

Überblick

In diesem Teil wird in die Thematik Media Information Systems eingeführt und die Vorlesung, sowie Begleitmaterial beschrieben.

Inhalt

1	Struktur der Vorlesung.....	3
2	Literatur.....	4
3	Eingesetzte Software.....	5
4	Informationssysteme	6
4.1	Anforderungen.....	8
4.2	Systemschichten.....	9
4.3	Datenhaltung.....	10
4.4	Anforderungen an DBMS.....	11
4.5	Geschichte von DBMS.....	12
5	Softwareentwicklung.....	15

5.1	Phasen der Softwareentwicklung.....	15
5.2	Qualität von Software	16
5.3	Methoden des Entwurfs.....	18
5.4	Programmablauf.....	21
6	Datenbankentwurf.....	30
6.1	Logisches Datenmodell	31
6.2	Physisches Datenmodell	31
6.3	Implementierung.....	32
7	Client/Server Architekturen.....	33

1 Struktur der Vorlesung

Die Veranstaltung hat zum Ziel, in den **Entwurf von Informationssystemen** einzuführen. Dies wird aus dem **Blickwinkel** eines **Software-Entwicklers** erfolgen, der **Web-basierte Anwendungen** entwickeln soll, die auf Backend-Systeme zugreifen. In den Backend-Systemen sind die Daten in **Datenbanken** abgelegt.

Zunächst wird auf die **Struktur von Informationssystemen** eingegangen, dann wird in die Thematik der **Datenbanken** eingeführt. Zur Entwicklung Web-basierter Anwendungen die Programmierung in **PHP** betrachtet. Den Abschluss bildet die „Mechanik“ des Zugriffs auf Datenbanken in PHP.

- 1.Informationssysteme
- 2.Datenbanken
- 3.PHP
- 4.Datenbankzugriff aus PHP
- 5.PHP und HTML-Formulare
- 6.JavaScript
- 7.Ajax

2 Literatur

Datenbanken

- Peter C. Lockemann; Architektur von Datenbanksystemen; dpunkt.verlag
- Can Türker; SQL:1999 & SQL:2003; dpunkt.verlag
- Links auf meiner Web-Seite der Veranstaltung

PHP

- Links auf meiner Web-Seite der Veranstaltung

JavaScript

- <http://www.fbi.fh-darmstadt.de/~schuette/Vorlesungen/JavaScript/home.htm>

3 Eingesetzte Software

Alle Software, die vorausgesetzt wird, ist frei verfügbar und lauffähig unter Linux, Windows und Mac OS.

Als Software wird zunächst ein **Datenbanksystem** verwendet. Wir nehmen **mySQL**.

<http://dev.mysql.com>

Als **Webserver** wird **Apache** vorausgesetzt.

<http://www.apache.de>

PHP als Programmiersystem wird im Zusammenspiel mit Apache verwendet.

<http://www.php.net>

Neben dieser obligatorischen Software sind Administrationswerkzeuge für mySQL und eine PHP-Entwicklungsumgebung sinnvoll.

- **phpMyAdmin**

Ab der nächsten Vorlesung sollten alle, die ein Notebook haben, WAMP, LAMP oder MAMP installiert haben und mit in die Veranstaltung bringen!

4 Informationssysteme

Informationssysteme kann man aus mehreren Blickwinkeln betrachten.

Technische Betrachtungsweise (vgl. Duden):

Ein Informationssystem ist ein System zur

- Speicherung,
- Wiedergewinnung,
- Verknüpfung und
- Auswertung von Informationen.

Ein Informationssystem besteht aus

- einer Datenverarbeitungsanlage,
- einem Datenbanksystem und
- den Auswertungsprogrammen.

Betrachtung als Bestandteil eines **soziotechnischen Systems**, welches alle informationsverarbeitenden Prozesse sowie die an ihnen beteiligten menschlichen und maschinellen Akteure in ihrer informationsverarbeitenden Rolle umfasst:

Ein Informationssystem besteht aus

- Menschen und
- Maschinen,

die Informationen

- erzeugen und/oder
- benutzen und

die durch Kommunikationsbeziehungen miteinander verbunden sind.

Beschreiben Sie das Informationssystem Ihres Projektes!

4.1 Anforderungen

Mehrere Anforderungen sind an Informationssysteme zu stellen.

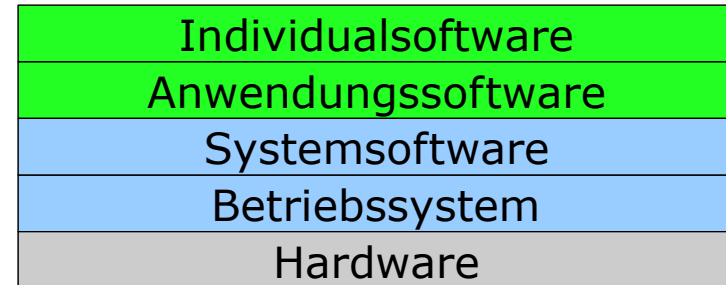
- **Korrektheit**
Fehlerfreie Lösung des vorgegebenen Problems
- **Effizienz**
Schnelle und ressourcensparende Bearbeitung der Aufgaben
- **Zuverlässigkeit/Robustheit**
Funktionsfähigkeit (auch) in außergewöhnlichen Situationen, wie z.B. bei fehlerhaften Eingaben des Benutzers
- **Benutzungsfreundlichkeit** und Ergonomie
Komfortable und benutzergerechte Bedienbarkeit
- **Wartbarkeit**
Einfache Fehlerbehebung, Anpassbarkeit an andere Umgebungen sowie Verbesserung und Erweiterung der bestehenden Funktionalität

Welche dieser Anforderungen trifft auf die Anwendung Ihres Projekts zu?

