令和 6 年度 情報工学実験 I 報告書

実験題目

プログラミング演習 4

指導教員

丸山教員，安細教員，周教員

実験日

- 令和 6 年 10 月 02 日 (水)〜令和 6 年 10 月 16 日 (水)

レポート

- 提出締切日: 令和 6 年 10 月 30 日 (水)
- 受理最終日: 令和 6 年 11 月 20 日 (水)
- 提出日: 令和 6 年 _____ 月 _____ 日 (_____)

報告者

2 年 31 番 氏名 橋本 千聡

共同実験者

川和 李圭, 鈴木 隆生, 安田 れん

1. 実験の目的

プログラムの共同開発演習を通して、議論などを伴うチームでのプログラム作成手法を理解する。

2. 実験の概要

- 1・2 週目: 実行環境の確認及び C 言語サンプル実行確認
  作成分担調整、分担一覧や全体構成の資料作成
- 3・4 週目: 各自の担当箇所を作成、単体動作確認
- 5・6 週目: 各自の作成の関数を統合して動作確認
  レポート報告内容のまとめ、レポート作成

3. 演習課題の報告

プログラム全体の概要

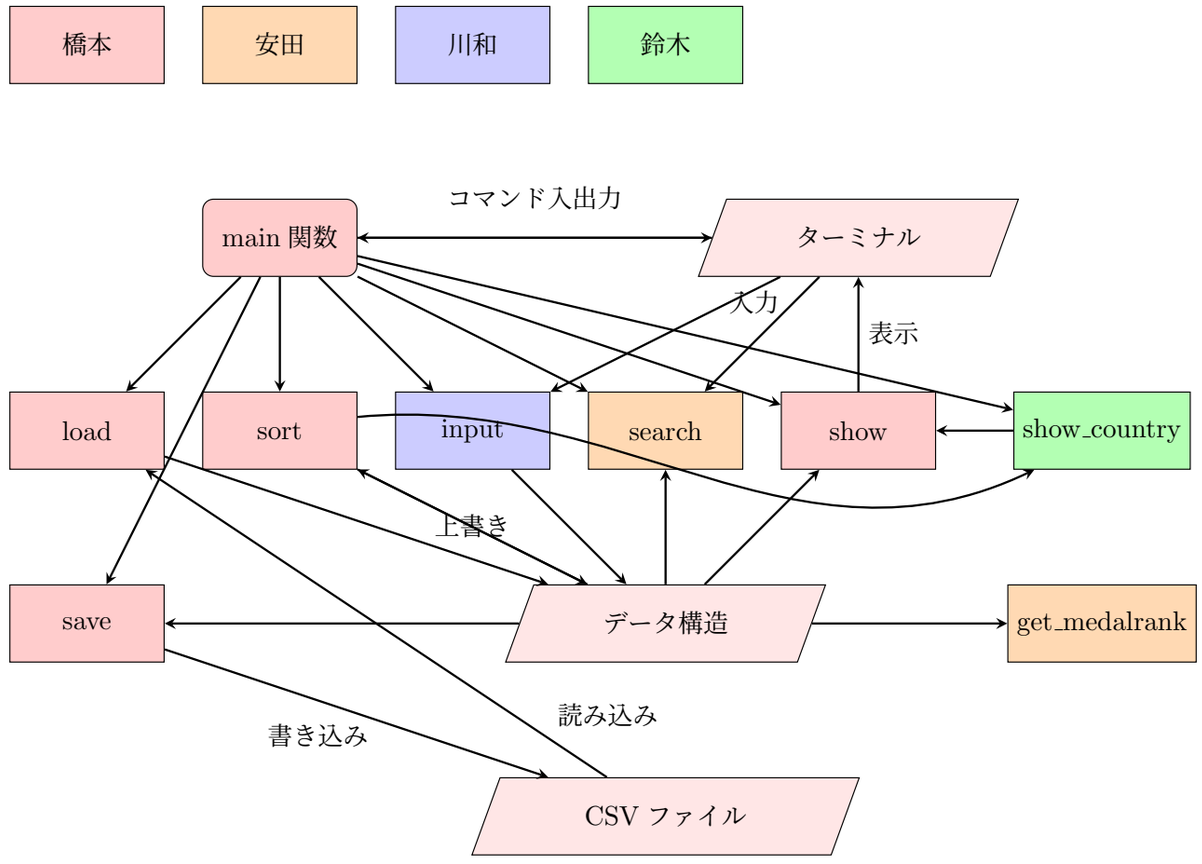

図 1: プログラムのフローチャート $+\alpha$

担当した機能の構成と説明

sort 関数

- 関数の説明: データ構造を並び替える
- 関数の入力: モード
- 関数の出力: なし (データ構造上書き)
- 関数の処理内容: merge sort を行い、mode によって国名金銀銅総メダル数のいずれかで並び替える

