ソフトウェア概論 A/B

-- 代入と制御構造 --

数学科 栗野 俊一 / 渡辺 俊一

2013/07/12 ソフトウェア概論

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
    - 本日 / 前期の内容の終わり / データ構造 (二周目の終了)

前回 (2013/07/05) の復習

- 前回 (2013/07/05) の復習
  - 二分法
    - 解(候補)集合を二つに分け、現時点の情報から解がどちらにあるか判定し選択する
    - 解(候補)集合が十分に小く(要素が一つになるなど..)なれば、それを解とする
  - 代入
    - 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(割当ら)れた値に「変化」する
    - 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
  - 局所変数宣言
    - 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

お知らせ

- 本日の予定
  - 代入 (続き)
  - while 文 : 代入を利用する制御構造
- 本日の目標
  - 演習
    - 課題の提出

前回 (2013/07/05) の課題

- 前回 (2013/07/05) の課題
  - 課題 1: (前々回の課題の残り)
    - 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)
  - 課題 2:
    - ファイル名 : 20130705-1-QQQQ.c (QQQQ は学生番号)
    - 内容 : 入力された三つの整数を小さい順に出す
    - 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 (2013/07/12)

- 本日 (2013/07/12) の課題
  - 課題 1: (前々回の課題の残り)
    - ファイル名 : 20130712-1-QQQQ.c (QQQQ は学生番号)
    - 内容 : 非負の整数を幾つか入力し、その和を求めて、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)
    - 条件 : できれば、while と代入文を利用する

代入(再)

- 代入とは (what)
  - 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(割当ら)れた値に「変化」する
    - 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
    - 逆に、「代入」が行われなければ、「変数の値」は「同じ」ままである
  - 表現
    - 「変数名 = 式」
      - 「式」には、その「変数」が含まれていて良い
      - 変数の「値の更新」ができる
- 代入が利用される理由 (why)：「効率が良い」から
  - 変数の値は何度でも参照できる
    - 一度計算した結果を変数に保存しておけば、(再計算せず)何度でも参照可能
  - 代入によって、同じ変数に別の値が保存できる
    - 一時的な値のために沢山の変数を用意せずに済む

while 文

- while 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 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 (逆も原理的には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 文の方が「効率」がよい

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);
  - print の動作原理
    - 後期にちゃんと話すので、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」(後期にちゃんと話す)
  - 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 ..)