4.2 Systemschichten

Man unterteilt Software in Schichten

- Betriebssystem
 - Ausführung von Benutzerprogrammen
 - Verteilung der Betriebsmittel
 - Aufrechterhaltung der Betriebsart
- Systemsoftware
 - Administration des Betriebssystems
 - Standard-Dienstprogramme
- Basis-Software
 - Standard-Anwendungen
- Anwendungssoftware
 - Software zur Lösung von anwendungsspezifischen Problemen
- Individualsoftware
 - Spezialsoftware
 - Software zur einmaligen Benutzung



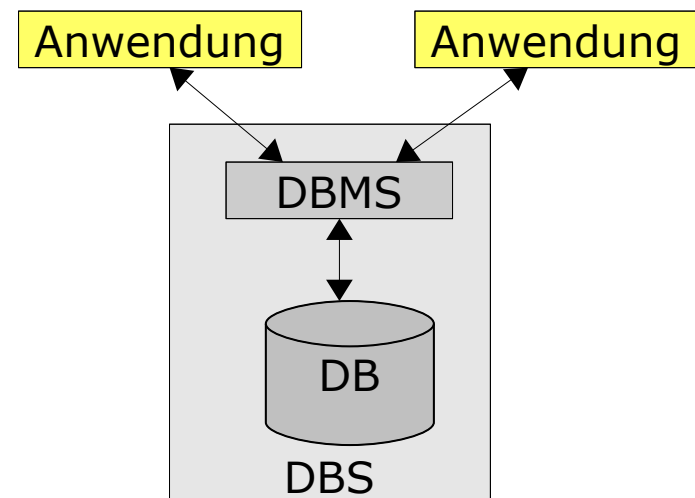
Nennen Sie jeweils Beispiele!

Welche Art von Software entwickeln Sie in Ihrem Projekt?

4.3 Datenhaltung

In jeder Schicht müssen **Daten persistent gespeichert** werden. Dazu sind unterschiedliche Methoden verfügbar. Hier interessiert uns hauptsächlich die Speicherung von Daten in **Datenbanksystemen** und nebenbei die Speicherung in Dateien des Dateisystems des zugrunde liegenden Betriebssystems.

- Eine **Datenbank** (DB) ist ein System zur Beschreibung, Speicherung und Wiedergewinnung von umfangreichen Datenmengen, die von mehreren Anwendungsprogrammen genutzt werden.
- Ein **Datenbank-Management-System** (DBMS) ist Gesamtheit der Software-Module, die die Verwaltung einer Datenbank übernehmen.
- Ein **Datenbanksystem** (DBS) ist die Kombination eines DBMS mit einer Datenbank.



Welche Datenbanksysteme kennen Sie?

4.4 Anforderungen an DBMS

In den Codd'schen Regeln sind die grundlegenden **Anforderungen** an ein DBMS zusammengefasst:

- **Integration**
einheitliche Verwaltung aller Daten
- **Operationen**
Datenspeicherung, Suchen, Ändern
- **Katalog**
Datenbeschreibungen der DB
- **Benutzersichten**
spezielle Ansichten (Auswahl, Struktur) auf Daten je nach Anwendung
- **Konsistenzüberwachung/Integritätssicherung**
Gewährleistung der Korrektheit von DB-Inhalten und korrekte Ausführung von Änderungen
- **Zugriffskontrolle**
Ausschluss unauthorisierter Zugriffe
- **Transaktionen**
Zusammenfassung von Operationen zu einer Einheit
- **Synchronisation**
Synchronisation von konkurrierenden Transaktionen

- **Datensicherung**
Wiederherstellung von Daten

4.5 Geschichte von DBMS

Datenbanken der **1. Generation**

- Erstmals **Trennung** zwischen **Programmier-und Datenbank-Datenmodell**
- Hierarchisches Datenmodell
 - seit Anfang der 70er Jahre im Einsatz
 - baumstrukturierte Daten, Vertreter: IMS
- Netzwerkmodell (CODASYL-Standard)
 - seit Mitte der 70er Jahre im Einsatz
 - Verallgemeinertes hierarchisches Modell
- Komplizierte Datenmodelle
- **Anfragen** müssen über **Datenbank-Guru** abgewickelt werden

Datenbanken der **2. Generation**

- **Relationales** Datenmodell
 - Theoretische Grundlage 1970
 - Erste Systeme Ende der 70er Jahre
 - Verbreitung seit Mitte der 80er Jahre
 - Vertreter: Oracle, Informix, **mySQL**
- Vorteile
 - Einfach zu begreifendes Datenmodell
 - Mathematische Grundlage
 - Hoher Standardisierungsgrad (SQL-92)
 - Erstmals verhältnismäßig einfache Ad-Hoc-Anfragen eines Nicht-Gurus möglich

Wir werden diese Art der DBS in der Vorlesung behandeln und im Projekt verwenden.

Datenbanken der **3. Generation**

- **Objektorientierung** in zwei Ausprägungen
 - Rein Objektorientiertes Datenmodell
 - Revolutionärer Ansatz
 - Quasi-Standard: ODMG
 - Vertreter: ObjectStore, O2
 - Objekt-relationales Datenmodell
 - Evolutionärer Ansatz (relationales Modell)
 - Standard: SQL-99
 - Vertreter: Oracle9 und Nachfolger

Tendenz in Unternehmen, die Datenbanksysteme seit Jahren im Einsatz haben:

langsame Migration von relationalen Datenbanken zu Objekt-relationalen Datenbanken mit XML Unterstützung

5 Softwareentwicklung

Die Entwicklung von Software geschieht heute ingenieurmäßiges. Dabei werden im Informatik-Bereich „Softwareengineering“ Methoden und Techniken vermittelt.

