プログラミング演習I
– 第6回 –
紙で考える＆言葉でプログラム＆ファイルの扱い

1　はじめに

第 6 回の演習の目標は以下の通り.

- プログラムの動作を追えるようにする.
- プログラムの間違い探し.
- 言葉で書かれた課題をプログラムとして設計する.
- リダイレクトを利用したファイルの扱い.

結構盛りだくさん.

2　紙で復習

2.1　プログラムの動作を追う

この節では，端末にプログラムを打ち込まずに頭と紙＆ペンで考える練習をしよう.

演習 6-1
次のプログラムの出力をプログラムを書かずに考えよう.

```
------------------------------------------------
i = 0
print(i, "\n")
print("--ここから while--\n")
while i < 3
  print(i, "\n")
  i = i + 1
end
print("--ここまで while--\n")
print(i, "\n")
------------------------------------------------
```

演習 **6-2**

次のプログラムの出力をプログラムを書かずに考えよう.

```
-----------------------------------------------
i = 0
print(i, "\n")
print("--ここから外 while--\n")
while i < 2
  j = 0
  print(i, ",", j, "\n")
  print("--ここから内 while--\n")
  while j < 3
    print(i, ",", j, "\n")
    j = j + 1
  end
  print("--ここまで内 while--\n")
  print(i, ",", j, "\n")
  i = i + 1
end
print("--ここまで外 while--\n")
print(i, ",", j, "\n")
-----------------------------------------------
```

演習 **6-3**

次のプログラムを実行した時にできるハッシュyasun_netaのキーと値の組を書いてみよう．キーボード入力は表の通りとする.

```
-----------------------------------------------
yasu_neta = Hash.new

i = 0
while i < 4
  print("ねた? >")
  neta = gets.chomp
  print("価格? >")
  kakaku = gets.chomp.to_i
  if kakaku < 300
    yasu_neta[neta] = kakaku
  end
  i = i + 1
end
-----------------------------------------------
```

キーボード入力		yasu_neta の中身	
ねた	価格	キー	値
kappa	120		
uni	300		
ohtoro	650		
anago	200		

3　なんかおかしいな？というときは...

プログラムを実行したけど，思った通りの結果にならないことは山ほどある．そんなときは，プログラムの処理を一つ一つ追っていくことが重要だ．コンピュータの中の小人さんになったつもりで，プログラムを実行してみよう．

演習 **6-4**
次のプログラム (行番号付き) は九九表を表示するプログラムのつもりだった．でも，なんかうまくいかない．

```
-----------------------------------------------
 1: i = 1
 2: j = 1
 3: while i <= 9
 4:   while j <= 9
 5:     print(i * j)
 6:     j = j + 1
 7:   end
 8:   print("\n")
 9:   i = i + 1
10: end
-----------------------------------------------
```

▼**i** や **j** の値を **1** 行 **1** 行処理する順に追って空欄を埋め，間違いを発見しよう．

行番号	操作内容	操作後の値	
		i	j
1	i に 1 を代入	1	-
2	j に 1 を代入	1	1
3	i(=1) と 9 を比較．i<=9 は真となり 4〜9 行目を実行する．	1	1
4	j(=1) と 9 を比較．j<=9 は真となり 5,6 行目を実行する．	1	1
5	i * j を表示する．結果は 1	1	1
6	j に 1 足す．	1	2
4	j(=2) と 9 を比較．j<=9 は真となり 5,6 行目を実行する．	1	2
5		1	2
6		1	
j が 3〜8 の時は同様（省略）			
4	j(=9) と 9 を比較．j<=9 は真となり 5,6 行目を実行する．	1	9
5	i * j を表示する．結果は 9	1	9
6		1	
4			

▼**i** や **j** の値を表示するプログラムに変更してみよう.

