Vorlesung Programmierung (Wintersemester 2014/2015)

1 Ziele der Vorlesung

- Vermittlung grundlegender Techniken zur Programmierung
- Gefühl für guten Stil entwickeln (Programme so schreiben, dass andere diese lesen können)
- Techniken zur Beherrschung komplexer Zusammenhänge (Abstraktion)
- Denken in Strukturen und Algorithmen
- Systematische Entwicklung von Programmen (Datenanalyse, Testfälle, Konstruktionsschemata)
- Grundkenntnisse verschiedener Programmierparadigmen (funktional, imperativ, objektorientiert)

2 Verwendete Programmiersprachen

Im wesentlichen **Scheme** (in der Syntax von **DrRacket**), wegen

- einfacher Syntax
- einfacher Umsetzung verschiedener Programmierparadigmen
- guter, frei verfügbarer Programmierumgebung (DrRacket)

3 Detaillierter Vorlesungsverlauf

24.10.: Einführung, Organisatorisches, grundlegende Begriffe wie Programmieren, Algorithmus, (partielle/totale) Korrektheit, Programmiersprache (Syntax, Semantik, Pragmatik), Stichworte zu Scheme, Arbeiten mit DrScheme

27.10.: Programmelemente: Ausdrücke (Zahlen, Kombination, Präfix-Notation), Abstraktion durch Benennung von Objekten (`define`), Umgebung, Termauswertung (informell), Berechnungsvorschriften und Prozeduren (`lambda`), Substitutionsmodell

31.10.: Kommentare, Entwurfsrichtlinie „Kommentar vor Umsetzung“, Verträge, Signaturen, Sorten, Entwurfsrichtlinie „Signatur vor Umsetzung“, Testfälle (`check-expect`), Entwurfsrichtlinie „Testfälle vor Umsetzung“, Überdeckungswerkzeug von DrRacket, Entwurfsrichtlinie „Ausreichende Testfälle“, bedingte Ausdrücke (`cond`, `if`, `and`, `or`, `not`), Stringkonstanten, Sortenkonstruktion mit `one-of`, Beispiel Aggregatzustand von Wasser, Konstruktionsschablone Fallunterscheidung

3.11.: Quadratwurzelberechnung nach Newton, Testfälle `check-within`, top-down/bottom-up Entwurf, Bedeutung von Namen, lexikalische Bindung, Prozeduren und Prozesse, Fakultätsfunktion, linearer rekursiver Prozess am Beispiel der Fakultätsfunktion, linearer iterativer Prozess am Beispiel der Fakultätsfunktion

7.11.: Baumrekursion am Beispiel der Fibonacci-Zahlen, Größenordnungen, O-Notation, Potenzrechnung (lineare und logarithmische Laufzeit), ggT (Euklidscher Algorithmus),

10.11.: Primzahltest durch Teilerprüfung, probabilistischer Primzahltest nach Fermat, Motivation zum Thema Datenabstraktion, Zusammengesetzte Daten am Beispiel von Schokokeksen, Konstruktoren, Selektoren

14.11.: Konstruktionsschablone „Zusammengesetzte Daten als Argumente“ Konstruktionsschablone „Zusammengesetzte Daten als Ausgabe“, Definition von Records (`define-record-procedures`), Beispiel rationale Zahlen, Operationen auf zusammengesetzten Daten Schichtenentwurf von Programmen, Module (`provide`, `require`)

17.11.: Definition gemischter Daten (`(signature (mixed ...))`), Konstruktionsschablone „Gemischte Daten als Argumente“, Darstellung von Listen am Beispiel von Zahlenlisten (Sorte `empty-list`, `empty`, `empty?`, Sorte `nonempty-list`, `cons`, `cons?`, `first`, `rest empty?`, `cons?`), Kasten-Zeiger-Darstellung, Listenoperation „Summe aller Elemente“, Konstruktionsschablone „Listen als Argumente“

21.11.: Definition von String-Listen, Definition beliebiger Listen, parametrisierte Records, Signaturkonstruktor, Signaturvariable, parametrisierte Paare, parametrisierte Listen in DrRacket, Listenoperation „Länge einer Liste“ und `concatenate`, Listenoperation `n-th`

24.11.: Baumstrukturen, Operation „Anzahl der Blätter“, symbolisches Differenzieren (Darstellung von Funktionsausdrücken und Implementierung ohne Vereinfachung von Funktionsausdrücken)

28.11.: Symbolisches Differenzieren mit Vereinfachung von Funktionsausdrücken, Schnittstelle für Mengen, Implementierung von Mengen durch ungeordnete Listen, Implementierung von Mengen durch geordnete Listen, Idee der Implementierung von Mengen durch geordnete Binärbäume