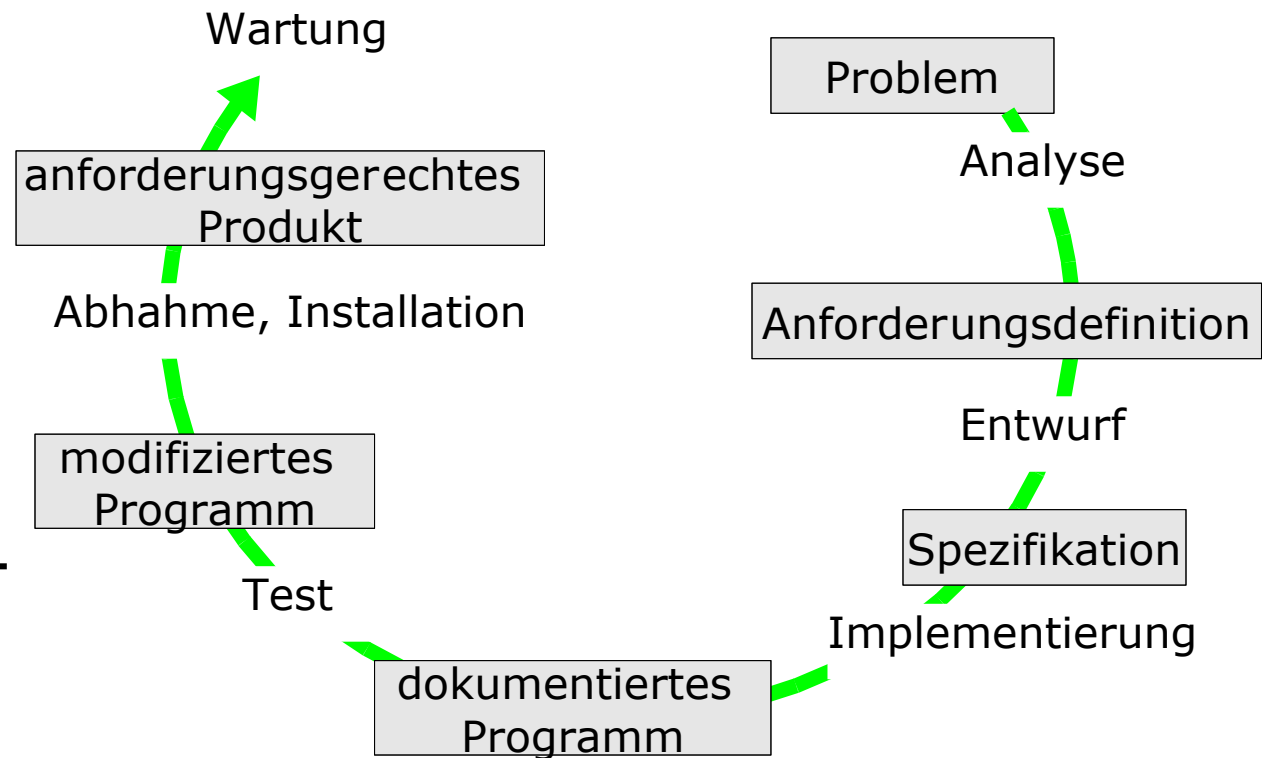
Hier sollen nur einige Aspekte beleuchtet werden.

5.1 Phasen der Softwareentwicklung

Software in **Phasen** entwickelt:

- **Analyse**
- **Entwurf und Konzeption**
- **Implementierung**
- **Test**
- **Einsatz und Wartung**

Jede Phase hat **spezifische Aktionen** und mündet in einer **Dokumentation**.



Für jede Phase existieren **Methoden**, die sicherstellen sollen, dass die produzierte Software eine hohe Qualität besitzt.

5.2 Qualität von Software

Software muss den Anforderungen der Kunden und denen der Hersteller entsprechen.

Äußere Qualitätskriterien beeinflussen die Ausführung eines Programms (Kundenanforderungen):

- Korrektheit,
 - Ein Programm ist korrekt, wenn es ein vorgegebenes Problem fehlerfrei gelöst ist.
 - Ab einer genügend großen Komplexität ist es nicht entscheidbar, ob ein Programm 100% korrekt ist.
 - Die Wahrscheinlichkeit, bei einer Fehlerbehebung in einem Programm einen neuen Fehler einzubauen, liegt bei ca. 1/3! Daher ist davon auszugehen, dass Software nicht völlig korrekt ist
- Robustheit,
 - Robuste Programme stürzen nie ab.
 - Geeignete Fehlerbehandlung für jeden möglichen Fehler, z.B. sinnvolle Reaktionen auf falsche Eingaben.

- Zuverlässigkeit,
 - Fehler treten nur selten auf und haben nur geringe Auswirkungen.
- Benutzungsfreundlichkeit und
 - Anwendung erfüllen software-ergonomische Kriterien.
- Effizienz
 - Laufzeit
 - Speicherplatz

Innere Qualitätskriterien beeinflussen Fehlersuche und Änderung der Software (Herstelleranforderungen):

- Wartbarkeit,
 - Suchen und Beheben von Fehlern
 - Portierung auf andere Betriebssysteme
 - Verbesserung und Anpassung
- Modularität und
- Dokumentation

- Systemdokumentation
- Benutzerdokumentation
- Installation und Administration

5.3 Methoden des Entwurfs

Bei der Software-Entwicklung spielt die bewusste Anwendung von Prinzipien, Methoden und Darstellungsmitteln eine wesentliche Rolle, z.B.

