情報処理概論
(Basic Theory of Information Processing)

第１回：情報処理・プログラミング言語の役割

概要

- 本講義の目的
- 情報処理・プログラミング言語の役割
- プログラミングを学ぶための常識

1　はじめに

1.1　講義の目的

国際開発工学ってなんだろう

- 国際化の背景
  - **交通・運輸・通信の高速化・低価格化**
    東京-シンガポール，東京-新潟間のコンテナ運賃
  - 経済的・精神的に世界中の問題が影響する。
- より複雑な問題
  - 資源消費，排出物の増大 (**地球環境問題**)
    地球の自然に対して大きな力を持ちはじめた人類
    ⇒　国際的連携・規制が必要
  - **文明の衝突**
- **解決できなくては，人類の存亡にかかわる。**
- しかし，このような問題の解決は非常に困難
  - 大丈夫という人もいる。(うちの大学は大丈夫か？)
  - いまひとつはっきりしない。とりあえず後回し。(都合の悪いこと知らんぷり。)

Engineering Transformationが必要

- **総合診療科**
  - 医療の細分化が進み，どの科に行けば良いかわからない。
  - 患者にとっては，内科，外科は関係ない。病気が直れば良い。
  - 現状：はじめにかかる科によって患者の運命が異なるなんてものではなく，
    同じ病院の第１内科or第２内科でも異なる(同じ術式は取らない)。
- **Military Transformation**
  - **テロとの戦い。**(もう，通常の大戦争はほとんど生じない。)
    ラムズフェルド(元)国防長官：「空軍のやつら，まだ空中戦をする気でいやがる。」
  - 陸，海，空，海兵，宇宙，情報部門が一体となった，迅速な対処が必要。
- **工学も基本的な考え方から変える必要がある。⇒ 国際開発工学**
  - 化工，機械，電気・情報，土木から考えるのではなく，問題から考える。
    (どうしても，自分の分野で解決しようとしてしまう。)
  - 分野が分かれてきたのには歴史がある。だけど現在はそれが最適？重複
  - 個々の分野の発展は重要。**だけど，それで解決できるの？**
    ロボット，ハイブリッドカー，太陽光発電，HDTV

国際開発工学と情報処理概論

- 国際開発工学は，**国の壁，分野の壁を越えて (突き破って，下をくぐり抜けて)**，真に人類福祉向上に貢献するための学問である。
- 地球共和国 or 地球連邦ができるまでがんばって行きましょう。(地球共和国ができると「国際」が存在しなくなる。)

- 情報処理は人間に例えれば**脳・神経系**にあたる。
- 知識・知恵・ノウハウなどすべては情報である。
- 開発途上国と先進国を大きく分ける教育も情報に基づいている。
- 情報処理は理系・文系を問わず，すべての分野で必要である。
- 開発のためには，**情報の伝達・共有**はとても重要である。
- 「情報処理概論」は，国際開発工学の中でも非常に根本的な学問である。

社会における情報処理の重要性

- 数値計算：弾道計算，熱・強度計算，気象予測，回路シミュレーション
  一秒間に5.609兆回の浮動小数点演算 (TSUBAME2.5@東工大, 11位 (2014.6))
- データベース：成績，預金，住民基本台帳
- 画像・音声処理：電話，ディジタルテレビ，IP電話，ディジタルカメラ
- 制御：自動車，プリンター，腕時計

ところで情報とは？

- 事象に対する不確実さを減らすことができるもの。
  (情報理論)

情報処理とは？

- 情報を加工すること。

情報を処理できるもの

- 人間 (生物)
- コンピュータ ＝ 情報を加工する機械 (現代の情報処理に欠かせない)

