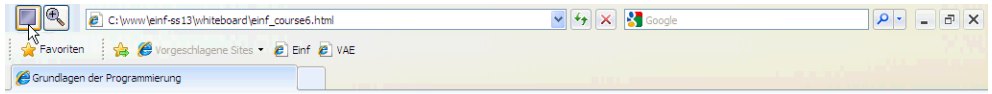
Script **generated by TTT**

Title: Einf_HF (10.06.2013)

Date: Mon Jun 10 14:14:30 CEST 2013

Duration: 94:11 min

Pages: 34



"Kunst des Programmierens". Grundlagen zu Datenstrukturen, Programmkonstrukte, Strukturierung von Programmen, objekt-orientierte Programmierung.

- Fragestellungen des Abschnitts:
  - Was ist ein Algorithmus?
  - Welche elementaren Datenstrukturen gibt es?
  - Was sind die grundlegenden Konstrukte einer Programmiersprache?
  - Was ist unter Objekt-orientierter Programmierung zu verstehen?
  - Was versteht man unter Modularisierung und Rekursion?

Einführung

Programmierung: Vorgang, in dem Anweisungen für den Computer erstellt werden. Aufgeschrieben in für Menschen verständlichen Sprachen, für Computer in Maschinensprache übersetzt (z:B. C) oder mit Interpreterprogramm interpretiert (z.B. Javascript).

Programm = Daten + Algorithmus

Algorithmus legt die Operationen fest, die auf den Daten ausgeführt werden.

Begriff des Algorithmus

Nach Spezifikation: Lösungsweg entwerfen. Da von Rechner auszuführen, jeden Schritt exakt vorschreiben. Algorithmus.

eine der ältesten Beschreibungstechniken für Abläufe; benannt nach dem Mathematiker Al-Khwarizmi (ca. 780-850), der am Hofe des Kalifen von Bagdad wirkte.

Informelle Charakterisierung eines Algorithmus

Detaillierte, explizite Vorschrift zur schrittweisen Lösung eines Problems, d.h. die

präzise formuliert,

in endlicher Form dargestellt und

effektiv ausführbar ist.

Die Aufgabe kann aus vielen Anwendungsgebieten stammen, z.B. Berechnung numerischer Werte, Text- und Bildverarbeitung, Handhabung von Objekten (Robotik), Zuteilung von Ressourcen, Steuerung von Geräten.

Formale Definition (nach H. Kübe):

Ein Algorithmus ist eine in der Beschreibung und Ausführung endliche, deterministische und effektive Vorschrift zur Lösung eines Problems, die effizient sein sollte.

Hierin bedeuten:

endlich in Beschreibung : der Algorithmus kann mit endlich viel Text komplett aufgeschrieben werden.

endlich in Ausführung : nach einer endlichen Zeit wird der Algorithmus beendet.

deterministisch : eindeutige Bestimmung des nächsten Schrittes.

effektiv : eindeutige Ausführbarkeit der Einzelschritte.

effizient : möglichst geringer Verbrauch an Ressourcen (z.B. Arbeitsspeicher).

Ein Algorithmus heißt

terminierend , wenn seine Ausführung für jede mögliche Eingabe nach einer endlichen Anzahl von Schritten ***endet*** .

determiniert , wenn er für eine bestimmte Eingabe bei allen Abläufen immer ***dieselbe Ausgabe*** liefert

Beispiel Personalführung

Beispiel Personalführung

Eingabe: Liste aller Mitarbeiter

wiederhole solange die Liste nicht leer ist

wiederhole für alle Mitarbeiter

führe willkürlich einen der folgenden Einzelschritte aus:

belobige den Mitarbeiter

tadle den Mitarbeiter

entlasse den Mitarbeiter (entferne ihn aus der Liste)

Ende Willkür

Ende Wiederhole

Ende Wiederhole

Ausgabe: Ursprüngliche Liste aller (auch ehemaliger) Mitarbeiter mit Lob- und Tadelvorgängen.

Algorithmus ist ***nichtdeterministisch*** , ***nichtterminierend*** , ***nichtdeterminiert.***

Formale Definition (nach H. Kübe):

Ein Algorithmus ist eine in der Beschreibung und Ausführung endliche, deterministische und effektive Vorschrift zur Lösung eines Problems, die effizient sein sollte.

