ソフトウェア概論 A/B

-- 関数 again --

数学科 栗野 俊一 / 渡辺 俊一

2014/05/30 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
- 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 : 20140530
  - 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(2014/05/23)の復習

- 前回(2014/05/23)の内容
  - 2 周目の開始
    - 1 周目の内容を再確認しながら、新しい事を説明する
  - 「Hello, World」again：様々な謎解き(一部)
    - 「main 関数」の定義
    - 「#include」の意味：ファイルを読み込む (printf の extern 宣言)
    - 「int ～ return 0;」:「void」との関係
  - 入力
    - プログラムに(プログラムが動き出して[実行時]から..)情報を与える仕組
    - cf. これまでは、「プログラム作成時」に情報を与えていた
    - 「getchar()」は、「キーボードから一文字入力」する関数
  - Input-Process-Output：入力した情報を加工して出力する
    - プログラムの基本設計構造

一周目の内容(再)

- 表現
  - プログラムの基本的な書き方(Hello World)
    - 幾つかのライブラリ関数の利用：printf, putchar, strcmp
  - 関数の作り方(特に、引数付き関数)
  - 幾つかのデータ型とその扱い
    - char：文字 / char *：文字列 / int：整数
- 操作
  - コンパイル、リンクの仕方 / make / makefile / 分割コンパイル
- 作成
  - 命令を組合せる三つの手段(この三つで「万能」になる)
    - 順接：命令を並べると、その順に実行される
    - 条件分岐：if 文で、条件によって二つの命令の一方だけを実行する
    - 再帰呼出し：関数内で自分を呼び出す事により、間接的に命令を繰り返す

三つの基本制御構造と万能性(再)

- 三つの基本制御構造：命令を「組合せて」新しい命令を作り出す仕組
  - f を関数, A,B を命令、p(x) を条件とする時、A,B から f を作る仕組
  - [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 !!

お知らせ

- 本日(2014/05/30)の予定
  - 表現
    - 制御構造
    - 関数(作成方法/引数)
    - データ型
    - 返り値と return 命令 (新)
- 本日(2014/05/30)の目標
  - 講議
    - 関数と三つの制御構造(順接/分岐/繰返[再帰])
    - 関数の値
  - 演習
    - 値を返す関数
    - 課題の提出

前回 (2014/05/23) の課題

- 前回 (2014/05/23) の課題
  - 課題 1:
    - ファイル名 : 20140523-1-QQQQ.c (QQQQ は学生番号)
    - 内容 : キーボードから一文字入力し、その文字によって異る国の挨拶をする
    - 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:
    - ファイル名 : 20140516-03-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)
    - 注意 : 前々回(2014/05/16)の課題なので、名前が 20140516 である事に注意
- 注意 : 前回(2014/05/23)の課題 2 は、今回(2014/05/30)に回す

本日 (2014/05/30) の課題

- 本日 (2014/05/30) の課題 (CST Portal のみ)
  - 課題 1:
    - ファイル名 : 20140530-1-QQQQ.c (QQQQ は学生番号)
    - 内容 : 二つの整数の積を返す関数
    - 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)
  - 課題 2:
    - ファイル名 : 20140530-2-QQQQ.c (QQQQ は学生番号)
    - 内容 : 自然数の階乗を返す関数
    - 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)
  - 課題 3:
    - ファイル名 : 20140530-3-QQQQ.c (QQQQ は学生番号)
    - 内容 : 二つの非負の整数の最大公約数を返す(ユークリッドの互除法)
    - 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 4:
    - ファイル名 : 20140523-02-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)
    - 注意 : 前回(2014/05/23)の課題なので、名前が 20140523 である事に注意

「関数」という考え方 (復習)

- 関数の定義とは(What)？
  - 「プログラムの一部」に「名前」を付ける事
    - 「名前」を「関数名」と呼ぶ
    - 「プログラムの断片」を「関数の本体」と呼ぶ
- 関数をどうやって利用する(How to)？
  - 「関数名」を指定するだけで「関数本体」が実行される(関数呼出し)
- 関数を定義する理由は (Why)？
  - 「プログラムの断片」に「名前」が付けられるので、分かり易い
    - もちろん、「断片の内容に対応した分り易い名前をつければ..」だが..
  - 「関数名前」を指定するだけで「関数本体」が実行される
    - 何度も同じ事をする場合に便利(プログラムが短くなる)
  - 「引数」を利用する事により「色々な断片」を「一つの関数本体」にまとめられる
    - 何度も似たような事をする場合に便利(プログラムが短くなる)
  - 一箇所の「関数本体」を直すだけで、多数の場所の命令を直す効果がある
    - 「コピペ」がバグの増殖を促す

「関数」の表現方法 (復習)

- 関数定義(の文法)
  - 「関数定義」は、「関数頭部」と「関数本体」に分けられる
  - 「関数頭部」は、「関数宣言」「関数名」「仮引数宣言」に分けられる
    - 「関数宣言」は、void(これまで)/int(main だけ)
    - 「関数名」は、自由にきめてよい(他と重複すると駄目だが..)
    - 「仮引数宣言」は、「(」+「仮引数宣言並び」+「)」
    - 「仮引数宣言並び」は、「void」か、「char *変数名」のカンマ(,)区切
    - 「関数本体」は、「{」+「命令列」+「}」
- 関数呼出し(の文法)
  - 「関数呼出し」は、「関数名」+ 「実引数並び」
  - 「実引数並び」は、'()' か、'(' + 「式」のカンマ並び + ')'

文字を引数に持つ関数と型宣言 (復習)

- これまでの関数
  - 引数がないか、文字列を引数としていた
    - 「char *」をお呪いとし、関数を呼び出す時に、文字列を指定
    - 変数には文字列が入っているとして、考える
- 文字を引数に持つ関数の場合
  - 引数宣言に「char」とする必要がある
- 型宣言
  - 「char *」/「char」は実は、「引数の型」を表現していた
    - 「char *」は「文字列」
    - 「char」は「文字」
  - 変数に、その型と異る値を入れようとすると「エラー」になる
- 「型」と「演算」
  - 「文字」に「1 を加える」と、「次の文字」
  - 「文字列」に「1 を加える」と、「短くなった文字列」
    - 同じ「1 を加える」という「演算」でも、「意味」が異る
  - 「演算」と「型」は「一組」で考える必要がある

s_input.h, s_print.h

- □ s_input.h, s_print.h
  - ○ 色々な型のデータの入出力を行う関数を利用するための「御呪い」
    - ▹ 使い方 : stdio.h と同様 include する
    - ▹ cf.「#include "s_print.h"」(「#include <s_print.h>」とはしない)
  - ○ 利用可能な関数
    - ▹ s_print_char(char ch) 文字(char 型)の出力 (putchar と同様)
    - ▹ s_print_string(char *str) 文字列(char * 型)の出力 (printf と同様)
    - ▹ s_print_int(int i) 整数(int 型)の出力 (printint と同様)
    - ▹ s_print_newline() 改行の出力 (putchar ('\n') と同様)
    - ▹ s_input_int() 整数値の入力
    - ▹ s_input_char() 文字の入力 (getchar() と同様)
    - ▹ s_input_string() 文字列の入力
- □ 準備
  - ○ s_input.h, s_output.h を c:/usr/c/include に置く

関数の返り値と return 命令

- □ void 型関数
  - ○ 関数の前に void が付けらている
    - ▹ 実は、C 言語でも、「特別(void 型)」な「関数」
    - ▹ cf.「数学の関数」とは違う
  - ○「数学の関数 f(x)」: x として f に何かを与えると f(x) という値が得られる
    - ▹ cf. f(x)=x^2 なら f(3)=3^2=9, sin(π/6)=1/2
- □ 非 void 型関数
  - ○ 関数の前に「関数値の型」が付ける
    - ▹「数学の関数」と同じように「値(返り値)」を計算して「返す」事ができる
    - ▹ cf. int f(int x) : 整数型の値を与えると整数型の値を返す関数
- □ 値を返す関数の作成方法
  - ○ 関数の前の型宣言は、関数の返す値の型を書く
  - ○ 関数が返す値は、return 命令の後ろに書く
    - ▹ return 命令が実行されると、その時点で関数が終了する事に注意