プログラミングを学ぶ

- **プログラミング言語**：コンピュータが行うことを指示を記述するための言語。
- **プログラミング**：指示内容をプログラミング言語を使って記述すること。
- プログラミングができる⇒ コンピュータを使いこなすことができる。
- 今までは，人間の知恵が本などの自然言語で集積されてきた。
  ⇒ **あいまい。再現性がない。利用するために時間がかかる。**
- 人間の知恵がプログラムに集積される時代になってきている。
  - 各種パッケージソフトウエア。CAD，CAM，CAEソフトウエア。
    偽造問題で有名になった建物の構造計算もデータを計算機に入力するだけ。
  - 職人が持っていた各種の製造ノウハウが製造機械に秘められる。
    プログラムはそのコントロールを担当。
  - 製造が非常に難しかったＩＣでさえ製造装置さえ買えば，かなり容易に作れるようになった。⇒ 設備投資の先読みの方が利益を出すために重要になってしまった。
  - 開発費がかかるソフトウエアは，デファクトスタンダード化が進む。
    日本のソフトウエア輸入9,000億円，輸出90億円(鮭の輸出と同程度)
- 「読み書きそろばん」⇒「読み書き**プログラミング**」

1.2　本講義について

プログラミングは簡単

- **この講義程度の内容を知っている中学生はいくらでもいる。**
  高校生ならばプログラム作成でバイトしているプロもいる。
- パソコンと本があれば，こんな講義はいらないんじゃないかとも思う。
- まあ,「きっかけは『情報処理概論』」程度。それでも**もしわからなかったら**
  - 本に載っている通りにプログラミングして動かしてみる (**まずまねる**)。
    (プロも新しいライブラリーの利用の際には，まずサンプルプログラムを動かす。)
  - 変数の値を表示させ，その変化を追ってみる。
  - そのあとで，自分で少しずつプログラムを変えてみる。
  - 例題を参考に自分でプログラムを作ってみる。

授業・評価方法

- プログラミング言語として Java を扱う。
- グループワークでプログラムを作成する (諸般の都合により)。
- 教科書：明解 Java(入門編)，柴田望洋著，Softbank Creative, 2007
- 期末試験，中間試験，宿題，グループワークの発表

松井玲奈

- 元SKE48
- 商業高校の資格試験：
  情報処理検定試験
  プログラミング部門
  (COBOL)第１級
  に合格している。
- この講義の中間・期末試験と上記試験問題(Java)を回覧する。

まあ，同レベル

この講義がわからないようだったら

- 松井玲奈が現れたら土下座もの

世界の富豪

世界の富豪20位の中に6人がプログラマー出身 (2015フォーブス誌)

- マイクロソフトのビル・ゲイツ(約9.6兆円)
- グーグルのセルゲイ・ブリン(約3.5兆円)
- グーグルのラリー・ペイジ(約3.5兆円)
- フェイスブックのマーク・ザッカーバーグ(約4兆円)
- アマゾンのジェフ・ベゾス(約4.1兆円)
- オラクルのラリー・エリソン(約6.5兆円)

20人の中で1代で財を築いた人は13人

プログラマーだから技術的に適切(可能，冗長でない)で，人々を惹きつけるサービスを提供できた。

プログラムを学んでおくと，よいことがあるかも。

講義内容・日程(予定)

第1回　6/13　情報処理・プログラミング言語の役割
第2回　6/16　Java プログラムの作成 (eclipse のインストール)
第3回　6/20　変数，演算
第4回　6/23　変数と演算を応用した Java プログラムの作成
第5回　6/27　制御構造 (条件分岐，繰り返し)
第6回　6/30　制御構造を応用した Java プログラムの作成
第7回　 7/4　配列
第8回　 7/7　配列を応用した Java プログラムの作成
第9回　7/11　**中間試験**
第10回　7/14　メソッド，グループによるプログラム作成（GP）1
第11回　7/18　メソッドに関するプログラムの作成，GP2
第12回　7/21　クラス，GP3 (海の日)
第13回　7/25　クラスに関するプログラムの作成，GP4
第14回　7/28　FORTAN，GP5
第15回　 8/1　グループワークで作成したプログラムの発表
第16回　 8/4　**期末試験**

2　プログラミングを学ぶための常識

2.1　プログラミング言語

コンピュータが行うことを記述するための言語

プログラミングパラダイム

- **変数と代入，手続き：**FORTRAN, C
- **関数を使って記述 (代入がない)：**LISP, Scheme
- **１階の述語論理 (証明が実行)：**prolog
- **オブジェクト指向：変数と手続きをまとめたオブジェクト：**C++, java

