Tutorium Funktionale Programmierung 2019

Part 1 - Introduction

VO - *Organisation* and *Einführung* (till slide 16)

Benedikt Dornauer, 08.10.2019

About the Tutorium

- **content:** topics about the lecture and proseminar
- **practice exercises** (no solutions for PR-sheets!)
- **possibility to ask questions** (preferred initially before Tutorium per email that I can prepare examples)
- **train yourself**

- **Time: Tuesday, 14:15-15:00**
- **Location: HSB 5 (Technikerstraße 13b)**

Feedback

How many have **problems installing Haskell**?

How many need **help using the terminal**?

How many **doesn´t know how to run a *file.hs***?

Haskell installation – links

- **OS Linux (recommended)**
  https://www.haskell.org/platform/linux.html

- OS Windows:

  https://www.haskell.org/platform/

- OS macOS

  https://www.haskell.org/platform/mac.html

Terminal in Linux (Basics)

Directory

Input	Explanation
mkdir dir	Make new directory *dir*
cd ..	Go up a directory
cd *dir*	Change directory to *dir*
ls (-l)	List files (detailed info)

Working with files

Input	Explanation
touch file	Create *file*
chmod 775 *file*	Change mode of *file* to 775
head file	Show first ten lines
rm file	Remove file

drwxrwxrwx

d = Directory
r = Read
w = Write
x = Execute

7	rwx	111
6	rw-	110
5	r-x	101
4	r--	100
3	-wx	011
2	-w-	010
1	--x	001
0	---	000

chmod 777

rwx | rwx | rwx
Owner | Group | Others

Image: https://www.reddit.com/r/linux/comments/ayditr/chmod_cheatsheet/

Terminal in Linux (Basics)

Important shortcuts

Input	Explination
Ctrl + c	suspend a process
Ctrl + l	clear terminal screen

Working with an editor e.g. nano

Input	Explanation
nano file	Open the file in nano
Strg+X	Close current file buffer / exit nano
Strg+O	Write the current file to disk (save)
…	…

Run code with ghci

File *temp.hs*

```
quad x = x * x
```

Run temp.hs

```
benediktdornauer@benediktdornauer-VirtualBox:/media/sf_VO_Tutorium_Funktionale_P
rogrammierung/Code/1$ ghci
GHCi, version 8.0.2: http://www.haskell.org/ghc/  :? for help
Prelude> :load temp.hs
[1 of 1] Compiling Main             ( temp.hs, interpreted )
Ok, modules loaded: Main.
*Main> quad 3
9
*Main>
```

Interpreter commands

Command	Meaning
`:load` ⟨*filename*⟩	load script ⟨*filename*⟩
`:reload`	reload current script
`:edit` ⟨*filename*⟩	edit script ⟨*filename*⟩
`:edit`	edit current script
`:type` ⟨*expression*⟩	show type of ⟨*expression*⟩
`:set` ⟨*property*⟩	change various settings
`:show` ⟨*info*⟩	show various information
`:!` ⟨*command*⟩	execute ⟨*command*⟩ in shell
`:?`	show help text
`:quit`	bye-bye!

Some useful commands

Set an editor

```
Prelude> :set editor nano
```

Edit a file in *ghci* (editor must be set)

```
*Main> :edit testfile.hs
Ok, modules loaded: Main.
```

Check out types

```
*Main> :type True
True :: Bool
```

*Type Bool will be discussed later on.

Exercise 1.1: Write code and run it

The "*Kostenfunktion*" and "Erlösfunktion" are given. The "Gewinnfunktion" can be expressed with G(x) = E(x)-K(x)

$$K(x) = 0,01x^3 - 9x^2 + 3000x + 10000$$

$$E(x) = 4000x$$

- Create new folder and create inside new file *tutorium_1_1.hs*
- Use an editor to add the function G(x) in Haskell syntax.
- Compile and run and test it with values

  8, 500 and 1000.

Discussion / Quiz

- [ ] Is *minimum* a pure function*? If so, why?*
- [ ] Is *Haskell* a pure language*?*
- [ ] *What is the **state** of variable x:= 20 ?*
- [ ] *What is the difference between **strict** and **non-strict**?*
- [ ] What does “*lazy evaluation*” mean?

Exercise 1.2: Several ways to evaluate expressions

- Illustrate all possible evaluation expressions using a tree for

$$(1 + 2 + 3)^2$$

Exercise 1.3: Evaluation Strategies

- *square x = x *x*
- *sum x y z = x + y + z*

Use the evaluation strategies *call-by-value (strict) / call-by-name (non-strict) / call-by-need (lazy) for*

square (sum 1 (2+1) 3)

until normal-form is reached.

Further example

- *d x y = x * y*
- *f x = x + x*

Use the evaluation strategies call-by-value (strict) / call-by-name (non-strict) / call-by-need (lazy) for

d (f 2) (f 2)

until normal-form is reached.

Questions? Need help? Feedback? etc.

- benedikt.dornauer@student.uibk.ac.at