# コンピュータ概論B
## ー ソフトウェアを中心に ー

#10 プログラミング・言語

京都産業大学
安田豊

# 自然言語と人工言語

- 自然言語
  - 人間が日常生活で用いている言語
  - 日本語、英語、etc..
- 人工言語
  - 人間が意図的に作り出した言語
  - エスペラント (1887年 L.L. Zamenhof , ポーランド)
  - 全てのプログラミング言語
- 言語の起源、獲得、自然さ
  - 言語学の分野でも研究テーマ
  - ここでは境界については気にしない

# 人工言語

- エスペラント
  - 延々と続く普及の努力、しかし普及せず
- 日本語ローマ字表記運動
  - 学習が難しい、西欧並みの文化水準へ
  - 明治、戦後などたびたび起きる
  - 何故成功しなかったか？
- 言語学でも（おそらくは）難しいテーマ
  - しかしプログラミング言語は普通に使われている
  - 教科書 pp.108

# プログラミング言語

- コンピュータと対話するために利用
  - 手順指示書のようなもの
- 例えばどのようなものか？

## C 言語でのプログラム例

```
main(argc, argv)
int    argc;
char   *argv[];
{
      FILE   *ifp,*ofp;
      int    i,j;
      char   ifile[256],ofile[256],dt[4096];

      if(argc == 1) {
            printf("Usage: %s infile
outfile¥n",argv[0]);
            exit(1);
      };
      strcpy(ifile,argv[1]);
      strcpy(ofile,argv[2]);

      printf("I:%s O:%s¥n",ifile,ofile);

      ifp=fopen(ifile,"rb");
      if(ifp == 0) {
            printf("Cannot open input file
%s.¥n",ifile);
            return(-1);
      };

      ofp=fopen(ofile,"wb");
      if(ofp == 0) {
            printf("Cannot open output file
%s.¥n",ofile);
            fclose(ifp);
            return(-1);
      };

      while(1) {
            fgets(dt, 255, ifp);
            if( dt[0] == 10 || ( dt[0] == 13 && dt[1] ==
10 ) ) break;
      };
      while( !feof(ifp) ) {
            l=fread(dt, sizeof(char), 256, ifp);
            fwrite(dt, sizeof(char), l, ofp);
      };

      fclose(ifp);
      fclose(ofp);
}
```

# 機械語

- 教科書 pp.109
- 機械語 (machine language)
  - コンピュータ (CPU) にとっての Native Language
  - 汎用性無し (CPU 依存、互換性無し)
  - CPU にとっては直接的でも
  - 人間には解釈が困難

例： Sparc 用バイナリ

```
7400 2e64 796e 616d
6963 002e 706c 7400
2e65 785f 7368 6172
6564 002e 6461 7461
002e 6461 7461 3100
2e62 7373 002e 7379
6d74 6162 002e 7374
7274 6162 002e 7374
```

## プログラミング言語

- 機械語を理解する
  - それでも古典的には機械語で書いた
  - 直接数字が何を意味するかを記憶している
- アセンブリ言語
  - さすがに機械語は辛いので、これをわかりやすい略語で記述
  - 最初のプログラミング言語
  - 教科書 pp.113

```
!  9   j=0;

       mov    0,%l0
       st     %l0,[%fp-12]

!  10  for(i=0;i<l;i++) {

       mov    0,%l0
       st     %l0,[%fp-8]
       ld     [%fp-8],%l1
       ld     [%fp-16],%l0
       cmp    %l1,%l0
       bge    .L18
       nop
```

## プログラミング言語

- アセンブリ言語
  - 機械語へ変換する必要がある
  - ほぼ一対一対応
  - 結果的に CPU 依存、ハードウェア依存
  - それでも機械語よりは楽に書ける
  - バグも減る（勘違いが減る）
- アセンブリ言語から機械語への変換
  - 略語を単語置換で自動翻訳するようなもの
  - これがプログラミング言語の本質