高級言語と低級言語

- 高級言語：人間が通常使う自然言語に近い。
- 低級言語：コンピュータが直接に実行できる言語に近い。

低級 ← 機械語 < アセンブリ言語 < C < Java → 高級

主なプログラミング言語

- **Fortran**：FORmular TRANslator，科学技術計算，スーパーコンピュータで利用。
- **BASIC**：Fortranを簡単化，昔の小学生はこれでプログラミングを勉強した。
- **PACAL**：構造化言語，begin end，関数内関数定義
- **COBOL**：事務処理用，昔は60万人COBOLプログラマーと言っていた。
- **C**：PACALを簡単化。UNIXなどのOSを記述，高級アセンブラ(悪口)。
- **C++**：Cをオブジェクト指向言語にしたもの
- **Perl**：テキスト処理，WebのCGI (Common Gateway Interface)に使われる。
- **Ruby**：Perlと似ている。はじめからオブジェクト指向，日本人が作った。
- **Python**：テキスト処理，インデントで構文，Googleが使っている。
- **Smalltalk**：最初のオブジェクト指向言語，XeroxのStarで採用。GUIもStarが最初。
- **Java**：バーチャルマシン上で動く。C++の文法を整理したオブジェクト指向言語
- **JavaScript**：HTMLに埋めこんで使う。Javaとは無関係。(Java アプレットはJava)
- **LISP**：LISt Processor。データ構造の１つであるリストを処理する。人工知能のための関数型言語。Emacsで使われている。(Lots of Irriatating Superflaous Parenthesis)
- **Prolog**：人工知能のための論理指向の言語。通産省主導した1980年〜1989年の第５世代計算機の研究で使われた。

なぜJavaを勉強するのか

1. オブジェクト指向である (現在のプログラミングの主流)。
2. 新しい言語なので，表現が整理されていてわかりやすい。
3. 組込みシステムがJavaで動いている。
4. Webのアプリケーションも Javaが多い。
5. **開発環境が無料** → 自宅のパソコンで復習できる。

   `http://www.oracle.com/technetwork/java/index.html`
6. ライブラリーが充実している (無料のものも多い)。
7. 最新の統合開発環境 EclipseはJavaがメインターゲット
8. 近年では，ランタイムコンパイラーによって，処理速度が結構速い。
9. Javaがわかれば，他の言語もだいたいわかる (C，C++はすぐにわかる)。
10. (東工大の次に優れている理工系総合大学) MIT も Java を最初に教えている。
11. Javaアプレットなどでも使われている。
12. 現在，日本プログラマーの半分は，Javaを使っているとか。
13. 安全 (スタックオーバーフロー攻撃)

Fortranも少し勉強します。

2.2 コンピュータの動作の仕組

ノイマン型コンピュータ (von Neumann computer)

(ストアードプログラム型コンピュータ)

- 処理することを命令に分解して，その**命令列をメモリー**に記憶する。
- メモリーの命令列を基本的には逐次処理していく。
- 処理内容が変っても計算機の構造は変更しなくてもよい。
- 処理内容が同じならば，処理対象が変っても計算機の構造を変更しなくてもよい。
- 参考：
  - それ以前の計算機は，処理内容が変ると配線などを変更していた。
  - 理論的には，**万能チューリングマシン**がもとになっている。
    この辺を勉強したい場合は,「帰納的関数」を勉強すると良い。

余談：ノイマンさんについて。

- 原子爆弾が最も効果的になる爆発高度 (580 feet) を計算
  神風特別攻撃隊に対する最適な配置を計算 (オペレーションズリサーチ)
- 京都への原爆投下を進言 (標的委員会)
- ソ連への核攻撃を提言

2進数

- 東工大生ならば知っていると思うけれど...
- 各桁を0と1だけで表わす。
- ONとOFFだけで表現できるので，**電気的な実現が容易**である。
- 2進数の $b_n b_{n-1} \cdots b_1 b_0$ (b_i は0 or 1) を，10進数に直すには，次の計算を行なう。

$$b_n 2^n + b_{n-1} 2^{n-1} + \cdots + b_1 2^1 + b_0 2^0$$

- 10進数 x を2進数に変換する。そのために，まず，$x_0 = x$ とおく。そして，x_i を2で割った余りを b_i，その商を x_{i+1} とする。これを繰り返し，x_n の商がではじめて0になったとすれば，$b_n b_{n-1} \cdots b_1 b_0$ が求める2進数である。

