**HOGESCHOOL ROTTERDAM / CMI**

**Functioneel programmeren  
(met Haskell)**

**INFSEN01-1**

Aantal studiepunten: 4

Modulebeheerder: Tony Busker

# Modulebeschrijving INFSEN01-1

|  |  |
| --- | --- |
| **Modulenaam:** | Functioneel Programmeren |
| **Modulecode:** | **INFSEN01-1** |
| **Aantal studiepunten en** | Dit studieonderdeel levert 4 studiepunten op, hetgeen overeenkomt met een studielast van 112 uur. |
| **Studiebelastinguren:** | De verdeling van deze 112 uur over de collegeweken is als volgt:  Begeleide colleges:  - Hoor-/werkcollege 8 weken: 8 \* 4 \* 50 minuten 27 uur  Onbegeleide uren:  - literatuurstudie (8 \* 4 uur) 32 uur- Opdrachten/cases (7 \* 5 uur) 35 uur- Toetsvoorbereiding + deelname 18 uur  - Totaal: 112 uur |
| **Vereiste voorkennis:** | Kennis van een imperatieve of object-georiënteerde programmeertaal is wenselijk. |
| **Werkvorm:** | Hoorcollege, practicum. |
| **Toetsing:** | Inleveropdrachten en een eindtoets (theorie en praktijk; zonder boek). |
| **Leermiddelen:** | * Miran Lipovača, <http://learnyouahaskell.com/chapters> [Lipo] |
| **Draagt bij aan competentie:** |  |
| **Leerdoelen:** | Na het met succes voltooien van deze module is de student in staat om voor een eenvoudig 'probeem' een oplossing te realiseren in een (pure) functionele programmeertaal (Haskell). Aan het eind van deze module kan de student:   * doel(nut) van functionele programmeren uitleggen; * concepten functioneel programmeren begrijpen en hanteren; * een probleem oplossen (ontwerpen) a.d.h.v. het functioneel programmeer paradigma * de oplossing programmeren (realiseren) in de functionele programmeertaal Haskell; |
| **Inhoud:** | In deze module wordt aandacht besteed aan:   * de principes/concept van functioneel programmeren * het gebruik van een hulpmiddel (tool) GHC (Glasgow Haskell Compiler). * de operatoren van Haskell * (abstracte) datatypes, classes, tuples en lijsten * Begrip 'currying' uitleggen en gebruiken * het gebruik van functies als parameter, lambda expressions * functies op lijsten toepassen (map, foldl, foldr, …) * listcomprehension, guards * recursie/pattern matching (lijsten) * Interactiviteit (IO) * hogere-orde functies, lazy evaluation * complexere functies op lijsten * combinatie van tuples en lijsten * monads |
| **Opmerkingen:** |  |
| **Modulebeheerder:** | Tony Busker |
| **Datum:** | januari 2015, v 1.02 (Tony Busker) |

# 1. Algemene omschrijving

## Inleiding

Functioneel programmeren en functionele programmeertalen zoals Scala en Clojure (F#) worden steeds populairder om diverse redenen. Mainstream (programmeer) talen als Java (generics), C# Linq, etc. krijgen steeds meer functionaliteit uit het functioneel programmeren. De architectuur Map/Reduce (Google/Hadoop) is ontleent aan het FP programmeerparadigma. Functioneel programmeren gaat dus een belangrijke plaats innemen in systeemontwikkeling.

“Java™ developers should learn functional paradigms now, even if they have no immediate plans to move to a functional language such as Scala or Clojure. Over time, all mainstream languages will become more functional” [IBM].

“LISP is worth learning for a different reason — the profound enlightenment experience you will have when you finally get it. That experience will make you a better programmer for the rest of your days, even if you never actually use LISP itself a lot.” – Eric S. Raymond

“SQL, Lisp, and Haskell are the only programming languages that I've seen where one spends more time thinking than typing.” – Philip Greenspun

“I do not know if learning Haskell will get you a job. I know it will make you a better software developer.” – Larry O’ Brien

Functionele talen gebruiken een ander paradigma dan imperatieve en objectgeoriënteerde talen. Ze gebruiken side-effect vrije functies als een elementaire bouwstenen in de taal. Dit maakt heel veel dingen mogelijk en maakt veel dingen moeilijker (of in de meeste gevallen anders dan wat mensen zijn gewend zijn).

Een van de grootste voordelen van functioneel programmeren is dat de volgorde van uitvoering van side effect vrije functies niet belangrijk is. Bijvoorbeeld in Erlang dit wordt gebruikt om concurrency een zeer transparante manier mogelijk te maken. En omdat functies in functionele talen zich zeer vergelijkbaar met wiskundige functies gedragen is het gemakkelijk om die in functionele talen te vertalen. In sommige gevallen kan dit de code beter leesbaar maken.

Het is erg moeilijk om bruikbare software te schrijven zonder IO, echter IO is moeilijk te implementeren zonder side-effect in functies. Dus de meeste ontwikkelaars zijn tot niet meer in staat dan het functioneel programmeren van het berekenen van een output op basis van een enkele input . In de moderne talen met een gemengd paradigma zoals C # of Scala is dit eenvoudiger.

Veel van de moderne talen hebben elementen van functionele programmeertalen. C # 3.0 heeft veel functioneel programmeren functies en ook in python is functioneel programmeren mogelijk. Het realiseren van concurrency (parallelisme) is in ‘normaal’ programmeren lastig en we krijgen meer en meer de beschikking over multiprocessor computers. Het functionele programmeer paradigma is impliciet parallelliseerbaar.

