

第2章:基本データ型

胡 振江

数型

- 整数 Int
 - 42 -42 0 100 ...
- 浮動小数点 Float
 - -42.10 42.10 0.0 100.0
- 算術演算子

+	加算	$2+3 \Rightarrow 5$
-	減算	$2-3.0 \Rightarrow -1.0$
*	乗算	$4*5 \Rightarrow 20$
/	除算	$8/3 \Rightarrow 2.66667$
^	べき乗算	$2^3 \Rightarrow 8$
div	整数除算	$\text{div } 5 \ 3 \Rightarrow 1$
mod	整数剰余	$\text{mod } 5 \ 3 \Rightarrow 2$

結合順位、順序

- 結合順位(異なる演算):
関数適用: 一番強い
^
* / div mod
+ -
 - 例
 - $3^4 * 5$
 - $3 * 7 + 4.1$
 - square $3 * 4$
 - 結合順序(同じの演算)
 - 左結合: $5-4-3$
 - 右結合: 5^4^3
 - 結合性: $5+4+3$

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
 $(+) :: \text{Num } a \Rightarrow a \rightarrow a \rightarrow a$
 $(+) \ x \ y = x + y$
- 更に拡張: 引数を演算子とともに括弧でくくる
 $(x \oplus) \ y = x \oplus y$
 $(\oplus x) \ y = y \oplus x$
 - $(*2)$: 2倍する 関数
 - $(1/)$: 逆数を求める 関数
 - $(/2)$: 2分する 関数
 - $(+1)$: つぎの値を得る 関数

例題:平方根の計算sqrt

- 仕様
 - $X \geq 0$ の時、 $\text{sqrt } x \geq 0$ かつ $(\text{sqrt } x)^2 = x$
 - この仕様が平方根を計算する方法を述べているのではない
 - 実際の計算機上の有限精度の算術計算うい許容しない。



小さい $\text{eps} > 0$ に対して
 $\text{sqrt } x \geq 0$ かつ $\text{abs}((\text{sqrt } x)^2 - x) \leq \text{eps}$

例題:平方根の計算sqrt (cont)

- Newton法
 $y(n+1) = (y(n) + x / y(n)) / 2$
例: 2の平方根の計算
 - $y(0) = 2$
 - $y(1) = (2+2/2)/2 = 1.5$
 - $y(2) = (1.5+2/1.5)/2 = 1.4167$
 - $y(3) = (1.4167+2/1.4167)/2 = 1.4142157$
 - ...

例題: 平方根の計算 sqrt (cont)

- 近似値yから新しい近似値を生成する関数
improve x y = (y + x/y) / 2
- 終了条件を判定する関数
satis x y = abs (y^2 - x) < eps
- ある条件pが真になるまで初期値に関するfを繰り返し適用する関数
until p f x | p x = x
| otherwise = until p f (f x)
- メイン関数
sqrt x = until (satis x) (improve x) x

プログラム sqrt.hs

```
sqrt1 x = until (satis x) (improve x) x
where
  improve x y = (y + x/y) / 2
  satis x y = abs (y^2 - x) < eps
  eps = 0.0001
```

前回のチェック

- 教科書の購入
- Hugsのインストール
- curry関数の定義は?
- sqrt.hs の理解
- (+(-x))はどんな関数か?
- 正しいのはどれ?
 - (*) x = (*x)
 - (+) x = (x+)
 - (-) x = (-x)

論理型 Bool

- 論理値: True, False
- 述語: 論理値を返す関数
 - E.g. even :: Int → Bool
- 比較演算子
 - == 等しい 1==1
 - /= 等しくない True /= False
 - < より小さい
 - > より大きい
 - <= より小さいかまたは等しい
 - >= より大きいまたは等しい
- 論理演算子
 - && 論理和 and p q (p `and` q)
 - || 論理積 or p q
 - not 論理否定 not p

例題: 閏年の判定

- 閏年とは、4で割り切れる年であるが、100で割り切れるならば400でも割り切れなくてはならない。

```
leap y = y `mod` 4 == 0 &&
        (y `mod` 100 /= 0 || y `mod` 400 == 0)
```

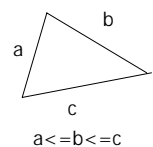
または

```
leap y | y `mod` 100 == 0 = y `mod` 400 == 0
      | otherwise        = y `mod` 4 == 0
```

