ソフトウェア概論 A/B

-- オセロゲーム盤(4) --

数学科 栗野 俊一 / 渡辺 俊一

2013/11/15 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 (2013/11/08) の復習

- 前回 (2013/11/08) の復習
  - プログラムファイルの分割
    - 分割コンパイルとリンク
    - ヘッダーファイルの役割
  - make と Makefile
    - 分割コンパイルの補助
    - 作業手順の記録(プログラミング)

お知らせ

- 本日の予定
  - オセロゲーム盤を作る(その四)
- 本日の目標
  - 講議
    - オセロゲーム盤の完成
    - グラフィックスの利用
  - 演習
    - オセロゲーム盤を動かしてみる(その四)

前回 (2013/11/08) の課題

- 前回 (2013/11/08) の課題
  - 課題 1:
    - ファイル名 : 20131108-1-QQQQ.mk (QQQQ は学生番号)
    - 内容 : Makefile の作成
    - ファイル形式 : テキストファイル(Makefile 形式)

本日の課題 (2013/11/15)

- 本日 (2013/11/15) の課題
  - なし

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)

グローバル変数の利用

- □ グローバル変数
  - ○ グローバル変数の性質
    - ▹ *全ての関数* で共有される (スコープが「プログラム、全体」)
    - ▹ *任意の時点* で利用可能 (エクステントが「実行時、何時でも」)
  - ○ グローバル変数の定義
    - ▹ 変数関数の外で宣言される変数は、グローバル変数となる
    - ▹ プログラム内では一度しか「定義」できない(一つのファイルだけ)
    - ▹ 他のファイル内で利用する場合は extern 宣言する(「宣言」は何度でも可)
    - ▹ cf.「関数名」も「グローバル」である事に注意
- □ グローバル変数の得失
  - ○ 性質：複数の関数で同じ名前で同じ変数が共有できる
    - ▹ 利点：「共有」ができるは嬉しい：何時でも何処でも同じモノが利用可能
    - ▹ cf.「auto 変数」は同じ「名前」でも異なる「変数」を表す
    - ▹ 欠点：「共有」して仕舞うのは悲しい：相互に影響してしまう
    - ▹ cf.「サイフが一緒」は嬉しいか？(おこずかいが欲い.. ?)
  - ○ 方針：可能な限り使わない(利用するのは必要最小限に抑える
    - ▹ 可能ならば static (静的)変数を利用する

スコープとエクステント

- 名前と、名前が指す実体の対応が有効な範囲
  - 空間範囲(スコープ) : どの「位置の範囲」で有効か？
    - グローバル変数 : どの位置でも有効
    - ローカル(自動)変数 : 宣言されたブロック(関数)内
    - 名前 / 静的的(コンパイル時) / 変数宣言の位置で決る
  - 時間範囲(エクステント) : どの「時点の範囲」で有効か？
    - グローバル変数 : プログラムの実行中何時でも
    - ローカル(自動)変数 : ブロック(関数)に入った時点から出るまで
    - 情報 / 動的(実行時) / 変数宣言の種類で決る(auto/static/global)

		エクステント	
		一時的	永遠
スコープ	局所的	自動 (auto) 変数	静的 (static) 変数
	全域的	ヒープ	大域変数

静的(static) 変数

- 静的(static) 変数とは
  - スコープが局所的で、エクステントが永遠の変数
    - 変数宣言に、キーワード static を先攻させると、静的変数になる
  - 静的(static) 変数のスコープ
    - block(関数内)での宣言：block(関数内)になる (auto 変数と同じ)
    - 関数の外での宣言：ファイル内になる
- 静的(static) 変数の用途
  - スコープに着目：スコープをファイルにしたい場合
    - グローバル変数はスコープが広過ぎる
    - 関連する関数を一つのファイルにまとめ、共通の static 変数を共有
  - エクテンツに着目：定数テーブルが欲しい場合
    - 初期化が可能なので、最初に値を設定して、後は共有