

プログラミング言語 第七回

担当: 篠沢 佳久
櫻井 彰人

平成23年 6月13日

本日の内容

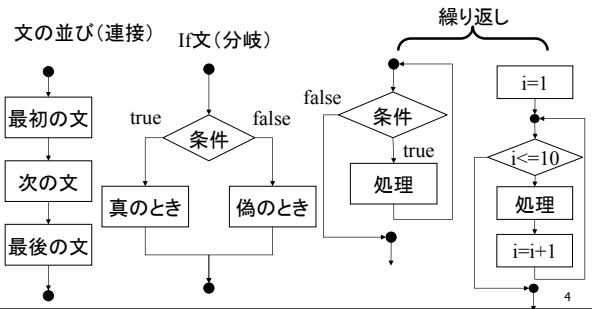
- 繰り返し(3)
- While

- 他のループ制御
- downto, upto, step, for
 - 参考ですが, 閲覧しておいて下さい

前回の復習

制御構造
繰り返し(1)

制御構造(復習)



無限の繰り返し

```
loop{  
  式  
}
```

式が永久に実行される
停止するために **break** を
用いる

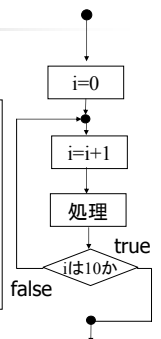
```
loop{  
  式  
  break 条件式  
}  
次の式
```

条件式を満たした場合のみ
停止する(loopブロックの
次の式を実行する)

無限の繰り返し

10回「こんにちは」を表示するプログラム

```
i = 0  
loop{  
  i = i+1  
  print( i, "回目のこんにちは¥n" )  
  break if i == 10  
}
```



回数の決まった繰り返し

```
times  
each
```

7

times①

- 同じ処理をn回繰り返したい
- n.times

```
n.times {  
  式  
}
```

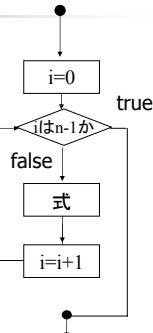
式をn回繰り返す

8

times②

```
n.times{ |i|  
  式  
}
```

n回式を繰り返す
iには自動的に0からn-1が代入される

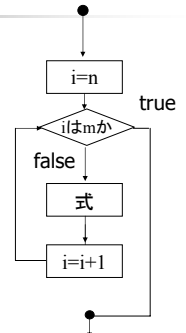


9

each

```
(n..m).times{ |i|  
  式  
}
```

m-n+1回式を繰り返す
iには自動的にnからmが代入される



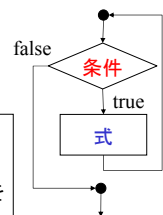
10

whileによる繰り返し

```
while 条件 do  
  式  
end
```

条件がtrueである場合は式を実行し、
false の場合は式を実行しない
すなわち条件がtrueである限りは式を
繰り返し実行する

do は省略することもできます



11

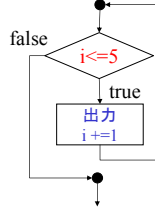
12

while による繰り返し(例①)

```
i = 1
while i <= 5 do
  puts( i.to_s + "回目です。" )
  i += 1
end
```

```
i = i+1
```

1回目です。
2回目です。
3回目です。
4回目です。
5回目です。



13

while による繰り返し(例①')

```
i = 1
while i <= 5 do
  puts( i.to_s + "回目です。" )
  i += 1
end
```

1回目です。
2回目です。
3回目です。
4回目です。
5回目です。

- ① 初期値 i=1
- ② 条件判定 -> true
- ③ puts による出力
- ④ i に1を加算(i=2)
- ⑤ 条件判定 -> true
- ⑥ putsによる出力

- ⑦ i に1を加算(i=3)
- ⑧ 条件判定 -> true
- ⑨ putsによる出力
- ⑩ i に1を加算(i=4)
- ⑪ 条件判定 -> true
- ⑫ putsによる出力

14

while による繰り返し(例①')

```
i = 1
while i <= 5 do
  puts( i.to_s + "回目です。" )
  i += 1
end
```

