

ソフトウェア概論 A/B

-- 引数付き関数 / 条件判定 / 繰返し --

数学科 栗野 俊一 / 渡辺 俊一

2017/05/12 ソフトウェア概

伝言

私語は慎むように !!

- 出席パスワード : 20170512
- 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(2017/04/28)の課題 20170428-01 について

□ 前回(2017/04/28)の課題 20170428-01 について

○ 課題の解き方の説明が不十分だった

- ▶ `make` の使い方の説明ができていなかった
- ▶ 基本命令の説明がされていなかった
- ▶ 課題のサンプルが変だった

□ 追加説明

○ `make` の使い方 (`foobar.c` を例に)

- ▶ `foobar.c` は `midi` の下に置く必要がある
- ▶ 実行するには、「`make BASE=foobar test`」とする

○ `midi` の基本命令

- ▶ `midi` の下の `api.txt` を参照の事

○ 課題のサンプル

- ▶ 更新した(古い方で出してもよい、新しい方でもよい)

前回(2017/04/28)の復習

□ 前回(2017/04/28)の内容

○ MIDI の利用 (音楽の演奏)

- ▶ `cd ~/c/20170428/midi`
- ▶ `sudo /home/soft/c/20170512/midi/midi-setup.sh`
- ▶ プログラム作成 (`#include "s_midi.h"` が必要)
- ▶ `make BASE=ファイル名 test`

○ make

- ▶ Makefile と一緒に利用して、処理の自動化を行う

○ ライブラリと API

- ▶ ライブラリ: 巨人の肩に乗る (過去の資産を利用する)
- ▶ API: ライブラリを利用する為の手續(約束: ライブラリ毎に異なる)

お知らせ

- 本日(2017/05/12)の予定
 - 引数付き関数を作ってみよう
 - 条件判定を試してみよう
 - 再帰呼び出しを試してみよう
- 本日(2017/05/12)の目標
 - プログラムの基本ブロックである関数を学ぶ
 - 演習
 - ▷ makefile と make
 - ▷ 引数付き関数の使い方と作り方
 - ▷ 条件判定をするプログラム：状態によって振舞を変更する
 - ▷ 再帰呼び出しをするプログラム：同じ事を必要なだけ繰り返す
 - ▷ 課題の提出

前回 (2017/04/28) の課題

○ 課題 20170512-01:

- ▶ ファイル名 : 20170512-01-QQQQ.c (QQQQ は学生番号)
- ▶ 内容 : 童謡を演奏するプログラムを作成しなさい
- ▶ 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)
- ▶ 曲は何でもよい

○ 課題 20170512-02:

- ▶ ファイル名 : 20170512-02-QQQQ.c (QQQQ は学生番号)
- ▶ 内容 : 童謡の歌詞を出力するプログラムを作成しなさい
- ▶ 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)
- ▶ 可能な限り引数付きの関数で..
- ▶ 曲は何でもよい

本日 (2017/05/12) の課題

□ 本日 (2017/05/12) の課題 (CST Portal のみ)

○ 課題 20170512-01:

- ▶ ファイル名 : 20170512-01-XXXX.c (XXXX は学生番号)
- ▶ 内容 : 引数付き関数で、if 文で条件判断をするプログラムを作成しなさい
- ▶ 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)

○ 課題 20170512-02:

- ▶ ファイル名 : 20170512-02-XXXX.c (XXXX は学生番号)
- ▶ 内容 : 底辺の長さが指定した文字列の二倍の長さ - 1 の横向のピラミッドを作成するプログラムを作成しなさい
- ▶ 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)
- ▶ 再帰呼び出しを利用する

関数の作り方 (その 1)

□ 関数の作り方(引数のない場合)

- 名前を決める

- ▷ cf. subfunc

- どの部分を関数にするかを決める

- 関数にする部分を取り出し、外に出し、ブロックにする

- ▷ ブロックにするには '{' と '}' で囲めばよい

- ▷ 名前を付ける (cf. void subfunc())

- もともと部分があった所に関数呼び出しを書込む

- ▷ cf. subfunc();

関数呼び出しの挙動

- 関数呼び出しは次のように振舞う
 - 関数呼び出しのある場所から関数の先頭にゆく
 - 関数の中身を実行する
 - 関数呼び出しのある場所の次に戻る
- 関数の引数とは
 - 関数の振舞いを変更するための情報 (パラメータ)
 - ▶ 同じ関数でも引数が異れば異なる振舞いをす
- 引数付きの関数の呼び出し
 - 関数の中の変数に、引数の値が入っている

引数付き関数の作り方

□ 引数とは

○ 関数に与える事により、関数にその引数に対応した挙動をさせるもの

- ▶ 引数付き関数の定義：引数の値によって挙動が変わる
- ▶ 引数付き関数の利用：指定したい挙動をさせるための値を指定する
- ▶ cf. 三角関数：引数の角度によって異なる値を返す

□ 引数付き関数の作り方

○ 似ている二つ関数を一つの引数付き関数にまとめる

- ▶ 関数の本体の部分を、同じ部分と違う部分に分ける
- ▶ 違う部分は「変数」に置き換えて、一つの関数定義にまとめる
- ▶ 関数の仮引数の所に、「変数」を追加する
- ▶ 呼出す側は、実引数に、「違っていた部分の内容」を指定する

条件分岐

□ 引数の内容によって振舞いを「大幅」に変更したい

○ if 文と strcmp 関数を利用して対応できる

▶ strcmp 関数 : 二つの文字列を比較する

○ if (!strcmp (A, B)) { X } else { Y }

▶ A と B が同じなら X を、そうでなければ Y を行う

○ 「else if」を使うと更に複数の命令が選べる

▶ if (C1) { P1 } else if (C2) { P2 } .. else { Pn }

▶ C1 の時 P1、そうでなく C2 の時は P2 .. いずれでもない時 Pn

○ おまじない

▶ #include <string.h>

○ strncmp (A, B, N);

▶ A と B の先頭の N 文字だけを比較する

▶ !strncmp ("abc", "abz", 3); : 等しくない

▶ strncmp ("abc", "abz", 2); : 等しい

再帰呼び出し

□ 文字列を順番にみてゆく

- 「"abc" + 1」は「"bc"」と同じ振舞いをする

 - ▶ どういう仕組みは今回は説明しない

- 次々と 1 を加えれば、どんどん短かくなる

 - ▶ 最も短くなったかは、空文字(" ")と比較すれば判定できる

□ 再帰呼び出し

- 普通の関数は、別の関数を呼出す事ができた

 - ▶ 「自分の中」で「自分自身」を呼出す事ができる !! : 再帰呼び出し

□ 再帰呼び出しと帰納法

- 再帰呼び出しは、帰納法の考え方で問題を解く場合に利用できる

 - ▶ 再帰呼び出しが上手く行く事は、帰納法で証明できる (数学との関係)

- 再帰呼び出しをする場合は次の二点が重要 (帰納法と同じ)

 - ▶ 最も小さい場合 (ここでは、文字列が "" の場合) には終了する

 - ▶ そうでない時は、再帰呼び出しするが、その時には文字列を短くする

再帰呼び出しの考え方

□ 目標

- 「全部」をやりたい

- ▶ でも一挙にはできない

□ 対策

- そこで問題を二つに分ける

- ▶ 扱いやすい一部分：これは、そのまま対処してしまう

- ▶ 残り全部：(残り)「全部」なので、再帰呼び出しする

□ 注意点

- 「全部」が空っぽの時に忘れずに処理する

三つの基本制御構造と万能性

□ 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 !!