Hierin bedeuten:

endlich in Beschreibung : der Algorithmus kann mit endlich viel Text komplett aufgeschrieben werden.

endlich in Ausführung : nach einer endlichen Zeit wird der Algorithmus beendet.

deterministisch : eindeutige Bestimmung des nächsten Schrittes.

effektiv : eindeutige Ausführbarkeit der Einzelschritte.

effizient : möglichst geringer Verbrauch an Ressourcen (z.B. Arbeitsspeicher).

Ein Algorithmus heißt

terminierend , wenn seine Ausführung für jede mögliche Eingabe nach einer endlichen Anzahl von Schritten ***endet*** .

determiniert , wenn er für eine bestimmte Eingabe bei allen Abläufen immer ***dieselbe Ausgabe*** liefert

Beispiel Personalführung

Generated by Targeteam

Formulierung eines Algorithmus

Sprache

In natürlicher oder formaler Sprache

Ausführung

Ausführung durch Menschen oder Machine. "Effektiv", wenn Folge von Bearbeitungsschritten, die der beabsichtigte Ausführende "beherrscht".

Beispiele für Algorithmen

Nicht maschinell ausführbares Beispiel

Maschinell ausführbares Beispiel

Nicht maschinell ausführbares Beispiel

Bestimme das Alter der ältesten Person im Raum

1. Gehe zur ersten Person
2. Frage Person nach dem Alter
3. Merke das Alter
4. Solange noch nicht alle Personen gefragt, wiederhole Schritte 4.a bis 4.c
   a) gehe zur nächsten Person
   b) frage nach dem Alter
   c) wenn das Alter größer als das gemerkte Alter, dann merke Dir das neue Alter
5. Älteste Person ist "gemerktes Alter" alt

Maschinell ausführbares Beispiel

Bestimme die größte Zahl aus einer Menge von Zahlen

1. Lies die erste Zahl
2. Initialisiere Platzhalter/Variable z mit der gelesenen Zahl
3. Lies die nächste Zahl
4. Wenn diese Zahl größer z, dann setze z auf diese Zahl
5. Wenn noch Zahlen vorhanden, dann gehe zu Schritt 3
6. Gib den Wert von z aus

Vorausgesetzt, dass Menge so organisiert, dass Maschine Zugriff auf "erste" und "nächste" Zahl ausführen kann.

Komponenten eines Algorithmus

Was wird verwendet, um Algorithmus zu formulieren?

Objekte und Anweisungen

In allgemeinster Form:

Objekte und

Anweisungen, die Operationen an Objekten realisieren.

Beispiele

Objekte sind z.B. Zahlen, Adressen, Textdokumente. Operationen sind z.B. Addition, Suche, Ausdrucken, Rechtschreibprüfung.

Operationen bringen Objekte aus Anfangszustand ggf. über Zwischenzustände in Endzustand. (EVA-Prinzip: Eingabe -> Verarbeitung -> Ausgabe)

Abstraktion

Bei Beschreibungen ist Abstraktion wichtig: Übergang von konkreten Gegebenheiten in allgemein gültige Strukturierung:

Abbildung der realen Objekte auf Daten, der Operationen auf vorgegebenen Vorrat von Anweisungen und Kontrollstrukturen (Zahlenmenge -> Liste mit Einfügen, Löschen, Sortieren, Suchen).

Abstraktion bedeutet: man konzentriert sich auf das Wesentliche; unwesentliche Anteile werden weggelassen.

Struktur eines Algorithmus

Blöcke

Operationen zu Blöcken zusammenfassen: strukturiertes / systematisches Programmieren. Block als komplexe Anweisung betrachten, Vorstufe zur Modularisierung. Je Block nur ein Eingang und ein Ausgang.

Struktur eines Algorithmus

Generelle Grundstrukturen für Operationen (wenn hinreichend verfeinert):

elementarer Einzelschritt : nicht weiter zerlegbare Verarbeitung, z.B. "schalte rotes Licht ein".