1回目です。
2回目です。
3回目です。
4回目です。
5回目です。

- ⑬ i に1を加算(i=5)
- ⑭ 条件判定 -> true
- ⑮ putsによる出力
- ⑯ i に1を加算(i=5)
- ⑰ 条件判定 -> false
- ⑱ 終了

15

前のページと同じプログラム(例①")

puts の書き方

```
i = 1
while i <= 5 do
  puts i.to_s + "回目です。"
  i += 1
end
```

1回目です。
2回目です。
3回目です。
4回目です。
5回目です。

```
i = 1
while i <= 5 do
  puts "#(i)回目です。"
  i += 1
end
```

16

これまで学んだ方法で書くと①

```
5.times{|i|
  puts((i+1).to_s + "回目です。")
}
```

```
(1..5).each{|i|
  puts(i.to_s + "回目です。")
}
```

17

これまで学んだ方法で書くと②

```
i = 1
loop {
  puts(i.to_s + "回目です。")
  break if i == 5
  i += 1
}
```

18

while による繰り返し(例②)

```
total = 0
i = 1
while i <= 10 do
  total += i
  i += 1
end
puts( total )
```

i=1 から開始

i<=10 の場合 do 以下を実行

i=11 となった場合、停止

iに1を加算

```
C:¥Ruby>ruby sample.rb
55
```

19

while による繰り返し(例②')

```
total = 0
i = 1
while total <= 100 do
  total += i
  i += 1
end
puts( i, " ", total )
```

i=1 から開始

total<=100 の場合 do 以下を実行

total > 100 となった場合、停止

iに1を加算

```
C:¥Ruby>ruby sample.rb
15 105
```

20

練習①

- while による繰り返し(例③~⑥)の出力結果を考えなさい
- 実行する前に結果を予想して下さい

21

while による繰り返し(例③)

どのような出力結果になるでしょうか

```
i = 10
while i != 0 do
  puts( i )
  i -= 1
end
```

i = i-1

```
i = 10
while i > 0 do
  puts( i )
  i -= 1
end
```

22

while による繰り返し(例④)

どのような出力結果になるでしょうか

```
i = 0
while i < 10 do
  puts( i * 2 )
  i += 1
end
```

```
i = 0
while i <= 10 do
  puts( i * 2 )
  i += 1
end
```

23

while による繰り返し(例⑤)

どのような出力結果になるでしょうか

```
i = 1000
while i > 0 do
  puts( i )
  i = i / 2
end
```

```
i = 2
while i < 100000 do
  puts( i )
  i = i * 2
end
```

24

while による繰り返し(例⑥)

```
i = 1
while i <= 5 do
  puts( i.to_s + "回目です." )
end
```

実行させるとどうなるでしょうか？

25

while による繰り返し(例⑥)

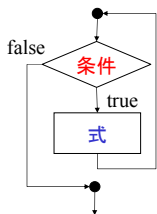
```
i = 1
while i <= 5 do
  puts( i.to_s + "回目です." )
end
```

i の値は1から変化しない

そのため条件式は常にtrue であるため、puts が永久に実行される

26

while による繰り返しにおける注意



条件がtrueである限り式は実行され続ける。そのため、条件がfalseとなり終了するように条件を設定しなければならない

27

times(each) と while の違い

- 同じである。ただし、コンセプトには結構な違いがある
 - times, each: 各回には、制御変数の値の違いしかない
 - while, loop: 各回には、制御変数以外の変数で違いがある。または、制御変数を自分で作ってあげないといけない

28

times(each)と while の違い (続)

- While, loop を使うのは
 - ファイルの読み込み(後述)のように、読んでみないとわからない場合
 - 次のテーマ
 - むずかし〜い計算なので、計算してみないとわからない場合
 - Collatz-角谷の予想
 - 素数を求める
 - 次回でてくる配列を用いる計算の中にある
 - 前の行為の結果が、影響を及ぼす場合

29

例: times と while の違い①

"繰り返し回数"が分っている場合

```
n = 10
n.times{ |i|
  puts( i )
}
```



```
i=0
while i<10 do
  puts( i )
  i = i + 1
end
```

