ソフトウェア概論 A/B

-- 繰返し(再起呼出し) --

数学科 栗野 俊一 / 渡辺 俊一

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 : 20140509
  - 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(2014/05/02)の復習

- 前回(2014/05/02)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

お知らせ

- 本日(2014/05/09)の予定
  - 文字列と文字の関係
  - 「再帰呼び出し」をしてみよう
  - 亀プログラム(再挑戦)
  - ミクを歩かせよう
- 本日(2014/05/09)の目標
  - 再起呼出しを利用した「繰返し」を学ぶ
  - 演習
    - 亀プログラム(再挑戦)
    - ミクを歩かせよう
    - 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
    - 課題の提出

前回 (2014/05/02) の課題

- 前回 (2014/05/02) の課題
  - 課題 2:
    - ファイル名 : 20140502-2-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)

本日 (2014/05/09) の課題

- 本日 (2014/05/09) の課題 (CST Portal のみ)
  - 課題 3:
    - ファイル名 : 20140502-3-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
    - 前回(2014/05/02)の課題の 3 なので、「20140502-3-QQQQ.c」である事に注意

引数付き関数の作り方(復習)

- □ 引数とは
  - ○ 関数に与える事により、関数にその引数に対応した挙動をさせるもの
    - ▷ 引数付き関数の定義：引数の値によって挙動が変る
    - ▷ 引数付き関数の利用：指定したい挙動をさせるための値を指定する
    - ▷ cf. 三角関数：引数の角度によって異なる値を返す
- □ 引数付き関数の作り方
  - ○ 似ている二つ関数を一つの引数付き関数にまとめる
    - ▷ 関数の本体の部分を、同じ部分と違う部分に分ける
    - ▷ 違う部分は「変数」に置き換えて、一つの関数定義にまとめる
    - ▷ 関数の仮引数の所に、「変数」を追加する
    - ▷ 呼出す側は、実引数に、「違っていた部分の内容」を指定する

条件分岐(復習)

- 引数の内容によって振舞いを「大幅」に変更したい
  - 小幅の場合は、「引数」で対処
  - if 文と strcmp 関数を利用して対応できる
    - strcmp (A, B) 関数 : 二つの文字列 A, B を比較する
    - !strcmp ("abc", "abc") : 等しい
    - !strcmp ("abc", "abz") : 等しくない
  - 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) 関数 : 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

再帰呼び出しとは

- 関数呼出し (Function Call)
  - 関数呼出しとは
    - 作成: 関数(Function)に引数(Argument)を与え、その機能(Function)を呼び出す事
    - 表現: 関数名の後ろに引数を並べて「(」と「)」で囲った物
    - 事例: printf("Hello, World\n") / strcmp ("abc", X)
  - 参考
    - 関数は「命令列に名前(関数名)を付けた」物(その命令列は関数の「本体」と呼ぶ)
    - 名前を利用して「命令列」を呼び出す事ができる(関数呼出し)
- 関数呼出しの階層構造
  - 関数の本体で、更に関数を呼び出す事ができる
    - その関数の中で更に関数を.. (階層構造になる..)
- 再起呼出し
  - ある関数(の本体)の中で、(直接、あるいは間接的に..) その関数を呼び出す事
    - 関数呼出しの関係に「輪(Loop:ループ)」ができる
    - 再起呼出しが現れないプログラムは、「木構造」になる

再帰呼び出し利用法

- □ 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、直接、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
    - そうしないと「底が抜けて」しまう

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
  - 任意のプログラムは、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 !!

文字列と文字

- C 言語での「文字」の扱い
  - 文字：文字をシングルクォート(「’」)ではさんだもの
    - cf. ’A’, ’a’, ’1’
    - 「一文字」と対応している「表現」
  - 文字の出力：putchar 関数を利用する
    - cf. putchar (’A’);
  - 文字列：文字列をダブルクォート(「"」)ではさんだもの
    - cf. "ABC", "123", "" (空文字列)
    - 「文字の並び(0個以上)」と対応している「表現」
    - <<注意>>：全角(日本語)は、一文字で、二文字分になる
  - 文字列の出力：printf 関数を利用する
    - cf. printf ("abc");
- 引数変数宣言における文字列と文字の区別
  - 文字の値を持つ変数の場合：「char」を使う
  - 文字列の値を持つ変数の場合：「char *」を使う
    - 「char * X」の意味は、「 X に * を付けて (*X) にすると char になる」

文字列の構造

- 文字列は、文字の並び + 文字の終りからなる
  - "ABC" == { 'A', 'B', 'C', '\0' }
    - 長さ 3 (n) の文字列は、4 (n+1) つの部分からなる
    - '\0' を EOS (End Of String) と呼ぶ
  - k 番目の文字の取り出し方 : [k] をつける (k は 0 から始まる事に注意)
    - cf. "ABC"[0] == 'A', "ABC"[3] == '\0', "ABC"[9] == ? (未定義)
  - 先頭の文字を取り出すには * を付けてもよい
    - cf. *"ABC" == 'A'
  - 先頭の文字を取り除いた残りを取り出すには 1 を加えればよい
    - cf. "ABC" + 1 == "BC"
- 文字列の判定 : strcmp を使うと、「二つの文字列が同じなら偽」になる
  - cf. strcmp ("ABC", "ABC") ==> 偽, strcmp ("ABC", "XYZ") ==> 真
  - 文字列が「空文字列(長さが 0 / 文字を含まない文字列)」は先頭が EOS かどうかで判定できる
    - (*"" == '\0') ==> 真