Rプログラミング（条件分岐）

条件分岐

前回の復習

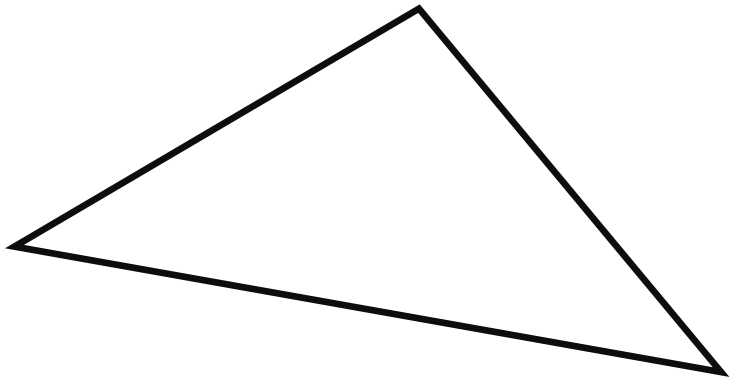
解答例：

```
heron01 <- function(a,b,c){
        s = (a+b+c)/2
        v = sqrt(s*(s-a)*(s-b)*(s-c))
  return(v)
}
```

ヘロンの公式：

$$S = \sqrt{s(s-a)(s-b)(s-c)}$$

ただし$(2s = a + b + c)$



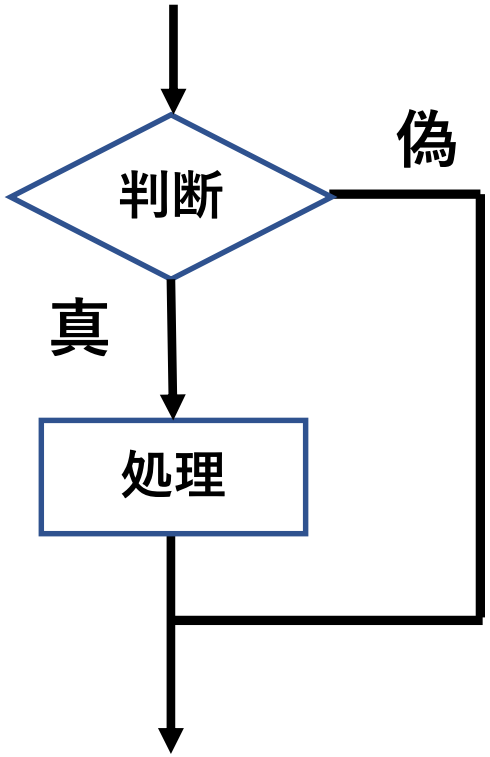
1：heron01(2,2,2)を実行してみよう。（正三角形）
2：heron01(1,2,5)を実行してみよう。（エラーメッセージ）？？

場合分け（絶対値）

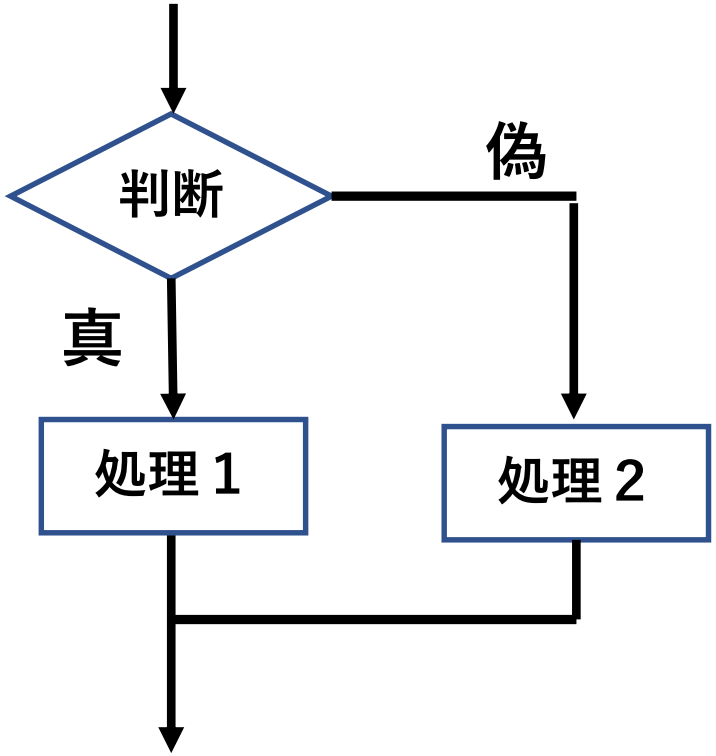
$$|x| = \begin{cases} x & (\text{if } x \geqq 0) \\ -x & (\text{if } x < 0) \end{cases}$$

