# ソフトウェア概論 A/B

-- 再帰 again --

数学科 栗野 俊一 / 渡辺 俊一

# 伝言

- □ 出席パスワード : 20200717
- □ 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
- □ 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

ソフトウェア概論 A/B (2020/07/17)

# 今後の予定(後ろから)

# 今後の予定(後ろから)

- 今後の予定
  - 2020/07/31 (講議最終日)
    - 試験 / Note-PC 必須 / PC のトラブル対応はしない / 課題提出最終日
    - リアルタイム (問題に関する質問のみ)
  - 2020/07/24 (講議最終日前)
    - 前期のまとめ(落穂拾い) / 模擬試験 / Note-PC 必須 / 環境を整える
    - リアルタイム (解説/質問/受付)
  - 2020/07/17 (本日)
    - 再帰 again
    - メディア授業最後
- リアルタイム講義
  - 来週(2020/07/24:模擬試験)/再来週(2020/07/31:試験)は、リアルタイム講義
    - 本日(2020/07/17) 18:00 ～ 19:00 に「meet 参加実験」可能
  - 講義時間内に講義を受けてください
    - 09:00 ～ 12:10
  - Google Meet を利用して TV 会議に参加してください
    - 会議 ID : hrh-kuav-nhm

ソフトウェア概論 A/B (2020/07/17)

# 前回(2020/07/10)のまとめ

# 前回(2020/07/10)の復習

- 前回(2020/07/10)の内容
  - 講義内容
    - 条件分岐
  - 演習
    - 課題の提出

ソフトウェア概論 A/B (2020/07/17)

# 今回(2020/07/17)の予定と課題

# 今回(2020/07/17)の予定

- 出席パスワード：20200717
  - 出席は CST Portal で取りますが、成績には(残念ながら?)無関係です
    - 単位を取りたいならば、課題を提出しましょう
- 本日(2020/07/17)の予定
  - 再帰構造の更なる理解
  - 入力(出力)という「副作用」
- 本日(2020/07/17)の目標
  - 課題の提出

# 先週 (2020/07/10) の課題

- 先週 (2020/07/10) の課題
  - 課題 20200710-01
    - ファイル名 : 20200710-01-QQQQ.c (QQQQ は学生番号)
    - 内容 :  異る国の挨拶をする ( switch 構文 版 )
    - 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 20200710-02:
    - ファイル名 : 20200710-02-QQQQ.c (QQQQ は学生番号)
    - 内容 : if 構文の入れ子 ( 文字種を判定する関数 )
    - 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)

# 今週 (2020/07/17) の課題

- 今週 (2020/07/17) の課題
  - 課題 20200717-01
    - ファイル名 : 20200717-01-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 20200717-02:
    - ファイル名 : 20200717-02-QQQQ.c (QQQQ は学生番号)
    - 内容 : 与えられた自然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 20200717-03:
    - ファイル名 : 20200717-03-QQQQ.c (QQQQ は学生番号)
    - 内容 : フィボナッチ数列の第 n 項を返す関数
    - 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)

ソフトウェア概論 A/B (2020/07/17)

# 再帰呼出し again

# 再帰呼出し again

- 再帰呼出し
  - ある関数の本体に、その関数自身の呼出しを含む事
    - 「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、『何時か』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

# 再帰呼出しアラカルト

- 再帰呼出しアラカルト
  - 再帰呼出し：「繰返し」を生む構造(『構文』と同様な仕組み)
    - 『何』を『どのよう』に『繰り返す』か？
- 単純な繰返し ( N 回の繰返し )
  - 同じ内容の繰返し / N に合わせた繰返し
- 蓄積型の繰返し
  - 計算結果を蓄積する (総和)
    - 関数値の利用 / 引数への蓄積
- 検索型の繰返し
  - 繰返しの途中で結果が中断される (目的が達成されるまで繰返し)
    - 繰返し回数の予想ができない (条件の成立時期が予測できない)
- 副作用(入出力等)の蓄積
  - 純粋な計算の蓄積ではなく、副作用(入出力)の蓄積を目的とする
- 多分岐再帰
  - 再帰呼出しが複数行われる
    - 単純な「繰返し」にはならない / 計算量が膨大になる可能性がある

ソフトウェア概論 A/B (2020/07/17)

# 入力(出力)という「副作用」

# 入力(出力)という「副作用」

- 数学の関数
  - 値を与えると値を返す
    - 何回利用しても同じ結果にしかならない
    - 例 : sin(x) + sin(x) == 2 * sin(x)
  - 「純粋関数」
    - 「副作用」を持たない関数 ( cf.「数学」の「関数」は「純粋」)
    - 数学的な操作 (置き換えや、交換等)が可能 :「(『論理的』に..)扱い易い」
    - 「実行順序」に意味がない(素直)
- C 言語の関数
  - 「副作用」を持つ場合がある
    - 「副作用」の例 : 入出力 ( 他にも色々ある )
  - 「実行順序」に意味がある (だから、「制御構造」に『意味』がある)
    - 「扱いが不便」だが「やり甲斐」がある

# 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

ソフトウェア概論 A/B (2020/07/17)

# s_random.h

# s_random.h

- s_random.h
  - 疑似乱数を利用するためのヘッダー
- インストール
  - c:\usr\c\include に保存
- 使い方
  - s_input.h/s_print.h と同様
    - ソースコード ( C ファイル ) の先頭で include
    - コンパイル時に「-I ~/c/include 」を追加
- 機能
  - s_init_random() : 乱数列の初期化 (main で、最初に一度呼ぶ)
  - s_random() : 乱数値を返す
    - 乱数値の型は整数型で、乱数値の分布は一様分布 (以下の関数も同様)
  - s_n_random(N) : 0 ～ N-1 の範囲の整数値の乱数値を返す
  - s_coin() : 0/1 の乱数値を返す
  - s_dice() : 1 ～ 6 の乱数値を返す

ソフトウェア概論 A/B (2020/07/17)

# 疑似乱数

# 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』」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( $\mu$ , $\sigma$ ^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

ソフトウェア概論 A/B (2020/07/17)

# おしまい