Script **generated by TTT**

Title: Einf_HF (18.06.2012)

Date: Mon Jun 18 14:17:56 CEST 2012

Duration: 87:32 min

Pages: 32

Modularisierung von Programmen

Systematischer Aufbau von Programmen durch Zerlegung in eigenständige Einheiten.

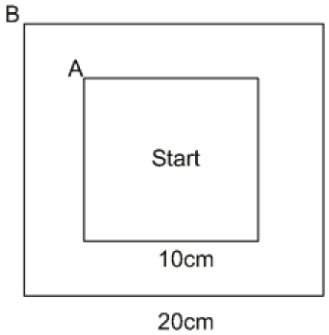
Beispielaufgabe

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)

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

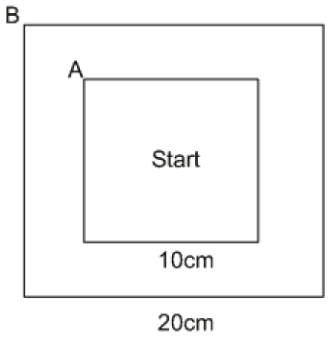
Beispielaufgabe

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)

Definition eines Moduls

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

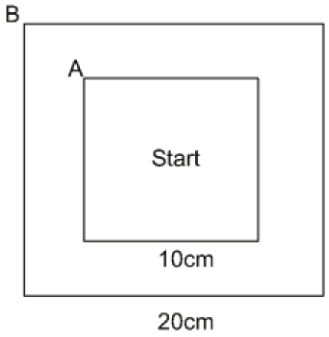
Beispielaufgabe

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

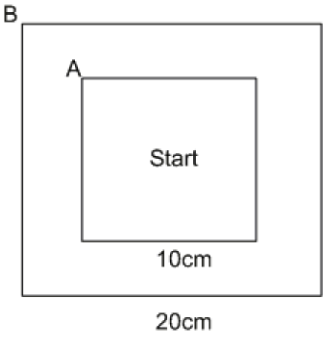
Beispielaufgabe

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

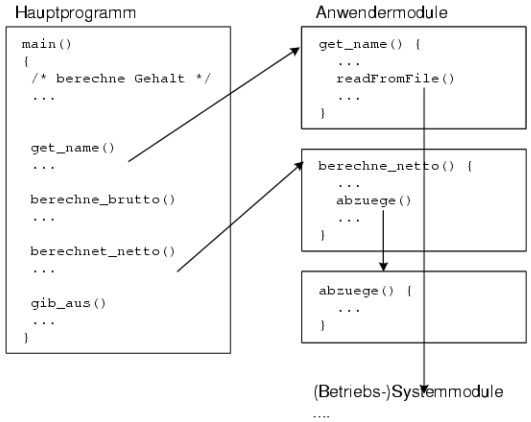
Strukturierung von Algorithmen

Zweck der Strukturierung

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

Beispielaufgabe

Hauptprogramm bestehend aus Teilmodulen



In vielen Programmiersprachen: Funktionen (Prozeduren) als Strukturierungsmittel.

Form

Jede Funktion hat die Form:

```
ergebnistyp funktionsname (parameterliste)
{
    Deklarationen der lokalen Variablen;
    Anweisungen
}
```

Beispiel in Java (Kopf einer Methode)

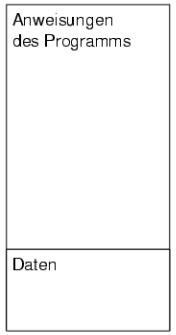
```
public int zahl_tage (int tag, int monat, int jahr)
```

Beispiel in Visual Basic (VBA)

```
public Function zahl_tage
   (tag As Integer, monat As Integer, jahr As Integer) As Long
```

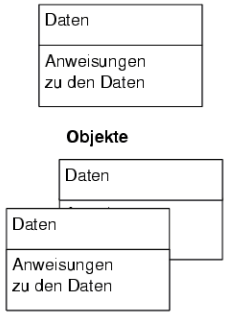
Hauptprogramm bestehend aus Teilmodulen

Prozedurales Programmieren



Anweisungen in einem großen Block (getrennt von den Daten)

Objektorientiertes Programmieren



Anweisungen jeweils bei den zugehörigen Daten

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.

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

"natürlichere" Darstellung bei bestimmten Algorithmen und Datenstrukturen.

