Script **generated by TTT**

Title: Einf_HF (11.06.2012)

Date: Mon Jun 11 14:17:11 CEST 2012

Duration: 88:42 min

Pages: 37

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?

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).

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

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.

Datentyp Verbund (struct)

Beispiel

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

Person

Programmiersprachliche Darstellung

ein neuer Datentyp

Beispiel gemäß der Programmiersprache C

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

max 30 Zeichen

" " "

max 100 "

Variable

Beispiel der Nutzung:

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

p.alter = 38

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

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?

Zuordnung von Werten an Variable

Variable: Bereich im Arbeitsspeicher, über Namen angesprochen. Für Daten beliebiger Datentypen, einzelne Variable i.a. auf bestimmten Datentyp festgelegt. Syntax: meist "=" oder ":=".

Beispiele

```
i = 2;
i = i + 2; /* Wert von i wird um 2 erhöht und wieder an i zugewiesen
*/
x = x * (y + 1);
```


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

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) {
```

Programmkonstrukte

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

Zuweisungen

Alternativanweisungen

Schleifen

Generated by Targeteam

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: 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;}
}
```

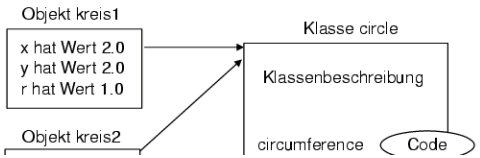
`double` bezeichnet eine Variable mit doppelter Genauigkeit.

Erzeugen eines Objekts

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

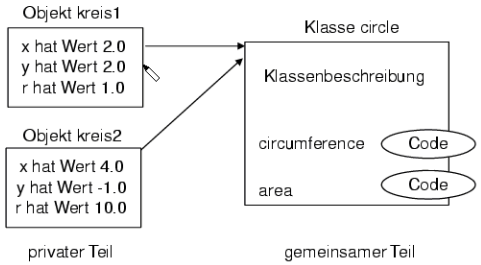
Beispiel

```
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()
..............
```



Erzeugen eines Objekts

```
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()
. . . . . .
```



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;

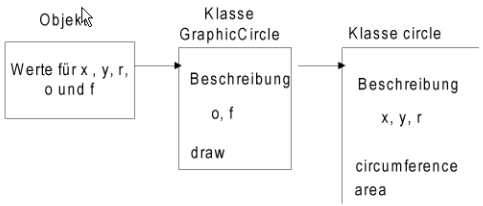
Unterklassen übernehmen Eigenschaften (Attribute, Operationen und Beziehungen) der Oberklasse(n); ggf. über mehrere Stufen. Betrifft Attribute und Methoden;

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

Mehrfachvererbung: eine Klasse hat mehrere direkte Oberklassen

Beispiel


```
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;}
}
```



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

"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?

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

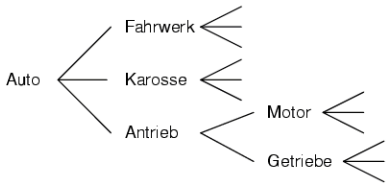
Schrittweises Verfeinern von Algorithmen

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

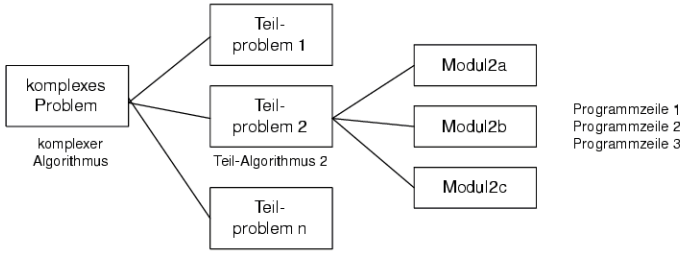
Beispiel: Entwicklung eines Autos



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

Allgemeines

Algorithmen-Zerlegung

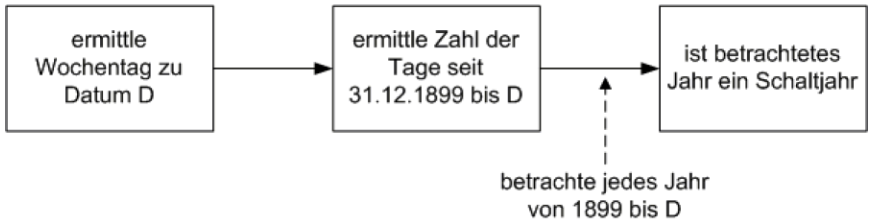


Modularisierung von Programmen

Systematischer Aufbau von Programmen durch Zerlegung in eigenständige Einheiten.

