

前期第 2 回課題

電子制御工学科：情報処理 2

担当教員：遠藤登

柴田健琉

(学籍番号：2024D14 名列番号：15)

提出日：2026 年 04 月 21 日

2026 年 04 月 20 日

1 はじめに

この課題のプログラムは以下の環境での実行が確認されている：

- OS: NixOS 25.11, Linux 6.19.12
- GCC: 15.2.0
- コンパイラフラグ: `-g -O1 -Wall -Wpedantic`

2 下準備：並び換えるデータの用意

課題に使用するデータは Linux の `/dev/urandom` で以下の `bash` コマンドで生成した。

乱数配列生成コマンド (X に任意の個数)

```
1 $ sed '1i\#pragma_once\' <(head -c X /dev/urandom | xxd -iC -n dataX) > dataX.h
```

出力されるヘッダーファイルには `unsigned char` 型の乱数配列 `dataX` と配列の長さ `dataX_len` が格納されている。

data10.h

```
1 #pragma once
2 unsigned char data10[] = {
3     0x4c, 0xa7, 0x72, 0x27, 0x5f, 0x7a, 0x49, 0x2e, 0xd2, 0x98
4 };
5 unsigned int data10_len = 10;
```

data100.h

```
1 #pragma once
2 unsigned char data100[] = {
3     0xf2, 0xfa, 0xfe, 0x5c, 0xab, 0x2c, 0x5c, 0x26, 0x9d, 0x54, 0x12, 0x56,
11    0xbf, 0x52, 0x42, 0x8c
12 };
13 unsigned int data100_len = 100;
```

data1000.h

```
1 #pragma once
2 unsigned char data1000[] = {
3     0x1a, 0x59, 0xcf, 0xee, 0xf0, 0x16, 0x17, 0x4e, 0x3e, 0xa0, 0x24, 0x97,
86    0xca, 0x30, 0x3e, 0x31
87 };
88 unsigned int data1000_len = 1000;
```

3 課題：3つのソートアルゴリズム

10 個, 100 個, 1000 個の乱数データに対し, 単純交換・単純選択・単純挿入ソートを実行し, 交換回数・比較回数を記録する.

3つのソートアルゴリズム

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  #include "data10.h"
6  #include "data100.h"
7  #include "data1000.h"
8
9  // struct to store swaps and comps during sorting
10 typedef struct {
11     size_t swaps;
12     size_t comparisons;
13 } Status;
14
15 // Common Sorting Algorithm Function Signature/Pointer
16 typedef void (*SortAlg)(unsigned char *, size_t, Status *);
17
18 // print first <head> and last <tail> elements of <arr>
19 void printArr(unsigned char *arr, size_t len, size_t head, size_t tail) {
20     for (size_t i = 0; i < len; i++) {
21         if (i >= len - tail || i < head) {
22             printf("%d_", arr[i]);
23         } else {
24             if (i % 50 == 0)
25                 printf(".");
26         }
27     }
28     putchar('\n');
29 }
30
31 // swap values
32 void swap(unsigned char *a, unsigned char *b) {
33     unsigned char t = *a;
34     *a = *b;
35     *b = t;
36 }
37
38 void bubble_sort(unsigned char *arr, size_t len, Status *stat) {
39     for (size_t i = 0; i < len - 1; i++) {
40         for (size_t j = 0; j < len - 1; j++) {
41             if (arr[j] > arr[j + 1]) {
42                 swap(&arr[j], &arr[j + 1]);
43                 stat->swaps++;
44             }
45         }
46     }
47 }
```

```

45         stat->comparisions++;
46     }
47 }
48 }
49
50 void selection_sort(unsigned char *arr, size_t len, Status *stat) {
51     for (size_t i = 0; i < len; i++) {
52         size_t min_idx = i;
53         for (size_t j = i; j < len; j++) {
54             if (arr[j] < arr[min_idx]) {
55                 min_idx = j;
56             }
57             stat->comparisions++;
58         }
59         swap(&arr[i], &arr[min_idx]);
60         stat->swaps++;
61     }
62 }
63
64 void insertion_sort(unsigned char *arr, size_t len, Status *stat) {
65     for (size_t i = 1; i < len; i++) {
66         for (size_t j = 0; j < i; j++) {
67             if (arr[j] > arr[i]) {
68                 swap(&arr[j], &arr[i]);
69                 stat->swaps++;
70             }
71             stat->comparisions++;
72         }
73     }
74 }
75
76 int main(void) {
77     unsigned char *datasets[3] = {data10, data100, data1000};
78     size_t datasetLengths[3] = {data10_len, data100_len, data1000_len};
79     char *labels[3] = {"bubble", "selection", "insertion"};
80     SortAlg algs[3] = {bubble_sort, selection_sort, insertion_sort};
81     unsigned char *buffer =
82         (unsigned char *) (malloc(sizeof(unsigned char) * 1000));
83     Status stats[3][3] = {{{0}}};
84
85     // For each 3 datasets, perform 3 different sorting algorithms on copied
86     // array
87     for (size_t n = 0; n < 3; n++) {
88         for (size_t m = 0; m < 3; m++) {
89             printf("%s_#%ld\n", labels[m], n + 1);
90             memcpy(buffer, datasets[n], datasetLengths[n]);
91             printf("Before:");
92             printArr(buffer, datasetLengths[n], 10, 10);
93             (*algs[m])(buffer, datasetLengths[n], &stats[n][m]);
94             printf("After:");
95             printArr(buffer, datasetLengths[n], 10, 10);
96         }
97         putchar('\n');

```

