プログラミング演習I
– 第9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 9 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.

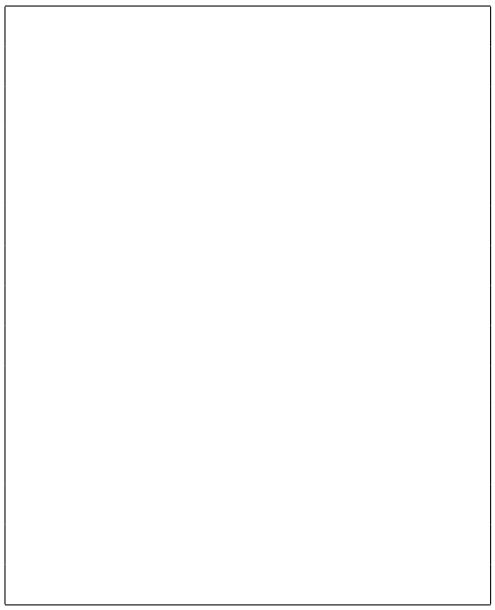
2　紙で復習

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 9-1
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 9-2

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 9-3

次のプログラムを実行した時にできるハッシュの中身を書いてみよう．キーボード入力は表の通り.

```
------------------------------------------------
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		ハッシュの中身	
neta	kakaku	ねた	価格
かっぱ	120		
うに	300		
大トロ	650		
穴子	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 9-4
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みよう．

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 9-5

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… キーボードからりんごに 80，みかんに 70，かきに 50 と入力したらあれ ???

```
------------------------------------------------
 1: kakaku_list = {"りんご" => 100, "みかん" => 50, "バナナ" => 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
 9:
10:   if kakaku_list[kudamono] > kakaku
11:     kakaku_list[kudamono] = kakaku
12:   else
13:     kakaku_list[kudamono] = kakaku
14:   end
15: end
------------------------------------------------
```

▼最初にハッシュ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	りんご	80	100 > 80 は true. kakaku_list["りんご"] に 80 が代入される	りんご	80
				みかん	
				バナナ	
1	みかん	70		りんご	
				みかん	
				バナナ	
2	かき	50		りんご	
				みかん	
				バナナ	

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 9-6
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．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．未完成／間違い部分を埋めてみよう．

```
------------------------------------------------
# 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 9-7

キーボード入力した得点から成績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

演習 9-8

要素数 6 の配列を用意し，適当な整数を入れておく．キーボードから入力した整数が配列に含まれているかどうかを判定するプログラムを擬似的な言語で設計する一例を以下に示そう．

▼擬似的な言語によるプログラム（レベル 1）

まず，問題文を分解し基本的な処理を書く．この時点ではプログラムの構文はそれほど意識しなくてよい．

```
要素数 6 の配列を用意し，適当な整数を入れておく．
キーボードから整数を入力できるようにする．

入力された整数が配列に含まれているかどうかを調べる．
判定結果を表示する．
```

▼擬似的な言語によるプログラム（レベル 2）

次に，プログラムとするのにまだ曖昧な部分「入力された整数が配列に含まれているかどうかを調べる」の手順を分解する．ここで，配列の中に「ある」と判定できるのは「あったその時」であるが,「ない」と判定できるのは「**全部調べて**なかった時」である．判定結果を表示するタイミングには注意しよう．

```
要素数 6 の配列 a を用意し，6 つの適当な整数で初期化する．
キーボードからの入力を整数として変数 kazu に代入する．

入力された整数 kazu と配列 a の要素を 1 つずつ比較する．
もし，同じだったらその旨を表示し，比較をやめる．
そうでなかったら何もしないで，比較を続ける．

全部調べて含まれていなかった場合はその旨を表示する．
```

▼擬似的な言語によるプログラム（レベル 3）

もう少しプログラム風に書けそうだ．含まれていなかったことが分かるように，flag という変数を導入する．

要素数 6 の配列 a を用意し，整数で初期化する．
キーボードからの入力を整数として変数 kazu に代入する．

変数 flag を 0 にする．（flag は含まれていれば 1，含まれていなければ 0 となるようにする）
配列 a の要素を調べるため，添字にあたる変数 i を 0 に初期化し，
i が 4 まで i を 1 ずつ増やしながら繰り返す．
　もし，入力した整数と配列の i 番目の要素が等しければ
　　ある旨を表示する．
　　含まれていたことを記憶するために flag を 1 にする．
　　比較をやめる．
　（そうでなかったら，なにもしない）

もし，flag が 0 のままなら（すなわち，全部調べて含まれていなかったということ）
　含まれていなかった旨を表示する．

▼ ruby プログラム

さて，このくらい書くと，あとは ruby のプログラムにするだけ．以下のようになるだろう．

```
-----------------------------------------------
a[8] = {1, 3, 9, 8, 7, 8}

kazu = gets.chomp.to_i

flag = 0
i = 0
while i < 5
  if kazu == a[i]
    print(kazu, "は", i, "番目にあります\n")
    flag = 1              # 調べている途中で含まれていれば flag は 1 になる
  end
  i = i + 1
end

if flag == 0              # 全部調べて含まれていなかった
  print(kazu, "はありません．\n")
end
-----------------------------------------------
```

演習 9-9

ハッシュを定義して，くだもの名と値段の組を 10 組格納し，表示するプログラムを作ってみよう．まずは，擬似的な言語で書いてみて，それを ruby プログラムとしてみよう．

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）

教科書

p.40 リダイレクトについて

演習 9-10
画面出力をファイルへ書き込む場合は次のようにするのを覚えているだろうか？(演習 2-24〜26 参照)

```
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．

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 9-11
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．

```
------------------------------------------------
fruit_list = Array.new(5, "")

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

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．

```
-----------------------------------------------
りんご
みかん
ぶどう
もも
なし
-----------------------------------------------
```

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 9-12

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．

```
-----------------------------------------------
りんご
みかん
ぶどう
もも
なし
かき
いちご
バナナ
マンゴー
キーウィー
.
-----------------------------------------------
```

演習 9-13
次のプログラムを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

```
-----------------------------------------------
りんご
200
みかん
50
ぶどう
250
もも
280
なし
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 9-14

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．

演習 9-15

県名を保存する配列 ken と栗の収穫量を保存する配列 kuri を用意し，キーボードから入力してこれらの配列にデータを全て格納した後に，表示するプログラムを作成しよう．配列の要素数は 10 とする．

次に，第 1 回レポート課題の設問 2 の県名と栗の収穫量のデータをファイルに保存して，このファイルを使ってプログラムを実行してみよう．