

# ソフトウェア概論 A/B

## -- データ構造 (2) --

(配列とその応用)

数学科 栗野 俊一 / 渡辺 俊一

2015/10/23 ソフトウェア概

# 伝言

---

## 私語は慎むように !!

### □ 色々なお知らせについて

- 栗野の Web Page に注意する事

<http://edu-gw2.math.cst.nihon-u.ac.jp/~kurino>

### □ 講義開始前に済ませておく事

- PC の電源を入れる
- ネットワークに接続しておく事
- 今日の資料に目を通しておく事

### □ 講義前の注意

- 講義前は、栗野は準備で忙しいので TA を捕まえてください

### □ やる気のある方へ

- 今日の資料は、すでに上っています

▶ どんどん、先に進んでかまいません

### □ 本日の CST Portal の出席パスワード : 20151023

- 出席は成績に影響しませんが、折角の機能なので、使いましょう

# 前回(2015/10/16)の内容

---

## □ 前回 (2015/10/16) の復習

### ○ コーディングとプログラムの機能

- ▶ コーディング : 計算機の内部(データ)と現実(情報)の間の対応規則
- ▶ プログラム(数値の計算)が現実の世界に影響を及ぼすのは、コーディングのため

### ○ データ構造

- ▶ 複雑な対象を表すにはデータを「組み合わせる」必要がある
- ▶ 「複数のデータを一つのデータに見せる」仕組みがあると便利 : データ構造

### ○ 構造体と配列

- ▶ 異なる型のデータを組合せる仕組み : 構造体
- ▶ 同じ型のデータを組合せる仕組み : 配列 ( 表現力が弱いけど添字が使えるのが強み )

# お知らせ

---

- 本日の予定

- データ構造 (2)

- ▶ 配列とその応用

- 本日の目標

- 演習

- ▶ 課題の提出

# 前回 (2015/10/16) の課題

---

## □ 前回 (2015/10/16) の課題

### ○ 課題 20151016-01:

- ▶ ファイル名 : 20151016-01-QQQQ.c (QQQQ は学生番号)
- ▶ 内容 : 極座標で表現されている点  $Q$  から、それと原点に対して対称な点  $R$  を求める

### ○ 課題 20151016-02:

- ▶ ファイル名 : 20151016-02-QQQQ.c (QQQQ は学生番号)
- ▶ 内容 : 構造体を利用し、平行移動を行う関数を作成する

### ○ 課題 20151016-03:

- ▶ ファイル名 : 20151016-03-QQQQ.c (QQQQ は学生番号)
- ▶ 内容 : 3次元ベクトルの差の計算

## □ ※

- ファイル形式は、いずれもテキストファイル(C言語プログラムファイル)

# 本日の課題 (2015/10/23)

---

## □ 本日 (2015/10/23) の課題

### ○ 課題 20151023-01:

- ▶ ファイル名 : 20151023-01-QQQQ.c (QQQQ は学生番号)
- ▶ 内容 : 複素数型の四則
- ▶ ファイル形式 : テキストファイル(C 言語プログラムファイル)

### ○ 課題 20151023-02:

- ▶ ファイル名 : 20151023-02-QQQQ.c (QQQQ は学生番号)
- ▶ 内容 : 二次元行列の和、差、積
- ▶ ファイル形式 : テキストファイル(C 言語プログラムファイル)

### ○ 課題 20151023-03:

- ▶ ファイル名 : 20151023-03-QQQQ.c (QQQQ は学生番号)
- ▶ 内容 : 配列を使って 5 個の数を入力し、その 5 倍と  $1/2$  を出す
- ▶ ファイル形式 : テキストファイル(C 言語プログラムファイル)

## □ ※

- ファイル形式は、いずれもテキストファイル(C 言語プログラムファイル)