1.12.: Implementierung von Mengen durch geordnete Binärbäume Huffman-Bäume, Präfixcode, Idee der Konstruktion von Huffman-Bäumen, Datenstrukturen für Huffman-Bäume, Decodierung mit Huffman-Bäumen

5.12.: Konstruktion von Huffman-Bäumen, Prozeduren höherer Ordnung, Prozeduren als Parameter, lokale Definitionen (Sonderform `letrec`), Gültigkeitsbereich/Geltungsbereich, lexikalische Bindung, Benutzung der lexikalische Bindung

8.12.: universelle Berechnungsverfahren auf Datenstrukturen `map-list` universelle Berechnungsverfahren (Nullstellenbestimmung durch Intervallhalbierung), Prozeduren als Ergebnis (Differenzierung von Funktionen, Funktionskomposition), Implementierung von Listen durch Funktionen höherer Ordnung
Berechnungen: Programmsignatur, Terme, Programm

12.12.: Substitution, Position, Teilterm, Ersetzung, Berechnungsschritt, Normalform, strikte und nicht-strikte Auswertung
Abstrakte Datentypen: Schnittstelle, Gesetze

15.12.: Implementierung eines ADT, Beispiele rationale Zahlen und Listen. Einführung in die strukturelle Induktion Beweistechnik strukturelle Induktion (am Beispiel von `length`), Korrektheit von `append` mittels struktureller Induktion

9.1.: Eigenschaften, Sonderformen `for-all` und `==>`, Test `check-property`, Testen von ADT-Gesetzen am Beispiel rationaler Zahlen, Mengen und Listen
Veränderbare Zustände, Sonderformen `set!` und `begin`, Beispiel Abheben vom Bankkonto, „Abhebungsprozessoren“ für mehrere Konten

12.1.: Bankkonto mit Nachrichtenweitergabe für Abheben und Einzahlen, Diskussion der Probleme der Zuweisung (Substitutionsmodell nicht mehr anwendbar, Auswertungsreihenfolge relevant, referentielle Transparenz, Wertgleichheit vs. Objektidentität), Modularität durch Zuweisung am Beispiel von Zufallszahlen und Monte-Carlo-Simulation; Umgebungsmodell: Umgebung, Rahmen, Bindung

16.1.: Umgebungsmodell: Auswertungsregeln für Variablen und Kombinationen, Prozeduranwendung, lambda-Ausdrücke, `define`, `set!`, `letrec` und Konstruktorprozeduren; Beispiele für das Umgebungsmodell: Darstellung einfacher Prozeduren, lokale Zustände, lokale Definitionen;
Mutatoren, veränderbare Datenobjekte

19.1.: Mutatoren für veränderbare Listen (`set-mfirst!`, `set-mrest!`), Implementierung veränderbarer Listen (`mcons`) durch Nachrichtenweitergabe und `set!` Umwandlungsprozedur `mlist->list`, Beispiel `append!`, Erzeugung zyklischer Strukturen, Mutatoren für beliebige Strukturen (`define-record-procedures-2`), Implementierung veränderbarer Bankkonten hiermit, Arbeiten mit mehreren veränderbaren Strukturen (Geldtransfer), Implementierung veränderbarer parametrischer Listen mit `define-record-procedures-parametric-2`, Structure Sharing und Identität, Bedeutung von `eq?`, Idee von Warteschlangen

23.1.: Implementierung von Warteschlangen; Simulation digitaler Schaltkreise, Schnittstellen, Schnittstelle von Drähten, Implementierung elementarer Gatter (Inverter, Und-Gatter, Oder-Gatter), Implementierung von Drähten als Objekte

26.1.: Struktur und Implementierung des Zeitplans, Sonden, Simulationsablauf; Probleme bei Zustandsänderungen in nebenläufigen Systemen

30.1.: Semaphore, Synchronisation von Zustandsänderungen mit Semaphoren, Verklemmungsproblem bei Semaphoren; Bemerkungen zu unterschiedlichen Programmiermodellen; Zusammenfassung
Einführung zur metalinguistischen Abstraktion, Evaluator; Syntaxbeschreibung mit EBNF

2.2.: Quotierung, Programme als Daten, Evaluator für MiniScheme: Syntax und abstrakte Syntax, Operationen und Datenstrukturen des Umgebungsmodells

6.2.: Implementierung der Operationen auf Umgebungen, Implementierung der Auswertungsregeln des Umgebungsmodells, Eingabeschleife, Beispielablauf; Implementierung der Auswertung in Normalordnung im Evaluator (Einführung von verzögerten Argumenten)
Zusammenfassung