

# ソフトウェア概論 A/B

## -- データ構造 (3) --

(様々な配列)

数学科 栗野 俊一 / 渡辺 俊一 ( TA: 栗原 望 / 小嶋 仁子 [M2] )

2018/12/14 ソフトウェア概

# 伝言

---

## 私語は慎むように !!

□ 出席パスワード : 20181214

□ 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

# 今後の予定

---

## □ 今後の予定(後ろから)

### ○ 2018/01/25 (講義最終日)

▷ 試験を行う

### ○ 2018/01/18 (講義最終日前) / 教室変更 (1434 教室)

▷ 模擬試験を行う (予定) / FILE 入出力

### ○ 2019/01/11

▷ データ構造 (5)

### ○ 2018/12/28, 2019/01/04

▷ 冬期休暇期間中 : この講義はない

### ○ 2018/12/21

▷ データ構造 (4)

### ○ 2018/12/14 (本日)

▷ データ構造 (3) : 様々な配列

# 前回(2018/12/07)の内容

---

## □ 前回 (2018/12/07) の復習

### ○ データ構造

- ▶ 複雑な対象を表すにはデータを「組み合わせる」必要がある
- ▶ 「複数のデータを一つのデータに見せる」仕組みがあると便利：データ構造

### ○ 構造体

- ▶ 異なる型のデータを組合せて一つのデータ型を作る仕組み

### ○ typedef ( [注意] この講義では、構造体を typedef と一緒に使う )

- ▶ 新しい型を作る仕組み ( 構造体に名前を付けて使う事ができる )
- ▶ 構文 : `typedef struct { 組み合わせ } 新しい型名`

### ○ 配列

- ▶ 同じ型のデータを組合せて一つのデータ型を作る仕組み(添字で要素が参照可能)

# お知らせ

---

## □ 本日の予定

### ○ データ構造 (3)

▷ 様々な配列

## □ 本日の目標

### ○ 演習

▷ 課題の提出

# 先週 (2018/12/07) の課題

---

## □ 先週 (2018/12/07) の課題

### ○ 課題 先週-01:

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

### ○ 課題 先週-02:

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

### ○ 課題 先週-03:

- ▷ ファイル名 : 先週-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)

# 今週 (2018/12/14) の課題

---

## □ 今週 (2018/12/14) の課題

### ○ 課題 今週-01:

- ▶ ファイル名 : 今週-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

### ○ 課題 今週-02:

- ▶ ファイル名 : 今週-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

### ○ 課題 今週-03:

- ▶ ファイル名 : 今週-03-QQQQ.c (QQQQ は学生番号)
- ▶ 内容 : 一行分の文字列を入力して、その中の文字列を全て大文字に変換する

## □ ※

- 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)

# 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みえる

- ▶ 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)