

ソフトウェア概論 A/B

-- データ構造 (3) --

(配列の様々な利用方法)

数学科 栗野 俊一 / 渡辺 俊一

2017/11/10 ソフトウェア概

伝言

私語は慎むように !!

- 出席パスワード : 20171110
- 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>
- 廊下側の一行は遅刻者専用です(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(2017/10/27)の内容

□ 前回 (2017/10/27) の復習

○ コーディングと I/O (入出力)

- ▶ 計算機(プログラム)の「内部表現(=データ)」は全て「数値」
- ▶ 現実(計算機の外)では「情報(様々な形式)」を扱いたい
- ▶ 情報とデータの対応規則がコーディング(規則)
- ▶ コーディング規則に従って、内部と外部の間の変換を行うのが I/O

○ 配列

- ▶ 配列 : 同じ型の変数をまとめたもの (cf. 構造体は、異なる型も可)
- ▶ 配列の特徴 : 添字(「式」が使える)で、要素(配列の一部となる変数)が間接参照できる

お知らせ

□ 本日の予定

○ データ構造 (3)

▶ 配列の様々な利用方法

□ 本日の目標

○ 演習

▶ 課題の提出

前回 (2017/10/27) の課題

□ 前回 (2017/10/27) の課題

○ 課題 20171020-02

- ▶ ファイル名 : 20171020-02-QQQQ.c (QQQQ は学生番号)
- ▶ 内容 : 構造体を利用し、平行移動を行う関数を作成する

○ 課題 20171020-03

- ▶ ファイル名 : PPCNAME-03-QQQQ.c (QQQQ は学生番号)
- ▶ 内容 : 3次元ベクトルの差の計算

○ 課題 20171027-01:

- ▶ ファイル名 : 20171027-01-QQQQ.c (QQQQ は学生番号)
- ▶ 内容 : 複素数型の四則

○ 課題 20171027-02:

- ▶ ファイル名 : 20171027-02-QQQQ.c (QQQQ は学生番号)
- ▶ 内容 : 二次元行列の和、差、積

○ 課題 20171027-03:

- ▶ ファイル名 : 20171027-03-QQQQ.c (QQQQ は学生番号)
- ▶ 内容 : 配列を使って 5 個の数を入力し、その 5 倍と $1/2$ を出す

□ ※

- 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 (2017/11/10)

□ 本日 (2017/11/10) の課題

○ 課題 20171110-01:

- ▶ ファイル名 : 20171110-01-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 20171110-02:

- ▶ ファイル名 : 20171110-02-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)

多次元配列

□ 配列：同じ物が、一列に並んだ物（一次元配列）

○ 配列の配列は？（配列が二次元に並んでいる）：二次元配列となる

▶ `int ary[3][3];` int 型変数 `ary[i][j]` が二次元 (3x3) に並んでいる

| | 0 | 1 | 2 |
|---|------------------------|------------------------|------------------------|
| 0 | <code>ary[0][0]</code> | <code>ary[0][1]</code> | <code>ary[0][2]</code> |
| 1 | <code>ary[1][0]</code> | <code>ary[1][1]</code> | <code>ary[1][2]</code> |
| 2 | <code>ary[2][0]</code> | <code>ary[2][1]</code> | <code>ary[2][2]</code> |

○ k 次元の配列の配列は？

▶ k+1 次元の配列になる

配列要素の参照

□ 配列要素の参照方法

- 基本は添字参照「`[]`」を利用する

 - ▶ 例 : `int ary[5];` の時 `ary[2]` は 3 番目の要素の参照

- 間接参照「`*`」で先頭の要素が参照できる (文字列の時と同じ)

 - ▶ 例 : `*ary` は `ary[0]` と同じ

- 配列名に整数値を加えると、配列の先頭の要素が「ない」ように振る舞う

 - ▶ 例 : `*(ary+3)` は `ary[3]` と同じ (文字列の時と同じ)

□ 配列要素参照「`[]`」と「`*`」の関係 (配列一般)

- 「`ARY[INDEX]`」と「`*(ARY+INDEX)`」は何時も同じ振舞いをする

□ 「配列名」の意味(詳細は後日)

- 「間接参照「`*`」や添字参照「`[]`」をする」事ができる「値」を表す

 - ▶ 「値」なので、関数に渡したり、関数の値にできる

配列と文字列

□ 文字列とは？

○ 文字の並びだった.. という事は..

▶ char の一次元配列が「文字列」？

□ 答は Yes でもあり No でもある

○ Yes : 「文字列」は「char の一次元配列」で実現されている

▶ char の一次元配列にできることは「文字列」にもできる

▶ 「文字列」を「char の一次元配列」のように扱って良い(変更はできないが)

▶ 「構造」(参照の仕方)は同じだが、「属性」(代入できるかどうか)は違う

○ No : 逆は真ではない(char の一次元配列が、文字列とは限らない)

▶ char の一次元の配列の要素を文字にし最後に EOS(0)を入れれば「文字列と同様に振る舞う」

▶ C 言語のコンパイラが、「文字列」を特別扱いしてくれる

配列と関数引数(注意!!)

□ 関数の引数にも配列を渡す事ができる

○ 結果：あたかも「配列自身」が渡されているように見える

▶ 関数内で、配列の要素の内容を書き換えると、呼出し元の配列の内容も変る

▶ 配列の最初のサイズは省略可能 ([] でよい) / 一次元配列は「*」でも良い

○ 比較：通常の変数(単純変数/構造体)は、「変数そのもの」ではなく「変数の値」が「コピーされて」渡される

▶ 関数内で引数変数の値を変更しても、呼出し元には影響しない

□ 関数の引数における配列の挙動

○ (まだ学んでいない..) 「ポインター」の概念が必要

▶ (近い内に話をするが..) 今回は、これには触れない

□ 関数における配列の扱い

○ しばらくは、「そんなもの」と捉える (後に、正しい理解をする)