- Methode der schrittweisen Verfeinerung (Stepwise Refinement)
- Top-down-Methode (Abstraktion nach unten)
- Bottom-up-Methode (Abstraktion nach oben)

Modularisierung und Objektorientierung sollen helfen, mit Komplexität fertig zu werden.

Modularisierung

- Beschreibung von Algorithmen in immer feineren Bausteinen (Module)
- Zusammenfügen der Module über Schnittstellen
- Standardschnittstellen erlauben den Austausch von Modulen
- Realisierung von Software in Modulen kann unabhängig voneinander in Teams vorgenommen werden. Das führt zur Erhöhung der Produktivität
- Module sind wiederverwendbar
- Jedes Modul kann einzeln auf Korrektheit getestet werden
- Datenkapselung in Modulen erhöht Sicherheit und Zuverlässigkeit

Modularisierung wird in Ihrem Projekt wichtig sein, damit unterschiedliche Teammitglieder parallel arbeiten können.

Objektorientierung

- "Ganzheitliche Sicht" in der Softwareentwicklung
- Repräsentation von Konzepten der realen Welt durch Objekte (Status und Verhalten)
- Klassen als Abstraktion von Objekten mit gleichen Attributen und Operationen (Methoden)
- Objekte solcher Klassen (Instanzen) können zur Laufzeit erzeugt werden
- Objekte kommunizieren über Messages
- Vererbung: Neue Klassen können vorhandene Klassen erweitern und adaptieren, wobei vorhandene Methoden und Attribute automatisch übernommen werden

Objektorientierung wird in Ihrem Projekt verwendet werden, um mit der Komplexität fertig zu werden.

5.4 Programmablauf

Wenn die Analyse beendet ist und die Struktur des Softwaresystems festgelegt ist sollten die Programmabläufe visuell dokumentiert werden. Das hilft vor der Programmierung, sich über das Programm unterhalten zu können und dient der Dokumentation.

Darstellungsmittel für Programmabläufe:

- Programmablaufpläne
- **Datenflussdiagramme**
- **Struktogramme**
 - sind graphische Darstellungsmittel für Algorithmen
 - unterstützen die strukturierte Programmierung
- Beschreibung der Beziehungen zwischen den Teilproblemen (Schnittstellen)
- Pseudocode
- Entscheidungstabellen

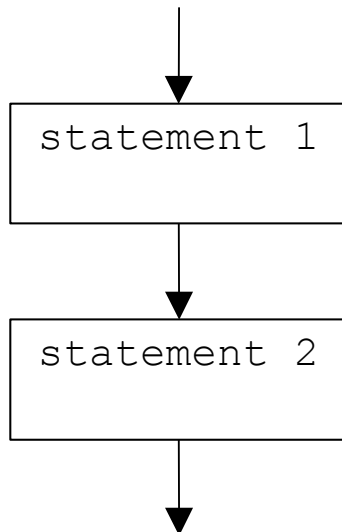
Sequenz

Die Anweisungen eines Programms werden in Aufschreibungsreihenfolge durchlaufen.

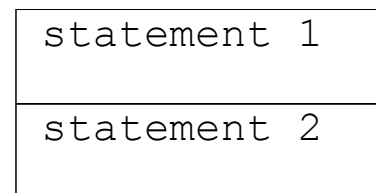
```
statement1;  
statement2;
```

Zuerst wird `statement1` ausgeführt, danach `statement2`.

Flussdiagramm



Struktogramm



Entscheidungen

Soll eine Aktion nur ausgeführt werden, wenn gewisse Bedingungen zutreffen, also Entscheidungen programmiert werden sollen, können Verzweigungen verwendet werden.

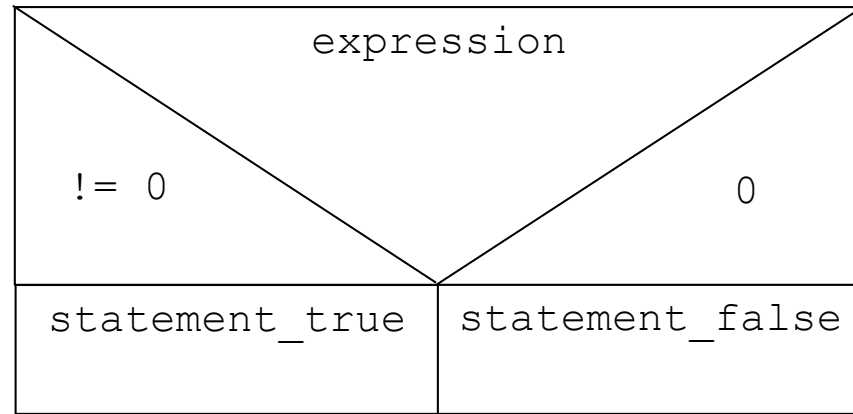
```
if (expression)
    statement
```

expression wird ausgewertet. Ist der Wert `true`, wird `statement` ausgeführt.