Sequenz : Folge von Einzelschritten (Anweisungen); Einzelschritte können zu Blöcken zusammengefasst werden. Beispiel

```
schalte rotes Licht ein;
warte eine Minute;
schalte gelbes Licht ein;
```

Alternative : über eine Bedingung gesteuerte Verzweigung,

```
falls Bedingung
    dann ja-Fall
    sonst nein-Fall

Beispiel
falls b ungleich 0 dann berechne a/b und gib Ergebnis aus;
    sonst melde Fehler
```

Auswahl (Selektion): Verallgemeinerung der Alternative,

Iteration : Wiederholung einer Menge von Einzelschritten.

```
wiederhole i = 1, .., n
    verarbeite aktuelles Konto Ki ;
```

Struktur eines Algorithmus

Sequenz : Folge von Einzelschritten (Anweisungen); Einzelschritte können zu Blöcken zusammengefasst werden. Beispiel

unabh. von Prog. sprache

```
schalte rotes Licht ein;
warte eine Minute;
schalte gelbes Licht ein;
```

Alternative : über eine Bedingung gesteuerte Verzweigung,

```
falls Bedingung
    dann ja-Fall
    sonst nein-Fall

Beispiel
falls b ungleich 0 dann berechne a/b und gib Ergebnis aus;
    sonst melde Fehler
```

Ergebnis der Bdg — ja / nein

Bedingung

ja-Fall

mein Fall

Auswahl (Selektion): Verallgemeinerung der Alternative,

Iteration : Wiederholung einer Menge von Einzelschritten.

```
wiederhole i = 1, .., n
    verarbeite aktuelles Konto Ki ;
```

Zahl mod 5

0, 1, 2, 3, 4

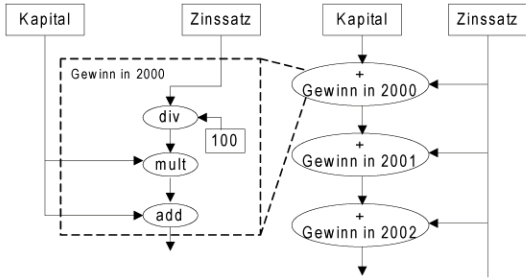
case Bdg :
0 :
1 :
2 :
3 :

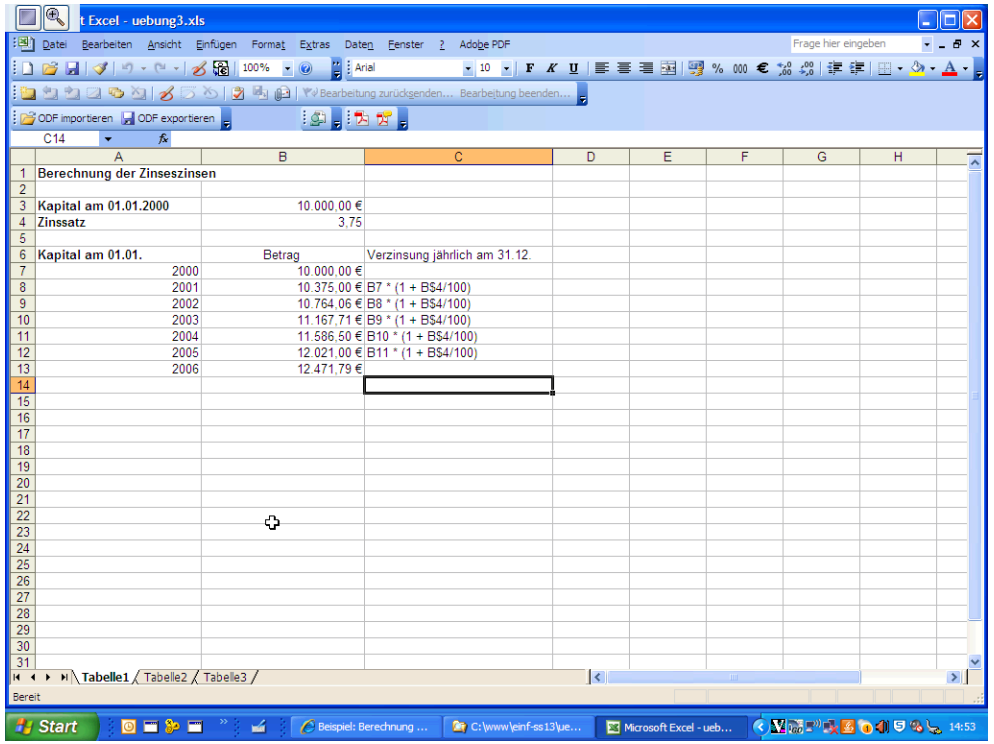
[Beispiel: Berechnung Zinseszinsen]

