ソフトウェア概論 A/B

-- 一周目のまとめと万能性の話 --

数学科 栗野 俊一

2011/05/20 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)
- 講義開始前にすませておくこと
  - PC の電源を入れる
  - ネットワークに接続しておくこと
  - 今日の資料に目を通しておくこと
- 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

前回の復習

- 前回の復習
  - 引数付き関数を作ってみよう
  - PC で Turtle Graphics (亀プログラム) をしてみよう
  - 条件判定をしてみよう
  - 再帰呼び出しをしてみよう

お知らせ

- 本日の予定
  - 3D モデリングデータを表示してみる
  - 再帰呼び出しの復習
  - コーディング
  - 万能計算
- 本日の目標
  - 演習
    - 3D モデリングデータを表示
    - 再帰プログラム
    - 課題の提出

先週 (2011/05/13) の課題

- □ 先週 (2011/05/13) の課題
  - ○ 課題 1:
    - ▹ ファイル名 : 20110513-1-QQQQ.c (QQQQ は学生番号)
    - ▹ 内容 : 漢字の「回」という文字(にみえる..) 絵を Turtle Graphics で書なさい
    - ▹ 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 : 20110513-2-QQQQ.c (QQQQ は学生番号)
    - ▹ 内容 : 引数付き関数で、if 文で条件判断をするプログラムを作成しなさい
    - ▹ 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: (この課題は今週の課題になったので提出期日がのびる)
    - ▹ ファイル名 : 20110513-3-QQQQ.c (QQQQ は学生番号)
    - ▹ 内容 : 底辺の長さが指定した文字列の二倍の長さ - 1 の横向のピラミッドを作成するプログラム
    - ▹ 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)
    - ▹ 再帰呼び出しを利用する

今週 (2011/05/20) の課題

- 今週 (2011/05/20) の課題
  - 課題 1:
    - ファイル名 : 20110520-1-QQQQ.txt (QQQQ は学生番号)
    - メタセコイヤのモデルデータの URL
  - 課題 2:
    - 先週 (2011/05/13) の課題 3 をしなさい
    - ファイル名などは先週の物を利用する
    - 既に、提出済の場合は再度提出する必要はない
  - 課題 3:
    - ファイル名 : 20110520-3-QQQQ.c (QQQQ は学生番号)
    - 内容 : 余りを計算するプログラム考えよ
    - 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)

ファイルの入手とインストール

- ファイルのダウンロード
  - 次の本日 (2011/05/20) のページからファイルをダウンロードする

    http://edu-gw2.math.cst.nihon-u.ac.jp/~kurino/2011/soft/20110520/20110520.html
  - ダウンロードするファイル
    - GLMetaseq.h, GLMetaseq.c, sample_GL.c, ninja.mqo
  - いずれも「c:\usr\soft」に保存する事
- ファイルをダンロードしたら次の作業を行う
  - コマンドプロンプトを開く
  - 「cd c:\usr\soft」とする
  - 「cc sample_GL.c GLMetaseq.c」
  - 実行してみる
    - 「sample_GL」とする
    - 「sample_GL foobar.mqo」とする。「foobar.mqo」は、モデリングデータファイル名

再帰呼び出し (再録)

- □ 文字列を順番にみてゆく
  - ○「"abc" + 1」は「"bc"」と同じ振舞いをする
    - ▹どうゆう仕組かは今回は説明しない
  - ○次々と 1 を加えれば、どんどん短かくなる
    - ▹最も短かくなったかは、空文字(「""」)と比較すれば判定できる
- □ 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
    - ▹自分自身も呼出す事ができる !! : 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
  - ○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 (再録)

- 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

課題 2 の考え方 (1)

- 具体例で考えてみる
  - 例：大きさが 4 の三角形を作る
    - down_triangle ("****");

****

***

**

*

  - 一行目と二行目以降に分割して考える

**** と ***

**

*

  - 一行目は単に、出力するだけ
    - printf ("****");
  - 二行目以降は、大きさが 3 の三角形になっている
    - 再帰呼び出しをする
    - down_triangle ("***") == down_triangle ("****" + 1);

課題 2 の考え方 (2)

- 大きさが 4 の三角形を作る

```
down_triangle ( "****" ) == printf ( "****" ); printf ( "\n" ); down_triangle ( "****" + 1 );
```

- 一般に大きさ n (length) の三角形を作る

```
down_triangle ( length ) == printf ( length ); printf ( "\n" ); down_triangle ( length + 1 );
```

- 再帰の最初の場合 (長さが 0) の処理を考える
  - 長さが 0 の時は何もしなくてよい
- 再帰を使って関数定義

```
void down_triangle ( char *length ) {
    if ( !strcmp ( length, "" ) ) { /* 長さが 0 の時 */
        /* なにもしなくてよい */
        } else {
          printf ( length );
            printf ( "\n" );
      down_triangle ( length + 1 );
        }
        }
```

一周目のまとめ (その 1 : Hello, World)

