

情報処理技法 (Javaプログラミング)2

第3回 アルゴリズム(1)

人間科学科コミュニケーション専攻
白銀 純子

第3回の内容

▼ アルゴリズム

前回の出席課題の解答

- ▼ 下記のプログラムを実行したとき、確実に標準出力に出力されるメッセージは何かを答えなさい。

```
try {
    String line[] = new String[5];
    int i;

    FileReader fr = new FileReader("sample.txt");
    BufferedReader br = new BufferedReader(fr);

    for(i = 0; i < 100; i = i + 1) {
        line[i] = br.readLine();
    }
    br.close();
    fr.close();
}
catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("指定された範囲外の添え字を使おうとしています。");
}
catch (NumberFormatException e) {
    System.out.println("数値に変換できない文字列を数値に変換しようとしています。");
}
catch (FileNotFoundException e) {
    System.out.println("指定されたファイルが存在しません。");
}
catch (IOException e) {
    System.out.println("指定されたファイルを読み込むことができません。");
}
```

確実に標準出力に
出力されるメッセージ

アルゴリズム



アルゴリズムとは

- ▼ アルゴリズム: ある問題を解決するときに必要な処理手順
 - ▼ プログラムを書くときには、必ずアルゴリズムを考える必要
 - ▼ プログラム: アルゴリズムをプログラミング言語を使って記述したもの

Ex. 焼きそば作りのアルゴリズム:

1. キャベツや肉などの具をフライパンで炒めなさい。
2. フライパンからいったん具を取り出さなさい。
3. めんをフライパンで炒めなさい。
4. 具をめんの入っているフライパンに戻しなさい。
5. 焼きそばソースを加えてさらに炒めなさい。