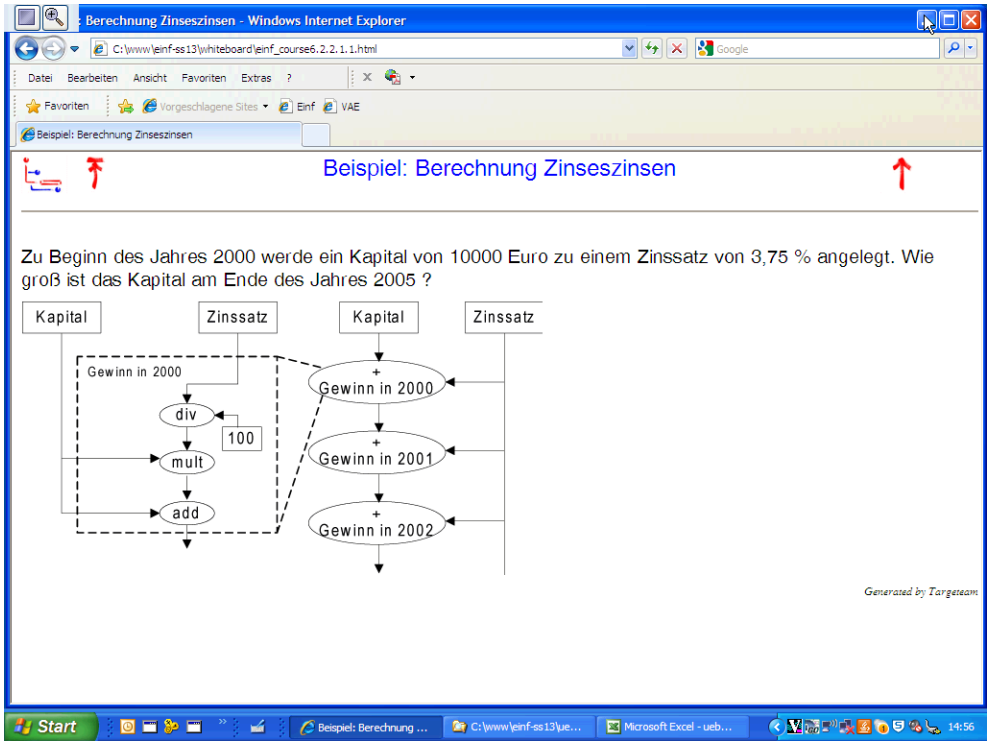
Mit diesen Strukturen können alle Algorithmen beschrieben werden.

Beispiel: Berechnung Zinseszinsen

Zu Beginn des Jahres 2000 werde ein Kapital von 10000 Euro zu einem Zinssatz von 3,75 % angelegt. Wie groß ist das Kapital am Ende des Jahres 2005 ?







Komponenten eines Algorithmus

Objekte und Anweisungen

In allgemeinster Form:

Objekte und

Anweisungen, die Operationen an Objekten realisieren.

Beispiele

Objekte sind z.B. Zahlen, Adressen, Textdokumente. Operationen sind z.B. Addition, Suche, Ausdrucken, Rechtschreibprüfung.

Operationen bringen Objekte aus Anfangszustand ggf. über Zwischenzustände in Endzustand. (EVA-Prinzip: Eingabe -> Verarbeitung -> Ausgabe)

Abstraktion

Bei Beschreibungen ist Abstraktion wichtig: Übergang von konkreten Gegebenheiten in allgemein gültige Strukturierung:

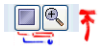
Abbildung der realen Objekte auf Daten, der Operationen auf vorgegebenen Vorrat von Anweisungen und Kontrollstrukturen (Zahlenmenge -> Liste mit Einfügen, Löschen, Sortieren, Suchen).

Abstraktion bedeutet: man konzentriert sich auf das Wesentliche; unwesentliche Anteile werden weggelassen.

Struktur eines Algorithmus

Blöcke

Operationen zu Blöcken zusammenfassen: strukturiertes / systematisches Programmieren. Block als komplexe Anweisung betrachten, Vorstufe zur Modularisierung. Je Block nur ein Eingang und ein Ausgang. Klammerung über spezielle Schlüsselwörter oder durch Klammerzeichen, z.B. {} in C oder Java.



Algorithmus

Vor Programm (Lösungsvorschrift) braucht man: Exakte Beschreibung des Problems.

Spezifikation als Ausgangspunkt für ein Programm

Vollständige, detaillierte, unzweideutige Problembeschreibung.

vollständig : alle relevanten Informationen sind berücksichtigt.

detailliert : alle Hilfsmittel und Grundaktionen sind aufgelistet, die zur Lösung zugelassen sind.

unzweideutig : klare Kriterien sind festgelegt, die bestimmen, wann eine Lösung akzeptabel ist.