```

98     }
99
100    for (size_t i = 0; i < 3; i++) {
101        for (int j = 0; j < 3; j++) {
102            printf("%s_%ld: _swaps:_%ld,_comps:_%ld\n", labels[j],
103                datasetLengths[i], stats[i][j].swaps,
104                stats[i][j].comparisions);
105        }
106    }
107
108    return 0;
109 }

```

3.1 実行結果

```

report cc ~/Documents/nit-info-proc-2-S1/src/cls02 $ ../build/cls02/main
bubble #1
Before: 76 167 114 39 95 122 73 46 210 152
After: 39 46 73 76 95 114 122 152 167 210
selection #1
Before: 76 167 114 39 95 122 73 46 210 152
After: 39 46 73 76 95 114 122 152 167 210
insertion #1
Before: 76 167 114 39 95 122 73 46 210 152
After: 39 46 73 76 95 114 122 152 167 210

bubble #2
Before: 242 250 254 92 171 44 92 38 157 84 . 42 43 186 219 7 162 191 82 66 140
After: 2 5 7 8 11 14 16 18 20 23 . 238 239 240 242 242 244 247 248 250 254
selection #2
Before: 242 250 254 92 171 44 92 38 157 84 . 42 43 186 219 7 162 191 82 66 140
After: 2 5 7 8 11 14 16 18 20 23 . 238 239 240 242 242 244 247 248 250 254
insertion #2
Before: 242 250 254 92 171 44 92 38 157 84 . 42 43 186 219 7 162 191 82 66 140
After: 2 5 7 8 11 14 16 18 20 23 . 238 239 240 242 242 244 247 248 250 254

bubble #3
Before: 26 89 207 238 240 22 23 78 62 160 . . . . . 28 110 25 89 129 178 202 48 62 49
After: 0 1 1 1 2 2 2 2 3 3 . . . . . 253 254 254 254 254 254 254 255 255
selection #3
Before: 26 89 207 238 240 22 23 78 62 160 . . . . . 28 110 25 89 129 178 202 48 62 49
After: 0 1 1 1 2 2 2 2 3 3 . . . . . 253 254 254 254 254 254 254 255 255
insertion #3
Before: 26 89 207 238 240 22 23 78 62 160 . . . . . 28 110 25 89 129 178 202 48 62 49
After: 0 1 1 1 2 2 2 2 3 3 . . . . . 253 254 254 254 254 254 254 255 255

bubble 10: swaps: 20, comps: 81
selection 10: swaps: 10, comps: 55
insertion 10: swaps: 20, comps: 45
bubble 100: swaps: 2571, comps: 9801
selection 100: swaps: 100, comps: 5050
insertion 100: swaps: 2205, comps: 4950
bubble 1000: swaps: 243054, comps: 998001
selection 1000: swaps: 1000, comps: 500500
insertion 1000: swaps: 94085, comps: 499500
report cc ~/Documents/nit-info-proc-2-S1/src/cls02 $

```

図1: 実行結果

データ数	単純交換		単純選択		単純挿入	
	比較	交換	比較	交換	比較	交換
10	81	20	55	10	45	20
100	9801	2571	5050	100	4950	2205
1000	998001	243054	500500	1000	499500	94085

表1: ソートアルゴリズムとデータ数での比較・交換回数

付録 A 付録

A.1 開発環境

A.1.1 OS

筆者の開発環境では NixOS という Nix パッケージマネージャーを用いた Linux ベースの x86_64OS を使用している。執筆時点での OS バージョンは 25.11 (Xantusia) で, Linux カーネルバージョンは 6.19.12 である。

A.1.2 開発パッケージ

筆者の開発環境は Nix という専用の関数型言語 Nix で記述された宣言的で再現可能な Unix 系 OS 用のパッケージマネージャーを使用して構築されている。

このパッケージマネージャーを用いて以下のパッケージを導入した:

- gcc15 - GNU Compiler Collection Version 15
- gdb - GNU Debugger
- gnumake - GNU Make
- gf - GDB Frontend 2
- clang-tools - Clang に付属するツール郡のスタンドアローンパッケージ

執筆時点で使用したパッケージレポジトリ nixpkgs/nixos-unstable のハッシュは sha256-vPKLpjhIVWdDrfiUM8atW6YkIggCEKdSAlJPzzhkQlw= である。

A.1.3 環境記述ファイル - flake.nix

開発環境は \LaTeX の執筆環境と共に flake.nix にて宣言・定義されている。

開発に用いられる一時シェルの宣言箇所を以下に記す:

開発環境の一時シェル

```
195 devShells.cc = pkgs.stdenv.mkDerivation {
196   inherit name;
197   shellHook = ''
198     export PS1="${name} cc \w \$ "
199   '';
200   buildInputs = with pkgs; [
201     gcc15
202     gdb
203     gf
204     gnumake
205     clang-tools
206   ];
207 }
```