# ソフトウェア概論 A/B

-- 変数宣言 / 代入 / 標準 I/O ライブラリ --

数学科 栗野 俊一 / 渡辺 俊一 ( TA: 栗原 望 / 小嶋 仁子 [M2] )

2018/10/19 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 : 20181019
- 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
- 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
  - 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

# 前回(2018/10/12)の内容

- 前回(2018/10/12)の内容
  - 設計
    - 数値的解法：「問題の答え(数値)」を「近似的」に求める手法(誤差を含む)
  - 数値計算の初歩
    - 方程式の解：二分探査/ニュートン法
    - 数値積分：リーマンの公式/台形公式/モンテ=カルロ法

# お知らせ

- 本日の予定
  - 代入と制御構造
    - 代入 / 局所変数の宣言 / while 構文
  - 標準 I/O ライブラリ
    - printf/scanf, fprint/fscanf, fopen/fclose
- 本日の目標
  - 演習
    - 課題の提出

# 今週 (2018/10/19) の課題

- 今週 (2018/10/19) の課題
  - 課題 20181019-01:
    - ファイル名 : 20181019-01-QQQQ.c (QQQQ は学生番号)
    - 内容 : printf による書式指定出力
  - 課題 20181019-02:
    - ファイル名 : 20181019-02-QQQQ.c (QQQQ は学生番号)
    - 内容 : scanf による書式指定入力
  - 課題 20181019-03:
    - ファイル名 : 20181019-03-QQQQ.c (QQQQ は学生番号)
    - 内容 : 変数宣言と代入文
  - 課題 20181019-04:
    - ファイル名 : 20181019-04-QQQQ.c (QQQQ は学生番号)
    - 内容 : while 構文
  - ※ 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)

# 代入

## □代入とは

### ○概念：「変数」に「値」を「割り当て」る「操作」

- ▹代入「後」は、その変数の値は、代入さ(割り当てら)れた値に「変化」する
- ▹(その変数が保持していた..) 代入「前」の値は、「失われ」る
- ▹代入の「前」と「後」という「時間」の概念の把握が必要となる

### ○表現：代入の構文

- ▹「変数名」=「式」 [例] a=1+2; (変数 a に 3 ( = 1+2 ) を代入)
- ▹「=」は、「代入」を表現する(等号[等しい]ではない!! / 等号は「==」)

## □局所変数宣言

### ○概念：局所変数を宣言する

- ▹関数(ブロック)内のみ(局所的)で有効(利用可能)な変数を宣言する
- ▹「仮引数変数(実は局所変数の一種)」以外にも、変数が増やせる

### ○表現：局所変数の宣言

- ▹「変数の型名」「変数名」 [例] int a; (整数型の変数 a を宣言)
- ▹cf. 仮引数変数は、実引数の値で、「代入済」の変数
- ▹未代入の変数の値は「未定(プログラムミスの代表例 !!)」
- ▹変数は宣言と同時に「初期化(最初の代入)」できる(すべき) [例] int a=1;

# while 構文

- while 構文
  - 概念 : 繰返しのため構文
    - 同じ命令を繰り返す事ができる ( cf. 再帰呼出し )
  - 表現 : while 構文
    - while (「条件」) { 「繰り返す命令」 }
    - 「条件」の部分は、if と同じ
    - 「繰り返す命令」の中には、「代入」が必須 (でないと「条件」が変化しない)
- while 文 vs 再帰
  - while 文は常に再帰に変換できる ( 実は原理的に逆も可能だが自明ではない )
    - func() { while (条件) { 文 } } → func() { if (条件) { 文; func(); } else {} }
  - その意味で、再帰の方が表現力がある(優秀)といえる
    - 逆に(工学のトレードオフの典型例)、while 構文の方が「効率」がよい

# 標準 I/O ライブラリの利用

- 標準 I/O ライブラリ
  - I/O とは Input(入力) と Output(出力) の事
    - Input : プログラムが、プログラムの外から、情報を読み込む(Read)
    - Output : プログラムが、プログラムの外へ、情報を書き出す(Write)
    - [ポイント] 「外」はプログラムの置かれている環境で異る (i.e OS)
    - 例 : 「あいさつ」をするとき日本とアメリカで異るの嫌(だから全て英語で.. それも困るが..)
  - 標準 I/O ライブラリの役割
    - 異る環境でも、プログラムとしては、同じ方法で I/O を行うための仕組
    - 環境の違いはライブラリが吸収 (挨拶は手を振るで OK)
  - 標準 I/O ライブラリの機能
    - I/O する物 : 文字、文字列、数値(基本データ)
    - I/O の相手先 : キーボード(I), ディスプレイ(O),ファイル(I/O)