制御変数を自分で指定しなければならない

30

例: times と while の違い②

"繰り返し回数"が分っている場合

```
n = 10
total = 0
n.times{ |i|
  total += i
}
puts( total )
```



制御変数

```
i=0
total = 0
while i < 10 do
  total += i
  i = i + 1
end
puts( total )
```

31

例: times と while の違い③

```
i = 1000
while i > 0 do
  puts( i )
  i = i / 2
end
```

```
i = 2
while i < 100000 do
  puts( i )
  i = i * 2
end
```

"停止条件"が分っているが"繰り返し回数"は分らない
→ 変数 i の値はループ内の処理によって変わっていく
→ times では書きづらい

32

例: times と while の違い④

偶数を10個求めるプログラム

```
i = 1
count = 0
while count != 10 do
  if i % 2 == 0 then
    print( i, "\n" )
    count += 1
  end
  i += 1
end
```

```
C:¥Ruby>ruby sample.rb
2
4
6
8
10
12
14
16
18
20
```

停止条件は「10個偶数が求まった時」
→実際に何回ループを回せばよいかは分からない

33

例: times と while の違い⑤

偶数を10個求めるプログラム

```
count = 0
while count != 10 do
  x = rand( 100 )
  if x % 2 == 0 then
    print( x, "\n" )
    count += 1
  end
end
rand(n)
0以上n未満の整数を生成
```

```
C:¥Ruby>ruby sample.rb
22
60
34
38
88
30
56
62
94
34
```

停止条件は「10個偶数が求まった時」
→実際に何回ループを回せばよいかは分からない

34

例: times と while の違い⑥

"繰り返し回数"は予め分からない

```
s = 0.0
n = 10
n.times{ |i|
  r = rand()
  s += r
}
a = s/n
puts( "平均= #{a}" )
```

"繰り返し回数"が予めわかっている

```
s = 0.0
n = 0
while s < 10.0 do
  r = rand()
  s += r
  n += 1
end
rand()
0以上1未満の乱数を生成
puts( "#(n) 回目で10を越えた" )
```

"停止条件"が予めわかっている

"継続条件"が予めわかっている

35

擬似乱数の復習

■ 平均、不偏分散、標準偏差を求めてみよう

```
s = 0.0
s2 = 0.0
n = 10
n.times{ |i|
  r = rand()
  s += r
  s2 += r*r
  puts( "#(r) は #(i) 番目の乱数です." )
}
a = s/n
v = (s2 - a*a*n)/(n-1)
sd = Math.sqrt( v )
puts( "平均= #{a}, 分散= #{v}, 標準偏差= #{sd}" )
```

36

break文①

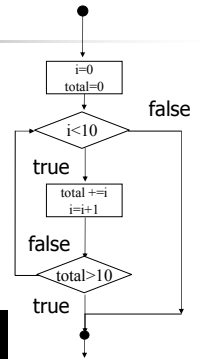
- 繰り返し(ループ)の中で使用
- そのbreakが所属するループを1つ抜ける

37

break文②

```
i = 0
total = 0
while i < 10 do
  total += i
  i += 1
  break if total > 10
end
print( i , " " , total )
```

```
C:\Ruby>ruby sample.rb
6 15
```



38

break文③

true となっているため、無限に実行される

```
i=0
loop{
  puts( i )
  i=i+1
  break if i > 10
}
```

```
i=0
while true do
  puts( i )
  i=i+1
  break if i > 10
end
```

条件式をtrueとすることによって無限ループとなる

39

無限ループ

true となっているため、無限に実行される

```
while true do
  puts( "こんにちは" )
end
```

Ctrlを押しながらcで停止

```
C:\Ruby>ruby sample.rb
こんにちは
こんにちは
こんにちは
こんにちは
こんにちは
こんにちは
こんにちは
こんにちは
こんにちは
こんにちは
こんにちは
```

40

break文④

```
i = 1000
while i > 0 do
  puts( i )
  i = i / 2
end
```

二つのプログラムとも同じ動作をします

```
i = 1000
while true do
  puts( i )
  i = i / 2
  break if i <= 0
end
```

```
C:\Ruby>ruby sample.rb
1000
500
250
125
62
31
15
7
3
1
```

