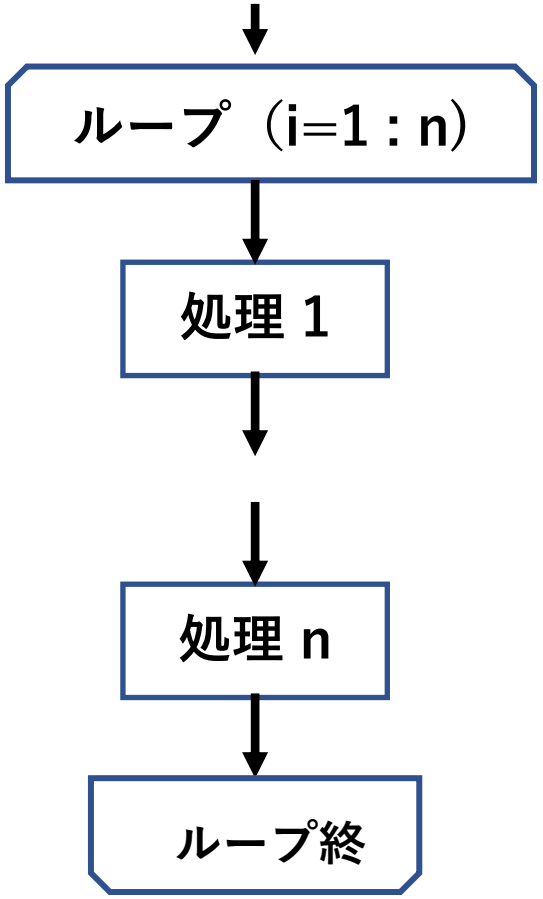
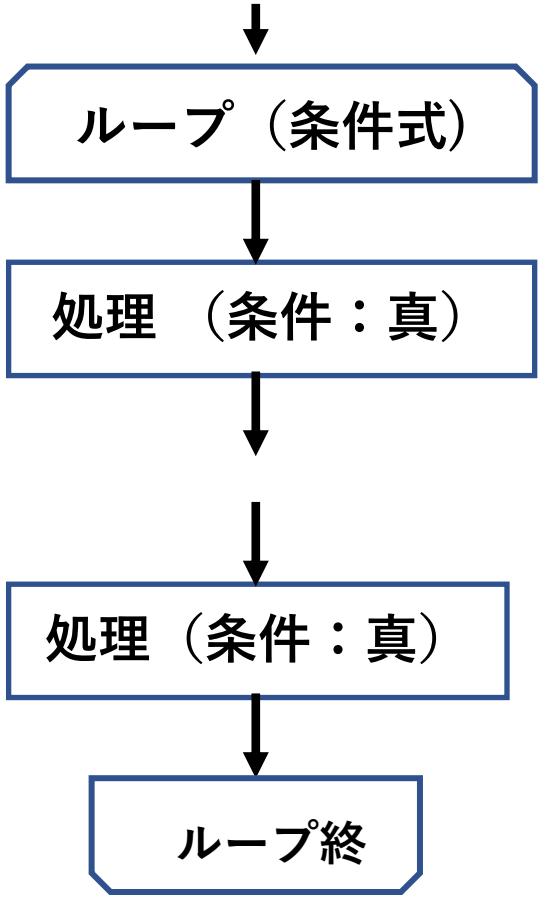
Rプログラミング（繰り返し）

繰り返し：フローチャート

for文



While文



For 文


```
# 自然数を入力し,1からnまでの平方和を出力する
example8_01 <- function(n){
    x <- 0
    for (i in 1:n){          #  変数iを1からn まで繰り返す
        x <- x + i^2         #  順次２乗してxに代入
    }
   return(x)                 #  処理を終了し結果を表示
}
```

```
for （変数 in リスト）{

  繰り返す処理
    ・・・
  繰り返す処理

}
```

```
> example8_01(1)
[1] 1
> example8_01(10)
[1] 385
> example8_01(100)
[1] 338350
> example8_01(1000)
[1] 333833500
```

For 文


```
# 自然数を入力し,1からnまでの平方和の途中経過も出力する

example8_02 <- function(n){
    x <- 0
    for (i in 1:n){          #  変数iを1からn まで繰り返す
        x <- x + i^2         #  順次２乗してxに代入
        print(x)             #  途中の結果を表示
    }
}
```

```
for (変数 in リスト){

  繰り返す処理
    ・・・
  繰り返す処理

}
```

```
> example8_02(4)
[1] 1
[1] 5
[1] 14
[1] 30
```

