チュートリアル

○Cherryとは

「Cherry」とは私が自作した処理系の名前であり、由来は日本の伝統的な花である「桜」から来ている。もともと桜という花は好きだったが、留学中のアメリカで、このSakuraの知名度がとても高く、日本人としてこの花を誇りに思えた。そのため、自分の初の処理系として、日本から発信し本物の「桜」のような知名度をもってもらいたいという幻想的な発想から「Cherry」と名付けた。Sakuraにしようと思ったが、この処理系はRubyで書かれており、そのRubyをリスペクトする意味を含めて「Cherry」とした。

○Cherryでできること

この処理系ではC言語ライクな記述の仕方が出来る。具体的には以下。

文：print文、if文、while文、for文、function文、call文
演算子：四則演算、剰余、等号、不等号、代入
その他：グローバル変数、ブロック、標準入力

function文とは一連の動作を保持するもの。call文はfunction文の呼び出し。記述の方法は、Cherryソース解説.pdfの構文式から抜粋。但し、文には演算子も含まれる。

```
print文 = ‘print’ andor(式5)
while文 = ‘while’ andor(式5) 文
if文 = ‘if’ andor(式5) 文 ( ‘else’ 文 )?
function文 = ‘function’ 変数 文
call文 = ‘call’ 変数
for文 = ‘for’ 文 ‘|’ 文 ‘|’ 文 ‘|’ 文

文字列 = ‘’’ 空白文字以外 ‘’’
変数 = 大文字を含めたa-zで始まる英数字
グローバル変数 = ‘#’ 大文字を含めたa-zで始まる英数字
```

等号と代入に関しては通常と異なり、それぞれ以下の通り。

= : <-
== : =-=
!= : =!=

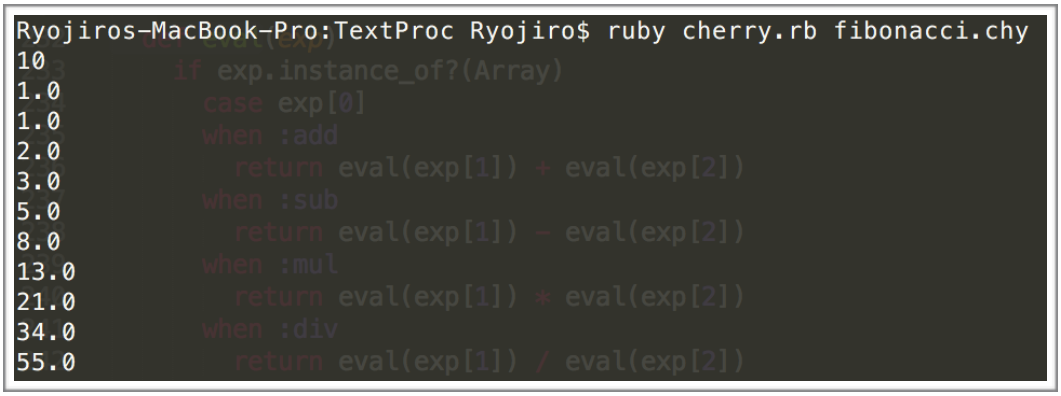
○とりあえずフィボナッチ数列

fibonacci.chy

```
function fibonacci {
  x <- 1
  y <- 1
  if #outputNum =!= 0 then print 1
  for i <- 0| i < #outputNum - 1| i <- i + 1 {
    print x
    temp <- x
    x <- y
    y <- temp
    x <- x + y
  }
}
#outputNum <- input
fibonacci()
```

これは標準入力を受け付け、その数だけフィボナッチ数列を順番に出力するプログラムである。変数の前に#をつけることで、その変数はグローバル変数となる。関数を呼び出す際に、プログラム内では新しくインスタンスを立ち上げているため、特定の変数を関数内で利用したいときはグローバル変数を使う必要がある。このプログラムでは、フィボナッチ数列を出力する部分を関数化しており、入力された値をグローバル変数に渡し、その後その関数を呼び出している。関数の呼び出し方は2通りあり、「call　関数名」とするか、「関数名＋()」とする。実行結果は以下の通り。

```
Ryojiros-MacBook-Pro:TextProc Ryojiro$ ruby cherry.rb fibonacci.chy
10
1.0
1.0
2.0
3.0
5.0
8.0
13.0
21.0
34.0
55.0
```