41

break文④

```
i = 1000
while i > 0 do
  puts( i )
  i = i / 2
end
```

ループの停止条件
→ i > 0 を満たさない時

```
i = 1000
while true do
  puts( i )
  i = i / 2
  break if i <= 0
end
```

ループの停止条件
→ なし(無限ループ)

break文によってループから抜ける

42

break文⑤

```
10.times{ |i|
  puts( i )
  break if i > 5
}
```

```
(10..100).each{ |i|
  puts( i )
  break if i > 20
}
```

times, eachにおいてもbreakでループを抜け出すことができる

43

next

- nextはもっとも内側のループの次の繰り返しにジャンプします。
- while であれば、「継続条件」の判定の直前が再開場所です。

44

プログラム例 (each と next)

```
(0..5).each { |i|
  if ( i==1 || i==4 ) then
    next
  end
  puts( "iは #{i}" )
}
```

「または」です

出力結果

```
iは 0
iは 2
iは 3
iは 5
```

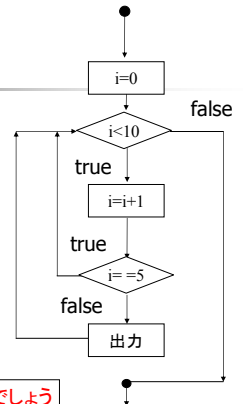
i が1または4の場合
→ next によってループの先頭に戻る
→ puts文は実行されない

45

next②

```
i=0
while i < 10 do
  i += 1
  if i == 5 then
    next
  end
  puts( i )
end
```

出力はどうなるでしょう



46

next②

```
i=0
while i < 10 do
  i += 1
  if i == 5 then
    next
  end
  puts( i )
end
```

i=5の場合、puts文は実行されない

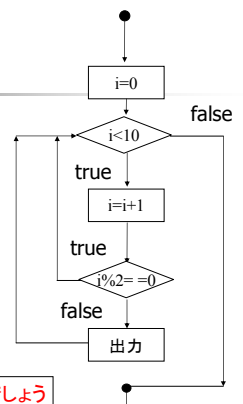
```
C:\Ruby>ruby sample.rb
1
2
3
4
6
7
8
9
10
```

47

next③

```
i=0
while i < 10 do
  i += 1
  if i % 2 == 0 then
    next
  end
  puts( i )
end
```

出力はどうなるでしょう



48

next③

```
i=0
while i < 10 do
  i += 1
  if i % 2 == 0 then
    next
  end
  puts(i)
end
```

いが偶数の場合、puts文は実行されない

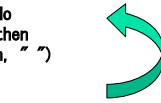
```
C:¥Ruby>ruby sample.rb
1
3
5
7
9
```

49

プログラム例(break,next)

```
n = 0
while n >= 0 do
  if n <= 10 then
    print( n, " ")
    n += 1
    next
  else
    print( "変数 n は10を越えました。" )
    break
  end
end
```

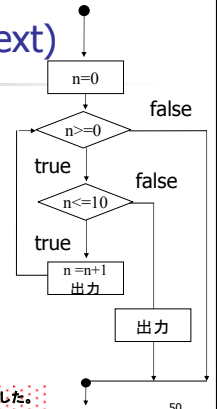
ループの先頭であるwhileに戻る



ループwhileを抜ける



0 1 2 3 4 5 6 7 8 9 10 変数 n は10を越えました。



50

繰り返しのまとめ

times, each
loop, while

51

繰り返しのまとめ①

- 繰り返し回数分かっている場合
 - times, each
- 繰り返し回数分からない場合
 - loop, while
- ただし、同じ動作をするにしても、いろいろな書き方があります

52

繰り返しのまとめ②

同じ動作をするプログラム

```
10.times{ |i|
  print( i, "¥n" )
}
```

```
(0..9).each{ |i|
  print( i, "¥n" )
}
```

C:¥Ruby>ruby sample.rb

```
0
1
2
3
4
5
6
7
8
9
```

```
i = 0
loop{
  break if i == 10
  print( i, "¥n" )
  i += 1
}
```

```
i = 0
while i < 10 do
  print( i, "¥n" )
  i += 1
end
```

