

9. Zusammenfassung

Themen dieses Kapitels:

- Zusammenfassung der Themen der Vorlesung
- Zusammenfassung der angestrebten Kompetenzen

Zusammenfassung der behandelten Themen

allgemeine Spracheigenschaften:

- Grundsymbole, Syntax, statische Semantik, dynamische Semantik
- kontext-freie Grammatik, Ableitungsbäume, EBNF-Notation, Ausdruckgrammatiken
- Gültigkeit von Definitionen, Verdeckungsregeln
- Variablenbegriff, Lebensdauer, Laufzeitkeller, statischer Vorgänger, Umgebungen
- Datentypen, abstrakte Typkonstrukturen, rekursive und parametrisierte Typen
- konkrete Ausprägungen der abstrakten Typkonstrukturen in Programmiersprachen
- Parameterübergabe: call-by-value, call-by-reference, call-by-result, call-by-value-and-result

Funktionale Programmierung:

- Rekursionsparadigmen: Induktion, Funktionen über rekursiven Datentypen
- Rekursionsformen: End-Rekursion, Zentral-Rekursion,
- Technik „akkumulierender Parameter“, Funktionen über Listen
- Berechnungsschemata mit Funktionen als Parameter; Currying

Logische Programmierung:

- Klauselformen: Fakt, Regel, Anfrage; prädikatenlogische Bedeutung
- Interpretationsschema: Backtracking, Suchreihenfolge
- Unifikation von Termen: Anwendbarkeit von Klauseln, Bindung von Werten an Variable
- Prolog-Notation

Zusammenfassung der angestrebten Kompetenzen (1)

1. Einführung

- Wichtige Programmiersprachen zeitlich einordnen
- Programmiersprachen klassifizieren
- Sprachdokumente zweckentsprechend anwenden
- Sprachbezogene Werkzeuge kennen
- Spracheigenschaften und Programmeigenschaften in die 4 Ebenen einordnen

2. Syntax

- Notation und Rolle der Grundsymbole kennen.
- Kontext-freie Grammatiken für praktische Sprachen lesen und verstehen.
- Kontext-freie Grammatiken für einfache Strukturen selbst entwerfen.
- Schemata für Ausdrucksgrammatiken, Folgen und Anweisungsformen anwenden können.
- EBNF sinnvoll einsetzen können.
- Abstrakte Syntax als Definition von Strukturbäumen verstehen.

Zusammenfassung der angestrebten Kompetenzen (2)

3. Gültigkeit von Definitionen

- Bindung von Bezeichnern verstehen
- Verdeckungsregeln für die Gültigkeit von Definitionen anwenden
- Grundbegriffe in den Gültigkeitsregeln von Programmiersprachen erkennen

4. Variable, Lebensdauer

- Variablenbegriff und Zuweisung
- Zusammenhang zwischen Lebensdauer von Variablen und ihrer Speicherung
- Prinzip des Laufzeitkellers
- Besonderheiten des Laufzeitkellers bei geschachtelten Funktionen

Zusammenfassung der angestrebten Kompetenzen (3)

5. Datentypen

5.1 Allgemeine Begriffe zu Datentypen

- Typeigenschaften von Programmiersprachen verstehen und mit treffenden Begriffen korrekt beschreiben
- Mit den abstrakten Konzepten beliebig strukturierte Typen entwerfen
- Parametrisierung und generische Definition von Typen unterscheiden und anwenden

5.2 Datentypen in Programmiersprachen

- Ausprägungen der abstrakten Typkonzepte in den Typen von Programmiersprachen erkennen
- Die Begriffe Klassen, Typen, Objekte, Werte sicher und korrekt verwenden
- Die Vorkommen von Typkonzepten in wichtigen Programmiersprachen kennen
- Speicherung von Reihungen verstehen

Zusammenfassung der angestrebten Kompetenzen (4)

6. Funktionen, Parameterübergabe

- Funktionen, Aufrufen und Parameterübergabe präzise mit treffenden Begriffen erklären können
- Die Arten der Parameterübergabe unterscheiden und sinnvoll anwenden können
- Die Parameterübergabe wichtiger Sprachen kennen

7. Funktionale Programmierung

- Funktionale Programme unter Verwendung treffender Begriffe präzise erklären
- Funktionen in einfacher Notation von SML lesen und schreiben
- Rekursionsparadigmen Induktion, Rekursion über Listen anwenden
- End-Rekursion erkennen und Programmieretechnik „akkumulierender Parameter“ anwenden
- Berechnungsschemata mit Funktionen als Parameter anwenden
- Programmieretechnik „Currying“ verstehen und anwenden

Zusammenfassung der angestrebten Kompetenzen (5)

8. Logische Programmierung

- Kleine typische Beispiele in Prolog-Notation lesen, verstehen und schreiben
- Interpretationsschema und prädikatenlogische Grundlagen verstehen
- Unifikation zum Anwenden von Klauseln einsetzen
- Anwendungen wie die Symbolische Differentiation verstehen