

ソフトウェア概論 A/B

-- データ構造 (4) --

(配列とアドレス値)

数学科 栗野 俊一 / 渡辺 俊一

2016/11/11 ソフトウェア概

伝言

私語は慎むように !!

- 出席パスワード : 20161111
- 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(2016/10/28)の内容

□ 前回 (2016/10/28) の復習

○ 配列と関数の引数

- ▶ 配列名を関数の引数に指定できる：読み出し元と呼出し先で配列が共有できる
- ▶ cf. 従来(配列以外)は、コピーが渡される(値を渡す事ができる)

お知らせ

□ 本日の予定

○ データ構造 (4)

- ▶ 配列と文字列 (前回の資料)
- ▶ 配列とアドレス値
- ▶ sizeof と型変換

□ 本日の目標

○ 演習

- ▶ 課題の提出

前回 (2016/10/28) の課題

□ 前回 (2016/10/28) の課題

○ 課題 20161021-02: (先々週[2016/10/21]の課題の積み残し)

- ▶ ファイル名 : 20161021-02-QQQQ.c (QQQQ は学生番号)
- ▶ 内容 : 二次元行列の和、差、積
- ▶ ファイル形式 : テキストファイル(C 言語プログラムファイル)

○ 課題 20161021-03: (先々週[2016/10/21]の課題の積み残し)

- ▶ ファイル名 : 20161021-02-QQQQ.c (QQQQ は学生番号)
- ▶ 内容 : 二次元行列の和、差、積
- ▶ ファイル形式 : テキストファイル(C 言語プログラムファイル)

○ 課題 20161028-01: (これは今週[2016/11/11]に回す)

- ▶ ファイル名 : 20161028-01-QQQQ.c (QQQQ は学生番号)
- ▶ 内容 : 配列内の浮動小数点数の合計を求める Sum 関数
- ▶ ファイル形式 : テキストファイル(C 言語プログラムファイル)

○ 課題 20161028-02: (これは今週[2016/11/11]に回す)

- ▶ ファイル名 : 20161028-02-QQQQ.c (QQQQ は学生番号)
- ▶ 内容 : 文字配列に入った文字列の途中に文字を挿入する
- ▶ ファイル形式 : テキストファイル(C 言語プログラムファイル)

□ ※

- 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 (2016/11/11)

□ 本日 (2016/11/11) の課題

○ 課題 20161028-01: (これは先週[2016/10/28]の課題の積み残し)

▶ ファイル名 : 20161028-01-QQQQ.c (QQQQ は学生番号)

▶ 内容 : 配列内の浮動小数点数の合計を求める Sum 関数

▶ ファイル形式 : テキストファイル(C 言語プログラムファイル)

○ 課題 20161028-02: (これは先週[2016/10/28]の課題の積み残し)

▶ ファイル名 : 20161028-02-QQQQ.c (QQQQ は学生番号)

▶ 内容 : 文字配列に入った文字列の途中に文字を挿入する

▶ ファイル形式 : テキストファイル(C 言語プログラムファイル)

□ ※

○ 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

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 : 半角は表現できるが全角は無理

▶ 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なる
 - ▶ 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
 - ▶ (サイズが異なるので..) 同じプログラムが、システムによって異なる振舞をする: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い場合がある(無限の場合:整数、実数等)

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」!!