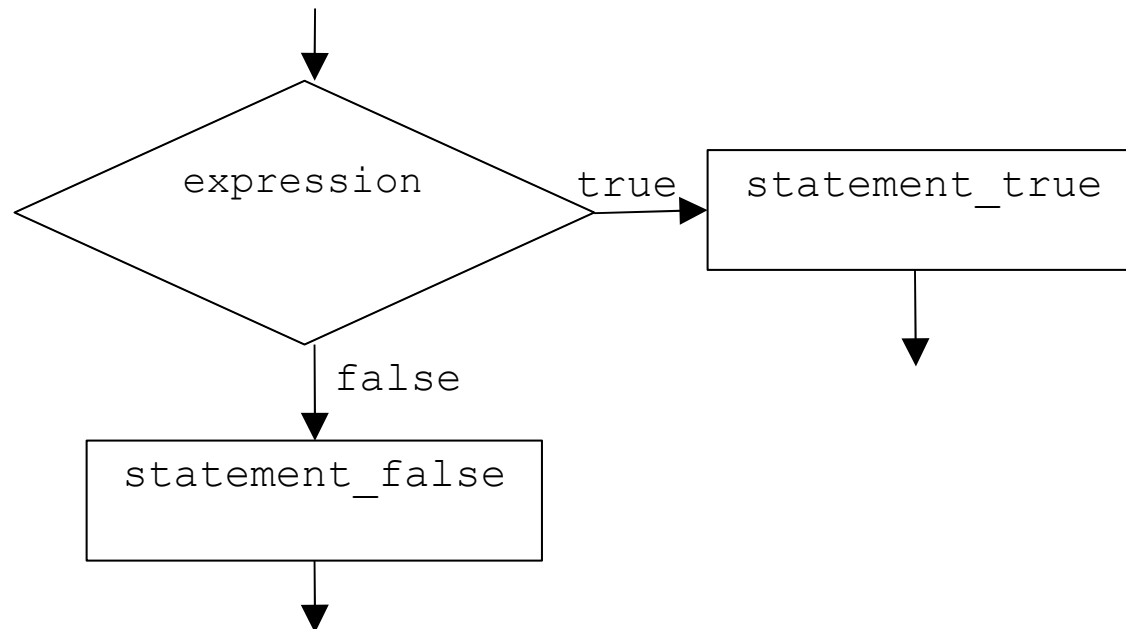
```
if (expression)
    statement_true
else
    statement_false
```

expression wird ausgewertet. Ist der Wert `true`, wird `statement_true` ausgeführt, **ansonsten** `statement_false`.

Struktogramm



Flussdiagramm



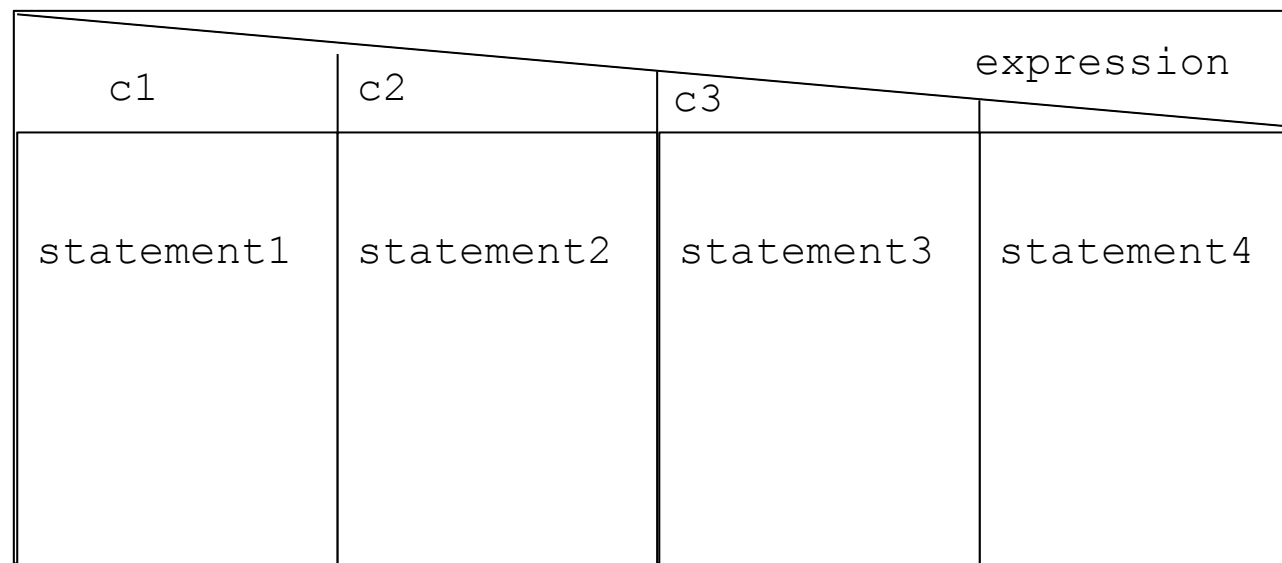
Mehrfachentscheidungen

```
switch ( expression ) {  
    case const1: statements1  
    case const2: statements2  
    default: statements3  
}
```

`expression` wird ausgewertet und zu der Stelle `consti` gesprungen, die mit dem `expression`-Wert übereinstimmt, bzw. zu `default`, wenn es keine Übereinstimmung gibt.

Bei reinen Fallunterscheidungen ist die letzte Anweisung von `statement` eine `break`-Anweisung, die bewirkt, dass das `switch`-Statement terminiert.

