

プログラミング言語論 #12

Higher Order Functions



2018-07-09



型による関数の拡張に話を戻し，データと関数の垣根を取り除く

C 言語標準関数 qsort

SYNOPSIS

```
#include <stdlib.h>

void qsort(void *base, size_t nmemb, size_t size,
           int(*compar)(const void *, const void *)) );

qsort - 配列を並べ替える
```

qsort 関数は、nmemb 個の大きさ size の要素をもつ配列を並べ変える。base 引数は配列の先頭へのポインタである。

比較関数（ポインタ）compar によって、配列の中身は昇順（値の大きいものほど後に並ぶ）に並べられる。比較関数の引数は比較されるふたつのオブジェクトのポインタである。

比較関数は、第 1 引数が第 2 引数に対して、1) 小さい、2) 等しい、3) 大きいのそれぞれに応じて、1) 負数、2) 0、3) 正数のいずれかを返さなければならない。

```
1  比較関数のプロトタイプ宣言; // qsortの宣言から自明
2
3  int main() {
4      // vecはソートされてない
5      qsort(vec, ..., ..., 比較関数);
6      // vecのソート終了
7  }
8
9  比較関数 (... a, ... b) {
10     .... = (cast) a;
11     .... = (cast) b;
12     if (...) return -1; // 小さい
13     if (...) return 0;  // 等しい
14     if (...) return -1; // 大きい
15 }
```

括弧の意義

$$f :: \text{Int} \rightarrow \text{Int} \rightarrow \text{Int}$$
$$g :: \text{Int} \rightarrow (\text{Int} \rightarrow \text{Int})$$
$$h :: (\text{Int} \rightarrow \text{Int}) \rightarrow \text{Int}$$

4つの組合せ

$$a \rightarrow b \rightarrow c$$

$$(a \rightarrow b) \rightarrow c$$

$$a \rightarrow (b \rightarrow c)$$

$$(a \rightarrow b) \rightarrow (c \rightarrow d)$$

プログラミング言語論専門用語

15

高階関数; higher order function

引数が関数である，または返値が関数である関数

利点



部分

部分

-

Haskell における高階関数

```
1  -- 簡単のため一旦型クラスの存在を忘れる
2  ni :: Int
3  ni == 2
4
5  f1 ::
6  f1 f = (f ni) < ni
7
8  -- | f1よりレベルが高い
9  f2 ::
10 f2 f = f ni
11
12 -- | 型クラスが必要なのでさらにレベルが高い
13 f3 ::
14 f3 f x = (f 2) < x
```

識別子の名前の自由度

```
1  g x = x a
2  x y = y a < 10
3  x y z = y z < z
```


qsort に見る C と Haskell の対応

C

```
void qsort(void *base, size_t nmemb, size_t size,  
           int(*compar)(const void *, const void *));
```

qsort 関数は、nmemb 個の大きさ size の要素をもつ配列を並べ変える。base 引数は配列の先頭へのポインタである。

比較関数 (ポインタ) compar によって、配列の中身は昇順 (値の大きいものほど後に並ぶ) に並べられる。比較関数の引数は比較されるふたつのオブジェクトのポインタである。

比較関数は、第 1 引数が第 2 引数に対して、1) 小さい、2) 等しい、3) 大きいのそれぞれに応じて、1) 負数、2) 0、3) 正数のいずれかを返さなければならない。


Haskell

```
-- | 近似解 (ポインタはない, 配列ではない)  
qsort :: [a] -> Int -> Int -> (a -> a -> Int) -> [a]  
-- | 本当は大小比較をするはずなので (上記は型制約を無視)  
qsort :: Ord a => [a] -> Int -> Int -> (a -> a -> Int) -> [a]
```

型宣言に合わせた関数定義の練習（入門篇）

使ってよいのはこれまでの資料に出てきた関数，演算子，型のみとする（注意：型変数は型の一種）

```
1  -- 型クラスを導入しなくてよいように型宣言付きの変数を用意
2  ni :: Int
3  ni = ni
4
5  f1 :: (Int -> Bool) -> Bool
6  f1
7
8  f2 :: (Int -> Int) -> [Int]
9  f2
10
11 f3 :: Int -> (Int -> a) -> (a -> b) - b
12 f3
```

 返値が関数になっている高階関数は現在の知識で定義可能か？

期末試験，小テストの要求レベル

使ってよいのはこれまでの資料に出てきた関数，演算子，型，
Eq, Num, Ord の型クラスとする。

型推論問題

```
1 h1 ::  
2 h1 x y = (x y) == y  
3  
4 h2 ::  
5 h2 x y = x < (y False)  
6  
7 h3 ::  
8 h3 x = (x True, x)
```

宣言に合わせた定義問題

```
1 -- 型クラス制約の導入  
2 g1 :: Ord a => a -> (a -> Bool) -> Bool  
3 g1  
4  
5 g2 :: (Num a, Num b) => (a -> b) -> a -> b -> [b]  
6 g2
```