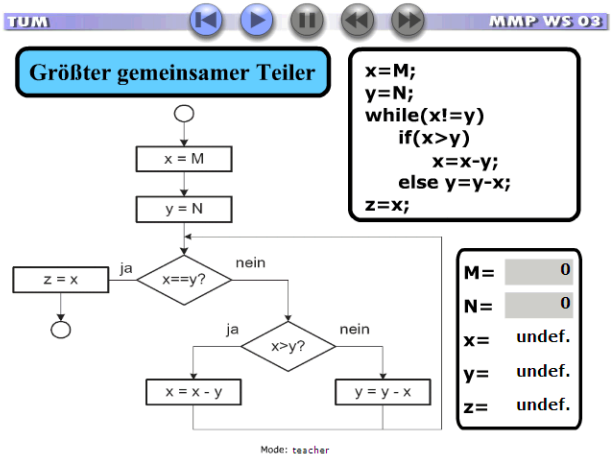
Begriff des Algorithmus

Komponenten eines Algorithmus

Darstellung von Algorithmen

Algorithmus und Programm

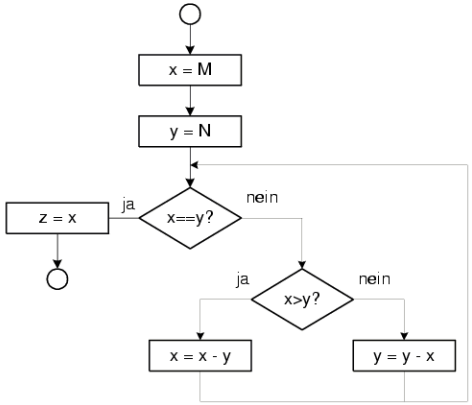
Animation größter gemeinsamer Teiler



Flussdiagramm

Beispiel: Größter gemeinsamer Teiler von M und N

(Euklid'scher Algorithmus)



x und y sind Variablen/Platzhalter.

Animation größter gemeinsamer Teiler

Darstellung von Algorithmen

Unterschiedliche Darstellungsmethoden: informell textuell, programmiersprachlich, graphisch (z.B. Flussdiagramme oder Nassi-Shneiderman).

Flussdiagramm

Programmiersprachliche Darstellung

```
x = M;
y = N;
while (x != y)
   if (x > y) x = x - y;
   else y = y - x;
z = x;
```

"x != y" bedeutet hier die Bedingung "x ungleich y".

Algorithmus und Programm

Programme sind spezielle Darstellungsformen für Algorithmen.

Zusammenhang

Programm: Formulierung eines Algorithmus in Sprache, die Computer versteht.

Aus Sicht der "Maschine": endliche Folge von ausführbaren Anweisungen.

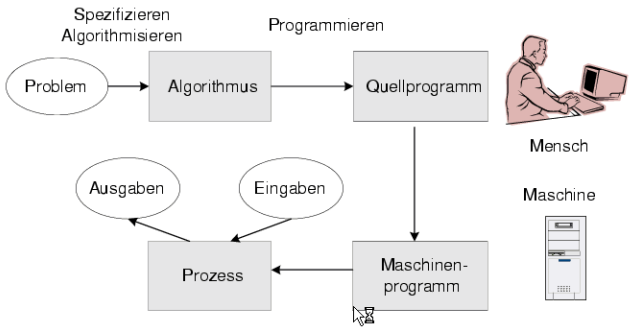
Damit Automatisierung des Algorithmus.

Programmiersprachen sind ein Hilfsmittel zur Abbildung der Daten- und Kontrollstrukturen des Algorithmus in die Sprachkonstrukte des betreffenden Computers (Maschinensprache, Arbeitsspeicher).

Ausführbare Programme: Abbildung der Daten- und Kontrollstrukturen in Interndarstellung des Computers (Objektcode). Verwendet Maschinensprache (abhängig vom jeweiligen Hersteller des Prozessors).

Zeitlicher Zusammenhang zwischen Algorithmus und Programm

Zeitlicher Zusammenhang zwischen Algorithmus und Programm



Grundkonstrukte in allen Programmiersprachen.

Informationen auf Rechner: Daten. Umsetzung Daten in Information: Interpretation.

