Script **generated by TTT**

Title: Einf_HF (08.06.2015)

Date: Mon Jun 08 14:15:30 CEST 2015

Duration: 90:33 min

Pages: 39

Programmkonstrukte

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

Zuweisungen

Alternativanweisungen

Schleifen

Generated by Targeteam

Alternativanweisungen

Auswahl von Alternativen abhängig von einer Bedingung

Beispiel 1

```
if (a > b)
   z= a; /* Zuweisung an die Variable z */
else
   z = b;
```

Beispiel 2 (Schachteln von Alternativen)

```
if (a > b) z = a;
else if (a < b) z = b;
else z = 0;
```

Verallgemeinert: Auswahl-Konstrukt (`switch, case`).

Generated by Targeteam

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) {
```

Objektorientierte Programmierung

Softwaresystem realisiert durch Menge von Objekten. Gegensatz prozedurale Programmierung: Anweisungen im Vordergrund.

Objektorientiertes Programmieren: Daten im Vordergrund. In Objekten zusammengefasst ("Verkapselung"). Funktionen lokal bei Objekten definiert.

Die Funktionen werden Methoden genannt.

[Objekt - Klasse]
[Erzeugen eines Objekts]
[Vererbung]

Objekt - Klasse

Objekt: Exemplar (Instanz) eines Gegenstandes oder Begriffs. Möglichst stark an reale Welt angelehnt.

Objekt

Ein Objekt hat

1. einen ***eindeutigen Objektnamen*** , z.B. vom Benutzer vergebene Namen. Zusätzlich interner Identifikator.
2. einen ***Zustand*** , definiert durch Attribute und die zugehörigen Attributwerte ("Instanzvariable", private Daten).
3. ein ***Verhalten*** , spezifiziert durch eine Menge von Operationen (Methoden), die auf den Attributen agieren und Operationen anderer Objekte aufrufen können;
4. Beziehungen zu anderen Objekten.

Notation für den Zugriff auf die Objektdaten

Objektname.Attributname = Attributwert

[Klasse]

Klasse

Objekte mit gleichen Attributen und gleichem Verhalten. Objekt "weiß", zu welcher Klasse es gehört.

Beispiel: ***Klasse*** Person, zu der die ***Objekte*** Schmidt, Mayer, Müller gehören.

Beispiel: Klassendefinition

```
public class circle {
    double x, y; // Koordinaten der Kreismitte
    double r; //Radius des Kreises
    // Konstruktor für die Erzeugung von Objekten:
    public circle (double xcoord, double ycoord, double radius)
       {x = xcoord; y = ycoord; r = radius;}
    // Methoden, die Umfang und die Fläche
    // als Ergebnis liefern:
    public double circumference(){return 2 * 3.14159 * r;}
    public double area(){return 3.14159 * r * r;}
}
```

`double` bezeichnet eine Variable mit doppelter Genauigkeit.

Schutzmechanismen

Kontrolle des Zugriffs auf Daten und Methoden von außen

public : Zugriff von außen möglich.

private : Zugriff nur von innerhalb der Klasse möglich.

Erzeugen eines Objekts

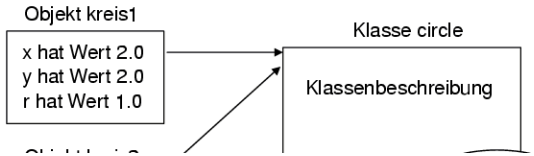
Nach Spezifikation einer Klasse: beliebig viele Objekte dazu erzeugbar ("Objektinstanziierung"). Jeweils eigene Attributwerte, jedoch Methoden gemeinsam.

Beispiel

```
circle kreis1, kreis2; // Vereinbarung zweier Variable
double flaeche, umfang;

kreis1 = new circle(2.0,2.0,1.0);
    // Erzeugen des Objekts kreis1
    // Mittelpunkt: (2,2), Radius: 1.0
    // Abfragen der Fläche von kreis1:
flaeche = kreis1.area();

