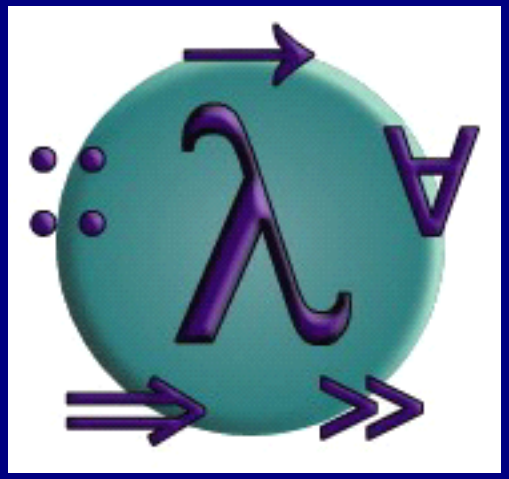
プログラミング Haskell



第2章　はじめの一歩

目　次

1.Hugs

Haskell には2つのメジャーな処理系があります。

1.インタープリターの Hugs

http://www.haskell.org/hugs/ からダウンロードできます。

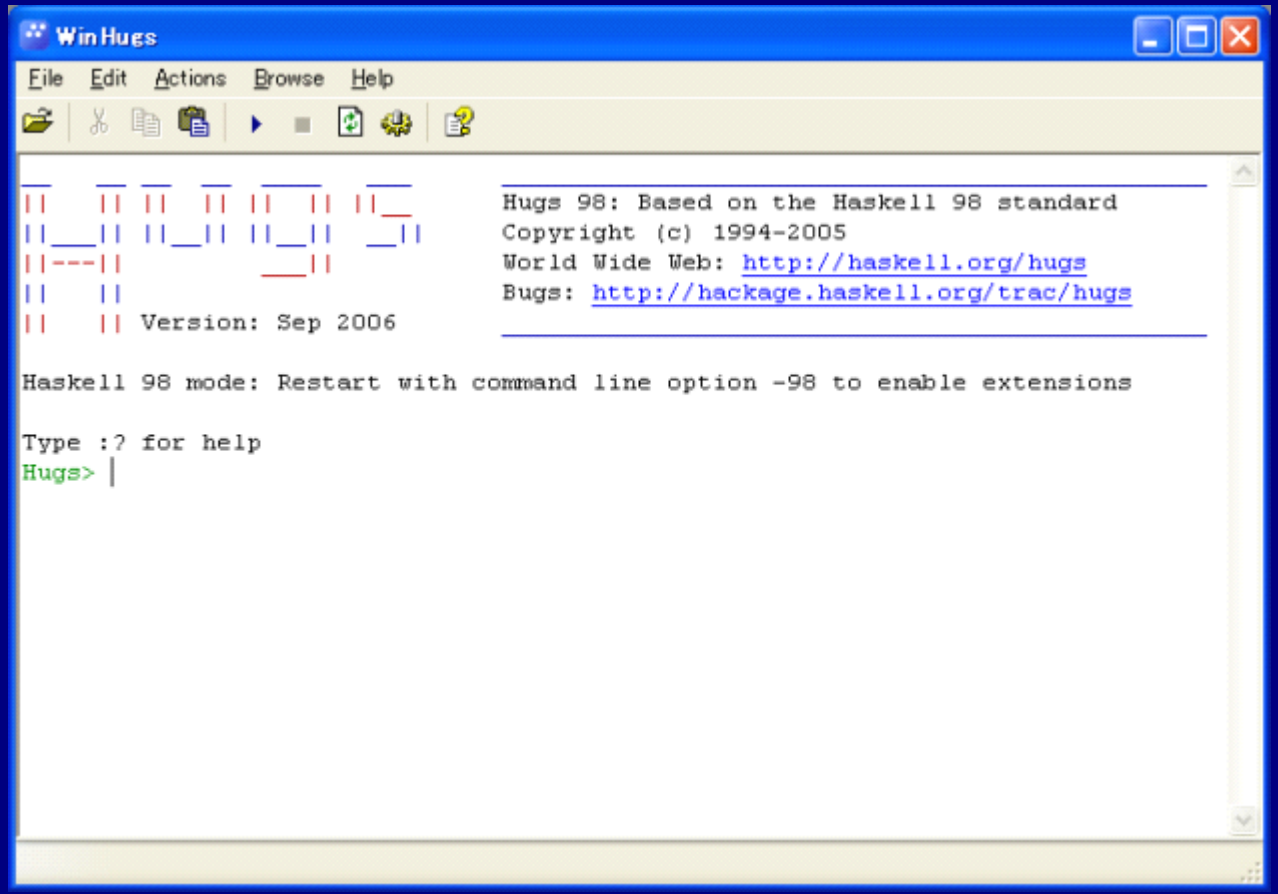
2.コンパイラーの GHC

http://www.haskell.org/ghc/ からダウンロードできます。

プログラムの開発、テストには、Hugs を使い、出来上がった
プログラムを GHC でコンパイルするのがよいと思います。

2.標準ライブラリ

Hugs を起動すると、最初に Prelude.hs という標準ライブラリを読み込む。



2.標準ライブラリ

Hugs>プロンプトは、Hugsシステムを式を評価する準備ができている。

例

```
Hugs> 2+3
5
Hugs> 2-3
-1
Hugs> 2 * 3
6
Hugs> 7 `div` 2
3
Hugs> 2 ^ 3
8
```

`はバッククォート
（Shift + @）
本では↑で表現

2.標準ライブラリ

演算子の優先順位（優先順位が高いほど強く結合する）

演算子	優先順位	結合性
!!	9	左
.	9	右
^	8	右
^^	8	右
**	8	右
*	7	左
/	7	左
`div`	7	左
`mod`	7	左
`rem`	7	左
`quot`	7	左

+	6	左
-	6	左
:	5	右
++	5	右
==	4	非
\=	4	非
<	4	非
<=	4	非
>	4	非
>=	4	非
`elem`	4	非
`notElem`	4	非

&&	3	右
||	2	右
>>	1	左
>>=	1	左
$	0	右
$!	0	右
`seq`	0	右

2.標準ライブラリ

標準ライブラリには、リストを処理する多彩な関数も提供している。

空でないリストの先頭の要素を取り出す

```
Hugs> head [1,2,3,4,5]
1
```

空でないリストの先頭の要素を取り除く

