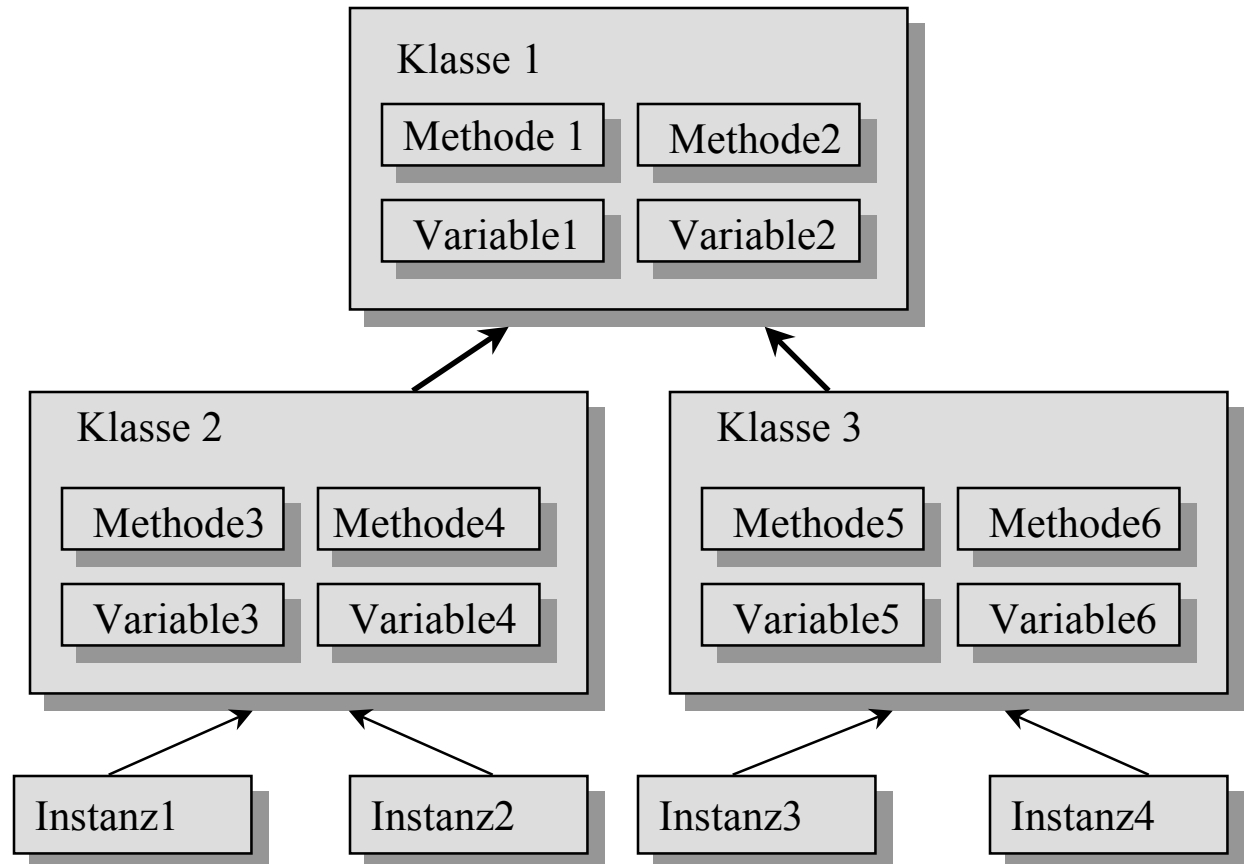
Datendefinition und Zugriff:

Feld von Strukturen:	a1 x; x[33].s12;
Struktur von Feldern:	s2 y; y.a2[14];
Liste von Feldern:	l1 z; first(z)[4];

Objektklassen

Schema zur Beschreibung von Objektinstanzen ähnlichen Verhaltens mit

- festgelegten Datenstrukturen/Variablen und Operationen/Methoden,
- Nachrichtenaustausch zwischen Objekten,
- Vererbung durch Einordnung in eine Klassenhierarchie



Beispiele:

- Lebewesen
- Fahrzeuge

- Programme
- Datentypen

Folie 1

Programmierung mit C Einführung

Programm:

Darstellung eines Algorithmus und seiner Datenstrukturen in einer dem Computer verständlichen Form

Programmiersprache:

legt Syntax und Semantik von Programmen fest,
ist Bindeglied zwischen Computer und Mensch

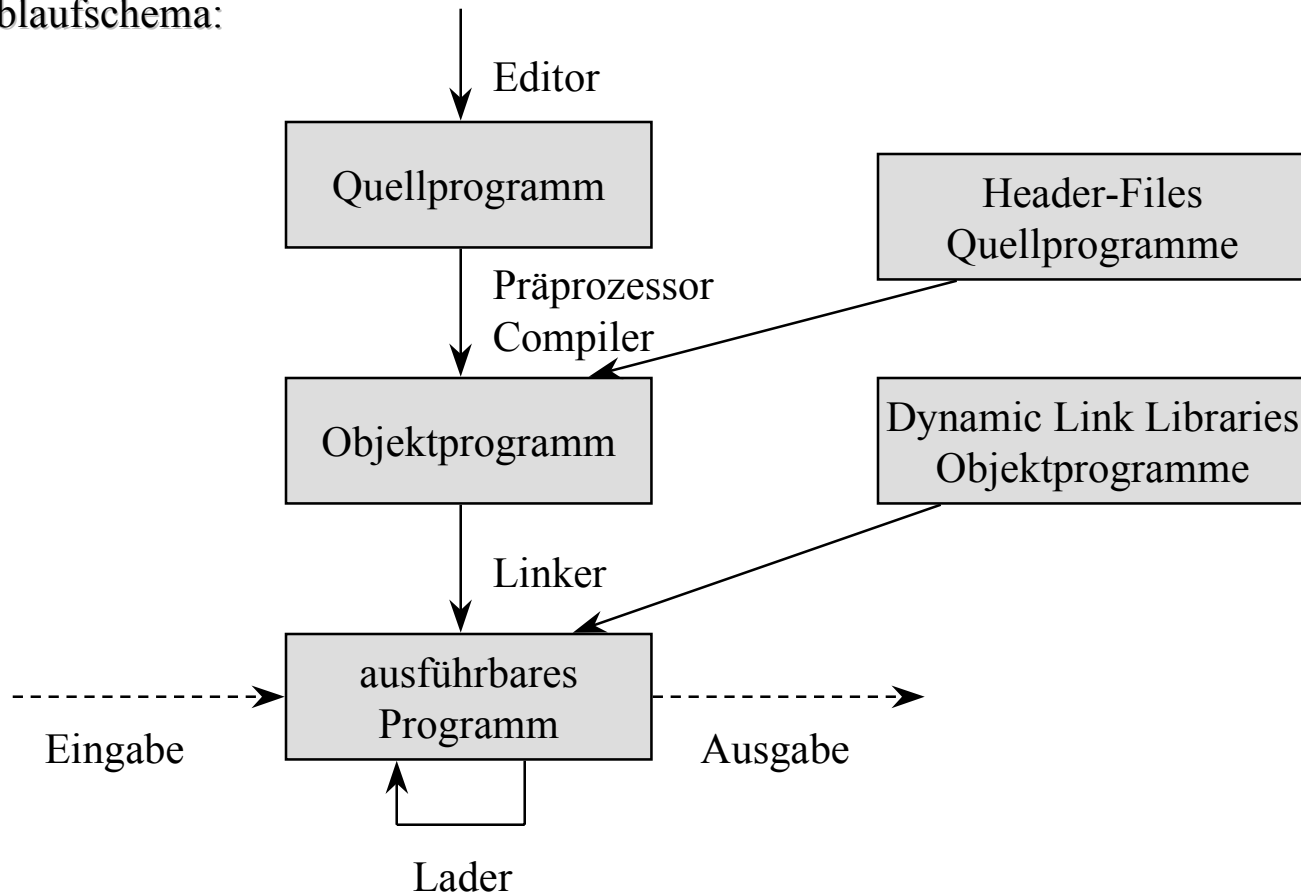
Eigenschaften von C:

- prozedurale Sprache
- entwickelt 1972 als Systemprogrammiersprache für Betriebssystem UNIX
- relativ hardwarenahe Sprache
- standardisierte Sprache (ANSI, 1988)
- gegenwärtig für alle Hardware- und Betriebssystemplattformen verfügbar

Folie 2

Programmentwicklung

Ablaufschema:



Folie 3

Das erste Programm

C-Programm:

```
#include <stdio.h>
void main ( )
{
    printf ("hello, world\n");
}
```

Ausgabe:



Folie 4

Prinzipieller Programmaufbau

```
#include <header-file1>
#include <header file2>

...
void main (void)
{Anweisung1;
  Anweisung2;
  ...}
```

Präprozessor-Anweisungen:
Hauptprogramm-Definition:
Hauptprogramm-Block:
Anweisungen:
Kommentare:

```
#include <header-file>
void main (void)
{...}
Anweisung;
// Text
/* Text */
```

Prinzipien:

- Vorkommen von main nur einmal pro Programm
- Abschluß jeder Anweisung mit Semikolon
- Unterscheidung zwischen Groß- und Kleinbuchstaben
- keine Formatierung
- Deklarationen vor erster Verwendung

Folie 5

Ein- und Ausgabe

Formatierte Ein- und Ausgabe:

printf: `printf (formatstring, dat1, dat2, ...)`
scanf: `scanf (formatstring, dat1, dat2, ...)`
formatstring:
“%c - char, %s - string, %d - int, %f - float”

Zeichen Ein- und Ausgabe:

getchar: `zeichen = getchar ()`
putchar: `putchar (zeichen)`

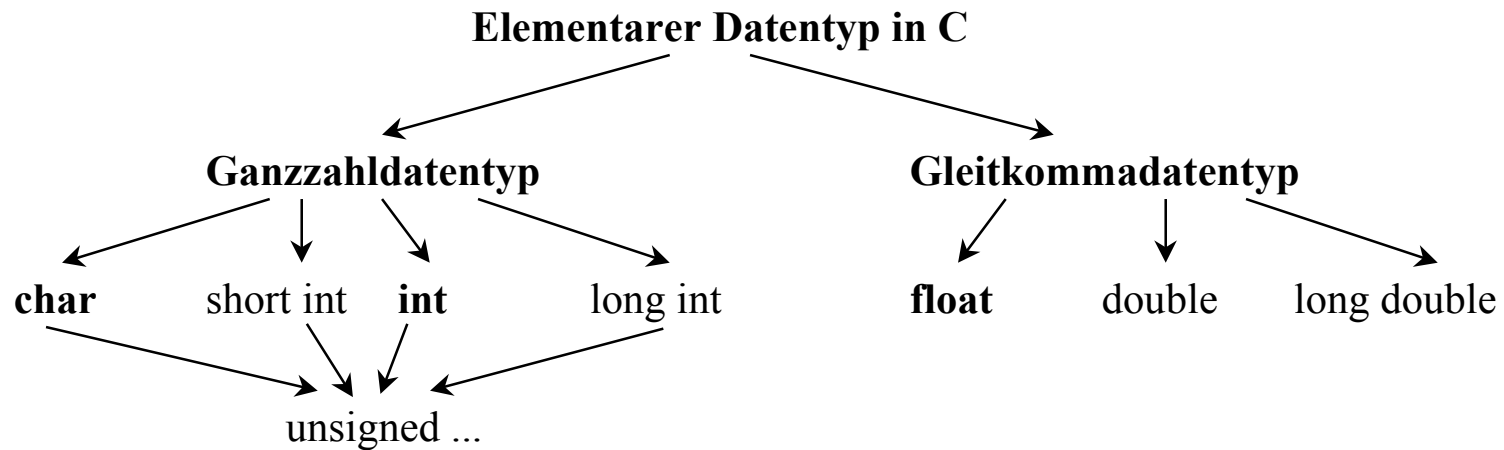
Weitere Ein- und Ausgabefunktionen:

`gets, puts, getline, read`

Beispiel:

```
#include <stdio.h>
#include <iostream.h>
void main (void)
{int zahl, ergebnis;
 printf (“Eingabe Zahl:”);
 scanf (“%d”, &zahl);
 ergebnis = zahl * zahl;
 printf
 (“Das Quadrat von %d ist gleich %d.\n\n”
 zahl, ergebnis);}
```

Elementare Datentypen



char: Typ: Zeichen, Länge: 8 Bit, Minimum: -128, Maximum: 127
Beispiele: 'a', 'W', '*'

int: Typ: ganze Zahl, Länge: 32 Bit, Minimum: -2147483648, Maximum: 2147483647
Beispiele: 0, 1, 1433, -345

float: Typ: Gleitkommazahl, Länge: 32 Bit, Minimum: +/-3.4E-38, Maximum: +/-3.4E38
Genauigkeit: 6 Ziffern
Beispiele: 1.222, -3.4E-23, 8.77777E2

Folie 5.2

Deklarationen in C

Variablen:

Typ Name;	bei mehreren Variablen gleichen Typs:
Typ Name = Wert;	Typ Name1, Name2, Name3 = Wert3, Name4 = Wert4, ...;

Beispiel: int zahl; char Buchstabe; float laenge;
int zahl1 = 7; char Buchstabe1 = 'q'; float laenge2 = 2.3, laenge3 = 0.1E3;

Konstanten:

```
const Typ Name = Wert;
```

Beispiele: const int radius = 7; const char ypsilon = 'y'; float pi = 3.14159;

Prinzipien:

- Gültigkeit in einem Block: {int a, b; ...} {...; float a, c; ...}
- Wert einer Konstanten kann nicht verändert werden
- Namen:
 - bestehend aus alphanummerischen Zeichen und “_”
 - max. 32 Zeichen
 - darf nicht mit Ziffer, sollte nicht mit “_” beginnen
 - keine Schlüsselwörter verwenden (int, void, if, while, ...)

