# 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

# 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 : 20141010
  - 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 (2014/10/03) の復習

- 前回 (2014/10/03) の復習
  - コーディングとプログラムの機能
    - コーディング：計算機の内部と現実の間の対応規則
    - 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数値を「組み合わせる」必要がある
    - 「複数のものを一つに見せる」仕組があると便利：データ構造
  - 構造体と配列
    - 異る型を組合せる仕組：構造体
    - 同じ型を組合せる仕組：配列 ( 表現力が弱いが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)
    - for 文
    - 配列とその応用
- 本日の目標
  - 演習
    - 課題の提出

# 前回 (2014/10/03) の課題

- 前回 (2014/10/03) の課題
  - 課題 1:
    - ファイル名 : 20141003-1-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
    - 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 2:
    - ファイル名 : 20141003-2-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
    - 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 3:
    - 前々回(2014/09/26)の課題 3 (20140926-3-QQQQ.c) を提出してください
    - 注意 : この課題 3 (20140926-3-QQQQ.c) は、今週の課題に回す

# 本日の課題 (2014/10/10)

- 本日 (2014/10/10) の課題
  - 課題 1:
    - ファイル名 : 20141010-1-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 2:
    - ファイル名 : 20141010-2-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 3:
    - ファイル名 : 20141010-3-QQQQ.c (QQQQ は学生番号)
    - 内容 : 0 ～ 99 迄の偶数を出力する
    - 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 4:
    - ファイル名 : 20140926-3-QQQQ.c (QQQQ は学生番号)
    - 内容 : 3 次元ベクトルの差
    - 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)

# for 文

- □ for 文とは
  - ○ 繰返しを記述する構文規則 ( cf. while )
    - ▹ for ( <初期化式>; <継続条件式>; <再初期化式> ) { <繰返し文> }
    - ▹ 初期化式 : 最初に一度だけ、必ず行われる文
    - ▹ 継続条件式 : 毎回、繰返し文の実行「前」に評価されこれが偽の場合は終了になる
    - ▹ 繰返し文 : for 文によって繰り返される命令
    - ▹ 再初期化式 : 繰返し文の実行の後に毎回実行される
- □ for 文と while 文の関係
  - ○ for 文と while 文は相互に書き換えができる
    - ▹ だから、while 文だけ知っていれば for 文は要らないのだが、for 文は便利なので..
  - ○ for 文から while 文 ( while が解っていれば for 文は解る )
    - ▹ for ( <初期化式>; <継続条件式>; <再初期化式> ) { <繰返し文> }
      - ◇ →
    - ▹ <初期化式>; while ( <継続条件式> ) { <繰返し文> <再初期化式>; }
  - ○ while 文から for 文 ( while は for 文の簡略形 )
    - ▹ while ( <継続条件式> ) { <繰返し文> }
      - ◇ →
    - ▹ for ( ; <継続条件式>; ) { <繰返し文> }
    - ▹ ( while 文は for 文の <初期化式>, <再初期化式> が空のもの )

