プログラミング基礎演習I
(第１・２回)

伊原彰紀

Akinori Ihara

akinori-i@is.naist.jp

（B306室）

履修条件･成績評価

- プログラミング基礎演習Ⅰ、Ⅱは、プログラミング初心者を対象としたものです。
- 情報系学科出身者は、修了の単位として認められません.
- プログラミング経験がある程度ある学生は，ソフトウェア開発演習Ⅰ、Ⅱを受講してください.

- 成績評価
  - 演習(50%)と課題レポート(50%)に基づいて単位認定を行う。
    - 演習の評価は、講義中に演習課題のプログラム実行結果を見せることで成績評価とする。
    - レポートはメールにて提出する。

スケジュール

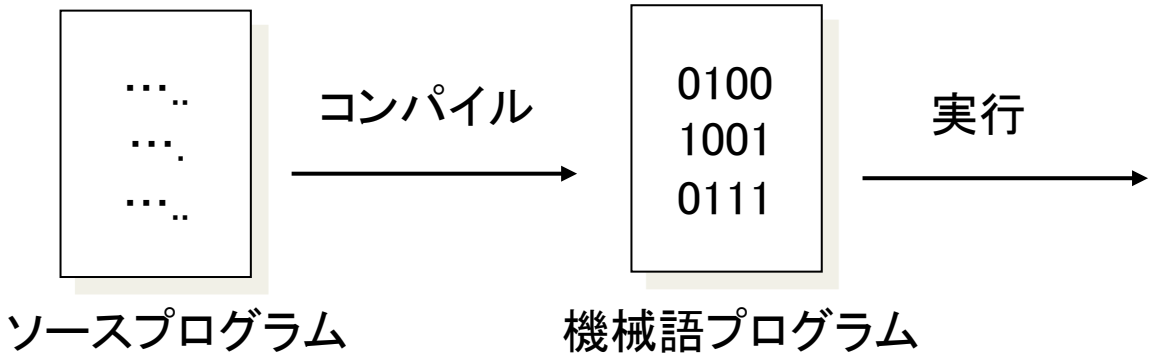
- 4/ 7(木) A207 第 1・ 2回:C言語とは. コンパイルと実行（伊原）
- 4/14(木) A207 第 3・ 4回:データ型と演算子（伊原）
- 4/21(木) A207 第 5・ 6回:制御の流れ（伊原）
- 4/28(木) A207 第 7・ 8回:関数（伊原）
- 5/12(木) A207 第 9・10回:配列（畑）
- 5/19(木) A207 第11・12回:演習（松本）
- 5/26(木) A207 第13・14回:ポインタの初歩（畑）
- 6/2 (木) A207 第15・16回:演習（畑）

C言語の特徴

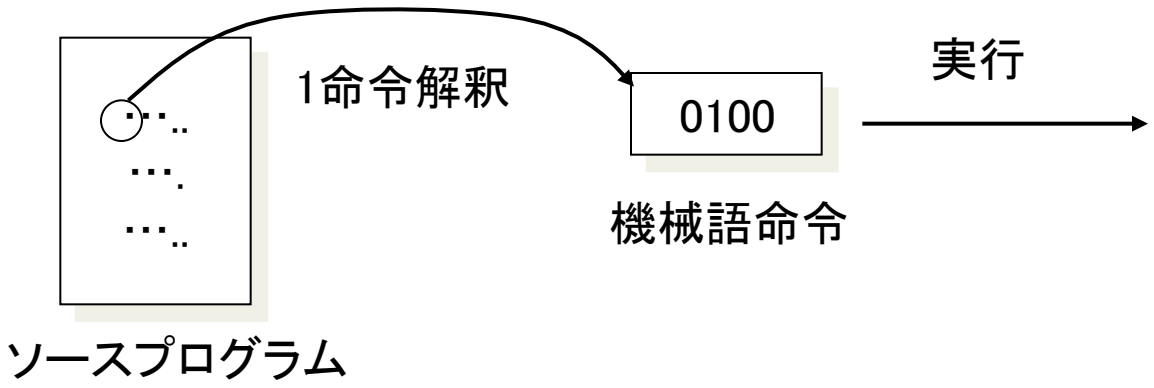
- 幅広い分野で使われている。
  - UNIXシステム，パソコン，組み込みソフトウェアなど
  - 手続き型の言語
    - 他に，オブジェクト指向型，関数型などがある。
  - 移植性が高い（Javaほどではないが）
    - プラットフォーム依存の言語仕様がない。
  - ハードウェアよりの低水準な処理も記述可能
    - メモリやCPUなどを直接利用できる。
    - インラインアセンブラが使用可能。
  - 処理が高速
  - 安全性が低い
    - 暴走の恐れがある。
  - コンパイラ言語である。

