基礎プログラミングII

第4回　メソッド定義と効果的活用（再帰的アルゴリズム）

メディア情報コース
平居　悠（ひらい　ゆたか）

到達目標

プログラミングを用いた実践的なデータ処理と情報システムの構築

- 定式化された処理を**関数の形**で記述し利用することができるようになる
- 再帰などの**アルゴリズム**を理解し問題に適用できるようになる
- 基礎的な**CGI**の仕組みの理解を通して**Webインターフェース**を設計できるようになる
- 多様な**社会事象への適用**を設計できるようになる
- 現実社会の課題に対応する**情報システム**を設計・作成できるようになる
- **生成AI**を学習の道具として利用できるようになる

前回

第１回　生成AIを効果的に利用した学習法
第２回　メソッド定義と効果的活用（関数的処理）
第３回　メソッド定義と効果的活用（データ集合処理）
第４回　メソッド定義と効果的活用（再起的アルゴリズム）
第５回　変数のスコープ・クラス設計（変数のスコープ規則）
第６回　変数のスコープ・クラス設計（クラス定義）
第７回　CGIと情報システム（CGIの基本要素）
第８回　CGIと情報システム（持続的な値のやり取り）
第９回　CGIと情報システム（実践的処理）
第１０回　チーム作品作成
第１１回　チーム作品の構築・trr試験
第１２回　チーム作品発表・評価
第１３回　合同成果発表会
第１４回　期末試験

前回の目標

配列とハッシュを用いてデータ処理できるようになる。

前回学んだこと

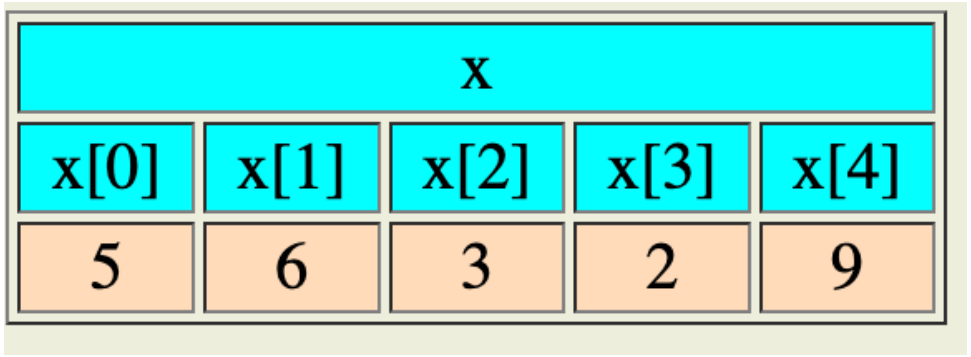
1. 配列処理メソッド
2. 様々な条件での並べ換え
3. データの格納と検索・絞り込み

配列と文字列

配列と文字列はどちらも要素の並びと捉えられる。

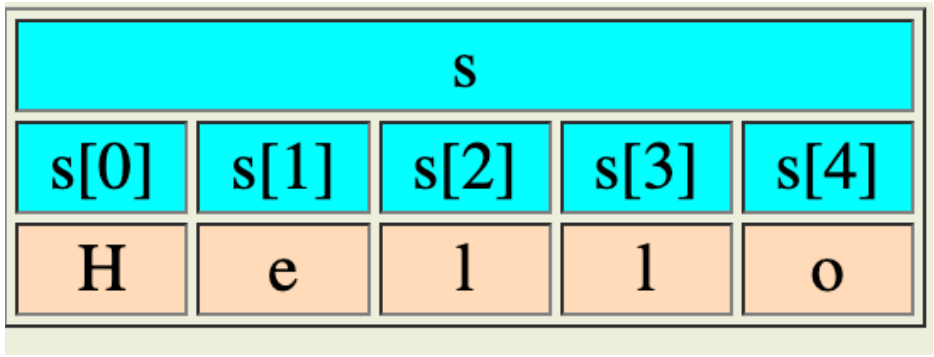
配列

x = [5, 6, 3, 2, 9]の格納イメージ



文字列

s = “Hello”の格納イメージ



配列処理メソッド

1. 集合的な処理を行うメソッド
2. 配列用のメソッド
3. 文字列固有のメソッド

each: 配列/ハッシュの各要素に対するブロックの評価

配列/ハッシュの各要素に対して後続するブロックを繰り返し評価する。

配列

要素を受ける1つのブロック変数で繰り返す。

```
x = [3, 4, 1, 2, 5]
x.each do |i|
  #i= 3, 4, 1, 2, 5で5回繰り返す
end
```

forループで書き換えられる。

```
for i in x
  #i= 3, 4, 1, 2, 5で5回繰り返す
end
```

ハッシュ

keyとvalueを受ける2つのブロック変数で繰り返す。

```
h = {“りんご” => 150, “みかん” => 30}
h.each do |item, price|
  #「item = “りんご”, price = 150」と「item = “みかん”, price = 30」の組で2回繰り返す
end
```

forループで書き換えられる。

```
for item, price in h
  #「item = “りんご”, price = 150」と「item = “みかん”, price = 30」の組で2回繰り返す
end
```