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

Aufrufabhängigkeiten der Teilaufgaben



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 wird als Modul mit folgendem Ablauf realisiert:

Setze Zähler auf 0

Addiere für jedes Jahr (vor dem aktuellen Jahr) seit 1900 die Zahl 365 zum Zähler, wenn Jahr ein Schaltjahr ist, dann addiere zusätzlich 1

Addiere für jeden vollendeten Monat im aktuellen Jahr die Zahl der Tage im Monat zum Zähler, wenn Monat Februar und Jahr Schaltjahr, dann addiere zusätzlich 1

Addiere die Zahl der vollendeten oder angebrochenen Tage im aktuellen Monat zum Zähler

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.

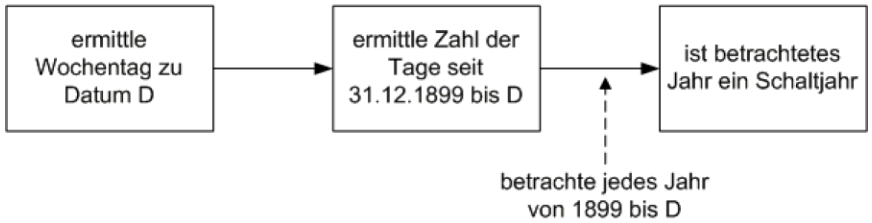
Aufgabenstellung: Ermittle zu einem gegebenen Datum nach dem 1.1.1900 (gegeben als Tag im Monat, Monat im Jahr und Jahr AD) den Wochentag.

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

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

Beispiel Zerlegung

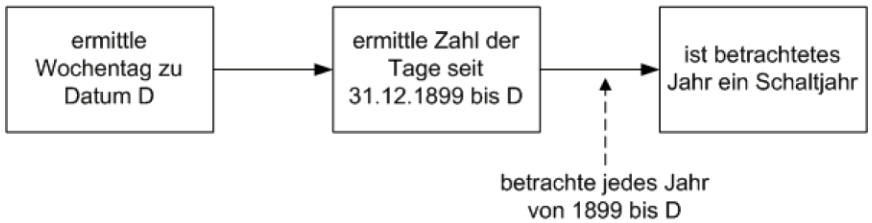
Aufgabenstellung: Ermittle zu einem gegebenen Datum nach dem 1.1.1900 (gegeben als Tag im Monat, Monat im Jahr und Jahr AD) den Wochentag.