# 情報の表現 (再)

---

## □「情報」の「表現」方法

- コンピュータは、「数値」の「計算」しかできない
- 現実には、「様々な情報」の「処理」がしたい
  - ▶ この二つのギャップを埋めるのは何か？

## □ 情報を巡る、三つの「形態」

- 情報：「現実」世界での「何か(例:音)」
  - ▶ Input(入力)：「情報」を「データ」に変換する
  - ▶ Output(出力)：「データ」を「情報」に変換する
  - ▶ I/O：ハードウェア (音:マイク/スピーカ) によって実現
- データ：「コンピュータ」内での「情報」の表現(例:正弦波)
  - ▶ Encode(符号化)：「データ」を「数値」に変換する
  - ▶ Decode(解釈)：「情報」を「データ」に変換する
  - ▶ コーディング規則：ソフトウェア(プログラム)によって実現
- 数値：「コンピュータ」が直接扱える形 (数値)
  - ▶ 「計算」する事ができる
  - ▶ 計算：コンピュータ自身の基本的な機能によって実現

# 情報の処理

---

## □ 情報処理の目的

- 「現実」世界での「何か(例:音)」を「操作」したい

- ▶ 機能 ( Function ) : 情報を操作する能力

## □ 情報処理の流れ

- 情報(Information)は、入力(Input)され、データ(Data)になる
- データ(Data)は、符号化(Encode)され、数値(Number)になる
- 数値(Number)は、計算(Cluculus)され、別の数値になる
- 別の数値は、解釈(Decode)され、別のデータになる
- 別のデータは、出力(Output)され、別の情報になる

## □ 同型構造

- 情報とデータは I/O により「同型構造」を持つ

- ▶ 対応は「全射」でなければならない

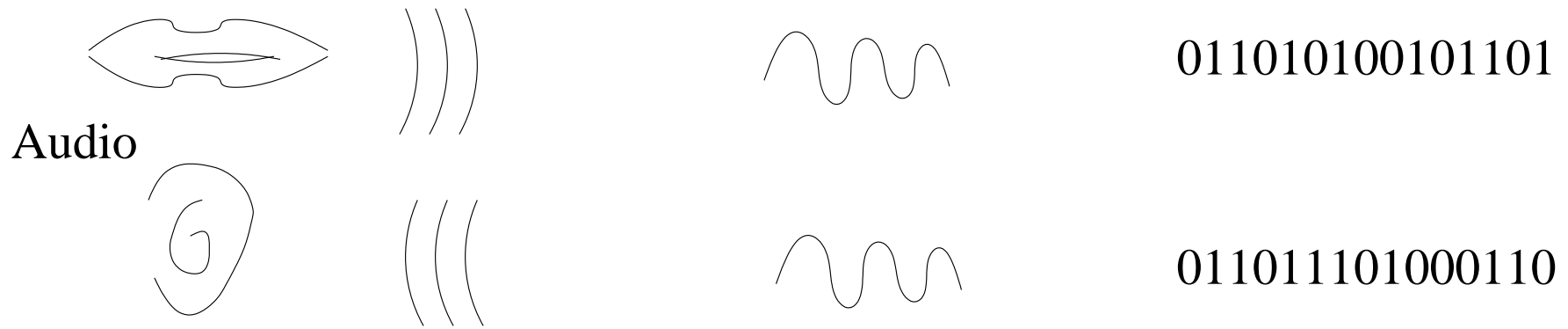
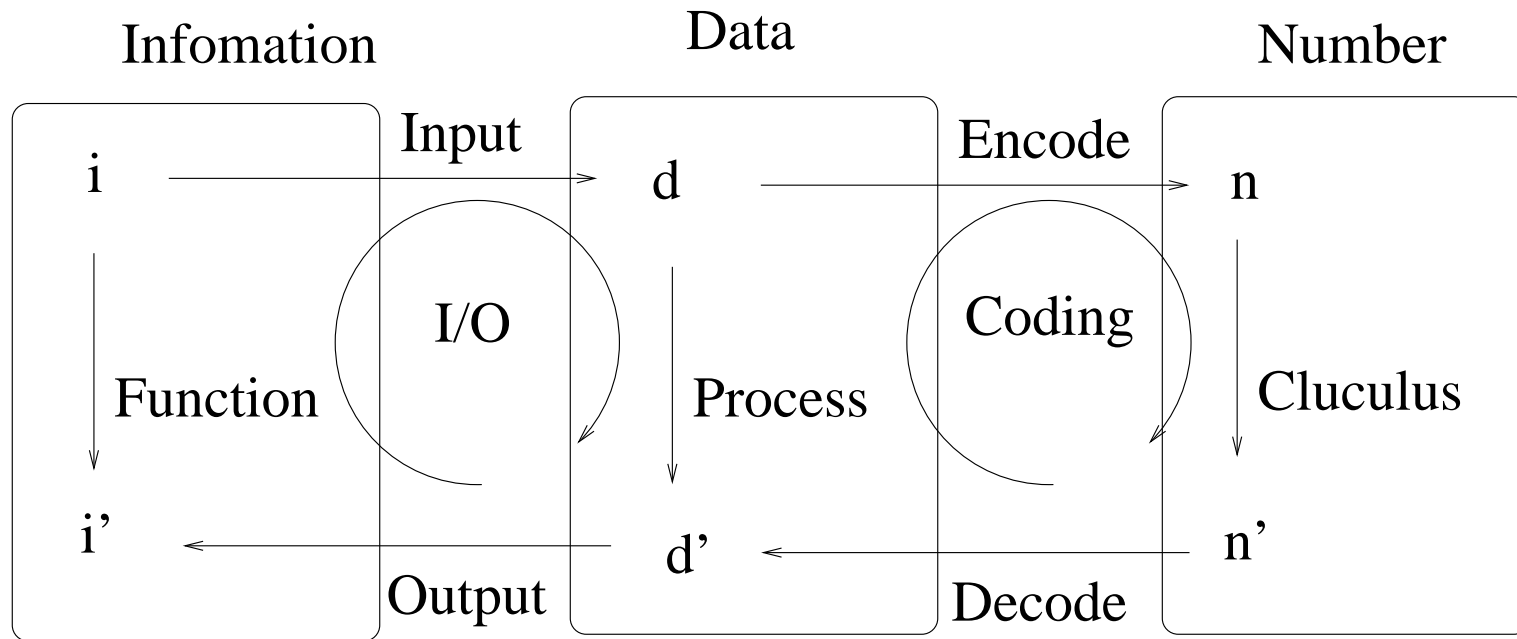
- データと数値は Coding により「同型構造」を持つ