○全ての機能を網羅する

ー演算子編

これは説明不要だと思われるため、ソースコードと実行結果のみ。

operator.chy

```
print '1+1'
print 1 + 1
print '10-2'
print 10 - 2
print '3+4'
print 3 * 4
print '10/2'
print 10 / 2
print '10%3'
print 10 % 3

print '1=-=1'
print 1 =-= 1
print '1=!=1'
print 1 =!= 1
print '2<2'
print 2 < 2
print '2<=2'
print 2 <= 2
print '3>3'
print 3 > 3
print '3>=3'
print 3 >= 3

print '1&&0'
print 1 && 0
print '1||0'
print 1 || 0
```

実行結果

```
Ryojiros-MacBook-Pro:TextProc Ryojiro$ ruby cherry.rb operator.chy
1+1
2.0
10-2
8.0
3+4
12.0
10/2
5.0
10%3
1.0
1=-=1
1
1=!=1
0
2<2
```

```
0
2<=2
1
3>3
0
3>=3
1
1&&0
0
1||0
1
```

```
Ryojiros-MacBook-Pro:TextProc Ryojiro$ ruby cherry.rb operator.chy
1+1
2.0
10-2
8.0
3+4
12.0
10/2
5.0
10%3
1.0
1=-=1
1
1=!=1
0
2<2
0
2<=2
1
3>3
0
3>=3
1
1&&0
0
1||0
1
```

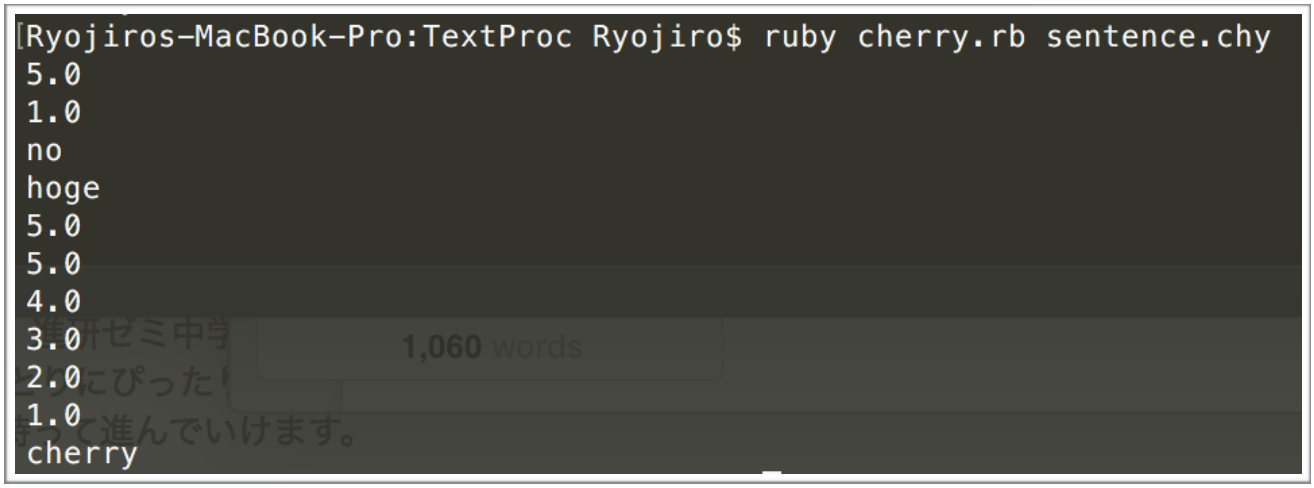
一文編

sentence.chy

```
x <- 5
#x <- 1
print x
print #x

if x =-= 1 then
  print 'yes'
else
  print 'no'

if #x =-= 1 then {
  print 'hoge'
  print x
}
else
  print 'hogehoge'

while x > 0 {
  print x
  x <- x - 1
}

function cherry print 'cherry'
call cherry
```

代入演算子を忘れていたのでこちらで。最初にグローバル変数とインスタンス変数が分けられていることを確認した。次にif文で、真のときと偽のときに分けられていることを確認。また2つ目のif文で、ブロックが機能していることを確認。そしてwhile文も同様に機能していることが確認出来た。最後に関数呼び出しが、callでも出来ることを確認。for文などは最初のfibonacci.chyで確認しているため省略。これで全ての機能が網羅していることが証明できた。実行結果は以下の通り。

```
Ryojiros-MacBook-Pro:TextProc Ryojiro$ ruby cherry.rb sentence.chy
5.0
1.0
no
hoge
5.0
5.0
4.0
3.0
2.0
1.0
cherry
```


Ryojiros-MacBook-Pro:TextProc Ryojiro$ ruby cherry.rb sentence.chy
5.0
1.0
no
hoge
5.0
5.0
4.0
3.0
2.0
1.0
cherry