ソフトウェア概論 A/B

-- データ構造 (6) --

(動的なメモリ管理)

数学科 栗野 俊一 / 渡辺 俊一

2016/11/25 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 : 20161125
- 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

前回(2016/11/18)の内容(1)

- □ 前回(2016/11/18)の内容
  - ○ 配列名(復習)
    - ▷ 配列名は、配列の先頭の要素のポインター値を表す
    - ▷ ポインター値は、関数の引数に渡す事ができる
    - ▷ ポインター値は、配列名と同じ役割をする
    - ▷ 関数の引数に配列名を指定すると、ポインター値が渡される
    - ▷ 関数内で、そのポインター値を利用して配列要素が参照できる(間接参照)
    - ▷ 配列要素の間接参照により、配列の要素は「共有」される(コピーされない)
  - ○「文字列」(復習)
    - ▷ C 言語では「文字列」は、「文字型の一次元配列」で表現されている
    - ▷ "ABC" は、名前のないサイズ 4 の文字型の配列 (最後に '\0' が入る)
    - ▷「文字列」の値はその名前のない配列の先頭の要素のポインター値になる
    - ▷ C 言語では、「文字列型」というものは存在せず、文字配列で代用

前回(2016/11/18)の内容(2)

□ 前回(2016/11/18)の内容(2)

- メモリモデル
  - メモリ：セル(1 byte のサイズ)の並び
  - アドレス：個々のセルには、それぞれ別々のアドレスがついている
  - セルの操作は、アドレスを利用して行う
  - セルは、1 byte の情報を記録でき、set/get が可能
  - 現実の計算機のメモリは、メモリモデルの(実)例になっている
- メモリモデルと C 言語の変数
  - C 言語の「変数」は、「1 個以上のセルの並び」で表現できる
  - 変数の性質は、セルの性質で説明できる
  - 現実に、変数が、実際のメモリ(=メモリモデルの例)で実現される事が多い
- メモリモデルによる変数の理解
  - メモリモデルを基本に、変数をメモリモデルを通して理解する事が望ましい

お知らせ

- 本日の予定
  - データ構造 (6)
    - ポインタ値とポインタ型
    - 動的なメモリ管理
- 本日の目標
  - 演習
    - 課題の提出

前回 (2016/11/18) の課題

- 前回 (2016/11/18) の課題
  - 課題 20161118-01:
    - ファイル名 : 20161118-01-QQQQ.c (QQQQ は学生番号)
    - 内容 : メモリ操作での和
  - 課題 20161118-02: (これは今週 2016/11/25 にまわす)
    - ファイル名 : 20161118-02-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)
  - 今回の新規追加の課題はなし

本日の課題 (2016/11/25)

- 本日 (2016/11/25) の課題
  - 課題 20161118-02: (これは前回 2016/11/18 の残り)
    - ファイル名 : 20161118-02-QQQQ.c (QQQQ は学生番号)
    - 内容 : アドレスを利用した間接参照
  - 課題 20161125-01:
    - ファイル名 : 20161125-01-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)

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)