Aufteilung der Aufgabe in mehrere Teilaufgaben

Teilaufgabe 1: Zahl der Tage seit dem 31.12.1899

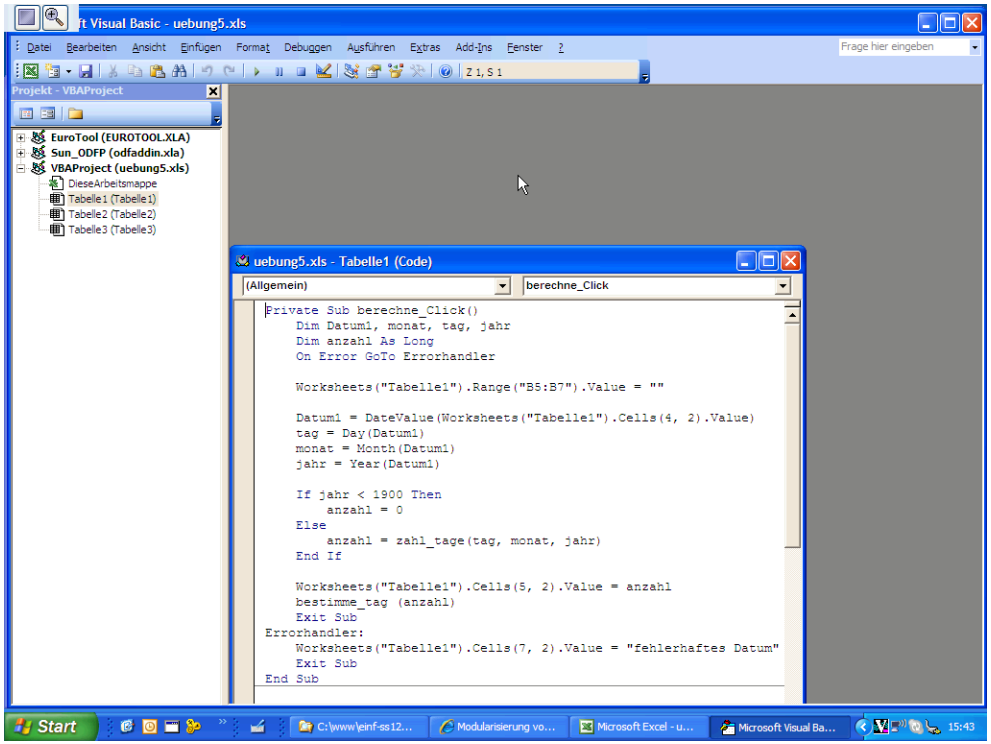
Teilaufgabe 2: Ermittlung ob Schaltjahr (Modul)