# printf

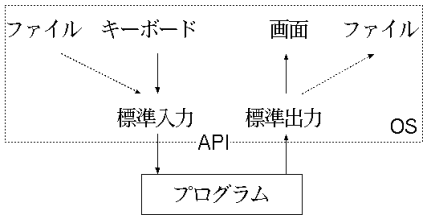
- printf：超高機能出力関数
  - print with format (書式付き出力)
    - 単なる文字列出力関数ではなかった ( cf. s_print_string : 単機能 )
  - 「書式('%' + 書式指示)」を指定する事により何(基本型+文字列)が出力できる
    - printf ( "%d", 123 ); / printf ( "%f", 1.23 ); / printf ( "%c", 'a' ); / printf ( "%s", "abc" );
  - 文字列の中に出力を埋め込む事ができる
    - int a=123; printf ( "int a=%d\n", a );
  - 複数のデータを一度に出力する事ができる
    - int a=123; double b=1.23; printf ( "int a=%d, double b=%f\n", a, b );
  - printf の動作原理
    - 後でちゃんと話すので、今回は我慢 !!

# scanf

- scanf : 超高機能入力関数
  - scan with format (書式付き入力)
    - 色々な型のデータを読み込む事ができる ( cf. s_input_int : 単機能 )
  - 「書式('%' + 書式指示)」を指定する事により何(基本型+文字列)が入力できる
    - int a; scanf ( "%d", &a );
    - !! a の前の「&」は「お呪い」(後でちゃんと話す)
  - 書式や機能などについても printf と同様に考えてよい
    - 文字列の中から値を取り出す事もできるのだが.. (結構難しいのでさけるのが無難 .. )

