ソフトウェア概論 A/B

-- 繰返し(再起呼出し) / Turtle Graphics --

数学科 栗野 俊一 / 渡辺 俊一

伝言

私語は慎むように !!

- 出席パスワード : 20170519
- 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 の電源を入れておく
  - ネットワークに接続しておく
  - 今日の資料に目を通しておく
- 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(2017/05/12)の復習

- 前回(2017/05/12)の内容
  - 引数付き関数を作ってみよう
    - 作成: 複数の関数で「共通でない部分を変数」にして「共通化」する
    - 表現: 変化する部分を「変数」にする / 引数に変数を宣言 / 変化する値を呼出し時に指定
  - 条件判定をしてみよう
    - 作成: 状況に応じて「複数の命令のどちらか一方」を実行したい
    - 表現: 二つの命令を if(条件) { 一方 } else { 他方 } とする / 条件の所に「!strcmp(変数,文字列)」を入れる
  - 再帰呼出しをしてみよう
    - 作成: 「全体」を実行するのに、「一部」と「残り」で済ませたい
    - 表現: 関数の定義の中で、自分自身を呼び出す(再帰呼出し)

お知らせ

- 本日(2017/05/19)の予定
  - PC で Turtle Graphics (亀プログラム) をしてみよう
  - 文字列と文字の関係
- 本日(2017/05/19)の目標
  - 再起呼出しを利用した「繰返し」を学ぶ
  - 演習
    - 再帰呼び出しをするプログラム : 同じ事を必要なだけ繰り返す
    - 亀プログラム / 文字の出力
    - 課題の提出

前回 (2017/05/12) の課題

- 前回 (2017/05/12) の課題
  - 課題 20170512-01:
    - ファイル名 : 20170512-01-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 20170512-02:
    - ファイル名 : 20170512-02-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)
    - 再帰呼び出しを利用する

本日 (2017/05/19) の課題

- 本日 (2017/05/19) の課題
  - 課題 20170519-01:
    - ファイル名 : 20170519-01-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 で書きなさい
    - 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)

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(2017/04/21 に作業していれば良い)
  - 2017/04/21 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/2017/soft/20170421/20170421.html
  - ダウンロードするファイル (c:\usr\c に保存)
    - lib.zip
    - include.zip
  - ダウンロードするファイル (c:\usr\c\20170519 に保存)
    - turtle.zip
- ファイルをダンロードしたら次の作業を行う
  - ubuntu 側で ~/c/20170421/turtle に移動する
    - cd ~/c/20170421/turtle
  - 「make test」で実行して、絵が出れば良い
    - [Enter] を押すと終了する

Turtle Graphics (亀プログラム)

- お呪い
  - #include "s_turtle.h" を冒頭にいれる
- 「亀」の操り方
  - 「亀」は、最初の状態では
    - 画面の真中にいます
    - 上を向いています
  - 「亀」への命令は次の三つ
    - s_turtle_move(); : 現在の位置に足跡を残し、現在の方向に一歩進みます
    - s_turtle_jump(); : 現在の位置に足跡を残さず、現在の方向に一歩進みます
    - s_turtle_turn(); : 現在の方向を時計回りに 45 度変更します
    - s_turtle_stop(); : 亀プログラムの終了 (「return 0;」の直前に実行)
- 「亀」プログラムの実行手順
  - サクラエディタで、C ファイルを作成(foobar.c とする)
    - c:\usr\c\20170519\turtle に保存する
  - ubuntu で次のコマンドを実行する
    - make BASE=foobar test

文字の入力と出力

- 文字の表現
  - 文字は「'」で挟む (cf.「文字列」は「"」で挟む)
    - 当分は、半角のみ、日本語の「文字」は扱わない
- 文字の出力
  - putchar(文字); を使う
    - 「putchar ('a');」で文字('a')が出力される
    - 改行文字は '\n' で表す : putchar ('\n') で改行する
- 文字の入力
  - getchar() を使う
    - 「getchar()」とすると、キーボードからの入力を待つ
    - 「putchar (getchar());」とすると、入力した文字が出力される
- 文字の計算
  - 「文字」に +1 (次の文字になる) や -1 (前の文字になる) もできる
    - 詳しくはまた、後日

条件分岐(再)

- 引数の内容によって振舞いを「大幅」に変更したい
  - if 文と strcmp 関数を利用して対応できる
    - strcmp 関数：二つの文字列を比較する
  - if (!strcmp (A, B)) { X } else { Y }
    - A と B が同じなら X を、そうでなければ Y を行う
  - 「else if」を使うと更に複数の命令が選べる
    - if (C1) { P1 } else if (C2) { P2 } .. else { Pn }
    - C1 の時 P1、そうでなく C2 の時は P2 .. いずれでもないと Pn
  - おまじない
    - #include <string.h>
  - strncmp (A, B, N);
    - A と B の先頭の N 文字だけを比較する
    - !strncmp ("abc", "abz", 3); : 等しくない
    - !strncmp ("abc", "abz", 2); : 等しい

再帰呼び出し

- 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

- 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

- 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、後は「組み合わせ」を考えるだけ !!

make と makefile

- make
  - プログラム作成作業の自動実行をおこなってくれる
    - makefile に作成作業の内容を記述すると、必要最低限の作業を自動実行する
    - make を実行すると、そのフォルダにある makefile (Makefile) を参照する
- makefile
  - make に作成作業を指示するためのファイル
- makefile の形式
  - コメント：'#' から行末(改行)までは、コメントとして無視される
  - マクロ：文字列に名前が付けられる
    - マクロ定義：「マクロ名」「=」「定義する文字列」
    - マクロ参照：${「マクロ名」}
  - 依存関係の定義：「目的物」を「材料」を使って「作り方」で作るように指示
    - 構文 (<TAB> の所には [Tab] キーの入力が入る)

      「目的物」:「材料」

      <TAB>「作り方」の手順 1

      <TAB>「作り方」の手順 2

      ...

      <TAB>「作り方」の手順 n