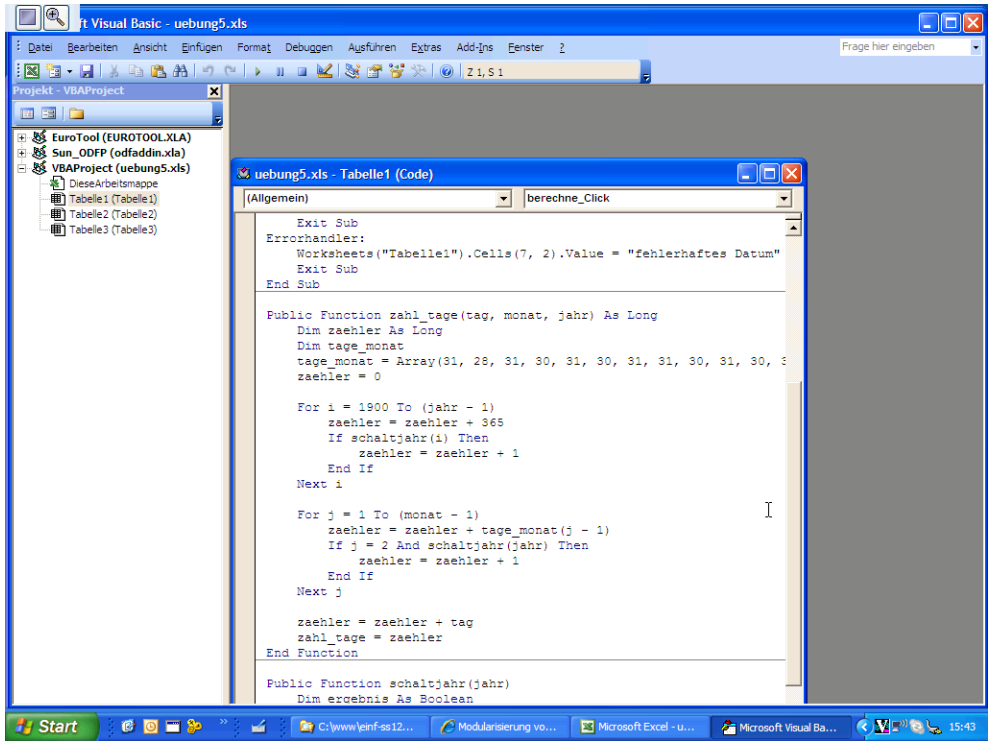
Aufrufabhängigkeiten der Teilaufgaben

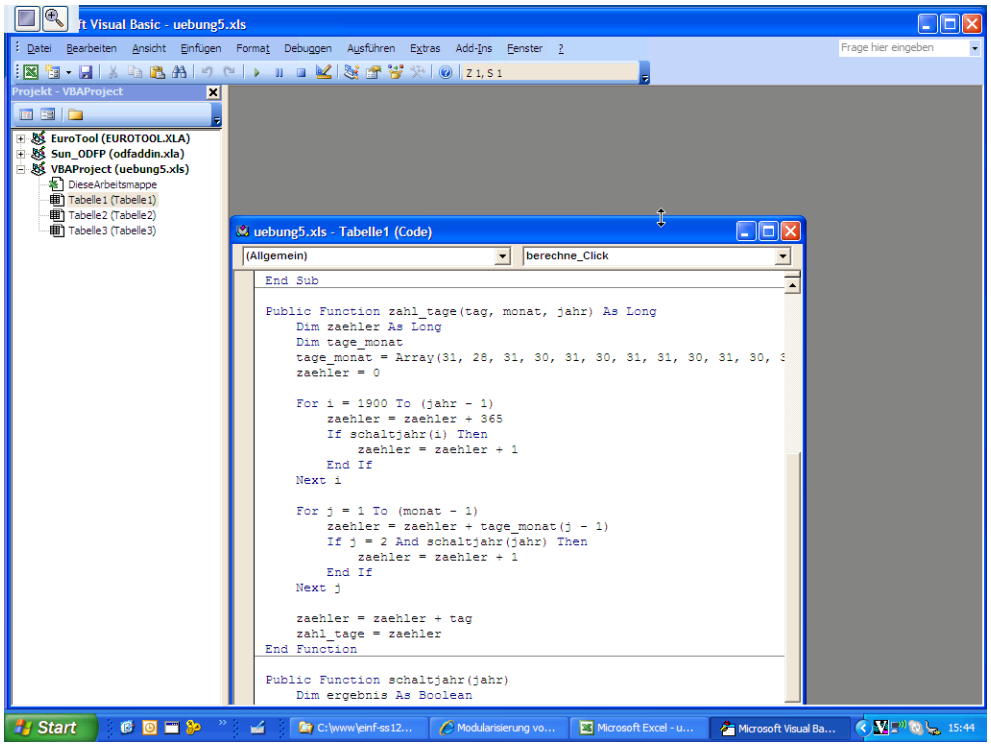


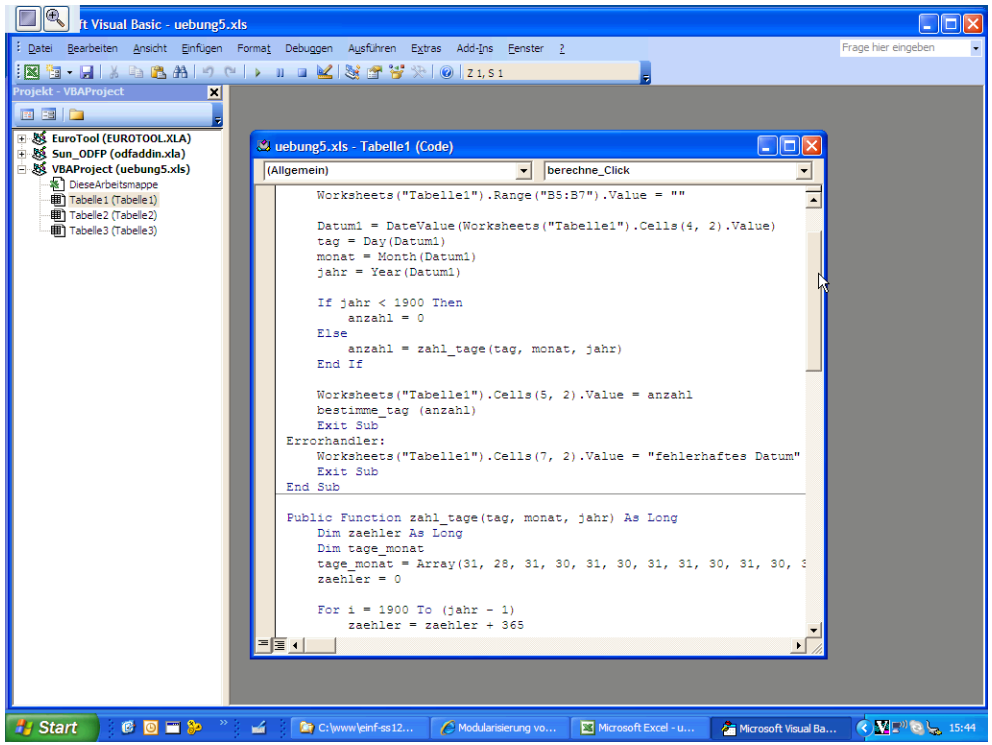
Beispiel für den 24.12.2000

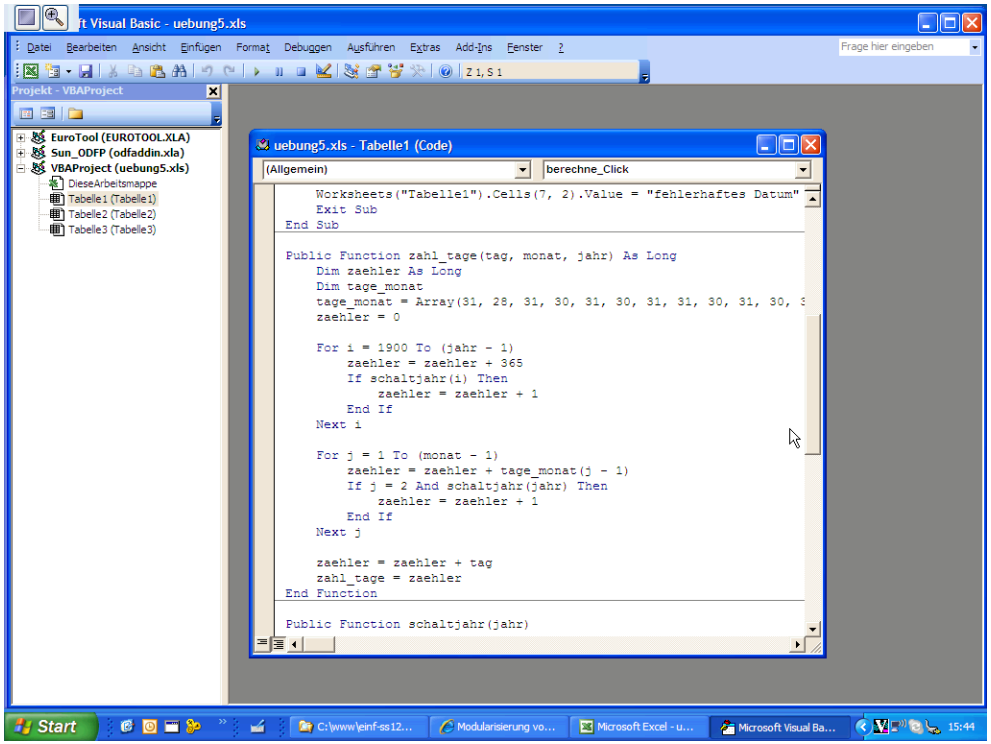
Generated by Targeteam

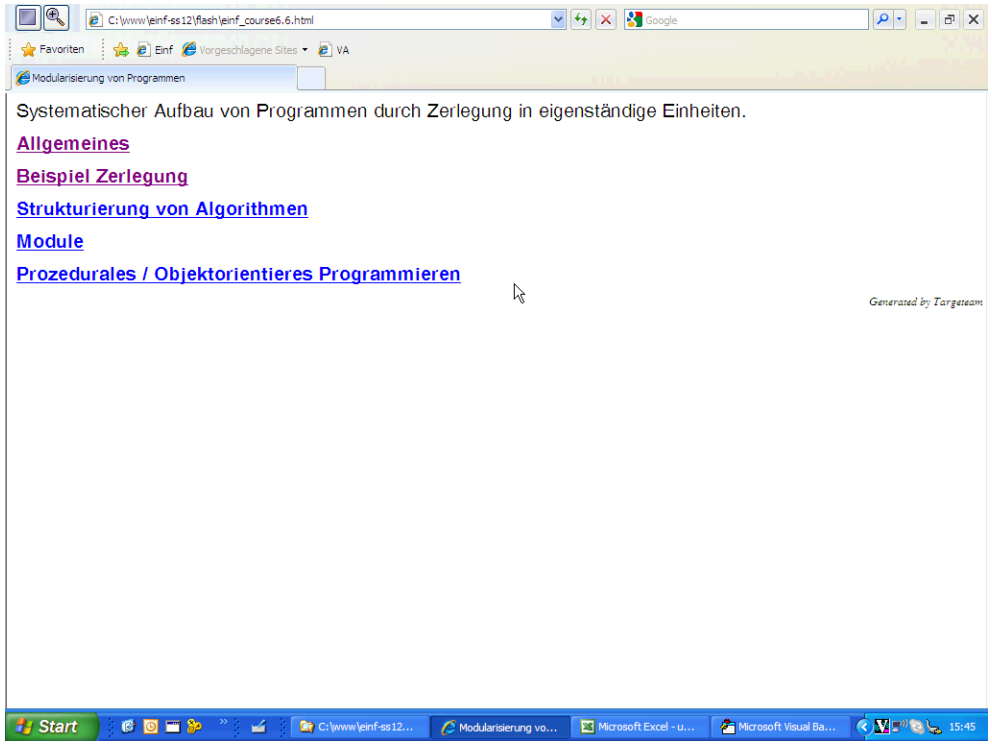

ft Visual Basic - uebung5.xls
Datei Bearbeiten Ansicht Einfügen Format Debuggen Ausführen Extras Add-Ins Fenster ?
Frage hier eingeben
Projekt - VBAProject
EuroTool (EUROTOOL.XLA)
Sun_ODFP (odfaddin.xla)
VBAProject (uebung5.xls)
DieseArbeitsmappe
Tabelle1 (Tabelle1)
Tabelle2 (Tabelle2)
Tabelle3 (Tabelle3)
uebung5.xls - Tabelle1 (Code)
(Allgemein)
berechne_Click
Private Sub berechne_Click()
Dim Datum1, monat, tag, jahr
Dim anzahl As Long