- 後日，この変換を行なうプログラムを書いてもらう予定。

10進数	2進数
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111
16	10000
100	1100100
1000	1111101000

不用とは思うが，例を示す。
2進数の101011を10進数にする。

$$
\begin{aligned}
& 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\
= \ & 32 + 8 + 2 + 1 \\
= \ & 43
\end{aligned}
$$

となるので，求める10進数は43となる。

10進数の43を2進数にする。

$$
\begin{aligned}
43 \div 2 &= 21 \ \cdots \ 1 \\
21 \div 2 &= 10 \ \cdots \ 1 \\
10 \div 2 &= 5 \ \cdots \ 0 \\
5 \div 2 &= 2 \ \cdots \ 1 \\
2 \div 2 &= 1 \ \cdots \ 0 \\
1 \div 2 &= 0 \ \cdots \ 1
\end{aligned}
$$

となるので，求める2進数は101011になる。

16進数 (p.147)

- 2進数で表わすと長くて見難い(**人間が**)。
- 10進数だと変換しなくてはいけない。
- 2進数を下の桁から4桁毎に区切り，0〜9とA〜Fまでを使って表わす。
- 2進数の101011を例にすると，

$$101011 = \boxed{10}\ \boxed{1011} = 2\mathrm{B\ H}$$

最後のHで16進数であることを表す。
(hexadecimal)

- もっと長くても大丈夫。

$$\begin{aligned} & 110010101011010111100011 \\ = \ & \boxed{110}\boxed{0101}\boxed{0101}\boxed{1010}\boxed{1110}\boxed{0011} \\ = \ & 655\mathrm{AE3\ H} \end{aligned}$$

となる。

16進数	2進数
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

全てのデータは数値にできる

- ゲーデル数化 (整数に対する論理式を整数で表すこと) がもと ???
- 文字：文字コードによって，1つの文字と2進数が対応づけられている。
- 文章：文字コードをならべて見れば，数となる。
- 音声：**サンプリング定理**を使う。
  - サンプリング定理：原信号の周波数が限られている場合，一定間隔でサンプルした値だけがあれば，信号を復原することができる。

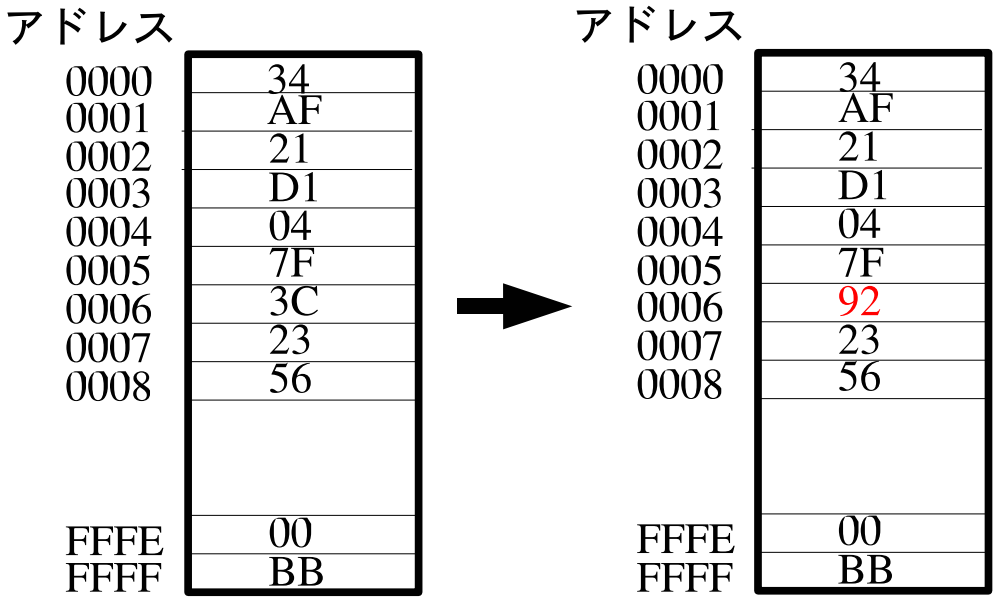
  従って，マイクロフォンから得られる電圧値を一定間隔の時刻で取り出して，列べれば，音声が数によって表現できる。(**数を列べたものも1つの数にできる。**)
- 静止画像：１枚の画像を縦横の分割してその区間の赤，緑，青の光の強さを数として列べれば良い。(分割されたものを**画素**と呼ぶ。)
  DVDは，横720×縦480画素，ハイビジョンやBlu-rayは，横1920×縦1080画素
- 動画像：静止画像を列べる。
  - 動画像に含まれる1枚の静止画像を**フレーム**と呼ぶ
    映画：24フレーム/秒
    テレビは，30 or 60 フレーム/秒

