ソフトウェア概論 A/B

-- 関数 again --

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 : 20150605
  - 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

s_print.h/s_input.h

- □ s_print.h/s_input.h
  - ○ 文字(char)、整数(int)、文字列(char *)の入出力を行う関数を使う場合に include
    - ▹ void s_print_char(char ch) : 文字 ch を出力する (putchar と同じ)
    - ▹ void s_print_string(char *string) : 文字列 string を出力する (printf と同じ)
    - ▹ void s_print_int(int i) : 整数 i を出力する (printint と同じ)
    - ▹ void s_print_newline() : 改行の出力 (putchar ('\n') と同様)
    - ▹ char s_input_char(void) : 文字を入力して値として返す (getchar() と同じ)
    - ▹ char *s_input_string() : 文字列を入力して値として返す
    - ▹ int s_input_int(int i) : 整数を入力して値として返す
- □ ファイルのダウンロードと配置
  - ○ 次の二つのファイルのダウンロードし、c:\usr\c\include に保存する
    - ▹ s_print.h
    - ▹ s_input.h
- □ 確認
  - ○ sample-016.c ～ sample-018.c を c:\usr\c\20150605 に保存
  - ○ 以下のコマンドで、sample-016.c の確認 (017, 018 も同様)
    - ▹ cc -I../include sample-016.c　　!!「-I../include」が必要
    - ▹ cc -o sample-016.exe sample-016.o
    - ▹ ./sample-016.exe

前回(2015/05/29)の復習 1

- □ 前回(2015/05/29)の内容
  - ○ 2 周目の開始
    - ▹ 1 周目の内容を再確認や謎解きをしながら、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「処理(加工)」して「出力」する
    - ▹ プログラムの基本設計構造

前回(2015/05/29)の復習 2 :「入力」の意義

- 「入力」以前
  - プログラムの動作は、全て、プログラムの「作成時」に決っていた (静的:static)
    - (基本..) プログラムの動作(実行結果)は、毎回同じ
- 「『入力』が有る」事の意義
  - 「入力」の内容によって、プログラムの動作を「実行時」に変更できる (動的:dynamic)
    - 一つのプログラムで、(「入力」という状況に応じた)様々な複数の動作(機能)が得られる
    - プログラムが「使える」状況が増える (一粒で N 度美味しい ??)
  - cf.「1+1」の計算をする(常に「2」を求める)プログラムには意味がない
    - 二つの数を入力して「その二つの数の足し算をする」プログラムなら意味がある
- 「入力」と「関数の引数」
  - 「関数の引数」は、「関数に対する入力」と考える事ができる
    - 引数の *ない* 関数 / 入力の *ない* プログラム → あまり役に立たない
    - 引数の *ある* 関数 / 入力の *ある* プログラム → 使い道が多い
- 「決定の遅延」という考え方
  - 「プログラム」は記述すると「固定」される → 柔軟性がなくなる
    - 「優柔不断」の勧め :「決める」のはできるだけ後にした方がよい

お知らせ

- 本日(2015/06/05)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(2015/06/05)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 (2015/05/29) の課題

- 前回 (2015/05/29) の課題
  - 課題 20150522-01: (前々回[2015/05/22]の課題 01)
    - ファイル名 : 20150522-01-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 20150522-02: (前々回[2015/05/22]の課題 02)
    - ファイル名 : 20150522-02-QQQQ.c (QQQQ は学生番号)
    - 内容 : キーボードから一行(改行まで..)文字列を読込み、それを逆順に出す
    - 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 (2015/06/05) の課題

- 本日 (2015/06/05) の課題
  - 課題 20150605-01:
    - ファイル名 : 20150605-01-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 20150605-02:
    - ファイル名 : 20150605-02-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 20150605-03:
    - ファイル名 : 20150605-03-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)

「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、自由に決めて良い(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

データとコード

- □ データ：値を持つもの
  - ○ 即値(定数値/定数表現), 変数, 関数呼出し, 式
    - ▹ 関数の実引数として渡せる物
    - ▹ 型を持つ / 計算できる / 入出力できる / 関数の値として返せる
- □ コード：値を操作するもの
  - ○ (今の所は..) 関数呼出し + ';' (しか、学んでいない)
    - ▹ ※ 直に、コードの別の例として、「return 命令」を学ぶ
    - ▹ 関数の本体部分に記述するもの
    - ▹ 関数の実引き数としては、渡せない

構文 : (単純な)コードから(複雑な)コードを作る仕組

- ○ 順接 (+ ブロック) / if 構文
- □ 関数定義：コードに名前を付ける仕組
  - ○ 「関数を定義する」事により、その関数の「関数呼出し」が可能となる
    - ▹ 関数は予め定義して(コンパイルして)置かないと、利用できない
    - ▹ printf 等の libray (ライブラリ)関数は、既にコンパイル済の物を利用している

関数の返り値と return 命令

- □ void 型関数
  - ○ 関数の前に void が付けられている
    - ▹ 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