

# プログラミング演習 I 課題3(第1回)

宇都宮大学工学部情報工学科

1

## 概要

- 第1回 単純な整列アルゴリズム(1)
  - C言語の復習(配列, マクロ定義, 代入演算子)
  - バブルソート
- 第2回 単純な整列アルゴリズム(2)
  - C言語の復習(3項演算子)
  - 選択ソートと挿入ソート
- 第3回 高速な整列アルゴリズム
  - CSVファイルの利用
  - クイックソート
  - 課題

2

## 整列アルゴリズムとは

3

## 整列(sorting)とは？

|   |      | 男       | 女       | 合計      |
|---|------|---------|---------|---------|
| 1 | 宇都宮市 | 227,108 | 226,175 | 453,283 |
| 2 | 足利市  | 77,806  | 81,234  | 159,040 |
| 3 | 栃木市  | 40,232  | 42,030  | 82,262  |
| 4 | 佐野市  | 61,766  | 63,433  | 125,199 |
| 5 | 鹿沼市  | 51,525  | 52,883  | 104,408 |
| 6 | 日光市  | 46,840  | 49,189  | 96,029  |
| 7 | 小山市  | 78,348  | 77,490  | 155,838 |
| 8 | 真岡市  | 31,913  | 31,047  | 62,960  |
| 9 | 大田原市 | 37,466  | 37,756  | 75,222  |

整列  
(合計)

|   |      | 男       | 女       | 合計      |
|---|------|---------|---------|---------|
| 1 | 宇都宮市 | 227,108 | 226,175 | 453,283 |
| 2 | 足利市  | 77,806  | 81,234  | 159,040 |
| 7 | 小山市  | 78,348  | 77,490  | 155,838 |
| 4 | 佐野市  | 61,766  | 63,433  | 125,199 |
| 5 | 鹿沼市  | 51,525  | 52,883  | 104,408 |
| 6 | 日光市  | 46,840  | 49,189  | 96,029  |
| 3 | 栃木市  | 40,232  | 42,030  | 82,262  |
| 9 | 大田原市 | 37,466  | 37,756  | 75,222  |
| 8 | 真岡市  | 31,913  | 31,047  | 62,960  |

栃木県住民基本台帳(平成18年3月)より

4

## 整列の目的



- 整列自体に目的がある場合
  - 人口統計、成績表、パケット通信など
- 前処理としての整列
  - 整列自体が目的ではなく、別の目的の前処理として整列しておいたほうが良い場合がある。
  - 探索(住所録検索、トランプ、麻雀など)
- 整列は情報処理の基本的なアルゴリズム

5

## 整列アルゴリズムの分類

- 内部整列
  - 主記憶上で整列
- 外部整列
  - 外部記憶上で整列(マージソートが適する)
- 比較による整列
  - キーの大小を比較してレコードの交換を行う
- 比較によらない整列
  - キーの特別な性質を利用して整列(いつも使えるとは限らない)
  - ビンソート、分布数え上げソート、基数ソート

6

## 比較による内部整列の種類

- 単純な整列
  - バブルソート、選択ソート、挿入ソート
- やや高速な整列
  - シェルソート
- 高速な整列
  - クイックソート、ヒープソート
  - マージソート

7

## C言語の復習

8

## 一次元配列

データ型名 変数名[サイズ];

- char s[80];
- int a[10];
- int b[5], c[8];
- double d[20];

### ■ C言語では配列の添え字は0から始まる

- int a[10];と宣言した場合、利用できるのはa[0]からa[9]まで。a[10]は利用できない(コンパイラはチェックしない)

9

## 一次元配列の初期化

### ■ 一次元配列の初期化

int a[3];  
a[0] = 3;  
a[1] = 7;  
a[2] = 5;

=

int a[3] = {3, 7, 5};

=

int a[3] = {3, 7, 5},

最後にカンマ(,)があっても良い

### ■ 文字配列の初期化

char s[4] = {'p', 'e', 'n', '\0'};

=

char s[4] = "pen";

=

char s[] = "pen";

サイズを省略しても良い

char s[3] = "pen";