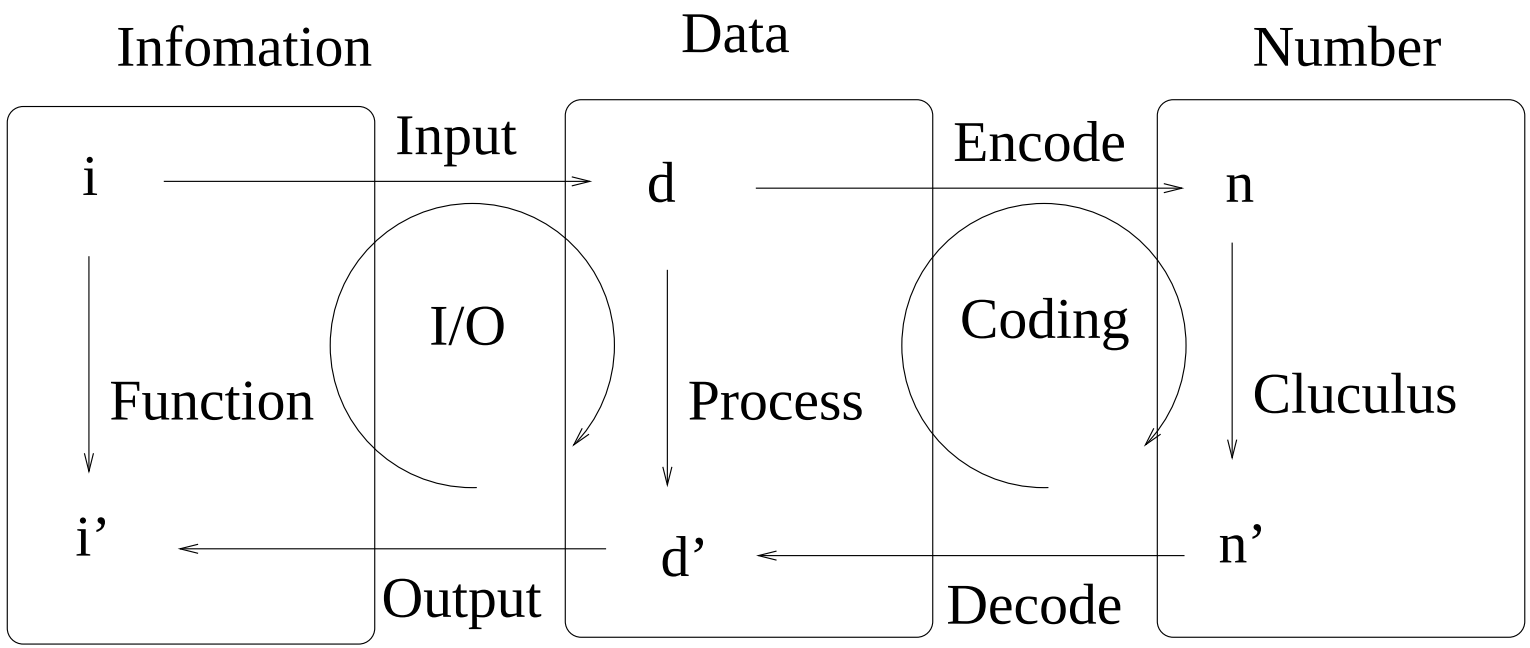
# 情報の表現 (再)

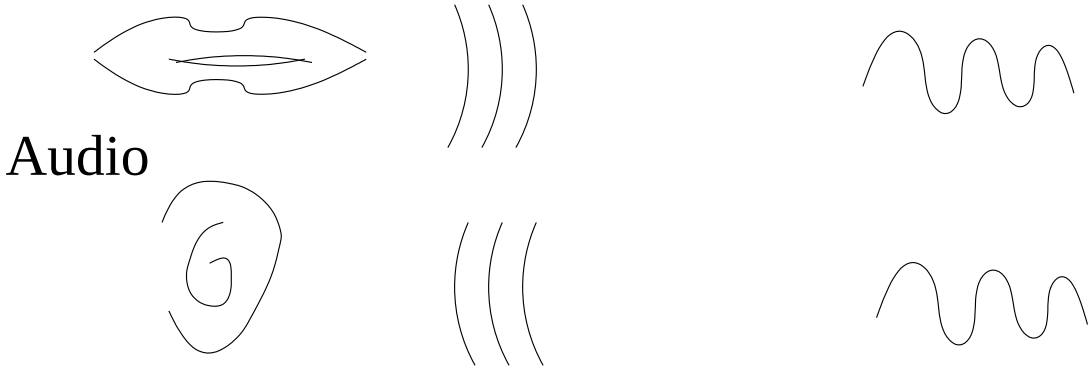
- 「情報」の「表現」方法
  - コンピュータは、「数値」の「計算」しかできない
  - 現実は、「様々な情報」の「処理」がしたい
    - この二つのギャップを埋めるのは何か？
- 情報を巡る、三つの「形態」
  - 情報：「現実」世界での「何か(例:音)」
    - Input(入力)：「情報」を「データ」に変換する
    - Output(出力)：「データ」を「情報」に変換する
    - I/O：ハードウェア ( 音:マイク / スピーカ ) によって実現
  - データ：「コンピュータ」内での「情報」の表現(例:正弦波)
    - Encode(符号化)：「データ」を「数値」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(Cluclus)される
    - これらは「可換」になっている

# 情報の処理の構造図




011010100101101

011011101000110

# 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(一対一でなう)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

- 配列
  - 同じデータが並んだ物を表現する仕組
    - 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

# 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)