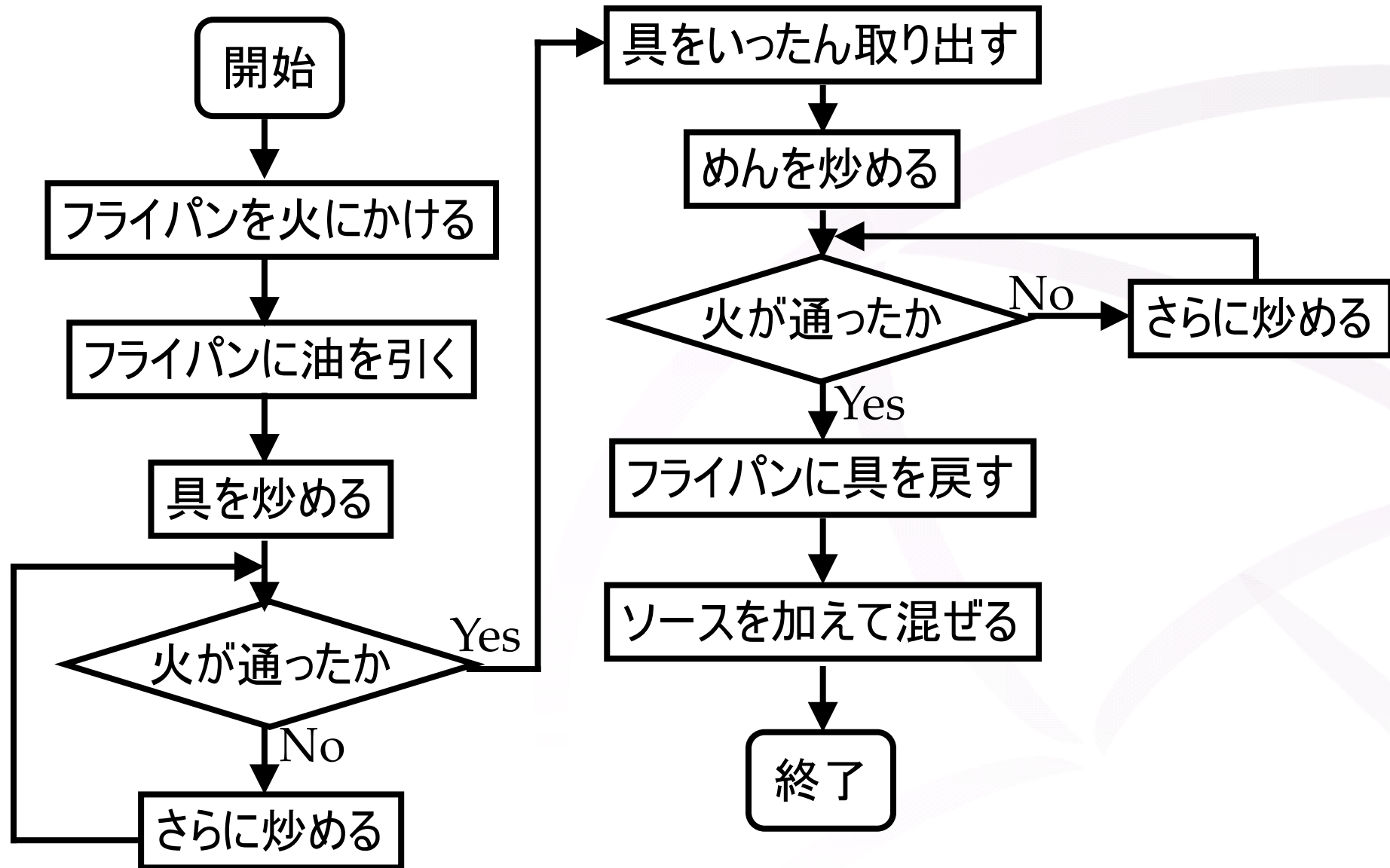
アルゴリズムの表現方法

- ▼ 文章で書く
 - ▼ 箇条書きで書くことも多い
- ▼ 図で書く
- ▼ プログラムで書く
 - ▼ プログラムが最終段階



プログラムで行う処理手順がわからなければ、まず文章で書き出して、それを詳細にしていく

焼きそば作成(図)



焼きそば作成(プログラム1)

焼きそばのアルゴリズムをプログラムの的に表現

```
public static void main(String[] args) {  
    // 開始  
    フライパンを火にかける;  
    フライパンに油を引く;  
    具を炒める;  
    while (火が通っていない) {  
        // Yes(「火が通ったか?」に対して「No」)  
        さらに炒める  
    }  
    // 「火が通ったか?」に対して「Yes」  
    具をいったん取り出す;  
    めんを炒める;  
}
```

焼きそば作成(プログラム2)

焼きそばのアルゴリズムをプログラムの的に表現(続き)

```
while (火が通っていない) {  
    // Yes(「火が通ったか?」に対して「No」)  
    さらに炒める;  
}  
// 「火が通ったか?」に対して「Yes」  
フライパンに具を戻す;  
ソースを加えて混ぜる;  
// 終了  
}
```

最大公約数のアルゴリズム

素因数分解

1. 1つ目の数を素因数分解する
2. 2つ目の数を素因数分解する
3. それぞれの素因数の共通項を掛け合わせる

Ex. 136と24の最大公約数

$$136 = 2 \times 2 \times 2 \times 17$$

$$24 = 2 \times 2 \times 2 \times 3$$

➡ 共通項: $2 \times 2 \times 2 = 8$

➡ 最大公約数: 8

最大公約数

- ▼ 2つの数の最大公約数を求めるアルゴリズムは2通り
 - ▼ 素因数分解で求める
 - ▼ ユークリッドの互助法で求める

ユークリッドの互助法

1. 2つの数のうち、大きい方を小さい方で割る
2. 1. の余りが0でない場合、その余りと小さい方の数を改めて2つの数とし、1. へ戻る
3. 1. の余りが0の場合、割った数が最大公約数となる

