Resümee (1)

- Ziele der Vorlesung

- Algorithmus “ägyptische” Multiplikation

$$f(a,b) = \begin{cases} a & \textit{falls } b = 1 \\ f(2a\ ,\ b/2) & \textit{falls } b \textit{ gerade} \\ a + f(2a,\ (b-1)/2) & \textit{sonst} \end{cases}$$

9	5
~~18~~	~~2~~
36	1
45	

- "Einsicht" in das dahinterliegende Prinzip

- Rekursion: Reduktion auf eine einfachere Instanz des gleichen Problems

- Der Algorithmus als rekursives Java-Programm

- Terminierung, Stack-Überlauf

- Korrektheitsnachweis $\forall\ a, b \in \mathbf{N}^{+}$: $f(a,b) = a \times b$ durch vollst. Induktion (über b)

- Robustes Programmieren
  - Ausnahmen (exceptions) durch **try** und **catch** behandeln
  - Exceptions, try, catch,... siehe Kap. 2.5 Buch von M.A. Weiss

- Multiplikationsalgorithmus iterativ (while-Schleife)
  - Verifikation mittels Invarianten

Resümee (2)

- Effizienz des (rekursiven) Multiplikationsalgorithmus
  - Elementaroperationen identifizieren
  - Vergleich "altägyptische Methode" mit der Schulmethode
- Funktionales Programmieren
- Elementare Berechnungsoperationen
  - "gerade", "halbiere", "verdopple", ...
- Java
  - Unterschiede zu C++
  - Bytecode, Applets
  - Programmstruktur
  - einfache Datentypen
  - Arrays

Resümee (3)

- Java
  - Typkonvertierung
  - Ein- / Ausgabe
  - Hüllenklassen
  - Strings
  - Kommandozeilenargumente

```
String msg = "Die";
msg = msg + " sieben";
```

- Java-Klassen als Datenstrukturen
  - Beispiel-Klasse "datum" (mit Methoden *"frueher_als", "gleich"...)*
  - this
  - klassenbezogene / instanzenbezogene Variablen und Methoden
  - Zugriffseinschränkungen, "information hiding" (public, private)

- Dynamische Klassen und Referenzen
  - Erzeugen von Objekten; überladene Konstruktoren
  - Zuweisung von Referenzen
  - Vergleich von Referenzen (== , !=)

- Bäume
  - Definition, Charakterisierung
  - Wurzelbaum
  - Darstellung von Wurzelbäumen
  - Binärbaum in Array-Darstellung

Resümee (4)

- Bäume
  - Syntaxbaum
- Syntaxdiagramme
  - Beispiel: Diagramme für arithmetische Ausdrücke
  - Syntaxanalyse von 1+2*3 und (1+2)*3
  - Darstellung des Weges durch Einrücken von (Nicht-)Terminalen (d.h. Generierung eines Syntaxbaums!)
- Syntaxanalyse durch rekursiven Abstieg
  - Implementierung von Syntaxdiagrammen durch Java-Methoden
  - Parser als Java-Programm
- Erzeugen von Syntaxbäumen beim rekursiven Abstieg
- Auswertung von Operatorbäumen
- Traversieren von Bäumen in inorder und postorder
  - liefert Infix- bzw. Postfixausdruck bei Operatorbäumen
- Stack-Implementierung
- Anwendungen mit Stacks
  - Transformation Infix $\rightarrow$ Postfix
  - Auswertung von Postfixausdrücken
- Klammerchecker
  (Analyse der Blockstruktur von Programmen)

Resümee (5)

- Klammerchecker
  (Analyse der Blockstruktur von Programmen)
  - mit Stack
  - rekursiv (Nutzung des Laufzeitstacks)

- Codegenerierung für eine Stackmaschine bei Infix-Ausdrücken

- Interpreter für Infix-Ausdrücke

- Interpreter, Compiler

- Übersetzung von Java-Ausdrücken nach Bytecode
  - Java-VM als Bytecode-Interpreter

- Pakete in Java

Resümee (6)

- Pakete in Java
  - Beispiel verkettete Liste; damit Realisierung eines Stack
  - Beispiel Bruchrechnung

- Klassenhierarchie / objektorientiertes Programmieren
  - Klassifizierung und Vererbung
  - Eigenschaften abgeleiteter Klassen
  - Zuweisungskompatibilität von Variablen versch. Hierarchiestufe
  - Zugriff auf Attribute und Methoden abgeleiteter Klassen

Resümee (7)

- Abstrakte Klassen und Methoden

- Polymorphie
  - late binding
  - Beispiel: polymorphe Listen

- Generische (typunabhängige) Algorithmen
  - Sortierverfahren für beliebige total geordnete Objekte
  - Bsp.: Sortieren von Zahlen bzw. Objekten der Klasse "Studi" bzw. geometrischen Objekten

- Java: Interfaces

- Java: Exceptions
  - try, catch, throws, throw
  - Definieren eigener Ausnahmen

Resümee (8)

- Binärbäume als Referenzstrukturen

- Binäre Suchbäume
  - Aufbau, Einfügen, Suchen, Löschen
  - symmetrisches Traversieren ("inorder")
  - Sortieren mit Suchbäumen (n log n Schritten)

- Binärsuche auf arrays
  - Algorithmus iterativ und rekursiv
  - Anwendungen
  - Effizienz
  - Korrektheit: Terminierung

Resümee (9)

- Backtracking
  - systematisches ("rekursives") Durchmustern eines Problembereiches
  - Beispiel: 8-Damen-Problem

    (Denkübung: wo genau erfolgt bei der rekursiven Lösung eigentlich der Backtracking-Schritt?)

