ソフトウェア概論 A/B

-- データ構造 (5) --

(ポインター型)

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 : 20211203
- 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

ソフトウェア概論 A/B (2021/12/03)

前回の内容

前回の内容

- 前回の内容：データ構造 (4)
  - 講義
    - 配列要素の参照方法：「ARY[INDEX]」と「*(ARY+INDEX)」
    - 関数への引数とした時の「配列名」の意味

本日(2021/12/03)の予定

ソフトウェア概論 A/B (2021/12/03)

本日(2021/12/03)の予定

本日(2021/12/03)の予定

- 本日(2021/12/03)の予定
  - データ構造 (5)
    - ポインター型
- 本日の目標
  - 演習
    - 課題の提出

ソフトウェア概論 A/B (2021/12/03)

課題

今週 (2021/12/03) の課題

- 今週 (2021/12/03) の課題
  - 課題 20211203-01: 文字列の並び
    - ファイル名 : 20211203-01-QQQQ.c (QQQQ は学生番号)
    - 内容 : 一つの文字配列に複数の文字列を保存する
- ※
  - 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)

先週 (2021/11/26) の課題

- □ 先週 (2021/11/26) の課題
  - ○ 課題 20211126-01: 文字列の入力
    - ▹ ファイル名 : 20211126-01-QQQQ.c (QQQQ は学生番号)
    - ▹ 内容 : キーボードから一行の文字列を入力し、その中にある 'a' の個数を出力する
- □ ※
  - ○ 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)

ソフトウェア概論 A/B (2021/12/03)

型のサイズ

型のサイズ

- 型のサイズ
  - データ(情報)はサイズを持つ
    - 例 1: char 型 のサイズ : 8 bit = 1 byte
    - 例 2: int 型のサイズ : 64bit = 4 byte
  - サイズ S byte のデータは 2^(8S) = 256^S の状態を表現できる
    - 例1 char 型 は 0 ～ 255 (256 通り) の状態 : 半角文字は表現できるが全角文字は無理
    - 例2 int 型 は -2^{63} (-2147483648) ～ 2^{63} - 1 (2147483647) までの 2^64 通り
    - cf. /usr/include/limits.h
  - その型のデータのサイズ
    - その型の状態数を表現 / その型の情報を記録するために必要な記憶領域サイズ
    - より多くの状態を表現したければ、より多くのサイズ(の記憶領域)が必要

sizeof 演算子

- sizeof 演算子
  - 前置演算子で、その後ろにあるデータのサイズを byte 単位で答える
    - 引数に「型名」を記述する事もできる
  - C 言語では、型に対するデータのサイズはシステムによって異る
    - cf. /usr/include/limits.h
    - 例 : int は、その計算機(32bit/64bit)で最適なサイズになる (sizeof(char) は 1)
    - 個々の計算機で「最適」なコードが作られる(可能性が高い):利点
    - (サイズが異るので..) 同じプログラムが、システムによって異る振舞をする:欠点
    - sizeof 演算子は、その「違い」を吸収する必要がある場合に利用
- C 言語における型情報
  - 型 : 表現形式 x 操作方法
  - 表現形式 : サイズ x 情報との対応形式
    - サイズは、表現対象の集合のサイズ(有限の場合)より大きくする(char)
    - 表現対象の一部としか対応していな場合がある(無限の場合:整数、実数等)

ソフトウェア概論 A/B (2021/12/03)

型変換

暗黙の型変換(型の昇格)

- char 型から int 型への型の昇格
  - 「計算」の場合、char 型の値は int 型に「(無条件に)昇格」する
    - char 型のサイズは 1 (= sizeof(char))
    - 'A' の値は、整数値 65 (ASCII Code) になる
    - cf. sizeof(char) == 1 / sizeof('A') == sizeof(int)
- int 型から double 型への型の昇格
  - int 型同士の計算は int 型のまま
  - double 型と int 型の混在式では、int 型から double 型への「型の昇格」が起きる
- 代入における型変換
  - 変数への代入では、値の型が、変数の型に変換される
    - 関数の「実引数(値)」は、関数の「仮引数(引数変数)」への代入となる
  - サイズの小さい方から大きい方の変換は問題ない
    - その逆(大きい方から小さい方)は「危険」!! (オーバーフローする)

ソフトウェア概論 A/B (2021/12/03)

メモリ操作を行う演算子

配列の添字, 間接(参照)演算子,アドレス演算子

- □ 文字列の操作 (復習)
  - ○「*」:間接(参照)演算子：文字列の先頭の文字を取り出す (sample-011.c)
    - ▹ *"abc" == 'a'
  - ○「[]」:添字演算子：「[n]」で n (整数値)で「n+1番目の文字」意味する
    - ▹ "abc"[0] == 'a', "abc"[1] == 'b', ..
  - ○「*」と「[]」の関係；文字列[n] == *(文字列+n)
    - ▹ "abc"[0] == *("abc"+0) == *("abc") == *"abc" == 'a'
- □ アドレス演算子「&」
  - ○ アドレス演算子「&」は 間接演算子「*」の逆演算を行う
    - ▹ (&(*("abc"))) == "abc"
  - ○ 変数に関しては逆が成立する
    - ▹「*(&var)」==「var」
  - ○ アドレス演算子「&」の正体
    - ▹ 変数に対応したメモリセルの「アドレス」を得る演算子

ソフトウェア概論 A/B (2021/12/03)

多次元の配列とメモリモデル

多次元の配列

- 二次元の配列
  - 一次元の配列を二次元的に利用する事ができる (sample-012.c)
    - 二つの添字からアドレスを計算すればよい (sample-013.c)
  - 始めから二次元の配列を宣言する事ができる (sample-014.c)
    - 配列の要素のアドレス計算は、自動的に行われる (sample-015.c)
    - 実態は「(一次元)配列の(一次元)配列」だが、慣例により「二次元配列」と呼ぶ
  - 応用例
    - 行列は、二次元配列で表現可能 (sample-016.c)
- 多次元の配列
  - 一つの型から新しい配列型を作るだけなのでいくらでも大丈夫(?)

ソフトウェア概論 A/B (2021/12/03)

整数型とメモリモデル

整数型とメモリモデル

- 整数型とメモリモデル
  - 文字型変数はメモリセル一つに対応
    - では、整数型変数は ... ? / 実は、連続したメモリセルに保存される
- sizeof 演算子 (sample-017.c)
  - その型の変数が、何個(byte)のセルを占めるかを教えてくれる
    - sizeof(char) == 1
    - sizeof(int) == 4 (32 bit の場合)
  - int 型の変数は 4 つのセルで表現される

ソフトウェア概論 A/B (2021/12/03)

おしまい