[N]: N番目の取り出し

配列/文字列の添字Nの位置の値を取り出す。

配列

```
x = [3, 4, 1, 2, 5]
x[0]
⇒3
x[-2]
⇒2
x[99]
⇒nil
```

文字列

```
s = “hoge”
s[0]
⇒“h”
s[-2]
⇒“g”
s[99]
⇒nil
```

reverse, reverse!: 逆順にする

配列/文字列の中身を全て逆順に並べたものを返す。

配列

```
x = [3, 1, 2, 4]
x.reverse
⇒ [4, 2, 1, 3]
```

xの値は変わらない。

```
x => [3, 1, 2, 4]
```

reverse!は破壊的に変更する。

```
x.reverse!
⇒ [4, 2, 1, 3]
x => [4, 2, 1, 3]
```

文字列

```
s = “Hello”
s.reverse
⇒ “olleH”
```

xの値は変わらない。

```
s => “Hello”
```

reverse!は破壊的に変更する。

```
s.reverse!
⇒ “olleH”
s => “olleH”
```

sort {ブロック}: 配列の内容のソート

配列の内容を並べ換える。ブロックを指定しない場合は各要素を演算子<=>で比較して小さい順（昇順）に並べ換える。

```
z = x.sort { |x, y| x <=> y}
```

大きい順（降順）に並べ換えたい場合

```
z = x.sort { |x, y| y <=> x}
```

sort_by{ブロック}: 配列の内容の効率的ソート

ソート基準とする値を取り出す式のみ指定。ハッシュのvalueの大小関係でソートしたい時に有用

商品名vs.単価リストを持つハッシュの商品名を単価の小さい（安い）順に並べ換える場合：

```
item = {
  “アルカリ異音水” => 150,
  “おでんつゆ” => 50,
  “マグロの煮こごり” => 500
}

item.keys.sort_by{ |x| item[x]}
```

chomp(rs), chomp!(rs): 文字列の末尾削除 (条件付)

文字列末尾から調べてrsという文字列があればそれを取り除く。rsを指定しない場合は改行文字が取り除かれる。

```
x = “abc\n”
x.chomp
⇒ “abc”
```

gsub(pattern){replace}, gsub!(pattern){re[lace}: 文字列の置き換え

元の文字列のうちpatternにマッチするものを全てreplaceに置き換え、その結果の文字列を返す。

```
x = “This is a pen”
y = x.gsub(/is/){“IS”}
y
⇒ “ThIS IS a pen”
```

一般的なソート（昇順ソート）

```
x = [9, 20, 10, 15, 8]
y = x.sort
y
```

⇒ [8, 9, 10, 15, 20]

国語の点が高い順に並べ換え

国語の点数を取り出すには、

```
x[“国語”].to_i
```

とすれば良いので、

```
score.sort_by{|x| x[“国語”].to_i}
```

で国語の点数で昇順にした配列が得られる。

データベース

値

ハッシュ

科目名	基礎プログラミング
科目コード	154
担当教員	鳥海三輔
開講時期	春学期
概要	プログラミングを通じて問題解決能力の向上を図る。

ハッシュ

科目名	応用プログラミング
科目コード	254
担当教員	飯森大和
開講時期	秋学期
概要	C言語を通じて問題解決に最適なアルゴリズムを適用する方策を学ぶ。

ハッシュ

科目名	応用もっけ論
科目コード	39
担当教員	酒田育造
開講時期	春学期
概要	その地域の「もっけ」を理解し、深い交流のきっかけとする。

前回の問い

- Rubyでデータの集合的な処理を行うにはどのようにすれば良いか？
  - 配列処理メソッドを用いる。
- データを並べ換えるメソッドは何か？
  - sort (昇順)、reverse (降順)を用いる。
- CSVファイルを処理するにはどのようにすれば良いか？
  - CSV.readを利用する。

タイピング練習スケジュール

第２回　　ホームポジション
第３回　　ローマ字
第４回　　英語初級
第５回　　日本国憲法
第６回　　ホームポジション
第７回　　ローマ字
第８回　　英語初級
第９回　　日本国憲法
第１０回　日本国憲法
第１１回　日本国憲法（trr試験、合格点：200点）

