ソフトウェア概論 A/B

-- 整数型/コマンドライン引数 --

数学科 栗野 俊一 / 渡辺 俊一 (TA: 栗原 望 / 小嶋 仁子 [M2])

2018/06/15 ソフトウェア概論

伝言

私語は慎むように !!

- 出席パスワード : 20180615
- 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
- VNC Server Address : 10.9.154.38
  - Password : vnc-2017
- 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
- 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(2018/06/08)の復習

- 前回(2018/06/08)の内容
  - 「繰り返し」の復習：再帰呼出しする関数の定義は「繰返し」を表現
    - 命令の組み合わせ：順接/条件分岐/繰返し、これがあれば「万能」になる
  - 「文字」の扱い：C 言語では「文字列」以外に「文字」が扱える
    - 文字のリテラル(文字自身の表現)：「’」+「一文字」+「’」
    - 例：「’A’」という表現は、「A」という「文字」を表す
    - 文字の入出力：getchar/putchar 関数で、文字が入出ができる

お知らせ

- 本日(2018/06/15)の予定
  - ハノイの塔(前回の残り)
  - 整数型
  - コマンドライン引数
  - エラー処理と exit 関数

今週 (2018/06/15) の課題

- 今週 (2018/06/15) の課題
  - 課題 20180615-01
    - ファイル名 : 20180615-01-QQQQ.c (QQQQ は学生番号)
    - 内容 : コマンドライン引数で指定した長さのハノイの塔を解く
    - 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 20180615-02
    - ファイル名 : 20180615-02-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 20180615-03
    - ファイル名 : 20180615-03-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
    - ヒント: sample-014-01.c, sample-015-01.c は利用して良い
    - 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)

先週 (2018/06/08) の課題

- □ 先週 (2018/06/08) の課題
  - ○ 課題 20180608-01
    - ▹ ファイル名 : 20180608-01-QQQQ.c (QQQQ は学生番号)
    - ▹ 内容 : 高さ 3 のハノイの塔を解く
    - ▹ 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 20180601-01: (先々週 [2018/06/01]の課題だが、先週 [2018/06/08] に回す)
    - ▹ ファイル名 : 20180601-01-QQQQ.c (QQQQ は学生番号)
    - ▹ 内容 : 与えられた文字列の文字を二度ずつ出力する関数を作成する
    - ▹ 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 20180601-02: (先々週 [2018/06/01]の課題だが、先週 [2018/06/08] に回す)
    - ▹ ファイル名 : 20180601-02-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関数を作成する
    - ▹ 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)

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 (2147483647) まで範囲の整数 (cf. limits.h)
  - 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

文字型の「計算」

- 文字型のデータの計算
  - 「文字」は、C 言語内では、「小さな整数値(-128～127)」として振る舞う
    - 整数型と同様に四則が計算できる
  - 文字と整数値の関係：ASCII Code 表を利用して「コーディング」されている
    - 例 : 'A' -> 65, 'B' -> 66, 'a' -> 97, 'b' -> 98, '0' -> 48
  - 整数値なので、「計算」ができる
    - 例 : 'A' + 1 => 65 + 1 => 66 => 'B'
  - 整数値なので、「比較」ができる
    - 例 : 'A' < 'B', 'A' == 'A'

コマンドライン処理

- 関数 main の引数
  - これまでは、「int main(void) {」としてきた
  - 本当は「int main(int argc, char *argv[]) {」と書く事もできる
    - 実は、「int main(int argc, char *argv[], char *environment) {」とも書ける
- プログラムの実行方法 (コマンドライン引数の指定)
  - これまで : ./foobar.exe としてきた
  - 実は : ./foobar.exe abc 123 ... と「実行時」に「コマンドライン引数」が指定できる
    - main 関数の中で argv[1] とすると、一つ目のコマンドライン引数 "abc" が取りだせる
    - 以下、argv[N] (N=1,2,...) とすると、N 番目のコマンドライン引数がが取りだせる
    - 実は argv[0] には、コマンド名 (./foobar.exe) が入っている

exit 関数

- exit 関数：プログラムの強制終了を行う
  - 機能：exit 関数が呼び出されると、プログラムはその時点で「終了」する
    - 残りの命令は実行されない
  - 使い方：exit (終了情報); (#include <stdlib.h> が必要)
    - 「終了情報」は整数値で与え、プログラムを呼出した側に渡される
    - 0：正常終了 / 0 以外：異常終了
  - 利用例
    - 全ての仕事が終了した (例：ゲーム等で「終了」ボタンを押した)
    - エラーが生じて、仕事が続けられない (例：誤った操作が行われた)
  - エラーで終了する場合は、エラーメッセージも出力する
    - エラーコードは、make や shell 等で参照(「$?」)できる
- main の return 命令の役割
  - main 関数で return の後ろに指定していた数
    - 実は exit の引数と同じ「終了情報」だった
    - 通常は「正常」なので「0」を指定していた

データ型と引数変数の型宣言

- □「型」とは(what)：数学の『空間』と同じような意味
  - ○「値集合」と、それらを対象とした「演算子」(群)の対
    - ▷例 1. n 次ベクトル空間 => n 次ベクトルの集合と、定数倍、和、内積、..
    - ▷例 2. 整数型 => 整数値の集合と、入出力、四則、比較、..
    - ▷例 3. 文字型 =>「文字」の集合と、入出力、比較、整数の加減算
    - ▷例 4. 群 => 集合と積
    - ▷例 5. 実数空間 => 実数全体の集合と、四則、比較、極限
    - ▷例 6. 関数空間 => 関数の集合と、定数倍、四則、微分、積分
  - ○「演算記号(演算子を表現する記号: +, -, *)」の「意味」
    - ▷「演算記号」は、何れかの「型」の「演算子」を表現する
    - ▷同じ「演算記号」でも、「型」が異れば、異る「演算子」になる
    - ▷例 : "3" + 1 => "", 3 + 1 => 4, ’3’ + 1 => ’4’
- □引数変数と型
  - ○「引数変数」は「値」(「値集合」の要素)を持っている
    - ▷「値」は、「型」を持っている(「値」はいずれかの「型」の「値集合」の要素)
    - ▷「引数変数」は、(それが保持する「値」に対応した)「型」を持つ(型宣言)