Listing 1: sort 関数のコード

```
#include "main.h"

// 安定な比較関数
int compare_by_mode(const country_data_type* a, const country_data_type* b,
    int mode) {
    if (mode == 0) {
        return strcmp(a->country, b->country); // 国名で昇順
    } else if (mode == 1) {
        return b->gold - a->gold; // 金メダル数で降順
    } else if (mode == 2) {
        return b->silver - a->silver; // 銀メダル数で降順
    } else if (mode == 3) {
        return b->bronze - a->bronze; // 銅メダル数で降順
    } else if (mode == 4) {
        return b->sum - a->sum; // 合計メダル数で降順
    }
    return 0; // デフォルトは等しいと見なす
}

// 安定なマージ関数
void merge(country_data_type arr[], int left, int mid, int right, int mode) {
    int n1 = mid - left + 1;
    int n2 = right - mid;
    country_data_type L[n1], R[n2];
    for (int i = 0; i < n1; i++)
        L[i] = arr[left + i];
    for (int j = 0; j < n2; j++)
        R[j] = arr[mid + 1 + j];

    int i = 0, j = 0, k = left;

    while (i < n1 && j < n2) {
        if (compare_by_mode(&L[i], &R[j], mode) <= 0) {
            arr[k] = L[i];
            i++;
        } else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

```

```
        while (i < n1) {
            arr[k] = L[i];
            i++;
            k++;
        }

        while (j < n2) {
            arr[k] = R[j];
            j++;
            k++;
        }
    }

    // マージソート関数
    void mergeSort(country_data_type arr[], int left, int right, int mode) {
        if (left < right) {
            int mid = left + (right - left) / 2;
            mergeSort(arr, left, mid, mode);
            mergeSort(arr, mid + 1, right, mode);
            merge(arr, left, mid, right, mode);
        }
    }

    // ソート関数
    void sort(int mode) {
        mergeSort(data, 0, data_size - 1, mode);
    }
```

load **関数**

- 関数の説明: CSV ファイルからデータを読み込む
- 関数の入力: ファイル名
- 関数の出力: なし (データ構造上書き)
- 関数の処理内容: 引数のファイル名から CSV ファイルを読み込み、データ構造に格納する

Listing 2: load 関数のコード

```
#include "main.h"

void load(char* filename){
    FILE *fp;
    char full_filename[256];
    snprintf(full_filename, sizeof(full_filename), "./data/%s.csv", filename);
    printd("load␣file:␣%s\n", full_filename);
    fp = fopen(full_filename, "r");
    if(fp == NULL){
        printf("ファイルが開けません\n");
        return;
    }
    printd("load␣data␣start\n");
    char buf[4][100];
    fscanf(fp, "%[^,],%[^,],%[^,],%s\n", buf[0], buf[1], buf[2], buf[3]);
    printd("header:␣%s,␣%s,␣%s,␣%s\n", buf[0], buf[1], buf[2], buf[3]);
```

```
17      data_size = 0;
18      while(fscanf(fp, "%[^,],%d,%d,%d\n", data[data_size].country, &data[data_size
            ].gold, &data[data_size].silver, &data[data_size].bronze) != EOF){
19          printd("load␣data[%d]:␣%s␣%d␣%d␣%d␣%d\n", data_size, data[data_size].
                country, data[data_size].gold, data[data_size].silver, data[data_size
                ].bronze, data[data_size].medal_rank);
20          data_size++;
21      }
22      get_medalrank(data_size);
23      get_sum();
24      fclose(fp);
25  }
```

save **関数**

- 関数の説明: データ構造を CSV ファイルに書き込む
- 関数の入力: ファイル名
- 関数の出力: なし
- 関数の処理内容: 引数のファイル名にデータ構造を書き込む

