- **Principles of Programming Languages (PoPL)**
- **Zeitpunkt:** August 2013
- **Dauer:** 30 Minuten
- **Prüfer:** Ronald Veldema
- **Beisitzer:** Georg Dotzler
- **Ergebnis:** Top, faire Benotung

Vorbemerkung
Syntax sowie 99% der Codebeispiele vollkommen unwichtig (hab keine 10 Zeilen geschrieben)!
Das A und O ist die Idee. Solange man das Konzept dahinter gut erklären kann, gerne auch an selbst gewählten Beispielen, hat man keinerlei Probleme!
Zügiges Vorankommen ist empfehlenswert. Dadurch zeigt sich schnell man hat Ahnung und falsche Antworten fallen nicht allzu sehr ins Gewicht, da sie gegen Ende hin evtl. durch „Bonusfragen" ausgeglichen werden können sollte man mit dem eigentlichen Stoff schon durch sein (zumind. hatte ich das Gefühl)!

Was ist ihre Lieblingssprache?
C++

OK. Wie kann man in C++ dynamisch den Typ feststellen?
Bzw. was ist überhaupt ein dynamischer Typ und ein statischer Typ?

- statischer Typ
  - × steht zur Übersetzungszeit fest
  - × kann sich nicht mehr ändern
- dynamischer Typ
  - × BaseClass *bc;
    bc = new DerivedClass();
  - × grundsätzlich könnte bc zur Laufzeit Referenz auf BaseClass oder DerivedClass enthalten

Ermittlung des dynamischen Typs mittels dynamic_cast<>().

In Java gibt es *Introspection*, was ist das, was kann man damit machen?
Damit kann man zur Laufzeit Metainformationen von Objekten auslesen, z.B. alle öffentlichen Funktionen.

Was sind *delegates*?
delegates sind Funktionspointer in C#, repräsentiert durch Objekte.
Sie sind mächtiger als klassische C-Funktionspointer, da sie explizit zu einer Funktion erzeugt werden und somit nachträglich nicht mehr auf eine andere Funktion zeigen können.

Was ist p*attern matching*?
Ist z.B. bei regulären Ausdrücken ein Basismechanismus, mit dem man nach benutzerdefinierten Kriterien einen String überprüfen kann.
Wird aber auch bei switch-case-Anweisung eingesetzt, um zu überprüfen welcher Fall angesprungen werden soll.

Imperative und funktionale Sprachen. Was ist das und worin unterscheiden sie sich?
Imperative Sprachen sind vergleichbar mit einem „Bauplan", Anweisung nach Anweisung.
Funktionale Sprachen verfügen über keine globalen Variablen und keine Seiteneffekte.
Bei funktionalen Sprachen spielt auch referenzielle Transparenz eine zentrale Rolle.
In funktionalen Sprachen erhält man das Resultat eig. immer abhängig von Funktionsargument.

Wie ist das mit IO in funktionalen Sprachen, z.B. printf(„Hello“)?
Unterschiedliche Möglichkeiten:

- referenzielle Transparenz für bestimmte Methoden aufheben
- *lazy evaluation* ausnutzen und IO Aufgaben ans Ende (zeitlich) des Programms verlagern, sprich aufschieben so lange wies geht

Ich möchte nun, dass Sie in einer funktionalen Sprache die Länge einer Liste berechnen.
In Miranda:
len x = 0 , if x == [] then 0
len x = 1 + len(tl(x)) , otherwise
Hier habe ich eig. 1:1 das Beispiel aus dem Skript gebracht. Hat aber wohl nicht gepasst. Er wollte hier auf *pattern matching* hinaus. Zusätzlich sollte ich erklären was tl() macht und diese Funktion ebenfalls hinschreiben.

Logische Sprachen. Was ist der Unterschied zu funktionalen Sprachen?
Aus einem Problem wird eine Datenbasis erstellt, bestehend aus Regeln und Fakten.
Programm findet bei einer Anfrage die Lösung quasi selbstständig.