コンパイラ言語とインタプリタ言語

- コンパイラ言語（C言語，COBOL…．）



- インタプリタ言語（BASIC, JavaScript….）



C言語と他の言語の比較

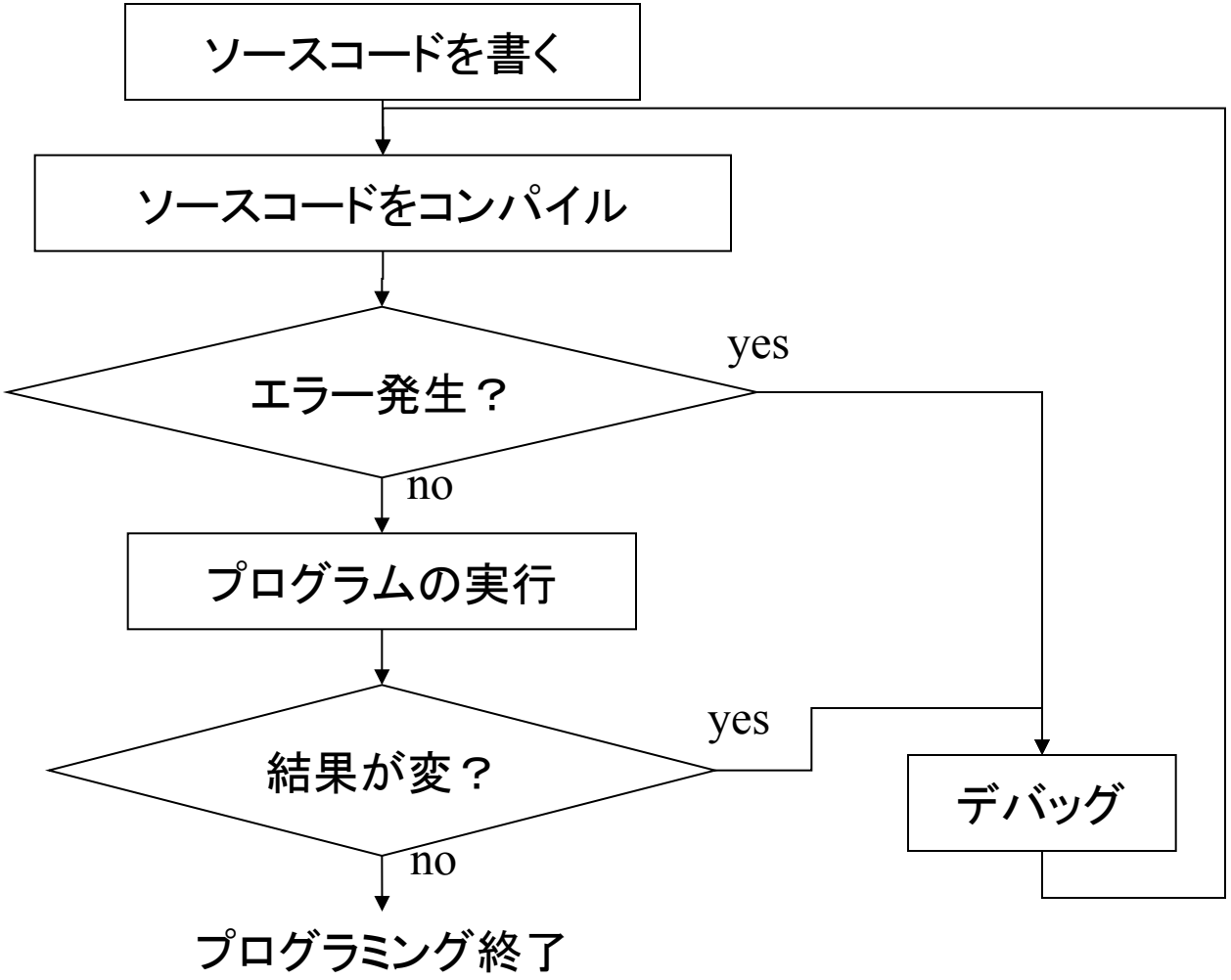
- COBOL，PL/I　　事務計算，銀行のオンラインシステム
- FORTRAN　　科学技術計算
- Pascal，LISP　　教育，研究，アルゴリズムの記述
- C　　ワープロ，ゲーム，OS，ネットワーク，
  ハードウェア制御，科学技術計算