タイピングの練習 (英語初級)

1. ブラウザを起動し、https://www.koeki-prj.org/trr/ に繋ぐ。
2. 学籍番号（Cは大文字、省略なし８桁）を入力する。
3. Koeki MAILに届いたパスコードをPasscode: 欄に入力する。

今回

第１回　生成AIを効果的に利用した学習法
第２回　メソッド定義と効果的活用（関数的処理）
第３回　メソッド定義と効果的活用（データ集合処理）
第４回　メソッド定義と効果的活用（再起的アルゴリズム）
第５回　変数のスコープ・クラス設計（変数のスコープ規則）
第６回　変数のスコープ・クラス設計（クラス定義）
第７回　CGIと情報システム（CGIの基本要素）
第８回　CGIと情報システム（持続的な値のやり取り）
第９回　CGIと情報システム（実践的処理）
第１０回　チーム作品作成
第１１回　チーム作品の構築・trr試験
第１２回　チーム作品発表・評価
第１３回　合同成果発表会
第１４回　期末試験

今回の目標

再帰的な方法を用いてプログラムを書けるようになる。

今回学ぶこと

1. 再帰とは
2. 再帰処理による並べ換え

今日の問い

- 再帰とは何か？
- 再帰処理を用いて並べ換えを行うにはどのようにすれば良いか？

今回学ぶこと

1. 再帰とは
2. 再帰処理による並べ換え

再帰

何かの中身にそれ自身が現れること

再帰的定義

何かの定義にそれ自身が登場するもの

再帰的定義の例：年齢

- 定義１：生まれた年月日から経過した年数
- 定義２：１年前の**年齢**に１を足したもの

再帰的定義

再帰を利用したプログラミング

例題：自然数に対する階乗

定義１：再帰的でない場合の定義

Nの階乗とは

1からNまでのすべての自然数を掛けたもの

再帰を利用したプログラミング

例題：自然数に対する階乗

定義２：再帰的定義

Nの階乗とは

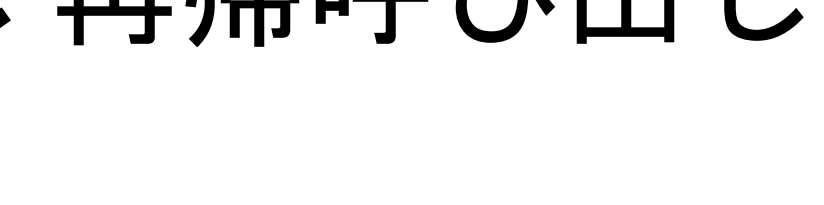
N-1の階乗とNを掛けたものである。ただし、1の階乗は1である。

Nの階乗を求めるメソッド（定義１）

```
def factorial (n)
    ans = 1
    i = 1
    while i <= n
        ans *= i
        i += 1
    end
    ans
end
```

Nの階乗を求めるメソッド（定義２）

```
def factorial (n)
    if n <= 1
        1
    else
        n * factorial (n-1)
    end
end
```



factorial.rbというファイルを作成し、右のコードを書いて実行してみよう。

```
#! /usr/koeki/bin/ruby
# coding: utf-8

def factorial (n)
    if n <= 1
        1
    else
        n * factorial (n-1)
    end
end

if ARGV[0] == nill
    STDERR.puts “階乗を計算したい数を指定して下さい”
    STDERR.puts “例：./factorial.rb 5”
    exit 0
end

n = ARGV[0].to_i
printf(“%dの階乗は%dです。\n”, n, factorial (n))
```

再帰的メソッドの注意点

- 問題を分割できるとき、分割部分を解く考え方と全体を解く考え方が同じとき、再帰を使える。
- 問題分割を繰り返し、これ以上問題が分割できない場合に結果そのものを返す条件ブロックを入れる（**脱出条件**）
- 分割した問題を再帰的に同じメソッドを呼んで得られた結果を合成したものを最後の結果として返す

今日の問い

- 再帰とは何か？
  - 何かの中身にそれ自身が現れること。
- 再帰処理を用いて並べ換えを行うにはどのようにすれば良いか？

今回学ぶこと

1. 再帰とは
2. 再帰処理による並べ換え

クイックソートの手順

大きな箱Xに任意個のデータが入っているとする。

1. Xの中のデータが１個以下なら並べ換え完了。そうでなければ次に進む。
2. Xの中から、任意のデータを１個抜き取り、その値をkとする。
3. kを抜き取ったあとの中のデータを順次見ていき、kより小さかったら左側に置く、そうでなければ右側に置くことをデータがなくなるまで繰り返す。
4. 左側に寄せたものをクイックソートする。
5. 右側に寄せたものをクイックソートする。
6. 左右のかたまりの中間にkを置いて並べ換え完了。

