ソフトウェア概論 A/B

-- データ構造 (5) --

(動的メモリ管理とキャスト)

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

2019/01/11 ソフトウェア概論

伝言

私語は慎むように !!

- 出席パスワード : 20190111
- 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

今後の予定

- 今後の予定(後ろから)
  - 2018/01/25 (講議最終日)
    - 試験を行う
  - 2018/01/18 (講議最終日前)
    - 模擬試験を行う (予定) / FILE 入出力
  - 2019/01/11 (本日)
    - データ構造 (5) : 動的メモリ管理とキャスト

前回(2018/12/21)の内容

- 前回 (2018/12/21) の復習
  - メモリモデル：アドレス(番地:アドレス値は正の整数値)が振られているセルの並び
    - 一つのセルが記録できるのは、1 byte の情報：型の概念はない
  - C 言語の「変数」：複数(1個以上)の連続したセルに対応
    - 変数名：その変数が対応するセルの先頭のアドレスを表す
    - 型：そのセルの情報を、「どのように扱うか(size も含む)」を表す
    - sizeof(変数)：その変数が対応しているセルの個数を表す：sizeof(型)
  - ポインタ値：配列名や、「&:ポインタ演算子」から得られる値
    - 「値」: (プログラム実行時は、)変数に対応するセルの「アドレス値」
    - 「型」:「型 *」型：(コンパイル時は、)「*:間接参照演算子」を付けると「型」になる型

お知らせ

- 本日の予定
  - データ構造 (5)
    - 動的メモリ管理とキャスト
- 本日の目標
  - 演習
    - 課題の提出

先週 (2018/12/21) の課題

- 先週 (2018/12/21) の課題
  - 課題 先週-01:
    - ファイル名 : 先週-01-QQQQ.c (QQQQ は学生番号)
    - 内容 : union の応用 (整数型変数のメモリ構造の出力)
- ※
  - ファイル形式は、いずれもテキストファイル(C 言語プログラムファイル)

今週 (2019/01/11) の課題

- □ 今週 (2019/01/11) の課題
  - ○ 課題 今週-01:
    - ▹ ファイル名 : 今週-01-QQQQ.c (QQQQ は学生番号)
    - ▹ 内容 :「三角形の形」をした配列
  - ○ 課題 今週-02:
    - ▹ ファイル名 : 今週-02-QQQQ.c (QQQQ は学生番号)
    - ▹ 内容 : 動的なメモリの利用
- □ ※
  - ○ ファイル形式は、いずれもテキストファイル(C 言語プログラムファイル)

一次元配列と多次元配列

- 多次元配列は配列の配列
  - 結局は、メモリ上の連続したセルに対応
    - アドレス値が計算できれば、要素が参照できる
- 二次元配列のアドレス計算
  - int dary[3][4]; int 型の 3 x 4 = 12 個の要素からなる
    - のアドレス値 : dary の表すアドレス値 + sizeof(int) * 4 * 1 + sizeof(int) * 2
    - 結局一次元の場合と同様にアドレス値の計算がされているだけ

型変換とキャスト演算

- 型情報：「そのデータをどのようなものとして扱うか」という情報
  - コンパイル時のみ意味がある(実行時は明示的にはわからない)
- 型変換
  - 値の型を変更する事
    - 例 : int -> double (1 -> 1.0)
  - 「型」は、「それをどう扱うか」を意味するので、型が違えば操作も違う
    - 例 :「3/2 -> 1」(int 型) /「3.0/2.0 -> 1.5」(double 型)
    - 型から操作を決めているのはコンパイラ / CPU には「型」はない
- キャストティング
  - 型変換を明示的に行う
    - 「(型名)」を値の前に先行する事により、型を変更できる
    - 例 :「(double)1 -> 1.0」/「(int)1.5 -> 1 」
    - 「型」だけでなく、「表現」が変化

ポインタ演算とキャスト

- ポインタ値の二つの情報
  - アドレス値 + 型情報
- アドレス値の変更
  - 整数値の加減算ができる
    - アドレス値は「加える整数値 x sizeof(型)」だけ変化する
- 型情報の変更
  - キャストによって、型情報を変更できる

動的メモリ領域の確保

- 変数領域の確保
  - 基本は変数宣言の時に、その変数に対応する領域が確保される
  - 変数は予め宣言しておく必要がある
    - どの位のデータを記録するかを予め決めておく必要がある
    - 記憶するデータの量が予想できない場合はどうすべきか？
- 動的メモリ領域の確保
  - malloc/calloc (alloc 関数)を利用して、動的にメモリを確保する事ができる
    - 確保された領域は使用が終ったら free で解放する必要がある
  - alloc 関数は、確保した領域へのポインター値を返す
    - 必要に応じて、サイズの指定とキャストが必要
    - 領域への参照は、必然的にポインター値経由となる(名前がない)