※C言語は、事務計算には向かない。
内部的に2進数で表現しているため、10進演算にやや弱い。
例：10進数の0.1 = 2進数の0.0001100110011001100…（循環小数）

C言語の歴史

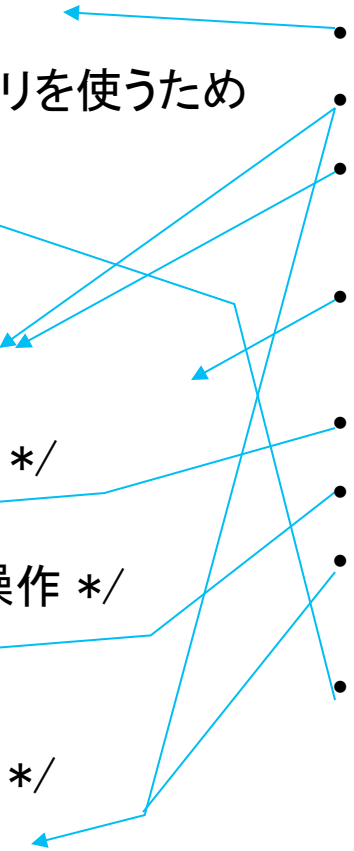
- 1960　Algol 60（国際委員会）
- 1963　CPL（ケンブリッジ大学、ロンドン大学）
- 1967　BCPL（ケンブリッジ大学 Martin Richards）
- 1970　B（ベル研究所 Ken Thompson）
- 1972　C（ベル研究所 Dennis Ritchie）
- 1983　ANSI C（米国規格協会）
- 1983　C++（AT&Tベル研究所
  Bjarne Stroustrup）

C言語プログラミングの進め方



Cプログラムの基本

```
/* プリプロセッサ命令 */
/* printf関数があるライブラリを使うため
   */
#include <stdio.h>

/* メイン関数の宣言 */
main() {
    /* 整数型の変数の宣言 */
    int i, j;
    /* 演算子により変数を操作 */
    i = 3;
    j = i * i;
    /* printf関数の呼び出し */
    printf("j = %d\n", j);
}
```



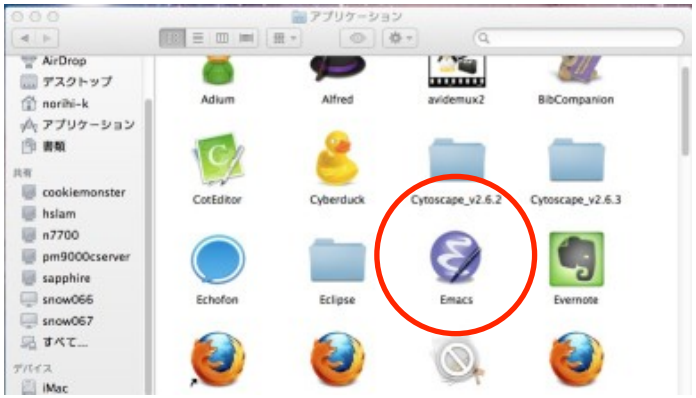
- コメントは/* … */
- 基本構成要素は関数
- プログラムは必ずmain関数から始まる
- 複数の文を{ … }で囲むと一つのブロックとなる
- データには定数と変数がある
- 定数や変数の値は演算子で操作
- 便利な関数をまとめたものがライブラリ
- ライブラリを使うためのファイルをプリプロセッサにより取り込む