53

繰り返しのまとめ③

同じ動作をするプログラム

```
i = 2
while i < 100 do
  print( i, "¥n" )
  i = i*2
end
```

```
i = 1
while true do
  i = i*2
  break if i > 100
  print( i, "¥n" )
end
```

```
i = 1
loop{
  i = i*2
  break if i > 100
  print( i, "¥n" )
}
```

C:¥Ruby>ruby sample.rb

```
2
4
8
16
32
64
```

54

他のループ制御

downto, upto, step, for

55

他のループ制御

- Ruby では他に、until, for などがある。
 - 拘る人へ: 正確には
 - Ruby の繰返制御構造は、while, until, for である
 - times, upto, downto, step は Integer のメソッド
 - each は Array 等の、Enumerable をインクルードするクラスのメソッド
 - ループの中断には、break以外、next, redo, retry がある

56

downto, upto, and step

- 例で学ぼう

7.downto(3) { |i| print(i, " ") } 7 6 5 4 3

3.upto(7) { |i| print(i, " ") } 3 4 5 6 7

2.step(12,3) { |i| print(i, " ") } 2 5 8 11

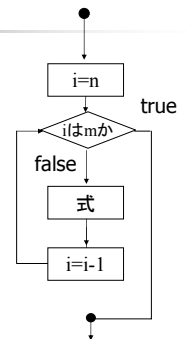
12.step(2,-3) { |i| print(i, " ") } 12 9 6 3

57

downto①

```
n.downto(m){ |i|
  式
}
```

n-m+1回式を繰り返す (n>m)
iには自動的にnからmが代入される



58

```
10.downto(0){ |x|
  printf( "x=%2d x^2=%3d\n", x, x**2 )
}
```

%2d
整数を二桁で表示する

x は10から0まで
1刻みで変化

```
C:\ruby>ruby sample.rb
x=10 x^2=100
x= 9 x^2= 81
x= 8 x^2= 64
x= 7 x^2= 49
x= 6 x^2= 36
x= 5 x^2= 25
x= 4 x^2= 16
x= 3 x^2=  9
x= 2 x^2=  4
x= 1 x^2=  1
x= 0 x^2=  0
```

59

```
0.downto(-10){ |x|
  printf( "x=%3d x^2=%6d\n", x, x**2 )
}
```

x は0から-10まで
1刻みで変化

```
C:\ruby>ruby sample.rb
x= 0 x^2=  0
x=-1 x^2= -1
x=-2 x^2= -8
x=-3 x^2= -27
x=-4 x^2= -64
x=-5 x^2= -125
x=-6 x^2= -216
x=-7 x^2= -343
x=-8 x^2= -512
x=-9 x^2= -729
x=-10 x^2= -1000
```

60

downto②

while で書いた場合

```
total = 0
10.downto(0){ |i|
  total += i
  break if total > 20
}
```

```
i = 10
total = 0
while i >= 0 do
  total += i
  break if total > 20
  i -= 1
end
```

61

downto③

downto と while を利用して書き直してみてください

```
total = 1
1.upto(10){ |x|
  total *= x
}
print( total )
```

```
total=
1×2×3×...×10
```

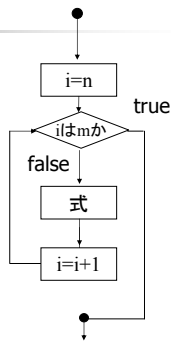
```
C:¥Ruby>ruby sample.rb
3628800
```

62

upto①

```
n.upto(m){ |i|
  式
}
```

m-n+1回式を繰り返す(n<m)
iには自動的にnからmが代入される



63

upto②

```
n.upto(m){ |i|
  式
}
```

```
(n..m).each { |i|
  式
}
```

基本的には each と同じ

64

upto③

```
0.upto(10){ |x|
  print( "x= ", x , "¥n" )
}
```

each で書いた場合

```
(0..10).each { |x|
  print( "x= ", x , "¥n" )
}
```

```
C:¥Ruby>ruby sample.rb
x= 0
x= 1
x= 2
x= 3
x= 4
x= 5
x= 6
x= 7
x= 8
x= 9
x= 10
```