Allgemeine Eigenschaften von Daten

1. Basisdaten: Zeichen, Wahrheitswerte (true, false), Zahlen (natürliche Zahlen, ganze Zahlen, Gleitkommazahlen).
2. Daten anordenbar, in Beziehung setzbar (z.B. Bäume, Listen)
3. Algorithmen hängen von gewählter Datenstruktur ab (Übersichtlichkeit, Effizienz).

Elementare Datentypen

Ausdrücke

Datentyp: Zusammenfassung von Werten gleicher Art (z.B. ganze Zahlen, Gleitkommazahlen, Zeichen).

Basis-Datentypen

Die Verwendung der Schlüsselwörter ("keywords") sind abhängig von der jeweiligen Programmiersprache.

`int` : Ausschnitt der ganzen Zahlen (im Rechner darstellbare ganze Zahlen)

`float` : Menge der Gleitkommazahlen

`double` : Menge der Gleitkommazahlen mit doppelter Genauigkeit

`char` : Menge der Zeichen

`boolean` : Werte `true` und `false`

Datentyp Verbund (struct)

Zusammenfassung von zusammengehörigen Daten unterschiedlicher Typen.

Beispiel

Vorname	Name	Adresse	Alter
Fritz	Müller	Hauptstr. 7	38
Hans	Albers	Bahnhofstr. 4	63

Programmiersprachliche Darstellung

Beispiel gemäß der Programmiersprache C

```
struct person {
    char vorname[30];
    char name[30];
    char adresse[100];
    int alter;
}
```

Beispiel der Nutzung:

```
struct person p; /* Deklaration der Person p */
p.vorname = "Fritz"; /* Vorname von Person p */
p.name = "Müller"; /* Name von Person p */
```

in Java: Zusammenfassung von zusammengehörigen Datenwerten zu Objekten, d.h. jede Tabellenzeile entspricht einem Objekt.

Beispiel

Vorname	Name	Adresse	Alter
Fritz	Müller	Hauptstr. 7	38
Hans	Albers	Bahnhofstr. 4	63

Programmiersprachliche Darstellung

Beispiel gemäß der Programmiersprache C

```
struct person {
    char vorname[30];
    char name[30];
    char adresse[100];
    int alter;
}
```

Beispiel der Nutzung:

```
struct person p; /* Deklaration der Person p */
p.vorname = "Fritz"; /* Vorname von Person p */
p.name = "Müller"; /* Name von Person p */
```

in Java: Zusammenfassung von zusammengehörigen Datenwerten zu Objekten, d.h. jede Tabellenzeile entspricht einem Objekt.

"Datenstruktur" (im Gegensatz zu struct-Datentypen): Datentyp mit Menge von zugehörigen Operationen.

Elementare Datentypen

Datentyp: Zusammenfassung von Werten gleicher Art (z.B. ganze Zahlen, Gleitkommazahlen, Zeichen).

Basis-Datentypen

Die Verwendung der Schlüsselwörter ("keywords") sind abhängig von der jeweiligen Programmiersprache.

`int` : Ausschnitt der ganzen Zahlen (im Rechner darstellbare ganze Zahlen)

`float` : Menge der Gleitkommazahlen

`double` : Menge der Gleitkommazahlen mit doppelter Genauigkeit

`char` : Menge der Zeichen

`boolean` : Werte `true` und `false`

Datentyp Verbund (struct)

Grundlagen der Programmierung

"Kunst des Programmierens". Grundlagen zu Datenstrukturen, Programmkonstrukte, Strukturierung von Programmen, objekt-orientierte Programmierung.

- Fragestellungen des Abschnitts:
  - Was ist ein Algorithmus?
  - Welche elementaren Datenstrukturen gibt es?
  - Was sind die grundlegenden Konstrukte einer Programmiersprache?
  - Was ist unter Objekt-orientierter Programmierung zu verstehen?
  - Was versteht man unter Modularisierung und Rekursion?

Programmkonstrukte

Anweisungen eines Programms weisen Werte an Variable zu oder steuern Kontrollfluss. Kontrollfluss bestimmt Folge der ausgeführten Anweisungen.