演習

問題１
自然数 n を代入して、1からｎまでの平方数（２乗）を出力する関数を作成しなさい。

問題２
collatz関数をs回繰り返す関数 collatzlist を作成しなさい。

問題１（ヒント）

問題１
自然数 n を代入して、1から n までの平方数（２乗）を出力する関数を作成しなさい。

```
sq <－function(n){

        for ( [          ] ) {

            x <-  [          ]

            print(x)
        }
}
```

問題１(解答例)

自然数 n を代入して、1から n までの平方数（２乗）を出力する関数を作成しなさい。

```
sq <−function(n){
        for ( i  in 1 : n ) {
                x <- i ^ 2
                print(x)
        }
 }
```

```
> sq(4)
[1] 1
[1] 4
[1] 9
[1] 16
```

注：print(x)をreturn(x)に変えるとどうなるだろう？

For 文

問題２（ヒント）　collatz関数をs回繰り返す関数 collatzlist を作成しなさい。

```
collatzlist <- function(n,s){

    for ( [          ]){                    # 変数をiにして1からsまで繰り返す

        if(n %% 2 ==0 ){
            n <- n/2
        }
          else{
            n <- 3*n+1
          }
      print(n)
    }
}
```

問題２(解答例)

collatz関数をs回繰り返す関数 collatzlist を作成しなさい。

```
collatzlist <- function(n,s){

      for ( i  in 1 : s){

            if(n %% 2 ==0 ){
                  n <- n/2
            }
              else{
                  n <- 3*n+1
              }
        print(n)
      }
}
```

実行

```
> collatzlist(4,3)
[1] 2
[1] 1
[1] 4
```

問題２(発展)　　出力をベクトル形式にする

```
collatzlist <- function(n,s){
    x <- n                                  # ベクトル x に n を代入する
        for ( i  in 1 : s){
          if(n %% 2 ==0 ){
                n <- n/2
        }
          else{
                n <- 3*n+1
            }
     x <- c(x, n)                           # ベクトル x に順次，結果を追加する
     }
  print(x)
}
```

結果

```
> collatzlist(3,10)
 [1]  3 10  5 16  8  4  2  1  4  2  1
```

問題２(発展)　外部でcollatz2を定義してcollatzlist2を定義すると，すっきりする

```
collatz2 <- function(n){
        if(n %% 2 ==0 ){
           x <- n/2
        }
         else{
           x <- 3*n+1
        }
}
```

出力なし関数

```
collatzlist2 <- function(n,s){
       x <- n;
             for ( i  in 1 : s){
                n <- collatz2(n)
                x <- c(x, n)
             }
       return( x )
}
```

while 文

自然数を入力し,1からnまでの平方和を出力する

このプログラムはexample8_01 と同じ

```
example8_03 <- function(n){
  x <- 0                    # xに0を代入

  i <- 1                    # iに1を代入

   while (i <= n){          # 変数iがnになるまで繰り返す

     x <- x + i^2           # 1^2から順にi^2を代入する

     i <- i + 1             # iを1だけ順次増やす

   }
   return(x)
}
```

```
while （条件式 ）{

  条件式が真：繰り返す処理
    ・・・
  条件式が真：繰り返す処理

}
```

While 文

問題２(発展)　collatzlist関数で1が出力されたら処理を終了

```
collatzlist3 <- function(n){
      x <- n
           while (      ){                    # nが１になるまで繰り返す

             n <- collatz2(n)

             x <- c(x, n)
           }

      print( x )
}
```

問題２(発展）解答例　collatzlist関数で1が出力されたら処理を終了

```
collatzlist3 <- function(n){
      x <- n;
            while ( n > 1){

               n <- collatz2(n)

               x <- c(x, n)
             }

       return( x )
}
```

実行

```
> collatzlist3(3)
[1]  3 10  5 16  8  4  2  1
```

問題3(発展)　1から100までで一番長い経路をだどって１になる自然数を求めよう

(1) 1から100までのcollatz数で1に至るまでのリスト数を定義するcollatzlist4を作成しなさい.

(2) 一番長いcollatz数の経路をグラフ化しなさい.

