
ÍNDICE GENERAL

Índice de figuras	XIII
Prólogo	XVII
Convenios	XXIII
I Programación funcional básica con HASKELL 98	1
1. Programación funcional	3
1.1. Funciones	3
1.2. Sesiones y declaraciones	4
1.3. Reducción de expresiones	6
1.3.1. Órdenes de reducción aplicativo y normal	9
1.3.2. Evaluación perezosa	9
1.4. Sobre HASKELL	11
2. Introducción a HASKELL	13
2.1. El lenguaje HASKELL	13
2.2. Tipos simples predefinidos	15
2.2.1. El tipo <i>Bool</i>	15
2.2.2. El tipo <i>Int</i>	16
2.2.3. El tipo <i>Integer</i>	16
2.2.4. El tipo <i>Float</i>	17
2.2.5. El tipo <i>Double</i>	18
2.2.6. El tipo <i>Char</i>	18
2.2.7. Operadores de igualdad y orden	19
2.3. Constructores de tipos predefinidos	20
2.3.1. Tuplas	20
2.3.2. Listas	21
2.3.3. El constructor de tipo (\rightarrow)	22
2.4. Comentarios	24
2.5. Operadores	24
2.5.1. Operadores frente a funciones	27
2.6. Comparación de patrones	28
2.6.1. Patrones constantes	29

2.6.2.	Patrones para listas	32
2.6.3.	Patrones para tuplas	33
2.6.4.	Patrones aritméticos	34
2.6.5.	Patrones nombrados o seudónimos	35
2.6.6.	El patrón subrayado	36
2.6.7.	Errores comunes	36
2.6.8.	Patrones y evaluación perezosa	37
2.7.	Expresiones <i>case</i>	38
2.8.	La función <i>error</i>	38
2.9.	Funciones a trozos	39
2.10.	Expresiones condicionales	40
2.11.	Definiciones locales	41
2.12.	Expresiones lambda	42
2.13.	Sangrado	43
2.14.	Ámbitos y módulos	45
2.15.	Ejercicios	46
3.	Funciones de orden superior y polimorfismo	49
3.1.	Parcialización	49
3.1.1.	Aplicación parcial	51
3.1.2.	Secciones	54
3.1.3.	Funciones de orden superior	56
3.1.4.	Una función de orden superior sobre naturales	58
3.2.	Polimorfismo	60
3.2.1.	La composición de funciones	62
3.2.2.	Otras funciones polimórficas	65
3.2.3.	Polimorfismo en listas	67
3.2.4.	Polimorfismo en tuplas	69
3.2.5.	Un iterador polimórfico sobre los naturales	70
4.	Definición de tipos	71
4.1.	Sinónimos de tipo	71
4.2.	Definición de tipos de datos	72
4.2.1.	Tipos enumerados	72
4.2.2.	Uniones	73
4.2.3.	Productos	74
4.2.4.	Tipos recursivos	78
4.2.5.	Tipos parametrizados (o polimórficos)	87
4.2.6.	Definiciones <i>newtype</i>	89
4.3.	Propiedades de funciones	90
4.3.1.	La propiedad universal de <i>foldNat</i>	97
4.4.	Sobrecarga y polimorfismo restringido	99
4.4.1.	Un ejemplo de sobrecarga	101
4.5.	Ejercicios	103

5. El sistema de clases de HASKELL	105
5.1. Tipos y clases de tipos. Jerarquía de clases	105
5.1.1. El sistema de clases	105
5.1.2. Declaración de clase	109
5.1.3. La clase <i>Eqde</i> PRELUDE	111
5.2. Contextos	112
5.2.1. Instancias paramétricas	115
5.3. Subclases. La clase <i>Ord</i> de PRELUDE	115
5.3.1. Un ejemplo: los enteros módulo <i>n</i>	118
5.3.2. Intersección de clases	118
5.4. Visualizando y leyendo datos. <i>Read</i> y <i>Show</i>	119
5.5. Las clases <i>Num</i> , <i>Integral</i> y <i>Fractional</i> de PRELUDE	122
5.5.1. Los tipos numéricos de HASKELL	123
5.5.2. Ambigüedad en las constantes numéricas	125
5.5.3. Promoción numérica	126
5.5.4. Ejemplo: los racionales como instancias genéricas	128
5.6. Ejercicios	129
6. Programación con listas	131
6.1. El tipo lista	131
6.1.1. Secuencias aritméticas. La clase <i>Enum</i>	132
6.2. Concatenación de listas	134
6.3. Inducción sobre listas	136
6.4. Selectores	137
6.5. Emparejando listas	140
6.6. Aplicando una función a los elementos de una lista	141
6.7. Filtros	142
6.8. Listas por comprensión	144
6.8.1. Semántica de listas por comprensión	147
6.9. Plegado de listas	148
6.9.1. <i>foldr</i>	148
6.9.2. La propiedad universal de <i>foldr</i>	151
6.9.3. <i>foldl</i>	153
6.10. Ordenación de listas	154
6.10.1. Ordenación por inserción	155
6.10.2. Ordenación por mezcla	156
6.10.3. Ordenación rápida	158
6.11. Problemas combinatorios	159
6.11.1. Los segmentos iniciales de una lista	159
6.11.2. Los segmentos consecutivos de una lista	160
6.11.3. Permutaciones de una lista	161
6.12. Otras funciones predefinidas	162
6.13. Ejercicios	164

