

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

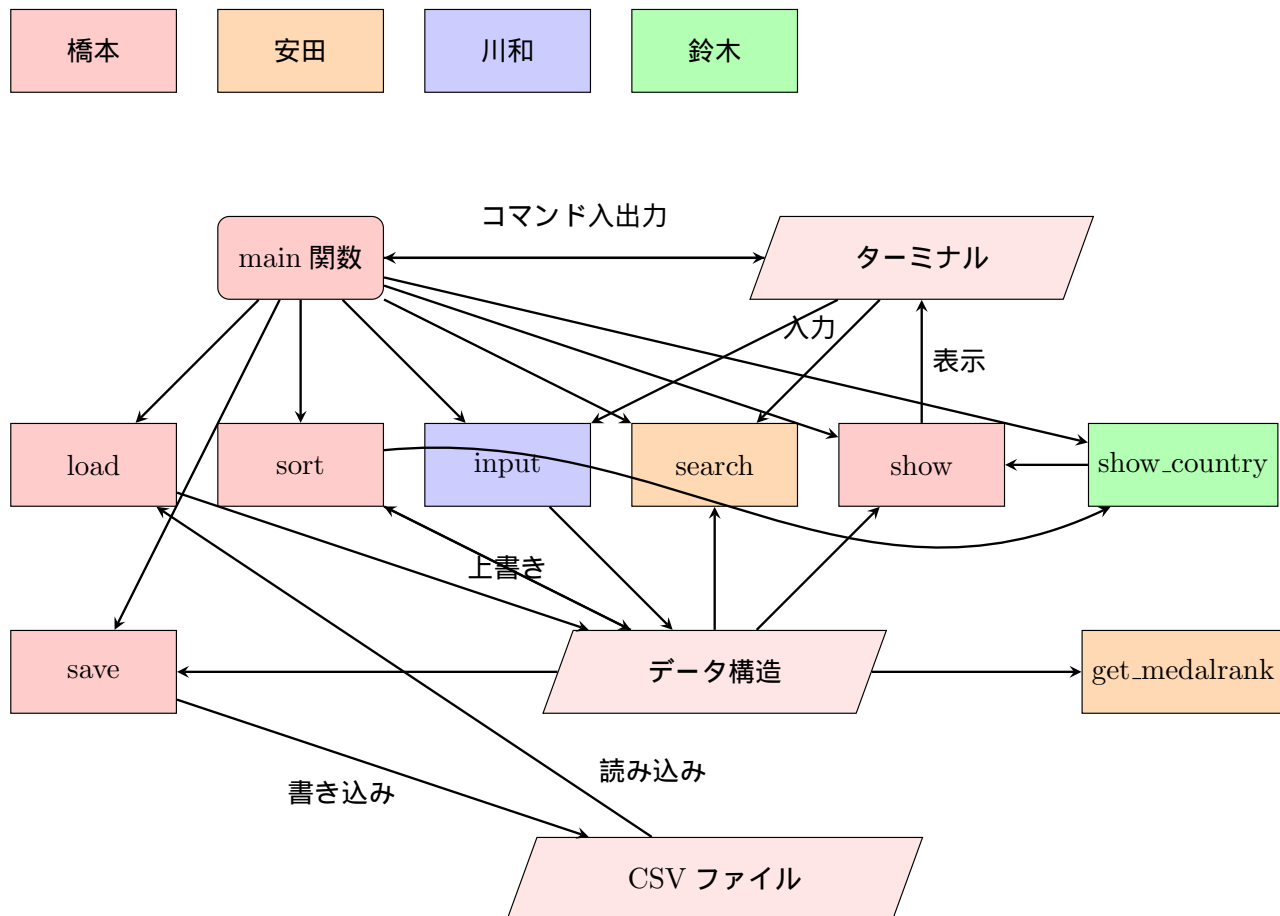


図 1: プログラムのフローチャート + α

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 (引数,int)
- 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: 比較関数

```
1 int compare_by_mode(const country_data_type* a, const country_data_type* b, int
   mode) {
2     if (mode == 0) return strcmp(a->country, b->country); // 国名で昇順
3     else if (mode == 1) return b->gold - a->gold; // 金メダル数で降順
4     else if (mode == 2) return b->silver - a->silver; // 銀メダル数で降順
5     else if (mode == 3) return b->bronze - a->bronze; // 銅メダル数で降順
6     else if (mode == 4) return b->sum - a->sum; // 合計メダル数で降順
7     return 0; // デフォルトは等しいと見なす
8 }
```

Listing 2: マージ関数

```
1 void merge(country_data_type arr[], int left, int mid, int right, int mode) {
2     int n1 = mid - left + 1;
3     int n2 = right - mid;
4     country_data_type L[n1], R[n2];
5     for (int i = 0; i < n1; i++) L[i] = arr[left + i];
6     for (int j = 0; j < n2; j++) R[j] = arr[mid + 1 + j];
7     int i = 0, j = 0, k = left;
8     while (i < n1 && j < n2) {
9         if (compare_by_mode(&L[i], &R[j], mode) <= 0) {
10             arr[k] = L[i];
11             i++;
12         } else {
13             arr[k] = R[j];
14             j++;
15         }
16         k++;
17     }
18     while (i < n1) {
19         arr[k] = L[i];
20         i++;
21         k++;
22     }
23
24     while (j < n2) {
25         arr[k] = R[j];
26         j++;
27         k++;
28     }
29 }
```

