基本データ型上の関数

胡 振江
東京大学 計数工学科
2006 年度

内容

- 数型（整数型と浮動小数点数型）とその上の関数
- 論理型とその上の関数
- 文字型とその上の関数
- 文字列型とその上の関数

整数型とその上の関数

整数型 (Int) はすべての整数から構成されている.

$$0, 45, -3453, 214748091, \ldots$$

算術演算子	使用例
$+$ (加算)	$2 + 3 \Rightarrow 5$
$-$ (減算)	$2 - 3 \Rightarrow -1$
$*$ (乗算)	$2 * 3 \Rightarrow 6$
/ (除算)	$3/2 \Rightarrow 1.5$
^ (ベキ乗算)	$2\hat{}3 \Rightarrow 8$
div (整数除算)	div 3 2 $\Rightarrow$ 1
	3 ‘div‘ 2 $\Rightarrow$ 1
mod (整数除余)	mod 5 3 $\Rightarrow$ 2
	5 ‘mod‘ 3 $\Rightarrow$ 2

結合順位

二項算術演算子の結合順位は次のようになる.（結合順位の高いものから順にしめしてある.）

べき乗算演算子	^
乗除演算子	$*$ / ‘div‘ ‘mod‘
加減演算子	$+$ $-$

注：**関数適用の結合は他のどの演算子よりも強い**

例：

$$3\hat{}4 * 5 + 2 = (3\hat{}4 * 5) + 2$$

$$\text{square } 3 * 4 = (\text{square } 3) * 4$$

演算子とセクション

- セクション：括弧でくくられた演算子

$$\begin{array}{lcl} (+) & :: & \mathsf{Int} \to \mathsf{Int} \to \mathsf{int} \\ (+)\ x\ y & = & x + y \end{array}$$

括弧でくくられた演算子が普通の prefixed 関数のように引数を適用することができる．また，引数として関数に渡したりすることができる．

$$\mathsf{both}\ f\ x = f\ x\ x$$

と定義すると，

$$\mathsf{both}\ (+)\ 3\ \Rightarrow\ (+)\ 3\ 3\ \Rightarrow\ 3+3\ \Rightarrow\ 6$$

- 更に拡張： 引数を演算子とともに括弧でくくる.

$$
\begin{aligned}
(x\oplus)\ y &= x \oplus y \\
(\oplus y)\ x &= x \oplus y
\end{aligned}
$$

例：

$$
\begin{aligned}
(*2): &\quad 2\text{倍する関数} \\
(1/): &\quad \text{逆数を求める関数} \\
(/2): &\quad 2\text{分する関数} \\
(+1): &\quad \text{つぎの値を得る関数}
\end{aligned}
$$

浮動小数点数型とその上の関数

浮動小数点数型 (Float) はすべての浮動小数点数から構成されている.

$$0.0, 4.5, -34.53, 2147.48091, \ldots$$

演算子	使用例
$+$ (加算)	$2.3 + 3.3 \Rightarrow 5.6$
$-$ (減算)	$2.5 - 3 \Rightarrow -0.5$
$*$ (乗算)	$2.5 * 2.5 \Rightarrow 6.25$
/ (除算)	$3.2/2 \Rightarrow 1.6$

数型上の関数の定義

例： 数の絶対値を返す関数 abs.

$$\begin{array}{lcl} \mathsf{abs} & :: & \mathsf{Num}\ a \Rightarrow a \to a \\ \mathsf{abs}\ x & = & \mathbf{if}\ x < 0\ \mathbf{then}\ -x\ \mathbf{else}\ x \end{array}$$

読みやすいために，次のように書いてもよい.

$$\begin{array}{lll} \mathsf{abs}\ x & \mid x < 0 & = -x \\ & \mid \mathbf{otherwise} & = x \end{array}$$

整数の符号を計算する関数 sign.

$$
\begin{array}{lcll}
\mathsf{sign} & :: & \mathsf{Int} \to \mathsf{Int} & \\
\mathsf{sign}\ n & | & n > 0 & = 1 \\
 & | & n == 0 & = 0 \\
 & | & n < 0 & = -1
\end{array}
$$

論理型とその上の関数

論理型 (Bool) は True と False だけを含む.