Ex. 136と24の最大公約数

$$\begin{aligned} &136 \div 24 = 5 \dots 16 \\ \rightarrow &24 \div 16 = 1 \dots 8 \\ &16 \div 8 = 2 \dots 0 \end{aligned}$$

 最大公約数: 8

筆算のアルゴリズム



自然数の加算(1)

▼ 2つの自然数の和を求める = 筆算

- ▼ n 桁目の数の和を求め、10以上になった場合には $n+1$ 桁目に1を加える

n が1から自然数の桁数まで繰り返す

▼ 準備

- ▼ 2つの自然数は同じ桁数とする
 - ▼ 50と200の場合、050と200として考える
- ▼ 加えた結果を入れる変数(配列が便利)を用意する
- ▼ 繰り上がりを入れる変数を用意し、0を代入しておく

自然数の加算(2)

1. 1の位から最上位の位まで、a. ～d. を繰り返す
 - a. その位の2つの数の和を求める
 - b. 下位からの繰り上がりがあればそれも加える
 - c. その和の1桁目を、求める和のその位の結果とする
 - d. その和の2桁目を、次の桁への繰り上がりとする
2. 最上位からの繰り上がりがあった場合は、求める和のその次(上位)の位の数とする

ATMのアルゴリズム



▼ 人間から見える手順

1. 「引き出し」ボタンを押す
2. キャッシュカードを入れる
3. 暗証番号を入力する
4. 引き出し金額を入力する
5. 現金を取り出す

▼ ATM側での処理手順

1. 初期画面を表示してボタンが押されるのを待つ
2. 押されたボタンによってサービスの種類を判別する
 - a. 「引き出し」ボタンの場合、3. へ進む
 - b. 「預け入れ」ボタンの場合、4. へ進む.....
3. サービスの種類が「引き出し」の処理
4. サービスの種類が「預け入れ」の処理
.....
5. 「ありがとうございました」の画面を表示して、1. へ戻る

▼ 「引き出し」の処理手順

- a. カードを入れてくださいと表示する
- b. 適切なカードかどうかを調べ、キャッシュカードではないカードであれば一. へ進む
 - 一. 入れられたカードを排出し、「キャッシュカードを入れなおしてください。」と表示する
 - 二. b. へ戻る
- c. 暗証番号を調べ、番号が間違っていたら一. へ進む
 - 一. 「暗証番号を入力しなおしてください。」と表示する
 - 二. c. へ戻る
- d. 「引き出す金額を入力してください。」と表示し、引き出し金額を入力してもらう
- e. 引き出し金額を調べる

▼ 「引き出し」の処理手順の続き

- f. 引き出し金額が残高よりも多ければ、一. へ進む
 - 一. 「残高を超えています。引き出し金額を入力しなおしてください。」と表示する
 - 二. f. へ戻る
- g. 出金処理を行い、預金残高を更新する
- h. 預金残高を表示し、5. へ進む

有名なアルゴリズム



並べ替え(ソート)

- ▼ ソート: 複数の数を小さい or 大きい順に並べること
 - ▼ バブルソート
 - ▼ 選択ソート
 - ▼ 併合ソート
 - ▼ クイックソート
 - ▼ etc.

バブルソート(1)

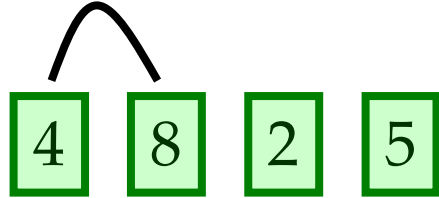
- ▼ 前から2つずつ、数の大きさを比較して、小さい数を後ろに送っていく
 - ▼ 最後まで調べると、最も小さな数が一番後ろにある
 - ▼ この作業を、並べ替える数の個数だけ繰り返すと、数が大きい順に並ぶ