// Erzeugen des zweiten Kreises
kreis2 = new circle(4.0, -1.0, 10.0);
    // Abfragen des Umfangs von kreis2:
umfang = kreis2.circumference()
.............
```



Objektorientierte Programmierung

Softwaresystem realisiert durch Menge von Objekten. Gegensatz prozedurale Programmierung: Anweisungen im Vordergrund.

Objektorientiertes Programmieren: Daten im Vordergrund. In Objekten zusammengefasst ("Verkapselung"). Funktionen lokal bei Objekten definiert.

Die Funktionen werden Methoden genannt.

Objekt - Klasse

Erzeugen eines Objekts

Vererbung

Vererbung

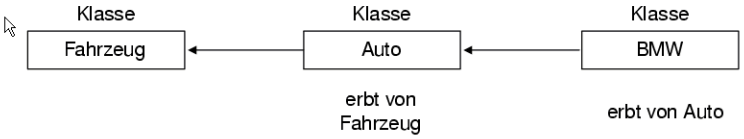
Werkzeug zum Organisieren und Konstruieren von Klassen; Wiederverwendung.

Definition - Vererbung

Seien K und K_i , i = 1,..,n Klassen; Vererbung ist eine Beziehung zwischen K und den K_i , wobei Struktur und Verhalten von K durch Struktur und Verhalten der K_i bestimmt wird; K "erbt" von den Klassen K_i ;

Beziehung zwischen Klassen; K Unterklasse (Subklasse) von K_i ; K_i Oberklasse (Superklasse) von K;

Unterklassen übernehmen Eigenschaften (Attribute, Operationen und Beziehungen) der Oberklasse(n); ggf. über mehrere Stufen. Betrifft Attribute und Methoden;



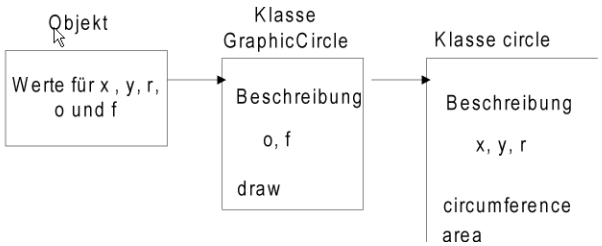
Einfachvererbung: eine Klasse hat nur eine direkte Oberklasse, d.h. n=1.

Mehrfachvererbung: eine Klasse hat mehrere direkte Oberklassen

Beispiel

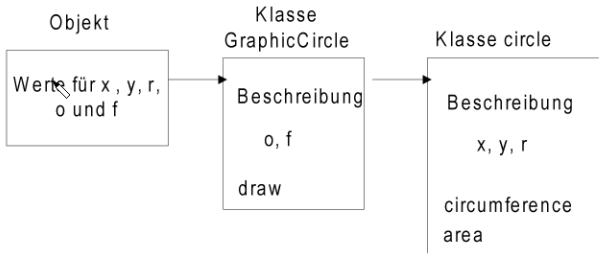
Beispiel

```
public class GraphicCircle extends circle {
    // Es werden automatisch die Variablen und Methoden
    // der Klasse circle geerbt;
    // nur die neuen Informationen müssen hier
    // aufgeführt werden:
    // Die Farben für Rand und Füllfläche.
    color o, f;
    public GraphicCircle (double xcoord, double ycoord, double radius, color
    outline, color fill)
       {super(xcoord,ycoord,radius); o = outline; f = fill;}
    public void draw(Window dw)
       {dw.drawCircle(x, y, r, o, f;}
}
```



Beispiel

