ソフトウェア概論 A/B

-- 繰返し(再起呼出し) [2] --

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 : 20170526
- 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(2017/05/19)の復習

- 前回(2017/05/19)の内容
  - 「再帰呼び出し」をしてみよう
    - 再起呼出しを利用した「繰返し」を学ぶ
  - 繰返しを含む亀プログラム

お知らせ

- 本日(2017/05/26)の予定
  - ミクを歩かせよう
  - 文字列と文字の関係
  - 様々な「再帰呼び出し」をしてみよう
  - 分割コンパイル
  - 整数型
- 本日(2017/05/26)の目標
  - 再起呼出しを利用した「繰返し」を学ぶ
  - 演習
    - 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 (2017/05/19) の課題

- 前回 (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 : 入れ子になった四角を書く関数を作りなさい
    - 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 (2017/05/26) の課題

- 本日 (2017/05/26) の課題
  - 課題 20170526-01:
    - ファイル名 : 20170526-01-QQQQ.c (QQQQ は学生番号)
    - 内容 : 与えられた文字列を逆順に出力する rprintf を定義しなさい
    - 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 20170526-02:
    - ファイル名 : 20170526-02-QQQQ.c (QQQQ は学生番号)
    - 内容 : 出力する繰返し回数を整数で指定する ntimeprint を作りなさい
    - 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 20170526-03:
    - ファイル名 : 20170526-03-QQQQ.c (QQQQ は学生番号)
    - 内容 : 負の整数も処理できる printint 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)

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
  - 本日のフォルダ(c:\usr\c\20170526)を作成
  - 次の本日 (2017/05/26) 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/20170526/20170526.html
  - ダウンロードするファイル
    - 20170526-update.zip (c:\usr\c に保存し、展開する)
    - miku.zip (c:\usr\c\20170526 に保存し、展開する)
- ファイルをダンロードしたら次の作業を行う
  - c:\usr\c\20170526-update の中身(lib,include)を c:\usr\c に移動する
    - 「上書きするか ?」と尋ねられるので「はい」と答える
  - ubuntu 側で ~/c/20170526/miku に移動する
    - cd ~/c/20170526/miku
  - 「make test」で実行して、ミクが動けば良い

再帰呼び出し (再)

- 文字列を先頭から順番に一文字ずつ見て行く
  - 「"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(長さ 0 に)なったかは、空文字列(「""」)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
- 再帰呼び出しと数学的帰納法
  - 再帰呼び出しは、数学的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、数学的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 (数学的帰納法と同じ)
    - 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)

- 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 !!

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

分割コンパイル

- C 言語で記述されたプログラムの構造
  - main 関数が必ず必要
    - 他の関数は main 関数から呼び出される
  - 関数の定義
    - ソースファイル (*.c) の中に記述する
    - 同じファイル内である必要はない
- 分割コンパイル
  - 関数を別のファイルで定義し、個々にコンパイルする事
    - 後でリンクにより一つの実行ファイルにまとめる
- 分割コンパイルの例
  - foo.c と bar.c から foobar.exe を作る場合
    - cc -c foo.c : foo.o が作られる
    - cc -c bar.c : bar.o が作られる
    - cc -o foobar.exe foo.o bar.o : foo.o/bar.o から foobar.exe が作られる
- extern 宣言
  - 他のファイルで定義されている関数を利用する場合に必要
    - 使う側の関数の前で extern 宣言する
    - extern 宣言の形式 :「extern」+「関数の頭部」+「;」

整数型

- □ 整数型
  - ○ C 言語でも整数(の一部)が扱える
    - ▹ 整数型 : -2^31 ～ 2^31 - 1 まで範囲の整数
  - ○ 整数の計算
    - ▹ 四則 (和: +, 差: -, 積: *, 商: /, 余り: %) の計算ができる
  - ○ 整数の比較 (if 文で利用する)
    - ▹ 等価 (等しい: ==, 等しくない: !=)
    - ▹ 大小比較 (大なり: >, 小さいなり: <, 以下: <=, 以上: >=)
  - ○ 整数の出力
    - ▹ sample-015.c の printint を利用する
    - ▹ ※ 他にも色々できるのだが、しばらくは触れない
- □ 整数の引数
  - ○ 関数の引数として、整数も指定できる
    - ▹ これまでは文字列だけだった
- □ 整数の引数を持つ関数の宣言
  - ○ 引数(変数)の前に「int」と記述する
    - ▹ cf. これまでは「char *」と記述していた