終端文字 '\0' が入るので、文字列長 + 1

サイズを省略しても良い

10

## 一次元配列を関数に渡す

```
#include <stdio.h>
/*配列の要素の総和を返す関数*/
int
sum(const int a[], int n) {
    int i;
    int s = 0;
    for (i = 0; i < n; i++) {
        s += a[i];
    }
    return s;
}
int
main(int argc, char** argv) {
    int a[5] = {1, 2, 3, 4, 5};
    printf("合計 = %d\n", sum(a, 5));
    return 0;
}
```

int sum(const int a[5], int n) =  
int sum(const int a[], int n)  
サイズを指定してもエラーにはならないが意味はない

配列の要素数を関数に教えてやる必要がある  
関数側ではどんな長さの配列がくるかわからない

複合代入演算子  
s = s + a[i]

•+= (a+=b; ⇔ a=a+b;)  
•-= (a-=b; ⇔ a=a-b;)  
•\*= (a\*=b; ⇔ a=a\*b;)  
•/= (a/=b; ⇔ a=a/b;)  
•%=, &=, ^=, |=, <=, >=

変数名(サイズなし)と  
配列の要素数

11

## 文字列を関数に渡す

```
#include <stdio.h>
/*文字列の長さを返す関数*/
size_t
strlen(const char str[]) {
    size_t len = 0;
    size_t i = 0;
    while(str[i++] != '\0'){
        len++;
    }
    return len;
}
int
main(int argc, char** argv) {
    char str[] = "UTSUNOMIYA";
    printf("長さ = %d\n", strlen(str));
    return 0;
}
```

strlen(const char\* str)

str[i]が'\0'でない限り

str[]={'U','T','S','U','N','O','M','I','Y','A','\0'}

```
/*文字列の長さを返す関数(定石記法)*/
size_t
strlen(const char* str) {
    size_t len = 0;
    while(*str++)
        len++;
    return len;
}
```

C言語のスキルアップのために  
•標準関数を実装してみる(まずはstring.hあたりから)  
•標準関数が用意されていない処理系もある

12

## 練習問題(1)

1. main()関数内でrand()関数を利用(#include <stdlib.h>が必要)して, 0以上100未満の整数の擬似一様乱数をn=10個生成し, それをint配列a[n]に格納する
2. 配列a[]の中身を確認(void print\_array(const int a[], int n)を利用)
3. 配列の中身を反転する関数void reverse\_int(int a[], int n)を作成せよ
4. 反転した配列a[]の中身を確認(void print\_array(const int a[], int n)を利用)

13

## 多次元配列

データ型名 変数名[サイズ1][サイズ2]...[サイズn];

- int a[2][3];
- double f[5][8][10];

### 多次元配列の初期化

```
int a[2][3];
a[0][0] = 1;
a[0][1] = 2;
a[0][2] = 3;
a[1][0] = 4;
a[1][1] = 5;
a[1][2] = 6;
```

```
int a[2][3] = {
    1,2,3,
    4,5,6
};
```

```
int a[][3] = {
    1,2,3,
    4,5,6
};
```

先頭のサイズは省略しても良い

カンマ(,)があっても良い

14

## マクロ定義 #define

### #define 文字列1 文字列2

- #define SIZE 100
- #define PI 3.14159265358979323846

### #define マクロ名(引数) 引数を含む文字列

- #define MUL(a,b) ((a)\*(b))
  - z = MUL(20,10); ⇔ z = ((20)\*(10));
  - #define MUL(a,b) a\*b
    - z = MUL(x+20,y+10); ⇔ z = x+20\*y+10;
  - #define ADD(a,b) (a)+(b)
    - z = ADD(x,y)\*10; ⇔ z=(x)+(y)\*10;

記号定数を用いると大域的に管理可能

マクロは便利で効率が良いが、その利用には注意が必要

15

## 練習問題(2)

1. #define INT\_SWAP(a,b) {int t;t=a;a=b;b=t;}を利用して配列の練習問題(1)で作成した中身を反転する関数void reverse\_int(int a[], int n)を改造せよ
2. 配列a[]の中身を確認main関数内で定義した文字列char str[]="UTSUNOMIYA"を反転(str[]="AYIMONUSTU")する関数void reverse\_char(char\* str)を作成せよ(但し、string.h利用不可)