- ▼ $x_1 < x_2$ ならば、 x_1 と x_2 を入れ替える
- ▼ $x_2 < x_3$ ならば、 x_2 と x_3 を入れ替える
- ▼ ...
- ▼ $x_{n-1} < x_n$ ならば、 x_n と x_{n-1} を入れ替える

- n 個の数でこの処理が終わった後に n 番目に最も小さな数
- $n-1$ 回この処理を繰り返すことで、大きい順に数を並べ替え

バブルソート(2)

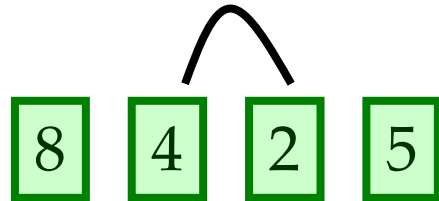
比べる



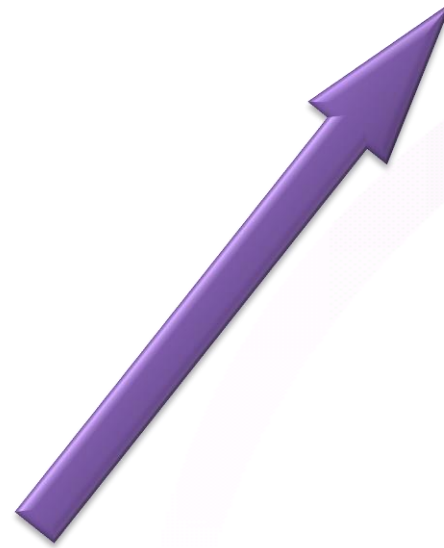
4より8の方が大きいので交換



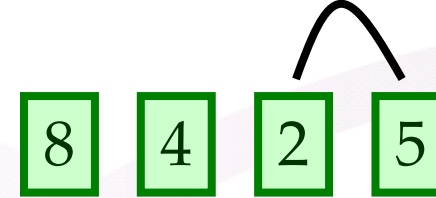
比べる



2より4の方が大きいのでそのまま



比べる



2より5の方が大きいので交換



- 最も小さな数が一番後ろに来る
- 次は、一番後ろの1つ前まで (8, 4, 5)で同じようにする

選択ソート(1)

- ▼ 変数 t : 並べ替えをする数の中で最も小さい数を入れておく変数
- ▼ 変数 i : 並べ替えをする数の中で、 t が最初から何番目の位置にあるかを表す変数
- ▼ 変数 j : 何回繰り返したかを数えるための変数
- ▼ 並べ替えをする数は n 個

選択ソート(2)

1. t に並べ替えをする数の一番最初の数を代入する
2. i に1を代入する
 - ▼ 1: 並べ替えをする数の一番最初の数の位置
3. j に2を代入し、1ずつ増やしながら n になるまで以下を繰り返す
 - ▼ t が j 番目の数より大きいならば、 j 番目の数を t に代入し、 i に j の値を代入する
 - この処理を1回することにより j の値を1増やす
 - j の値は、現在調べている数の位置になる
4. 並べ替えをする数の一番最後の数と t を入れ替える

選択ソート(2)

4 8 2 5

ステップ1:

- tに4を代入する
- iに1を代入する

ステップ2:

- jに2を代入する
- tの値(4)と8を比べる
- tの値の方が小さいのでそのまま

ステップ3:

- jの値を1増やす(3になる)
- tの値(4)と2を比べる
- tの値の方が大きいのでtに2を代入し、iにjの値(3)を代入する

ステップ4:

- jの値を1増やす
- tの値(2)と5を比べる
- tの値の方が小さいのでそのまま

ステップ5:

- tの値(2)を最後に置く
- これまで最後だった数(5)をi番目に入れる



4 8 5 2

- 最も小さな数が一番後ろに来る
- 次は、一番後ろの1つ前まで(8, 4, 5)で同じようにする

探索(1)