Beispiel 'Fibonacci-Zahlen'

Wertebereich: n>=1, mit n natürliche Zahl

Algorithmus

Modulname

```
Modul fib(n)
Falls n kleiner 3
   dann Ergebnis = 1
   sonst Ergebnis = fib(n-1) + fib(n-2)
```

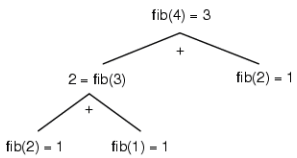
(d.h. n = 1 oder n = 2)

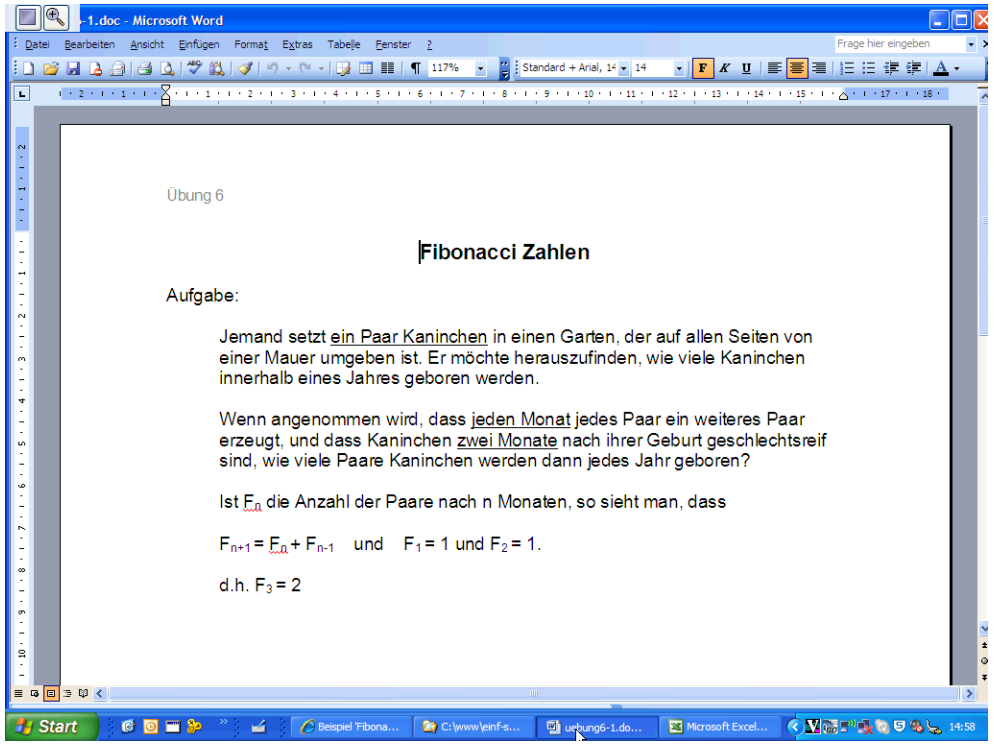
n-def

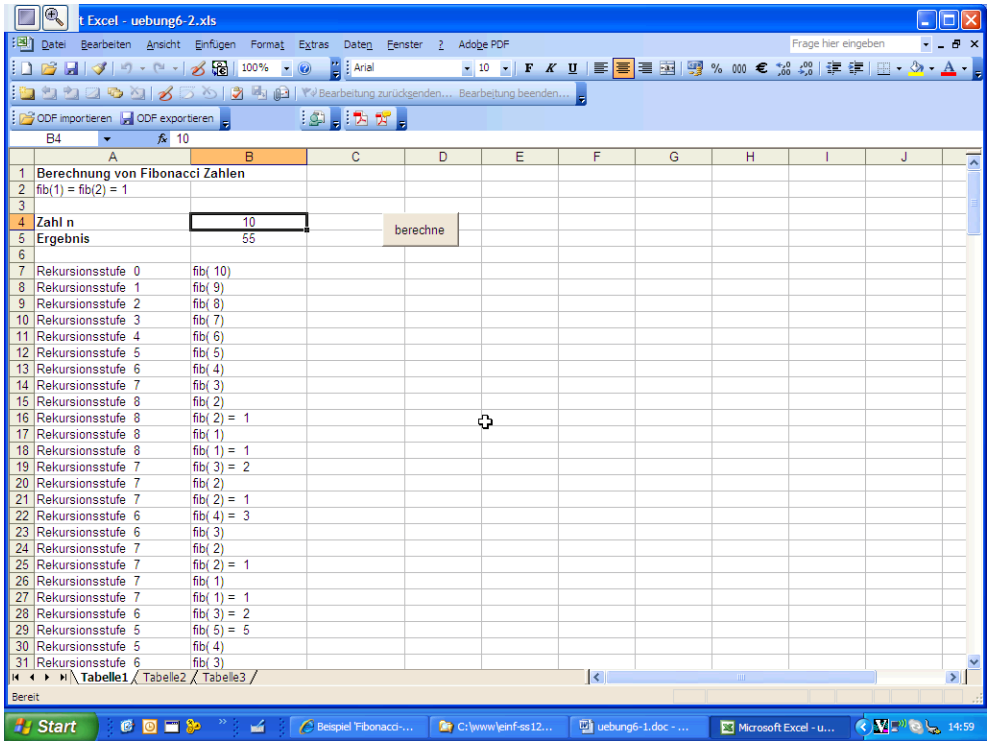