7. Entrada y salida	169
7.1. Operaciones de entrada y salida	169
7.1.1. El problema de la entrada y salida	169
7.1.2. El tipo <i>IO</i>	170
7.1.3. Excepciones	172
7.2. Un formateador de textos	174
7.2.1. Una implementación ineficiente	175
7.2.2. Una implementación eficiente	177
7.2.3. Utilidades para el manejo de documentos	179
7.2.4. Una clase de tipos documentables	179
7.2.5. Ejemplos	180
II Programación avanzada	183
8. Evaluación perezosa. Redes de procesos	185
8.1. Evaluación perezosa	185
8.1.1. Argumentos estrictos	185
8.1.2. Procesando estructuras infinitas en forma perezosa	186
8.2. Listas parciales y listas infinitas	189
8.2.1. Aproximaciones o listas parciales	189
8.2.2. Inducción sobre listas parciales	190
8.3. Redes finitas de procesos	191
8.3.1. La criba de Eratóstenes	193
8.3.2. El triángulo de Pascal	195
8.3.3. Procesos con varias entradas	196
8.3.4. La lista de factoriales	196
8.3.5. Los números de Fibonacci	198
8.3.6. Sucesiones genéricas	199
8.3.7. Los números de Hamming	202
8.4. Sucesiones contadoras	203
8.5. Ejercicios	207
9. Programación con árboles y grafos	211
9.1. Árboles	211
9.1.1. Funciones de orden superior sobre árboles	213
9.2. Árboles binarios	215
9.2.1. Árboles binarios de búsqueda	215
9.2.2. Funciones de orden superior para árboles binarios	219
9.2.3. Inducción para árboles binarios	220
9.3. Arrays	222
9.3.1. Una implementación ineficiente	222
9.3.2. Una implementación eficiente	224
9.4. Grafos y búsqueda en grafos	226
9.4.1. Búsqueda en anchura y en profundidad	227
9.4.2. Los grafos como instancias de una clase uniparamétrica	229

9.5.	Grafos con pesos	232
9.5.1.	Grafos con pesos como instancias de clases biparamétricas	232
9.5.2.	Una clase HASKELL para grafos con pesos	234
9.6.	Ejercicios	236
10.	Programación modular y tipos abstractos de datos	243
10.1.	Módulos	243
10.2.	Bibliotecas estandarizadas	243
10.3.	Declaraciones de módulos	243
10.4.	Importación	245
10.4.1.	Importación cualificada	246
10.5.	Tipos abstractos de datos	247
10.6.	Representación	247
10.6.1.	Representación con una interfaz no sobrecargada	247
10.6.2.	Representación con una interfaz sobrecargada	249
10.7.	El TAD Conjunto (<i>Conjunto</i>)	253
10.8.	El TAD Lista Ordenada (<i>OrdLista</i>)	254
10.9.	El TAD Diccionario (<i>Diccionario</i>)	255
10.10.	Un índice KWIC (KeyWord In Context)	257
10.10.1.	Datos del problema	258
10.10.2.	La función <i>kwic</i>	259
10.10.3.	Algunas funciones y sinónimos de tipos útiles	259
10.10.4.	La función <i>creaNoClaves</i>	261
10.10.5.	Modelo de datos para la resolución	261
10.10.6.	La función <i>kwic'</i>	262
10.11.	Ejercicios	264
11.	Programación con mónadas	265
11.1.	Concepto de mónada	265
11.2.	Clases de constructores de tipos	266
11.2.1.	La clase <i>Functor</i>	266
11.2.2.	La clase <i>Monad</i>	269
11.2.3.	La clase <i>MonadPlus</i>	272
11.3.	Interpretación de las propiedades	273
11.3.1.	Interpretación del operador ($>=>$)	276
11.4.	Relación entre funtor y mónada	277
11.5.	La notación <i>do</i>	280
11.6.	Ejemplos de mónadas	283
11.6.1.	La mónada identidad	283
11.6.2.	La mónada escritora	284
11.6.3.	La mónada lectora	286
11.6.4.	La mónada de transformadores de estado	288
11.6.5.	La lista como mónada indeterminista	291
11.6.6.	La mónada error	292
11.7.	Operaciones con mónadas	296
11.7.1.	Combinando mónadas	297