```
// Es werden automatisch die Variablen und Methoden
// der Klasse circle geerbt;
// nur die neuen Informationen müssen hier
// aufgeführt werden:
// Die Farben für Rand und Füllfläche.
color o, f;
public GraphicCircle (double xcoord, double ycoord, double radius, color
outline, color fill)
   {super(xcoord,ycoord,radius); o = outline; f = fill;}
public void draw(Window dw)
   {dw.drawCircle(x, y, r, o, f;}
}
```



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?

Einführung

Algorithmus

Datentypen und Ausdrücke

Programmkonstrukte

Objektorientierte Programmierung

Modularisierung von Programmen

Rekursion

Entwurf von Algorithmen

Generell sinnvolle Vorgehensweise:

Spezifikation der Problemstellung

Bestimmung der Definitionsbereiche und der Datentypen

Suche nach / Vergleich mit bekannten Algorithmen, eventuell Erweiterung oder Anpassung der bekannten Algorithmen

Zerlegung des Problems in Teilprobleme ("divide and conquer"-Vorgehensweise); auf jeder Zerlegungsebene Verwendung von

Sequenz von Schritten

Fallunterscheidung (Alternativen)

Iteration von Schritten (Schleifen)

Wiederhole diese Vorgehensweise für jedes Teilproblem

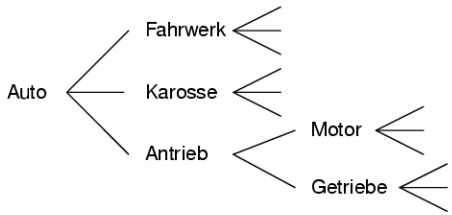
Schrittweises Verfeinern von Algorithmen