```
Hugs> tail[1,2,3,4,5]
[2,3,4,5]
```

空でないリストの(0番目から数えて)n番目の要素を取り出す

```
Hugs> [1,2,3,4,5]!!2
3
```

2.標準ライブラリ

リストの先頭からn番目の要素を取り出す

```
Hugs> take 3[1,2,3,4,5]
[1,2,3]
```

リストから先頭のn個の要素を取り除く

```
Hugs> drop 3[1,2,3,4,5]
[4,5]
```

リストの長さを計算する

```
Hugs> length[1,2,3,4,5]
5
```

数値のリストに対し要素の和を計算する

```
Hugs> sum[1,2,3,4,5]
15
```

2.標準ライブラリ

数値のリストに対し要素の積を計算する

```
Hugs> product [1,2,3,4,5]
120
```

二つのリストを連結する

```
Hugs> [1,2,3] ++ [4,5]
[1,2,3,4,5]
```

リストを逆順にする

```
Hugs> reverse[1,2,3,4,5]
[5,4,3,2,1]
```

0で割る or 空リストの最初の要素を取り出す

```
Hugs> 1 `div` 0  or  head[]
Error
```

3.関数適用

数学では関数を適用する場合、引数を括弧で囲む。
二つの値を掛ける場合、単に一方を他方に続けて書く

```
f(a,b) + c d
```

関数fを二つの値aとbに適用し、cとdの積を足し合わせるる

Haskellでは関数の適用は単に空白文字を使う。
乗算には乗算演算子 ＊ を明記する。

```
f a b + c * d
```

関数適用は、他の全ての演算子よりも高い結合順位を持つ

f a + bは、f(a + b)ではなく (f a) + b である。

3.関数適用

数学とHaskellの関数適用表記の違い

例	数学	Haskell
	f(x)	f x
	f(x,y)	f x y
	f(g(x))	f (g x)
	f(x,g(y))	f x (g y)
	f(x)g(y)	f x * g y