11.7.2. Transformadores monádicos	298
11.8. Ejercicios	302
III Aplicaciones	303
12. Algoritmos numéricos programados funcionalmente	305
12.1. Introducción	305
12.2. Búsqueda de puntos fijos	305
12.3. Extrapolación por paso al límite	308
12.4. Álgebra lineal numérica	312
12.5. Series de potencias	314
12.6. Ejercicios	316
13. Puzzles y solitarios	319
13.1. Algunos problemas combinatorios	319
13.1.1. El producto máximo con un conjunto de dígitos	319
13.1.2. El problema de las vasijas	320
13.1.3. El problema de los caníbales y los misioneros	322
13.1.4. El solitario de Abreu	324
13.2. La sopa de letras	327
13.2.1. Un esbozo de la solución	327
13.2.2. Buscando las apariciones de una palabra	329
13.2.3. Movimiento de matrices y lecturas de líneas	330
13.3. El problema de las ocho reinas	333
13.3.1. Soluciones mediante listas por comprensión	333
13.3.2. Solución mediante búsqueda en grafos	336
13.3.3. Grafos acíclicos	338
13.4. Programación funcional estilo PROLOG	339
13.5. Ejercicios	342
14. Analizadores	345
14.1. Analizadores y la clase <i>Read</i>	345
14.1.1. Gramáticas y la notación BNF	346
14.1.2. El tipo <i>Read</i>	347
14.1.3. La clase <i>Read</i>	356
14.2. Analizadores monádicos	357
14.2.1. Secuenciación	358
14.2.2. Alternancia	359
14.2.3. Filtros	360
14.2.4. Iteración	361
14.2.5. Elección parcial	362
14.2.6. Un analizador para términos	363
14.3. Ejercicios	366

15. Simulación	369
15.1. Generación de aleatorios por congruencias	369
15.1.1. Programando secuencias pseudo-aleatorias	370
15.1.2. Algunos resultados teóricos	373
15.2. Simulación en el juego del póquer	374
15.2.1. Generando un mazo de cartas	374
15.2.2. Búsqueda de ciertas jugadas: parejas, tríos, etc.	375
15.2.3. Contando todas las jugadas	376
15.3. Obteniendo semillas y aleatorios a través del sistema	379
15.3.1. Un modelo monádico para la simulación	380
15.4. El juego de la lotería primitiva	383
15.4.1. Realizando escrutinios	383
15.4.2. Generación de sorteos	384
15.4.3. Estudio estadístico de ciertas combinaciones	385
15.4.4. Descripción monádica del juego de la primitiva	387
15.5. Simulación monádica de juegos con dados	388
15.5.1. Mezclando valores producidos por varias acciones	389
15.5.2. Plegando valores monádicos	389
15.5.3. Repetición de varias tiradas con varios dados	391
15.5.4. Contabilizando jugadas	392

IV Aspectos teóricos **395**

16. Técnicas de programación y transformaciones de programas	397
16.1. Inducción estructural	397
16.2. Parámetros acumuladores	405
16.2.1. Los números de Fibonacci	405
16.2.2. Cálculo del factorial de un natural	410
16.2.3. Plegados estrictos	414
16.3. Transformación de programas. El modelo desplegar/plegar	417
16.3.1. Un ejemplo sencillo de transformación	417
16.3.2. Las reglas desplegar/plegar	418
16.3.3. Reducción de la complejidad por transformación	420
16.3.4. Corrección parcial de los programas transformados	425
16.4. Programas a partir de especificaciones	425
16.4.1. Especificaciones ejecutables	426
16.4.2. Especificaciones no ejecutables	426
16.4.3. Transformación de una especificación no ejecutable	427
16.5. Semántica denotacional de un lenguaje imperativo	430
16.5.1. Representación de entornos con tuplas	432
16.5.2. Representación de entornos con funciones	434
16.5.3. El lenguaje imperativo de Dijkstra	437
16.5.4. Una semántica determinista para el lenguaje de Dijkstra	439
16.5.5. Una semántica indeterminista para el lenguaje de Dijkstra	441
16.6. Ejercicios	443

17. Introducción al λ-cálculo	451
17.1. Sintaxis del lambda cálculo	451
17.2. δ -reducción y β -reducción	453
17.2.1. λ -teorías	457
17.2.2. Eta-conversión y extensionalidad	459
17.2.3. Reducción generada por un programa	460
17.3. Formas normales. Teoremas de Church-Rosser	461
17.4. Órdenes de reducción. Teorema de estandarización	465
17.5. Lambda definibilidad	468
17.5.1. Operaciones lógicas	469
17.5.2. Computabilidad	469
17.5.3. Puntos fijos y recursión	471
17.5.4. Listas en el λC	473
17.6. Los sistemas de tipos de Church y de Curry	474
17.6.1. Propiedades del sistema λ_{\rightarrow} Curry	475
17.6.2. La correspondencia de Howard-Curry-de Bruijn	478
17.7. Ejemplos prácticos de inferencia de tipos	478
17.7.1. Caso de un único argumento	479
17.7.2. Caso de varios argumentos	480
17.7.3. Caso en que aparecen otras variables predefinidas	482
17.8. Inferencia de tipos en presencia de recursión	484
17.9. Inferencia de tipos en presencia de patrones	487
17.10. Reglas elementales para inferencia de tipos	490
17.11. El λ -cálculo polimórfico	492
17.11.1. Inferencia de tipos en presencia de polimorfismo	494
17.11.2. Un teorema de parametricidad para funciones polimórficas	496
Bibliografía	499
Índice alfabético	503