ソフトウェア概論 A/B

-- データ構造 (多次元配列/動的なデータ型) --

数学科 栗野 俊一

2011/11/25 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)
- 講義開始前にすませておくこと
  - PC の電源を入れる
  - ネットワークに接続しておくこと
  - 今日の資料に目を通しておくこと
- 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

前回の復習：配列

- メモリモデルとポインター値
  - メモリモデル: 計算機の情報はメモリに記録される
    - メモリ: 1 byte の Cell の集まりで、個々の Cell にアドレスが付く
    - 変数: メモリの 1 個以上の Cell の集まりで、演算情報をもつ
  - ポインター値：アドレス値 + 型情報
  - 添字
    - pa[i] == *(pa + i)
  - 配列：同じ物の直積(並び)を作る
    - メモリモデルの抽象化
  - 配列の宣言
    - 型 配列名[サイズ] : 必要なサイズのメモリが自動的に確保される

お知らせ

- 本日の予定
  - 多次元配列
  - 動的メモリの確保
- 本日の目標
  - 演習
    - 課題の提出

前回 (2011/11/18) の課題

- 今週 (2011/11/18) の課題
  - 課題 1: [参考: sample-006.c]
    - ファイル名 : 20111118-1-QQQQ.c (QQQQ は学生番号)
    - 内容 : 5 つの整数を読み込んで、その逆順に出力するプログラム
    - 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: [参考: sample-011.c]
    - ファイル名 : 20111118-2-QQQQ.c (QQQQ は学生番号)
    - 内容 : 5 つの整数を読み込んで、その全部と真ん中の三つの総和を計算するプログラム
    - 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)
  - [注意] 前回は三つ課題があったがその中の最後は今週に回した

今週 (2011/11/25) の課題

- 今週 (2011/11/25) の課題
  - 課題 1: [sample-007.c 参照]
    - ファイル名 : 20111125-1-QQQQ.c (QQQQ は学生番号)
    - 内容 : 3 次行列の積を計算するプログラム
    - 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)

配列 (復習)

- 配列
  - 同じ物の直積(並び)を作る
    - メモリモデルの抽象化
  - 配列の宣言
    - 型 配列名[サイズ]
- 配列宣言のポイント
  - 必要なサイズのメモリが自動的に確保される
  - 同じ型の変数を n 個同時に作る事ができる
    - 多数のデータを保存・処理するのに都合がよい[001]
    - [注意]「代入式」、代入文はそれ自身値をもつ。値は代入する値。
- 配列と関数の引数[002]
  - 「配列名」は「ポインター定数」となる
    - 渡されるのは先頭のアドレスだけ
    - 配列のサイズの情報は明示的に与える必要がある (文字列の場合は EOS)
    - 関数はそのサイズが正しいかどうかを知る事ができない
  - ポインター値を利用して、メモリの内容を直接書き換える事ができる
    - 変数名を利用した方法だと制限(名前の有効範囲)がある[003]
    - ポインター値の場合はその制限がないが、気を付ける必要がある(ポインター一般の問題)

多次元の配列

- 二次元の配列
  - 一次元の配列を二次元的に利用する事ができる[004]
    - 二つの添字からアドレスを計算すればよい
  - 始めから二次元の配列を宣言する事ができる[005]
    - 配列の要素のアドレス計算は、自動的に行われる[006]
    - 実態は「(一次元)配列の(一次元)配列」だが、慣例により「二次元配列」と呼ぶ
  - 応用例
    - 行列は、二次元配列で表現可能[007]
- 多次元の配列
  - 一つの型から新しい配列型を作るだけなのでいくらでも大丈夫(?)

動的なメモリ利用

- 変数宣言
  - メモリ領域を確保し、名前に結び付られる
    - 予め(考えて)プログラムの中に記述して置く必要がある(開発時に決定)
  - 利用できる変数の個数や種類(型)が固定されてしまう (static[静的]なメモリ管理)
    - 配列のサイズも固定 !!
- 動的メモリ管理 [008]
  - malloc で、必要なサイズのメモリが確保できる (ヒープから確保)
    - 決定を遅らせて、プログラムの実行時に決めてもよい (dynamic[動的]なメモリ管理)
    - ただし、自分で後始末 (free) が必要
- 動的 vs 静的
  - 動的 : サイズや範囲を自分の好きにできる(権利)/自分で後始末が必要(義務)
  - 静的 : 名前やメモリ管理など色々、面倒をみてもらえる(便利)/制限がある(対価)