Struktogramm:



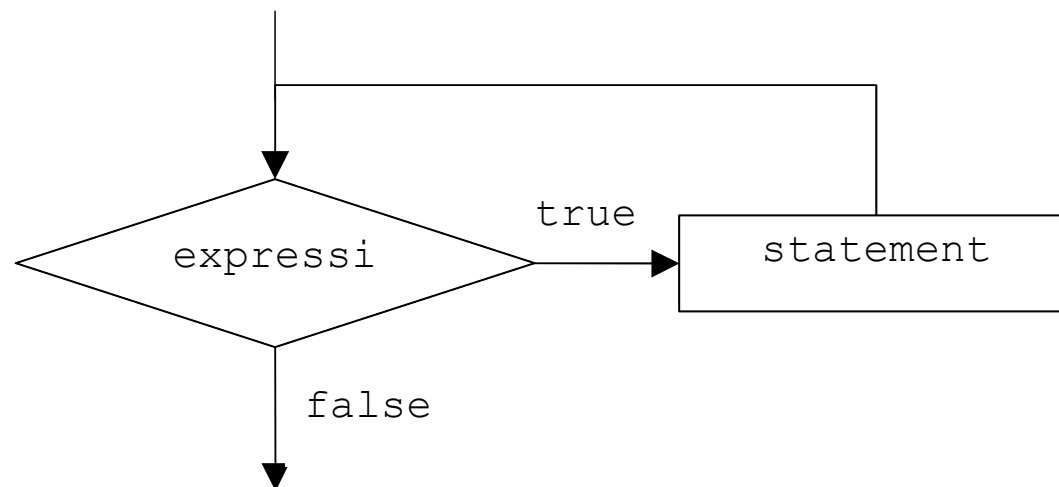
Schleifen

```
while ( expression )  
    statement
```

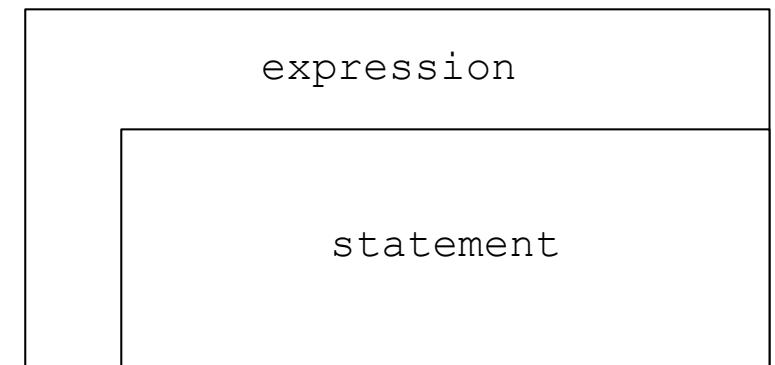
Zuerst wird die Bedingung *expression* geprüft; ist sie zu `true` auswertbar, wird *statement* ausgeführt. Dann wird *expression* wieder geprüft.

Also wird *statement* solange ausgeführt, bis *expression* `false` ist.

Flussdiagramm

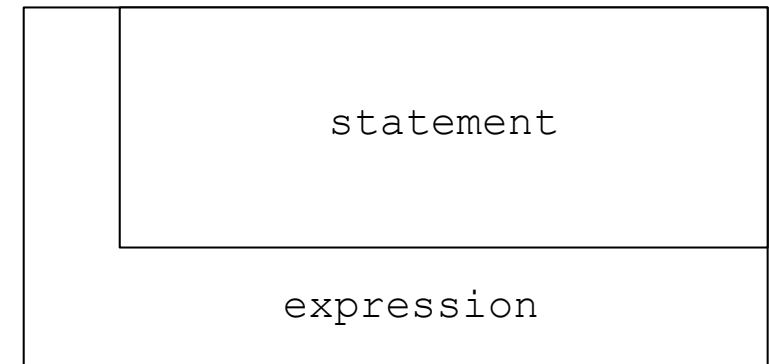
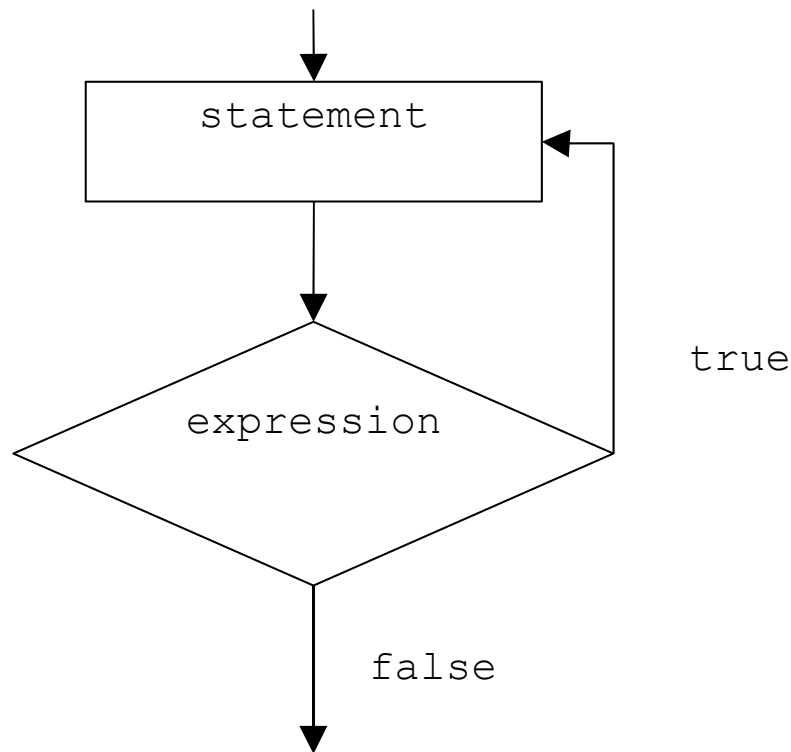


Struktogramm



```
do  
  statement  
while (expression)
```

Zuerst wird *statement* ausgeführt. Dann wird die Bedingung *expression* geprüft; ist sie zu `true` auswertbar dann wird *statement* wieder ausgeführt.