クイックソートによるカードの並べ換え

1. 並べ換えるべき枚数が１枚以下なら終了、それ以上なら次に進む。
2. まず１枚のカードを机の真中に置く。
3. 手に残ったカードから１枚ずつ引いていき、真中に置いたカードの数字より小さかったら左側に置く。そうでなければ右側に置く。
4. 左側に置いた山を左側の机スペースでクイックソートする。
5. 右側に置いた山を右側の机スペースでクイックソートする。
6. おしまい。

クイックソートを行うメソッドqsortの設計

- qsortメソッドは引数として配列を受け取る。
- 左側の箱、右側の箱としてそれぞれ、配列left、配列rightを利用する。
- left、rightに対して再帰的にqsortする。
- 最後にleft、真中の値、rightを並べて完了。

クイックソートを行うメソッドqsortの設計

```
def qsort (data)
    if データが１個以下なら
        dataそのものを返して並べ換え完了
    else
        left (左側配列)を新規作成
        right (右側配列)を新規作成
        変数dataからデータを1個取り出す
        これをkとする
        残ったdata全てに対して以下を繰り返す
            if kより小さかったら
                leftに追加
            else
                rightに追加
            end
        繰り返し終わり
        「leftをqsortしたもの」とkと「rightをqsortしたもの」をその順に結合したものを結果として並べ換え完了
    end
end
```

配列の初期化

x = [] または、x = Array.new(個数, 値) で行う。

(復習) shift: 先頭要素の取り出し

配列の先頭要素を取り出し、その値を返す。配列は一つずつ前に詰められる。

```
x = [3, 1, 2, 4]
x.shift
⇒ 3
x => [1, 2, 4]
```

push(val): 先頭への要素の追加

配列の末尾にvalを追加し、その配列を返す。

```
x = [3, 1, 2, 4]
x.push(20)
⇒ [3, 1, 2, 4, 20]
x
⇒ [3, 1, 2, 4, 20]
```

(復習)メソッド“<<”: 末尾への破壊的要素追加

末尾に新しい要素を追加する。元の配列や文字列自体が変更される。

配列

```
x << 5
=> [3, 4, 1, 2, 5]

x
=> [3, 4, 1, 2, 5]
```

文字列

```
s << “hoge”
=> “abchoge”

s
=> “abchoge”
```

元の値を直接書き換える操作を**破壊的操作**という。

(復習)メソッド“+”: 連結・結合

メソッド“+”は2つのオブジェクトを繋げた新しい配列/文字列を返す。

配列

x = [3, 4, 1, 2]
y = [9, 8, 7]

x + y
=> [3, 4, 1, 2, 9, 8, 7]

x => [3, 4, 1, 2]
y => [9, 8, 7]

文字列

s = “abc”
t = “hoge”

s + t
=> “abchoge”

s => “abc”
t => “hoge”

(復習) join (sep): 配列を結合して文字列化

配列の各要素の間に文字列sepを挟んで連結した文字列を返す。sepを省略した場合は、配列の要素をそのまま文字列化する。

```
x = [3, 1, 2, 4]
x.join("/")
⇒ "3/1/2/4"
```

```
x.join(", ")
⇒ "3, 1, 2, 4"
x.join()
⇒ "3124"
```

クイックソートを行うメソッドqsort

```
def qsort (data)
    if data.length <= 1
        data
    else
        left = Array.new
        right = Array.new
        k = data.shift

        for x in data
            if x < k
                left << x
            else
                right << x
            end
        end
        qsort(left).push(k) + qsort(right)
    end
end
```

配列を読み込むメソッド

標準入力から順次数値を読み取り配列にするメソッド

```
def readNumArray()
    ary = []                                #入力が続く間
    while line=gets                         #整数に直して
        datum = line.chomp!.to_i            #aryに追加していく
    end
    ary                                     #最後にaryを返す
end
```

配列を画面表示するメソッド

```
def printArray(a)
    printf(“[%s]\n”, a.join(“,  ”))
end
```

クイックソートプログラム完成形

これまでに紹介したメソッドを組み合わせたプログラムqsort.rb (https://www.koeki-prj.org/~yuuji/2025/pf2/05/qsort.rb) を保存して実行し、１行に１つずつデータを入れて小さい順に並べ換えできることを確認しよう。