Direktes Vorgehen oft schwierig, da Problemumfang sehr groß. Oft nicht klar, welche zusätzlichen Aspekte mitzuberücksichtigen sind.

Problem: Unüberschaubarkeit bei komplexen Aufgaben

Lösung: Zuerst in größeren, gröberen Einheiten denken, dann diese zerlegen

Beispiel: Entwicklung eines Autos



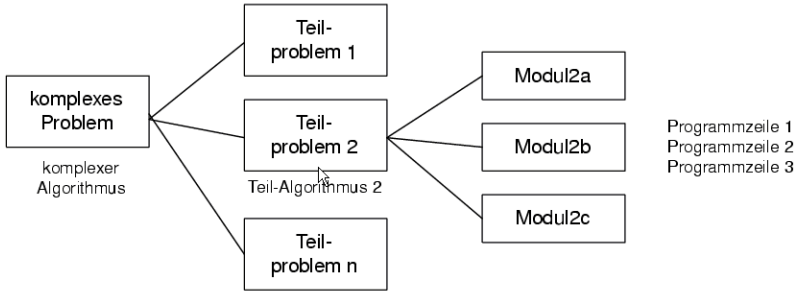
Übertragung dieses Ansatzes auf die Zerlegung von Algorithmen in kleinere und überschaubarere Einheiten.

Allgemeines

Entwurf von Algorithmen

Schrittweises Verfeinern von Algorithmen

Algorithmen-Zerlegung



Beispiel Zerlegung

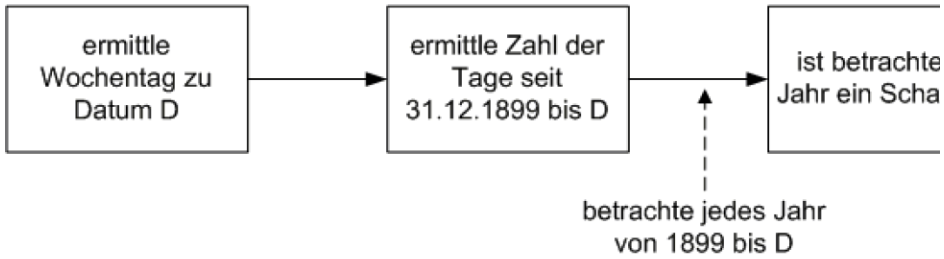
Aufgabenstellung: Ermittle zu einem gegebenen Datum nach dem 1.1.1900 (gegeben als Tag im Monat, Monat im Jahr und Jahr AD) den Wochentag.

Aufteilung der Aufgabe in mehrere Teilaufgaben

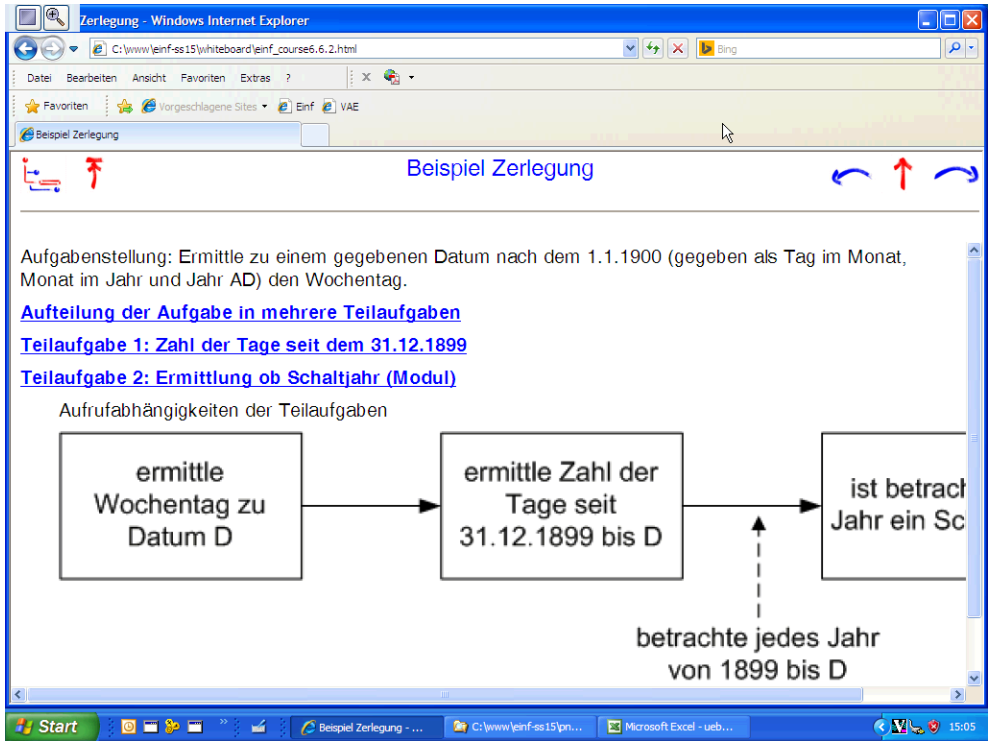
Teilaufgabe 1: Zahl der Tage seit dem 31.12.1899

Teilaufgabe 2: Ermittlung ob Schaltjahr (Modul)

Aufrufabhängigkeiten der Teilaufgaben



Beispiel für den 24.12.2000



Aufteilung der Aufgabe in mehrere Teilaufgaben

Der 31.12.1899 war ein Sonntag

Ermittle die Zahl der Tage zwischen dem 31.12.1899 und dem gegebenen Datum (Modul)

Dividiere diese Zahl durch 7 und ermittle den Rest der Division

Wenn der Rest 0 ist, dann ist das Datum ein Sonntag, bei Rest 1 ein Montag, bei Rest 2 ein Dienstag, ...