## プログラミング言語

- プログラミング言語とは？
  - 人間の思考をCPUへの作業指示に変えるための表現言語
  - 思考を助け（拡張し）
  - 楽に表現でき
  - コンピュータを決定的に動作させるもの。
  - その表現作業を楽にすることも重要
- 教科書 pp.109 に良い図が

## プログラミング言語の特徴

- 教科書 pp.110
- 決定的動作のために
  (解釈系による違いを防ぐ)
  - 文脈・前後関係なしに一義的に決まる
  - 一点一字の間違いも許されない
  - 限定的な語彙（C言語では予約語は 32）
  - 簡単な文法

## 具体例：アセンブリ言語

- 教科書 pp.112
  - 低水準プログラミング言語の代表
  - 自然言語に近いほど「高水準」と考える
  - 歴史的には低水準から高水準へと進化
- 特徴
  - CPU 依存（移植性がない）CPU の機能を直接指定
  - 機械語とほぼ一対一
  - ニーモニックコード (mnemonic : 記憶を助ける)
  - 可読性が悪く、保守性が悪い
  - 作成が難しく、生産性が低い
  - 機械語に変換するソフトをアセンブラと呼ぶ

## 具体例：C

- 特徴
  - (アセンブラよりは)高水準な言語
  - CPU 依存なし (CPU 命令は直接指定できない)
  - メーカー、OS が変わっても再利用できる
  - 移植性が高い
  - ニーモニックより更に抽象度が高く、自然言語に近い（プログラマにやさしい）語彙と文法
  - コンパイラ(Compiler)と呼ばれるソフトを利用してアセンブリ言語または機械語に変換
  - 可読性が高く、保守しやすい
  - プログラミングが容易で生産性が高い

## 抽象度

- 二つの整数を加算する場合

```
47 F0 E0 57
F0 E4 E7 F0
E8
```

- 機械語
  - 一時使用するレジスタ番号、メモリアドレスを指定
- アセンブリ言語
  - レジスタ番号などはまだ残る

```
LD G1, F0E0
ADD G1, F0E4
ST G1, F0E8
```

- 高水準言語
  - ハードウェアの中身から独立している
  - 抽象度が高まった、移植性が高まった

```
a=b+c
```

## プログラミングとは

- 教科書では料理の例
  - 悪くはないが、、、、
- 音楽の例
  - プログラムとコンピュータは楽譜と楽器の関係に似る
  - プログラマは音楽家に似る
  - 音楽を作ったとしても、楽譜の書き方が分からなければ演奏家に伝えられない
  - やりたいことがあってもプログラムの書き方が分からないとコンピュータに指示できない
  - 音楽を楽譜に移す作業＝目的をコンピュータの動作手順に変える作業＝プログラミング

## プログラミングとは

- 自己表現の一つの手段である
  - やりたいこと（その中でコンピュータにやらせたいことを）を表現する
- 利用者の立場ではどうか？
  - アプリケーションをソフトを使うだけ
  - 音楽を聞くだけに似る
  - 悪くはない
  - 音楽の価値・意味の半分しか手にしていない

## プログラミングとは？

- 音楽と同じように
  - アマチュアが作らないわけではない
  - プロだから作るわけでもない
  - 作りたいから作る、作る必要があるから作る
- ただし
  - 現時点では業務システムを作るようなプロ向けの言語と開発システムに焦点が当たっているのは事実
  - これからはコンピュータを使って自己能力を拡張するために学ぶ機会が増えるだろう
  - それに適した言語・環境も増えてくるだろう

## 人間と機械のあいだで

- プログラミング言語の条件 (教科書pp.116)
  - 処理の考え方を計算機に処理可能な手順に変え、
  - 具体的な手順指示を文法に従って書き、
  - 機械語まで「噛み砕いて」はじめて実行される
- 人間と機械の間で、
  - 人間にわかりやすく書かれなければならない
    →論理明晰で、可読性にすぐれた記述が重要
  - 機械にわかりやすく書かれなければならない
    →厳しい文法に則って書く
  - プログラマは両者のインタフェイスとなっている