

---

# Inhaltsverzeichnis

<b>1</b>	<b>Motivation und Einführung</b>	<b>1</b>
1.1	Funktionale Programmierung	2
1.1.1	Motivation und Anwendung	2
1.1.2	Warum gerade Haskell?	3
1.2	Grundbegriffe und Prinzipien der Programmentwicklung	3
1.3	Installation und Verwendung von Haskell	4
1.3.1	Installation der aktuellen Version	5
1.3.2	Die ersten Schritte in Hugs	6
1.3.3	Arbeiten auf der Konsole	6
1.3.4	Die Entwicklungsumgebung winhugs	7
1.3.5	Erstellung und Verwendung von Skripten	8
1.4	Haskell ist mehr als ein Taschenrechner	9
1.5	Vordefinierte Funktionen der Prelude	10

## Teil I Haskell-Grundlagen

<b>2</b>	<b>Einfache Datentypen</b>	<b>15</b>
2.1	Wahrheitswerte	16
2.1.1	Negation	17
2.1.2	Konjunktion	17
2.1.3	Disjunktion	18
2.1.4	Exklusiv-Oder	18

2.1.5	Boolesche Algebra .....	19
2.1.6	Boolesche Gesetze .....	19
2.2	Zahlentypen .....	21
2.2.1	Datentyp Int .....	21
2.2.2	Datentyp Integer .....	22
2.2.3	Datentypen Float und Double .....	23
2.3	Zeichen und Symbole .....	23
2.4	Übungsaufgaben .....	24
<b>3</b>	<b>Funktionen und Operatoren .....</b>	<b>25</b>
3.1	Funktionen definieren .....	26
3.1.1	Parameterübergabe .....	27
3.1.2	Reservierte Schlüsselwörter .....	28
3.1.3	Wildcards .....	28
3.1.4	Signaturen und Typsicherheit .....	29
3.1.5	Pattern matching .....	30
3.1.6	Pattern matching mit case .....	31
3.1.7	Lokale Definitionen mit where .....	31
3.1.8	Lokale Definitionen mit let-in .....	32
3.1.9	Fallunterscheidungen mit Guards .....	33
3.1.10	Fallunterscheidungen mit if-then-else .....	34
3.1.11	Kommentare angeben .....	34
3.2	Operatoren definieren .....	35
3.2.1	Assoziativität und Bindungsstärke .....	35
3.2.2	Präfixschreibweise – Operatoren zu Funktionen .....	36
3.2.3	Infixschreibweise – Funktionen zu Operatoren .....	37
3.2.4	Eigene Operatoren definieren .....	37
3.3	Lambda-Notation .....	38
3.4	Übungsaufgaben .....	39

<b>4</b>	<b>Rekursion als Entwurfstechnik</b>	41
4.1	Rekursive Fakultätsfunktion	42
4.2	Lineare Rekursion	43
4.3	Kaskadenförmige Rekursion	44
4.4	Verschachtelte Rekursion	45
4.5	Wechselseitige Rekursion	46
4.6	Endständige Rekursion	46
4.7	Übungsaufgaben	46
<b>5</b>	<b>Einfache Datenstrukturen</b>	49
5.1	Listen	50
5.1.1	Zerlegung in Kopf und Rest	51
5.1.2	Rekursive Listenfunktionen	53
5.1.3	Zusammenfassen von Listen	55
5.1.4	Automatische Erzeugung von Listen	56
5.1.5	Automatisches Aufzählen von Elementen	57
5.1.6	Lazy evaluation	59
5.1.6.1	Unendliche Listen	59
5.1.6.2	Das Sieb des Eratosthenes	60
5.1.7	Listen zerlegen	61
5.2	Tupel	62
5.2.1	Beispiel pythagoräisches Tripel	63
5.2.2	Beispiel $n$ -Dameproblem	64
5.3	Zeichenketten	65
5.4	Übungsaufgaben	67
<b>6</b>	<b>Funktionen höherer Ordnung</b>	69
6.1	Mapping	71
6.2	Filtern	72
6.3	Faltung	73
6.3.1	Faltung von rechts mit Startwert	74
6.3.2	Faltung von rechts ohne Startwert	77