その他の例題

```
analysis a b c
| a+b<=c      = 0
| a==b && b==c = 1
| a/=c && (a==b || b==c) = 2
| a<b && b<c   = 3
```

```
Pred n | n==0 = 0
      | n>0   = n-1
```



文字と文字列

- 文字型 Char
‘a’ ‘7’ ‘ ’
- 基本関数
 - $\text{ord} :: \text{Char} \rightarrow \text{Int}$
 - $\text{chr} :: \text{Int} \rightarrow \text{Char}$
 - 例: $\text{ord } 'b' \rightarrow 98$
 $\text{chr } 98 \rightarrow 'b'$
 $\text{chr } (\text{ord } 'b' + 1) \rightarrow 'c'$

例題

- 文字が数字であることを判定する関数
 $\text{isDigit } x = '0' \leq x \ \&\& \ x \leq '9'$
- 小文字を主時に変える関数
 $\text{capitalise } x$
 - | $\text{isLower } x = \text{chr } (\text{offset} + \text{ord } x)$
 - | otherwise = xwhere $\text{offset} = \text{ord } 'A' - \text{ord } 'a'$

文字列

- 文字列型 String (e.g., [Char])
“a” “hello”
- 文字列の比較は通常の辞書式順に従う
“hello” > “hallo”
“Jo” < “Joanna”
- 関数
 - $\text{show} :: a \rightarrow \text{String}$
 $\text{show } 100 \rightarrow \text{“100”}$
 $\text{show } \text{True} \rightarrow \text{“True”}$
 $\text{show } (\text{show } 100) \rightarrow \text{“”100””}$
 - 文字列をつなぐ接続演算子 ++
“hello” ++ “ ” ++ “world” \rightarrow “hello world”

組

- (T_1, T_2, \dots, T_n)
 - $(17.3, '+') :: (\text{Float}, \text{Char})$
 - $(3, 6) :: (\text{Int}, \text{Int})$
- 順序: 辞書式順序
 - $(\text{“s”}, 4) < (\text{“s”}, 5)$
- 関数
 - $\text{fst } (x, y) = x$
 - $\text{snd } (x, y) = y$

例題1: 2次方程式

- 2次方程式の根を求める関数
 $\text{roots} :: (\text{Float}, \text{Float}, \text{Float}) \rightarrow (\text{Float}, \text{Float})$
 $\text{roots } (a, b, c) \mid d \geq 0 = (r_1, r_2)$
where
 - $r_1 = (-b + r) / (2 * a)$
 - $r_2 = (-b - r) / (2 * a)$
 - $r = \text{sqrt } d$
 - $d = b^2 - 4 * a * c$

例題2: 有理数

- 有理数の表現: 対
 $x/y \rightarrow (x, y)$
- 問題:
 - 有理数の正規化 $(18, 16) \rightarrow (9, 8)$
 - 有理数の四則演算
 - 有理数の比較
 - 有理数の表示

有理数の正規化

$\text{norm } (x,y) \mid y \neq 0 = (u \text{ `div` } d, v \text{ `div` } d)$
where $u = \text{sign } y * x$
 $v = \text{abs } y$
 $d = \text{gcd } (\text{abs } u) \ v$

$\text{sign } x \mid x > 0 = 1$
 $\mid x = 0 = 0$
 $\mid x < 0 = -1$

有理数上の四則演算

$\text{radd } (x,y) \ (u,v) = \text{norm } (x*v + u*y, y*v)$
 $\text{rsub } (x,y) \ (u,v) = \text{norm } (x*v - u*y, y*v)$
 $\text{rmul } (x,y) \ (u,v) = \text{norm } (x*u, y*v)$
 $\text{rdiv } (x,y) \ (u,v) = \text{norm } (x*v, y*u)$

有理数の比較

$\text{compare' op } (x,y) \ (u,v) = \text{op } (x*v) \ (y*u)$

$\text{requals} = \text{compare' } (==)$
 $\text{rless} = \text{compare' } (<)$
 $\text{rgreater} = \text{compare' } (>)$


有理数の表示

$\text{showrat } (x,y)$
= if $v = 1$ then show u
else show $u \ ++ \ "/" \ ++$ show v
where $(u,v) = \text{norm } (x,y)$

パターン (1)

- 等式の左側にパターンを用いて関数を定義することができる。

- 論理値パターン

$\text{cond True } x \ y = x$
 $\text{cond False } x \ y = y$

 $\text{cond } p \ x \ y \mid p == \text{True} = x$
 $\mid p == \text{False} = y$

パターン (2)

