# ソフトウェア概論 A/B

-- 関数, 分割コンパイル, リンク --

数学科 栗野 俊一 / 渡辺 俊一

# 伝言

## 私語は慎むように !!

- 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 の電源を入れる
  - ネットワークに接続しておく事
  - 今日の資料に目を通しておく事
- 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

# 今後の予定

- 今後の予定 (後ろから)
  - 2013/07/26 前期講議最終日
    - 試験 / Note-PC 必須 / PC のトラブル対応はしない
  - 2013/07/19 前期講議最終日前
    - 前期のまとめ / 模擬試験 / Note-PC 必須 / 環境を整える
  - 2013/07/12
    - 前期の内容の終わり / データ構造 (二周目の終了)
  - 2013/07/05
    - 次週 / 代入と制御構造
  - 2013/06/28
    - 本日 / 関数と分割コンパイル

# 前回 (2013/06/21) の復習

- 前回 (2013/06/21) の復習
  - Data(操作[計算]されるモノ) と Code(操作するモノ)の違い
    - Data には、「型」と「値」がある
    - 型 : 値の集合と操作の仕方 / 値 : 値の集合の要素
    - (例1) 整数型 : 値の集合 = { ..,-1,0,1,.. }, 操作 = { +,-, .. }
  - C 言語で扱える Data の型 (型名)
    - 整数型 (int) / 文字型 (char) / 浮動小数点数型(double) / 文字列型? (char *)
  - 色々な物の入出力 ( s_input.h/s_print.h )

# お知らせ

- 本日の予定
  - 整数型/浮動小数点型/文字型
  - 関数と制御構造
- 本日の目標
  - 演習
    - 課題の提出

# 前回 (2013/06/21) の課題

- 前回 (2013/06/21) の課題
  - 課題 1:
    - ファイル名 : 20130621-1-QQQQ.c (QQQQ は学生番号)
    - 内容 : キーボードから入力された二つの整数型の値の四則と余りの結果を表示する
    - ファイル形式 : テキストファイル(C 言語プログラムファイル)
  - 課題 2:
    - ファイル名 : 20130621-2-QQQQ.c (QQQQ は学生番号)
    - 内容 : キーボードから入力された二つの浮動小数点数型の四則の結果を表示する
    - ファイル形式 : テキストファイル(C 言語プログラムファイル)

# 本日の課題 (2013/06/28)

- 本日 (2013/06/28) の課題
  - 課題 1:
    - ファイル名 : 20130628-1-QQQQ.c (QQQQ は学生番号)
    - 内容 : キーボードから入力された一行の文字列を全て大文字にして出力する
    - ファイル形式 : テキストファイル(C 言語プログラムファイル)
  - 課題 2:
    - ファイル名 : 20130628-2-QQQQ.c (QQQQ は学生番号)
    - 内容 : 入力された数の 3 乗根(の近似値)を求める
    - ファイル形式 : テキストファイル(C 言語プログラムファイル)

# C 言語で扱える色々な型

- C 言語で利用できる(基本)型(のいくつか..)
  - char 型(文字型) : 一つの文字を表現
    - 例 'a' : 文字「a」を表す。
  - int 型(整数型) : 整数を表現
    - 例 123 : 整数「123」を表す。整数同士の割り算は、小数点以下切り捨て。
  - double 型(浮動小数点型) : 実数(小数点数)を表現
    - 例 12.34 : 実数「12.34」を表す。
  - [注意] C 言語では「文字列型」はないが、文字列を扱う場合は「char *」と指定する
- 関数の引数ならびに、返り値(かえりち)の型は上記で指定
  - 「ない」場合は「void」と指定する

# 関数

- 関数とは
  - 幾つかの値を引数として渡すと何かの処理をして結果を値として返す
    - 例 minus_int / minus_double / sin / sqrt
- 関数宣言の頭部
  - 「返り値型 関数名 ( 引き数型 仮引数 [ , 引数型 仮引数 .. ] )」
    - 関数の前には「返値(かえり値)」の型を書く (返値がない場合は void にする)
    - 仮引数の前には「引数の型」を付ける
  - 仮引数：実引数で指定した値を持つ (型を一致させる必要がある)
- 関数宣言の体部
  - 「 { 命令列 }」
    - 命令列の式の中で変数」が利用できる
  - 「return 式」
    - 関数の実行をそこで終了し、式の値を関数の値として返す