Listing 3: マージソート関数

```

1 void mergeSort(country_data_type arr[], int left, int right, int mode) {
2     if (left < right) {
3         int mid = left + (right - left) / 2;
4         mergeSort(arr, left, mid, mode);
5         mergeSort(arr, mid + 1, right, mode);
6         merge(arr, left, mid, right, mode);
7     }
8 }
9
10 // ソート関数
11 void sort(int mode) {
12     mergeSort(data, 0, data_size - 1, mode);
13 }

```

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 4: load 関数のコード

```

1 #include "main.h"
2
3 void load(char* filename){
4     FILE *fp;
5     char full_filename[256];
6     snprintf(full_filename, sizeof(full_filename), "./data/%s.csv", filename);
7     printf("load_file: %s\n", full_filename);
8     fp = fopen(full_filename, "r");
9     if(fp == NULL){
10         printf("ファイルが開けません\n");
11         return;
12     }
13     printf("load_data_start\n");
14     char buf[4][100];
15     fscanf(fp, "[%^],[%^],[%^],[%^]\n", buf[0], buf[1], buf[2], buf[3]);
16     printf("header: %s, %s, %s, %s\n", buf[0], buf[1], buf[2], buf[3]);
17     data_size = 0;
18     while(fscanf(fp, "[%^],[%^],[%^],[%^]\n", data[data_size].country, &data[data_size].gold, &data[data_size].silver, &data[data_size].bronze) != EOF){
19         printf("load_data[%d]: %s %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
- 関数の処理内容: 引数のファイル名にデータ構造を書き込む

Listing 5: 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     printf("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     printf("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].
17             silver, data[i].bronze);
18         printf("save_data[%d]: %s %d %d %d\n", i, data[i].country, data[i].gold,
19             data[i].silver, data[i].bronze);
20     }
21     fclose(fp);
22 }
```

show 関数

- 関数の説明: データ構造を表示する
- 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 6: show 関数のコード

```
1 void show(){
2     printf("show_data\n");
3     printf("+-----+\n");
4     printf("|          国名          | 金 | 銀 | 銅 | ^ef^be^92^ef^be
5         ^80^ef^be^9e^ef^be^99^ef^be^97^ef^be^9d^ef^bd^b8|\n");
6     printf("+-----+-----+-----+-----+-----+\n");
7     for(int i=0;i<data_size;i++){
8         printf("| %20s | %02d | %02d | %02d | %03d |\n", data[i].country,
9             data[i].gold, data[i].silver, data[i].bronze, data[i].medal_rank);
10    }
11    printf("+-----+\n");
12 }
```

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 7: main 関数のコード

```
1 #include "main.h"
2
3 country_data_type data[1000];
4 int data_size=0;
5 int debug_mode=0;
6
7 int main(int argc, char* argv[]){
8     printf("\033[2J\033[0;0H");
9     printf("Welcome to medal ranking system\n");
10    printf("type help to show help\n");
11    if(argc>1 && strcmp(argv[1],"debug")==0){
12        debug_mode=1;
13        printf("debug mode\n");
14    }
15    while(1){
16        printf("$ ");
17        char command[100];
18        scanf("%s",command);
19        printf("input command: %s\n",command);
20        if(strcmp(command,"help")==0){
21            printf("show help\n");
22            printf("+-----+-----+\n");
23            printf("| コマンド          | 説明\n");
24            printf("+-----+-----+\n");
25            printf("| help            | ヘルプを表示します\n");
26            printf("| input          | データを追加します\n");
27            printf("| sort           | データをソートします\n");
28            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            printf("input data start\n");
39            input();
40            printf("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         printf("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         printf("load_data_start\n");
66         load(filename);
67         printf("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         printf("save_data_start\n");
73         save(filename);
74         printf("save_data_end\n");
75     }else if(strcmp(command,"exit")==0){
76         printf("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 }

```

printf 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 8: printf 関数のコード

```
1 void printf(const char *format, ...) {
2     if(debug_mode){
3         // 可変引数进行处理するための準備
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 }
```

Qiita に詳しい記事として投稿しています。以下の URL から確認できます。

<https://qiita.com/kzs321kzs/items/e5d20f22c774d0731b6a#comment-a46dd6122ffd448d53aa>

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

統合テストの方法

1. 各自のブランチに main ブランチの変更を merge します
2. main ブランチ管理者が main ブランチに merge します
3. google colaboratory を使って統合後の結果を確認します

結果

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

4. 実験の感想

今回の実験を通して、チームでのプログラム開発の難しさを実感しました。

特に、各自の担当箇所を統合する際に、関数の引数や返り値の型が合わないなどの問題が発生しました。

これを解決するために、事前に関数の仕様を共有することが重要だと感じました。

また、git を使ったバージョン管理の重要性を再認識しました。

特に、コンフリクトが発生した際に、どのように解決するかを学ぶことができました。

最後に、google colaboratory を使って統合後のプログラムを実行することで、各自の担当箇所が正しく統合されているかを環境に依存せず確認することができました。