- Spieltheorie
  - endliche rein strategische 2-Personen-Nullsummenspiele mit vollständiger Information
  - Spielbäume
  - Strategien
  - Auszahlungsmatrix

- Gewinnstrategie, optimale Strategie

Resümee (10)

- Minimax-Algorithmus
  - Analogie zur Auswertung von Operatorbäumen von Ausdrücken
- Auswertung von Spielbäumen, partielle Spielbäume
- α-β-Algorithmus
  - Schnitte
  - α-β-Schranken
  - Effizienz
- Optimierte Spielbaumanalyse
  - spekulative Suchfenster, last move improvement, Nullfenster
- Pragmatisches zur Spielbaumanalyse
- Rekursives Problemlösen
- "Türme von Hanoi"-Spiel:
  - rekursive Lösung

Resümee (11)

- “Divide et impera”-Paradigma
  - Bsp.: rekursive Minimumsbestimmung

- Mergesort
  - bottom-up / top-down
  - array bzw. verkettete Liste
  - Zeitaufwand proportional zu $n \log n$ (n = Anzahl der Elemente)

- Aufwand von Algorithmen
  - Begriffe
  - O-Notation, asymptotische Zeitkomplexität
  - Beispiel: Beweis für O(n log n) Zeitkomplexität bei mergesort

Resümee (12)

- Simulation
  - Definition, Zweck, Anwendungsgebiete
  - Modell und Modellierung

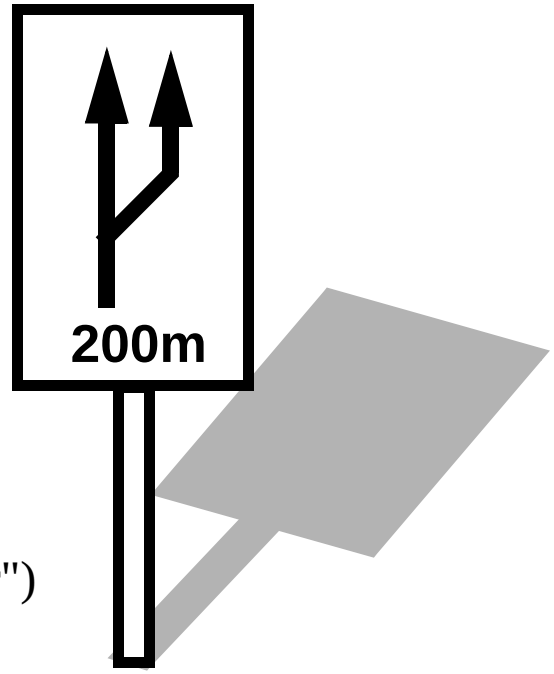
- Zeitgesteuerte Simulation ("Weizen, Mäuse, Katzen")
  - "korrekte" Umsetzung der Spezifikation in ein Simulationsprogramm
  - Grösse von Δt
  - Darstellung des Kenngrössenverlaufs
  - "Grenzen des Wachstums" (Weltmodell)

- Ereignisgesteuerte Simulation
  - Vorantreiben der Simulation (und Simulationszeit) durch Ereignisse
  - Beispiel Call-Center eines Reisebüros

Resümee (13)

- Ereignisbegriff
  - Modellierung quasi-paralleler Abläufe durch Ereignisse
- Simulatorzyklus; Ereignisroutinen; Ereigniseinplanung
- Ereignislisten als "abstrakte Datentypen" (insert, get_min)
  - Realisierung als sortierte Liste
  - als unsortierte Liste
  - als Heap ("partiell sortiert"!)
- Heaps
  - Definition (Binärbaum...)
  - Realisierung als array
  - Implementierung (insert, get_min) in Java
- Anwendungsmöglichkeiten von Heaps
  - Heapsort (mit garantiert n log n Zeitaufwand)

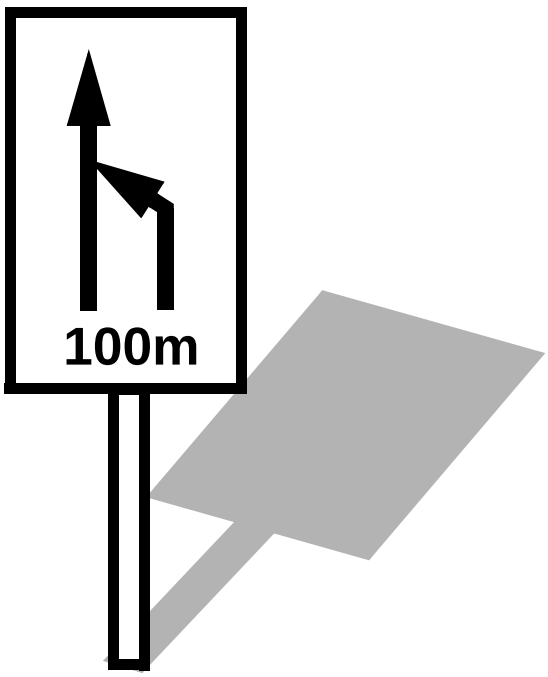
- Prozesse
  - Multitasking
  - Prozesszustände
  - Kontextwechsel
- Threads in Java
  - Erzeugen von Threads
  - Beispiel für parallele Threads ("HinHer")



Resümee (14)

- Methoden zur Kontrolle von Threads

- Scheduling von Threads
  - Prioritäten
  - "Simulation" eines Zeitscheiben-Schedulers



- Multithreading: race conditions

- Atomarität
  - Synchronisationsprobleme bei Nicht-Atomarität
  - Problematik von Lösungen mittels Thread-Prioritäten

- Kritische Abschnitte
  - Safety, Liveness, Fairness

- Synchronized-Konstrukt
  - Realisierung kritischer Abschnitte
  - synchronized-Konstrukt