4.Haskellプログラム

- 標準ライブラリで提供されている機能と同様に機能を定義できる。
- エディタを使ってスクリプトを作成し、スクリプトのファイル名には、拡張子を***.hs***とする。

テキストエディタで入力して、test.hsで保存

```
double x = x + x

quadruple x = double (double x)
```

保存先は、わかりやすいところがいいよね。

4.Haskellプログラム

loadコマンドでスクリプトを読み込むことが出来る。
標準ライブラリとtest.hsの両方の関数が使用できる。

```
Hugs> :load test.hs
```

```
Hugs> :load "c:¥Haskell¥test.hs"
```

test.hs をダブルクリックする。

```
Main> quadruple 10
40

Main> take (double 2) [1,2,3,4,5,6]
[1,2,3,4]
```

4.Haskellプログラム

test.hs を下記コードに書き換え保存します。

```
factorial n = product [1..n]

average ns  = sum ns `div` length ns
```

ファイルを書き換えても、自動で再読み込みはしません。その場合、reloadコマンドを実行する。

```
Main> :reload

Main> factorial 10
3628800

Main> average [1,2,3,4,5]
3
```

4.Haskellプログラム

コマンド	意味
:load *name*	ファイル *name* をロードする
:reload	現在のファイルを再びロードする
:edit *name*	ファイル *name* を編集する
:edit	現在のファイルを編集する
:type *expr*	expr の型を表示する
:?	すべてのコマンドを表示する
:quit	Hugs を終了する

最初の1文字に短縮することもできる。例 :load → :l

4.Haskellプログラム

- 命名規則
  新しく関数を定義する場合、関数と引数の
  名前は、小文字で始める。

  myFun　　**fun1**　　**arg_2**　　**x'**

  引数がリストの場合、名前の最後にsを付け
  複数の値を表現することを示す慣習がある。

  xs　　**ns**　　**css**

4.Haskellプログラム

以下のキーワードは、特別な意味を持つため
関数名や引数には使えない。

case	class	data	default	deriving	do
else	if	import	in	infix	infixl
infixr	instance	let	module	newtype	of
then	type	where			

4.Haskellプログラム

- レイアウト規則
  あるレベルの定義は、完全に同じカラムから始める。

```
a = 10
b = 20
c = 30
```

```
a = 10
 b = 20
c = 30
```

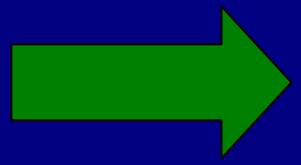
```
 a = 10
b = 20
 c = 30
```

4.Haskellプログラム

行頭揃えによりグループ化できる。また、波括弧の中に定義をセミコロンで区切ってもよい。

```
a = b + c
    where
      b = 1
      c = 2
d = a * 2
```

暗黙的なグループ化



```
a = b + c
    where
      {b = 1;
       c = 2}
d = a * 2
```

明白的なグループ化

4.Haskellプログラム

- コメント

  普通のコメントは、「--」で始まり、行末まで

  囲みコメントは、「{-」で始まり、「-}」まで

```
-- 製の整数の階乗
factorial n = product [1..n]
-- 整数のリストの平均
average ns = sum ns `div` length ns
```

```
{-
double x = x + x
quadruple = double (double x)
-}
```

5.この章の参考文献

- HugsやGHC

  http://www.haskell.org
- Programming in Haskell 解答やスライド等
  http://www.cs.nott.ac.uk/~gmh/book.html

6.練習問題

1.次の式に括弧を付けよ。

```
2 ^ 3 * 4
2 * 3 + 4 * 5
2 + 3 * 4 ^ 5
```

↑は累乗なので「^」で記述

2.この章の例題を*Hugs*を用いて実行せよ。

3.以下のプログラムにはエラーが3つある。
エラーを修正し、*Hugs*で正しく動くか確かめよ。

```
N = a `div` length xs
    where
        a = 10
      xs = [1,2,3,4,5]
```

6.練習問題

4.この章で紹介したライブラリ関数を使って、空でないリストの最後の要素を取り出す関数 *last* を定義せよ。さらに他の定義も考えよ。

5.同様に、空でないリストから最後の要素を取り除くらいライブラリ関数 *init* が、二通りの方法で定義できることを示せ。