比較演算子	例
== (等しい)	$1 == 1$
/ = (等しくない $\neq$)	True / = False
< (より小さい)	$4 < 5$
> (より大きい)	$5 > 4$
<= (より小さいかまたは等しい $\leq$)	$4 <= 5$
>= (より大きいかまたは等しい $\geq$)	$4 >= 5$

論理演算子	例
&&	論理積 $\wedge$
||	論理和 $\vee$
not	論理否定 $\neg$

論理型上の関数の定義

xor:

$$\begin{aligned} \mathsf{xor} &:: \mathsf{Bool} \to \mathsf{Bool} \to \mathsf{Bool} \\ \mathsf{xor}\ p\ q &= (p \wedge \neg q) \vee (\neg p \wedge q) \end{aligned}$$

imply:

$$\begin{aligned} \mathsf{imply} &:: \mathsf{Bool} \to \mathsf{Bool} \to \mathsf{Bool} \\ \mathsf{imply}\ p\ q &= \neg p \vee q \end{aligned}$$

leap: 閏年を判定する関数.

$$\begin{aligned} \mathsf{leap} &:: \mathsf{Int} \to \mathsf{Bool} \\ \mathsf{leap}\ y &= y\ \text{`}\mathsf{mod}\text{`}\ 4 == 0 \wedge \\ &\quad \mathsf{imply}\ (y\ \text{`}\mathsf{mod}\text{`}\ 100 == 0)\ (y\ \text{`}\mathsf{mod}\text{`}\ 400 == 0) \end{aligned}$$

文字型とその上の関数

文字型 (Char) は ASCII (American Standard Code for Information Interchange) 文字の集まりである.

上位3ビット→ / ↓下位4ビット	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAC	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF/NL	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

関数

- ord :: Char → Int: 文字を対応する ASCII 符号の整数に変換する.

$$\text{ord } 'b' \Rightarrow 98$$

- chr :: Int → Char: ASCII 符号の整数を対応する文字に変換する.

$$\text{chr } 98 \Rightarrow 'b'$$

- 関係演算子：文字の間は比較できる.

文字型上の関数の定義

- idDigit: 文字が数字であることを判定する関数.

$$\begin{array}{lll} \mathsf{isDigit} & :: & \mathsf{Char} \to \mathsf{Bool} \\ \mathsf{isDigit}\ x & = & '0' \leq x \wedge x \leq' '9' \end{array}$$

- capitalise: 小文字を大文字に変える関数.

$$\begin{array}{lll} \mathsf{captalise} & :: & \mathsf{Char} \to \mathsf{Char} \\ \mathsf{capitalise}\ x & | & \mathsf{isLower}\ x\ = \mathsf{chr}(\mathsf{offset} + \mathsf{ord}\ x) \\ & | & \mathbf{otherwise}\ = x \\ & & \mathbf{where}\ \mathsf{offset} = \mathsf{ord}\ 'A' - \mathsf{ord}\ 'a' \end{array}$$

文字列型とその上の関数

文字列型 (String) は文字の列の集まりである.

$$\text{"\,"},\ \text{"hello"},\ \text{"This is a string."}$$

- show :: $a \to$ String: 任意の型のデータを文字列に変換する.

$$\begin{aligned}
&\text{show } 100 \ \Rightarrow\ \text{"100"}\\
&\text{show True} \ \Rightarrow\ \text{"True"}\\
&\text{show } (\text{show } 100) \ \Rightarrow\ \text{"\textbackslash"100\textbackslash""}
\end{aligned}$$

- $+\!\!+$ :: String $\to$ String $\to$ String: 二つの文字列をつなぐ連接演算子.

$$\text{"hello"} +\!\!+ \text{" "} +\!\!+ \text{"world"} \ \Rightarrow\ \text{"hello world"}$$

- 比較演算子：文字列の比較は通常の辞書式順に従う.

練習問題

- Hugs システムを使って，基本型上の関数をテストする．
- 教科書の 2.1-2.3 を読み，教科書中の練習問題を考える．