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

2019/12/13 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 : 20191213
- 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
- VNC Server Address : VNCSERVER
  - Password : VNCPASS
- 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

今後の予定

- 今後の予定(後ろから)
  - 2020/01/24 (講議最終日)
    - 試験を行う
  - 2020/01/17 (休講 ??)
    - 講義があるなら、落穂拾い or アプリ紹介
  - 2020/01/10 (講議最終日前)
    - 模擬試験を行う
  - 2020/01/03
    - 冬期休暇期間中：この講議はない
  - 2019/12/27
    - 落穂拾い
  - 2019/12/20 (次週)
    - データ構造 (7) : File I/O
  - 2019/12/13 (本日)
    - データ構造 (6) : 動的メモリ管理とキャスト

前回(2019/12/06)の内容

- 前回(2019/12/06)の内容
  - メモリモデル
    - メモリはセルの集まり : セルには 1 byte 分の情報を記録
    - セルには、セルの場所を表す「番地番号」が一意の連続した整数値がついている
    - セルの値の取り出しや設定(代入)は、「番地番号」を利用してセルを指定して行う
  - C 言語の配列との比較
    - メモリは一つ : 配列は、複数 (先頭の場所を配列名、サイズは宣言で指定)
    - セルのサイズは 1 byte : 配列は要素のサイズが色々(宣言型できまる)
    - セルの指定は番地番号で行う : 配列は配列名と添字 (一般の変数は名前だけ)
  - メモリと変数の関係
    - セルの(1 以上の..)並びが、変数 (変数の特質はセルの特質)
  - 「ポインター値」とは何か？
    - 基本は番地番号(実行時)だが、型属性も持つ(コンパイル時)

本日(2019/12/13)の予定

- 本日(2019/12/13)の予定
  - 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

今週 (2019/12/13) の課題

- □ 今週 (2019/12/13) 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)

先週 (2019/12/06) の課題

- 先週 (2019/12/06) 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：ライブラリ関数 strcpy と同じ振舞をする mystrcpy を作成しなさい
- ※
  - 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)