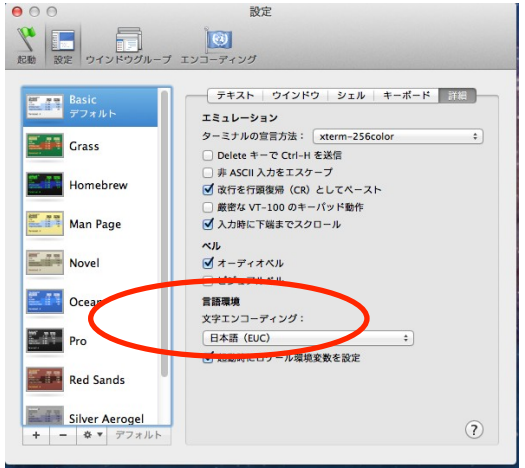
よくあるコンパイルエラーの原因

- 全角スペースを使っている。
- 中括弧 { や } が全角 ｛ や ｝ になっている。
- 文の後ろのセミコロン ; が抜けている。
- 文字列 “……” の閉じ忘れ。
- コメント文 /＊ …… ＊/ の閉じ忘れ。
- “X” と ‘X’ の間違い。

Emacsとターミナル

- Emacsの起動
  - 左下の「ファインダー」
    →すべて
    →Application →Emacs
  - （環境によってはNetwork Applicationsの下にあることも）
- ターミナルの起動
  - 右下の「アプリケーション」
    →ユーティリティ
    →ターミナル
- ターミナルの日本語設定
  - 左上の「ターミナル」
    →環境設定→詳細
    →文字エンコーディング
    - UTF-8に
- Emacsの日本語設定
  - Options→Mule(Multilingual Environment)
    →Set Language Environment→UTF-8





ターミナルのコマンド（1）

- ディレクトリ（フォルダ）に関するコマンド：
  - pwd　　… 現在自分が居るディレクトリ名を表示させる。
  - ls　　… 現在自分が居るディレクトリの中身を表示させる。

  - cd　　… ホームディレクトリに移動する。
  - cd XXX　　… XXXというディレクトリに移動する。
  - mkdir XXX　　… ディレクトリXXXを新たに作る。

ターミナルのコマンド（2）

- ファイルに関するコマンド:
  - cat XXX　　… ファイルXXXの中身を表示させる。
  - less XXX … ファイルXXXの中身を1ページずつ表示させる。
  - mv XXX YYY　　… ファイルXXXの名前をYYYに変える。
  - exit で終了。

テキストエディタ Emacs

- 適当なエディタを使用：Emacs 等
- Emacsの基本操作例：
  - 立ち上げ：　**emacs↲**
  - ファイル読み込み：　**Ctrl-x　f　filename**
  - 保存：　**Ctrl-x　Ctrl-s**
  - 　　（ファイル名を聞いてくるので指定する）
  - 取り消し：　**Ctrl-g**
  - アンドゥ：　**Ctrl-x u**
  - 終了：　**Ctrl-x　Ctrl-c**
  - 漢字変換：　**Ctrl-**

日本語入力

- **[Command]+[Space]**

コンパイル

- C言語コンパイラコマンド **cc**

例1. **cc file.c** ↲

（a.out という名の実行ファイルを自動生成）

例2. **cc -o jikkou file.c** ↲

（jikkou という名の実行ファイルを生成）

make xxx

- xxx.cをコンパイルしてxxxという実行ファイルを作る。

プログラムを書いてみる（準備）

- 作業場所を作る

mkdir enshu↵

（enshuという名のディレクトリを作成）

ls↵

（ディレクトリが出来ているか確認）

cd enshu↵

（enshu ディレクトリへ移動）

例題1：プログラムを書いてみる（1）

- test1.c という名のソースコードを書いてみる。
  - emacs test1.c ↓
  - （エディタ emacsを使用してtest1.cを作成）
  - 以下のように書いてみる（例題1）:

```
#include <stdio.h>

main()
{
    printf("Hello NAIST!!");
}
```

定型文
(標準ライブラリの指定)

画面に文字出力

例題1：コンパイルを行う(1)

