Advanced Programming

Course Introduction

Troels Henriksen

Course perspective

Course organisation

Programming in Haskell

Why I think this course is worthwhile

> *The purpose of this course is to provide practical experience with sophisticated programming techniques and paradigms from a language-based perspective. The focus is on high-level programming and systematic construction of well-behaved programs.*
>
> \- from course description

- Solving complex problems in a **maintainable** and **reusable** way.
- Allowing principled **reasoning** about program behaviour.
- Performing rigorous **testing**.

Why I think this course is worthwhile

> *The purpose of this course is to provide practical experience with sophisticated programming techniques and paradigms from a language-based perspective. The focus is on high-level programming and systematic construction of well-behaved programs.*
>
> \- from course description

- Solving complex problems in a **maintainable** and **reusable** way.
- Allowing principled **reasoning** about program behaviour.
- Performing rigorous **testing**.
- Pushing **separation of concerns** further than you have seen before.
- **This is how I program!**

Advanced Programming and Professional Wrestling

Christopher Daniels performing a *flying crossbody* on Jonny Storm.

Photo by Hugo Fernandes

`https://commons.wikimedia.org/wiki/File:Christopher_Daniels_1.jpg`

Kayfabe

The fact or convention of presenting staged performances as genuine or authentic.

[1] By my standards.

[2] In week 3.

Kayfabe

The fact or convention of presenting staged performances as genuine or authentic.

The Kayfabe of AP

- Principled design works better the larger your system is and the longer it has to be maintained.
- For practical reasons we look at small systems that are not maintained over time and that would probably work if implemented any reasonable way.
- We pretend the systems we study are a lot bigger than they actually are.

[1] By my standards.

[2] In week 3.

Kayfabe

The fact or convention of presenting staged performances as genuine or authentic.

The Kayfabe of AP

- Principled design works better the larger your system is and the longer it has to be maintained.
- For practical reasons we look at small systems that are not maintained over time and that would probably work if implemented any reasonable way.
- We pretend the systems we study are a lot bigger than they actually are.

I promise you will get to write real[1] programs in this course![2]

[1] By my standards.

[2] In week 3.

Which languages and tools you will use

- All programming is in **Haskell**.
  - Purely functional.
  - Somewhat similar to F#.
  - Lazy.
- We use standard compilers and build tools.
  - Feel free to use fancy editor integration, but it is not required.
  - Tool expertise not part of learning goals; we try to streamline as much as possible.
  - Windows users will (probably) have to use WSL2.
- We assume programming proficiency, roughly corresponding to an undergraduate degree in computer science.

How we hope this course will change your life

Alternatively: *why is this course interesting even your main interest is AI, algorithmics, user interfaces, or bossing people around?*

How we hope this course will change your life

Alternatively: *why is this course interesting even your main interest is AI, algorithmics, user interfaces, or bossing people around?*

1. Reasoning about computation as a mathematical construct.

How we hope this course will change your life

Alternatively: *why is this course interesting even your main interest is AI, algorithmics, user interfaces, or bossing people around?*

1. Reasoning about computation as a mathematical construct.
2. Clean separation and modularisation.

How we hope this course will change your life

Alternatively: *why is this course interesting even your main interest is AI, algorithmics, user interfaces, or bossing people around?*

1. Reasoning about computation as a mathematical construct.
2. Clean separation and modularisation.
3. Taking generalisation and abstraction further.

How we hope this course will change your life

Alternatively: *why is this course interesting even your main interest is AI, algorithmics, user interfaces, or bossing people around?*

1. Reasoning about computation as a mathematical construct.
2. Clean separation and modularisation.
3. Taking generalisation and abstraction further.
4. How to parse input data in a principled and convenient manner.

How we hope this course will change your life

Alternatively: *why is this course interesting even your main interest is AI, algorithmics, user interfaces, or bossing people around?*

1. Reasoning about computation as a mathematical construct.
2. Clean separation and modularisation.
3. Taking generalisation and abstraction further.
4. How to parse input data in a principled and convenient manner.
5. How to structure (potentially large) concurrent systems in a robust manner.

How we hope this course will change your life

Alternatively: *why is this course interesting even your main interest is AI, algorithmics, user interfaces, or bossing people around?*

1. Reasoning about computation as a mathematical construct.
2. Clean separation and modularisation.
3. Taking generalisation and abstraction further.
4. How to parse input data in a principled and convenient manner.
5. How to structure (potentially large) concurrent systems in a robust manner.
6. How to effectively test complex systems without writing a million unit tests.

Course perspective

Course organisation

Programming in Haskell

Teachers

Troels (*course administrator*)

`athas@sigkill.dk`

Mikkel

`mkm@di.ku.dk`

Course structure

Recommended reading

- AP Course Notes.
- *Programming in Haskell* by Graham Hutton.
- Various papers.