While 文

問題3(発展)　1から100までで一番長い経路をだどって１になる自然数を求めよう（解答例）

```
collatzlist4 <- function(n){
  x <- NULL

      for ( i  in 1:n){

        y <- length(collatzlist3(i))

        x <- c(x,y)

      }

    return(x)
}

#実行
collatzlist4(100)
```

For 文（発展）

 # 3の倍数ならX, 3の倍数でなく５の倍数ならY,そうでないならその数値を出力

```
> example8_02 <- function(n){
+   x <- as.character(NULL)                    # xは文字型としてなにも入っていないことにする
+   y <- 0
+   for (i in 1:n){
+     if (i %% 3 == 0)       y <- "X"
+     else if (i %% 5 == 0) y <- "Y"
+     else                        y <- i
+     x <- c(x,y)                                # xはベクトル型として、リストとして数値を追加する
+   }
+   return(x)
+ }
```

```
> example8_02(50)
 [1] "1"  "2"  "X"  "4"  "Y"  "X"  "7"  "8"  "X"  "Y"  "11"
[12] "X"  "13" "14" "X"  "16" "17" "X"  "19" "Y"  "X"  "22"
[23] "23" "X"  "Y"  "26" "X"  "28" "29" "X"  "31" "32" "X"
[34] "34" "Y"  "X"  "37" "38" "X"  "Y"  "41" "X"  "43" "44"
[45] "X"  "46" "47" "X"  "49" "Y"
```

while 文（文中にif を入れる）

 # 0~4までの整数値を入力し，プログラムが選んだ整数値（乱数）と一致するかどうか。

```
> example8_04 <- function(){
+   i <- 0
+   while (i = = 0){
+     x <- readline("0から4までの整数値を選んでください：")
+     y <- floor(runif(1)*5)                    # runif(n):0~1までの乱数をn個出力
+     cat("私が選んだ数値は",y,"です。")          # floor:入力値を越えない最大の整数値
+     if (x = = y){
+       cat("正解！¥n")
+       i <- 1
+     }
+     else{
+       cat("残念！¥n")
+     }
+   }
+   cat("お疲れさまでした。¥n")
+ }
```

```
> example8_04()
0から4までの数値を選んでください：2          # 2を入力
私が選んだ数値は 1 です。残念！
0から4までの数値を選んでください：4          # 4を入力
私が選んだ数値は 4 です。正解！
お疲れさまでした。
>
```

繰り返し（発展）


```
> example9_01 <- function(vec){
      tmp <- vec[2]
      vec[2] <- vec[1]
      vec[1] <- tmp
      return(vec)
}
```

 # 入力した1番目と2番目を入れ替えるプログラム

例：vec=c(4,2,5,1,3)の時, length(vec)=5

4,2,5,1,3 →2,4,5,1,3

```
> example9_02 <- function(vec){
   n <- length(vec)
   for (j in 1:(n-1)){
      if (vec[j] > vec[j+1]){
         tmp <- vec[j+1]
         vec[j+1] <- vec[j]
         vec[j] <-tmp
       }
    print(vec)
  }
}
```

 # 隣り合う数値の昇順に入れ替えるプログラム

例：vec=c(4,2,5,1,3)の時, length(vec)=5

j=1：4,2,5,1,3 →2,4,5,1,3
j=2：2,4,5,1,3 →2,4,5,1,3
j=3：2,4,5,1,3→2,4,1,5,3
j=4：2,4,1,5,3→2,4,1,3,5

繰り返し（発展）


```
> example9_03 <- function(vec){
    n <- length(vec)
    for (i in 1:(n-1)){
      for (j in 1:(n-i)){
        if (vec[j] > vec[j+1]){
          tmp <- vec[j+1]
          vec[j+1] <- vec[j]
          vec[j] <-tmp
        }
      }
    }
    return(vec)
  }
```

 # 入力した数値を昇順に並べ替えるプログラム

 # バブルソート法という.

例：vec=c(4,2,5,1,3)の時, length(vec)=5

i=1：4,2,5,1,3 →2,4,5,1,3→2,4,5,1,3 →2,4,1,5,3→2,4,1,3,**5**
i=2：2,4,1,3,5 →2,4,1,3,5→2,1,4,3,5 →2,1,3,**4,5**
i=3：2,1,3,4,5→1,2,3,4,5→1,2,**3,4,5**
i=4：1,2,3,4,5→**1,2,3,4,5** (完成)

```
> x <- floor(runif(10)*100)
> x
 [1] 87 47  9 51 31 51 86 74 10 53
> example9_01(x)
[1]  9 10 31 47 51 51 53 74 86 87
```

#　10個の整数を入力（乱数）