Schleifen

Wiederholte Ausführung von Anweisungen

Beispiel 1

```
N = 10
sum = 0
for (i = 0; i < N; i=i+1) sum = sum + i;
```

Bestandteile einer for-Schleife

Anfangswert der Laufvariable i, z.B. i =0

Endwert der Laufvariable i, z.B. i < N

Inkrementierung (Hochzählen) der Laufvariable i nach jedem Schleifendurchlauf, z.B. i = i + 1

Anweisungen, die bei jedem Schleifendurchlauf ausgeführt werden, z.B. sum = sum + i

Beispiel 2

```
sum = 1
while (i > 0) {
   sum = sum * i;
   i = i - 1;
}
```

Beispiel 3

Einfache unendliche Schleife

```
while (true) {
```

Wiederholte Ausführung von Anweisungen

Beispiel 1

```
N = 10
sum = 0
for (i = 0; i < N; i=i+1) sum = sum + i;
```

Bestandteile einer for-Schleife

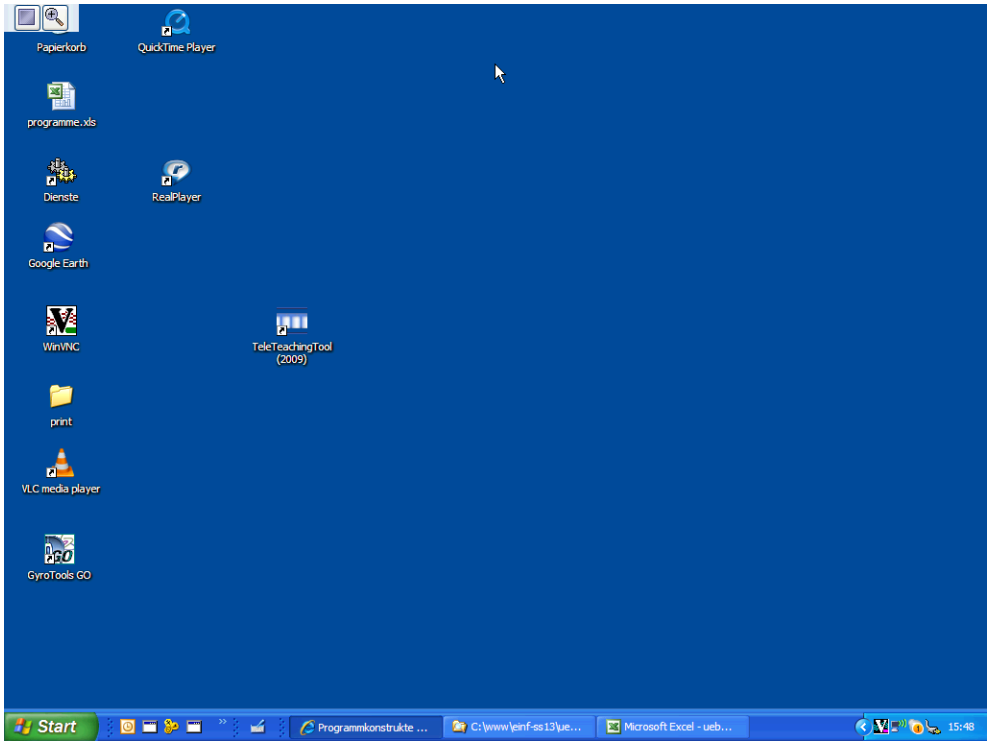
- Anfangswert der Laufvariable i, z.B. i =0
- Endwert der Laufvariable i, z.B. i < N
- Inkrementierung (Hochzählen) der Laufvariable i nach jedem Schleifendurchlauf, z.B. i = i + 1
- Anweisungen, die bei jedem Schleifendurchlauf ausgeführt werden, z.B. sum = sum + i

Beispiel 2

```
sum = 1
while (i > 0) {
   sum = sum * i;
   i = i - 1;
}
```

Beispiel 3

Einfache unendliche Schleife




```
N = 10
sum = 0
for (i = 0; i < N; i=i+1) sum = sum + i;
```

Bestandteile einer for-Schleife

- Anfangswert der Laufvariable i, z.B. i =0
- Endwert der Laufvariable i, z.B. i < N
- Inkrementierung (Hochzählen) der Laufvariable i nach jedem Schleifendurchlauf, z.B. i = i + 1
- Anweisungen, die bei jedem Schleifendurchlauf ausgeführt werden, z.B. sum = sum + i

Beispiel 2

```
sum = 1
while (i > 0) {
   sum = sum * i;
   i = i - 1;
}
```

Beispiel 3

Einfache unendliche Schleife

```
while (true) {
   ......
}
```