Proseminar *Programming Languages*

Einführung

Sommersemester 2023; 5. April 2023

Thomas Noll et al.
Software Modeling and Verification Group
RWTH Aachen University

`https://moves.rwth-aachen.de/teaching/ss-23/proglang/`

Software Modeling and Verification Chair

RWTH AACHEN UNIVERSITY

Übersicht

Programmiersprachen

Gängige Programmiersprachen

- Im Informatikstudium: Java, C, C++, Python, JavaScript, Haskell, Prolog, Swift, C#, ...
- Laut `https://t3n.de/news/it-jobs-2022-entwickler-karriere-programmiersprachen-1520923/`: „Zwei Drittel aller Entwicklerjobs für Javascript, Python und Java“

Seminarthema

Ziel dieses Proseminars ist die Vorstellung verschiedener Programmiersprachen,

- die historisch bedeutsam sind und/oder
- interessante Sprachkonzepte (erstmals) implementieren.

(Hierbei werden bewusst Sprachen weggelassen, die bereits im Rahmen des allgemeinen Informatikstudiums besprochen werden.)

Fragestellungen

- Programmierparadigma (prozedural/objektorientiert/funktional/...)
- Historische Entwicklung
- Typisches Anwendungsgebiet (industriell/akademisch/ausbildungsbezogen)
- Abstraktionsgrad (high-level/maschinennah)
- Typische Sprachkonstrukte (Syntax und [formal spezifizierte?] Semantik)
- Typsystem (streng/schwach, statisch/dynamisch, . . .), Datenstrukturen, Speicherverwaltung
- Unterstützung nebenläufiger oder paralleler Programmierung
- Kostenlose/kommerzielle Implementierungen (Interpreter/Compiler)

Zielsetzung

Ziele des Proseminars

- Selbstständiges Einarbeiten in ein neues Thema
- Literaturrecherche
  - Einstiegsreferenz auf Webseite
  - Schulung in Infromatikbibliothek
- Verständliches Präsentieren
- Kurzdarstellung des Inhalts in einer wissenschaftlich orientierten Ausarbeitung

(Mögliche) Bearbeitung in Zweiergruppen

- Vortrag: gemeinsamer Foliensatz (aber zwei separate Vorträge)
- Ausarbeitung: gemeinsame Anfertigung

Anforderungen Vortrag

Vortrag

- 20-minütiger Vortrag (zwei pro Thema)
- Zielgruppengerechte Präsentation der Inhalte
- übersichtliche Folien:
  - $\leq$ 15 Textzeilen
  - sinnvoller Einsatz von Farben
- LaTeX/`beamer`-Vorlage wird zur Verfügung gestellt
- Vortrag in Deutsch oder Englisch

Anforderungen Ausarbeitung

Ausarbeitung

- Selbstständiges Verfassen einer Zusammenfassung von gut 5 Seiten
- Vollständiges Literaturverzeichnis
- Korrektes Zitieren
- Plagiarismus:
  Die nicht gekennzeichnete übernahme fremder Inhalte führt zum sofortigen Ausschluss.
- Schriftgröße 12pt, übliche Seitenränder
- Titelseite mit Thema, Titel Proseminar, Semester, Name, Datum
- LaTeX-Vorlage wird zur Verfügung gestellt
- Sprache Deutsch oder Englisch
- Korrekte Sprache wird vorausgesetzt

Übersicht

Themenauswahl

Verfahren

- Themenliste wurde/wird ausgehändigt
- Priorisierte Auswahl
- ggf. Angabe Wunschpartner(in)
- Abgabe bis (spätestens) Dienstag, 11. April per Mail/im Sekretariat
- Wir bemühen uns (ohne Garantie) um ein „optimales“ Matching
- Zuordnung der Themen und Betreuer bis Ende nächster Woche online

Rücktritt vom Proseminar

- Bis zu drei Wochen nach Themenvergabe: ohne Folgen
- Danach: Fehlversuch

Bibliothekseinführung

Einführung in die Literaturrecherche

- Einweisung in themenspezifische Literaturrecherche
- Dauer: ca. zwei Stunden
- Teilnahme für BSc-Studierende verpflichtend
- Bedarf bitte auf Themenblatt vermerken
- Termine zur Auswahl:
  - Freitag, 14. April 2023 10:30 Uhr
  - Montag, 17. April 2023 12:00 Uhr
  - Mittwoch, 19. April 2023 16:30 Uhr
  - Montag, 24. April 2023 16:00 Uhr
  - Freitag, 5. Mai 2023 10:30 Uhr
- Mögliche Termine auf Themenliste vermerken, Bestätigung per Mail

Deadlines

Folgende Termine sind verpflichtend:

- 11. April: Frist für Themenauswahl
- 5. Mai: letzte Rücktrittsmöglichkeit
- 15. Mai: Vorlage des detaillierten Vortragsstruktur
  - nicht: „1. Einleitung/2. Hauptteil/3. Zusammenfassung“
  - sondern: detaillierte Strukturübersicht (Abschnittsüberschriften, wesentliche Definitionen/Aussagen/Beispiele, ...)
- 12. Juni: vollständige Fassung der Vortragsfolien
- 03./04./05. Juli (?): Blockseminar
- 17. Juli: Einreichung der Ausarbeitung

