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

2019/11/15 ソフトウェア概論

伝言

私語は慎むように !!

- 出席パスワード : 20191115
- 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
- VNC Server Address : VNCSERVER
  - Password : vnc-2018
- 廊下側の一列は遅刻者専用です(早く来た人は座らない)
- 講義開始前に済ませておく事
  - PC の電源を入れておく
  - ネットワークに接続しておく
  - 今日の資料に目を通しておく
- 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

前回(2019/11/08)の内容

- 前回(2019/11/08)の内容

本日(2019/11/15)の予定

- 本日(2019/11/15)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

今週 (2019/11/15) の課題

- 今週 (2019/11/15) の課題
  - 課題 今週-01:
    - ファイル名 : 今週-01-QQQQ.c (QQQQ は学生番号)
    - 内容 : 複素数型の四則
  - 課題 今週-02:
    - ファイル名 : 今週-02-QQQQ.c (QQQQ は学生番号)
    - 内容 : 二次元行列の和、差、積
  - 課題 今週-03:
    - ファイル名 : 今週-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)

先週 (2019/11/08) の課題

- □ 先週 (2019/11/08) の課題
  - ○ 課題 20191108-01:
    - ▹ ファイル名 : 20191108-01-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 20191108-02:
    - ▹ ファイル名 : 20191108-02-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 20191108-03:
    - ▹ ファイル名 : 20191108-03-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)

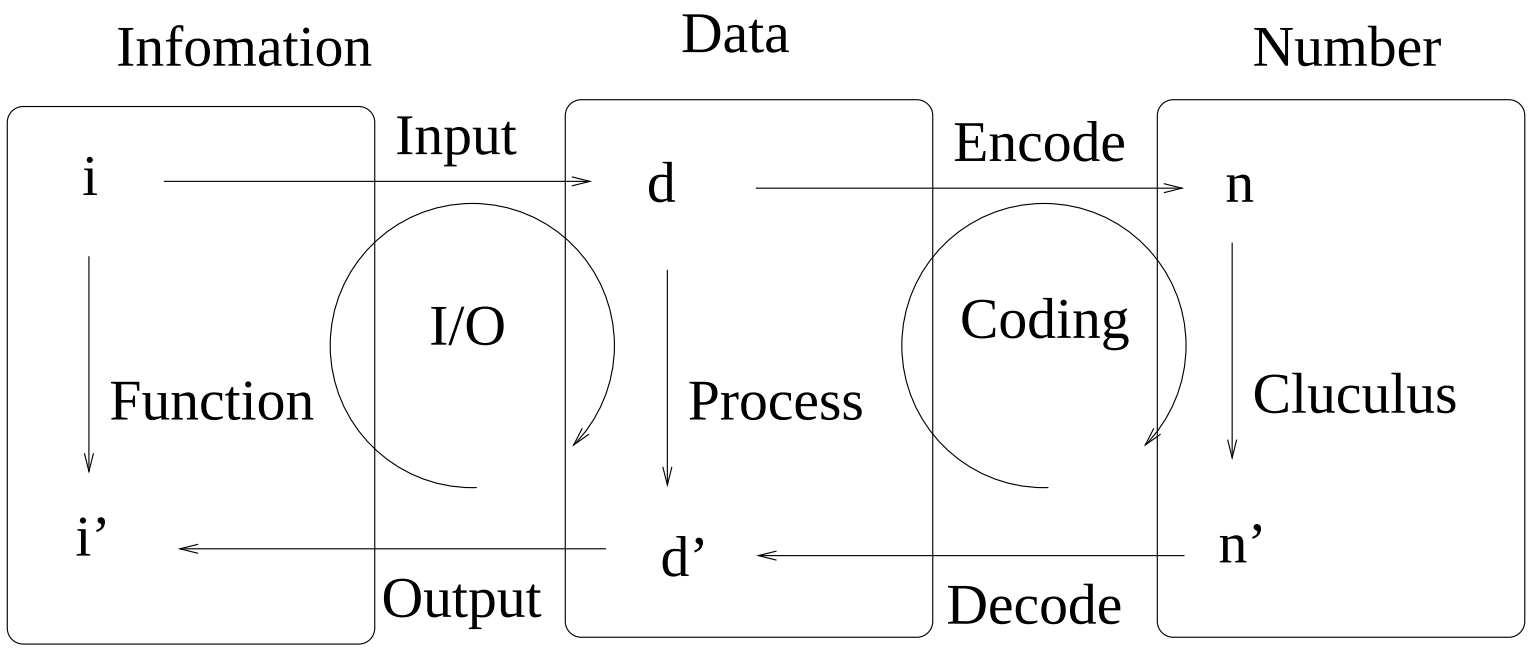
情報の表現 (再)

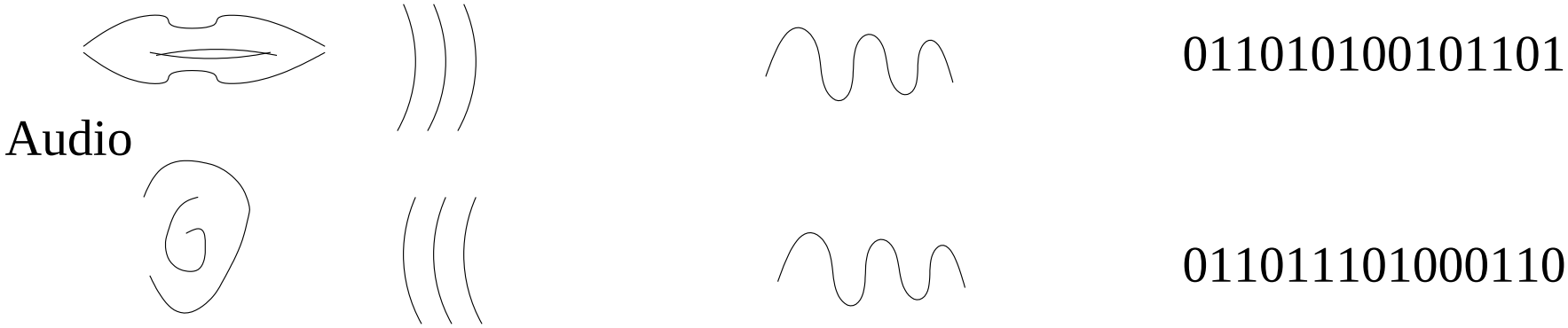
- 「情報」の「表現」方法
  - コンピュータは、「数値」の「計算」しかできない
  - 現実は、「様々な情報」の「処理」がしたい
    - この二つのギャップを埋めるのは何か？
- 情報を巡る、三つの「形態」
  - 情報：「現実」世界での「何か(例:音)」
    - Input(入力)：「情報」を「データ」に変換する
    - Output(出力)：「データ」を「情報」に変換する
    - I/O：ハードウェア (音:マイク / スピーカ) によって実現
  - データ：「コンピュータ」内での「情報」の表現(例:正弦波)
    - Encode(符号化)：「データ」を「数値」に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


Infomation
Data
Number
Input
Encode
i
d
n
I/O
Coding
Function
Process
Cluculus
i’
d’
n’
Output
Decode


Audio
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

- 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

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された物 (同じでないといけない)
    - ▷ <-> 構造体：異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」になる事が多い
    - ▷「繰返し」と「配列」は相性がよい (for 文)
- □ 配列使い方(そのニ)
  - ○「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