Assignments

- **Six** weekly group assignments.
- Preferably **three students per group**.
- **Graded** with 0-4 points.
- **No resubmissions.**
- Exam qualification: at least 12 points and **at least one point** per assignment.
- **First one handed out today.**

Assignments and exercises

The AP Student Algorithm

1. Repeat for every week:

Assignments and exercises

The AP Student Algorithm

```
1. Repeat for every week:
    1.1 Consume learning material:
        ▶ Attend lectures.
        ▶ Read texts.
```

Assignments and exercises

The AP Student Algorithm

1. Repeat for every week:
   1.1 Consume learning material:
      - Attend lectures.
      - Read texts.

   1.2 Solve programming exercises.

Assignments and exercises

The AP Student Algorithm

1. Repeat for every week:
   1.1 Consume learning material:
      - Attend lectures.
      - Read texts.

   1.2 Solve programming exercises.
   1.3 Solve programming assignment.

Assignments and exercises

The AP Student Algorithm

1. Repeat for every week:
    - 1.1 Consume learning material:
        - Attend lectures.
        - Read texts.
    - 1.2 Solve programming exercises.
    - 1.3 Solve programming assignment.
    - 1.4 Hand in assignment with your group.

Assignments and exercises

The AP Student Algorithm

1. Repeat for every week:

   1.1 Consume learning material:

   - Attend lectures.
   - Read texts.

   1.2 Solve programming exercises.

   1.3 Solve programming assignment.

   1.4 Hand in assignment with your group.

2. Attend exam.

Assignments and exercises

The AP Student Algorithm

1. Repeat for every week:
    1.1 Consume learning material:
    - ▶ Attend lectures.
    - ▶ Read texts.

    1.2 Solve programming exercises.
    1.3 Solve programming assignment.
    1.4 Hand in assignment with your group.
2. Attend exam.
3. **Optional**: Attend re-exam.

Assignments and exercises

The AP Student Algorithm

1. Repeat for every week:
    - 1.1 Consume learning material:
        - Attend lectures.
        - Read texts.
    - 1.2 Solve programming exercises.
    - 1.3 Solve programming assignment.
    - 1.4 Hand in assignment with your group.
2. Attend exam.
3. **Optional**: Attend re-exam.
4. **Optional**: Return to 1.

The assignments build *directly* on the code you see in the exercises.

Physical teaching activities

What and where

Lecture:	Tuesday 10-12 at Aud 01.
Exercises:	Thursday 10-12 at various locations around the world.
Lecture:	Thursday 13-15 at Store UP-1.
Exercises:	Thursday 15-17 at various locations around the world.
Study café:	Friday 13-16 at Lille UP-1.

- Work on the exercises by yourself after Tuesday lecture.
- Continue (with TA help) on Thursday.
- Attend café on Friday if you wish.
- See course website for classrooms.

Course resources

- **Course website**: `https://github.com/diku-dk/ap-e2024-pub`
- **Absalon**: used for handins, announcements, and discussions.
  - **Action item:** Add yourself to an *Assignment Group*.
- **Discord**: `https://discord.gg/dJgTJ7mry7`
  - Not mandatory, but many students seem to prefer it over the Absalon forum.
  - Use your Absalon name, not your sweet hacker nick.

Classes (*hold*) and TAs

Hold 1: Robert

Hold 2: Francisco

Hold 3: Therese

Hold 4: Joachim (**online**)

Hold 5: Mikkel

Hold 6: Ian

Hold 7: Rasmus

Hold 8: Thomas

You find your *hold number* on Absalon.

Learning material

No mandated textbook.

Learning material

No mandated textbook.

But…

- We recommended chapters from *Programming in Haskell* by Graham Hutton.
  - If you don’t like its style, feel free to read equivalent chapters from books such as *Learn You a Haskell for Great Good* or *Real World Haskell*, or any other that the Internet recommends.

Learning material

No mandated textbook.

But…

- We recommended chapters from *Programming in Haskell* by Graham Hutton.
  - If you don't like its style, feel free to read equivalent chapters from books such as *Learn You a Haskell for Great Good* or *Real World Haskell*, or any other that the Internet recommends.
- We will provide some classic *research papers*.
  - You can make do without reading these, but they often provide very good background material to help your understanding.

Learning material

No mandated textbook.

But…

- We recommended chapters from *Programming in Haskell* by Graham Hutton.
  - If you don't like its style, feel free to read equivalent chapters from books such as *Learn You a Haskell for Great Good* or *Real World Haskell*, or any other that the Internet recommends.
- We will provide some classic *research papers*.
  - You can make do without reading these, but they often provide very good background material to help your understanding.
- We have written *course notes*.
  - `https://diku-dk.github.io/ap-notes/`
  - One chapter per week; the idea is that they roughly correspond to lecture content.
    - And things that don't fit anywhere else.
  - Newly written this year, still WIP, please report bugs if you find any.
    - Incomprehensible text is a bug.

Questions?

Course perspective

Course organisation

Programming in Haskell

Rest of the lecture takes place in Emacs.