Teilaufgabe 2: Ermittlung ob Schaltjahr (Modul)

Teilaufgabe wird als Modul mit folgendem Ablauf realisiert:

Wenn nicht durch 4 teilbar, dann nein.

Wenn nicht durch 100 teilbar, dann ja.

Wenn nicht durch 400 teilbar, dann nein, sonst ja.

Beispiel für den 24.12.2000

Zähler = 0

Gehe von 1900 bis 1999 und addiere jeweils 365 oder 366; Ergebnis ist 36524.

Gehe von Januar bis November und addiere 31, 28+1, 31, 30, 31, 30, 31, 31, 30, 31, 30 = 335

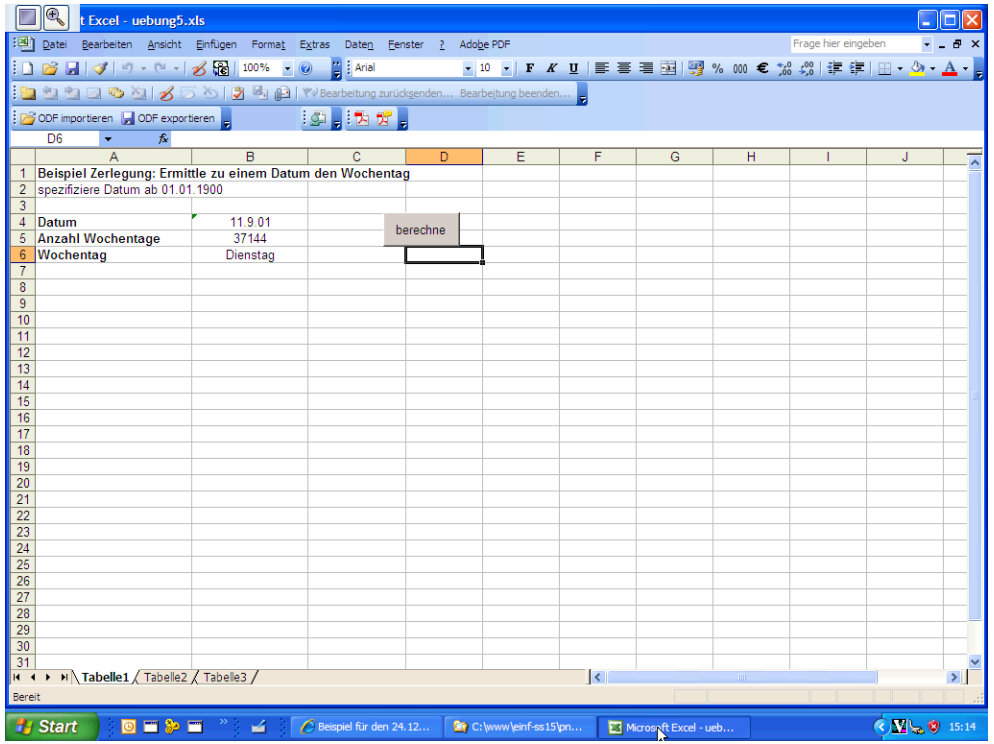
Addiere 24

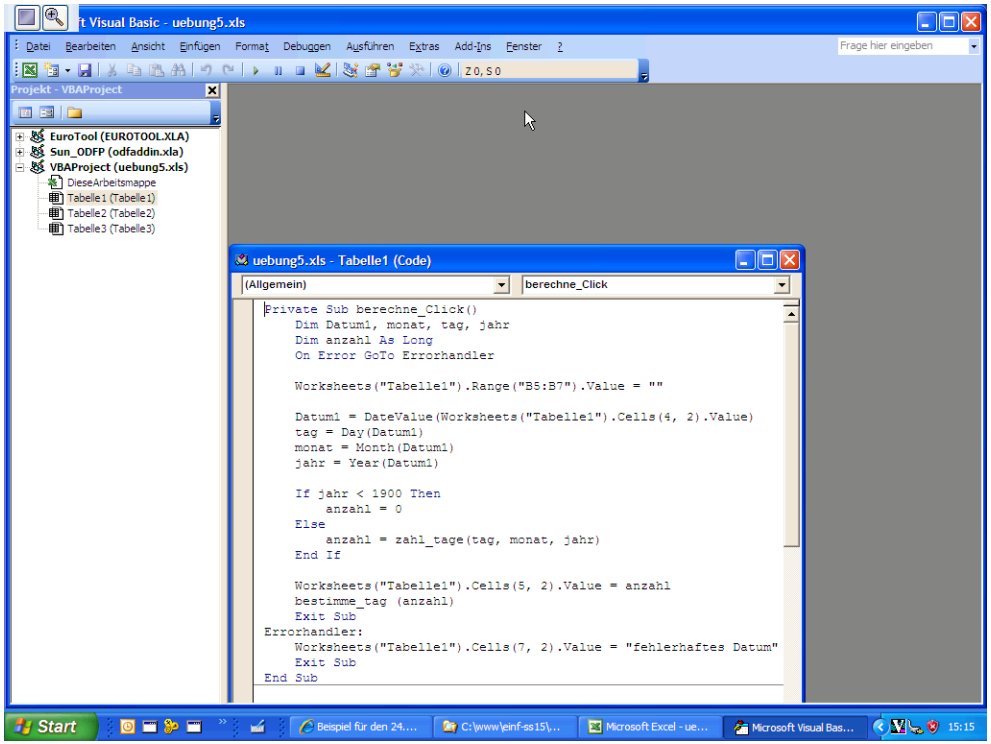
Zähler = 36883

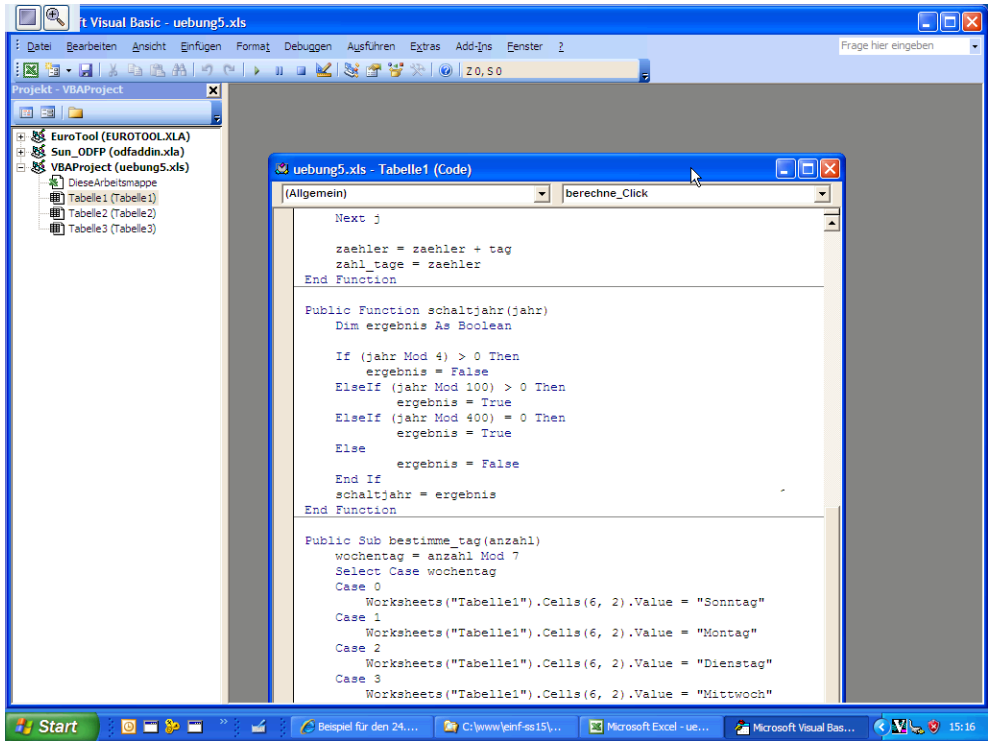
36883/7 = 5269

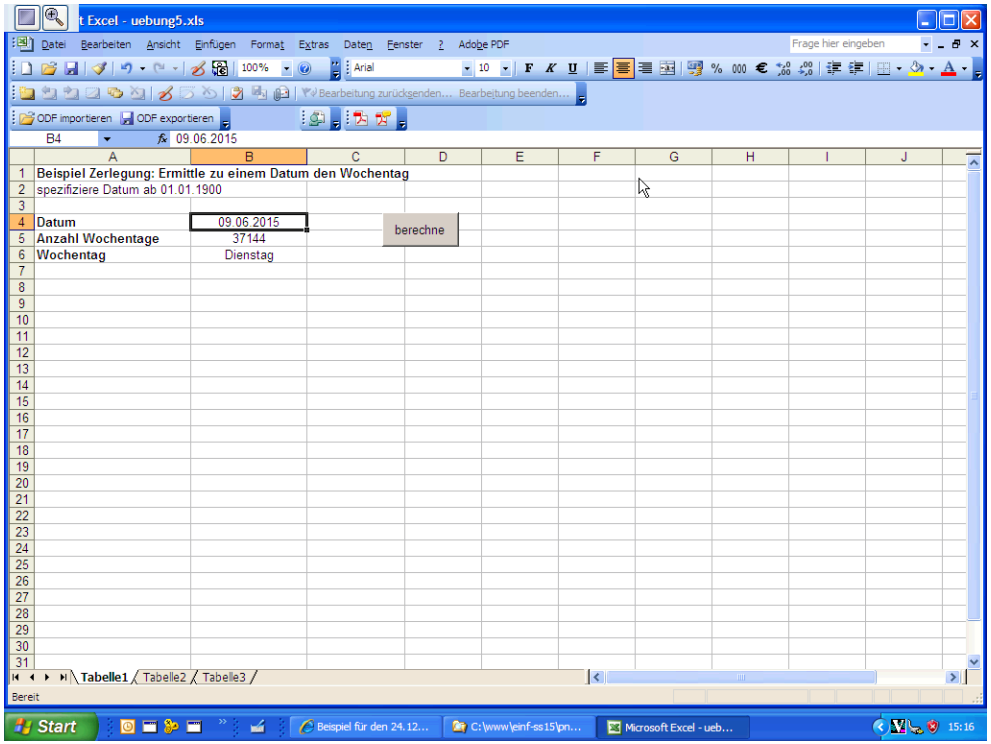
Rest der Division ist 0, also ist der 24.12.2000 ein Sonntag

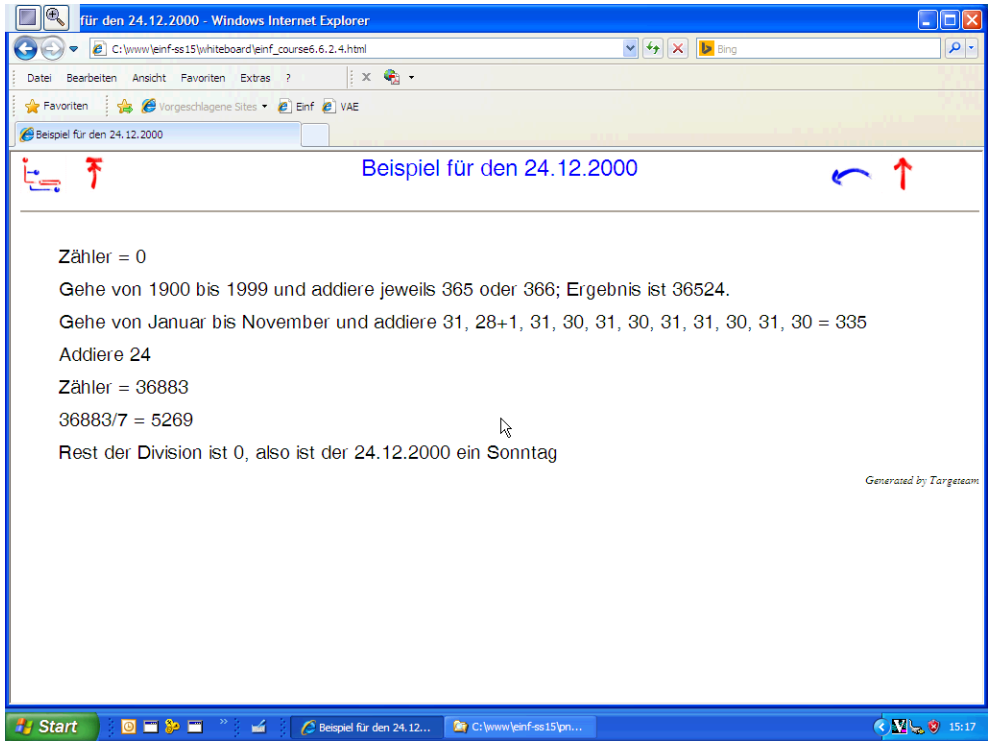
Generated by Targeteam

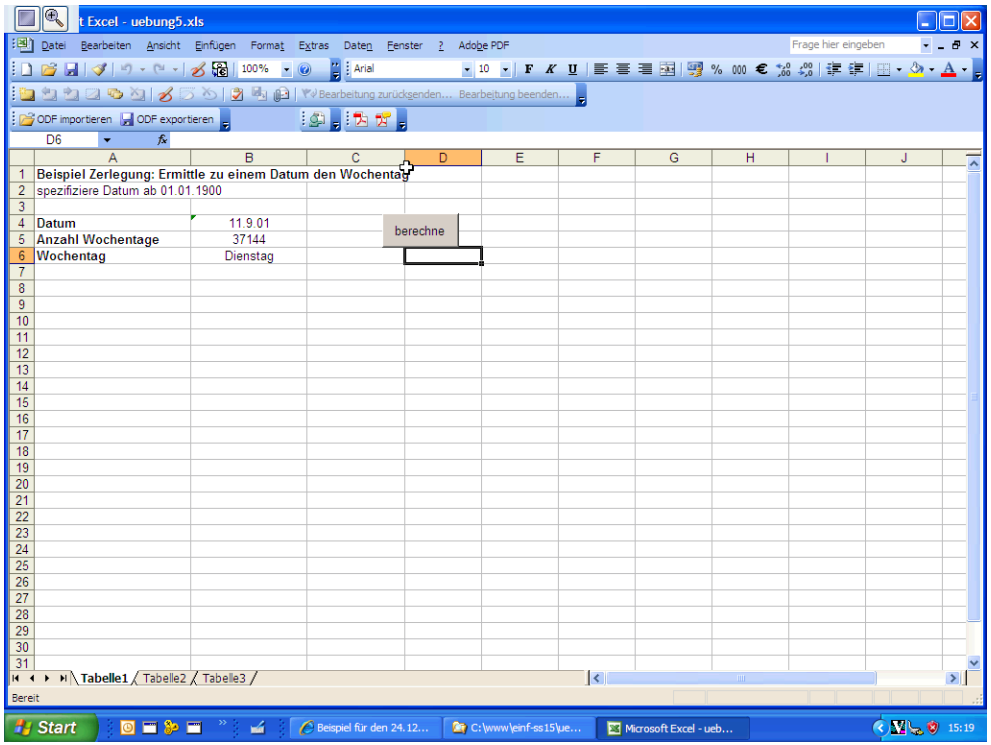


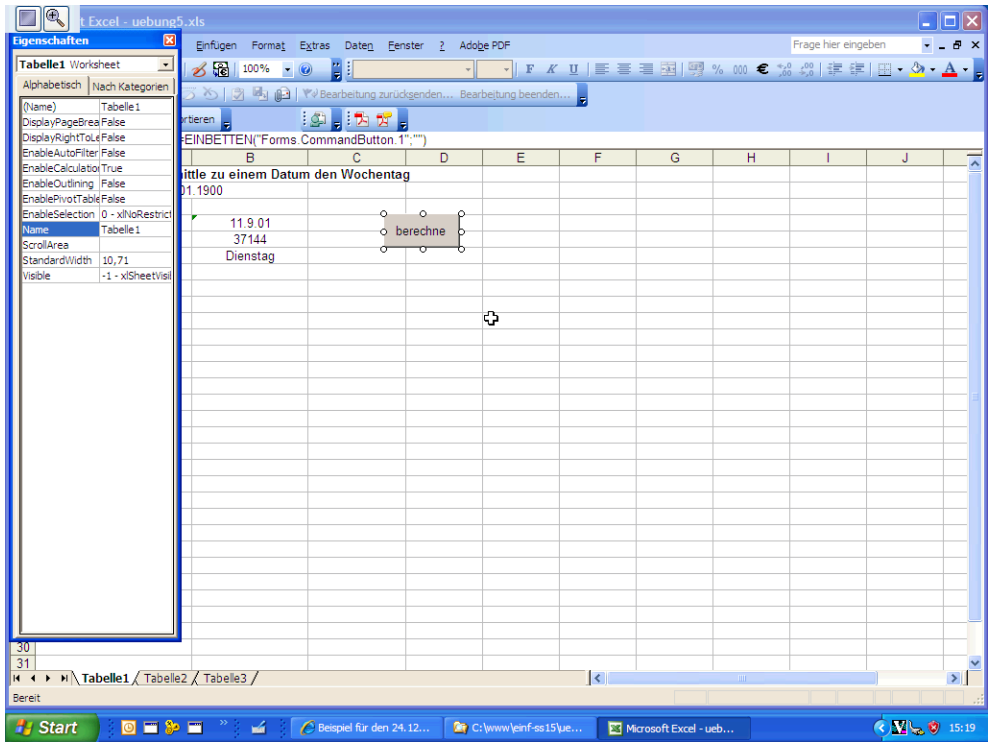


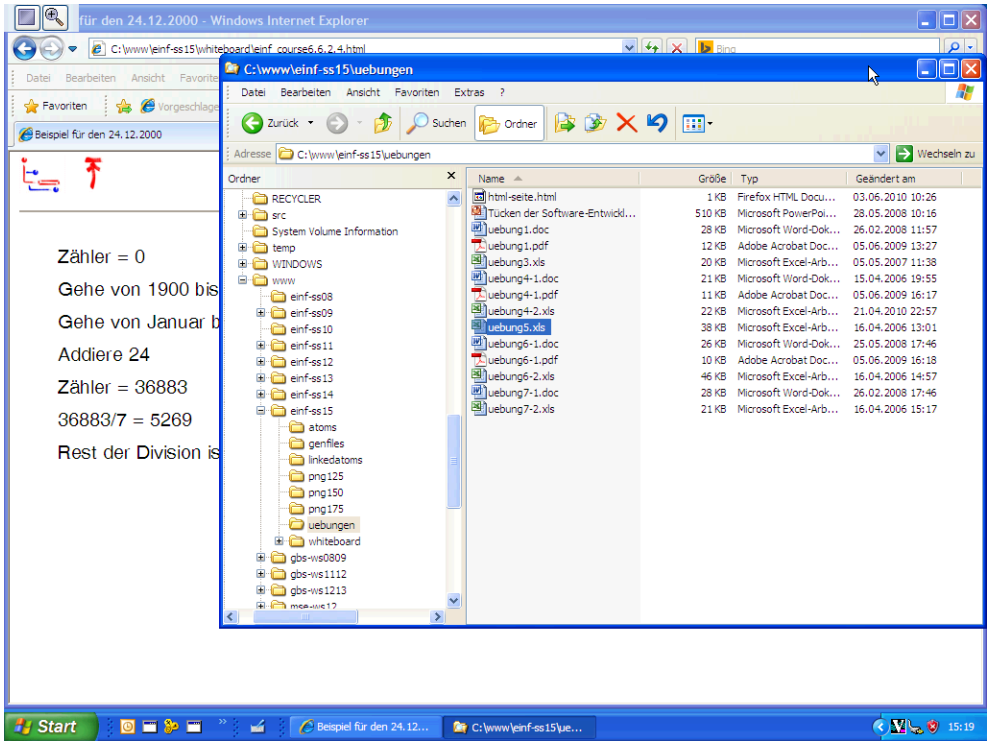



t Excel - uebung5.xls
B4
09.06.2015
Beispiel Zerlegung: Ermittle zu einem Datum den Wochentag
spezifiziere Datum ab 01.01.1900
Datum
09.06.2015
Anzahl Wochentage
37144
Wochentag
Dienstag
berechne
Tabelle1
Tabelle2
Tabelle3
Bereit


für den 24.12.2000 - Windows Internet Explorer
C:\www\einf-ss15\whiteboard\einf_course6.6.2.4.html
Beispiel für den 24.12.2000