▼ たくさんのデータから目的のデータを見つけること

Ex. 高校の生徒の得点を管理するプログラム

- 出席番号5番の生徒の英語の点数を知りたい
→ 高校の生徒のデータから、出席番号が「5」というものを探す
- 「東京子」という生徒の国語の点数を知りたい
→ 高校の生徒のデータから、名前が「東京子」というものを探す

探索(2)

▼探索のアルゴリズム

▼逐次探索

▼2分探索

▼自己組織化探索

▼2次元探索

▼補間探索

▼ハッシュ法

逐次探索

▼ データを前から順番に比較して探していく方法

1. データが配列 $a[0] \sim a[N]$ に入っている
2. 「 $a[i] = x$ 」(x は探したいデータ)となる「 i 」を探す
 - ▼ 「 i 」は $0 \sim N$ の整数

▼ 欠点: データが配列の後ろのほうにある場合に見つけるまでに時間がかかる

2分探索(1)

- ▼ ソートされたデータの中から目的のデータを探す方法
- 1. データが配列 $a[0] \sim a[N]$ に入っている
- 2. 中央のデータ「 $a[(0 + N)/2]$ 」と「 x 」を比較する
 - ▼ x : 探したいデータ
- 3. 2. の結果、 $a[m] < x$ の場合、データは右半分に $x \leq a[m]$ の場合、データは左半分に存在
 - ▼ $m: (0 + N)/2$
 - ▼ 右半分: $a[m+1] \sim a[N]$
 - ▼ 左半分: $a[0] \sim a[m]$

2分探索(2)

4. データが右半分にある場合、 $a[m+1] \sim a[N]$ で2. をする
5. データが左半分にある場合、 $a[0] \sim a[m]$ で2. をする
6. 「 $a[\text{left}] \sim a[\text{right}]$ 」での探索で、「 $\text{left} = \text{right}$ 」となると探索終了
7. 目的のデータは「 $a[\text{left}]$ 」

2分探索(3)

「1, 2, 3, 4, 5, 6, 7, 8, 9, 10」の中から「9」を見つけたい

- $a[(0+9)/2]$ (値: 5)と「9」を比較し、「9」の方が大きいので、「9」は右半分($a[(0+9)/2+1] \sim a[9]$)の範囲にある
- $a[(5+9)/2]$ (値: 8)と「9」を比較し、「9」の方が大きいので、「9」は右半分($a[(5+9)/2 + 1] \sim a[9]$)の範囲にある
- $a[(8+9)/2]$ (値: 9)と「9」を比較し、「9」と同じなので、「9」は左半分($a[(5+9)/2] \sim a[(8+9)/2]$)の範囲にある
- $a[(7+8)/2]$ (値: 8)と「9」を比較し、「9」の方が大きいので、「9」は右半分($a[(7+8)/2+1] \sim a[8]$)の範囲にある
- 探索する配列の添え字が同じになったので、探索終了
- 結果: $a[8]$ が回答

やってみよう!

▼ 逐次探索

1. 配ったカードをバラバラの順序にする
2. その中から「18」を探してみる

▼ 2分探索

1. 配ったカードを昇順に並べる
2. カードの中から「18」を探してみる

やってみよう!(1)

- ▼ ユークリッドの互助法のプログラム
 - ▼ 2つの数を標準入力で
- ▼ ATMの「引き出し」と「預け入れ」の機能のプログラム
 - ▼ 預金残高と暗証番号は、あらかじめプログラム中に書いておく
 - ▼ カードに関する処理はなしでOK
 - ▼ 出金処理と入金処理は、標準入力で処理
 - ▼ 出金: 「xx円出金し、yy円残っています。」と表示
 - ▼ 入金: 入金する金額を標準入力で入力

やってみよう!(2)

▼「アルゴリズム」という教材に挑戦してみよう

▼ <http://home.jeita.or.jp/is/highschool/algo/index.html>

▼ アルゴリズムの考え方を身につけるのに良い教材