

# 計算機科学実験及び演習 1 報告書

## 課題 11

杉本風斗

学籍番号: 1029232337

提出日: 2012 年 7 月 25 日

### 1 外部仕様

```
$ ./gya-tasukete11
```

標準入力から入力を受け取る。1 行目にソーティングモジュールの番号を, 2 行目以降にソートするデータを空白または改行区切りを与える。

指定したソーティングモジュールを用いて入力されたデータをソートし, ソートしたデータを標準出力に出力する。

ソーティングモジュールの番号は, それぞれ 0 はバブルソートモジュール, 1 はクイックソートモジュールに対応する。

ソーティングモジュールの番号は以下のようにコマンド引数の 1 番目に与えることもできる。

```
$ ./gya-tasukete11 0 < data
```

#### 1.1 プログラム名 (コマンド名)

gya-tasukete11

#### 1.2 プログラム引数

ソーティングモジュールの番号 (与えない場合は, 標準入力から受け取る)。

#### 1.3 プログラムの機能

1 行目は用いるソーティングモジュールの番号, 2 行目以降に入力データを標準入力から受け取る。

ソーティングモジュールにはバブルソート, クイックソートを用いることができる。

入力データを指定されたソーティングモジュールでソートし, 1 行に 10 コずつ出力する.

## 1.4 入出力データおよび参照ファイル

指定された入力データを用いた.

## 1.5 実行例

バブルソートモジュールを指定し入力データを与える例を示す.

### 1.5.1 入力

```
0
3198 4399 2962 1572 2704 395 2537 46 672 0
12137 0 300 568 1794 498 3015 1284 1299 1439
(略)
712 1964 828 894 2293 2560 505 1202 432 0
0 2 0 0 1035 946 2616 2426 1147 1371
```

### 1.5.2 出力

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
(略)
```

## 1.6 エラー条件とエラー処理機能

不正なソーティングモジュール番号が入力に与えられたとき標準エラーにエラーメッセージを出力する.

### 1.6.1 入力例

```
$ ./gya-tasukete11 -1
```

### 1.6.2 エラー出力

そのソーティングモジュール番号は不正です: -1

## 1.7 アルゴリズムの流れ図

# 2 内部仕様

## 2.1 モジュール仕様

### 2.1.1 swap

```
void swap(int *a, int *b);
```

モジュールの機能 指定された 2 つの配列ポインタの中身を入れ替える.

モジュールインタフェース

a, b: int 型配列ポインタ

戻数 void

内部変数

int temp

論理 一時変数を宣言;

一時変数を用いて a と b を入れ替える;

### 2.1.2 bubblesort

```
void bubblesort(int *array, int n);
```

モジュールの機能 与えられたサイズ n の配列をバブルソートする.

モジュールインタフェース 引数 array: int 型配列ポインタ

出力 void

論理 for (0 <= i < n - 1)

for (0 <= j < n - 1)

if j+1 番目よりも j 番目のほうが大きい

swap(j 番目, j+1 番目);

### 2.1.3 qsort2

```
void qsort2(int *array, int p, int q);
```

モジュールの機能 配列 array を p から q までの範囲でクイックソートする.

モジュールインタフェース 引数 array: int 型配列ポインタ p,q: int 型区間

出力 void

内部変数 1. int pivot: ピボット

論理 if (p >= q)  
    return;

ピボットを配列から適当に選ぶ;  
配列の中身をピボットの前にピボットよりも小さい数を, 後ろにピボットよりも大きい数が  
くるようにする;

qsort2(p からピボットの直前);  
qsort2(ピボットの後ろから q まで);

#### 2.1.4 w\_qsort2

void w\_qsort2(int \*array, int n);

モジュールの機能 与えられたサイズ n の配列をクイックソートする.  
qsort2 に渡すだけ.

#### 2.1.5 input

void input(int \*array, int \*n);

モジュールの機能 標準入力からデータを受け取る.

モジュールインタフェース 1. array は入力データ列  
2. n は int 型ポインタ. 入力データのサイズを代入する.

論理 int i = 0;  
    while (一行読み取る) {  
        while (数をひとつ読み取って array に代入)  
            i++;  
    }  
  
    \*n = i;

#### 2.1.6 output

void output(int \*array, int n);

モジュールの機能 標準出力に array の中身を整形して出力する.

モジュールインタフェース 1. array は入力データ列  
2. n は入力データのサイズ

論理 for (配列 array)  
中身を出力;

### 3 プログラム開発の経過

問題の分析と解法の検討 10 分

モジュール構造設計 10 分

モジュール内論理設計/プログラミング 10 分

プログラムテスト, デバッグ 10 分

仕様書の作成 1ヶ月

### 4 感想

仕様書はクソだ.

## 付録

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_N 100000
#define MAX_OUTPUT 10
#define FUNC_N 2

void swap(int *a, int *b);
void bubblesort(int *array, int n);
void qsort2(int *array, int p, int q);
void w_qsort2(int *array, int n);
void input(int *array, int *n);
void output(int *array, int n);

int main(int argc, char *argv[]) {
    char buf[1000];
    int array[MAX_N];
    int n, func_i;

    void (*func[FUNC_N])(int *, int) = { bubblesort, w_qsort2 };

    // batch mode
    if (argc >= 2) {
        sscanf(argv[1], "%d", &func_i);
    } else {
        fgets(buf, 1000, stdin);
        sscanf(buf, "%d", &func_i);
    }

    if (!(0 < func_i && func_i < FUNC_N)) {
        fprintf(stderr, "そのソーティングモジュール番号は不正です: %d\n", func_i);
        exit(1);
    }

    input(array, &n);
    func[func_i](array, n);
    output(array, n);

    return 0;
}
```

```

// 入力行数を返す
void input(int *array, int *n) {
    char buf[MAX_N];
    int i = 0, start_i, read_i;

    while(fgets(buf, MAX_N, stdin) != NULL) {
        start_i = read_i = 0;
        while (sscanf(buf + start_i, "%d %n", &array[i++], &read_i) != EOF) {
            start_i += read_i;
        }
    }

    *n = i;
}

void output(int *array, int n) {
    int i;
    for(i = 0; i < n; i++) {
        if ((i+1) % MAX_OUTPUT == 0) {
            printf("%8d\n", array[i]);
        } else {
            printf("%8d", array[i]);
        }
    }
    printf("\n");
}

// array を区間 p,q でソートする
// p <= q
void qsort2(int *array, int p, int q) {
    int i, m, pivot = p;

    if (p >= q)
        return;

    // 先頭から3つとってその中央値をピボットにする
    if (p - q >= 2) {
        for (i = 0; i < 3; i++) {
            if ((array[p+(i-1)%3] < array[i] && array[i] < array[p+(i+1)%3]) || (array[p+(i+1)%3] < array[
                pivot = p;
            }
        }
    }
}

```

```

    swap(&array[pivot], &array[p]);

    for (i = p+1, m = p; i <= q; i++) {
        if (array[i] < array[p]) {
            swap(&array[++m], &array[i]);
        }
    }

    swap(&array[m], &array[p]);

    swap(&array[pivot], &array[p]);

    qsort2(array, p, m-1);
    qsort2(array, m+1, q);
}

void w_qsort2(int *array, int n) {
    qsort2(array, 0, n - 1);
}

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

void bubblesort(int array[], int n) {
    int i, j;

    for (i = 0; i < n - 1; i++)
        for (j = 0; j < n - 1; j++)
            if (array[j] > array[j+1])
                swap(&array[j], &array[j+1]);
}

```