以下のように適宜 print メソッドで値を表示してみるとどこが間違っているかが明らかになる．上で手で書いた i, j と比較してみよう.

```
-----------------------------------------------
i = 1
j = 1
while i <= 9
  print("A ", i, ":", j, "\n")   # <=== ここで表示する
  while j <= 9
    print("B ", i, ":", j, "\n") # <=== ここで表示する
#    print(i * j)     # <=== ここはとりあえずコメントに
    j = j + 1
  end
  print("\n")
  i = i + 1
end
-----------------------------------------------
```

演習 **6-5**

次のプログラムはハッシュに値を登録するプログラムなんだけど... キーボードから apple に 80，orange に 70，persimmon に 50 と入力したらあれ ???

```
-----------------------------------------------
 1: kakaku_list = {"ringo" => 100, "orange" => 50, "banana" => 80}
 2:
 3: i = 0
 4: while i < 5
 5:   print("くだもの名?> ")
 6:   kudamono = gets.chomp
 7:   print("価格?> ")
 8:   kakaku = gets.chomp.to_i
 9:
10:   if kakaku_list[kudamono] > kakaku
11:     kakaku_list[kudamono] = kakaku
12:   else
13:     print("既に登録されています．\n")
14:   end
15:   i = i + 1
16: end
17:
18: # ハッシュの一覧表示
19: kakaku_list.each {|key, value|
20:   print(key, "は", value, "円です．\n")
21: }
-----------------------------------------------
```

▼最初にハッシュkakaku_list に格納されているものを書いてみよう

key	value

▼キーボード入力がある毎に実行される処理を追ってみよう

i	キーボード入力		10 行目の比較とその結果	その結果ハッシュの中身は？
	kudamono	kakaku		key / value
0	apple	80	100 > 80 は true. kakaku_list["apple"] に 80 が代入される	apple: 80 orange: banana: :
1	orange	70		apple: orange: banana: :
2	persimmon	50		apple: orange: banana: :

3.1 プログラムを言葉で書く

プログラムを設計する場合，いきなりプログラミング言語（本演習では ruby）で書くのではなく，まず言葉（擬似的な言語）で書いてみる方法がある．言葉で書くのは簡単で自然に表現できるので，大まかな設計ができる．そうしておいてから，それをプログラムに翻訳していくという手順を追うと，うまくいくことも多い．

演習 **6-6**
飲酒可能かどうかのプログラムを擬似的な言語で書くと以下のようになる．

```
------------------------------------------------
キーボードから年齢の入力を受け取る．

もし，受け取った年齢が 20 歳以上であれば
  飲酒 OK！でも，控えめに...
そうでなければ，
  飲んじゃダメ！まだ，未成年！
------------------------------------------------
```

この程度であれば，プログラムの構造を意識して書けると思う．さて，これを ruby プログラムにすると，次のようになる．未完成部分が 2 箇所あるので埋めてみよう．なお，言葉で書いた部分は適宜コメントとして残しておくとよい．

```
-----------------------------------------------
# キーボードから年齢の入力を受け取り，変数 age に代入する．
age = gets.chomp

if                    # もし，年齢が 20 歳以上であれば
  print("飲酒 OK！でも，控えめに... \n")
else                                  # そうでなければ，
  print("飲んじゃダメ！まだ，未成年！\n")
end
-----------------------------------------------
```

演習 **6-7**

キーボード入力した得点から成績 (A, B, C, D) を判定するプログラムを擬似的な言語で書いた後で，ruby プログラムにしてみよう．

擬似的な言語によるプログラム	ruby プログラム

4　ファイルからデータを入出力する（リダイレクト）

演習 **6-8**
画面出力をファイルへ書き込む場合は次のようにする．

```
Z:\progI>ruby filename.rb > output.txt
```

>を「出力のリダイレクト」と呼ぶ．ここで，output.txt は出力ファイルの名前である．別になんでもかまわないが，拡張子はファイルの形式にあわせよう．このようにして，filename.rb を実行すると，結果は画面に出力されずに output.txt という名前のファイルに書き込まれる．

では，画面出力をするプログラムを何か作成して，画面出力をファイルに書き込もう．できたファイルを Meadow 等のエディタで開いて，出力結果を確認しよう．

演習 **6-9**
似た方法でファイルの中身をプログラムの入力にすることが簡単にできる．まずは，以下のプログラムを作って使ってみよう．

```
-----------------------------------------------
# 5 つのくだもの名を入れる配列の初期化
fruit_list = Array.new(5, "")

# くだもの名をキーボードから受け取り，配列に格納
i = 0
while i < 5
  print("くだもの名を入力してください> ")
  fruit = gets.chomp
  if fruit == "."
    break
  end

  fruit_list[i] = fruit
  i = i + 1
end

# くだもの名の表示
print("\n")
print("----------------------------\n")
i = 0
while i < fruit_list.size
  print(fruit_list[i], "\n")
  i = i + 1
end
print("----------------------------\n")
-----------------------------------------------
```

次に，以下のようなデータファイルを作成しよう．これは，くだもの名が 1 行に 1 つずつ書かれた形式である．ファイル名は適切なものを考えよう．

```
-----------------------------------------------
apple
orange
grape
peach
pear
-----------------------------------------------
```

プログラムの実行を次のようにしてみよう．なお，fruit.rb が作成したプログラム，fruit_list.txt がくだもの名のリストである．

```
Z:\progI>ruby fruit.rb < fruit_list.txt
```

結果はどうなったか?