- 自然数(負でない整数)パターン

$\text{pred } 0 = 0$
 $\text{pred } (n+1) = n$

 $\text{count } 0 = 0$
 $\text{count } 1 = 1$
 $\text{count } (n+2) = 2$

関数

- 関数はあらゆる型の値を引数にとりうるし、あらゆる種類の値を結果として返すことができる。
 - 例: 高階関数
 - 引数として関数をとる、あるいは
 - 結果として関数を返す
- 微分演算子: 引数 - 関数
結果 - 導関数

→ 関数の性質

関数合成

- 二つの関数を合成する演算子 .
 $(.) :: (b \rightarrow c) \rightarrow (a \rightarrow b) \rightarrow (a \rightarrow c)$
 $(f . g) x = f (g x)$
- 結合的
 $(f . g) . h = f . (g . h)$

(二項)演算子

- 二項演算子は関数とよく似ている。異なる点は2つの引数の前に置くのではなく間に書くということだけある。
- セクション: 演算子 → 関数
 $\otimes \oplus \clubsuit \diamond \heartsuit \spadesuit$
 $2 + 3 \rightarrow (+) 2 3 \rightarrow (2+) 3 \rightarrow (+3) 2$
- バッククオート: 二引数関数 → 演算子
 $\text{div } 5 3 \rightarrow 5 \text{ `div ` } 3$

逆関数

- 単射関数
 - $\forall x \text{ in } A, y \text{ in } B. f x == f y \rightarrow x == y$
- 全射関数
 - $\forall y \text{ in } B, \exists x \text{ in } A. f x = y$
- 逆関数
 - $f^{-1}(f x) = x$
 - 例 $f x = (\text{sign } x, \text{abs } x)$
 $f^{-1}(s, a) = s * a$

お知らせ

- 次回の講義:
教育用計算機センター5階
8:30から

正格関数と非正格関数

- 正格関数
 - 定義: $f \perp = \perp$
 - 例: $\text{square } (1/0) = \perp$
- 非正格関数: 正格でない関数
 - 例: $\text{three } x = 3$
 $> \text{three } (1/0)$
 3

非正格な意味論の利点

- 相等性に関する議論しやすい
 - $2 + \text{three } x = 5$
 - 単純で統一的な置換操作
 - プログラムの正当性を議論しやすい
- 関数を定義して、新たら制御構造を定義することができる。


```
cond p x y | p      = x
          | otherwise = y
recip x = cond (x==0) 0 (1/x)
正格な意味論では recip 0 = ⊥
非正格な意味論では recip 0 = 0
```

簡約戦略

$f \ x_1 \ x_2 \ \dots \ x_n$ の評価

- 先行評価
 - 引数優先評価戦略
 - x_1, x_2, \dots, x_n を評価したら f を評価する。
- 遅延評価
 - (外側の)関数優先評価戦略
 - f をまず評価する。

型の同義名

- 距離、角度、位置を引数にとり、角度と距離で示される新しい位置に場所を移動する関数 `move`:


```
move :: Float → Float → (Float, Float) → (Float, Float)
move d a (x, y) = (x + d * cos a, x + d * sin a)

type Position = (Float, Float)
type Angle = Float
type Distance = Float

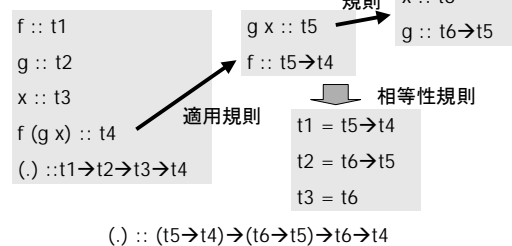
move :: Distance → Angle → Position → Position
```

型推論

- 適用規則
 - $f \ x :: t \rightarrow \exists t'. x :: t', f :: t' \rightarrow t$
- 相等性規則
 - $x :: t, x :: t' \rightarrow t = t'$
- 関数の規則
 - $t \rightarrow u = t' \rightarrow u' \rightarrow t = t', u = u'$

(.) $f \ g \ x = f \ (g \ x)$

引数名と結果に型を割り当てる



$f \ x \ y = \text{fst } x + \text{fst } y$

仮定

$\text{fst} :: (a, b) \rightarrow a$

$(+) :: \text{Int} \rightarrow \text{Int} \rightarrow \text{Int}$