Folie 5.3

Operatoren und Ausdrücke in C

Operatoren:

Sonderzeichen, die festlegen, welche Operationen mit den Operanden (Daten) auszuführen sind

Ausdrücke:

Verknüpfung von Operanden durch mehrere Operatoren

Eigenschaften von Operatoren:

Bedeutung:	Auszuführende Operation (Berechnung, Zuweisung, Zugriff, Aufruf, ...)
Stelligkeit:	Anzahl der beteiligten Operanden (1, 2, ...)
Priorität:	Rang mit Stufen 1 für höchste Priorität bis 16 für niedrigste Priorität
Assoziativität:	Abarbeitungsreihenfolge von links (>) oder von rechts (<)

Bei gleicher Priorität unterschiedlicher Operatoren: Abarbeitung von links nach rechts,
außer bei Priorität 2 und 15

Explizite Festlegung der Abarbeitungsreihenfolge durch Klammerung (...) möglich

Beispiele: $a = b + c * d;$

$a = (b + 3) * 4$

$h = a / b + ((c - d) / (e + f)) * g;$

Folie 5.4

Arithmetische Operatoren in C

Operator	Bedeutung	Stelligkeit	Priorität	Assoziativität
+	Addition	2	5	>
-	Subtraktion	2	5	>
*	Multiplikation	2	4	>
/	Division	2	4	>
%	Modulo	2	4	>
+	unäres Plus	1	2	<
-	unäres Minus	1	2	<

Anmerkungen: - implizite Datentypumwandlung zu mächtigeren Datentyp
 - explizite Datentypumwandlung durch **Typ (Ausdruck)**

Beispiele: `int a = 6, b = 2, c;`
`float d = 2.5, e, f;`
`e = a - b * d;`
`c = int (e);`
`f = 3 / 4 * 5.0`
`f = 3.0 / 4 * 5.0`

Folie 5.5

Vergleichsoperatoren in C

Operator	Bedeutung	Stelligkeit	Priorität	Assoziativität
<	kleiner	2	7	>
<=	kleiner oder gleich	2	7	>
>	größer	2	7	>
>=	größer oder gleich	2	7	>
==	gleich	2	8	>
!=	ungleich	2	8	>

Anmerkungen:

- kein gesonderter Datentyp boolean
- falsch: 0
- wahr: jeder Integerwert ungleich 0

Beispiele: `int a = 2, b = 3, c, d, e;`
`c = a < b;`
`d = a > b;`
`e = a == b;`

Folie 5.6

Logische Operatoren in C

Operator	Bedeutung	Stelligkeit	Priorität	Assoziativität
&&	Konjunktion (Und)	2	12	>
	Disjunktion (Oder)	2	13	>
!	Negation (Nicht)	1	2	<
&	bitweise Konjunktion	2	9	>
	bitweise Disjunktion	2	11	>
^	bitweise exklus. Disj.	2	10	>
~	bitweise Negation	1	2	<
<<	bitweise Linksschieben	2	6	>
>>	bitweise Rechtsschieben	2	6	>

Anmerkungen: Jede Datenstruktur hat eine interne binäre Verschlüsselung

Beispiele: `int a = 0, b = 2, c = 3, d, e, f, g;`
`d = a&&!b;`
`e = a || b || c;`
`f = c << 2;`
`g = c & 5;`

Folie 5.7

Zuweisungsoperatoren in C

Operator	Bedeutung	Stelligkeit	Priorität	Assoziativität
=	Zuweisung	2	15	<
x=	Zuweisung nach Operation x	2	15	<
++	Inkrementierung	1	1, 2	>, <
--	Dekrementierung	1	1, 2	>, <

Anmerkungen: $x \in \{+, -, *, /, \%, \ll, \gg, \&, |, \wedge\}$

Beispiele: `int a = 3, b = 3, c = 3, d = 1;`
`d = d + 1;`
`d += 1;`
`d ++;`
`a += b * = c << = 2;`

Folie 5.8

Sonstige Operatoren in C

Operator	Bedeutung	Stelligkeit	Priorität	Assoziativität
()	Funktionsaufruf	2	1	>
[]	Arrayelement	2	1	>
.	Strukturelement	2	1	>
,	Aufzählung	2	16	>
&	Adresse	1	2	<
*	Inhalt	1	2	<

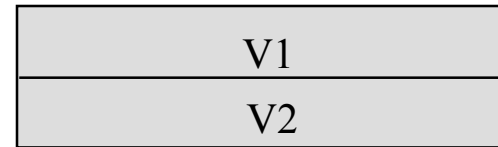
Anmerkungen: Weitere Operatoren: -->, .* , -->* , :, ::, (type), sizeof(type)

Beispiele: `c = sqrt(a)`, `feld [a]`, `struktur.a`, `&a`, `*a`

Folie 6.1

Kontrollstrukturen in C

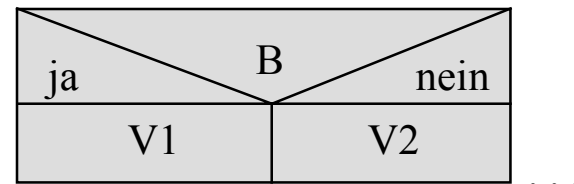
Normaler Programmablauf:
Abarbeitung von **Folgen** von Anweisungen



Änderung des Programmablaufes:
durch Kontrollstrukturen (Steueranweisungen)

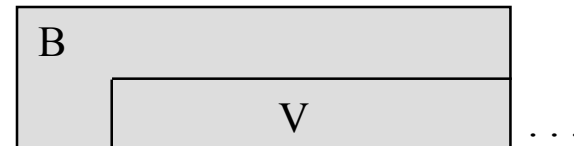
- **Alternativen**

- if
- switch



- **Wiederholungen**

- for
- while
- do



- **Sprünge**

- break
- continue
- goto
- return

unstrukturierte Programmierung
möglich

Folie 6.2

Grundlegende Anweisungen

Grundlegende Anweisungen:

- | | |
|---------------------------|---|
| - Leere Anweisung: | keine Aktion
<i>Beispiele:</i> ; |
| - Ausdrucksanweisung: | Auswertung des Ausdrucks mit eventuellen Seiteneffekten
<i>Beispiele:</i> int i; i = 1; j < 2; 3; |
| - Markierte Anweisung: | Benennung einer Anweisung als Sprungziel
Marke: Anweisung
<i>Beispiele:</i> Fehler:printf("Fehler"); Add: a = a + 4; |
| - Verbundanweisung/Block: | Zusammenfassung mehrerer Anweisungen zu einer Anweisung durch Klammerung {}
Bildung eines Deklarationsabschnitts
<i>Beispiele:</i> {int a; a = 2; printf("%d", a);} |



Steueranweisungen bauen auf grundlegenden Anweisungen auf

Folie 6.3

if - Anweisung

Syntax:

```
if (Ausdruck) Anweisung1  
if (Ausdruck) Anweisung1 else Anweisung2
```

Semantik:

Wenn Ausdruck ungleich 0, Ausführung von Anweisung1,
sonst Ausführung von Anweisung2

Bemerkungen:

- Abschluß der Anweisung durch “ ; “
- Verbundanweisungen und Verschachtelungen möglich

Beispiele:

```
if (a==2) b = 2;  
if (a!=0) b = b / a; else printf("Fehler: Division durch 0!");
```

```
if (a>-1&&a<1)  
    {b = a; printf("%d", b);} else  
    {b = 1/a; printf("%d", b);}
```

```
if (a<0)  
    {if (a%2) printf("negativ, ungerade"); else printf("negativ, gerade")} else  
    {if (a%2) printf("positiv, ungerade"); else printf("positiv, gerade")}
```

Folie 6.4

switch - Anweisung

Syntax:

```
switch (Ausdruck) Anweisung
```

Semantik:

In Abhängigkeit der Auswertung des Ausdruckes: Verzweigung zu der Marke im Anweisungsteil, die den Wert bezeichnet, und Ausführung der folgenden Anweisungen

Bemerkungen:

- typisch: Verbundanweisungen mit markierten Einzelanweisungen
- Marken: **case Wert:**, **default:**
- Abarbeitungsunterbrechung mit **break**

Beispiele:

```
int a; char b;  
switch (a%4)  
{case 1: b = 'a'; break;  
  case 2: b = 'b'; break;  
  case 3: b = 'c'; break;  
  default: b = ''; printf("unerlaubte Zahl");}  
switch (a)  
{case 1: printf("Das");  
  case 2: printf("ist");  
  case 3: printf("ja");  
  default: printf("Spitze");}
```

Folie 6.5

for - Anweisung

Syntax:

```
for (Ausdruck1; Ausdruck2; Ausdruck3) Anweisung
```

Semantik:

Ausführung von Ausdruck1, dann:
solange Ausdruck2 wahr ist, Ausführung von Ausdruck3 und Anweisung

Bemerkungen:

- typisch: Ausdruck1 - Initialisierung Laufvariable
Ausdruck2 - Schleifenbedingung Laufvariable
Ausdruck3 - Veränderung Laufvariable
- Verbundanweisungen und Verschachtelungen möglich

Beispiele:

```
for (int n = 10; n <=100; n = n + 3) {printf("%d", n);}
```

```
int n, i, prod;  
for (n = 1; n < 11; n++)  
  {for (i = 1; i <11; i++)  
    {prod = n * i;  
     printf("%d mal %d ist gleich %d \n", n, i, prod)}}}
```

Folie 6.6

while-/ do - Anweisung

Syntax:

```
while (Ausdruck) Anweisung  
do Anweisung while (Ausdruck);
```

Semantik:

Ausführung der Anweisung, solange Ausdruck wahr ist
(abweisende bzw. nichtabweisende Schleife)

Bemerkungen:

- Ausdruck ist Schleifenbedingung
- Verbundanweisungen und Verschachtelungen möglich

Beispiele:

```
int sum = 0; n = 0; while (n<=100) {sum += n; n ++;} printf("%d", sum);  
int n = 10; do {printf("%d", n*n*n); n --;} while (n !=0);
```

```
int anz, n = 1;  
float zahl = 0, sum = 0, mittelw;  
printf("Anzahl: "); scanf("%d", &anz);  
while (n <= anz)  
    {printf("Zahl: "); scanf("%f", &zahl); sum = sum + zahl; n++;}  
if (anz !=0)  
    {mittelw = sum/anz; printf("Mittelwert:%f", mittelw);}
```

Folie 6.7

break-, continue-, goto-, return - Anweisung

break-Anweisung: Beenden einer Wiederholung oder mehrfachen Alternative
break;

continue-Anweisung: Beenden des aktuellen Schleifendurchlaufes einer Wiederholung und Fortsetzen mit dem nächsten Schleifendurchlauf
continue;

goto-Anweisung: Sprung zu einer Marke im Rahmen einer Funktion
goto Marke;

return-Anweisung: Verlassen einer Funktion und Rückgabe eines Wertes
return (Wert);

```
→ ... ;  
while (Ausdruck)  
{ ... ;  
  continue;  
  ... ;  
  break;  
  ... ; }  
→ ... ;
```

```
→ { ... ;  
  Marke1: Anweisung1;  
  ... ;  
  goto Marke2;  
  ... ;  
  goto Marke1;  
  ... ;  
  Marke2: Anweisung2;  
  ... ; }
```

Folie 7.1

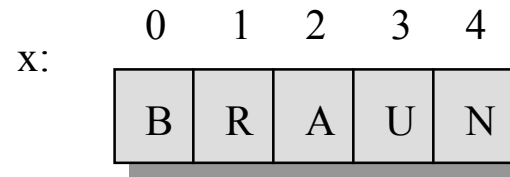
Komplexe Datenstrukturen

Komplexe Datenstrukturen setzen sich aus elementaren Datenstrukturen oder wiederum komplexen Datenstrukturen zusammen.

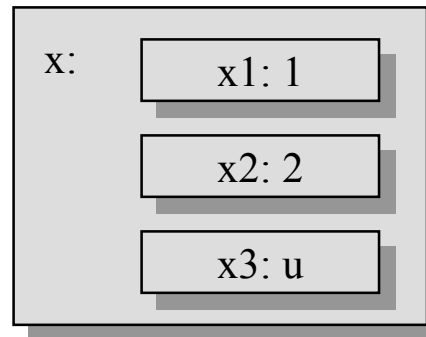
Datenstrukturen sind **Datentypen** zugeordnet.