- □ Hello, World
  - ○ C 言語プログラミング手順
    - ▹ C プログラムの作成方法 (エディタ)
    - ▹ C プログラムのコンパイル方法 (cc コマンド)
    - ▹ C プログラムの実行方法 (cc が上手く行けば exe ファイルが手に入る)
- □ 色々なライブラリ
  - ○ C 言語プログラムの中から呼出すと「何か」がおきる
    - ▹ printf(); -- 文字列を画面に出力する
    - ▹ s_midi.h -- 音を鳴らす
    - ▹ s_turtle.h -- 亀を操る
    - ▹ sample_gl.c -- 3D モデルを表示させる
  - ○ ライブラリによって、C 言語から色々な事ができる
    - ▹「何をするか」は色々 (ドを鳴らす, 点を書く, 3D モデルを表示する)
    - ▹「どんな関数を呼ぶか」も色々 (基本は「調べ」問題 : Web を参照 !!)
    - ▹「どう呼出すか」は共通 (<=「C 言語」を学ぶ意味 !!)
  - ○「巨人の肩に立つ」
    - ▹ 既に色々なライブラリがあるので、それを利用すれば楽に色々できる !!

一周目のまとめ (その 2 : プログラム)

- □ プログラムの「命令実行」の意味
  - ○ 関数呼び出し
    - ▷ 関数を一度呼ぶとその関数に対応した現象がおきる
    - ▷ メッセージの出力 / 音 / 描画
- □ プログラミングとは？
  - ○ 「関数の呼び出し」の並びを作る
    - ▷ 並びの蓄積したものが「プログラムの実行結果」
  - ○ 望みの結果を作るには？
    - ▷ その結果を作る「命令並び」を実行すればよい
    - ▷ 「どの関数をどの順番に実行するか」を考える
- □ 命令並び
  - ○ 命令 (関数呼び出し)を並べると、それがその順に実行される
    - ▷ 最も基本的な「プログラムの作成方法」
    - ▷ 実行する順を考え、その順に命令を並べるだけ
  - ○ 他の形は、最終的に、この命令並びと同じものになる
    - ▷ 他の形の「意味」は、それと同じ結果になる命令並びと考えればよい

一周目のまとめ (その 3 : 関数)

- □ 関数
  - ○ 命令並びに名前を付けたもの
    - ▷ 名前で、その命令並びを呼出す事ができる
  - ○ 関数の利点
    - ▷ 「名前」がつくので判りやすい
    - ▷ 何度も利用できるので、プログラムが短くなる
    - ▷ 関数の中だけで考えてよい (狭い範囲で考えるとよい)
    - ▷ [ポイント] いずれも、人間にとって「わかりやすくなる」
- □ 引数付き関数
  - ○ 関数の本体となる命令の一部の内容を置き換える事ができる
    - ▷ 本体の中にある「変数」は、関数呼び出しの引数で指定した値になる
    - ▷ 引数を利用する事により、にている複数の関数が一つにまとめられる

一周目のまとめ (その 4 : 条件分岐と再帰)

- □ 条件分岐
  - ○ 引数で指定された値によって挙動を変えらえる
    - ▷ if 文を利用して、「実行条件」を調べる
    - ▷ 条件によって、二つの選択肢の内の一つを選んで実行できる
  - ○ 「条件判断」としては、文字列の比較のみ
    - ▷ strcmp (A, B) : これも実はライブラリ関数
- □ 再帰呼び出し
  - ○ ある関数の本体の中で自分自身を呼出す事ができる
    - ▷ 数学的帰納法と同じ考え方
  - ○ 全部をするには
    - ▷ とりあえず一つやる
    - ▷ のこり全部をする : 「残り」は、+1 で文字列を短くする事で実現
  - ○ カラッポの場合に注意

コーディング

□コーディングとは

○「表現」と「意味」の対応付けを行う作業

▷例："*" の並びと、その並んだ個数を対応付て考える

文字列	数
	0
*	1
**	2
***	3
...	

▷「"*" の並び」を画面に出す事を「数」を出すと「解釈」する

○コーディングによって、色々なものが「計算できるように」なる

命令(計算)	文字列	数
printf ("***\n");	***	3 の出力
printf ("**"); printf ("***\n");	*****	5 (= 2 + 3) の出力

▷文字列出力が「計算」に対応

コーディングによる計算

- □ コーディングによる計算
  - ○ コーディングを介して、文字を出力するプログラムが「計算」になる
    - ▹ 四則
    - ▹ fibnacci
- □ コーディングによる「数」の表現
  - ○ 数学入門 A/B で学んだ
    - ▹ n,m が自然数の時 : (n,m)/～ で整数を表現
    - ▹ n が整数、m が自然数の時 : (n,m)/～ で有理数を表現
- □ コーディングの利点
  - ○ 既存の物の組み合わせで、新しい事ができる
    - ▹ ライブラリを作成する事は、新しい「数」を作るのと同等

万能性

- □ 万能性
  - ○ 原理的に可能な事は何でもできる
  - ○ 万能性の条件
    - ▹ 「できる事」が十分に大きい
    - ▹ コーディグができる
- □ 今週まで学んだ事
  - ○ プログラムを組上げる三つの要素
    - ▹ 順接(並べる)、条件分岐、再帰呼び出し
  - ○ コーディングの考えかた
    - ▹ 万能の条件が満されている !!
    - ▹ ただし、コーディングを通すとまどろっこしい
    - ▹ 直接、「計算」できた方が嬉しい：ライブラリ問題
- □ 次回以降
  - ○ より詳しい内容を学びながら、もう一周する
    - ▹ Hello, World 再び