Listing 3: save 関数のコード

```
1   #include "main.h"
2
3   void save(char* filename){
4       FILE *fp;
5       char full_filename[256];
6       snprintf(full_filename, sizeof(full_filename), "./data/%s.csv", filename);
7       printd("save␣file:␣%s\n", full_filename);
8       fp = fopen(full_filename, "w");
9       if(fp == NULL){
10          printf("ファイルが開けません\n");
11          return;
12      }
13      printd("save␣data␣start\n");
14      fprintf(fp, "Country,Gold,Silver,Bronze\n");
15      for(int i = 0; i < data_size; i++){
16          fprintf(fp, "%s,%d,%d,%d\n", data[i].country, data[i].gold, data[i].
                silver, data[i].bronze);
17          printd("save␣data[%d]:␣%s␣%d␣%d␣%d\n", i, data[i].country, data[i].gold,
                data[i].silver, data[i].bronze);
18      }
19      fclose(fp);
20  }
```

show **関数**

- 関数の説明: データ構造を表示する
- 関数の入力: なし
- 関数の出力: なし
- 関数の処理内容: printf のフォーマット機能を駆使してテーブル形式でデータ構造を表示する

Listing 4: show 関数のコード

```
void show(){
    printd("show␣data\n");
    printf("+-------------------------------------------------+\n");
    printf("|␣␣␣␣␣␣国名␣␣␣␣␣␣␣␣␣␣␣␣␣|␣␣金␣␣|␣␣銀␣␣|␣␣銅␣␣|^^ef^^be^^92^^ef^^be
        ^^80^^ef^^be^^9e^^ef^^be^^99^^ef^^be^^97^^ef^^be^^9d^^ef^^bd^^b8|\n");
    printf("+----------------------+------+------+------+-------+\n");
    for(int i=0;i<data_size;i++){
        printf("|␣%20s␣|␣␣%02d␣␣|␣␣%02d␣␣|␣␣%02d␣␣|␣␣%3d␣␣|\n",data[i].country,
            data[i].gold,data[i].silver,data[i].bronze,data[i].medal_rank);
    }
    printf("+-------------------------------------------------+\n");
}
```

main 関数

- 関数の説明: メイン関数
- 関数の入力: 引数 DEBUG の有無
- 関数の出力: 標準出力
- 関数の処理内容: bash をベースとしたターミナルでのコマンド入力を受け付け、各関数を呼び出す
- その他: デバッグモードを有効にすると、各関数の詳細情報を表示する