On Error GoTo Errorhandler
Worksheets("Tabelle1").Range("B5:B7").Value = ""
Datum1 = DateValue(Worksheets("Tabelle1").Cells(4, 2).Value)
tag = Day(Datum1)
monat = Month(Datum1)
jahr = Year(Datum1)
If jahr < 1900 Then
anzahl = 0
Else
anzahl = zahl_tage(tag, monat, jahr)
End If
Worksheets("Tabelle1").Cells(5, 2).Value = anzahl
bestimme_tag (anzahl)
Exit Sub
Errorhandler:
Worksheets("Tabelle1").Cells(7, 2).Value = "fehlerhaftes Datum"
Exit Sub
End Sub
Start
15:43


ft Visual Basic - uebung5.xls
Datei Bearbeiten Ansicht Einfügen Format Debuggen Ausführen Extras Add-Ins Fenster ?
Frage hier eingeben
Projekt - VBAProject
EuroTool (EUROTOOL.XLA)
Sun_ODFP (odfaddin.xla)
VBAProject (uebung5.xls)
DieseArbeitsmappe
Tabelle1 (Tabelle1)
Tabelle2 (Tabelle2)
Tabelle3 (Tabelle3)
uebung5.xls - Tabelle1 (Code)
(Allgemein)
berechne_Click
Exit Sub
Errorhandler:
Worksheets("Tabelle1").Cells(7, 2).Value = "fehlerhaftes Datum"
Exit Sub
End Sub
Public Function zahl_tage(tag, monat, jahr) As Long
Dim zaehler As Long
Dim tage_monat
tage_monat = Array(31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30,
zaehler = 0
For i = 1900 To (jahr - 1)
zaehler = zaehler + 365
If schaltjahr(i) Then
zaehler = zaehler + 1
End If
Next i
For j = 1 To (monat - 1)
zaehler = zaehler + tage_monat(j - 1)
If j = 2 And schaltjahr(jahr) Then
zaehler = zaehler + 1
End If
Next j
zaehler = zaehler + tag
zahl_tage = zaehler
End Function
Public Function schaltjahr(jahr)
Dim ergebnis As Boolean
Start
15:43