└ Aufruf des Moduls fib

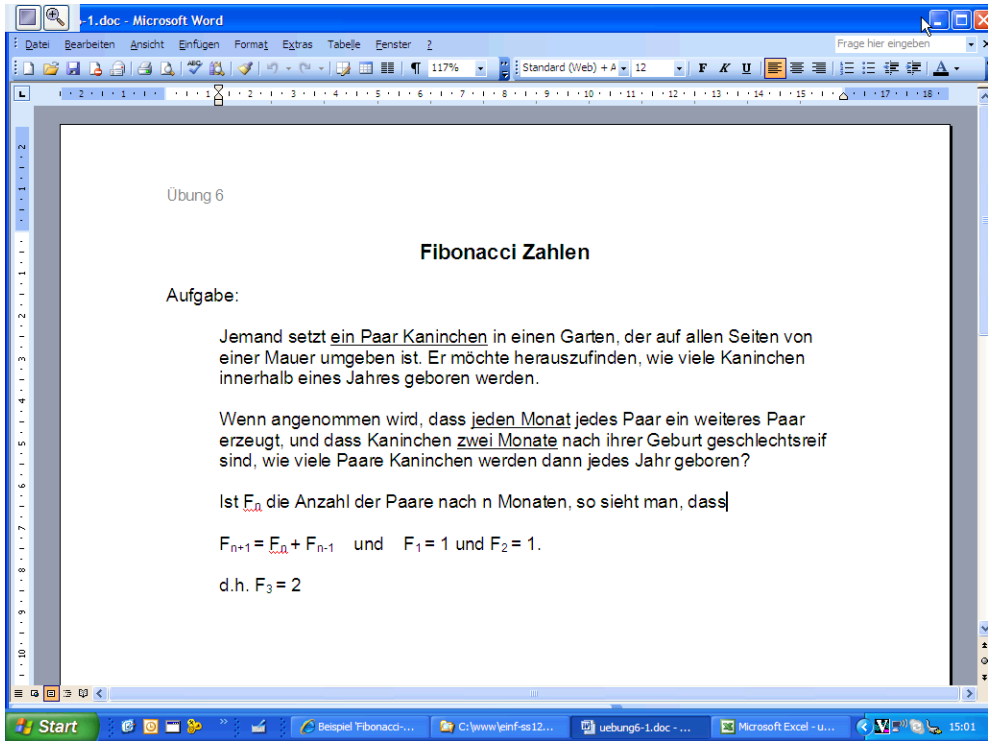
Ausführungslauf, z.B. fib(4)

```
fib(4)
- fib(3)
-- fib(2) = 1
-- fib(1) = 1
- fib(3) = 2
- fib(2) = 1
fib(4) = 3
```









Rekursion

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

"natürlichere" Darstellung bei bestimmten Algorithmen und Datenstrukturen.