Beispiel für den 24.12.2000
Zähler = 0
Gehe von 1900 bis 1999 und addiere jeweils 365 oder 366; Ergebnis ist 36524.
Gehe von Januar bis November und addiere 31, 28+1, 31, 30, 31, 30, 31, 31, 30, 31, 30 = 335
Addiere 24
Zähler = 36883
36883/7 = 5269
Rest der Division ist 0, also ist der 24.12.2000 ein Sonntag
Generated by Targeteam


t Excel - uebung5.xls
D6
Beispiel Zerlegung: Ermittle zu einem Datum den Wochentag
spezifiziere Datum ab 01.01.1900
Datum
11.9.01
Anzahl Wochentage
37144
Wochentag
Dienstag
berechne
Tabelle1
Tabelle2
Tabelle3
Bereit


Eigenschaften
Tabelle1 Worksheet
Alphabetisch
Nach Kategorien
(Name) Tabelle1
Name Tabelle1
ScrollArea
StandardWidth 10,71
Visible -1 - xlSheetVisil
11.9.01
37144
Dienstag
berechne
Tabelle1
Tabelle2
Tabelle3
Bereit



Modularisierung von Programmen

Systematischer Aufbau von Programmen durch Zerlegung in eigenständige Einheiten.

Allgemeines

Beispiel Zerlegung

Strukturierung von Algorithmen

Module

Prozedurales / Objektorientieres Programmieren

Generated by Targeteam

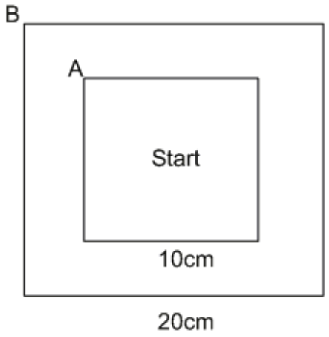
Beispielaufgabe

Zeichnen von zwei konzentrischen Quadraten

Plotterfunktionen

Teilaufgaben

Es existieren die folgenden beiden Teilaufgaben (wobei der Zeichenstift am Start in der Mitte des inneren Quadrats ist).



