ソフトウェア概論 A/B

-- データ構造 (6) --

(動的メモリ管理とキャスト)

数学科 栗野 俊一 / 渡辺 俊一

伝言

私語は慎むように !!

- 出席パスワード : 20171208
- 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)
  - 2018/01/26 (講議最終日)
    - 試験を行う
  - 2018/01/19 (講議最終日前)
    - 模擬試験を行う
  - 2018/01/12 (講議最終日前)
    - 落穂拾い (2)
  - 2017/12/29 / 2018/01/15
    - 冬期休暇期間中：この講議はない
  - 2017/12/22
    - 落穂拾い (1)
  - 2017/12/15 (次週)
    - データ構造 (7) / 動的データ構造 / ファイル I/O
  - 2017/12/08 (本日)
    - データ構造 (6)：動的メモリ管理とキャスト

前回(2017/11/24)の内容 [1]

- 前回 (2017/11/24) の復習 [1] : メモリ･モデル
  - メモリ･モデル :「主記憶」という考え方
    - セル : 1 byte の情報が記録できる箱 (char 型変数と同様)
    - メモリ : セルが並んだもの (個々のセルは番地[アドレス] で区別)
    - セルの特性 : 値の参照、代入が可能 -> C 言語の変数の性質
  - メモリと C 言語の変数の関係
    - C 言語の変数の性質(参照や代入)は、セルの性質からきていた
    - 「番地(アドレス)」は、「変数名」に対応している
  - 文字型変数とセル
    - 文字型変数は一つのセルからなる
  - 2 進数
    - bit/byte と「数(十進数)」の対応関係

前回(2017/11/24)の内容 [2]

- 前回 (2017/11/24) の復習 [2] :「アドレス」の効用
  - 配列の添字, 間接(参照)演算子,アドレス演算子
    - 「配列名の値」は、「先頭の要素の『アドレス』」を表す
    - 「* (間接(参照)演算子)」は、「アドレス」から「変数」を作る
    - 「& (アドレス演算子)」は、「変数」から「アドレス」を作る
  - 「アドレス」の効用
    - 「アドレス」は「値」なので、(*,& を使って)「『変数』そのもの」を「間接的に」関数に渡せる
    - 「便利」だが、「危険」
  - scanf の正体
    - 変数の前に「&」をつけるのは、アドレスを渡していた

お知らせ

- 本日の予定
  - 前回の残り
    - 文字型以外の変数とメモリモデル
    - 多次元配列
  - データ構造 (6)
    - 動的メモリ管理とキャスト
- 本日の目標
  - 演習
    - 課題の提出

前回 (2017/11/24) の課題

- 前回 (2017/11/24) の課題
  - 課題 20171124-01:
    - ファイル名 : 20171124-01-QQQQ.c (QQQQ は学生番号)
    - 内容 : メモリ操作での和
  - 課題 20171124-02:
    - ファイル名 : 20171124-02-QQQQ.c (QQQQ は学生番号)
    - 内容 : アドレスを利用した間接参照
- ※
  - 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)

本日の課題 (2017/12/08)

- 本日 (2017/12/08) の課題
  - 課題 20171208-01:
    - ファイル名 : 20171208-01-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)

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)