6.3.3	Faltung von links mit Startwert .....	78
6.3.4	Faltung von links ohne Startwert .....	79
6.3.5	Unterschied zwischen Links- und Rechtsfaltung .....	79
6.4	Entfaltung .....	80
6.4.1	Definition von unfold .....	80
6.4.2	Mapping durch unfold .....	81
6.5	Zip .....	82
6.6	Unzip .....	82
6.7	Funktionskompositionen .....	83
6.8	Currying .....	85
6.9	Übungsaufgaben .....	87
<b>7</b>	<b>Eigene Typen und Typklassen definieren .....</b>	<b>89</b>
7.1	Typsynonyme mit type .....	90
7.2	Einfache algebraische Typen mit data und newtype .....	91
7.2.1	Datentyp Tupel .....	94
7.2.2	Datentyp Either .....	94
7.2.3	Datentyp Maybe .....	95
7.2.4	Datentypen mit mehreren Feldern .....	96
7.3	Rekursive Algebraische Typen .....	98
7.4	Automatische Instanzen von Typklassen .....	99
7.4.1	Typklasse Show .....	100
7.4.2	Typklasse Read .....	100
7.4.3	Typklasse Eq .....	101
7.4.4	Typklasse Ord .....	101
7.4.5	Typklasse Enum .....	101
7.4.6	Typklasse Bounded .....	102
7.5	Eingeschränkte Polymorphie .....	102
7.6	Manuelles Instanziiieren .....	102
7.7	Projekt: Symbolische Differentiation .....	105
7.7.1	Operatorbaum .....	106

7.7.2	Polynome berechnen .....	107
7.7.3	Ableitungsregeln .....	107
7.7.4	Automatisches Auswerten .....	108
7.8	Eigene Klassen definieren .....	109
7.9	Übungsaufgaben .....	110
<b>8</b>	<b>Modularisierung und Schnittstellen .....</b>	<b>111</b>
8.1	Module definieren .....	112
8.2	Sichtbarkeit von Funktionen .....	112
8.3	Qualified imports .....	114
8.4	Projekt: Adressbuch .....	114
8.4.1	Modul Woerterbuch .....	114
8.4.2	Modul Adressbuch .....	115
8.4.3	Modul TestAdressbuch .....	116
8.5	Übungsaufgaben .....	117
 <b>Teil II Fortgeschrittene Haskell-Konzepte</b>		
<b>9</b>	<b>Laufzeitanalyse von Algorithmen .....</b>	<b>121</b>
9.1	Motivation .....	122
9.2	Landau-Symbole .....	123
9.2.1	Obere Schranken $O$ .....	123
9.2.2	Starke obere Schranken $o$ .....	124
9.2.3	Untere Schranken $\Omega$ .....	124
9.2.4	Starke untere Schranken $\omega$ .....	125
9.2.5	Asymptotisch gleiches Wachstum $\Theta$ .....	125
9.2.6	Definition über Grenzwerte von Quotientenfolgen .....	125
9.3	Umgang mit Schranken und Regeln .....	125
9.4	Übersicht wichtiger Laufzeiten .....	127
9.5	Best, Worst und Average Case .....	127
9.6	Analysieren der Laufzeit .....	127
9.6.1	Fakultätsfunktion .....	128