Vorgehensweise: Typdeklaration --> Variablen- oder Konstantendefinition

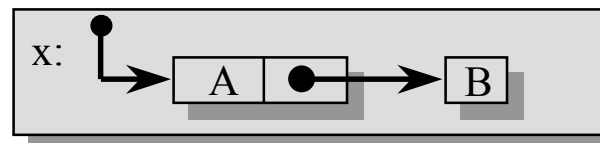
- **Felder:**



- **Strukturen:**



- **Zeiger:**



Folie 7.2

Leerer und Aufzählungsdatentyp

bisher: Elementare Datentypen: char, int, float

Leerer Datentyp: enthält keine Datenstrukturen und Operationen
Anwendung: Definition von Funktionen und Zeigern

void

Aufzählungsdatentyp: Menge benannter Integerkonstanten
implizite Wertzuordnung: 0 als Startwert, dann aufsteigend
explizite Wertzuordnung: setzen Startwert, dann aufsteigend

```
enum Typname {Komponentenliste};  
enum Typname Name;
```

Beispiele:

```
enum farben {rot, gelb, gruen, blau};  
enum farben a; a = gruen; printf("%d", a + 1);  
enum Wochentage {Montag = 1, Dienstag, Mittwoch};  
enum Wochentage b;
```

Folie 7.3

Feld

Syntax:

```
Typ Name [Dimension];  
Typ Name [Dimension1][Dimension2] ...;
```

Semantik: Definition eines Feldes mit Angabe von Typ, Name und Dimensionen

Zugriff: auf Element über Index

```
Name [Index]
```

Bemerkungen:

- Verbindung von Typdeklaration und Variablendefinition
- Dimension ist positive ganze Zahl
- Initialisierung und Verschachtelung möglich

Beispiele:

```
int zahlen1 [9]; zahlen1 [0] = 3;  
int zahlen2 [4] = {11, 22, 33, 44}; int zahl = zahlen2 [0] + zahlen2 [2];  
char wort1 [5] = {'G', 'r', 'u', 'e', 'n'}; printf("%s", wort1 [1]);  
char wort2 [] = "Blau"; wort2 [0] = 'G'; wort2 [1] = 'r'; printf("%s", wort2);  
int wert [4] [3] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,}; int i, j;  
for (i = 0; i < 4; i++)  
    {for (j = 0; j < 3; j++) printf("%d", wert [i] [j] );  
    printf("/n");}
```

Folie 7.4

Zeiger

Syntax:

```
Typ * Name;
```

Semantik:

Definition eines Zeigers mit Angabe eines Namens auf eine Datenstruktur eines bestimmten Typs

Zugriff:

auf den Inhalt (Dereferenzierung)
auf die Datenstruktur selbst
auf die Adresse (Referenzierung)

```
* Name  
Name  
& Name
```

Bemerkungen:

- Verbindung von Typdeklaration und Variablendefinition
- Zeiger sind ganze Zahlen; arithmetische Operationen erlaubt
- Typen müssen übereinstimmen (Ausnahme: Typ void)
- Initialisierung und Verschachtelung möglich

Beispiel:

```
int wert1 = 7, zahl = 8; float wert2 = 22.2;  
int *zeig1 = &wert1; float *zeig2 = & wert2;  
zeig1 = & zahl;  
wert1 = * zeig1;  
char wert3 [13] = "Zeichenkette", char *zeig3 = & wert3 [0];  
char * zeig4 = "Zeichenkette";  
zahl = 0; while (* zeig4++) zahl ++; printf("%d", zahl);
```

Folie 7.5

Struktur

Syntax:

```
struct Typname {Komponentenliste};  
struct Typname Name;  
union Typname {Komponentenliste};  
union Typname Name;
```

Semantik:

Definition einer Struktur mit Angabe von Typname, Name und Komponenten

Zugriff:

auf Komponente über Name

```
Name Kompname
```

Bemerkungen:

- Komponenten besitzen Typ
- union stellt Platz nur für eine aktuelle Komponente bereit
- Initialisierung und Verschachtelung möglich

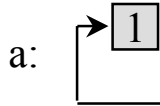
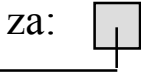
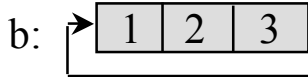
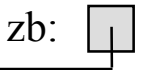
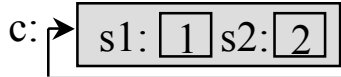

Beispiel:

```
struct geburt {int tag, monat; char name [80];};  
struct geburt g1; g1.tag = 12; strcpy (g1.name, "Anton");  
struct geburt g2 = {13, 11, "Anna"};
```

```
struct qualifikation {char * schulbildung; char * beruf};  
struct qualifikation personal [100];  
personal[24].schulbildung = "Abitur";  
personal[25].beruf = "Leerer";  
personal[25].beruf[2] = 'h';
```

Folie 7.6

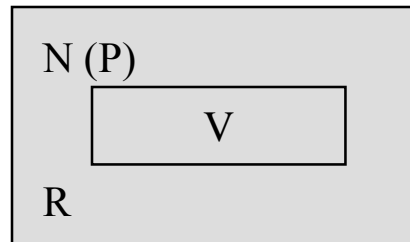
Zusammenfassung

		Daten:	Zeiger:
Zahl:	Def.:	<pre>int a = 1;</pre> 	<pre>int *za = &a;</pre> 
	Zugr.:	<pre>a;</pre>	<pre>* za;</pre>
Feld:	Def.:	<pre>int b [3] = {1, 2, 3};</pre> 	<pre>int *zb = &b [0];</pre> 
	Zugr.:	<pre>b [0]; b [1];</pre>	<pre>*zb; * (zb + 1);</pre>
Struktur:	Def.:	<pre>struct s {int s1; int s2;} c = {1, 2};</pre> 	<pre>struct s * zc = &c;</pre> 
	Zugr.:	<pre>c .s1; c .s2;</pre>	<pre>(* zc) .s1; (*zc) .s2;</pre>

Folie 8.1

Funktionen

- Ziel:**
- Zerlegung eines Programmes in mehrere kleine, überschaubare Bausteine
 - Allgemeinheit, Wiederverwendbarkeit der Bausteine
 - getrennte Erstellung, Übersetzung und Testung der Bausteine
- Vorgehensweise:**
- Funktionsdeklaration --> Funktionsdefinition
 - notwendig: Schnittstellen zwischen Funktionen



- Praxis:**
- Schaffung eigener Funktionen und Funktionsdateien
 - Header-Dateien für vordefinierte Funktionen in Bibliotheken

Folie 8.2

Funktionsdeklaration

Funktionsdeklaration: Vereinbarung eines Funktionsprototyps mit:

- Speicherklasse
- Rückgabedatentyp und Name der Funktion
- Datentypen der formalen Parameter

Speicherklasse Rückgabetyf Funktionsname (Parameterliste)

Bemerkungen:

- Speicherklassen: **extern, static**
- Speicherklasse, Datentypen können entfallen
- Defaultwerte: extern, nullstellig, int
- Deklaration vor erster Verwendung

Beispiele:

```
f1 (); void f2 (void);  
int f3 (int); float f4 (int, float, float, char);  
extern f5 (); static f6 ();  
float berechnung (float a, float b, char c);
```

Folie 8.3

Funktionsdefinition

Funktionsdefinition: Vereinbarung des Programmcodes einer Funktion mit:

- Rückgabedatentyp und Name der Funktion
- Datentypen und Namen der formalen Parameter
- Funktionsblock als Programmcode

Rückgabetyt Funktionsname (Formalparameterliste) Funktionsblock

Bemerkungen:

- formale Parameter im Funktionsblock als lokale Datenobjekte verfügbar
- Rückgabe eines Funktionswertes eines bestimmten Typs mit:
return Ausdruck;

Beispiele:

```
void fehlerausgabe (void) {printf("Fehler\n");}
int rechteckflaeche (int l, int b) {return (l*b);}
float kreisflaeche (float r) {const float pi = 3.14; float f; f = pi*r*r; return f;}

float berechnung (float zahl1, float zahl2, char ope)
{switch (ope)
 {case '+': return (zahl1 + zahl2);
  case '-': return (zahl1 - zahl2);
  default: fehlerausgabe (); return 0;}}
```

Folie 8.4

Funktionsaufruf

Funktionsaufruf:

Aktivieren des Funktionsblockes mit aktuellen Parametern an Stelle der formalen Parameter

Funktionsname (Aktualparameterliste)

Bemerkungen:

- Aktivierungs- und Parameterübergabe
- Ersetzung formaler durch aktuelle Parameter
- Anzahl und Typen der formalen und aktuellen Parameter müssen übereinstimmen
- Hauptfunktion **main** wird beim Programmstart aktiviert
- Merken des Zustandes vor dem Aufruf in einem Kellerspeicher

Beispiele:

```
fehlerausgabe ();  
int rflaeche; rflaeche = rechteckflaeche (2, 4);  
float kflaeche = kreisflaeche (3.0);  
printf("%f", berechnung (1.7, 1.7, '+'));  
float a;  
a = berechnung (float (rechteckflaeche (2,5)), kreisflaeche (3.5), '-');
```

Folie 8.5

Parameterübergabe

Wertübergabe:

- Funktion arbeitet mit Kopie des Wertes
- Wert darf gelesen und lokal verändert werden
- Original bleibt unverändert
- zusätzlicher Raum- und Zeitbedarf

Beispiel:

```
int a = 2;  
printf("%d", a); fkt (a); printf("%d", a);  
void fkt (int*f) {printf("%d", *f); *f = 4; printf("%d", *f);}
```

Zeigerübergabe:

- Funktion arbeitet mit Adresse des Wertes
- Wert darf gelesen und verändert werden
- Original wird mit verändert
- wenig zusätzlicher Raum- und Zeitbedarf

Beispiel:

```
int a = 2;  
printf("%d", a); fkt (&a); printf("%d", a);  
void fkt (int*f) {printf("%d", *f); *f = 4; printf("%d", *f);}
```

Folie 8.6

Übergabe von Feldern und Strukturen

- Felder:**
- Übergabe als Zeiger auf das erste Element
 - Übergabe als Wert wird implizit in Zeiger umgewandelt

Beispiele:

```
int a[3] = {1, 2, 3};
```

```
printf("%d", a[1]); fkt (&a[0]); printf("%d", a[1]);  
void fkt (int*f) {printf("%d", f[1]); f [1] = 4; printf("%d", f[1]);}
```

```
printf("%d", a[1]); fkt (a); printf("%d", a[1]);  
void fkt (int f[3]) {printf("%d", f[1]); f [1] = 4; printf("%d", f[1]);}
```

- Strukturen:**
- Übergabe als Zeiger auf die Struktur
 - Übergabe als Wert der Struktur

Beispiele:

```
struct s {int s1; int s2;} a = {1, 2};
```

```
printf("%d", a.s1); fkt (&a); printf("%d", a.s1);  
void fkt (struct s*f) {printf("%d", (*f).s1); (*f).s1 = 4; printf("%d", (*f).s1);}
```

```
printf("%d", a.s1); fkt (a); printf("%d", a.s1);  
void fkt (struct s f) {printf("%d", f.s1); f.s1 = 4; printf("%d", f.s1);}
```

Folie 8.7

Rekursive Funktionen

rekursive Funktion:

Funktion, die sich selbst aufruft

- direkt: in der Definition der Funktion
- indirekt: in der Definition einer von der Funktion aufgerufenen Funktion

Beispiel:

```
// fakultaet.c
#include <stdio.h>
long double fakultaet (int n)
    {long double erg;
    if (n<=0) return 1;
    erg = n * fakultaet (n - 1);
    return erg;}

void main (void)
    {int zahl; long double ergebnis;
    printf("Fakultaet von? ");
    scanf("%d", &zahl);
    ergebnis = fakultaet (zahl);
    printf("%d! = %f\n", zahl, ergebnis);}
```

Folie 9.1

Arbeit mit Dateien in C

Datei: externer Datenspeicher
kann vom Programm aus gelesen oder beschrieben werden

FILE* stream;	Definition eines Datenzeigers
stream = fopen (datei, mode);	Öffnen einer datei zum Lesen "r", Schreiben "w", Anhängen "a" oder "r+", "w+", "a+"
fclose (stream);	Schließen einer Datei

Operationen:	- fprintf	Schreiben einer Datenstruktur
	- fscanf	Lesen einer Datenstruktur
	- fputc, fputs, fgetc, fgets	Zeichen-, Zeichenkettenoperationen
	- Dateiende: feof (stream)	

Folie 9.2

Beispiel Dateischreiben:

```
#include <stdio.h>
void main (void) {
    FILE* stream;
    stream = fopen ("d:\\zahlen.tab", "w");
    fprintf (stream, "dezimal: \t oktal: \t hexadez.: \n");
    for (int n = 0; n < 256; n++)
        fprintf (stream, "%d \t\t %o \t\t %x \n", n, n, n);
    fclose (stream);}
}
```

Beispiel Dateilesen:

```
#include <stdio.h>
void main (void) {
    FILE* stream; char s [100]; int a, b, c;
    stream = fopen ("d:\\zahlen.tab", "r");
    fscanf (stream, "%s %s %s", s, s, s);
    for (int n = 0; n < 256; n++)
        fscanf (stream, "%d %o %x", &a, &b, &c);
    fclose (stream);}
}
```

Folie 9.3

Standardbibliotheken

C: Funktionsbibliotheken mit Deklarationen in Header-Dateien

Header-Dateien:	- ctype.h	(Zeichen)
	- locale.h	(Internationales)
	- math.h	(Mathematik)
	- setjmp.h, signal.h, assert.h	(Prozesse)
	- stdarg.h	(Parameter)
	- stdio.h, iostream.h	(Datei, Stream)
	- stdlib.h	(Speicher, Transformation, Sortieren, ...)
	- string.h	(Zeichenketten)
	- time.h	(Datum, Uhrzeit)
	- wchar.h	(Zeichen)

Beispiele: math.h: double sin(double), double pow(double,double), double floor(double)
string.h: char* strcpy(char*, char*), char* strcmp(char*, char*, int), int strlen(char*)
stdlib.h: void* malloc(int), void free(void*), void exit(int), int rand(void)
stdio.h: FILE* fopen(char*, char*), int printf(char*, ...), int fgetc(FILE*)

Folie 9.4

Kommandozeilenparameter

Übergabe von Parameter an die Funktion main beim Programmstart

Parameterliste von main:	- int argc	Anzahl der Kommandozeilenparameter
	- char * argv []	Zeiger auf die Kommandozeilenparameter
	- char * envp []	Zeiger auf Umgebungsvariablen
Rückgabewert von main:	void oder int	

Beispiel:

Programm:

```
//passwort.cpp
#include <iostream.h>
#include <string.h>
int main (int argc, char * argv[])
{if (strcmp ("xyz32". argv [1]))
    {cout << "falsches Passwort!"; return 2;}
else cout << "Hallo!";
return 1;}
```

Aufruf:

passwort xyz32

Ausgabe:

Hallo

Rückgabe:

1

Folie 9.5

Präprozessor

Aufgabe:

- Einfügen von Quelltextdateien und Bibliotheken
- Bedingte Übersetzung
- Makrodefinition und -ersetzung
- Setzen von Compileroptionen (implementationsabhängig)

Kommandokennzeichen:

...

