ソフトウェア概論 A/B

-- 関数 again(2)/乱数/入力ループ --

数学科 栗野 俊一 / 渡辺 俊一

2014/06/06 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
- 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
- 本日の CST Portal の出席パスワード : 20140606
  - 出席は成績に影響しませんが、折角の機能なので、使いましょう

前回(2014/05/30)の復習

- 前回(2014/05/30)の内容
  - 作成
    - 関数の作り方(再)
    - 値を持つ関数：関数は値を「返す」事ができる
    - 英語の「Function」の意味：関数(値を返す)、機能(値を返さない)
  - 表現
    - 「return 式;」: 関数の返す値(返り値)を「式」で計算して返す命令
  - 操作/機能
    - s_print.h/s_output.h の利用方法

お知らせ

- 本日(2014/06/06)の予定
  - 作成
    - 制御構造(再)
    - 入力ループ
  - 操作/機能
    - 乱数関数
- 本日(2014/06/06)の目標
  - 講議
    - 制御構造の復習
  - 演習
    - 再帰関数/入力ループ
    - 課題の提出

前回 (2014/05/30) の課題

- 前回 (2014/05/30) の課題
  - 課題 1:
    - ファイル名 : 20140530-1-QQQQ.c (QQQQ は学生番号)
    - 内容 : 二つの整数の積を返す関数
    - 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 : 20140530-2-QQQQ.c (QQQQ は学生番号)
    - 内容 : 自然数の階乗を返す関数
    - 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)
- 注意 1 : 前回(2014/05/30)の課題 3 は、今回(2014/06/06)に回す
- 注意 2 : 前回(2014/05/30)の課題 4 は、取り敢えず、やらなくてよい

本日 (2014/06/06) の課題

- 本日 (2014/06/06) の課題 (CST Portal のみ)
  - 課題 1:
    - ファイル名 : 20140606-1-QQQQ.c (QQQQ は学生番号)
    - 内容 : 数当てをするプログラム
    - 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 : 20140606-2-QQQQ.c (QQQQ は学生番号)
    - 内容 : 与えられた整数の素因数を表示するプログラム
    - 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 3:
    - ファイル名 : 20140530-3-QQQQ.c (QQQQ は学生番号)
    - 内容 : 二つの非負の整数の最大公約数を返す(ユークリッドの互除法)
    - 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)
    - 注意 : 前回(2014/05/30)の課題なので、名前が 20140530 である事に注意

条件分岐(再)

- 条件分岐
  - 条件によって、「異る二つの振舞の一方を実行(分岐)」する事
- if 構文
  - 条件分岐を記述する「構文(記述形式)」の一つ
    - 構文：「if (条件) { 命令1 } else { 命令2 }」
    - 意味：「条件」が成立すれば、「命令1」が、そうでない時には、「命令2」が実行される
  - 「単独」の if 構文は、「二択」だが、「if 構文」の組み合わせで「多択」が可能
- if 構文の亜種
  - else 節の省略
    - 「if(条件) { 命令 } else {}」の「else{}」は省略可能
  - else if 句：次の二つは同じ意味
    - 「if (条件1) { 命令1 } else {if (条件2) { 命令2 } else {命令3}}」
    - 「if (条件1) { 命令1 } else if (条件2) { 命令2 } else {命令3}」

再帰呼出し(再)

- 再帰呼出し
  - ある関数の本体に、その関数の呼出しを含まれている事
    - 「その関数の一度の呼出し」で「その関数の呼出しが」が何度も呼び出される可能性が生じる
  - 注意：安易に「再帰呼出し」を記述すると、「無限ループ」になる
    - 「再帰呼出しを終了させる」仕組を「意図的に導入」する必要がある
    - 「条件分岐」が必須
  - 再帰句：P は、停止条件、A は繰り返す命令、X' は X から計算され、いつか P(X)が成立する
    - 記述形式：f(X) { if (P(X)) {} else { A(X); f(X'); }
    - 意味：f(X) を呼ぶと、P(X''..')が成立するまで A(X) を繰り返す
    - ポイント：「A(X) を繰り返す」ために「f() を再帰で定義」する
- 再帰関数
  - 再帰呼出しを行う形で定義された関数
- 繰返し：同じ命令(記述)を何度も呼び出す仕組
  - 再帰呼出しを利用する事により、「繰返し」が実現できる
  - 再帰関数は、「何らかの繰返し」を実現している

インプット・ループ：入力による繰り返し

- 一般的な再帰関数
  - 引数の値によって、挙動(繰返し回数)が変化する
    - 引数が確定すれば、繰返し回数も確定する
- インプット・ループ
  - 入力(インプット)の値によって繰返し回数を制御したい
    - 典型的な応用:「終わり」まで「繰り返す」
- インプット・ループを作るパターン
  - 入力関数を引数とする再帰関数を作ればよい
  - インプット・ループ句
    - 記述形式 : f(X) { if (P(X)) {} else { A(X); f(input()); } }
    - 意味 : 入力 X が P を満すまで A(X) を繰り返す
    - ポイント : f を最初に呼び出す場合も f(input()) の形にする

乱数

- 「乱数列」とは
  - 定義：{r_n} (n = 1, 2, ..) が「乱数列」であるとは
    - r_{i+1} が、それ迄に与えられた r_1, r_2, .., r_i から予想できない数列になっている場合
    - cf. (不正のない)サイコロを(無作意に)投げて、出た目を並べた数列は「乱数列」になる
    - cf. n 番目の数 r_n を n から求める「規則(例えば r_n = n^2)」がある場合は乱数でない
  - [注意]「予想できない」を「数学的に表現する」のは難しい(西川先生を困らせてみよう)
    - 現実的には「予想をしない(数学の立場)」か「乱数のようにみえる数列(疑似乱数:計算機の立場)」を取る
- 「乱数」とは
  - 「乱数列 {r_n}」の i 番目の要素 r_i を取った物の事(なので、その値を事前に予想できない)

「乱数の分布」と「疑似乱数」

- 「乱数」の分布
  - 「自然現象に現れる乱数:{r_n}」は、「個々の要素(r_i)の振舞」は解らない
    - 「集合としての振舞(の一部)」である「分布」は解っている(仮定してもよい..)事が多い
  - 一様分布：r_i はある範囲の数で、どれかが現れる確率がどれも同じ
    - cf. さいころの出目は、{ 1 ～ 6 } で、どれも等確率(1/6)
  - 正規分布：{r_i} の分布は、正規分布 N(μ , σ ^2) に従う
    - cf. 自然現象における雑音等
- 乱数の応用
  - 個々の現象が「起きる規則が知られていない/予想できない」場合に「乱数」として扱う
    - ただし、予め「対象となる現象」の分布(統計を用いる)を調べる
    - その乱数が、その分布に従うと「仮定」する
- 「疑似乱数列」とは
  - 「乱数列」のように*見える*ように「規則的に作成された」数列