65

```
total = 0
0.upto(10){ |x|
  total += x
}
print( total )
```

```
total = 0
10.downto(0){ |x|
  total += x
}
print( total )
```

```
total = 0
11.times{ |x|
  total += x
}
print( total )
```

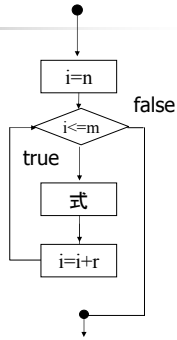
```
total = 0 ; i=0
while i <= 10 do
  total += x
  i += 1
}
print( total )
```

66

step①

```
n.step(m,r){ |i|  
  式  
}
```

nからmまで刻み幅はrで繰り返す
iには n, n+r, n+2*r, ... が入る



67

step②

```
0.step(10,2){ |x|  
  print( x , "¥n" )  
}
```

```
C:¥ruby>ruby sample.rb  
0  
2  
4  
6  
8  
10
```

```
1.step(10,2){ |x|  
  print( x , "¥n" )  
}
```

```
C:¥ruby>ruby sample.rb  
1  
3  
5  
7  
9
```

68

step③

```
10.step(0,-2){ |x|  
  print( x , "¥n" )  
}
```

```
C:¥ruby>ruby sample.rb  
10  
8  
6  
4  
2  
0
```

```
9.step(1,-2){ |x|  
  print( x , "¥n" )  
}
```

```
C:¥ruby>ruby sample.rb  
9  
7  
5  
3  
1
```

69

step④

```
total = 0  
1.step(50,2){ |i|  
  total += i  
}  
print( total )
```

```
C:¥Ruby>ruby sample.rb  
625
```

whileを用いた場合

```
total = 0  
x = 1  
while x <= 50 do  
  total += x  
  x += 2  
end  
print( total )
```

70

step⑤

```
total = 0  
49.step(1,-2){ |i|  
  total += i  
}  
print( total )
```

```
C:¥Ruby>ruby sample.rb  
625
```

whileを用いた場合

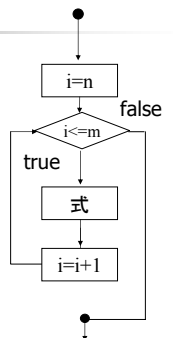
```
total = 0  
x = 49  
while x >= 1 do  
  total += x  
  x -= 2  
end  
print( total )
```

71

forループ①

```
for i in n..m do  
  式  
end
```

```
(n..m).each{ |i|  
  式  
}
```



72

forループ②

```
for x in 1..10 do
  puts( 2**x )
end
```

x=1,2,...,9,10

```
C:¥ruby>ruby sample.rb
2
4
8
16
32
64
128
256
512
1024
```

73

forループ③

```
for x in -10..10 do
  puts( 2**x )
end
```

x=-10,-9,...,9,10

```
total = 0
for x in 1..10 do
  total += x
end
print( total )
```

x=1,2,...,9,10

74

forループ④

upto と while を利用して書き直してみてください

```
for x in 0..100 do
  a = x / 100.0 * Math::PI
  printf( "%f %f\n", a , Math.sin( a ) )
end
```

%f
浮動点小数表示

```
C:¥Ruby>ruby sample.rb
0.000000 0.000000
0.031416 0.031411
0.062832 0.062791
0.094248 0.094108
0.125664 0.125333
0.157080 0.156434
0.188496 0.187381
```

75

練習問題

練習問題

- 練習問題①から③を行ないなさい。
- 簡単な人は、web 上から自習問題をダウンロードして試してみてください。(回答とは別に)自分で考えて下さい。
- プログラムと実行結果をワープロに貼り付けて、keio.jp から提出して下さい。

77

練習①

- ① while を用いて1以上10以下の整数の積 ($1 \times 2 \times 3 \times \dots \times 10$)を求めるプログラムを書きなさい
- ② while を用いて n^2 ($n=10,11,\dots,20$)を求めプログラムを書きなさい

78