9.6.2	Elemente in Listen finden .....	129
9.6.3	Listen umdrehen .....	130
9.6.4	Potenzen .....	131
9.6.5	Minimum einer Liste .....	133
9.7	Übungsaufgaben .....	135
<b>10</b>	<b>Arrays, Listen und Stacks .....</b>	<b>137</b>
10.1	Arrays .....	138
10.1.1	Statische Arrays .....	138
10.1.2	Dynamische Arrays .....	140
10.2	Liste und Stack .....	141
10.3	Listen sortieren .....	142
10.3.1	SelectionSort .....	142
10.3.2	InsertionSort .....	143
10.3.3	BubbleSort .....	144
10.3.4	QuickSort .....	146
10.3.5	MergeSort .....	148
10.3.6	BucketSort .....	149
10.3.7	RadixSort .....	150
10.4	Algorithmen auf Stacks .....	152
10.4.1	Umgekehrte Polnische Notation .....	152
10.4.2	Projekt: Klammertest .....	153
10.5	Übungsaufgaben .....	155
<b>11</b>	<b>Warteschlangen .....</b>	<b>157</b>
11.1	Implementierung über Listen .....	158
11.2	Amortisierte Laufzeitanalyse .....	159
11.2.1	Bankiermethode .....	159
11.2.2	Analyse der Warteschlange .....	160
11.3	Erweiterung um Lazy Evaluation .....	160
11.4	Angepasste amortisierte Analyse .....	161
11.5	Beispielanwendung .....	163
11.6	Übungsaufgaben .....	163

---

<b>12 Bäume</b>	165
12.1 Implementierung der Datenstruktur Baum	166
12.2 Balancierte Bäume	167
12.3 Traversierungen	168
12.3.1 Pre-, In- und Postorder	168
12.3.2 Levelorder	169
12.4 Übungsaufgaben	170
<b>13 Wörterbücher</b>	171
13.1 Analyse und Vorüberlegungen	172
13.2 Implementierung	173
13.3 Laufzeitanalyse	174
13.4 Übungsaufgaben	175
<b>14 Prioritätswarteschlangen</b>	177
14.1 Operationen und mögliche Umsetzungen	178
14.2 Realisierung mit einer Liste	178
14.3 Realisierung mit einem Binärbaum	178
14.4 Zwei Bäume verschmelzen	180
14.5 Amortisierte Laufzeitanalyse von merge	181
14.6 Beispielanwendung	182
14.7 Übungsaufgaben	183
<b>15 Random-Access Listen</b>	185
15.1 Realisierung mit einem Suchbaum	186
15.1.1 Preorder versus Inorder bei Binärbäumen	186
15.1.2 Liste vollständiger Binärbäume	187
15.1.3 Verschmelzen mit Greedy-Strategie	187
15.2 Implementierung der grundlegenden Listenfunktionen	189
15.3 Implementierung von elementAn	190
15.4 Beispielanwendung	191
15.5 Übungsaufgaben	192

<b>16 Graphen</b>	193
16.1 Definition und wichtige Begriffe	194
16.2 Abstrakter Datentyp Graph	195
16.2.1 Adjazenzliste und Adjazenzmatrix	195
16.2.2 Implementierung der Adjazenzliste	196
16.3 Algorithmen auf Graphen	197
16.3.1 Traversieren von Graphen	198
16.3.1.1 Tiefensuche im Graphen	199
16.3.1.2 Breitensuche im Graphen	200
16.3.1.3 Implementierung von Tiefen- und Breitensuche	200
16.3.2 Topologisches Sortieren	204
16.4 Übungsaufgaben	207
<b>17 Monaden</b>	209
17.1 Einführung und Beispiele	210
17.1.1 Debug-Ausgaben	210
17.1.1.1 Rückgabewert und Funktionskomposition	210
17.1.1.2 Eigene Eingabetypen definieren	211
17.1.1.3 Identitätsfunktion	211
17.1.2 Zufallszahlen	212
17.2 Monaden sind eine Typklasse	213
17.3 do-Notation	214
17.3.1 Allgemeine Umwandlungsregeln	214
17.3.2 Umwandlungsregeln für if-then-else	215
17.3.3 Beispiel	216
17.4 Vordefinierte Monaden	217
17.4.1 Monade Writer	217
17.4.2 Monade Reader	218
17.4.3 Monade State	220
17.4.4 Monade List	222