ft Visual Basic - uebung5.xls
Datei Bearbeiten Ansicht Einfügen Format Debuggen Ausführen Extras Add-Ins Fenster ?
Frage hier eingeben
Projekt - VBAProject
EuroTool (EUROTOOL.XLA)
Sun_ODFP (odfaddin.xla)
VBAProject (uebung5.xls)
DieseArbeitsmappe
Tabelle1 (Tabelle1)
Tabelle2 (Tabelle2)
Tabelle3 (Tabelle3)
uebung5.xls - Tabelle1 (Code)
(Allgemein)
berechne_Click
End Sub
Public Function zahl_tage(tag, monat, jahr) As Long
Dim zaehler As Long
Dim tage_monat
tage_monat = Array(31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30,
zaehler = 0
For i = 1900 To (jahr - 1)
zaehler = zaehler + 365
If schaltjahr(i) Then
zaehler = zaehler + 1
End If
Next i
For j = 1 To (monat - 1)
zaehler = zaehler + tage_monat(j - 1)
If j = 2 And schaltjahr(jahr) Then
zaehler = zaehler + 1
End If
Next j
zaehler = zaehler + tag
zahl_tage = zaehler
End Function
Public Function schaltjahr(jahr)
Dim ergebnis As Boolean
Start
15:44


ft Visual Basic - uebung5.xls
Datei Bearbeiten Ansicht Einfügen Format Debuggen Ausführen Extras Add-Ins Fenster ?
Frage hier eingeben
Projekt - VBAProject
EuroTool (EUROTOOL.XLA)
Sun_ODFP (odfaddin.xla)
VBAProject (uebung5.xls)
DieseArbeitsmappe
Tabelle1 (Tabelle1)
Tabelle2 (Tabelle2)
Tabelle3 (Tabelle3)
uebung5.xls - Tabelle1 (Code)
(Allgemein)
berechne_Click
Worksheets("Tabelle1").Range("B5:B7").Value = ""
Datum1 = DateValue(Worksheets("Tabelle1").Cells(4, 2).Value)
tag = Day(Datum1)
monat = Month(Datum1)
jahr = Year(Datum1)
If jahr < 1900 Then
anzahl = 0
Else
anzahl = zahl_tage(tag, monat, jahr)
End If
Worksheets("Tabelle1").Cells(5, 2).Value = anzahl
bestimme_tag (anzahl)
Exit Sub
Errorhandler:
Worksheets("Tabelle1").Cells(7, 2).Value = "fehlerhaftes Datum"
Exit Sub
End Sub
Public Function zahl_tage(tag, monat, jahr) As Long
Dim zaehler As Long
Dim tage_monat
tage_monat = Array(31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30,
zaehler = 0
For i = 1900 To (jahr - 1)
zaehler = zaehler + 365
Start
15:44


ft Visual Basic - uebung5.xls
Datei Bearbeiten Ansicht Einfügen Format Debuggen Ausführen Extras Add-Ins Fenster ?
Frage hier eingeben
Projekt - VBAProject
EuroTool (EUROTOOL.XLA)
Sun_ODFP (odfaddin.xla)
VBAProject (uebung5.xls)
DieseArbeitsmappe
Tabelle1 (Tabelle1)
Tabelle2 (Tabelle2)
Tabelle3 (Tabelle3)
uebung5.xls - Tabelle1 (Code)
(Allgemein)
berechne_Click
Worksheets("Tabelle1").Cells(7, 2).Value = "fehlerhaftes Datum"
Exit Sub
End Sub
Public Function zahl_tage(tag, monat, jahr) As Long
Dim zaehler As Long
Dim tage_monat
tage_monat = Array(31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30,
zaehler = 0
For i = 1900 To (jahr - 1)
zaehler = zaehler + 365
If schaltjahr(i) Then
zaehler = zaehler + 1
End If
Next i
For j = 1 To (monat - 1)
zaehler = zaehler + tage_monat(j - 1)
If j = 2 And schaltjahr(jahr) Then
zaehler = zaehler + 1
End If
Next j
zaehler = zaehler + tag
zahl_tage = zaehler
End Function
Public Function schaltjahr(jahr)
Start
C:\www\einf-ss12...
Modularisierung vo...
Microsoft Excel - u...
Microsoft Visual Ba...
15:44


C:\www\einf-ss12\flash\einf_course6.6.html
Favoriten
Einf
Vorgeschlagene Sites
VA
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
Start
C:\www\einf-ss12...
Modularisierung vo...
Microsoft Excel - u...
Microsoft Visual Ba...
15:45