条件分岐：フローチャート

if 文



if 文 else文



条件分岐（if 文：真の場合の処理）


```
example7_03<- function(x){
      if (x < 0){
          x <- -x
       }
     return(x)
  }
```

if（条件式）{

“条件式が真のとき実行される処理”

}

```
> example7_03(2)
[1] 2

> example7_03(-3)
[1] 3

> example7_03(0)
[1] 0
```

条件分岐（if文とelse文：真と偽の場合の処理）


```
example7_04    <-   function(n){
      if (n %% 2 == 0){          # nが2で割り切れる
          x  <-  n / 2
   }
      else{
          x  <-  3 * n + 1
   }
    return(x)
  }
```

```
if (条件式){

    条件式が真のときに実行
}

 else{

      条件式が偽のときに実行
}
```

Collatz 関数

```
> example7_04(13)
[1] 40

> example7_04(40)
[1] 20
```

演習

問１：
３つの正の数値(a, b, c)を小さい順に入力して、その長さを三辺とする三角形の面積を求める関数を作成しなさい。ただし、３つの数値から三角形が出来ない場合は999を出力する。

問２：（発展）
３つの正の数値(a, b, c)をランダムに入力して、その長さを三辺とする三角形の面積を求める関数を作成しなさい。ただし、３つの数値から三角形が出来ない場合は999を出力する。

演習の解答例

問１：

```
heron02 <- function(a,b,c){
        if (c < a+b){
         s <- (a+b+c)/2
         v <- sqrt(s*(s-a)*(s-b)*(s-c))
       }
         else{
         v <- 999
       }
return(v)
}
```

演習の解答例

問２

```
heron03 <- function(a,b,c){
          if (c < a+b && b < c+a && a < b+c ){
              s <- (a+b+c)/2
              v <- sqrt(s*(s-a)*(s-b)*(s-c))
           }
           else{
             v <- 999
           }
         return(v)
}
```

条件分岐（if文とelse文の組み合わせ）


```
example7_05 <- function(x){
   if (x > 0){
     cat("positive number¥n")
   }
   else if (x < 0){
     cat("negative number¥n")
   }
   else{
     cat("zero¥n")
   }
 }

> example7_05(2)
positive number
> example7_05(-3)
negative number
> example7_05(0)
zero
```

条件分岐　（if を並べる）


```
example7_06 <- function(n){
  if (n %% 2 = = 0){
    cat(n,"は2の倍数¥n")
  }
  if (n %% 3 = = 0){
    cat(n,"は3の倍数¥n")
  }
  if (n %% 5 = = 0){
    cat(n,"は5の倍数¥n")
  }
}
```

```
> example7_06(8)
8 は2の倍数
> example7_06(75)
75 は3の倍数
75 は5の倍数
> example7_06(77)
```

77は2,3,5とは互いに素であるから何も表示されない

再帰呼出（発展）


```
example7_08 <- function(n){          # 自然数を入力し，1からその数までの平方の和を求める．
  if (n = = 1){
    x <- 1
  }
  else{
    x <- n^2 + example7_08(n-1)      # 自分自身の定義の中に自分を呼ぶ（再帰的定義）．
  }                                  # 数列の漸化式も再帰的定義
  return(x)
}
```

数列の漸化式での表現

$a_1 = 1,$

$a_n = n^2 + a_{n-1}$

```
> example7_08(1)
[1] 1
> example7_08(10)
[1] 385
> example7_08(100)
[1] 338350
> example7_08(1000)                  # この方法には限度があることが分かる例
エラー: 評価があまりに深く入れ子になっています。無限の再帰か options(expressions=)？
```