問題

qsortメソッドを書き換えて、数値の大きい順に並べ換えるようにせよ。実際に実行してそれを確かめよ。

今日の問い

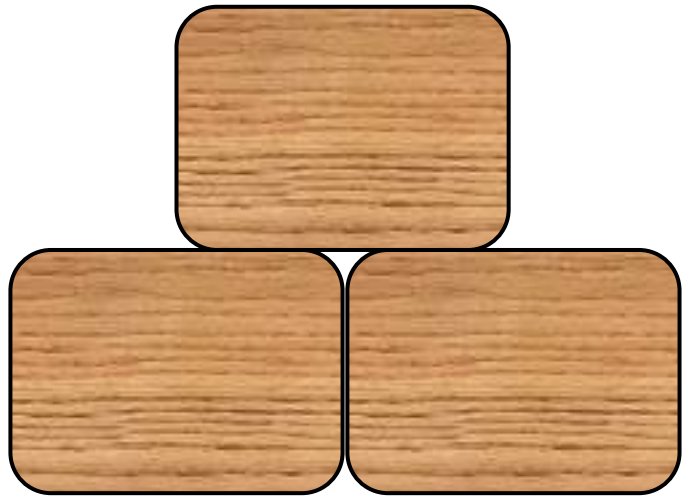
- 再帰とは何か？
  - 何かの中身にそれ自身が現れること。
- **再帰処理を用いて並べ換えを行うにはどのようにすれば良いか？**
  - **配列を２つに分け、それぞれを再帰的に並べ換える。**

授業内課題

次ページのプログラムをゼロから作成する過程を見せつつ意味をTAに説明せよ。作成予定のプログラムをs4基礎プロII(F)の「#04授業内課題 (再帰)」に指示どおり書き込んでからTAを呼ぶこと。

授業内課題

積み木を三角形に積み上げた家を作る。２段の家を作るときは２つの積み木の上に１個の積み木を載せて完成、３段の家を作るときは３個の積み木の上に２個の積み木を載せさらにその上に１個の積み木を載せて完成。以下同様にn段の家を作るときはn個の積み木から組み上げるルールで作るものとする。



1. n段の家を作るときに必要な個数を求める方法を再帰的な定義で示せ。
2. その定義を利用し、実際に必要な個数を求めるメソッドを作れ。
3. 上で作ったメソッドを利用しn段の家を作るときに必要な積み木の個数を求めるプログラムbrick.rbを作成せよ。

今回の目標

再帰的な方法を用いてプログラムを書けるようになる。

今回学ぶこと

1. 再帰とは
2. 再帰処理による並べ換え

次回

第１回　生成AIを効果的に利用した学習法
第２回　メソッド定義と効果的活用（関数的処理）
第３回　メソッド定義と効果的活用（データ集合処理）
第４回　メソッド定義と効果的活用（再起的アルゴリズム）
第５回　変数のスコープ・クラス設計（変数のスコープ規則）
第６回　変数のスコープ・クラス設計（クラス定義）
第７回　CGIと情報システム（CGIの基本要素）
第８回　CGIと情報システム（持続的な値のやり取り）
第９回　CGIと情報システム（実践的処理）
第１０回　チーム作品作成
第１１回　チーム作品の構築・trr試験
第１２回　チーム作品発表・評価
第１３回　合同成果発表会
第１４回　期末試験

事前課題（中間試験練習問題）

「出席番号,氏名,身長(cm),体重(kg)」が記されたCSVファイルsntw.csv (https://www.koeki-prj.org/~yuuji/2025/pf2/exam-mid/sntw.csv)がある。以下のことを行うプログラムをどれか１つ選んで作成し、実行結果とともに提出せよ。

このcsvファイルを自動的に開いて

1. 全ての行をそのまま出力するプログラムprintline.rb
2. 全ての氏名を順次出力するプログラムprintname.rb
3. 全員の体重の合計を出力するプログラムprintaw.rb
4. 全員の体重の平均値を出力するプログラムprintaw.rb
5. 全員のデータを読んでから身長について降順に「氏名、身長」を出力するプログラムsortheight.rb
6. 全員のデータを読み取った後出席番号を質問し、その出席番号の生徒の氏名・身長・体重を出力するプログラムsearchcsv.rb
7. 全員のデータを読み身長または体重どちらか好きな方で並べ替えた順で全てのデータを出力するプログラムsortany.rb

事前課題提出方法

s4基礎プロII(F)の「#05事前課題 (中間試験)」に指示どおり書き込む。提出後修正して良い。その場合は「編集」リンクから書き換えること。

締切：**10月27日 (月)**