

ソフトウェア概論 A/B

-- Hello World again --

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

2018/06/22 ソフトウェア概

伝言

私語は慎むように !!

- 出席パスワード : 20180622
- 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>
- 廊下側の一行は遅刻者専用です(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(2018/06/15)の復習

- 前回(2018/06/15)の内容
 - 整数型
 - ハノイの塔

お知らせ

- 本日(2018/06/22)の予定
 - コマンドライン引数(前回の残り)
 - エラー処理と `exit` 関数(前回の残り)
 - 「Hello World」again

一周目の取り零し

□ 一周目

○ 再起の話で、とりあえず、一通りの事は話した事になる(一周目)

▶ 二周目に入りたい (今までの「内緒/お呪い」をなんとかする)

▶ 一周目の復習をしつつ、もう一度始めから

□ 一周目の取り零し

○ 2018/06/08 の内容が終っていない (課題も放置状態)

▶ まず、2018/06/08 の内容を行う

一周目の内容

□ 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表現(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

再利用：ライブラリとサンプルの利用

□ 再利用によるソフトウェア作成

- スクラッチ(何もない所)から、作るのは効率が悪い

 - ▶ 「在るモノ」は利用しよう(「車輪の再発明」は良くない)

- 創造性の原理：「在るモノ」は作るな

□ 差分プログラミング：サンプルプログラムの利用

- 既に「*正しく* 動く」事が解っているプログラムを変更する

 - ▶ 「動きが変に成った」なら、「最後に変更した所が変(元に戻してみよ)」

 - ▶ 少しずつ、「作っては試す」を繰り返す(一度に完成しようとしなない)

□ ライブラリの利用：他の人の作ったプログラムを *丸々* 利用

- 約束事があるので守る必要がある

 - ▶ ヘッダーファイル (*.h) の利用

 - ▶ API：利用するための規則や役割の理解

 - ▶ ライブラリの利用：Makefile と make を利用

- 一周目：stdio.h/s_midi.h/s_turtle.h/s_miku.h/s_hanoi.h

 - ▶ 色々な事例をやってみた

□ 自分の作成したプログラムの利用

- 分割コンパイルと make

三つの基本制御構造と万能性(再)

□ 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] $f(x) \{ \text{if} (p(x)) \{ A \} \text{else} \{ B \} \}$

▷ f は $p(x)$ が成立すれば A そうでなければ B をする

○ [繰返] $f(x) \{ \text{if} (p(x)) \{ A f(x') \} \text{else} \{ \} \}$

▷ 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(2018/06/22)の予定

- 表現 : 「hello world」 again (「御呪い」を減らす)

- ▶ 「main 関数」/ 「#include」/ 「return 0」の役割

- ▶ 文字の入力(getchar())

- 作成 : データ型 / 入力 / 入力(Input)-処理(Process)-出力(Output)

□ 本日(2018/06/22)の目標

- 講義

- ▶ 「hello world」に拘る幾つかの謎を解く

- ▶ ライブラリの理解

- ▶ 一周目で身に付けられなかった内容を今度こそ獲得

- 演習

- ▶ 文字の入力

- ▶ 課題の提出

今週 (2018/06/22) の課題

□ 今週 (2018/06/22) の課題

○ 課題 20180622-01

- ▶ ファイル名 : 20180622-01-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なる国の挨拶をする
- ▶ 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 20180622-02:

- ▶ ファイル名 : 20180622-02-QQQQ.c (QQQQ は学生番号)
- ▶ 内容 : キーボードから一行(改行まで..)の文字列を読み込み、それを逆順に出力する
- ▶ 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)

先週 (2018/06/15) の課題

□ 先週 (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)

「Hello, World」再び

- 最も単純なプログラム : Hello, World
 - 完全で、解り易く、十分に役に立つプログラム
 - ▶ プログラム作成のスタートポイント
- 「Hello, World」の考え方(旧)
 - 「{」～「}」の中にある「printf ("Hello, World\n");」だけに注目
 - この部分を書き換えれば良い
 - ▶ 残りの部分は「御呪い」とすれば良い
- 「Hello, World」の謎
 - 御呪い:「#include」/「int」/「return 0;」は何をしている ??
 - ▶ 少し、「謎」を解いてみる

gene knockout

□ 遺伝子ノックアウト

○ その「遺伝子」の役割が解らなければ、それを壊して、何が起きるか見る

▶ 「働かなくなった機能」があれば、「それが、壊した遺伝子の機能」に関する

□ プログラム・ノックアウト

○ 動くプログラム(Hello, World)から、一部分を取り除いた(ノックアウト)したら..?

▶ ノックアウト 1 : printf() .. メッセージがでなくなった

▶ ノックアウト 2 : 全部 .. コンパイルは OK / リンクでエラー

▶ ノックアウト 3 : 「#include」.. コンパイル時に「警告」

▶ ノックアウト 4 : 「#include」と printf .. 1 と同じ

▶ ノックアウト 5 : 「void」にして「return 0;」を削る .. 問題ない

▶ ノックアウト 6 : 「#include」の代わりに「extern」.. 問題ない

○ 大雑把なまとめ

▶ main は必要 : 最も短いプログラム

▶ 「#include」: extern 宣言と関係するらしい..

▶ 「return 0;」/「int」: 「void」に交換すると良いらしい..

▶ 「printf」: まだ、謎があるようだ(変な extern 宣言)

○ 謎の解明 : 幾つかの謎は解けたが、まだ解明されない謎や、新しい謎が...

「関数」という考え方

□ 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か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

□ 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、その型と異な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」が異なる
- 「演算」と「型」は「一組」で考える必要がある

文字の入力関数 getchar

□ 関数 getchar()

- この関数を呼ぶ度に、キーボードから新しい一文字を読み込む
- その読み込んだ文字を値とする
 - ▶ 既に沢山入力されていれば、その最初の文字を返す
 - ▶ 逆に、まだ、文字が入力されていなければ入力されるまで待つ

□ [ポイント]

- 関数呼出しの結果として「(返り)値」を持つ事がある
- キーボードからの文字入力ができる(関数の値として「文字」が返える)
 - ▶ 「入力」は、改行キー([Enter]キー)を押す事によって引き起こされる
 - ▶ 「入力(文字列)」には、この改行キー迄を含む

入力-処理-出力

□ 入力-処理-出力

○ プログラムの基本構造

▷ 情報を入力し、それを処理した後、出力する

□ 入力-処理-出力の基本パターン (No.1)

○ `main` で入力を行い、処理関数を呼ぶ

○ 処理関数で処理を行い、`printf` を呼ぶ

○ `printf` で出力を行う