Beispiel Summe

Beispiel 'Fibonacci-Zahlen'

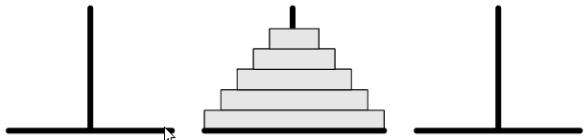
Beispiel 'Größter gemeinsamer Teiler'

Beispiel 'Türme von Hanoi'

Generated by Targeteam

Beispiel 'Türme von Hanoi'

Gegeben: drei senkrechte Stäbe, n Scheiben verschiedener Größe. Aufgabe: Turm von Scheiben von Stab auf anderen übertragen, jeweils nur eine Scheibe bewegen, nie größere Scheibe auf kleinerer.



Rekursive Lösung für n Scheiben

Aufrufschachtelung für n=3

Zahl der Schritte: 2^n -1; n ist die Anzahl der zu bewegenden Scheiben.

Anwendung in der Praxis

Lagerhaltungssystem des Hamburger Container-Hafens: Stapelung der Container entsprechend ihrer Abholung.

Animation Türme von Hanoi

Generated by Targeteam

übertrage n Scheiben von b nach a:

übertrage n-1 Scheiben von b nach c (d.h. in das Zwischenlager)

letzte Scheibe von b nach a

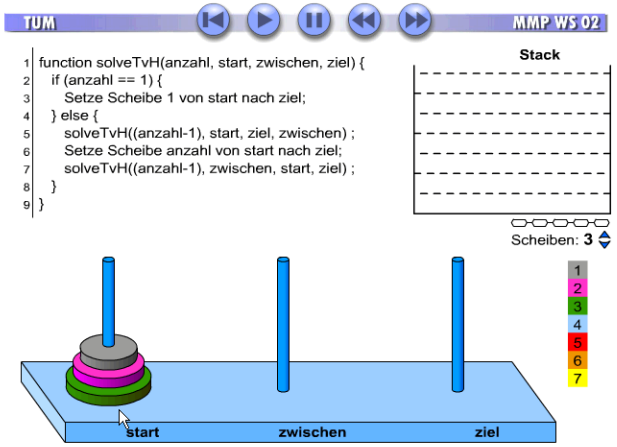
übertrage n-1 Scheiben von c nach a (d.h. vom Zwischenlager zum Ziel)

Algorithmus

Die Definition von solveTvH beinhaltet einen Aufruf von solveTvH (mit anderen Parametern)

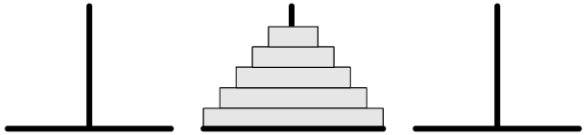
```
Modul solveTvH(n, Quelle, Ziel, Zw)
  /* Zw ist das Zwischenlager */
Falls n = 1
dann bewege Scheibe von Quelle nach Ziel
sonst
  solveTvH(n-1, Quelle, Zw, Ziel);
  bewege Scheibe von Quelle nach Ziel;
  solveTvH(n-1, Zw, Ziel, Quelle);
```


Animation Türme von Hanoi



Beispiel 'Türme von Hanoi'

Gegeben: drei senkrechte Stäbe, n Scheiben verschiedener Größe. Aufgabe: Turm von Scheiben von Stab auf anderen übertragen, jeweils nur eine Scheibe bewegen, nie größere Scheibe auf kleinerer.



Rekursive Lösung für n Scheiben

Aufrufschachtelung für n=3

Zahl der Schritte: 2^n -1; n ist die Anzahl der zu bewegenden Scheiben.

Anwendung in der Praxis

Lagerhaltungssystem des Hamburger Container-Hafens: Stapelung der Container entsprechend ihrer Abholung.

Animation Türme von Hanoi

Aufrufschachtelung für n=3

Aufruf solveTvH (3, b, a, c) führt zu den nachfolgenden Rekursionen:

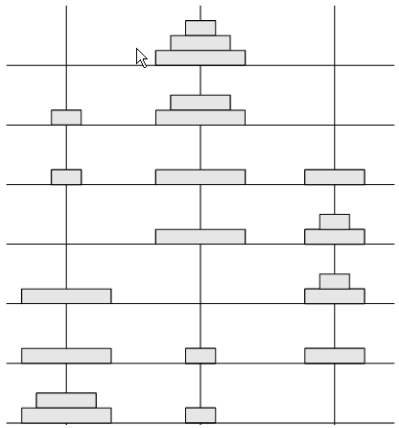
3bac (2bca (1bac (b → a); b → c; 1acb (a → c));

b → a;