Beispiel (von Aufgabe über Flussdiagramm zum Programm)

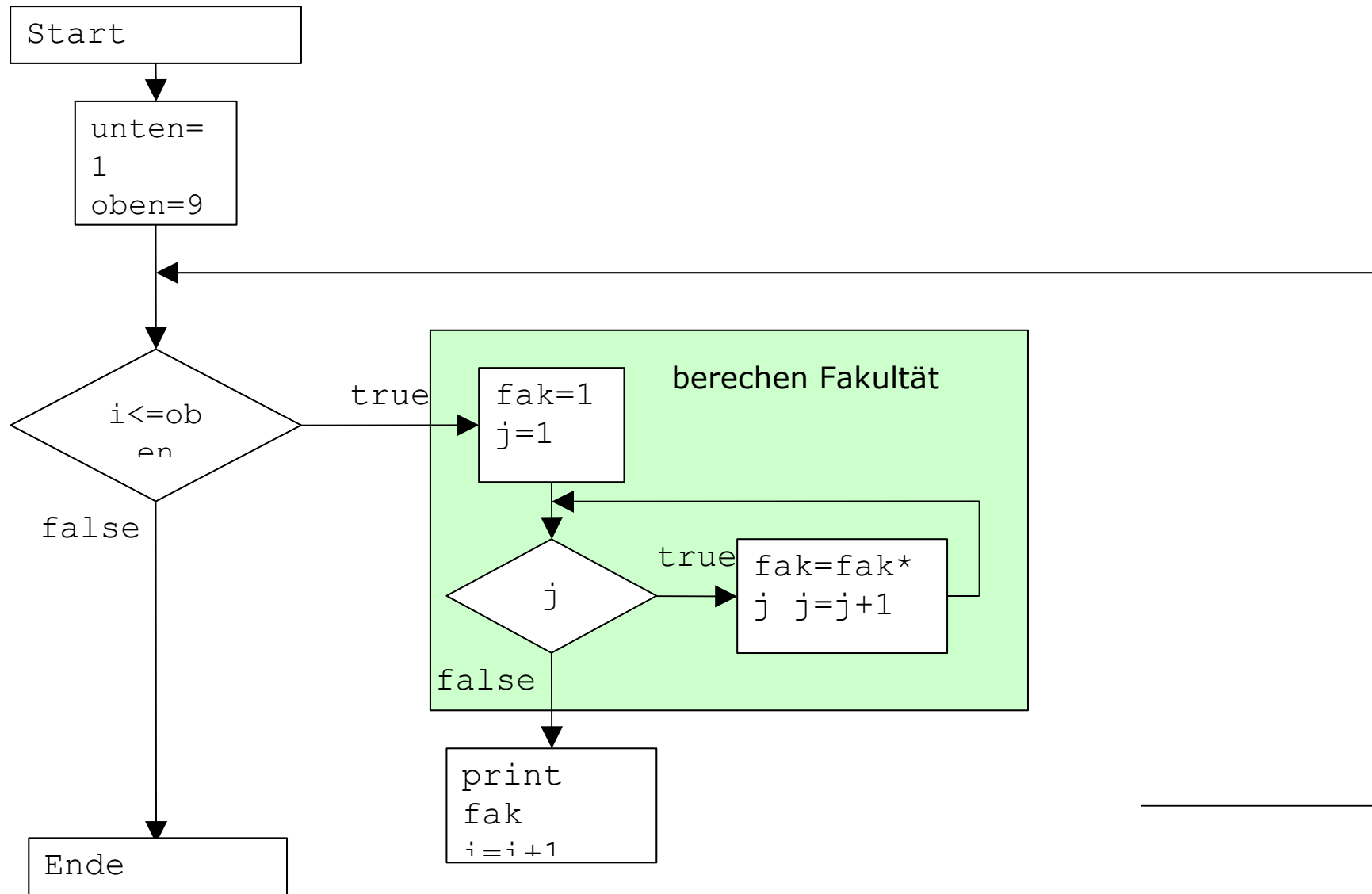
Aufgabe:

Von dem Zahlenbereich 1 bis 9 soll für jede Zahl die Fakultät berechnet und ausgegeben werden.

Verfahren:

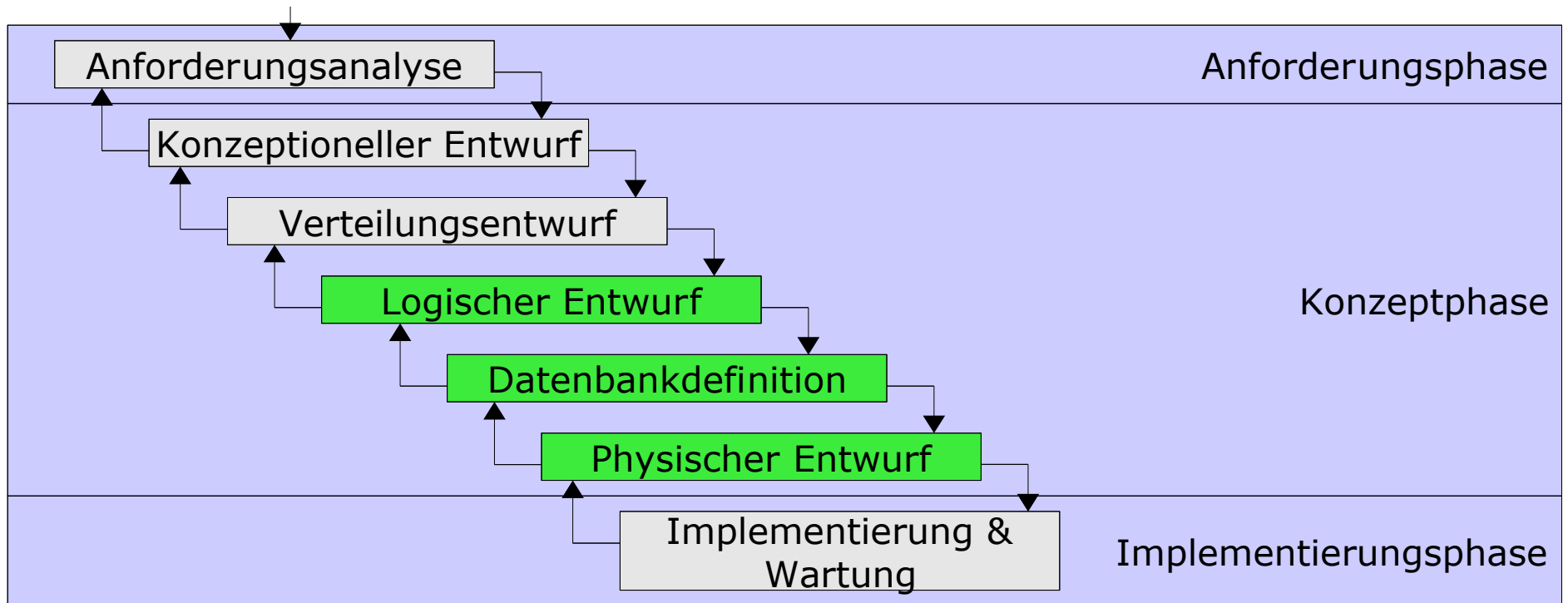
Ausgabe von $n!$ für n zwischen 1 und 9 wobei $n! := 1*2*3* \dots *n$.

Flussdiagramm:



6 Datenbankentwurf

Der Entwurf von (Datenbank-) Anwendungen startet mit einer Anforderungsanalyse und durchläuft mehrerer **Phase** (nach Heuer), bis letztlich die Anwendung produktiv betrieben werden kann:



In der **Anforderungsanalyse** werden die **Aspekte der realen Welt** erarbeitet. Der **Datenbankentwurf** findet in der **Konzeptphase** statt und hat als Ergebnis ein **physisches Datenmodell**.

6.1 Logisches Datenmodell

Der **Ausgangspunkt** ist das Ergebnis der Anforderungsanalyse, in dem die reale Welt auf eine „Miniwelt“ reduziert wird.

Dabei wird mittels **Techniken**, wie

- **ER-Modellierung**
- UML-Modellierung

als **Ergebnis** ein **logisches Datenmodell** entwickelt.

Dies findet in der Analysephase Ihres Projekte statt!

6.2 Physisches Datenmodell

Der **Ausgangspunkt** ist das **logische Datenmodell**.

Dabei wird mittels **Techniken**, wie

- **Relationale Datenmodellierung**

ein physisches Datenmodell entwickelt.

Das **Ergebnis** sind **SQL-Anweisungen**, die auf ein konkretes Datenbanksystem umsetzbar sind.

Dies findet in der Konzeptphase Ihres Projekte statt!

6.3 Implementierung

Weitere Schritte in der Praxis sind dann in der Implementierungsphase:

- Auswahl, Kauf, Installation, Konfiguration eines DBMS
- Anlegen der Datenbank
- u.U. Migration von Altdaten
- Administration
- Leistungsoptimierung
- Umsetzung von Sicherheitsaspekte
- Anwendungsentwicklung
- Backup und Recovery Maßnahmen
- Going Live

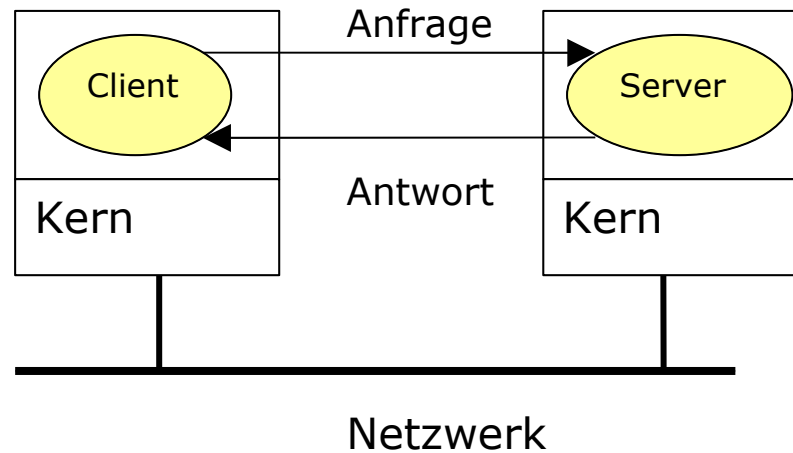
Dies findet in der Realisierungsphase Ihres Projekte statt!

7 Client/Server Architekturen

Heute werden häufig Anwendungen als Client/Server-Systeme realisiert.

Dabei wird meist ein **Anfrage/Antwortprotokoll** verwendet:

Ein Client-Prozess sendet eine Anfragenachricht, in der er einen bestimmter Dienst nachfragt an einen Server. Der Server erfüllt den Dienst, in dem er die nachgefragten Daten (oder eine Fehlermeldung) zurück zum Client sendet.



Client/Server Architekturen

Dabei werden unterschiedliche Architekturen:

- 2-Tier
- 3-Tier

und Protokolle verwendet:

- Sockets
- RMI
- CORBA
- Webservice-Protokolle

Die ist Gegenstand von Veranstaltungen wie „Verteilte Systeme“.

Wir betrachten eine 3-Tier Architektur, wie sie heute Standard in allen großen Unternehmen ist:

