

ソフトウェア概論 A/B

-- Hello, World 再び --

数学科 栗野 俊一 / 渡辺 俊一

2013/06/14 ソフトウェア概

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>

□ 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

前回 (2013/06/07) の復習

□ 前回 (2013/06/07) の復習

前回は、car game を作った

○ 幾つか、新しい事も触れた

□ 文字の表示効果

○ putchar ('\r') : カーソルを行頭に移動する

○ putchar ('\n') : カーソルを次の行の行頭に移動する

▷ 現在行が画面の下端の場合は、画面がスクロールする

□ 文字のリアルタイム入力

○ _kbhit : 文字が入力されていれば、if 文でチェックできる

○ _getch : _kbhit() でキーが押されていた場合に、そのキー

□ 乱数の発生 : rand() で、乱数を得る事ができる

□ 余りの計算 : 数値の余りの計算は '%' で行う

□ プログラムの終了 : exit(0) でプログラムが終了できる

□ 文字列の計算 : 数を加えるだけでなく、引く事もできる

お知らせ

□ 本日の予定

- Hello, World(再)
- 入力と出力(再)
- 関数(再)
- 整数型
- 引数の型宣言(再)

□ 本日の目標

- 演習
 - ▷ 課題の提出

前々回の課題 (2013/05/31)

□ 前回 (2013/06/07) の課題

- 前は cargame の話だったので、課題はなかった

□ 前々回 (2013/05/31) の課題

○ 課題 20130531-01

- ▶ キーボードから一文字入力し、その文字によって異なる国の挨拶をする
- ▶ ファイル名 : 20130531-1-XXXX.c (XXXX は学生番号)
- ▶ 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 20130531-02

- ▶ キーボードから一行(改行まで..)文字列を読み込み、それを逆順に出す
- ▶ ファイル名 : 20130531-2-XXXX.c (XXXX は学生番号)
- ▶ 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)

本日の課題 (2013/06/14)

□ 本日 (2013/06/14) の課題

○ 課題 1:

- ▶ ファイル名 : 20130614-1-YYYY.c (YYYY は学生番号)
- ▶ 内容 : コマンド引数に指定した言語名よって挨拶を行う
- ▶ 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)

"Hello World" 再び

□ "Hello World" 再び

○ hello.c

- ▶ 単純だが完全に動く意味のあるプログラム
- ▶ プログラムを作る場合の雛形として利用できる(これを書き換えてプログラムを作る)

□ main 関数

○ C 言語のプログラムには必ず一つ必要

○ 最初に呼び出される : コマンドラインから引数を与えられる

- ▶ 「int argc, char *argv[]」

○ 他の関数を呼び出し、様々な「命令」を実行する

○ 結果を返す : if errorlevel で「結果」が利用できる

- ▶ 「return 0;」

コマンドライン引数

- プログラムに引数を与える
 - コマンドラインに、プログラム名の後ろに文字列を複数指定
 - ▶ 文字列は、空白で区切られる
 - ▶ 引数を「"」(ダブルクォーテーション)で括めば、空白を入れる事ができる
 - コマンドライン引数は、`main` 関数の引数となる
 - ▶ `argc` - コマンドライン引数の個数 + 1
 - ▶ `argv[]` - コマンドライン引数の文字列 (`argv[1]` が一つ目の引数)

main 関数の返値

□ main 関数の最後の「return 0;」

- main 関数の返り値として、「0」を指定している

 - ▶ 別に 0 でなくても良いのだが..

- main 関数の返り値は、コマンドのエラーコードを表すのが普通

 - ▶ 0 を返すと、正常 / 0 以外を返すと、なんらかの異常が基本ルール

□ main 関数の返り値の利用

- main 関数の返り値は、batch ファイルの `errorleve` でチェックできる

"Hello World" 再び (3)

□ 「#include <stdio.h>」とは？

- 「#include」: 他のファイルの内容を「ここに読み込め」という命令
 - ▶ 自分でコピーペースとしなくてもよい
 - ▶ 自分で指定すれば、任意のファイルを include できる
- 「stdio.h」: STanDard Input Output library Header
 - ▶ 標準入出力関連ライブラリーに関する宣言が記述されている
 - ▶ ファイルのある場所は C:\MinGW\include
- printf/putchar (文字列/文字の出力関数) の為に必要

関数(再び)

□ 関数

○ 関数とは => 命令列に名前を付けたもの

- ▶ 関数定義時：自分で、命令列に名前を付けて、新しい関数が作れる
- ▶ 関数実行時：名前を指定する(関数呼び出し)だけで、その命令列が実行できる

○ 引数付き関数とは => 命令の一部を仮引数で表現した関数

- ▶ 関数定義時：命令の一部を仮引数(変数)で表現し、そこでは内容を指定しない
- ▶ 関数実行時：仮引数に入る内容は関数呼び出し時に「実引数」で指定する

○ ライブラリ => 他の人が作った関数定義の集まり

- ▶ 中身を知らなくても、機能が解れば、関数を呼び出す事で利用可能
- ▶ cf. printf, s_midi_play, ..

文字(再)

□ 文字とは？

○ シングルクォーテーション('')で挟まれた一つの「文字」

- ▶ cf. 'a' は「a」という文字、'0' は「0」という文字
- ▶ '\'(エンマーク/バックスラッシュ)はメタ文字(特別扱いの文字)
- ▶ '\n' は改行をする文字、'\t'はタブ、'\a' は音を鳴らす文字
- ▶ '\\' は \' 自身を表す、\'\"で\''を表現できる

□ 文字の画面(標準出力)への出力

○ 「putchar (文字)」で、「文字」を出力できる

- ▶ putchar ('a') で「a」が出力される
- ▶ putchar ('\a') で音になる

文字列(再)

□ 文字列とは？

○ ダブルクォーテーション(「"」)で挟まれた「文字」の並び

- ▶ 「\」(エンマーク/バックスラッシュ)はメタ文字(特別扱いの文字)
- ▶ 「\n」は改行、「\t」はタブ、「\a」は音を鳴らす
- ▶ 「\\」は「\」自身を表す、「\"」を使えば「"」を表現できる

□ 文字列での「計算」

○ 「+1」: 文字列に 1 を加える : 先頭の文字が取り除かれる

- ▶ "abc" + 1 は "bc" と同じ(ように振る舞う)

○ 「頭に * を付ける」文字列の先頭の「文字」を取り出す

- ▶ *"abc" は 'a' と同じ(ように振る舞う)

○ 「後ろに「[数値]」を付けると「数値番目の文字」が取り出せる

- ▶ "abc"[2] は 「*("abc" + 2)」と同じで 'c' となる

○ 文字列は「文字」の並びで、最後に '\0' (EOS : End Of String) がある

- ▶ "abc" は 'a', 'b', 'c', '\0' の四つの「文字」が並んだもの
- ▶ "" (空文字列) は、EOS 一つからなる

関数と引数(再)

□ 関数宣言の頭部

- 「void 関数名 (char * 仮引数 [, char * 仮引数 ..])」

- ▶ 関数の前には「void」を付ける (お呪い)

- ▶ 仮引数の前には「char *」を付ける (お呪い)

- 仮引数 : 「文字列」の代わりに利用できる「変数」

□ 関数宣言の体部

- 「 { 命令列 } 」

- ▶ 命令列の中で「文字列」が表れて良い所に「変数」が利用できる

□ 「char *」とは何か？

- 引数の「型」宣言 (「文字列」を表現する『もの』)

- ▶ 引数に何が入るか(情報の種類)を指定している

- 引数に入る情報の種類によって、「型の宣言」も変更する必要がある

- ▶ 今迄は「文字列」しか扱わなかったので「char *」固定だった

- ▶ 他の種類のデータを処理するなら対応して変更する (cf. char)

□ 関数の引数の型の一致

- 仮引数(の変数)の型と、実引数(の値)の型は一致している必要がある

整数型

□ 整数型

○ C 言語での整数

- ▶ 表現できる範囲は限られている
- ▶ 32bit の場合は -2147483648 から 2147483647

○ 宣言 : `int` で行う

○ 計算 : 四則が可能 `+`, `-`, `*`, `/`

- ▶ `/` は整数割り算なので、小数点以下は切捨てになる

○ 比較 : 大小比較、等号、不等号が使える

- ▶ `a > b` : `a` が `b` より大きい
- ▶ `a >= b` : `a` が `b` 以上
- ▶ `a == b` : `a` と `b` が等しい (= でないことに注意 !!)
- ▶ `a != b` : `a` と `b` が等しくない

□ 整数型の出力 (当分は..)

- `s_print.h` の中の `s_print_int` を使う (`sample-001.c`)
- `s_print_string` で文字列が出力できる
- `s_print_newline` で、改行