- test1 という実行形式を生成する。

  `cc -o test1 test1.c`

  (もしくは **`make test1`**)↲

  `./test1`↲ と入力すると何が起きるか？

例題2:プログラムを書いてみる(2)

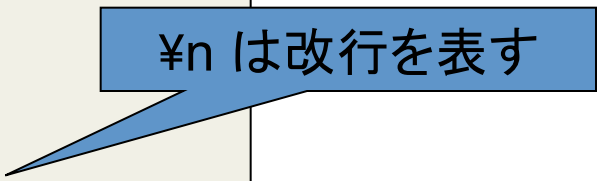
- test2.c という名のソースコードを書いてみる。
  - 以下のように間違いを書いてみる (例題2)。
  - コンパイルするとどのようなメッセージがでるか？

```
#include <stdio.h>

main()
{
   printf("Hello NAIST!!")
}
```

例題3：プログラムを書いてみる(3)

- test3.c
  - test2.c を以下のように改造 (例題3)。
  - コンパイル・実行し, 前との差を確認せよ。

```
#include <stdio.h>

main()
{
    printf("Hello NAIST!!\n");
}
```



- Macでは, \の代わりにバックスラッシュ＼を使う。

例題4：プログラムを書いてみる(4)

- ソースコードには極力コメントを入れる
  - ソースコードだけではただの呪文の羅列
  - 何を行っているか自分・他人にわかりやすくする

```
#include <stdio.h>

main()
{
 /* Display a greeting message */
 printf("Hello NAIST!!\n");
}
```

/* */ で囲まれた部分がコメント.
実行には影響しない.

- コンパイル・実行し、前と違いがないことを確認せよ。

演習問題

演習1-1 自分の学籍番号と名前を表示するプログラムを作成せよ。ただし、学籍番号と名前は別の行に表示すること。

表示例

```
0123456
奈良先　太郎
```

演習1-2 顔の絵を文字で作成せよ。

表示例

```
---------
| ^ ^ |
|  -  |
---------
```

文字化けする人は

- ターミナルの文字コード設定をUTF-8ではなく**EUC**にしてみてください.

例題5：変数を使う（整数）

```
#include <stdio.h>
main()
{
   int year;
   year = 2014;
   printf("This year is %d\n", year);
}
```

整数を扱う型宣言

代入演算

整数を表示する

%dで表示する
変数を指定

例題６：変数を使う（小数）

```
#include <stdio.h>
main()
{
   float height;
   height = 170.5;
   printf(“I am %f cm tall\n", height);
}
```

小数を扱う型宣言

代入演算

例題7：足し算を行う

```
#include<stdio.h>

main()
{
      int i, j, k;

      i = 1;
      j = 2;
      k = i + j;
      printf(“k = %d\n”, k);
}
```

3つの変数i,j,kの型宣言

iとjの足し算結果をkに代入する.

代入と演算

- 変数への数の代入(ステップ2)
  (変数名) ＝ （算術式);
- 四則演算

  ｋ ＝ ｉ ＋ ｊ ； 加算

  ｋ ＝ ｉ － ｊ ； 減算

  ｋ ＝ ｉ ＊ ｊ ； 乗算

  ｋ ＝ ｉ ／ ｊ ； 除算

  ｋ ＝ ｉ ％ ｊ ； 剰余

予約語

- 次の識別子は予約語（キーワード）として使用されるため、別の用途 (変数名・関数名) に使用してはならない。

auto	break	case	char	const
continue	default	do	double	else
enum	extern	float	goto	if
int	long	return	short	signed
sizeof	static	struct	switch	typedef
union	unsigned	void	volatile	while

演習問題

演習1-3 int型(整数)の変数を用いて、自分の誕生日を表示するプログラムを作成せよ.

表示例

```
私の誕生日は4月19日です。
```

演習1-4 float型(浮動小数点数)の変数を用いて、何かの数値を出力せよ.

表示例

```
円周率は3.14159265
```

演習問題

演習1-5 float型の変数xとyに，それぞれ1.5と2.3を代入し，その和と積を表示せよ.

表示例

```
和は3.80000です.
積は3.45000です.
```

演習1-6 int型の変数xとyに，それぞれ15と7を代入し，x÷yの商と剰余を表示せよ.

表示例

```
商は2です.
剰余は1です.
```