17.5	Ein- und Ausgaben .....	224
17.5.1	Stream-basierte Eingaben .....	224
17.5.2	Monade IO .....	225
17.5.2.1	Bildschirm Ausgaben .....	226
17.5.2.2	Tastatureingaben .....	227
17.5.2.3	Eingabepufferung .....	227
17.5.2.4	Beispiel: Hangman .....	228
17.5.3	Dateien ein- und auslesen .....	230
17.6	Übungsaufgaben .....	231
<b>18</b>	<b>Programme verifizieren und testen .....</b>	<b>233</b>
18.1	Beweis durch vollständige Induktion .....	234
18.1.1	Die fünf Peano-Axiome .....	234
18.1.2	Beweiskonzept .....	235
18.1.2.1	Gaußsche Summenformel .....	235
18.1.2.2	Vier- und Fünf-Euro-Münze .....	236
18.1.2.3	Fakultätsfunktion .....	236
18.1.3	Vollständige Induktion über Strukturen .....	237
18.1.3.1	Induktion über Listen .....	238
18.1.3.2	Induktion über Bäume .....	239
18.2	QuickCheck .....	240
18.2.1	Beispiel: Sortieren .....	241
18.2.2	QuickCheck für eigene Typen verwenden .....	242
18.2.3	Testdatengeneratoren .....	242
18.3	Übungsaufgaben .....	243
<b>19</b>	<b>Berechenbarkeit und Lambda-Kalkül .....</b>	<b>245</b>
19.1	Der Lambda-Kalkül .....	246
19.2	Formale Sprachdefinition .....	246
19.2.1	Bezeichner .....	246
19.2.2	$\lambda$ -Funktion .....	247
19.2.3	Applikation .....	247

19.2.4	Reguläre $\lambda$ -Ausdrücke	247
19.3	Freie und gebundene Variablen	248
19.4	$\lambda$ -Ausdrücke auswerten	248
19.4.1	$\alpha$ -Konversion	248
19.4.2	$\beta$ -Reduktion	249
19.5	Boolesche Algebra	250
19.5.1	True und False	250
19.5.2	Negation	251
19.5.3	Konjunktion und Disjunktion	251
19.6	Tupel	251
19.6.1	2-Tupel	252
19.6.2	First und Second	252
19.7	Listen	252
19.7.1	<i>Head</i> – Kopf einer Liste	253
19.7.2	<i>Tail</i> – Rest einer Liste	253
19.7.3	<i>Empty</i> – Test auf eine leere Liste und Nil	253
19.8	Arithmetik	254
19.8.1	Natürliche Zahlen	254
19.8.2	<i>Zero</i> – Der Test auf Null	254
19.8.3	<i>S</i> – Die Nachfolgerfunktion	255
19.8.4	Die Addition	256
19.8.5	Die Multiplikation	257
19.8.6	Die Vorgängerfunktion	257
19.9	Rekursion	258
19.9.1	Alternative zu Funktionsnamen	258
19.9.2	Fixpunktkombinatoren	259
19.10	Projekt: $\lambda$ -Interpreter	260
19.10.1	$\lambda$ -Ausdrücke repräsentieren	261
19.10.2	Auswerten von $\lambda$ -Ausdrücken	261
19.10.3	Freie und gebundene Variablen	263
19.10.4	Wörterbuch für Substitutionen	263
19.10.5	$\lambda$ -Parser	265
19.11	Übungsaufgaben	267

<b>Lösungen der Aufgaben</b> .....	269
<b>Literaturverzeichnis</b> .....	277
<b>Sachverzeichnis</b> .....	281

Haskell-Intensivkurs

Ein kompakter Einstieg in die funktionale  
Programmierung

Block, M.; Neumann, A.

2011, XX, 300 S. 64 Abb., 5 Abb. in Farbe. Mit  
Online-Extras., Softcover

ISBN: 978-3-642-04717-6