ソフトウェア概論 A/B

-- データ構造 (7) --

(動的データ構造 / ファイル I/O)

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 : 20171222
- 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
    - 落穂拾い
  - 2017/12/29 / 2018/01/05
    - 冬期休暇期間中：この講議はない
  - 2017/12/22 (本日)
    - データ構造 (7) / 動的データ構造 / ファイル I/O

前回(2017/12/15)の内容 [1] : ポインタ値とポインタ値操作

- □ 前回 (2017/12/15) の復習 : ポインタ値とポインタ値操作
  - ○ ポインタ値 : アドレス値 + 型情報 (サイズ + データの扱い型)
    - ▹ アドレス値 : 実行時の概念 (実際に関数の引数で渡される「値」)
    - ▹ 型情報 : コンパイル時の概念 (関数の仮引数変数の型宣言で記述)
  - ○ ポインタ値の操作 : アドレス値と、型情報が、其々個別に変更可能
    - ▹ アドレス値の操作 : 「整数値」を加えると、「整数値 * sizeof(型)」だけ変化
    - ▹ 型情報の操作 : キャスト演算子「(型 *)」を先行して、型情報が変えられる
  - ○ 多次元の配列 (配列の配列 [の其のまた...])
    - ▹ 結局は、最終的に memory モデル (cell の並び) に
    - ▹ アドレス値の計算が異るだけ (int a1[3][2] <-> int a2[2*3])

前回(2017/12/15)の内容 [2] : 一般のキャスト(型変換)

- 前回(2017/12/15)の内容 [2] : 一般のキャスト(型変換)
  - 型変換 : 特定な型の値を別の異る型の値に変換する仕組み
  - 型変換の実行 : 三通り (代入/キャスト/昇格)
    - 代入 (変数 = 式) : 左辺の「式」の値は、「変数」の型に変換されて代入 (関数への実引数渡しや return も代入の一種)
    - キャスト : 値の前にキャスト演算子「(型記述)」を先行してその値の型を変換する(明示的な強制型変換)
    - 昇格 : 型が混在する計算では、暗黙の内に型の変換(型の昇格)が行われる
  - (一般の)キャスト(型変換)
    - キャスト演算子「(型)」を先行して「型変換」が行える
    - 一般に「型変換」を行うと、「データそのもの」も変化するのが普通
    - 例 : (int) 1.5 -> 1 / (double)1 -> 1.0
  - 型の広さ : 表現可能な範囲の事
    - double > int > char
    - 広い方から狭い方に型変換すると、情報が失われてしまうので注意(危険)
  - 型の昇格 : 暗黙の内に自動的に行われる型の変換
    - 異る型の混在する式では、(必要なら)「型の昇格」がおきる : 3.0/2 -> 3.0/2.0 -> 1.5
    - 「型の昇格」は一般に「型の広い(より多くの数が表現できる)方」に行われる(ので安全)

お知らせ

- 本日の予定
  - データ構造 (7)
    - 動的データ構造 / ファイル I/O
- 本日の目標
  - 演習
    - 課題の提出

前回 (2017/12/15) の課題

- □ 前回 (2017/12/15) の課題
  - ○ 課題 20171215-01:
    - ▹ ファイル名 : 20171215-01-QQQQ.c (QQQQ は学生番号)
    - ▹ 内容 :「三角形の形」をした配列
- □ ※
  - ○ 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 (2017/12/22)

- 本日 (2017/12/22) の課題
  - 課題 PPNAME-01: (これは前々回 PPDATE の課題)
    - ファイル名 : PPNAME-01-QQQQ.c (QQQQ は学生番号)
    - 内容 : 動的なメモリの利用
  - 課題 20171222-01:
    - ファイル名 : 20171222-01-QQQQ.c (QQQQ は学生番号)
    - 内容 : 整数値のキュー(queue)
  - 課題 20171222-02:
    - ファイル名 : 20171222-02-QQQQ.c (QQQQ は学生番号)
    - 内容 : 二つのファイルを比較して最初に異る場所を表示する
- ※
  - 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)

動的なデータ構造

- 静的なデータ構造
  - 配列/単純変数：保存できるデータのサイズは固定
    - alloc を使っても、配列ならば、その中に入るデータの個数は固定
    - メモリの無駄を覚悟すれば、「大きなサイズの配列」でも代用可
- 動的なデータ構造
  - 保持できるデータ数が可変長(本質的に動的)
    - alloc を利用しないと扱えない
- 動的なデータ構造の例
  - List (リスト)：複数の要素を「並べた」もの

構造体へのポインター値

- 構造体のタグ
  - 構造体には、構造体を区別する名前(tag:タグ:構造体名)がつけられる
    - struct タグ { .. }
  - 「sturct タグ」は構造体型の名前として利用できる
    - これまでは、これを typedef でサボッていた
  - ポインタによる自己参照を行う場合は、タグがどうしても必要になる
- 構造体へのポインタ
  - 構造体へのポインタを利用して、構造体の要素にアクセスできる
    - p が、要素 x を持つ構造体へのポインタなら (*p).x で参照で可能
    - 構造体へのポインタの要素の参照はよくあるので、省略記号がある
    - 「(*p).x」の代りに「p -> x 」という表現が利用できる

ファイル I/O

- ファイル操作
  - ファイル：データを恒常的記憶するための仕組み
    - メモリの内容は、PC の電源を切ると失われる
  - リダイレクション(復習)
    - 「<」、「>」を利用して、プログラムの入出力をファイルに変更可能
    - せいぜい、入力と出力に一つずつしかファイルが利用できない
  - ファイル I/O ライブラリ
    - 任意のファイルに対する読み(Read/Input)書き(Write/Output)する機能

FILE 構造体

- FILE 構造体による File I/O
  - fopen 関数を利用して、ファイル I/O を管理する FILE 構造体へのポインタ値を得る
    - FILE 構造体は、「どのファイルの何処を参照しているか」を管理する情報
    - FILE 構造体を経由して、少しずつファイルからデータを得る事が可能
    - open に失敗した場合、fopen 関数は NULL を返す
  - 利用が終った FILE 構造体は fclose で必ず close する
    - 特に write 時には close しないとデータが失わる(保存されない)可能性が生じる
    - read の場合も、close しないと「資源の無駄使い」になる
    - cf. memory の alloc/free と同様な関係
  - データの入出力
    - fgetc/fputc : 文字単位の File I/O
    - fscanf/fprintf : ファイルに対する printf/scanf
    - fread/fwrite : ファイルに対する固定長のデータの I/O