Operationen:

#include <Header-FileName>

#include "User-File-Name"

#if Ausdruck

Programm1

#else

Programm2

#endif

#define Makroname Text

#define Makroname (p1, ..., pn) Text

Beispiele:

```
#include <string.h>
```

```
#include "d:\user\oertel\program1.cpp"
```

```
#if __cplusplus
```

```
cout << a;
```

```
#else
```

```
printf(a);
```

```
#endif
```

```
#define mwst 0.15
```

```
sum = betrag + betrag * mwst;
```

```
#define kubik (x) x * x * x
```

```
volumen = kubik (x);
```

Datenbanksysteme - Einführung -

Datenbanktechnologie:

die auf ein Datenbanksystem gestützte Verfahrensweise bei der Speicherung, Verwaltung und Nutzung von Informationenn eines Anwendungsbereiches

Schwerpunkte: Architektur von Datenbanksystemen, konzeptueller Datenbankentwurf, relationales Datenmodell, relationale Sprache, relationale Entwurfstheorie, (Datenintegrität/Datensicherheit, Mehrbenutzersynchronisation, externe Schnittstellen)

Datenbanksystem (DBS): Zusammenfassung von Datenbank und Datenbankmanagementsystem
Datenbank (DB): Sammlung von Datenbeständen einschließlich deren Beschreibung
Datenbankmanagement-system (DBMS): Softwaresystem zur Verwaltung und Bereitstellung von Datenbeständen
Datenmodell (DM): Konzepte für die Abbildung von Diskursbereichen in Datenbanken (Strukturen, Operationen, Integritätsbedingungen)
Datenbanksprache (DL): Sprache zur Definition (DDL) sowie Manipulation und Abfrage (DML) von Datenbeständen

Folie 10.2

Qualitätsmerkmale

- Datenumfang: Verwaltung von Datenbeständen praxisrelevanter Größenordnung
- Zugriffszeit: Datenbereitstellung entspricht zeitlich den Anforderungen der Nutzer
- Redundanzarmut: Mehrfachhaltung von Daten wird vermieden oder reduziert
- Nutzerfreundlichkeit: Nutzer können auf die Daten in einer ihnen gemäßen Form zugreifen
- Konsistenz: Gewährleistung eines gültigen Abbildes der Realität
- Sicherheit: Verfügbarkeit bei Fehlfunktionen von System oder Nutzer
- Effizienz: relativ geringer Aufwand zur Erfüllung der Nutzeranforderungen
- Standardisierung: einheitlich definierte Anwendungsschnittstellen
- Integration: Anwendungsprozesse können durch gemeinsame Datenhaltung zusammenarbeiten
- Fundierung: Theoretische Absicherung von Strukturen und Operationen



Datenunabhängigkeit:

Nutzer werden entlastet von Organisation der Datenverwaltung

Folie 10.3

Entity-Relationship-Modell (ERM)

Formalismus zur Abbildung von Diskursbereichen als Gesamtheit von Gegebenheiten (Entities), zwischen denen Beziehungen (Relationships) bestehen.

Diskursbereich:

Ausschnitt der Realität, der für eine bestimmte Anwendung relevant ist

Beispiel: Universität mit Studenten, Assistenten, Professoren, Lehrveranstaltungen, Prüfungen

Ziele:

- Entwicklung eines schematischen, zeitunabhängigen Modells des Diskursbereiches
- Unterstützung der frühen Entwurfsphasen
- Graphische Veranschaulichung von Zusammenhängen
- Kommunikationsmittel zwischen Anwender, Entwickler und Implementator
- Abbildbarkeit in Datenmodelle (insbesondere in das Relationenmodell)

Folie 10.4

Entities

- Entity: diskrete, identifizierbare Abstraktion eines Gegenstandes der Anschauung oder des Denkens
- Entity-Typ: Verallgemeinerung (Klassifikation) gleichartiger, ähnlicher Entities
- Diagramm-darstellung: Entity-Typ als Rechteck mit Namen des Entity-Typs

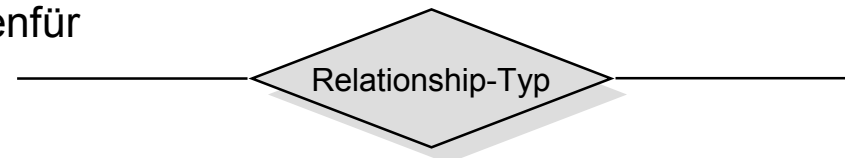
Beispiel: Student, Assistent, Professor, Lehrveranstaltung



Relationships

- Relationship: Beziehungen zwischen je einem Entity von mehreren nicht notwendig verschiedenen Entity-Typen
- Relationship-Typ: Verallgemeinerung (Klassifikation) gleichartiger, ähnlicher Relationships
- Diagrammdarstellung: Relationship-Typ als Rhombus mit Namen des Relationship-Typs, durch ungerichtete Kanten mit korrespondierenden Entity-Typen verbunden

Beispiel: lesen, hören, prüfen, voraussetzen, arbeitenfür



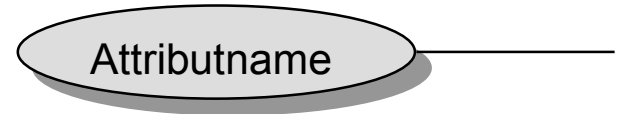
Folie 10.5

Attribute

Attribute: Merkmal zur Beschreibung von Entities und Relationships;
besitzt einen Namen und eine Merkmalswertemenge (Domäne)

Diagrammdarstellung: Attribut als Kreis/Ellipse mit Attributnamen, durch ungerichtete Kante mit Entity- oder Relationship-Typ verbunden

Beispiele: Name, Vorname, Adresse, Fachgebiet, Note, Titel, ...



Schlüssel

Schlüssel: Teilmenge von Attributen,
- deren Werte beliebige Ausprägungen eines Entity- oder Relationship-Typs identifizieren
- und die im Hinblick auf diese Eigenschaft minimal sind

Primärschlüssel: pragmatisch ausgewählter schlüssel eines Entity- oder Relationship-Typs
Der Primärschlüssel eines Relationship-Typs enthält implizit die Primärschlüssel der benachbarten Entity-Typen.

Diagrammdarstellung: Attribut mit Unterstreichung

Beispiel: Persnr, Matrnr, Vorlnr,
(Persnr, Vorlnr), (Vorgänger, Nachfolger), ...



Folie 10.6

Zuordnungscharakteristik

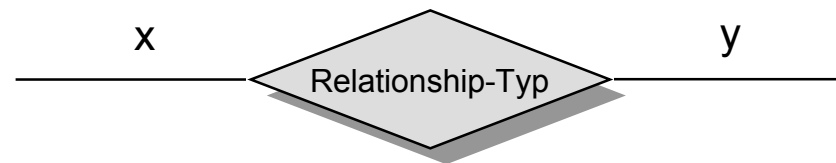
Charakterisierung eines Relationship-Typs R zwischen zwei Entity-Typen E1, E2:

- 1:1-Beziehung: In R kann ein Entity aus E2 mit höchstens einem Entity aus E1 in Beziehung stehen und umgekehrt
- 1:n-Beziehung: In R kann ein Entity aus E2 mit höchstens einem Entity aus E1 in Beziehung stehen
- m:n-Beziehung: Keine Einschränkung

Erweiterung auf mehrstellige Relationship-Typen möglich.

Diagrammdarstellung:

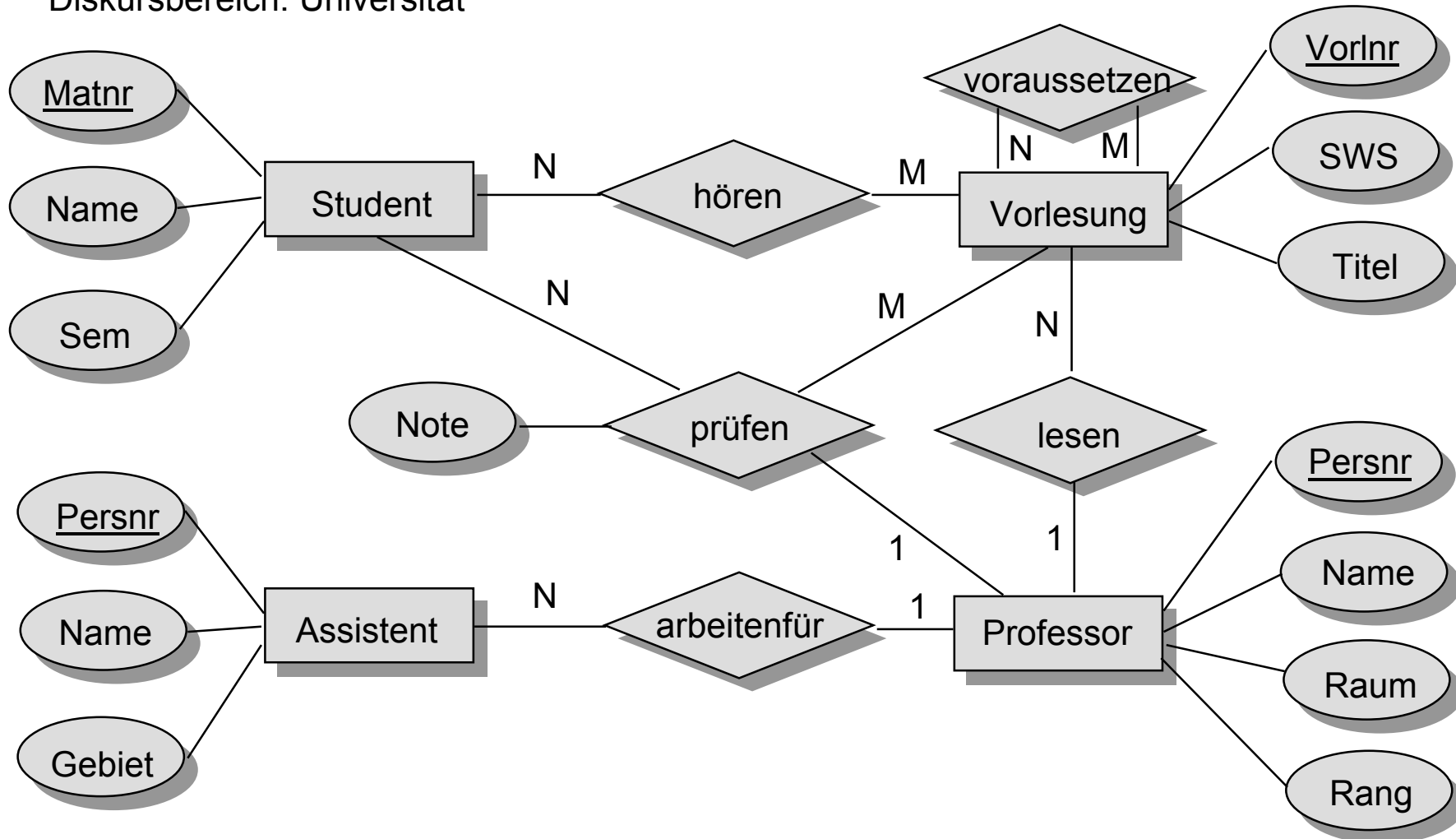
Beschriftung der Kanten des Relationship-Typs mit 1, m, n, ...



Beispiele: lesen (1:n), hören (m:n), prüfen (m:n:1)

ERM - Diagramm

Diskursbereich: Universität



Relationales Datenmodell (RDM)

Relationen, die als Menge von Tupeln bestimmten Integritätsbedingungen genügen, werden mittels Operationen der Relationenalgebra manipuliert.

Strukturen:

Eine Relation R ist eine Teilmenge des kartesischen Produkts von Wertemengen V_j , die Attributen A_j zugeordnet sind ($j=1, 2, \dots, n$): $R=(A_1, A_2, \dots, A_n) \subseteq V_1 \times V_2 \times \dots \times V_n$

Die Elemente $r = (v_1, v_2, \dots, v_n) \in R$ mit $v_j \in V_j$ von R werden als Tupel bezeichnet.

Beispiele: Student (Matrnr, Name, Sem)
 Professor (Persnr, Name, Rang, Raum)
 Vorlesung (Vorlnr, Titel, SWS, Leser)

Zuordnung: Relation ---> Tabelle
 Tupel ---> Zeile
 Attribut ---> Spalte

Operationen:
 (Relationen-
 algebra)

Vereinigung (\cup), Differenz ($-$), Durchschnitt (\cap)
 Projektion (π), Selektion (σ)
 Produkt (\times), Verbund (\bowtie)
 Division (\div), Umbenennung (ρ)

--> Mengenverarbeitung
 --> Attribut-, Tupelauswahl
 --> Relationenverbindung
 --> Quantifizierung, ...