静止画像の例

```
P3
# CREATOR: The GIMP’s PNM Filter Version 1.0
512 512
255
226
137
125
226
137
125
223
137
133
223
136
128
226
```

2.3 メモリー

- **情報を記憶するところ。**(記憶できなければ，処理もできない。)
- 1 **bit**：情報量の基本単位で，0 or 1 のような2者択一の情報を表わすことができる。
- 1 **byte** = 8 bit
- 1 **word** CPUが基本的に扱う情報量(現在のパソコンならば，32 bitまたは64 bit)。



- 一般的にメモリーには，1 byteごとに番値(**アドレス**)が付けられている。
- メモリーのデータにアクセスするときは，必ず**アドレスを指定**する。その上で，
  - そのアドレスのデータを取得
  - そのアドレスにデータを格納

  することができる。逆にそれしかできない。テーブルやリストも直接は記憶することができない。**データ構造やクラス**が重要
- **内容を指定してデータを取ってくることはできない。**(連想メモリー)

2.4 CPU

- CPU : Central Processing Unit
  コンピュータの中心：情報処理を実際に行なうもの。
- CPU の代表例：

  Core 2: Intel 社，パソコンの CPU の標準

  Athlon64： AMD 社，Core 2 と命令が互換のプロセッサで，主にパソコン用

  MIPS： スタンフォード大学が開発した RISC-1 から発展した。効率良いプログラム実行が可能である。PlayStation，PlayStation 2 に使われている。

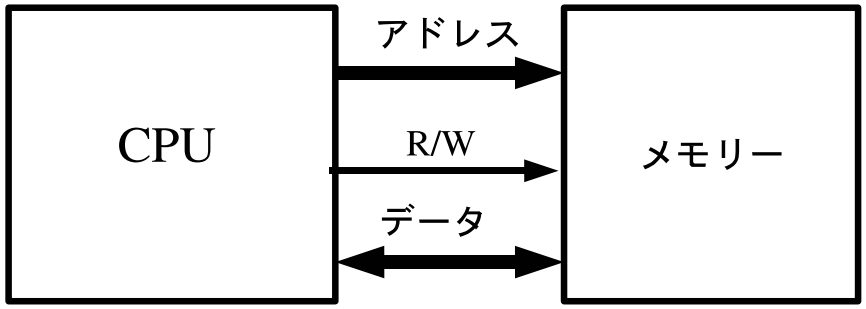
  PowerPC： IBM と Motrola の共同開発。マッキントッシュに登載されていた。また，プリンター，コピー機などにも登載されている。Wii，XBox 360，PlayStation 3 には，この発展型が登載されている。

  SH： 日立製作所。2 オペランド命令セットの RISC アーキテクチャがユニークだった。カーナビなどで使われている。携帯のアプリケーションプロセッサとしてがんばろうとしているが。。。

