ソフトウェア概論 A/B

-- オセロゲーム板 --

数学科 栗野 俊一 / 渡辺 俊一

2013/10/18 ソフトウェア概論

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 (2013/10/11) の復習

- □ 前回 (2013/10/11) の復習
  - ○ データ構造：データ(形式)の「組み合わせ」によって新しいデータ(形式)を作る仕組
    - ▹ コーディング：現実の「情報」と計算機内の「データ」の対応関係
    - ▹ プログラムの機能：「データ」操作する事により「情報」を操作する事
    - ▹ 情報の構造とデータの構造：「情報」の複雑さに対して「データ」の構造化で対応
  - ○ typedef：新しい型名を作る(型の定義を行う命令)
    - ▹ typedef 既存の型 新しい型名; → 以下、「新しい型名」で「既存の型」を表す。
    - ▹ cf.「typedef int Number;」→ 以下、「int x;」と「Number x;」は *ほぼ* 同じ意味
  - ○ 構造体：異なるデータ(型)を組み合わせ(て新しい型を作)る
    - ▹ [数学] 直積空間を作る操作
    - ▹ 例: struct { double x; int y }; → <x,y> \in double × int
    - ▹ 構造内の要素は、「.(ピリオド)」と「タグ名」で参照
    - ▹ struct { double x; int y } v; → v.x, v.y が要素
  - ○ 配列：同じ(型)を幾つか組合せる
    - ▹ [数学] 順序対を作る操作
    - ▹ 例 : int a[5]; → a \in int × int × int × int × int, a[i] \in int (i=0,..,4)
    - ▹ 配列の要素は、「[] (ブラケット)」と「添字(何番目)」で参照
    - ▹ 例 : int a[5]; → a[0], a[1], .., a[4] が要素

お知らせ

- 本日の予定
  - オセロゲーム板を作る
- 本日の目標
  - 演習
    - オセロゲーム板を動かしてみる

前回 (2013/10/11) の課題

- 前回 (2013/10/11) の課題
  - 課題 1:
    - ファイル名 : 20131011-1-QQQQ.c (QQQQ は学生番号)
    - 内容 : 極座標で表現されている点 Q から、それと原点に対して対称な点 R を求める
    - 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 2:
    - ファイル名 : 20131011-2-QQQQ.c (QQQQ は学生番号)
    - 内容 : 構造体を利用し、平行移動を行う関数を作成する
    - 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)

本日の課題 (2013/10/18)

- 本日 (2013/10/18) の課題
  - なし

当分の方針

- 現状
  - 前回で、一応二周目は終わりで、三周目に入りたい
    - 基本は「Call By Need」:必要に応じて解説
    - 基本はアプリケーションを作りながら、復習 / 新しい事を学ぶ
- 暫く
  - オセロのゲーム盤を作ってみる

オセロゲーム

- オセロ(リバーシ)ゲームとは
  - 8x8 のゲーム盤と一方側が白で、他方が黒のコマを利用する
    - ゲーム盤は横方向には A ～ H 縦方向には 1 ～ 8 の番号がふられる
    - 初期状態では D4, E5 に白, E4, D5 に黒がおかれている
  - 一方が黒、他方が白をもち、交互にコマを打つ
    - 自分の打つコマと既に置かれている自分のコマで相手のコマを挟むとその相手のコマは反転し、自分のコマになる
    - 自分が打つ事ができるのは相手のコマを挟める時だけ
    - 自分が打つ事ができない場合はパスとなり相手の番になる
  - 盤面がすべてコマで埋まるか、あるいはパスが連続した場合はゲーム終了
  - ゲーム終了状態で盤面場のコマの数を数え数の多い方が勝ちとなる

オセロゲームの設計

- □ データ構造の設計
  - ○ ゲーム盤の定義 (board.h)
    - ▹ ボードの大きさ (#define BOARD_SIZE 8)
    - ▹ BInfo 盤面マスの中の内容 (EMPTY, WHITE, BLACK)
    - ▹ ボード自身 typedef struct { BInfo info[BOARD_SIZE][BOARD_SIZE];} Board;
  - ○ ゲーム盤の処理 (board.c)
    - ▹ Board init_board (); 初期ゲーム盤を作る
    - ▹ void print_board (Board board); ゲーム盤を表示

二次元配列

- 配列：同じ物が、一列に並んだ物 (一次元配列)
  - 配列の配列は？ (配列が二次元に並んでいる)：二次元配列となる
    - BInfo info[BOARD_SIZE][BOARD_SIZE]; info[i][j] が二次元に並んでいる

	0	1	..	6	7
0	info[0][0]	info[0][1]	..	info[0][6]	info[0][7]
1	info[1][0]	info[1][1]	..	info[1][6]	info[1][7]
			..		
6	info[6][0]	info[6][1]	..	info[6][6]	info[6][7]
7	info[7][0]	info[7][1]	..	info[7][6]	info[7][7]

  - k 次元の配列の配列は？
    - k+1 次元の配列になる

配列と文字列

- 文字列とは？
  - 文字の並びだった.. という事は..
    - char の一次元配列が「文字列」？
- 答は Yes でもあり No でもある
  - Yes :「文字列」は「char の一次元配列」で実現されている
    - char の一次元配列にできることは「文字列」にもできる
    - 「文字列」を「char の一次元配列」のように扱って良い
  - No : 逆は真ではない(char の一次元配列は、文字列とは限らない)
    - char の一次元の配列の要素を文字にし最後に EOS(\0)を入れれば文字列と同様に振る舞う
    - C 言語のコンパイラが、「文字列」を特別扱いしてくれる

分割コンパイル

- 分割コンパイル
  - 一つのファイルに全部入れても良いが..
    - 複数に分割する事も出来る
  - 分けるメリット
    - 柔軟性が生れる：リンク時にも制御できる(Test Case)
    - 相互の影響が小くなる：編集時の置換の失敗
  - 分けるデメリット
    - 情報の共有が難しい：ヘッダーファイルの利用
    - コンパイルリンクの手間手間が増える：make & Makefile の利用

色々な変数の宣言とスコープ

- 変数の宣言
  - これまでは関数の中だけだった(ローカル変数)
    - 関数の中だけで有効
    - ブロック内だけで有効な変数もつくれた
- 関数の外でも変数は宣言できる(グローバル変数)
  - 複数の関数から共通の変数を利用する事ができる
    - 関数間に変数を経由した「結び付き」ができる
  - グローバル変数の得失
    - 「結び付き」が「便利さを生む」のは事実(便利なので)
    - 常に意識する必要がある(腐れ縁になってしまう可能性が..)
  - グローバル変数の利用はできるだけさける
    - 情報は、引数と返り値でやり取りする
    - ただし、効率は悪くなる(コピーがおきるので..)
    - 効率を高める方法はあるが..(後に、ポインターの話をする)