# 条件分岐

- 式の値によって振舞を変更したい場合に利用する

```
if ( 条件式 ) {
    「条件式」が成立 ( 0 以外.. ) の場合に実行する命令
} else {
    「条件式」が不成立 ( 0 になる ) の場合に実行する命令
}
```

  - 「条件式」の値
    - C 言語では、0 を「偽」、0 以外を「真」として扱う
  - strcmp ( char *strA, char *strB );
    - 二つの文字列 strA を strB と比較し、小さいと負、等しいと0、大きいと正の整数が返る
    - ! は、0 を 1 に 1 以外を 0 にする
    - !strcmp ( strA, strB ) : strA と strB が同じなら 1 になる

# 再帰呼び出し

- 再帰呼び出し
  - 普通の関数は、別の関数を呼出す事ができた
    - 「自分の中」で「自分自身」を呼出す事ができる !! : 再帰呼び出し
- 再帰呼び出しと帰納法
  - 再帰呼び出しは、帰納法の考え方で問題を解く場合に利用できる
    - 再帰呼び出しが上手く行く事は、帰納法で証明できる (数学との関係)
  - 再帰呼び出しをする場合は次の二点が重要 ( 帰納法と同じ )
    - 最も小さい場合 ( ここでは、文字列が "" の場合 ) には終了する
    - そうでない時は、再帰呼び出しするが、その時には文字列を短くする

# 再帰呼び出しの考え方(再)

- 目標
  - 「全部」をやりたい
    - でも一挙にはできない
- 対策
  - そこで問題を二つに分ける
    - 扱いやすい一部分：これは、そのまま対処してしまう
    - 残り全部：(残り)「全部」なので、再帰呼び出しする
- 注意点
  - 「全部」が空っぽの時を忘れずに処理する

# 三つの基本制御構造と万能性(再)

- 三つの基本制御構造
  - f を関数, A,B を命令、p(x) を条件とする時、次の三つの基本構造がある
  - [順接] f() { A B }
    - f は A をしてから B をする
  - [分岐] f(x) { if ( p(x) ) { A } else { B } }
    - f は p(x) が成立すれば A そうでなければ B をする
  - [繰返] f(x) { if ( p(x) ) { A f(x') } else {} }
    - f は p(x) が成立する限り A を行う
    - x' は x から計算される
- 万能性
  - 任意のプログラムこの三つの基本制御構造で構成可能
    - 「三つの基本制御構造」を憶えればあとは組み合わせを考えるだけ !!

# 整数型(再)

- int 型
  - C 言語での整数
    - 表現できる範囲は限られている
    - 32bit の場合は −2147483648 から 2147483647
  - 宣言：int で行う
  - 計算：四則が可能 +, -, *, /
    - / は整数割り算なので、小数点以下は切捨てになる
  - 比較：大小比較、等号、不等号が使える
    - a > b : a が b より大きい
    - a >= b : a が b 以上
    - a == b : a と b が等しい ( = でないことに注意 !! )
    - a != b : a と b が等しくない
- 整数型の出力 (当分は..)
  - s_print.h の中の s_print_int を使う ( sample-001.c )

# 浮動小数点数型(再)

- □ double 型
  - ○ C 言語での実数 (小数点付きの数を表現する)
    - ▹ C 言語内での表現 : 小数点付きの数 ( cf. 123.456 )
    - ▹ 出力 : s_print_double ( s_print.h 内 )
    - ▹ 入力 : s_input_double ( s_input.h 内 )
  - ○ 様々な数学的な関数が利用できる
    - ▹ sin/cos/exp/log/etc.. cf #include <math.h>
  - ○ (当然) 四則の計算ができる : 3.0/2.0 → 1.5 ( cf. 3/2 → 1 )
  - ○ 近似計算を行う
    - ▹ (((1.0)/3.0)*3.0) は 1.0 に戻らない
    - ▹ 浮動小数点数 a, b を比較する場合は、「a==b(等値比較)」は使わず、「(a-b)< 0.0001 (近似比較)」を使う

# 二分探査法

- 問題
  - 連続関数 f(x) に対して、f(x)=0 となる x を求める
    - 仮定 : 区間 [a,b] に対し f(a)*f(b) < 0
    - つまり f(a) と f(b) の符号が異るので、区間[a,b]に少くても根が一つある(中間値の定理)
    - m を a,b の中点とすると、根は [a,m], [m,b] の少くてもどちらか一方にある(性質 <*>)
  - 二分探査法
    - <*> に基いて、区間をせばめる方法