  ARM： 組み込み用プロセッサの代表。携帯端末などに使われる。

- CPU への命令は,「**機械語**」によって書かれている。
  機械語は，プロセッサが直接理解することができる，0 と 1 の列である。

2.5 CPUとメモリー



- CPUとメモリーは接続され，情報処理の大部分はこの両者が共同して行なう。

  アドレス： CPUがメモリーに対してアドレスを指定する。

  R/W： CPUがメモリーに対して，動作が「読み出し」か「書き込み」かを指定する。

  データ： CPUとメモリーの間でデータをやり取りする (**双方向**)。

- メモリーには，次のものが格納されている。

  プログラム： CPUがデータを処理するための命令の列び。

  データ： CPUが処理するデータ。

- 命令列は，アドレスの**昇順**に格納する。
  ⇒ CPUは基本的には，アドレスの順番に命令を取ってくる。

- データは必要に応じて，読み出され，書き込まれる。

2.6 機械語

- 機械語は単純。
  CPU は単純な命令を高速時実行し，それを組み合わせて複雑な処理を行なう。
  - 内部レジスタや指定したアドレスのメモリーのデータを使って演算し，その結果を内部レジスタや指定したアドレスのメモリーに書きこむ。
    (演算を行なわずに，データ移動だけの場合もある)
  - 命令を読みこむアドレスを変更する。
  - 指定したアドレスのメモリーのデータや内部レジスタの値や，演算結果などの情況に応じて，命令を読みこむアドレスを変更する。

  低級言語：
  - 機械にはわかりやすい。(機械語が最も低級な言語) 人間にはわかりにくい。
  - 1つ1つの命令がプリミティブ。

  高級言語：
  - 機械にはわかりにくい。人間にはわかりやすい。
  - 1つ1つの命令がまとまった意味を持っている。

プログラム例

機械語(最も低級)

```
01000000010000110000010000000001
01000000010000110011010100100001
10101100001000100011010010000000
10100100001000100000000000000101
10101100111000000000000000011001
```

アセンブリ言語(機械語と1対1対応)

```
ADD  r1 r2 r3
SUBI r1 r2 0001H
LD   r1 [r2 + 5H]
ST   r1 [r2 + 3480H]
JNZ  r7 0019H
```

Java

```
r1 = r2 + r3;
r1 = r2 - 1;
r1 = a[r2];
rray[r2] = r1;
if ( r7 == 0) {
  命令列
}
```

- 機械語は何を意味しているかわかり難い(昔の人は，一目見てわかった)。
- アセンブリ言語で書けば，各命令の意味はわかるが，プログラムが長くなると大変になる。
- Java で書いた方がが人間にはわかりやすい。大きなプログラムの場合，機能ごとにオブジェクトで分割する。

コンパイラ，インタープリタ

- JavaやCなどの高級言語は，CPUが直接実行することができない。
- **コンパイラー** ：高級言語で書かれたプログラムを機械語に翻訳する。
  FORTRAN, COBOL, C, C++などで使われている。
- **インタープリター**：高級言語の文を1つ読みこんで解釈し実行することをくり返す。
  (高級言語から中間言語に落してから，1つ1つを実行する場合もある。)
- Javaの場合は少し複雑。
  - コンパイラーがJavaで書かれたプログラムを，バーチャルマシン(**仮想的なコンピュータ**)の機械語に直す。(バイドコード)
  - 実行するCPUの機械語で書かれたバーチャルマシンのシミュレータが，インタープリターのようにして，バーチャルマシンの機械語を実行する。
  - 近年では高速化のために，実行時に，バーチャルマシンの機械語を，実行するCPUの機械語に変換して実行するものもある(**ランタイムコンパイラー**)。
  - Javaのコンパイルおよび実行コマンド

```
javac プログラム名.java          (コンパイル)

java  クラス名                   (実行)
```

2.7　まとめ

- 講義の目的と内容(プログラミングを学ぶ)
- プログラミングを学ぶための常識
- なぜJavaを勉強するか
- コンピュータの動作の仕組(ノイマン型コンピュータ)
- 2進数，16進数
- メモリー，CPU
- 機械語，高級言語，低級言語

今日の最後に

- 次回はeclipseをインストールして，Javaを実際に動かす。

  `http://www.eclipse.org/`

  `http://mergedoc.sourceforge.jp/`（日本語version）
- **質問はいつでも受けつける。**
- **中学生でもわかる**ような内容なので，ゆっくりと**自分の頭**で考えてみること。
- プログラミングは論理的思考能力を鍛えるためにも良い。
- **次回は計算機室に集合！**