Übersicht

Prozedurale Programmiersprachen

1. Algol („Algorithmic Language“, 1958)
   - systematische Unterstützung strukturierter Programmierung (geschachtelte Funktionsdefinitionen)

```
BEGIN DISPLAY("Hello, World!") END.
```

2. BASIC („Beginners' All-purpose Symbolic Instruction Code“, 1964)
   - einfache, für Anfänger geeignete Programmiersprache (mit Zeilennummern und Sprunganweisungen)

```
10 PRINT "Hello, World!"
```

3. COBOL („Common Business-Oriented Language“, 1959)
   - Programmierung kaufmännischer Anwendungen, stark an natürliche Sprache angelehnt

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO-WORLD.
PROCEDURE DIVISION.
    DISPLAY 'Hello, World!'.
    STOP RUN.
```

4. Fortran („Formula Translator/Translating System“, 1957)
   - für numerische Berechnungen in Wissenschaft, Technik und Forschung; gilt als erste (jemals realisierte) höhere Programmiersprache

```
program Hello
  print *, "Hello, World!"
end program Hello
```

Prozedurale Programmiersprachen (Fortsetzung)

5. Pascal (benannt nach Blaise Pascal, 1970)
   - für Ausbildungszwecke (Strukturierung, starke Typisierung)

```
program hello(output);
begin
    writeln('Hello, World!');
end.
```

6. Modula-2 (1978)
   - Weiterentwicklung von Pascal um Modularisierungskonzepte

```
MODULE PrintHelloWorld;
FROM InOut IMPORT WriteString, WriteLn;
BEGIN
        WriteString('Hello world!');
        WriteLn;
END PrintHelloWorld.
```

7. PL/I („Programming Language One“, 1964)
   - entwickelt in 1960er Jahren als allgemeine Programmiersprache für alle Anwendungsgebiete

```
HELLO:  PROCEDURE OPTIONS (MAIN);
        FLAG = 0;
LOOP:   DO WHILE (FLAG = 0);
          PUT SKIP DATA('HELLO WORLD!');
        END LOOP;
END HELLO;
```

Objektorientierte Programmiersprachen

8. Ada (benannt nach Ada Lovelace, 1977)
   – strukturierte Programmiersprache mit statischer Typisierung für eingebettete und Echtzeitsysteme, später erweitert um Objektorientierung

```
with Ada.Text_IO;
procedure Hello is
begin
   Ada.Text_IO.Put_Line ("Hello, World!");
end Hello;
```

9. Eiffel (benannt nach Gustave Eiffel, 1986)
   – konsequent objektorientierter Ansatz für Programmieren im Großen

```
class
    HELLO_WORLD
create
    hello
feature
    hello
        do
            io.put_string ("Hello World!")
        end
end
```

Objektorientierte Programmiersprachen (Fortsetzung)

10. Simula („Simulation Programming Language“, 1962)
    - gilt als erste objektorientierte Programmiersprache; basiert auf Algol

```
Begin
    while 1=1 do begin
        outtext("Hello World!");
        outimage;
    end;
End;
```

11. Smalltalk (1972)
    - Nachfolger von Simula, rein objektorientiert

```
Transcript show: 'Hello World!'.
```

Funktionale Programmiersprachen

12. APL („A Programming Language“, 1966)
    - algorithmische Notation u.a. für Mathematikunterricht

```
'Hello World!'
```

13. Erlang (Verweis auf Agner Krarup Erlang und Abkürzung „Ericsson Language“, 1987)
    - zugeschnitten auf Anwendungen in der Telekommunikation (Parallelität, Verfügbarkeit, Fehlertoleranz, hot code replacement)

```
-module(hello).
-export([hello_world/0]).
hello_world() -> io:fwrite("hello, world\n").
```

14. Lisp („List Processing“, 1958)
    - angelehnt an ungetypten Lambda-Kalkül

```
(format t "Hello, World!")
```

15. ML („Meta Language“, 1984)
    - funktionale Programmiersprache (statische Typisierung, Polymorphie, automatische Speicherbereinigung) mit imperativen Konstrukten

```
print "Hello, World\n";
```

16. SASL („St Andrews Static/Standard Language“, 1972)
    - rein funktionale Programmiersprache

```
'Hello World!", nl
```

Multi-Paradigmen-Programmiersprachen

17. F# (2002)
    - funktional, imperativ, objektorientiert

```
printfn "Hello World!"
```

18. Ruby (benannt nach Rubin-Juwel, 1995)
    - dynamische Typisierung, Reflexion, automatische Speicherbereinigung

```
puts "Hello, World!"
```

19. Rust (benannt nach Rostpilz, 2010)
    - Typsystem formal bewiesen

```
fn main() {
    println!("Hello world!");
}
```

20. Scala („Scalable Language“, 2004)
    - funktionale und objektorientiert, interagiert mit Java-Anwendungen

```
println("Hello world!")
```