Integrität:

Schlüssel-, Domänen-, Entity-Integrität, referentielle Integrität

Folie 10.9

Überführungsvorschriften ERM ---> RDM

Entity-Typ:	Bildung eines Relationsschemas mit dem Namen, den Attributen und den Schlüsseln des Entity-Typs
Relationship-Typ m:n:	Bildung eines Relationsschemas mit dem Namen des Relationship-Typs, den eigenen Attributen und den Primärschlüsseln der beteiligten Entity-Typen
Relationship-Typ 1:n:	Aufnahme des Primärschlüssels des 1-Entity-Typs als Fremdschlüssel und der Attribute des Relationship-Typs in das Relationsschema des n-Entity-Typs
Relationship-Typ 1:1:	Aufnahme des Primärschlüssels eines 1-Entity-Typs als Fremdschlüssel und der Attribute des Relationship-Typs in das Relationsschema des anderen 1-Entity-Typs
Relationship-Typ mehrstellig:	analog zu Relationship-Typ m:n

Folie 10.10

Beispielrelationen 1

Assistent:			
<u>Persnr</u>	Name	Fachgebiet	Boss
3002	Platon	Ideenlehre	2125
3003	Aristoteles	Syllogistik	2125
3004	Wittgenstein	Sprachtheorie	2126
3005	Rhetikus	Planetenbewegung	2127
3006	Newton	Keplersche Gesetze	2127
3007	Spinoza	Gott und Natur	2134

Professor:			
<u>Persnr</u>	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Student:		
<u>Matrn</u>	Name	Sem
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
26830	Aristoxenos	8
27550	Schopenhauer	6
28106	Carnap	3
29120	Theophrastos	2
29555	Feuerbach	2

Folie 10.11

Beispielrelationen 2

Vorlesung:			
<u>Vorlnr</u>	Titel	SWS	Leser
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

voraussetzen:	
<u>Vorgänger</u>	<u>Nachfolger</u>
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

hören:	
<u>Matrn</u>	<u>Vorlnr</u>
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022
25403	5022
29555	5001

prüfen:			
<u>Matrn</u>	<u>Vorlnr</u>	<u>Persnr</u>	<u>Note</u>
28104	5001	2126	1
25403	5041	2125	2
27550	4630	2137	2

Folie 11.1

Relationale Datenbanksprache SQL

Datenbanksprache (DL):

- Datendefinitionssprache (DDL)
 - Datenmanipulationssprache (DML):
 - Änderungssprache
 - Anfragesprache (QL)
-

Datenmanipulationssprache/Anfrage

Standardanfrageschema:

select ...	Projektion
from ...	Relation/Produkt/Join
where ...	Selektion/Differenz/Durchschnitt
all/any/in/exists ...	Division/Quantifizierung
group by ...	Aggregation
union ...	Vereinigung
order by ...	Sortierung

Folie 11.2

Einfache Anfragen:

select	Spalte(n)
from	Tabelle(n)
where	Bedingung;

Anzeigen von Spalte(n)
der Zeilen aus Tabelle(n),
die einer Bedingung genügen

Anfragen über einer Tabelle:

Beispiele:

```
select Matrnr,Name,Sem  
from Student  
where Sem<5;
```

```
select*  
from Assistent;
```

```
select distinct Sws  
from Vorlesung  
where Leser=2125 and not Titel='Ethik';
```

Anfragen über mehreren Tabellen:

Beispiel:

```
select Name, Titel  
from professor, Vorlesung  
where Persnr=Leser  
and Titel='Logik';
```

```
select Name, Titel  
from Student, hören, Vorlesung  
where Student.Matrnr=hören.Matrnr  
and hören.VorlNr=Vorlesung.VorlNr;
```

```
select s.Name, v.Titel  
from Student s, hören h, Vorlesung v  
where s.Matrnr=h.Matrnr  
and h.VorlNr=v.VorlNr;
```

Folie 11.3

Spaltenauswahl:

Spaltenname, Konstante
*
distinct, all

Auswahl von Spalten, Konstanten
Auswahl aller Spalten
Duplikatebeseitigung

Tabellenauswahl:

Tabellenname
Aliasname

Auswahl von Tabellen
temporäre Umbenennung

Zeilenauswahl:

Spaltenname, Konstante
=, !=, <>, >, >=, <, <=
and, or, not
between ... and
like '...%..._...'
is null, is not null

Angabe von Spalten und Konstanten
Vergleichsoperator
logische Verknüpfung
Bereichsangabe
Zeichenmuster
Leerwert

Geschachtelte Anfragen:

```
select * from Assistent
where Boss =
(select Persnr from Professor
where Name = 'Sokrates');
```

```
select * from prüfen
where Note =
(select avg (Note) from prüfen);
```

Folie 11.4

Mengenoperationen:

union
(intersect, except)

Vereinigung von Tabellen
(Durchschnitt, Differenz)

Beispiele:

```
(select Name from Assistent)
```

```
union
```

```
(select Name from Professor);
```

```
(select VorlNr from Vorlesung)
```

```
except
```

```
(select VorlNr from hören);
```

```
select VorlNr from Vorlesung
```

```
where VorlNr not in
```

```
(select VorlNr from hören);
```

Quantifizierung:

all, any, some
in, not in
exists, not exists

für alle, für ein
Element von, nicht Element von
es existiert ein, es existiert kein

Beispiele:

```
select Name  
from Student  
where Sem >= all  
(select Sem  
from Student);
```

```
select Name  
from Professor  
where not exists  
(select *  
from Vorlesung  
where Leser = Persnr);
```

```
select *  
from Vorlesung  
where VorlNr in  
(5001, 5041);
```

Folie 11.5

Gruppierungsfunktion:

```
count, sum, avg  
max, min
```

Ausführen von Operationen auf
Tupelmengen

Beispiele:

```
select avg (Sem)  
from Student;
```

```
select min (Note), max (Note)  
from prüfen  
where Vorlnr = 5001 or Vorlnr = 5041;
```

```
select count (*)  
from hören;
```

Gruppierung:

```
group by Spalte(n)  
having Bedingung
```

Gruppierung der Zeilen der Ergebnistabelle
und Ausführung von Gruppierungsfunktionen

Beispiele:

```
select Leser, sum (Sws)  
from Vorlesung  
group by Leser;
```

```
select Leser, sum (Sws)  
from Vorlesung  
group by Leser  
having avg (Sws) > 3;
```

```
select Leser, Name, sum (Sws)  
from Vorlesung, Professor  
where Leser = Persnr and Rang = 'C4'  
group by Leser, Name  
having avg (Sws) > 3;
```

Folie 11.6

Elementfunktion:

```
+ , - , * , / , ...  
char_length, substring, ||, ...  
current_time, current_date, + , - , * , ...
```

arithmetische Funktion
Zeichenkettenfunktion
Datumsfunktion

Beispiele:

```
select Matrnr, Note - 1  
from prüfen  
where Vorlnr = 5001;
```

```
select Rang || '-Professur'  
from Professor;
```

```
select Matrnr, Sem, current_date  
from Student;
```

Sortierung:

```
order by Spalte(n)  
asc/desc
```

Festlegen der Sortierreihenfolge
der Ergebnistabelle (aufsteigend
oder absteigend)

Beispiele:

```
select Persnr, Name, Raum  
from Professor  
where Rang = 'C4'  
order by Rang desc, Name asc;
```

```
select Name  
from Student  
order by Matrnr;
```

Folie 11.7

Datenmanipulationssprache/Änderung

Standardänderungsoperationen:

insert ...	Einfügen
update ...	Änderung
delete ...	Löschen

Einfügen:

```
insert into Tabelle  
values Tupel;  
insert into Tabelle  
Anfrage;
```

Einfügen von Zeilen
in eine existierende Tabelle

Beispiele:

```
insert into Professor  
values (2136, 'Curie', 'C4', null);
```

```
insert into Student (Matrn, Name)  
values (25000, 'Goethe');
```

```
insert into Assistent (Persnr, Nmae)  
select Matrnr, Name  
from Student  
where Sem > 15;
```

Folie 11.8

Ändern:

update Tabelle
set Werteänderung
where Bedingung

Ändern von Zeilen
in einer existierenden Tabelle

Beispiele:

```
update Vorlesung  
set Sws = 2;
```

```
update student  
set Sem = Sem + 1;
```

```
update Professor  
set Raum = 213  
where name = 'Russel';
```

```
update Vorlesung  
set Titel = 'Grundlagen', Sws = 3, Leser = 2125  
where Vorlnr = 5001;
```

Löschen:

delete from Tabelle
where Bedingung

Löschen von Zeilen
in einer existierenden Tabelle

Beispiele:

```
delete from prüfen;
```

```
delete from Vorlesung  
where Vorlnr = 5001;
```

```
delete from student  
where Sem >= 15;
```

```
delete from voraussetzen  
where Vorgänger in  
(select Nachfolger  
from voraussetzen);
```

Datendefinitionssprache

Standarddefinitionsoperationen:

Erzeugen (**create**), Ändern (**alter**). Löschen (**drop**) für:
Tabelle (**table**), Sicht (**view**), ...
und Erteilung (**grant**) und Entzug (**revoke**) von Rechten

Datentypen: char(n), varchar(n), number(p), number (p,s), date, long, blob, ...
Namen: 1.Zeichen Buchstabe, dann Buchstaben, Ziffern, _ oder \$

Tabellen:

```
create table Tabellen-Name  
(Spalten-Name Typ,  
Spalten-Name Typ not null,  
Spalten-Name Typo not null unique,  
...)
```

Definition einer Tabelle
mit Spalten
und Integritätsbedingungen

Beispiele:

```
create table Professor  
(persnr number (5) not null unique,  
Name varchar (20) not null,  
Rang char (2),  
Raum number (4));
```

```
create table voraussetzen  
(Vorgänger number (5),  
Nachfolger number (5));
```

Folie 12.1

Datenkommunikation

- Einführung -

Datenkommunikation:

Austausch binär codierter Daten zwischen Kommunikationspartnern
Hauptgebiete: Struktur und Funktion von Daten- und Rechnernetzen und Diensten

Datennetz:

Nachrichtennetz für die Übertragung von Daten zwischen über Schnittstellen angeschlossenen Partnern

Beispiele: analoge Netz, ISDN, ATM-basiertes ISDN, Mobilfunknetz, Datex-P, Datex-M, ...

Rechnernetz:

Verteiltes System von autonomen über ein Datennetz miteinander kommunizierenden Computern, das seinen Nutzern Dienste zur Verfügung stellt

Klassifikation von Rechnernetzen nach räumlicher Ausdehnung und Rechneranzahl:

- lokale Netze (LAN): Ethernet, Token-Ring, FDDI-Netz, ...
- flächendeckende Netze (WAN): Internet, Intranet, ...

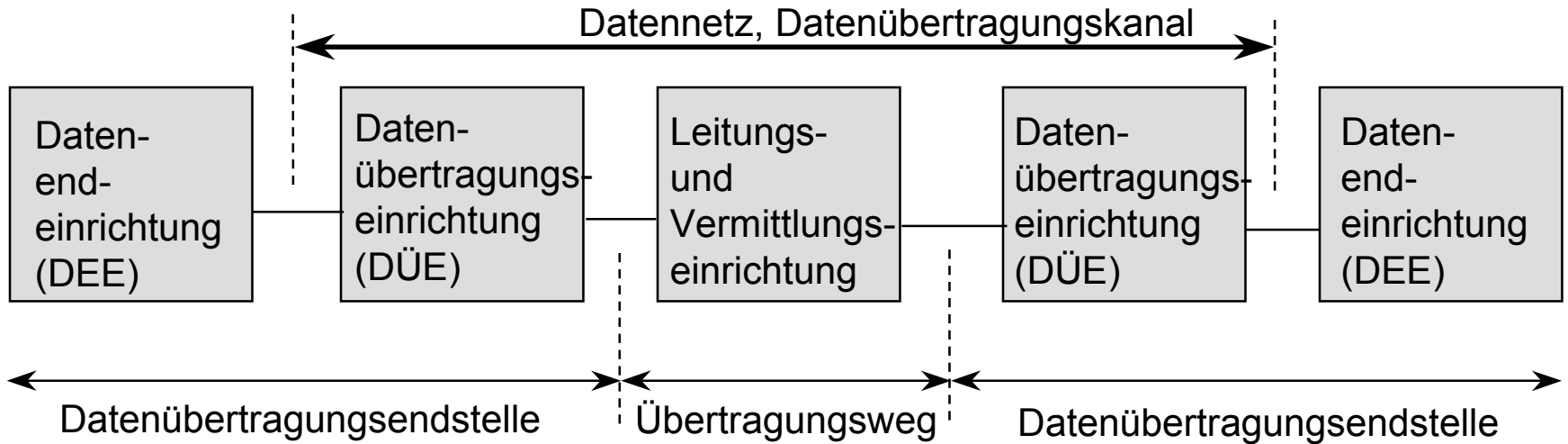
Client-Server-Technologie:

- Erbringung von Diensten für einen Nutzer (Client) durch einen Anbieter (Server)

Folie 12.2

Aufbau eines Daten- /Rechnernetzes:

Datenübertragungssystem:



Datenendeinrichtung:

Komponente, die Quelle oder Senke von Daten enthält, z.B. Rechner, Terminal, Mikrophon, Lautsprecher

Leitungs- und Vermittlungseinrichtung:

Komponente zur eigentlichen Datenübertragung, z.B. Leitung, Funkverbindung, diverse Übertragungskomponenten

Datenübertragungseinrichtung:

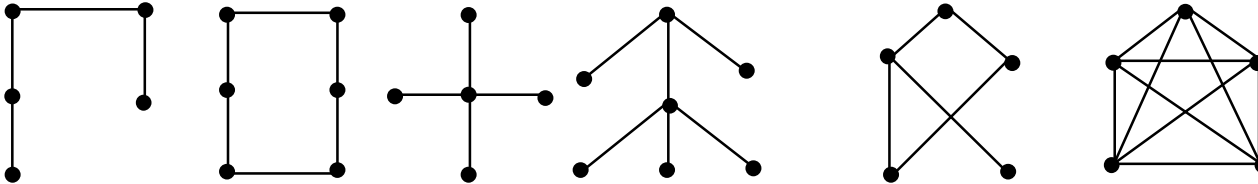
Komponente, die von der DEE gesendete Daten an den Übertragungsweg anpaßt und empfangene Daten in eine von der DEE erkennbare Form zurückverwandelt, z.B. Modem, Fehlerkorrektureinrichtung

Eigenschaften eines Übertragungskanals:

- Leistung:
 - Bandbreite: für die Signalübertragung zur Verfügung stehender Frequenzbereich (Hertz)
 - Kanalkapazität: Anzahl der pro Zeiteinheit übertragbaren Informationseinheiten (Bit/Sekunde)
- Modulation:
 - Beeinflussung eines deterministischen Trägersignals durch informationstragende Signale:
Amplituden-, Frequenz-, Phasenmodulation
- Richtung:
 - Wechselseitige Nutzung durch Signale:
Simplex-, Halbduplex-, Duplexbetrieb
- Verbindung:
 - Art der Verteilung der an das Netz übergebenen Daten:
Punkt-zu-Punkt, Mehrpunkt
- Synchronisation:
 - Herstellung des Zeitbezuges zwischen eintreffenden Signalen und Empfänger:
synchron (Bit, Zeichen), asynchron (Start, Stop)
- Fehlerbehandlung:
 - Schutz der Datenübertragung vor Störeinflüssen:
Fehlererkennung (Paritätsbits, Blockprüfzeichen), Fehlerkorrektur (Selbstkorrektur, Wiederholung)

Netzstruktur/Netztopologie:

Zuordnung zwischen mehreren Datenend- und Datenvermittlungseinrichtungen:
Linien-, Ring-, Stern-, Baum-, teilvermaschtes, vollvermaschtes Netz



Vermittlungsprinzipien:

Art und Weise, wie Datenströme im Netz zwischen Kommunikationspartnern weitergeleitet werden

- **Leitungsvermittlung:**
Feste direkte leitungsmäßige Verbindung zwischen Sender und Empfänger für die Dauer der Datenübertragung
- **Paketvermittlung:**
Daten werden in kleinen adressierten Datenfragmenten dem Netz übergeben und einzeln an den Empfänger weitergeleitet

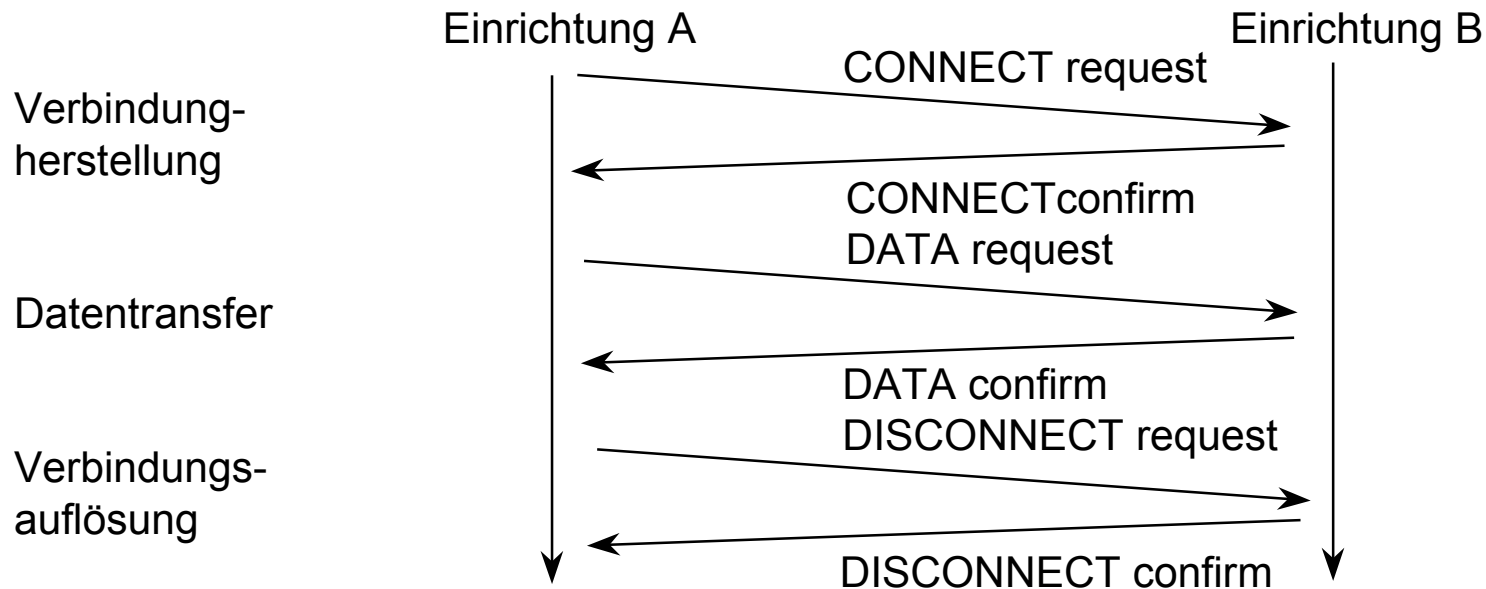


Vor- und Nachteile: Zeitverhalten, Netzauslastung, Fehlerbehandlung, Sicherheit, ...

Folie 12.5

Kommunikationsprotokolle:

- Kommunikationsprotokoll:** Gesamtheit aller syntaktischen und semantischen Festlegungen für den Datenaustausch zwischen Kommunikationspartnern
Bestandteile: Verbindungsaufnahme, Verbindungsabbau, Reaktion auf Ereignisse, zeitlicher Ablauf, Formate für Nutzerdaten, Steuerdaten
- Dienstprimitive:** implementationsunabhängige Beschreibung elementarer Kommunikationsvorgänge
- Ablaufdiagramm:** graphische Darstellung des sequentiellen Ablaufs von Kommunikationsprozessen



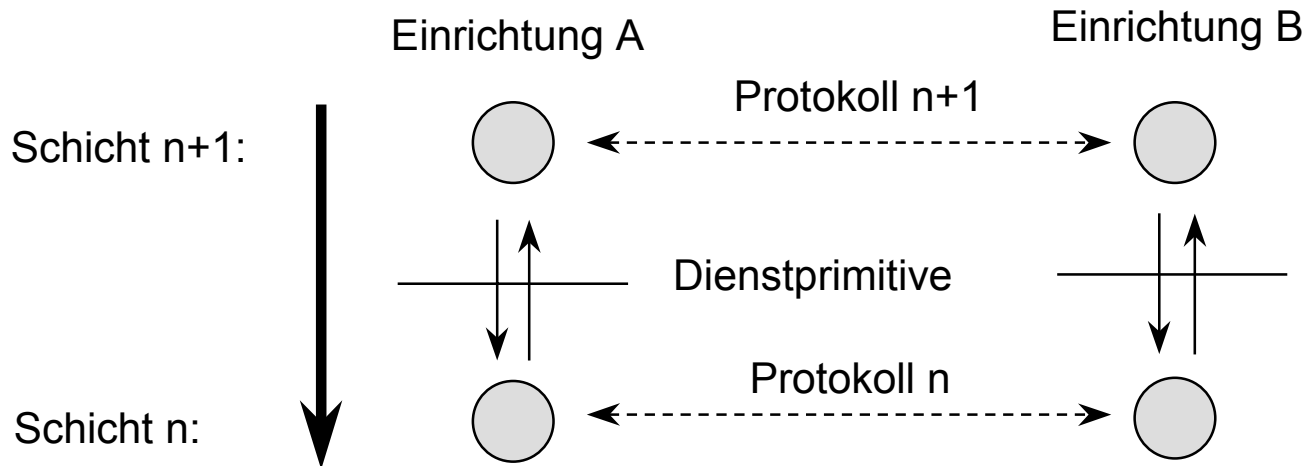
Folie 12.6

Dienstmodell:

Dienstmodell: Abstraktes Darstellungsmittel für Kommunikationsbeziehungen in hierarchischen Mehrschichtenarchitekturen von Rechnernetzen auf der Basis von Diensten

Dienstbenutzer: abstrakte Einrichtung, die einen Dienst nutzt

Diensterbringer: abstrakte Einrichtung, die einen Dienst anbietet



- verbindungsorientiert:

Zwischen den Kommunikationspartnern wird eine logische Verbindung aufgebaut. Dann werden Daten unter Einhaltung der Reihenfolge übertragen. Schließlich wird die Verbindung aufgelöst.

- verbindungslos:

Es existiert keine ständige Verbindung zwischen den Kommunikationspartnern. Mit Adresse versehene Daten werden der Übertragungseinrichtung übergeben. Dann sorgt Übertragungseinrichtung selbständig für Weiterleitung der Daten. Reihenfolge der gesendeten Daten ist beim Empfang nicht garantiert.

7-Schichten-Architektur:

(OSI-Referenzmodell der ISO)

7: Anwendungsschicht (Application Layer)

direkt nutzbare Anwendungen, wie Dateitransfer, Computerfernbedienung, Email

6: Darstellungsschicht (Presentation Layer)

Mechanismen zum Austausch unterschiedlicher Informationsdarstellung, wie Komprimierung, Verschlüsselung, Transformation

5: Sitzungsschicht (Session Layer)

Datenverarbeitung, wie Verbindungsaufbau, Aktivitätsverwaltung, Synchronisation, Dialogsteuerung

4: Transportschicht (Transport Layer)

nachrichtenorientierte, Ende-zu-Ende-Übertragung, wie schrittweises Weiterreichen einzelner Datenpakete durch das Netz

3: Netzschicht (Network Layer)

Nachrichtenübertragung zwischen direkt verbundenen Rechnern, wie Routing, Adreßermittlung

2: Sicherungsschicht (Data Link Layer)

Gewährleisten der Nachrichtenübertragung über den Übertragungskanal, wie Fehlerbehandlung

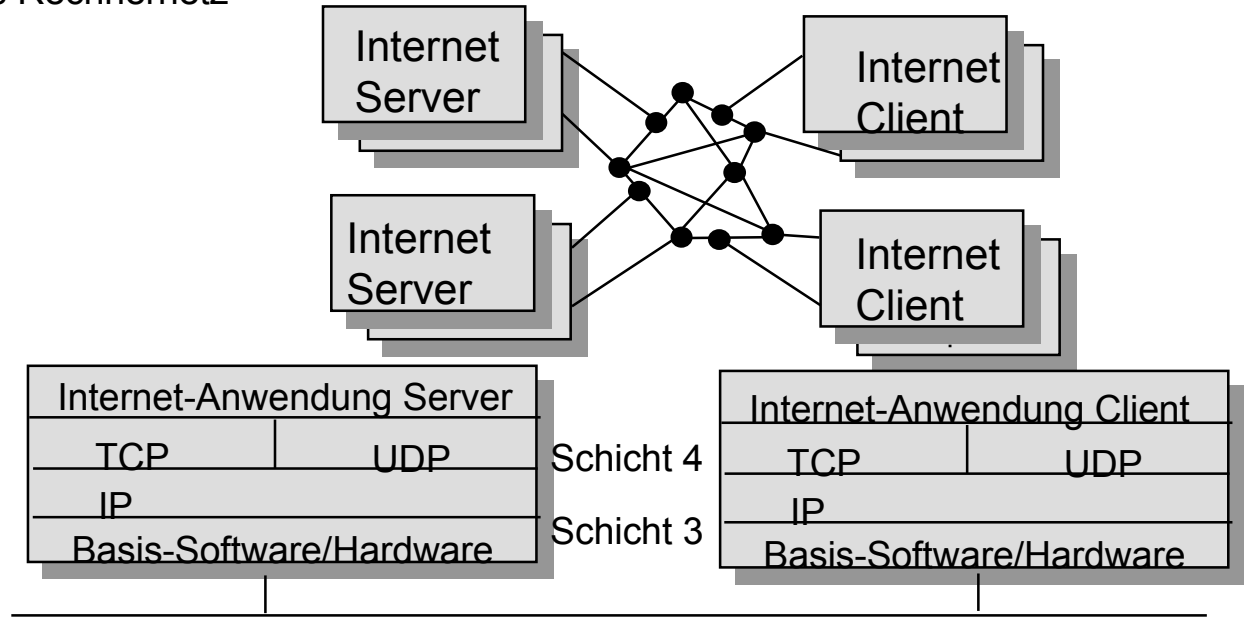
1: Bitübertragungsschicht (Physical Layer)

physische Übertragung von Bit-Strömen, wie mechanische und elektrische Festlegungen, Spannung, Frequenz, Takt, Stecker

Folie 12.8

Internet: TCP/IP-basiertes, weltweites, aus mehreren Teilnetzen bestehendes heterogenes Rechnernetz

Netzwerkstruktur:
(teilvermaschtes Netz mit Servern und Clients)



Protokollhierarchie:
(Architektur aus 4 Schichten, von denen die mittleren festgelegt sind)

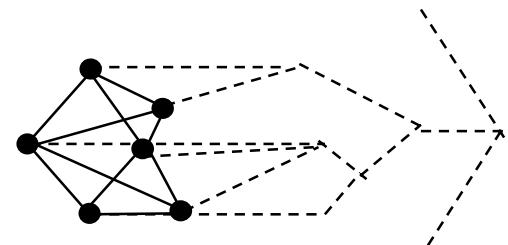
TCP (transmission control protocol): verbindungsorientierter, zuverlässiger, vollduplex Dienst

UDP (user datagram protocol): einfacher, verbindungsloser, unzuverlässiger Dienst

IP (internet protocol): verbindungsloser, unzuverlässiger Dienst

IP-Adressen: 32 Bit langes Wort, das einen Rechner im Internet physisch eindeutig adressiert,
z.B. 141.76.30.99, 192.112.36.4

DNS (domain name system): hierarchische Zuordnung von logischen Namen zu IP-Adressen.
z.B. iki378.inf.tu-dresden.de, orcl1.oracle.com



Folie 13.1

Sprachen für die Datenkommunikation

Werkzeuge und Dienste im Internet

- Telnet: Online-Zugriff für die Arbeit auf anderen Rechnern
- FTP: Übertragung von Dateien zwischen Rechnern
- Email: Nachrichtenaustausch zwischen Rechnern
- HTML: Sprache zur Herstellung von Hypertext-Dokumenten
- WWW: hypertextbasierter/hypermedialer, verteilter Informationsbeschaffungsdienst
- Sonstige: News, Finger, Wais, Whois, Archie, Gopher