Wer ein Fahrrad besitzt ist reich. Ronald und Georg besitzen Fahrräder. Formulieren Sie das in Prolog.
reich(X) :- fahrrad(X)
fahrrad(ronald).
fahrrad(georg).
?- reich(R).

georg und ronald sind Konstanten. fahrrad(X) sind Tatsachen. reich(X) :- fahrrad(X) ist eine Regel.

Ich möchte nun wissen, wer alles reich ist. Was passiert da?
R wird mit X unifiziert/gebunden. R wird mit ronald unifiziert und liefert erste Lösung.
Zweite Lösung durch *backtracking*. R wird ungebunden von ronald und mit georg unifiziert, liefert zweite Lösung.

Ich möchte nun, dass Sie in einer logischen Sprache die Länge einer Liste berechnen.
len [] = 0
len [_|X] = 1 + len(X)
Auch hier wollte er auf *pattern matching* sowie das Schlüsselwort *is* hinaus und die Tatsache das die letzte Variable immer das Resultat enthält. Die Lösung war wohl falsch. Ich habs nicht hinbekommen, konnte aber erklären wie es ablaufen muss.
len [] = 0
len[X,R] = R1 is [_|X], len[X, R1] + 1 (oder so in der Art^^, logische Programmierung ist nicht meine Stärke…)

Was ist Datenparallelismus?
Datenparallelismus != Datenflussprogrammierung! Bei Datenparallelismus wird ein Array von Funktionen auf einen Datensatz angewandt.

Was ist *Constraint Programming*?
Beispiel:
cond(A,B,C) = 7; C [2..4];
if A == true then B else C;
Da cond(A,B,C) = 7, und C [2..4] muss B = 7 sein und somit A = true!
Allgemein: Typisch für domänenspezifische Sprachen wo es Sinn macht Wertebereiche von Variablen für die Domäne anzupassen.

Was ist ein Präprozessor?
Transformiert Code noch vor dem Compiler.

Was macht der Präprozessor daraus?
#define x 8
if (x > x.x) x++
#define x 8
if (8 > 8.8) 8++

***Transactional Programming*, Transaktionen, wie funktioniert das?**
Transaktionen werden zur Synchronisation eingesetzt.
Synchronisation benötige ich bei parallelen Abläufen, bspw. wenn Leser und Schreiber zum gleichen Zeitpunkt auf einen Speicherbereich zugreifen.
Unterschiedliche Implementierungen: *mutual exclusion* (lock / unlock), *test-set-rollback/commit*.

Wie funktioniert das, test-set-rollback/commit?
Wert wird gelesen. Wert wird zwischengespeichert. Wert wird erhöht.
Danach wird geprüft, ob globaler Wert gleich dem ursprünglich gelesenem ist.
Ja: commit; Nein: rollback

Wie ist das bei GoogleGo mit Parallelismus?
Go nutzt zur Synchronisation/Kommunikation zwischen *goroutinen channel*.
goroutinen entsprechen Threads.
channel sind bei Go bidirektional, sprich Besitzer des *channel* können sowohl schreiben als auch lesen.

Sie haben ein Array von Zahlen und möchten davon das Maximum bestimmen.
Wie machen Sie das mit Go?
Beispiel:
int zahlen[100];
Entscheidung des Programmierers welche Lastverteilung er vornehmen, sprich wie viele *goroutinen* er zur Berechnung des Maximums instanziieren möchte, z.B. fünf.
Danach die fünf Teile des Arrays auf die *goroutinen* aufteilen.
channel erstellen, der solange blockiert bis er fünf Maxima von den fünf *goroutinen* zurückbekommt.
Aus den fünf Maxima das Ergebnis ermitteln.

Was ist a*spect-oriented programming*?
Programm wird durch Skeleton/Modell repräsentiert.
In dieses Model kann man an unterschiedlichen Stellen (pointcuts) und zu unterschiedlichen Zeitpunkten (before, after, around) Code einfügen/einpatchen.