Listing 5: main 関数のコード

```
#include "main.h"

country_data_type data[1000];
int data_size=0;
int debug_mode=0;

int main(int argc, char* argv[]){
    printf("\033[2J\033[0;0H");
    printf("Welcome␣to␣medal␣ranking␣system\n");
    printf("type␣help␣to␣show␣help\n");
    if(argc>1 && strcmp(argv[1],"debug")==0){
        debug_mode=1;
        printd("debug␣mode\n");
    }
    while(1){
        printf("$␣");
        char command[100];
        scanf("%s",command);
        printd("input␣command:␣%s\n",command);
        if(strcmp(command,"help")==0){
            printd("show␣help\n");
            printf("+---------------+--------------------------+\n");
            printf("|␣コマンド␣␣␣␣␣␣␣|␣説明␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|\n");
            printf("+---------------+--------------------------+\n");
            printf("|␣help␣␣␣␣␣␣␣␣␣␣|␣ヘルプを表示します␣␣␣␣␣␣␣␣|\n");
            printf("|␣input␣␣␣␣␣␣␣␣␣|␣データを追加します␣␣␣␣␣␣␣␣|\n");
            printf("|␣sort␣␣␣␣␣␣␣␣␣␣|␣データをソートします␣␣␣␣␣␣|\n");
            printf("|␣show␣␣␣␣␣␣␣␣␣␣|␣データ一覧を表示します␣␣␣␣|\n");
```

```
29              printf("|␣show_medalrank␣|␣メダルランク順に表示します|\n");
30              printf("|␣show_country␣␣␣|␣国名順に表示します␣␣␣␣␣␣␣␣|\n");
31              printf("|␣show_sum␣␣␣␣␣␣␣|␣合計メダル数順に表示します|\n");
32              printf("|␣search␣␣␣␣␣␣␣␣|␣データを検索します␣␣␣␣␣␣␣␣|\n");
33              printf("|␣exit␣␣␣␣␣␣␣␣␣␣␣|␣プログラムを終了します␣␣␣␣|\n");
34              printf("|␣load␣␣␣␣␣␣␣␣␣␣|␣データを読み込みます␣␣␣␣␣␣|\n");
35              printf("|␣save␣␣␣␣␣␣␣␣␣␣|␣データを保存します␣␣␣␣␣␣␣␣|\n");
36              printf("+----------------+--------------------------+\n");
37          }else if(strcmp(command,"input")==0){
38              printd("input␣data␣start\n");
39              input();
40              printd("input␣data␣end\n");
41              for(int i=0;i<5;i++) printf("\033[A\033[K");
42              printf("追加 [%d]:␣国名:%s␣金:%d␣銀:%d␣銅:%d,␣メダルランク
                    :%d\n",data_size,data[data_size-1].country,data[data_size-1].gold,
                    data[data_size-1].silver,data[data_size-1].bronze,get_medalrank(
                    data_size-1));
43          }else if(strcmp(command,"sort")==0){
44              printd("sort␣data\n");
45              printf("ソートモードを入力してください␣国順 0:,␣金 1:,␣銀 2:,␣銅 3:,␣合計
                    4:\n>>>␣");
46              int mode;
47              scanf("%d",&mode);
48              sort(mode);
49              printf("\033[A\033[K\033[A\033[K");
50              printf("ソート完了␣モード (:␣%d)\n",mode);
51          }else if(strcmp(command,"show")==0){
52              show();
53          }else if(strcmp(command,"show_medalrank")==0){
54              show_medalrank();
55          }else if(strcmp(command,"show_country")==0){
56              show_country();
57          }else if(strcmp(command,"show_sum")==0){
58              show_sum();
59          }else if(strcmp(command,"search")==0){
60              search();
61          }else if(strcmp(command,"load")==0){
62              char filename[100];
63              printf("ファイル名を入力してください␣>>>␣");
64              scanf("%s",filename);
65              printd("load␣data␣start\n");
66              load(filename);
67              printd("load␣data␣end\n");
68          }else if(strcmp(command,"save")==0){
69              char filename[100];
70              printf("ファイル名を入力してください␣>>>␣");
71              scanf("%s",filename);
72              printd("save␣data␣start\n");
73              save(filename);
74              printd("save␣data␣end\n");
75          }else if(strcmp(command,"exit")==0){
76              printd("プログラムを終了します\n");
77              break;
```

```
78          }else{
79              printf("%というコマンドはありません s\n",command);
80          }
81      }
82      return 0;
83  }
```

printd **関数**

- 関数の説明: デバッグモード時にのみ標準出力する関数
- 関数の入力: フォーマット文字列, 可変長引数
- 関数の出力: なし
- 関数の処理内容: デバッグモード時にのみ標準出力する

Listing 6: printd 関数のコード

```
1  void printd(const char *format, ...) {
2      if(debug_mode){
3          // 可変引数を処理するための準備
4          va_list args;
5          va_start(args, format);
6
7          // デバッグメッセージの前に追加する文字列
8          printf("\033[2m[DEBUG]␣");
9
10         // を使用して可変引数に対応する形でフォーマットされた出力を行う vprintf
11         vprintf(format, args);
12
13         // デバッグメッセージの終わりに追加する文字列
14         printf("\033[0m");
15
16         // 可変引数の処理を終了
17         va_end(args);
18     }
19 }
```

担当した機能の単体テストの方法と結果

単体テストの方法

1. 機能を更新します
2. git を使って dev ブランチなど各自のブランチに push します
3. google colaboratory を使って main 関数から作成した関数を呼び出します

結果

以下の URL から各自のコミットを確認できます。

- https://github.com/kazu-321/jouhoukougakuzikken_programming/commits/main?author=len-0202
- https://github.com/kazu-321/jouhoukougakuzikken_programming/commits/main?author=kawawarika

- https://github.com/kazu-321/jouhoukougakuzikken_programming/commits/main?author=ryuusei899

また、以下の URL から google colaboratory の結果を確認できます。
https://colab.research.google.com/github/kazu-321/jouhoukougakuzikken_programming/blob/main/colab.ipynb

統合したプログラムの結合テストの方法と結果

4. 実験の感想