Folie 13.2

Telnet

Ziel:	Arbeit auf Rechnern im Netz, als ob Tastatur und Bildschirm direkt an diesen angeschlossen wären
Voraussetzungen:	Zugangsberechtigung auf anderem Rechner (Account) keine lokale Installation erforderlich
Arbeitsweise:	Programme laufen auf entferntem Rechner Bildschirmausgaben und Tastatureingaben auf lokalem Rechner

Sprache:

Beispiel:

telnet

open Rechnername/IP-Adresse

login: Nutzername

Password: *****

Kommandoeingabe ...

exit

Kommandos:

softwarespezifisch für entfernten Rechner

telnet

open rechner1.inf.tu-freiberg.de

login: Jasper

Password: *****

cd files

ls

rm file1.txt

cp file2.txt file3.txt

exit

Folie 13.3

FTP

Ziel:	Transfer von Dateien zwischen verschiedenen Rechnern
Voraussetzung:	Zugangsberechtigung oder anonymer Zugang auf anderen Rechner
Arbeitsweise:	Herstellen Verbindung zwischen Rechnern, Orientierung in Verzeichnissen (README, INDEX, pub, anonymous), Einstellung von Optionen, Dateitransfer

Sprache:

Beispiel:

ftp
open Rechnername/IP-Adresse
login: Nutzernamen/anonymus
Password: *****/eigene Email-Adresse
Kommandoeingabe ...
close
quit

ftp
open rechner1.inf.tu-freiberg.de
login: Jasper
Password: *****
cd pub
dir
delete file1.txt
rename file0.txt file1.txt
ascii
put file2.txt
get README
binary
lcd transfer
mget *.tar.Z
mget bild*.*
close
quit

Kommandos: **help**

ascii, binary (Transfermodus einstellen)
pwd, dir, cd, lcd (Arbeit mit Verzeichnissen)
get, mget, put, mput (Dateitransfer)
mkdir, rmdir, delete, mdelete, rename
(Erzeugen, Löschen, Umbenennen von Verzeichnissen und Dateien)

Folie 13.4

Email

Ziel:	Austausch elektronischer Post zwischen Nutzern unterschiedlicher Rechner
Voraussetzungen:	Installation eines Mail-Servers Besitz einer Email-Adresse: <code>Nutzername@Domänenname</code>
Arbeitsweise:	Empfangen und Lesen, Erstellen und Senden, Verwalten von Mails

Sprache:

elm, mail, ...

Kommandoeingabe ...

exit

Kommandos (softwarespezifisch):

previous Bereitstellen vorherige Mail

next Bereitstellen nächste Mail

save Speichern

delete Löschen

print Drucken

new Erzeugen

send Senden

reply Antworten

forward Weiterleiten

Versenden von Nichttextdateien: Uuencode/Uudecode, MIME, Attachment

Aufbau einer Email:

Sender:

To: durch Komma getrennte Empfängeradressen

Subject: Betreff/Titel

Cc: zusätzliche Empfängeradressen zur
Kenntnisnahme

Empfänger:

From: Absenderadresse

Subject: Betreff/Titel

To: Empfängeradresse

Id: Identifikationsnummer

Date: Absendedatum und -uhrzeit

Received: Weg durchs Netz

Folie 13.5

WWW

Ziel:	Hypertextbasierter/hypermedialer, verteilter Informationsbeschaffungsdienst
Voraussetzungen:	Web-Server bzw. Web-Clients (Browser)
Arbeitsweise:	Hypertext-/Hypermediadokumente mit Texten, multimedialen Objekten und Verweisen auf andere Dokumente werden unter eindeutigen Adressen auf Servern verwaltet. Clients können mittels Browsern Dokumente anzeigen, aktivieren und Verweise verfolgen.

- Geschichte:
- 1970 ARPA Net
 - 1974 TCP/IP
 - 1990 WWW

Eindeutiger Dokumentenidentifikator (URL):

- Bestandteile:
- Zugriffsmethode/Protokoll: http:, file:, ftp:, mailto:, news:, gopher:, ...
 - Rechnername: //www.informatik.tu-muenchen.de
 - Dateiname: /oo/info/progr

Sprache: HTML (Hypertext Markup Language) ---> Textsatzsprache mit Hypermedialinks

Browseroperationen:

- Navigation: Goto, Back, Forward, Home, History, ClearHistory, Hotlist
- File: NewWindow, Reload, Refresh, ViewSource, Save, Print, MailTo, CloseWindow
- Option: LoadImages, Fonts, LoadToLocalDisk
- Annotation: Add, Delete, Edit

Entwicklungstool: menügestützter Aufbau von Web-Seiten

Folie 13.6

Html

Ziel: Sprache zur Beschreibung von Textdokumenten mit Verweisen auf andere Textdokumente und zusätzlichen Einbindungsmöglichkeiten für andere, insbesondere multimediale, Dokumente und aktivierbare Programme

Syntax: <Tag_Name Attribut= Wert> Dokumenttext </Tag_Name>
Schließendes Tag, Dokumenttext und Attributangabe können teilweise entfallen

<html>...</html>	Dokument	<hr>...</hr>	horizontale Linie
<head>...</head>	Kopf	...	Liste
<title>...</title>	Titel	...	nummerierte Liste
<isindex>	Index	...	Listenelement
<body>...</body>	Körper	<dl>...</dl>	Beschreibung
...	Sprungmarke	<dt>...	Beschreibungselement
<h1>...</h1>	Überschrift	...	Verweis auf Dokument
...	Fettdruck	...	Verweis auf Sprungmarke
<i>...</i>	Kursivdruck		Bild
<p>	Absatz	<sound src="...">	Sound
 	neue Zeile		Video
<pre>...</pre>	direkter Text	<table...>...</table>	Tabelle
<big>...</big>	größere Schrift	<applet...>...</applet>	Applet
<small>...</small>	kleinere Schrift	<form...>...</form>	Formular
...	Schriftgröße		
...	Schriftfarbe		

Html-Beispiel:

```
<html>
<head>
<title>
Ein kurzes Beispiel
</title>
</head>
<body>
<h1>
&Uuml;berschrift
</h1>
Hier kommt der normale Text, der aus
mehreren Zeilen bestehen kann. <br>
<h3>
Zwischen&uuml;berschrift
</h3>
Listen werden verwendet, um
Aufz&auml;hlungen zu strukturieren.
<ul>
<li> Punkt 1
<li> Punkt 2
</ul>
```

```
<a href="#Ende">Hier</a> befindet sich ein
Verweis auf das Ende des Dokumentes.
<p>
Weitere Details zu diesem Dokument sind zu
finden unter:
<a href="http://server1.inf.uni-kl.de/data/details.html">
Detailverweis </a><br>
Hier erscheint ein Bild <br>
<br>
zu dem Musik
<sound src="file:/sound.aud">
gespielt wird.
<p>
<a name="Ende">
<i>
<font color=green>
Das ist das Ende der Beispieldatei.
</i>
</font>
</body>
</html>
```

Folie 14.1

Softwaretechnologie

- Programmieren im Kleinen: Softwareerstellung für eine kleine, überschaubare, abgeschlossene Aufgabenstellung, die von einer Person bewältigt werden kann
- Programmieren im Großen: Softwareerstellung für eine umfangreiche, komplexe Aufgabenstellung, die eine Bearbeitung durch ein Entwicklerteam erforderlich macht.

Softwareprodukt:

Programmsystem mit Programmen, Daten und Dokumentationen, das für eine Problemstellung in Auftrag gegeben, entwickelt, genutzt und gewartet wird

Softwaretechnologie (Softwareengineering):

geplantes, systematisches und wirtschaftliches Vorgehen bei der Softwareentwicklung unter Anwendung von Prinzipien, Methoden, Werkzeugen, Normen und Hilfsmitteln, die den Prozeß technisch und organisatorisch unterstützen

Softwarequalität

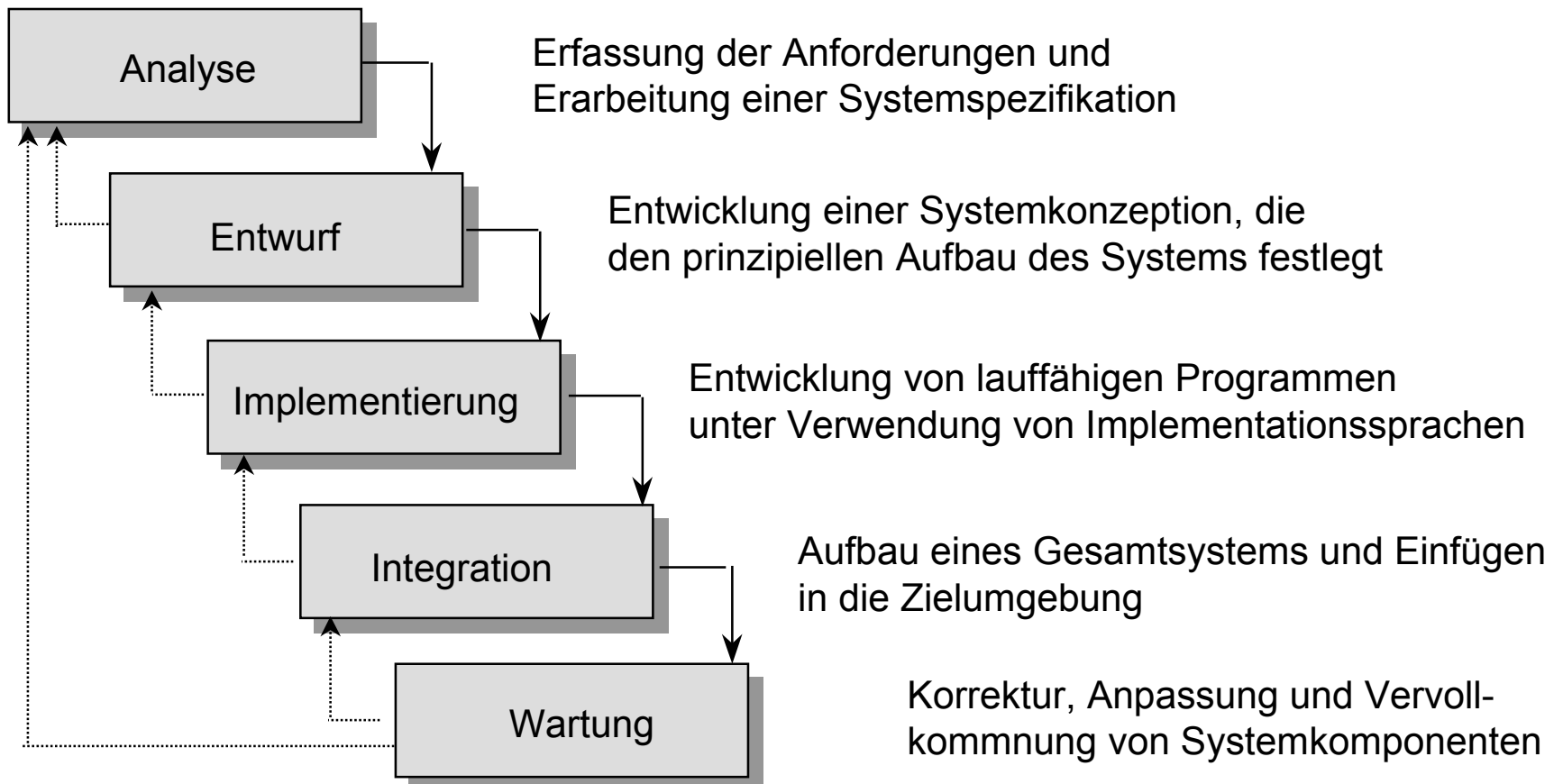
komplexe Größe, die sich aus einer Menge von Eigenschaften zusammensetzt

Benutzbarkeit:	Leichtigkeit der Bedienung durch den typischen Benutzer
Integrität:	Fähigkeit, Komponenten vor unberechtigtem Zugriff und unberechtigter Veränderung zu schützen
Effizienz:	ökonomische Nutzung von Computerressourcen wie Speicherplatz und Prozessorzeit
Korrektheit:	exakte Erfüllung der Programmspezifikation
Zuverlässigkeit:	fehlerfreies Funktionieren in einem Beobachterzeitraum
Verknüpfbarkeit:	Möglichkeit, zwei Produkte miteinander zu verbinden
Wartbarkeit:	Aufwand, einen Fehler im System zu lokalisieren und zu beheben
Testbarkeit:	Möglichkeit, die Erfüllung von Anforderungen praktisch nachzuweisen
Flexibilität:	Möglichkeit, ein Produkt zu modifizieren und anzupassen
Portabilität:	Möglichkeit, ein Produkt von einer Hard-/Softwareumgebung auf eine andere zu übertragen
Wiederverwendbarkeit:	Möglichkeit, ein Produkt in einer anderen Umgebung einzusetzen
Standardisierung:	Nutzung von standardisierten Sprachen und Modellen

Softwarelebenszyklus

Entwicklung, Nutzung, Wartung und wiederholte Weiterentwicklung eines Softwareproduktes

Phasenmodell: Einteilung des Softwarelebens in inhaltlich abgegrenzte Abschnitte und Festlegung der zeitlichen Aufeinanderfolge



Methoden und Werkzeuge

generelle Klassifikation:

- organisatorisch (Projektmanagement, Arbeitsorganisation, Kommunikation, Vorgehensmodell, ...)
- konstruktiv (Strukturierungs-/Darstellungsmittel, Compiler, Generator, Dokumentationsrichtlinien, ...)
- analytisch (Test, Metrik, Verifikation, Zertifizierung, ...)