## Relatie met andere onderwijseenheden

Deze module legt een basis voor het functioneel programmeer paradigma. Deze is terug te vinden in vele programmeertalen. Maar ook in een architectuur (data operating system) als HADOOP en SPARK maken gebruik van dit paradigma.

## Leermiddelen

Verplicht:

* Boek/Web
  + Titel : Learn You A Haskell for a Great Good [Miran]
  + Url : <http://learnyouahaskell.com/chapters>
  + Aueur : Miran Lipovača
  + Jaar : 2011

Aanbevolen:

* Boek
  + Auteur(s) : Graham Hutton
  + Titel : Programming in Haskell
  + Uitgever : Cambridge University Press
  + ISBN : 978-0521692694
* Boek
  + Auteur(s) : Bryan O'Sullivan, Don Stewart, and John Goerzen
  + Titel : Real World Haskell
  + Uitgever : O'Reilly
  + ISBN : 978-0596514983
* Boek
  + Auteur(s) : Simon Thompson
  + Titel : The Craft of Functional Programming, 3rd Ed.
  + Uitgever : Addison-Wesley Professional
  + ISBN : 978-0201882957

# Programma

| Week | Literatuur | Lesinhoud | Producten |
| --- | --- | --- | --- |
| 1 | *[Miran] 1, 2* | * Algemene inleiding * installeren en werken met Haskell en GHCi * Oefenopgaven week 1 [exercises week 1] |  |
| 2 | *[Miran] 3, 4* | * Types and classes * Oefenopgaven week 2 [exercises week 2] |  |
| 3 | [Miran] 5 | * Defining functions, List comprehension * Oefenopgaven week 3 [exercises week 3] * Toetsopdracht 1 [assignment 1] |  |
| 4 | [Miran] 6 | * Recursive functions, higher order functions * Oefenopgaven week 4 [exercises week 4] |  |
| 5 | [Miran] | * Interactive programs * Oefenopgaven week 5 [exercises week 5] * Eindopdracht [assigment 2] | Inleveren: assignment 1 |
| 6 | *[Miran]* Hoofdstuk 7 | * Declaring types and classes * Oefenopgaven week 6 [exercises week 6] |  |
| 7 | *[Miran]* | * Lazy evaluation * Oefenopgaven week 7 [exercises week 7] |  |
| 8 | *[Miran* | * Werken aan eindopdracht | Inleveren: assignment 2 |
| 9 |  | TOETS |  |
| 10 |  |  |  |

1. **Toetsing en beoordeling**

## 3.1 procedure

Het eindcijfer voor deze module wordt bepaald door:

* Inleveropdrachten: (maximaal 1.5 punten van het eindcijfer voor elke opdracht). Deze **inleveropdrachten** **tellen alleen mee als het cijfer van het tentamen voldoende (>= 5.5) is**. De eerste inleveropdracht moet op papier en digitaal worden ingeleverd aan het begin van de practicumles van onderwijsweek 5. De tweede inleveropdracht moet op papier en digitaal worden ingeleverd aan het eind van de practicumlessen. Beide opdrachten moeten duidelijk voorzien zijn van naam, studentnummer en klas (groep). Te laat ingeleverde inleveropdrachten worden niet beoordeeld. Inleveropdrachten kunnen niet worden herkanst. Het kan in uitzonderlijke gevallen voorkomen dat opdrachten door overmacht (ziekte) niet op tijd kunnen worden ingeleverd; in dat geval dient dit onmiddellijk (dus niet achteraf) bij de docent gemeld te worden. De docent beslist of er inderdaad sprake van overmacht is, tenzij er een doktersverklaring of vergelijkbaar wordt overlegd.
* Inleveropdrachten mogen individueel of in groepjes van maximaal 2 studenten worden gemaakt. Beide studenten van een duo moeten vragen over elk aspect van een inleveropdracht kunnen beantwoorden. De docent interviewt studenten a.d.h.v. de assignments om dat te toetsen.
* De eindtoets: Het resultaat van het tentamen wordt vermenigvuldigd met (= maximaal 7 punten) en vormt de rest van het eindcijfer. De eindtoets bestaat uit een theoriegedeelte (geen hulpmiddelen) en een praktijkgedeelte. De toetsduur is 90 minuten.

**3.2 Herkansing**

Deze module kan in de volgende onderwijsperiode worden herkanst door de eindtoets opnieuw te maken.

Inleveropdrachten kunnen niet worden herkanst.

**Bijlage 1: Toetsmatrijs**

De toetsmatrijs geeft de relatie weer tussen de leerdoelen en de vragen dan wel opdrachten van een toets. Tevens geeft de toetsmatrijs aan op welk niveau het leerdoel getoetst wordt.

|  |  |  |
| --- | --- | --- |
|  | **Leerdoelen** | **Verwijzing naar opdracht/ vraag/ criteria** |
|  | *doel(nut) van functionele programmeren kunnen uitleggen;* | tentamen |
|  | *concepten functioneel programmeren begrijpen en kunnen hanteren;* | tentamen |
|  | *een probleem oplossen (analyseren, ontwerpen) a.d.h.v. het functioneel programmeer paradigma* | tentamen/opdrachten |
|  | *de oplossing programmeren (realiseren) in de functionele programmeertaal Haskell* | tentamen/opdrachten |