16

# バブルソート

17

- n要素int配列aに適当な整数が入っている
- 配列を昇順(小から大)に整列

## バブルソートのアルゴリズム

- 配列の後ろから先頭に向かってスキャンしていき、もし隣り合う2つの要素の大小関係が逆であったら、それらを入れ替える

18

## バブルソートの過程(スキャン1回目)

|      |    |    |    |    |    |    |    |
|------|----|----|----|----|----|----|----|
| a[0] | 20 | 20 | 20 | 20 | 20 | 20 | 3  |
| a[1] | 6  | 6  | 6  | 6  | 6  | 3  | 20 |
| a[2] | 55 | 55 | 55 | 55 | 3  | 6  | 6  |
| a[3] | 87 | 87 | 87 | 3  | 55 | 55 | 55 |
| a[4] | 3  | 3  | 3  | 87 | 87 | 87 | 87 |
| a[5] | 45 | 13 | 13 | 13 | 13 | 13 | 13 |
| a[6] | 13 | 45 | 45 | 45 | 45 | 45 | 45 |

19

## バブルソートの過程(スキャン2回目)

比較対象外

|      |    |    |    |    |    |    |    |
|------|----|----|----|----|----|----|----|
| a[0] | 3  | 3  | 3  | 3  | 3  | 3  | 3  |
| a[1] | 20 | 20 | 20 | 20 | 20 | 6  | 6  |
| a[2] | 6  | 6  | 6  | 6  | 6  | 6  | 20 |
| a[3] | 55 | 55 | 55 | 13 | 13 | 13 | 13 |
| a[4] | 87 | 87 | 13 | 55 | 55 | 55 | 55 |
| a[5] | 13 | 13 | 87 | 87 | 87 | 87 | 87 |
| a[6] | 45 | 45 | 45 | 45 | 45 | 45 | 45 |

20

## バブルソートの過程(スキャン3回目)

比較対象外

|      |    |    |    |    |    |
|------|----|----|----|----|----|
| a[0] | 3  | 3  | 3  | 3  | 3  |
| a[1] | 6  | 6  | 6  | 6  | 6  |
| a[2] | 20 | 20 | 20 | 20 | 13 |
| a[3] | 13 | 13 | 13 | 13 | 20 |
| a[4] | 55 | 55 | 45 | 45 | 45 |
| a[5] | 87 | 45 | 55 | 55 | 55 |
| a[6] | 45 | 87 | 87 | 87 | 87 |

21

## バブルソートの過程(スキャン4回目)

比較対象外

|      |    |    |    |    |
|------|----|----|----|----|
| a[0] | 3  | 3  | 3  | 3  |
| a[1] | 6  | 6  | 6  | 6  |
| a[2] | 13 | 13 | 13 | 13 |
| a[3] | 20 | 20 | 20 | 20 |
| a[4] | 45 | 45 | 45 | 45 |
| a[5] | 55 | 55 | 55 | 55 |
| a[6] | 87 | 87 | 87 | 87 |

交換が1度も行われなければ  
終了しても良い

22

## バブルソートの過程(スキャン5回目)

比較対象外

|      |    |    |    |
|------|----|----|----|
| a[0] | 3  | 3  | 3  |
| a[1] | 6  | 6  | 6  |
| a[2] | 13 | 13 | 13 |
| a[3] | 20 | 20 | 20 |
| a[4] | 45 | 45 | 45 |
| a[5] | 55 | 55 | 55 |
| a[6] | 87 | 87 | 87 |

23

## バブルソートの過程(スキャン6回目)

比較対象外

|      |    |    |
|------|----|----|
| a[0] | 3  | 3  |
| a[1] | 6  | 6  |
| a[2] | 13 | 13 |
| a[3] | 20 | 20 |
| a[4] | 45 | 45 |
| a[5] | 55 | 55 |
| a[6] | 87 | 87 |

要素数n-1回のスキャンで終了

24

## バブルソートのアルゴリズム

- スキャンが1増えるたびに、比較の回数は1減る。
- あるスキャンにおいて、交換が一回も行われ  
ない＝整列されている
  - 各スキャンで交換が1度も行われなければ、以降  
のスキャンを行う必要はない。



25

## 練習問題(3)

1. main関数内でrand()関数を利用して0～  
RAND\_MAXの擬似乱数をn個生成、配列  
a[n]に格納
2. 配列a[]をバブルソートによって昇順(小さい  
ものから大きいもの)に並び替える関数void  
bubble\_sort(int a[], int n)を作成

26

## プログラム演習のための参考資料

- stdio.h,stdlib.h,time.hの場所
  - C:¥Program Files¥Microsoft Visual Studio  
10.0¥VC¥include

27