Position auf A; Quadrat 10 cm zeichnen

Position auf B; Quadrat 20 cm zeichnen

Definition eines Moduls

Nutzung des Moduls (Hauptprogramm)

Generated by Targeteam

Plotterfunktionen

Gegeben sind folgende Plotter-Grundfunktionen:

Bewege (x): bewege Zeichenstift um Länge x in aktuelle Zeichenrichtung

Links (x): drehe Zeichenrichtung um x Grad nach links

Rechts (x): drehe Zeichenrichtung um x Grad nach rechts

Stift heben

Stift senken

Generated by Targeteam

Plotterfunktionen

Gegeben sind folgende Plotter-Grundfunktionen:

- Bewege (x): bewege Zeichenstift um Länge x in aktuelle Zeichenrichtung
- Links (x): drehe Zeichenrichtung um x Grad nach links
- Rechts (x): drehe Zeichenrichtung um x Grad nach rechts
- Stift heben
- Stift senken

Definition eines Moduls

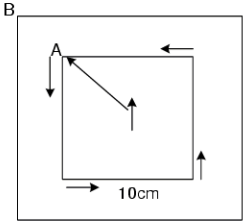
Kopf	public Quadratzeichnen(Größe)
Rumpf	Stift senken
	wiederhole 4-mal
	Bewege (Größe)
	Links (90)
	Stift heben

Nutzung des Moduls (Hauptprogramm)

- Links (45)
- Bewege (10cm*1/2* sqrt(2))
- Links (135)
- Quadratzeichnen (10cm)
- Rechts (135)
- Bewege ((20cm-10cm)*1/2*sqrt(2))
- Links (135)
- Quadratzeichnen (20cm)
- Rechts (180)

"sqrt(2)" berechnet die Wurzel von 2.

B



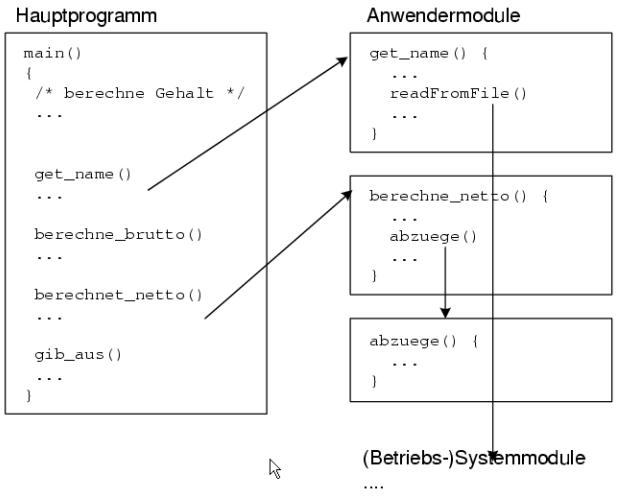
Strukturierung von Algorithmen