- ▶ 同型構造によって、機能(Function)は、処理(Process)を経由して計算(Cluculus)される

- ▶ これらは「可換」になっている



# 情報の処理の構造図



# コーディングの仕組み

---

## □ コーディングの仕組み

### ○ 基本は「直積」(既存の情報の組み合わせ)で行う

- ▶ 例 1: 平面上の「点」の表現 → 「座標」=「数値の二つ組(x,y)」で表現
- ▶ 例 2: 有理数の表現 → 「分母・分子」=「数値の二つ組 p/q」で表現
- ▶ 直積は、空間を拡大する

### ○ 「制約」で、「同型」にする

- ▶ 無意味な組み合わせ(中への対応)や、重複(一対一でない)はプログラムで処理
- ▶ 例 1: 有理数で 3/0 に対応するものはない (エラー / 例外処理)
- ▶ 例 2: 有理数で 3/6 と 1/2 は同じ物に対応 (同値類 / 正規化)

## □ C 言語でのコーディング

### ○ データの表現

- ▶ 直積: 構造体
- ▶ 制約: 正規化のプログラムを作成する

### ○ 機能の実現

- ▶ 「計算(数値の操作)」によって、「機能」を実現するプログラムを作る

# 構造体 (再)

---

## □ 型

### ○ 「空間」の名前

- ▶ 集合(どんな要素が含まれているか..) と機能(どんな演算ができるか..) からなる

### ○ 基本型

- ▶ 予め C 言語で、定義されており、始めから利用できる型 ( int, double, etc.. )