演習 6-10

繰り返しを 100 までに変更し，こんどは次のようなファイルで試してみよう．

```
-----------------------------------------------
apple
orange
grape
peach
pear
persimmon
strawberry
banana
mango
kiwi
.
-----------------------------------------------
```

演習 **6-11**
次のプログラムを作成し，その下に示した入力ファイルを使って実行してみよう．
プログラム

```
-----------------------------------------------
fruit_price_table = Hash.new(0)

i = 0
while i < 5
  print("くだもの名を入力してください> ")
  fruit = gets.chomp
  if fruit == "."
    break
  end
  print("値段を入力してください> ")
  price = gets.chomp.to_i

  fruit_price_table[fruit] = price
  i = i + 1
end

print("\n")
print("-----------------------------\n")
fruit_price_table.each{|fruit, price|
  print(fruit, "の値段は", price, "円です．\n")
}
print("-----------------------------\n")
-----------------------------------------------
```

入力ファイル

```
-----------------------------------------------
apple
200
orange
50
grape
250
peach
280
pear
130
-----------------------------------------------
```

以上の演習は，ファイルからデータを読み込むための一番簡単な方法である．ポイントは 2 つある．

ポイント１ メソッド gets を使ってキーボードからの入力を読み込むプログラムを作る．

ポイント２ 入力のリダイレクト < に続けてファイル名を指定して，キーボード入力の代わりにファイルの中身を入力にする．

[注]: プログラム中で gets を呼び出す順番とファイルに書かれているデータの順が一致していないと正しい結果は得られない．

なお，もう少し高度な方法はプログラミング演習 II で取り扱うのでお楽しみに．

演習 **6-12**

20 個の数値の合計と平均を求めるプログラムを作ろう．ただし，実行は 20 個の数値は 1 行 1 データとしてファイルに保存してあるものを利用する．

入出力の組み合わせ

さて，最後に入力と出力の組み合わせをやってみよう．

演習 **6-13**

ファイルへの出力の基本を復習しよう．まず，以下のプログラムを実行しよう．

```
-----------------------------------------------
i = 0
while i < 10
  print("もうすぐ終わりだー！\n")
  i = i + 1
end
-----------------------------------------------
```

次に，画面出力をファイルに書き込んで確かめよう．次のようにする．

```
Z:\progI>ruby program6-13.rb > output.txt
```

演習 **6-14**

今度はファイル入力の復習．まずは以下のプログラムを作成しよう．

```
-----------------------------------------------
owari = ""

i = 0
while i < 10
  nyuryoku = gets.chomp
  owari = owari + nyuryoku
  i = i + 1
end

print(owari, "\n")
-----------------------------------------------
```

次に，前の演習で作成した output.txt をキーボード入力の代わりにしてみよう．次のようにする．

```
Z:\progI>ruby program6-14.rb < output.txt
```

演習 **6-15**
さて最後に組合せ．前の演習のプログラムを次のように実行してみよう．出来たファイル output2.txt はどうなっているか? 何が起こったか考えてみよう．

```
Z:\progI>ruby program6-15.rb < output.txt > output2.txt
```

演習 **6-16**
20 個の数値を小さい順に並べた結果をファイルに出力しよう．20 個の数値はファイルに記録してあるとする．

演習 **6-17**
次の入力ファイルを読み込んで，それぞれのキャラに強さに応じて（強さの 10%の）ボーナス点を追加して，結果をファイルに出力しよう．入力ファイルの形式はキャラ名，強さ，ボーナスの順で 3 行 1 組となっていて，出力ファイルもこの形式とする．
入力ファイル

```
------------------------------------------------
uni
100
0
kani
80
0
oni
500
0
tani
700
0
saru
130
0
kani
80
0
usu
500
0
------------------------------------------------
```