2cab (1cba (c → b); c → a; 1bac (b → a)))

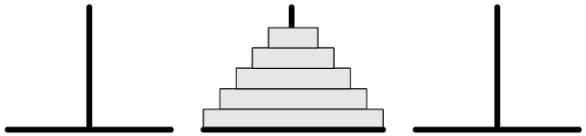
3bac entspricht dem Aufruf solveTvH (3, b, a, c)

b → a entspricht: bewege Scheibe von b nach a



Beispiel 'Türme von Hanoi'

Gegeben: drei senkrechte Stäbe, n Scheiben verschiedener Größe. Aufgabe: Turm von Scheiben von Stab auf anderen übertragen, jeweils nur eine Scheibe bewegen, nie größere Scheibe auf kleinerer.



Rekursive Lösung für n Scheiben

Aufrufschachtelung für n=3

Zahl der Schritte: 2^n -1; n ist die Anzahl der zu bewegenden Scheiben.

Anwendung in der Praxis

Lagerhaltungssystem des Hamburger Container-Hafens: Stapelung der Container entsprechend ihrer Abholung.

Animation Türme von Hanoi

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

In diesem Kapitel werden einige Klassen von Algorithmen vorgestellt, insbesondere Suchverfahren und Sortierverfahren.

- Fragestellungen des Abschnitts:
  - Welche Möglichkeiten gibt es, Datenmengen im System darzustellen?
  - Welche Möglichkeiten gibt es, in Datenmengen zu suchen?
  - Welche Möglichkeiten gibt es, Datenmengen zu sortieren?
  - Was versteht man unter der Komplexität eines Algorithmus?

Datenstrukturen

Suchverfahren

Sortierverfahren

Komplexität

Datenstruktur = Menge von Daten eines bestimmten Typs zusammen mit den auf der Menge ausführbaren Zugriffsoperationen.

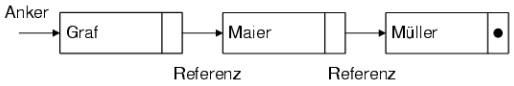
Beispiel: natürliche Zahlen zusammen mit Grundrechenoperationen

Bei Programmierung wichtiges Hilfsmittel zur Organisation der Daten für die maschinelle Verarbeitung.

Listen

Graphen

Folge von Elementen mit Reihenfolge.



Anker definiert Verweis auf 1. Element.

Referenz definiert Verweis auf nächstes Element.

Beispiele

Wort: Liste von Zeichen.

Personaldaten: Liste von Datensätzen, wobei jeder Datensatz eine Person beschreibt.

Operationen auf Listen

Erzeugen einer Liste, Einfügen eines Listenelements, Suchen eines Listenelements mit bestimmten Eigenschaften, Löschen eines Listenelements.

[Reihungen (Array)]

[Allgemeine Listen]

Reihungen (Array)

Liste mit fester Zahl von Elementen gleichen Typs, gemäß Reihenfolge numeriert.

1	Graf
2	Maier
3	Müller

Reihungen meist in Programmiersprachen direkt unterstützt.

Deklaration durch:

```
boolean buffer[500]
```


einfach verkettete Listen

Listenelement: Inhalt und Referenz auf nächstes Listenelement.

doppelt verkettete Listen

Listenelement: Inhalt und Referenzen auf nächstes und vorhergehendes Listenelement.

Beispiel

```
class elem {
    elem previous;
    char content;
    elem next;
}
```


Datenstrukturen

Datenstruktur = Menge von Daten eines bestimmten Typs zusammen mit den auf der Menge ausführbaren Zugriffsoperationen.

Beispiel: natürliche Zahlen zusammen mit Grundrechenoperationen

Bei Programmierung wichtiges Hilfsmittel zur Organisation der Daten für die maschinelle Verarbeitung.

[Listen]

[Graphen]

Datenstrukturen und Algorithmen

In diesem Kapitel werden einige Klassen von Algorithmen vorgestellt, insbesondere Suchverfahren und Sortierverfahren.

- Fragestellungen des Abschnitts:
  - Welche Möglichkeiten gibt es, Datenmengen im System darzustellen?
  - Welche Möglichkeiten gibt es, in Datenmengen zu suchen?
  - Welche Möglichkeiten gibt es, Datenmengen zu sortieren?
  - Was versteht man unter der Komplexität eines Algorithmus?