### ○ 導出型

- ▶ プログラマが作成する型 ( 集合の定義と、機能は自分で実装する.. )

### ○ typedef

- ▶ typedef によって、新しい型に名前をつける事ができる
- ▶ 変数の宣言と、代入が可能になる
  - ◇ 構文 : typedef 型 新しい型名;
  - ◇ 例 1 : typedef int myInt; myInt を int で定義
  - ◇ 例 2 : typedef struct { int x; int y } Point;

## □ 構造体

### ○ 複数の既存の型から、その直積となる新しい型を作る

- ▶ 例1 : int x と int y の組み合わせ
- ▶ struct { int x; int y; }; ≡ { <x,y> | x:int, y:int }
- ▶ struct {int x;int y} v; // Point v;
- ▶ v ≡ <p,q>, v.x ≡ p, v.y ≡ q
- ▶ 例 2 : 三次元 ( int x, int y, int z ) の場合

# 配列(再)

---

## □ 配列

- 同じデータが並んだ物を表現する仕組み

  - ▶ 例: `double a0,a1,a2 -> double a[3]`

- 配列名：データの並びが入る変数の代表名

  - ▶ 添字「`[ + 整数値 ]`」を付けて、要素が参照できる

- 配列の宣言

  - ▶ 配列を利用する(宣言する)場合は、「`配列名[サイズ]`」の形にする

  - ▶ サイズ個数の変数がまとめて用意される

  - ▶ 参照する場合は `0 ~ サイズ-1` まで

  - ▶ 例: `int ary[10];` とすると `ary[0] ~ ary[9]` が使える

# 配列プログラミング (集合操作)

---

## □ 配列 vs 集合

- 配列は、複数の同じ型の変数(配列の要素)をまとめたもの
  - ▶ 個々の要素は、同じ型の値を保持する
  - ▶ 一つの配列(が保持する値の集まり) は、その型の「集合」を表す

## □ 集合操作

- 配列の要素(複数)への操作を「繰返し」で表現する
- 「集合全体への操作」が、「個々の要素の操作の繰返し」になる

## □ <注意>

- 配列は、集合のように要素数が変化したりはしない
- 配列の異なる要素(変数)が同じ値を持つ事もある

# データ構造(配列)の応用

---

## □ 配列

- 同じ種類の物が複数組み合わされた物 (同じでないといけない)

- ▶ <-> 構造体 : 異なる種類の物が複数組み合わされた物 (同じ種類でもよい)

## □ 配列使い方(その一)

- 同じ種類がまとまっている物

- ▶ -> 集合を表現している

- ▶ 「集合」を表現する場合は、配列の方が良い

- 「集合の操作」は「要素の操作の繰返し」になることが多い

- ▶ 「繰返し」と「配列」は相性がよい (for 文)

## □ 配列使い方(その二)

- 「集合」から、「集計」を行う

- ▶ 全体の情報を集約して一つのデータ書き換える作業

- ▶ 例 : 総和、平均、最大値、最小値..

# 多次元の配列

---

## □ 多次元の配列

- 配列の配列が作れる：多次元配列

- [例]

  - ▷ `int d[3][4]; /* 二次元 3 × 4 の 12 個の要素を持つ配列 */`

- 次元は幾つでも増やす事ができる

  - ▷ `int t[3][4][5]; /* 三次元配列 */`

## □ データ構造の利用

- データ構造を利用する事により、様々な機能がより簡単に表現できる

# データ構造とプログラム

---

- データ構造とプログラム構造は対応している
  - 基本プログラム構造：順接, 繰返し, 条件分岐
  - データ構造：構造体, 配列, 共用体(後述)
- データ指向
  - データ型をきちんと考えると、プログラムが自動的にできる
    - ▶ まず、データ型をきっちりと考える
- オブジェクト指向
  - 更にデータ型とその型に対するプログラムをまとめて扱う仕組(class 概念)を持つ
    - ▶ 今回は言葉だけ紹介 (C++ 言語や java では中心概念)