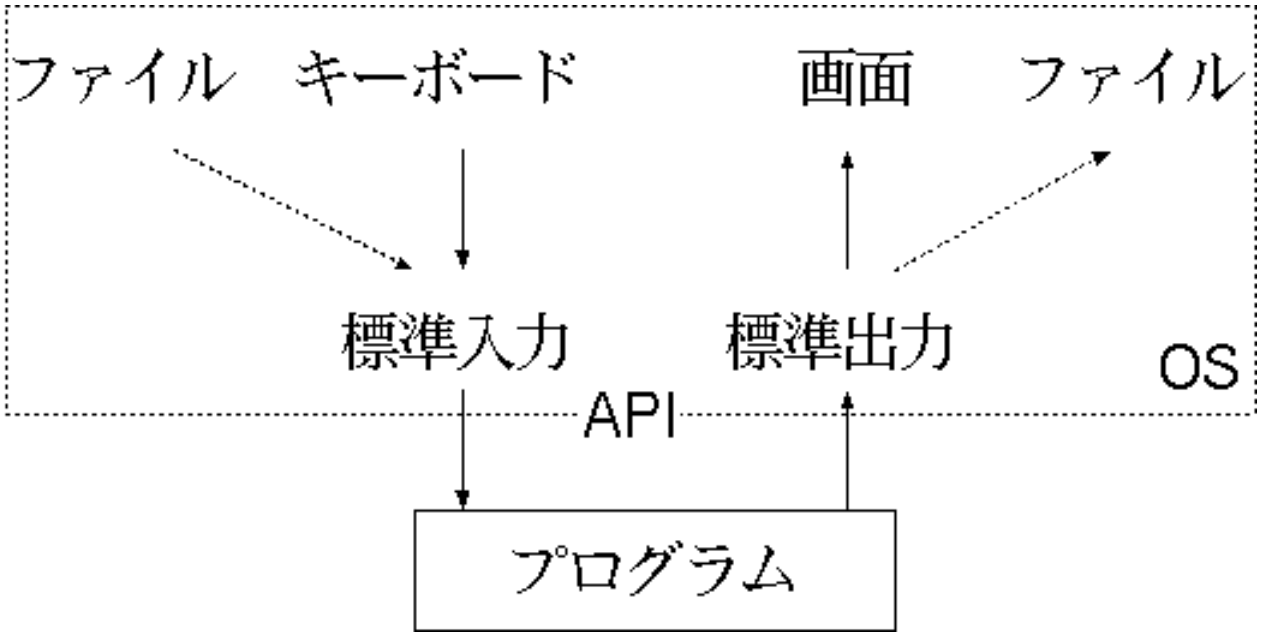
# プログラムの「標準 I/O」とは

- □「標準」I/O とは？
  - ○getchar は、キーボードからの入力 / putchar は、画面への出力
    - ▷なら、「標準」は「キーボード」と「画面」の事？
    - ▷No, But, Yes (ちがう、だけど、そうなんだ..)
    - ▷「?? それって、意味不明なんだけど...」
  - ○つまり..
    - ▷「標準入力」は、飽く迄も「標準入力」で、「キーボード」と異る (だから、No)
  - ○しかし..
    - ▷「普段」、「標準入力」は「キーボード」に「対応付け」されている(だから、Yes)
    - ▷「標準出力」と「画面」の関係も同様
  - ○改めて「標準 I/O」とは？
    - ▷そのプログラムにとって、「特に指定為ずに入出する先」の事
    - ▷プログラムは、「入出する先が本当は何になっているか」は、気にしない
    - ▷対応付けは、OS が行うので、「プログラムには無関係」な事柄



# リダイレクション

- 標準 I/O とデバイス
  - 「標準 I/O」と、「実際の I/O 先(デバイス)」の対応は？
    - 普段は OS が、「キーボード」と「画面」に対応付けている




(C) 坂本直志(東京電機大学工学部情報通信工学科) http://edu.net.c.dendai.ac.jp/literacy/2004/12/index.xml

- リダイレクション(入出力切り替え)とは
  - 「標準 I/O」の先を「別のデバイス」に切り換える事
    - 本当は「色々な物(デバイス)にリダイレクト(切替)」可能だが、ここではファイルだけ

# リダイレクション(redirection)の方法

- コマンドラインでのリダイレクションの方法
  - 「標準出力」を「ファイル」へ「リダイレクション」
    - コマンドラインに「> 出力ファイル名」とすると「標準出力」が「出力ファイル」に切り替わる
    - そのコマンドの標準出力先(これまでは、画面)が「出力ファイル」に切り替わる
  - 「標準入力」を「ファイル」からに「リダイレクション」
    - コマンドラインに「< 入力ファイル名」とすると「標準入力」が「入力ファイル」に切り替わる
    - そのコマンドの標準入力元(これまでは、キーボード)が「入力ファイル」に切り替わる

# stdin/stdout

- C 言語での標準入出力 ( standartd Input/Output )
  - putchar/getchar, printf/scanf は、「標準入出力」に I/O を行う
    - プログラムで入出力を切り換える事ができる
  - C 言語で標準で利用できる入出力先
    - 標準入力 : stdin / 標準出力 : stdout / 標準エラー出力 : stderr
  - fputc/fgetc, fprintf/fscanf では、標準入出力以外の I/O 先が指定できる
    - putchar(ch) は fputc ( ch, stdin ) と同じ
    - getchar() は fgetc ( stdin ) と同じ
    - scanf ( fmt, .. ) は fscanf ( stdin, fmt, .. ) と同じ
    - printf ( fmt, .. ) は fprintf ( stdout, fmt, .. ) と同じ
  - stderr : 標準エラー出力
    - 標準出力(stdout)が、「本来の情報の出力先」を意味するのに対し、「異状な場合の特別な情報出力」を行う
    - OS で、標準出力をリダイレクトしても、標準エラーは影響を受ない

# ファイル I/O

- ファイルを対象とする I/O
  - ファイルは Open してから利用を開始し、利用が終ったら Close する必要がある
    - fopen : ファイルを open する関数 (ファイルを「開く」関数)
    - fclose : ファイルを close する関数 (ファイルを「閉じる」関数)
- ファイルポインター
  - ファイルポインターって？
    - 「ファイル情報管理構造体」へのポインター(詳しくは後日)
  - ファイルを open すると、「ファイルポインター」が手に入る
  - ファイル(外にある)を内部で扱うには「ファイルポインター」を経由する
  - ファイルを close する時にも「ファイルポインター」を指定する
- ファイルへの I/O
  - ファイルポインタを使って、fprintf/fscanf で行う
  - 文字の入出は fputc/fgetc も利用できる
- ファイル I/O の注意
  - ファイルは Open しないと使えない / 利用が終ったら必ず Close する