Zweck der Strukturierung

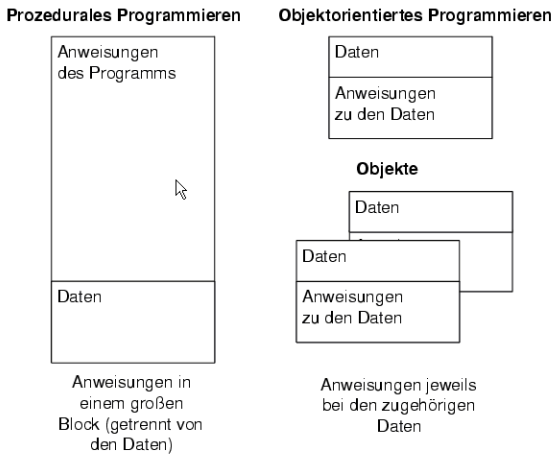
- Vereinfachte Darstellung, höhere Lesbarkeit
- Wiederverwendung von Teilen in anderen Kontexten

Beispielaufgabe

Hauptprogramm bestehend aus Teilmodulen



Prozedurales / Objektorientieres Programmieren



Prozedurales Programmieren

Sequenz von Anweisungen, ausführen, warten. Basiselemente: Sequenz, Auswahl (if-then-else), Iteration. Trennung von Daten und Anweisungen.

Objektorientiertes Programmieren

Objekte statt Anweisungen. Objekt hat Attribute, Methoden und Ereignisbehandlung. Zusammenfassung von Daten und Anweisungen

Modularisierung von Programmen

Systematischer Aufbau von Programmen durch Zerlegung in eigenständige Einheiten.

Allgemeines

Beispiel Zerlegung

Strukturierung von Algorithmen

Module

Prozedurales / Objektorientieres Programmieren

Rekursion

Spezielle Form der Modularisierung. Zu definierendes Modul wird in seiner Definition selbst benutzt.

"natürlichere" Darstellung bei bestimmten Algorithmen und Datenstrukturen.

Beispiel Summe

Definiertes Ende einer rekursiven Schachtelung.

```
if (n > 0)
   summe = summe(n-1) + n;
else summe = 0; /* definiertes Ende, wenn n <= 0 */
```


Beispiel 'Fibonacci-Zahlen'

Beispiel 'Größter gemeinsamer Teiler'

Beispiel 'Türme von Hanoi'