Entwicklungswerkzeug:

Meta-System zum Aufbau anderer Systeme

CASE (Computer Aided Software Engineering):

aufeinander abgestimmte, durchgängige Anwendung von Softwarewerkzeugen im Prozeß der Entwicklung von Softwaresystemen
(Upper CASE, Lower CASE)

Merkmale: Integration, Offenheit, Nutzerinterface, Verwaltung
Mehrnutzerfähigkeit, Automatisierung, Konsistenz

Beispiele: Developer Studio (Microsoft)
Innovator (MID)
Designer (Oracle)
Rational Rose (Rational)

Phasenspezifische Methoden und Werkzeuge

● Analyse:

- Recherche, Interview, Diskussion
- Machbarkeitsstudie
- Function-Point-Methode
- Pflichtenheft
- > Spezifikation, Anforderung, Wunsch

● Entwurf:

- Zerlegung durch Dekomposition und Schichtung (Modularisierung, bottom-up, top-down)
- funktionsorientierte Zerlegung (Struktogramm)
- datenorientierte Zerlegung (ERM)
- objektorientierte Zerlegung (UML)
- > Architektur, Algorithmus, Datenstruktur

● Implementierung:

- strukturierte Programmierung
- Editor, Generator
- Compiler, Interpreter
- Testumgebung, Debugger
- System- und Anwenderdokumentation
- > Programm, Daten, Dokumentation

● Integration:

- Installation (Setup, Make)
- Middleware, Tuner, Monitor, Simulator
- Abnahme, Freigabe
- Schulung, Vertrieb
- > System, Endprodukt

● Wartung:

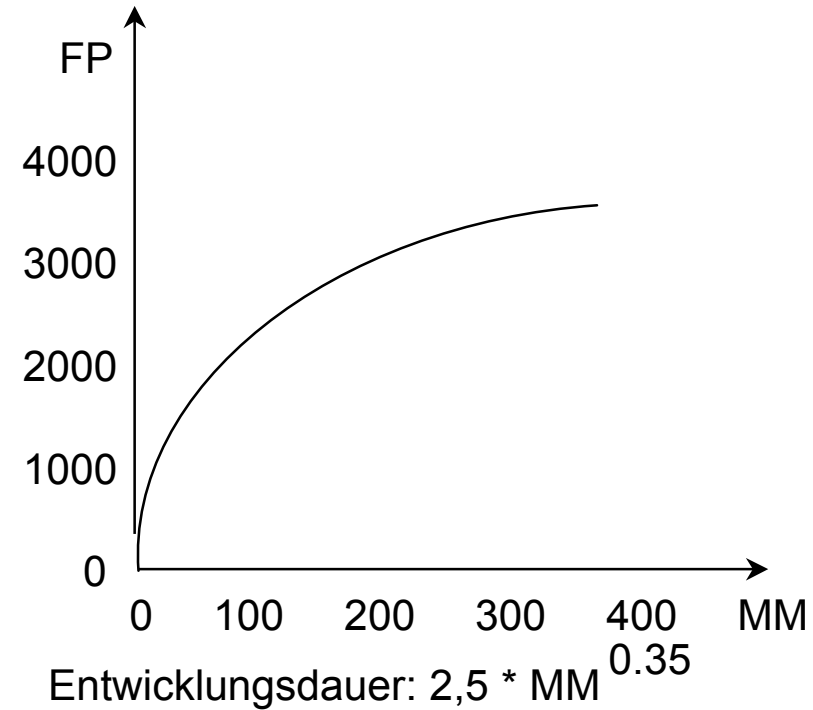
- Betrieb, Nutzung
- Support, Hotline
- Patch, Upgrade, Migration
- > neues System
neue Anforderung

Folie 14.6

Softwaremetrik: quantifizierte Beschreibung der Eigenschaften von Software und deren Entwicklungsprozeß

Analyse: - Komplexität der Anforderungen (Function-Point-Methode)

Kriterium / Merkmal	einfach	mittel	komplex	gesamt
Eigabe	... *3=...	... *4=...	... *6=...	...
Ausgabe	... *4=...	... *5=...	... *7=...	...
Daten	... *5=...	... *7=...	... *10=...	...
...				
...				
...				
ungewichtete Function Points (UFP)				...
gewichtete Function Points (FP=UFP*c)				...



Entwurf: - Anzahl der Knoten und Kanten eines Struktur- oder Steuergraphen

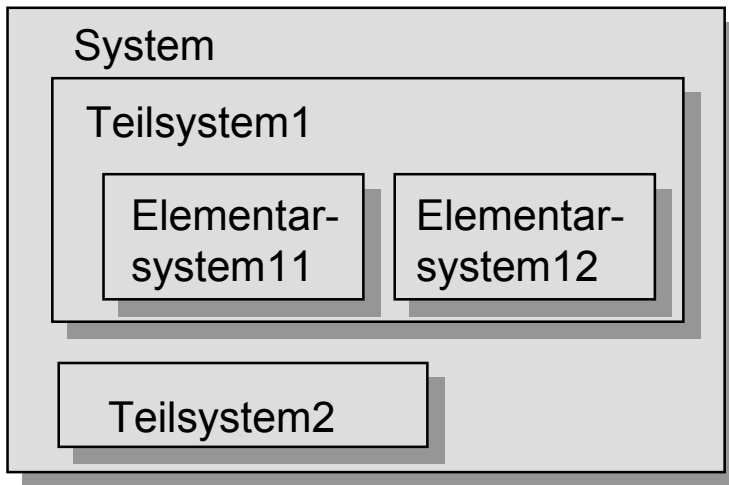
Implementation: - Anzahl der Quellcodezeilen eines Programmes
 - Speicher und Zeitbedarf eines Programmes

Folie 14.7

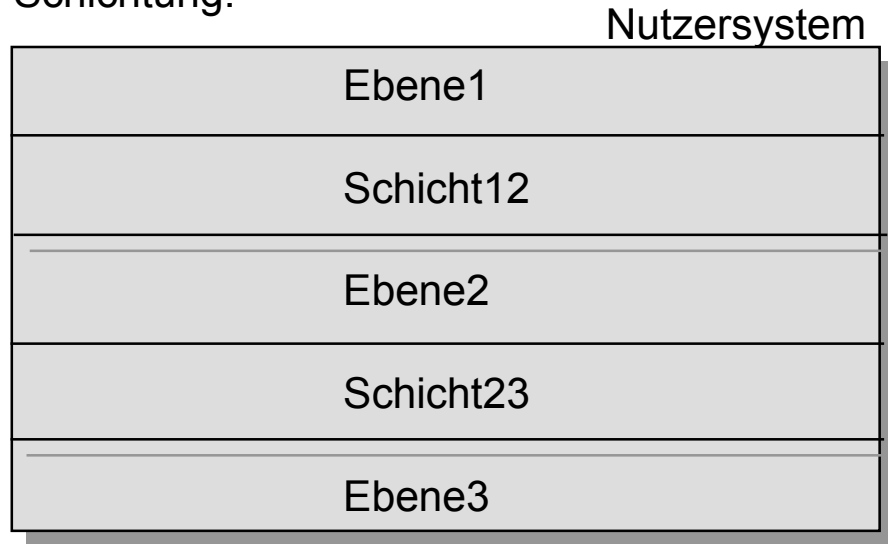
Architekturkonzept:

allgemeines Prinzip der Zerlegung eines Systems in Bausteine und der Festlegung von Beziehungen zwischen diesen

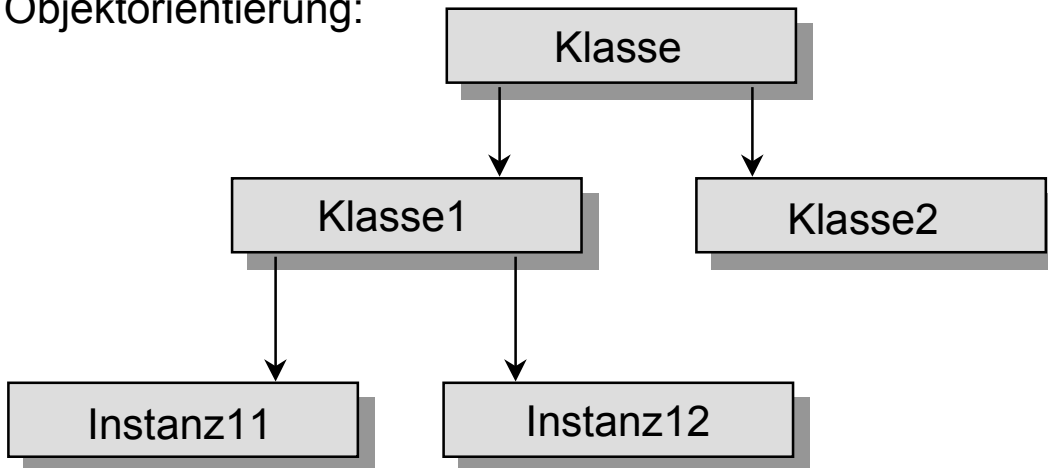
Dekomposition:



Schichtung:



Objektorientierung:



Basissystem

+ Kommunikation zwischen Bausteinen

Projektmanagement

Projekt: Vorhaben, das durch die Einmaligkeit des zu erreichenden Zieles, der existierenden Voraussetzungen, der Abgrenzung gegenüber anderen Vorhaben und der spezifischen Organisation gegeben ist

Projektmanagement: Gesamtheit von Führungsaufgaben, -organisationen, -techniken und -mitteln zur Abwicklung eines Projektes

Wichtige Komponenten:

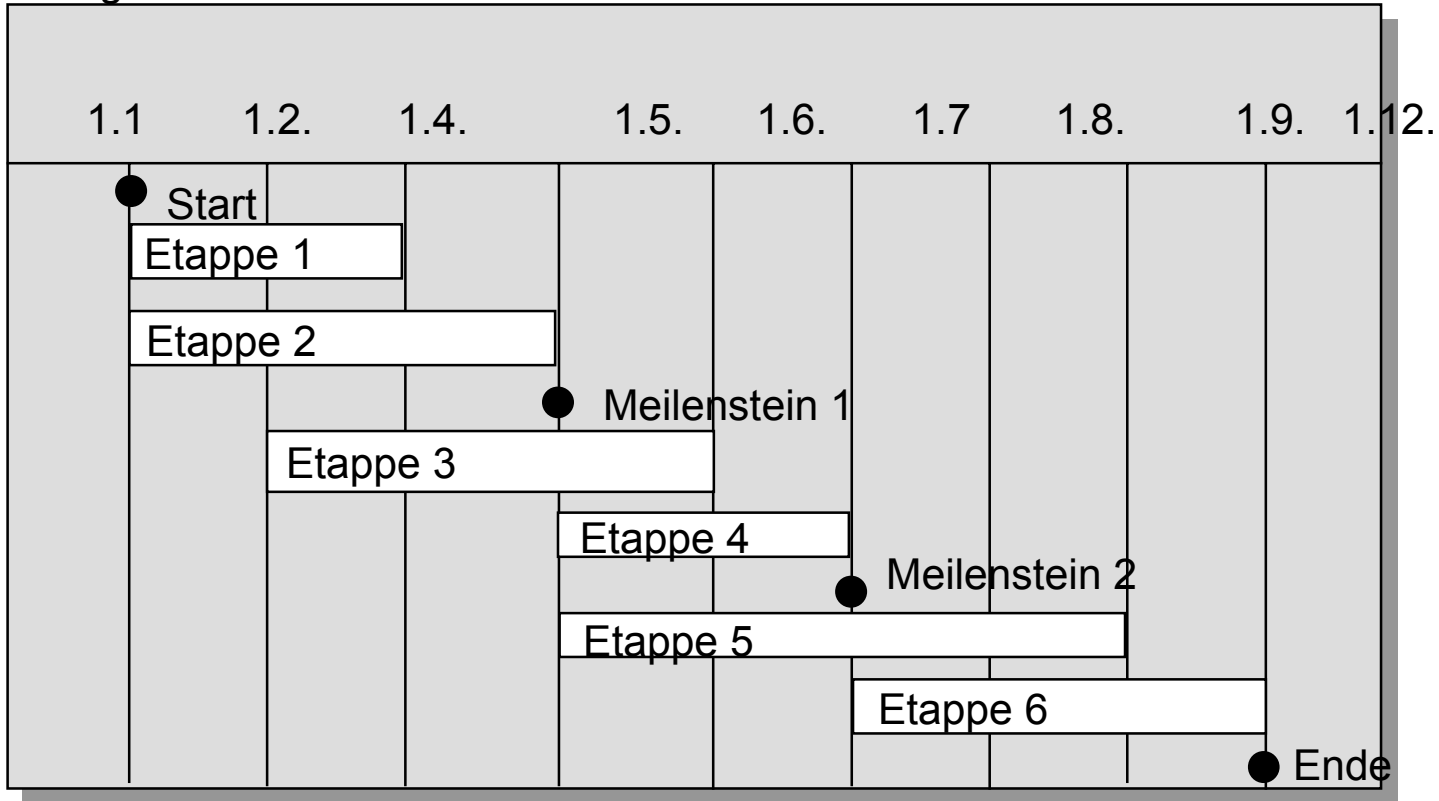
- Zielfestlegung (Projektantrag, Pflichtenheft)
- Planung von zeitlichen, personellen, technischen und finanziellen Ressourcen (Arbeitsplan, Organisations- und Leitungsstrukturen, Nutzung von Hardware und Software, Finanzierung)
- Überwachung des Verlaufes (Verbrauch von Ressourcen, Meilensteine, Berichterstattung, Kooperation)
- Projektverwaltung (Verwaltung von Aufgaben, Ergebnissen, Werkzeugen, Hilfsmitteln, Versionen)
- Übergabe an Auftraggeber (Systembereitstellung, Abschlußbericht)

Vorgehensmodell: Richtlinie für das Projektmanagement, die die wesentlichen Arbeitsetappen mit Inhalten, Abhängigkeiten und zeitlichen Reihenfolgen festlegt

- typische Modelle:**
- Wasserfallmodell
 - Prototyping
 - Spiralmodell
 - Reverse Engineering
- Darstellungsmittel:**
- Gantt-Diagramm
 - PERT-Chart

Folie 14.9

Gantt-Diagramm:



PERT-Chart:

