システムソフトウェア特論演習

課題０１　説明レポート

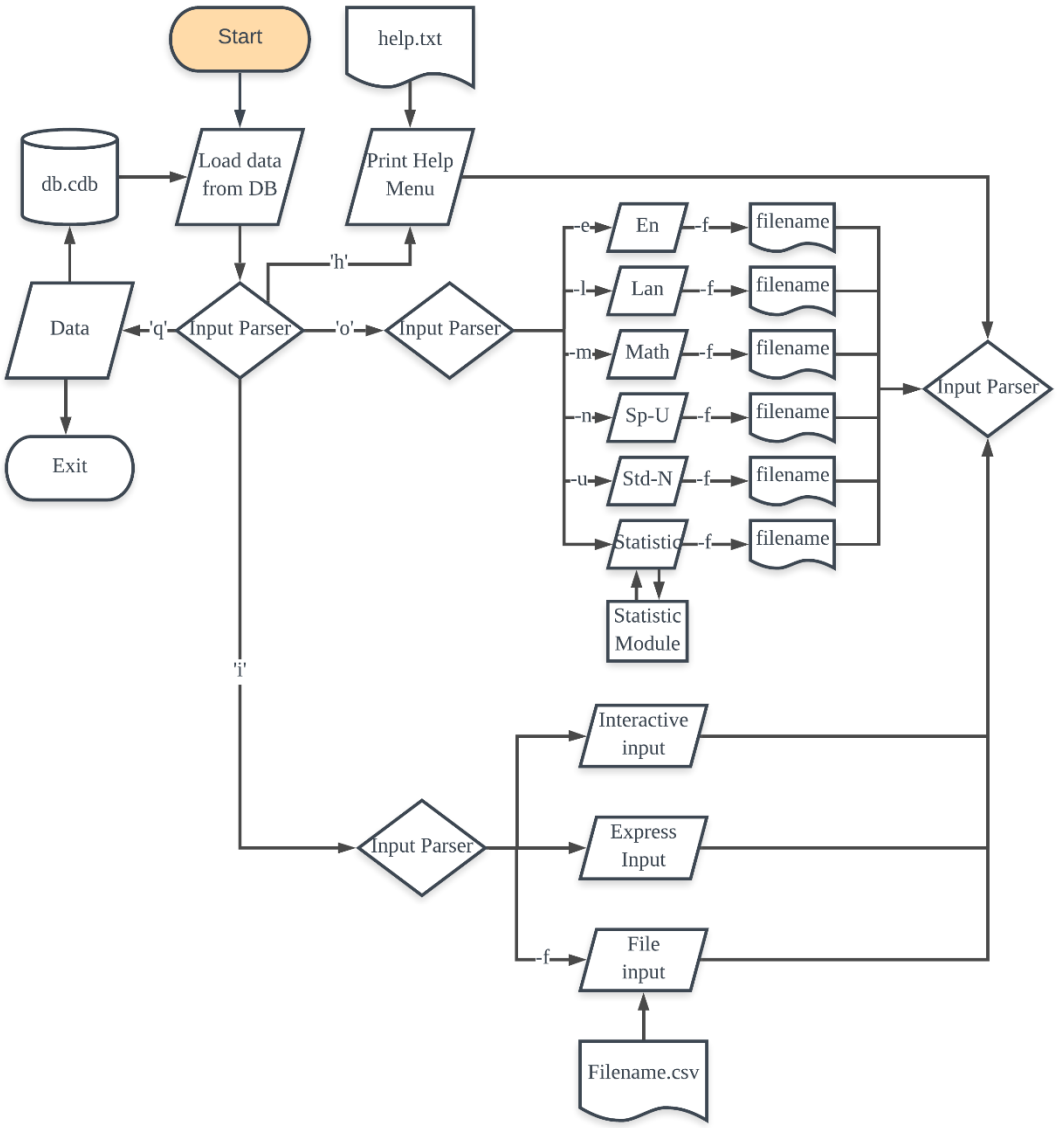
劉　立坤

学籍番号：2IE19337P

Contents

1. System Overview

1.1 System Diagram



1.2 Modules' functions

Module	Function
Input parser	Accept and analyze user's input then pass parameters and options to each function
Interactive data input	Accept user's input with prompt information
Express data input	Accept student information directly
Csv reader	Read student information from given csv file
Help	Print out help information
Statistic module	Calculating mean and standard deviation of all students, also sorting out each subject to find minimum and maximum grade
Output generator	Generate output according to user input, can both print to console or specified file

2. Code details

2.1 Constants and Variables

2.1.1 Global Constants and Variables

Global Constants

Constant name	Initial value	Function
MAX_INPUT	100	Max character count for user input
MAX_STD_NUM	50	Max student number
MAX_NAME_LENGTH	50	Max length for students' name
MAX_FILE_LENGTH	50	Max acceptable length for file name(input and output)
MAX_FILE_BUFFER_SIZE	500	Max acceptable character count for a single line in input file

Global Variables

Variable Name	Type	Function
std_cout	int	Store the total student number
stats_data	4X4 float matrix	Store statistical data Eng_AVG | Lan_AVG | Math_AVG | Total_AVG Eng_Dev | Lan_Dev | Math_Dev | Total_Dev Eng_MIN | Lan_MIN | Math_MIN | Total_MIN Eng_Max | Lan_MAX | Math_MAX | Total_MAX
std_info	struct std	Storing student data: char name: student name int eng_grade: English grade int lan_grade: language grade int math_grade: math grade int total_grade: total grade of 3 subjects

2.1.2 Local Variables

Variable name	Type	Function
*fp	FILE Pointer	Point to an open file for input and output
buf	String	File read buffer
std_name	String	Storing student name
temp	String	Storing temporary data
eng_grade	int	Temporarily storing English grade
lan_grade	int	Temporarily storing language grade
math_grade	int	Temporarily storing math grade
b_point	int	Count the break point of loop
Eng_sum	int	The total point of all students' English grade
Lan_sum	int	The total point of all students' language grade
Math_grade	int	The total point of all students' math grade
En_avg	Float	The mean of all students' English grade
Lan_avg	Float	The mean of all students' language grade
Ma_avg	Float	The mean of all students' math grade
Total_avg	Float	The mean of all students' total grade
En_stdev	Float	Standard deviation of all students' English grade

Lan_stdev	Float	Standard deviation of all students' language grade
Ma_stdev	Float	Standard deviation of all students' math grade
Total_stdev	Float	Standard deviation of all students' total grade
Sq_dev	Float	Temporary deviation during calculation process
True_name_len	Short	The actual name length of a given student
Input	String	Input from user
Usr_input	String	User input besides command
Length	Short	User input's length
File_name	String	User designated file name
Std_name	String	Student's name
F_flag	Short	Indicate whether file input or output command is activated.
Subcmd	Short	User's input subcommand in binary format: 0001 0000 find student with specified name 0000 1000 list all students 0000 0100 list all students with English grade 0000 0010 list all students with language grade 0000 0001 list all students with math grade 000x xxxx combinational command(Not available yet, empty slot only)

2.2 Program description

2.2.1 express_help_menu

Input: none

Return: none

```
void express_help_menu() {
    printf("i 成績の追加\n"
           "i -f ファイル入力\n"
           "i 名前 英語成績 国語成績 数学成績 省力入力\n"
           "o 集計結果表示\n"
           "o -f 集計結果ファイル出力\n"
           "o -u (-f ファイル名) 登録されている学生の名前一覧表示(ファイル出力)\n"
           "o -n 学生名 (-f ファイル名) 引数で与えた学生の成績を表示(ファイル出力)\n"
           "o -e (-f ファイル名) 英語の点数が高い方から順に、学生名とその点数を表示(ファイル出力)\n"
           "o -l (-f ファイル名) 国語の点数が高い方から順に、学生名とその点数を表示(ファイル出力)\n"
           "o -m (-f ファイル名) 数学の点数が高い方から順に、学生名とその点数を表示(ファイル出力)\n");
}
```

Description:

Program's built in help menu in case help file could not be accessed.

2.2.2 struct comparing functions

Input: two struct pointers that needs to be compared

Return: integer value which indicate compare result

```
int struct_cmp_by_name(void *a, void *b) {
    std *ia = (std *const *) a;
    std *ib = (std *const *) b;
    return strcmp(ia->name, ib->name);
} //Quick sort's compare function by Name

int struct_cmp_by_en(void *a, void *b) {
    std *ia = (std *const *) a;
    std *ib = (std *const *) b;
    return (int) (ia->eng_grade - ib->eng_grade);
} //Quick sort's compare function by English

int struct_cmp_by_lan(void *a, void *b) {
    std *ia = (std *const *) a;
    std *ib = (std *const *) b;
    return (int) (ia->lan_grade - ib->lan_grade);
} //Quick sort's compare function by lananese

int struct_cmp_by_ma(void *a, void *b) {
    std *ia = (std *const *) a;
    std *ib = (std *const *) b;
    return (int) (ia->math_grade -
ib->math_grade);
} //Quick sort's compare function by Math

int struct_cmp_by_total(void *a, void *b) {
    std *ia = (std *const *) a;
    std *ib = (std *const *) b;
    return (int) (ia->total_grade - ib->total_grade);
} //Quick sort's compare function by Total Grade
```

Description:

These functions are compare functions that required by C programming language's qsort(void *base, size_t nitems, size_t size, int (*compar)(const void *, const void*)) function

2.2.3 csv_reader

Input: data source's file name, current student in storage, data struct pointer

Return: none

```
void csv_reader(char *file_name, int *pstd_count, std *pstd_info)
{
    FILE *fp;
    char buf[MAX_FILE_BUFFER_SIZE] = "";
    char std_name[MAX_NAME_LENGTH] = "";
    char temp[MAX_NAME_LENGTH] = "";
    int en_grade, lan_grade, b_point, ma_grade, counter;
    counter = *pstd_count;
    if (buf == NULL)                                    //Memory check
        printf("No memory available.\n");

    if ((fp = fopen(file_name, "r")) == NULL)//File validation check
        printf("File could not be opened.\n");

    while (fgets(buf, 255, fp) != NULL) {
        int str_length = strlen(buf);          //Get current line's length

        for (int i = 0; i < str_length; i++) {
            if (buf[i] != ',')
                std_name[i] = buf[i];
            else {
                b_point = i + 1;
                break;
            }
        }                                                //Extracting name
        for (int i = b_point; i < str_length; i++) {
            if (buf[i] != ',')
                temp[i - b_point] = buf[i];
            else {
                b_point = i + 1;
                en_grade = atoi(temp);
                break;
            }
        }                                       //Extracting English grade
```

Description:

This function accepts user's designated file and read in csv data then write them into struct array and update total student count. This function is also used to read in database.

```
        for (int i = b_point; i < str_length; i++) {
            if (buf[i] != ',')
                temp[i - b_point] = buf[i];
            else {
                b_point = i + 1;
                lan_grade = atoi(temp);
                break;
            }
        }                                   //Extracting language grade

        for (int i = b_point; i < str_length + 2; i++) {
            if (buf[i] != '\n')
                temp[i - b_point] = buf[i];
            else {
                ma_grade = atoi(temp);
                break;
            }
        }                                   //Extracting Math grade
        strcpy(pstd_info[*pstd_count].name,
               std_name);           //Write current line's data into struct
        pstd_info[*pstd_count].eng_grade = en_grade;
        pstd_info[*pstd_count].lan_grade = lan_grade;
        pstd_info[*pstd_count].math_grade = ma_grade;
        pstd_info[*pstd_count].total_grade = en_grade + lan_grade
+ ma_grade;
        *pstd_count += 1;                   //Student count +1
        memset(std_name, 0,
               strlen(std_name));           //Clean up temporary
std_name
        memset(buf, 0,
               strlen(buf));                //Clean up temporary buffer
    }
    counter = *pstd_count - counter;
    fclose(fp);                             //Some Clean Up work
}//CSV Data Reader
```

2.2.4 interactive_data_input

Input: current student in storage, data struct pointer

Return: none

```
void interactive_data_input(int *pstd_count, std *pstd_info) {
    printf("%d 人目の成績を入力してください\n", *pstd_count + 1);
    printf("名前:");
    scanf("%s", pstd_info[*pstd_count].name);
    printf("英語:");
    scanf("%d", &pstd_info[*pstd_count].eng_grade);
    printf("国語:");
    scanf("%d", &pstd_info[*pstd_count].lan_grade);
    printf("数学:");
    scanf("%d", &pstd_info[*pstd_count].math_grade);
    printf("%d 人目の成績を登録しました\n", *pstd_count + 1);
    pstd_info[*pstd_count].total_grade =
            pstd_info[*pstd_count].eng_grade +
pstd_info[*pstd_count].lan_grade +
pstd_info[*pstd_count].math_grade;
    *pstd_count += 1;
}                       //Interactive Data Input Module
```

Description:

This function write prompt into console and accept user's input one at a time.

2.2.5 qck_data_input

Input: user's quick input data, string's length, current student in storage, data struct pointer

Return none

```
void qck_data_input(char *src_str, int str_length, int
*pstd_count, std *pstd_info) {
    char std_name[20] = "";  //Initialize some local variable
    char temp[20] = "";
    int en_grade, lan_grade, b_point, ma_grade;
    for (int i = 0; i < str_length; i++) {
        if ((src_str[i + 1] < 48) || (src_str[i + 1] > 57))
            std_name[i] = src_str[i];
        else {
            b_point = i + 1;
            break;
        }
    }                                       //Get the name
```

Description:

This function accepts user's express input data and break them into name, English grade, language grade and math grade. Then it takes the extracted data and write them into struct array.

```
    for (int i = b_point; i < str_length; i++) {
        if (src_str[i] != ' ')
            temp[i - b_point] = src_str[i];
        else {
            b_point = i + 1;
            en_grade = atoi(temp);
            break;
        }
    }//Get English Grade

    for (int i = b_point; i < str_length; i++) {
        if (src_str[i] != ' ')
            temp[i - b_point] = src_str[i];
        else {
            b_point = i + 1;
            lan_grade = atoi(temp);
            break;
        }
    }//Get language Grade

    for (int i = b_point; i < str_length + 2; i++) {
        if (src_str[i] != '\n')
            temp[i - b_point] = src_str[i];
        else {
            ma_grade = atoi(temp);
            break;
        }
    }//Get Math Grade

    strcpy(pstd_info[*pstd_count].name, std_name);          //Data Injection
    pstd_info[*pstd_count].eng_grade = en_grade;
    pstd_info[*pstd_count].lan_grade = lan_grade;
    pstd_info[*pstd_count].math_grade = ma_grade;
    pstd_info[*pstd_count].total_grade = en_grade + lan_grade + ma_grade;
    *pstd_count += 1;
}//Quick data input module
```

2.2.6 sorting_module

Input: current student in storage, data struct pointer, sorting option

Return: none

```
void sorting_module(int *pstd_count, std
*pstd_info, char opt) {
   switch (opt) {
      case 'e':       //Ascend order by English grade
         qsort(pstd_info, *pstd_count, sizeof(std),
struct_cmp_by_en);
         break;
      case 'l':    //Ascend order by language grade
         qsort(pstd_info, *pstd_count, sizeof(std),
struct_cmp_by_lan);
         break;
      case 'm':         //Ascend order by math grade
         qsort(pstd_info, *pstd_count, sizeof(std),
struct_cmp_by_ma);
         break;
      case 't':          //Ascend order by total grade
         qsort(pstd_info, *pstd_count, sizeof(std),
struct_cmp_by_total);
         break;
      case 'n':                  //Ascend order by name
         qsort(pstd_info, *pstd_count, sizeof(std),
struct_cmp_by_name);
         break;
      default:                              //Error option
         printf("Sort option error.\n");
   }
}              //Sorting function selector(e, l, m, t, n)
```

Description:

This function accepts current student number, data array and sorting option as parameters and then call quick sorting function and pass in required parameters.

2.2.7 statistics_module

Input: current student in storage, data struct pointer, statistic matrix pointer, file flag, file name

Return: none

```
void statistics_module(int *pstd_count, std *pstd_info, float
*pstats_data, short f_flag, char *file_name) {
   int eng_sum = 0;
   int lan_sum = 0;
   int math_sum = 0;
   float en_avg, lan_avg, ma_avg, total_avg, en_stdev, lan_stdev,
ma_stdev, total_stdev;
   float sq_dev = 0;
//Initiating required local variable

   for (int i = 0; i < *pstd_count; i++)
//Calculating sum of each subject
      eng_sum += pstd_info[i].eng_grade;

   for (int i = 0; i < *pstd_count; i++)
      lan_sum += pstd_info[i].lan_grade;

   for (int i = 0; i < *pstd_count; i++)
      math_sum += pstd_info[i].math_grade;

   en_avg = (float) eng_sum / *pstd_count;
//Calculating average grade of each subject
   lan_avg = (float) lan_sum / *pstd_count;
   ma_avg = (float) math_sum / *pstd_count;

   total_avg = en_avg + lan_avg + ma_avg;
//Calculating total average by add
```

Description:

This function accepts student number, student information struct array, statistic matrix pointer, file flag, file name as input.

When been called, this function first calculates the sum of each subject and calculate the average grade. The total average score is the sum of all subjects.

```
for (int i = 0; i < *pstd_count; i++)
{                       //Calculating each subject's standard deviation
    sq_dev += pow((float) (pstd_info[i].eng_grade - en_avg), 2);
}
en_stdev = sqrt(sq_dev / (float) *pstd_count);
sq_dev = 0;

for (int i = 0; i < *pstd_count; i++) {
    sq_dev += pow((float) (pstd_info[i].lan_grade - en_avg), 2);
}
lan_stdev = sqrt(sq_dev / (float) *pstd_count);
sq_dev = 0;

for (int i = 0; i < *pstd_count; i++) {
    sq_dev += pow((float) (pstd_info[i].math_grade - en_avg), 2);
}
ma_stdev = sqrt(sq_dev / (float) *pstd_count);
sq_dev = 0;

for (int i = 0; i < *pstd_count; i++) {
    sq_dev += pow((float) (pstd_info[i].total_grade - total_avg),
2);
}
total_stdev = sqrt(sq_dev / (float) *pstd_count);

*pstats_data = en_avg;          //Write results into statistic matrix
*(pstats_data + 1) = lan_avg;
*(pstats_data + 2) = ma_avg;
*(pstats_data + 3) = total_avg;
*(pstats_data + 4) = en_stdev;
*(pstats_data + 5) = lan_stdev;
*(pstats_data + 6) = ma_stdev;
*(pstats_data + 7) = total_stdev;
```

Description:

This part of code calculates the standard deviation of each subject as well as the deviation of total score by the equation:

$s_N = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}$

then write them into the statistic matrix.

```
    sorting_module(pstd_count, pstd_info, 'e');
//Find min and max score for each subject and write them into
statistic matrix
    *(pstats_data + 8) = pstd_info[0].eng_grade;
    *(pstats_data + 12) = pstd_info[*pstd_count - 1].eng_grade;
    sorting_module(pstd_count, pstd_info, 'l');
    *(pstats_data + 9) = pstd_info[0].lan_grade;
    *(pstats_data + 13) = pstd_info[*pstd_count - 1].lan_grade;
    sorting_module(pstd_count, pstd_info, 'm');
    *(pstats_data + 10) = pstd_info[0].math_grade;
    *(pstats_data + 14) = pstd_info[*pstd_count - 1].math_grade;
    sorting_module(pstd_count, pstd_info, 't');
    *(pstats_data + 11) = pstd_info[0].total_grade;
    *(pstats_data + 15) = pstd_info[*pstd_count - 1].total_grade;

if (f_flag == 1) {                                  //Write result into
file if it is required by user
    FILE *fp;
    if ((fp = fopen(file_name, "w+")) == NULL) {
        printf("Cannot open file to write.\n");
    }
    for (int i = 0; i < *pstd_count; i++) {
        fprintf(fp, "%s,%d,%d,%d\n", pstd_info[i].name,
pstd_info[i].eng_grade, pstd_info[i].lan_grade,
                pstd_info[i].math_grade);
    }
    fclose(fp);
    printf("%s　ファイルに出力しました\n", file_name);
}
```

Description:

This part of code finds the minimum and maximum of each subject as well as total grade respectively by calling sort module.

It then figures out whether user is requested a file output. If requested, the code generates a csv file with specified name.

```
    printf("       平均   :");
    for (int i = 0; i < 4; i++)
        printf("    %3.1f 点", *(pstats_data + i));

    printf("\n   最高点   :");
    for (int i = 12; i < 16; i++) {
        if (i == 15)
            printf(" ");
        printf("    %4d 点", (int) *(pstats_data + i));
    }

    printf("\n   最低点   :");
    for (int i = 8; i < 12; i++) {
        if (i == 11)
            printf(" ");
        printf("    %4d 点", (int) *(pstats_data + i));
    }
    printf("\n 標準偏差   :");
    for (int i = 4; i < 8; i++) {
        if (i == 7)
            printf(" ");
        printf("     %4.2f", *(pstats_data + i));
    }
    printf("\n");
}
```

Description:

This part of code prints out the statistic result into console.

2.2.8 output_generator

Input: file flag, file name, student name, current student in storage, data struct pointer, options

Return: none

```
void output_generator(short file_flag, char *file_name, char
*pstd_name, int *pstd_count, std *pstd_info, int option) {
   short true_name_len;
   switch (option) {
      case 16://Printing out specified student
         for (int i = 0; i < *pstd_count; i++) {
            if (strcmp(pstd_info[i].name, pstd_name) == 0) {
               if (file_flag == 1) {
                  FILE *fp;
                  fp = fopen(file_name, "w+");
                  fprintf(fp, "名前,英語,国語,数学\n");
                  fprintf(fp, "%s,%d,%d,%d", pstd_info[i].name,
pstd_info[i].eng_grade, pstd_info[i].lan_grade,
pstd_info[i].math_grade);
                  fclose(fp);
                  printf("%s　ファイルに出力しました\n", file_name);
               } else {
                  printf("名前");
                  true_name_len = strlen(pstd_info[i].name);
                  if ((true_name_len % 2) == 0) {
                     for (int j = 0; j < true_name_len; j++) {
                        printf(" ");
                     }
                  } else {
                     for (int j = 0; (j < true_name_len - 1); j++) {
                        printf(" ");
                     }
                  }
                  printf("英語　　数学　　国語　　合計\n");
                  printf("%s　%4d 点　%4d 点　%4d 点　%4d 点",
pstd_info[i].name, pstd_info[i].eng_grade, pstd_info[i].lan_grade,
 pstd_info[i].math_grade, pstd_info[i].total_grade);
               }}}
         break;
```

Description:

This function accepts parsed commands from user then give out desired output.

This portion of code finds out specified student and print out as file or to console according to requirements.

```
case 8:
//Print out logged student
   sorting_module(pstd_count, pstd_info, 'n');
   if (file_flag == 1) {
      FILE *fp;
      fp = fopen(file_name, "w+");
      fprintf(fp, "名前\n");
      for (int i = 0; i < *pstd_count; i++)
         fprintf(fp, "%s\n", pstd_info[i].name);
      fclose(fp);
      printf("%s　ファイルに出力しました\n",
file_name);
   } else {
      printf("登録学生一覧\n");
      for (int i = 0; i < *pstd_count; i++) {
         printf("%s\n", pstd_info[i].name);
      }
   }
   break;
```

Description:

This option prints out all student's name that is logged onto the system.

```
case 4:         //Print out student name and English grade by descend order
    sorting_module(pstd_count, pstd_info, 'n');
    true_name_len = strlen(pstd_info[*pstd_count - 1].name);
    sorting_module(pstd_count, pstd_info, 'e');
    if (file_flag == 1) {
        FILE *fp;
        fp = fopen(file_name, "w+");
        fprintf(fp, "--- 英語の成績 ---\n 登録者数: %d 人\n 名前,点数\n", *pstd_count);
        for (int i = *pstd_count - 1; i >= 0; i--)
            fprintf(fp, "%s %d\n", pstd_info[i].name, pstd_info[i].eng_grade);
        fclose(fp);
        printf("%s   ファイルに出力しました\n", file_name);
    } else {
        printf("--- 英語の成績 ---\n 登録者数   :   %d 人\n", *pstd_count);
        printf("名前");
        if ((true_name_len % 2) == 0) {
            for (int j = 0; j < true_name_len; j++) {
                printf(" ");
            }
        } else {
            for (int j = 0; (j < true_name_len - 1); j++) {
                printf(" ");
            }
        }
        printf("    点数\n");
        for (int i = *pstd_count - 1; i >= 0; i--)
            printf("%-*s %3d 点\n", (true_name_len + 4), pstd_info[i].name, pstd_info[i].eng_grade);
    }
    break;
```

Description:

This part of code firstly calls for sorting module to sort all students by name to find the name with the longest length in order to determine the space that required dynamically.

It then calls for sorting module to sort students by English score on ascending order then print them out in descending order.

If the file flag is raised, this code will also write result into the specified file.

```
        case 2:
            sorting_module(pstd_count, pstd_info, 'n');
            true_name_len =
strlen(pstd_info[*pstd_count - 1].name);
            sorting_module(pstd_count, pstd_info, 'l');
            if (file_flag == 1) {
                FILE *fp;
                fp = fopen(file_name, "w+");
                fprintf(fp, "--- 国語の成績 ---\n 登録者
数: %d 人\n 名前,点数\n", *pstd_count);
                for (int i = *pstd_count - 1; i >= 0; i--)
                    fprintf(fp, "%s %d\n",
pstd_info[i].name, pstd_info[i].lan_grade);
                fclose(fp);
                printf("%s   ファイルに出力しました\n",
file_name);
            } else {
                printf("--- 国語の成績 ---\n 登録者
数   :   %d 人\n", *pstd_count);
                printf("名前");
                if ((true_name_len % 2) == 0) {
                    for (int j = 0; j < true_name_len; j++) {
                        printf(" ");
                    }
                } else {
                    for (int j = 0; (j < true_name_len - 1);
j++) {
                        printf(" ");
                    }
                }
                printf("    点数\n");
                for (int i = *pstd_count - 1; i >= 0; i--)
                    printf("%-*s %3d 点\n",
(true_name_len + 4), pstd_info[i].name,
pstd_info[i].lan_grade);
            }
            break;
```

Description:

This part of code firstly calls for sorting module to sort all students by name to find the name with the longest length in order to determine the space that required dynamically.

It then calls for sorting module to sort students by language score on ascending order then print them out in descending order.

If the file flag is raised, this code will also write result into the specified file.

```
        case 1:
            sorting_module(pstd_count, pstd_info, 'n');
            true_name_len = strlen(pstd_info[*pstd_count - 1].name);
            sorting_module(pstd_count, pstd_info, 'm');
            if (file_flag == 1) {
                FILE *fp;
                fp = fopen(file_name, "w+");
                fprintf(fp, "--- 数学の成績 ---\n 登録者数: %d 人\n 名前,
点数\n", *pstd_count);
                for (int i = *pstd_count - 1; i >= 0; i--)
                    fprintf(fp, "%s %d\n", pstd_info[i].name,
pstd_info[i].math_grade);
                fclose(fp);
                printf("%s   ファイルに出力しました\n", file_name);
            } else {
                printf("--- 数学の成績 ---\n 登録者数   :   %d 人\n",
*pstd_count);
                printf("名前");
                if ((true_name_len % 2) == 0) {
                    for (int j = 0; j < true_name_len; j++) {
                        printf(" ");
                    }
                } else {
                    for (int j = 0; (j < true_name_len - 1); j++) {
                        printf(" ");
                    }
                }
                printf("   点数\n");
                for (int i = *pstd_count - 1; i >= 0; i--)
                    printf("%-*s %3d 点\n", (true_name_len + 4),
pstd_info[i].name, pstd_info[i].math_grade);
            }
            break;
        default:
            printf("Too many input arguments.\n");
    }
}
```

Description:

This part of code firstly calls for sorting module to sort all students by name to find the name with the longest length in order to determine the space that required dynamically.

It then calls for sorting module to sort students by math score on ascending order then print them out in descending order.

If the file flag is raised, this code will also write result into the specified file.

If no case was matched, the code will quit.

2.2.9 input_parser

Input: current student count, students' information struct array, statistic data matrix pointer

Return: 1 (quit signal)

```
int input_parser(int *pstd_count, std *pstd_info, float
*pstats_data) {
    char input[MAX_INPUT] = "";
    char usr_input[MAX_INPUT - 2] = "";
    short length;
    char file_name[MAX_FILE_LENGTH] = "";
    char std_name[MAX_NAME_LENGTH] = "";
    short f_flag = 0;
    short subcmd = 0;

    printf(":");
    fgets(input, MAX_INPUT, stdin);
    length = strlen(input) - 1;

    if (input[0] =='i')
{                                                   //Option for i
        if (length > 1) {
            if (input[2] == '-' && input[3] == 'f') {
                for (int i = 5; i < length; i++) {
                    file_name[i - 5] = input[i];
                }                                       //Get File Name

                csv_reader(&file_name, pstd_count, pstd_info);
//Load File Input Module
            } else {
                strcpy(usr_input,
                       input + 2);                  //choke input command
                qck_data_input(&usr_input, length - 2,
pstd_count,pstd_info);
            }  //Load Quick Data Input Module
        } else {
            interactive_data_input(pstd_count, pstd_info);
            getchar();
        }//Load Interactive Data Input Module
    }
```

Description:

This function accepts parameters from main function and accepts external input from user then analyze user input and dispatch global variables and parameters into different function.

This particular piece of code analyzes input command:

1. "i" only: load interactive information input module.

2. "i –f XXXX": get file name and pass parameters into csv_reader.

3. "i" with information: analyze the express input and pass them into quick data input module.

```
if (input[0] == 'q')
    return 1;//Exit Program Upon signal 1

if (input[0] == 'h') {
    FILE *fp;
    char buf[200] = "";
    printf("\n");
    if (buf ==
        NULL)
{
//Memory check
        printf("No memory available. Switching to
Streamlined help mode\n");
        express_help_menu();
    }

    if ((fp = fopen("help.txt", "r")) == NULL)
{     //File validation check
        printf("Help file doesn't exist or could not be
opened. Switching to Streamlined help mode\n");
        express_help_menu();
    }

    while (fgets(buf, 255, fp) != NULL)
        printf("%s", buf);

    free(buf);
//Some Clean Up work
    fclose(fp);
}
```

Description:

If “q” input is detected, this part of code will return a value of one which will cause main function to exit with 0.

If “h” input is detected, this part of code will open up help file (“help.txt”) and print out information in it.

In case file isn’t exist or can’t be opened, it will load previous express help function and provide streamlined help information.

```
if (input[0] == 'o') {
   if (length > 1) {
      for (short index = 1; index < length; index++) {
         if (input[index] == '-' && input[index + 1] == 'f') {
            f_flag = 1;      //File output option set to true
            for (int i = index + 3; i <= length; i++) {
               if (input[i] != ' ' && input[i] != '\n')
                  file_name[i - (index + 3)] = input[i];
               else break; } //Get File Name
         } else if (input[index] == '-' && input[index + 1] == 'n') {
            for (int i = index + 3; i < length; i++) {
               if (input[i] != '\0') {
                  if (input[i] == ' ' && input[i + 1] == '-')
                     break;
                  else {std_name[i - (index + 3)] = input[i]; }} else
                  break; } //Get student name
            subcmd |= 16; //Specific student option available: 0x10
         } else if (input[index] == '-' && input[index + 1] == 'u') {
            subcmd |= 8;} //List All Students: 0x08
         else if (input[index] == '-' && input[index + 1] == 'e') {
            subcmd |= 4;
         } //List all English Grades in Dec Order: 0x04
         else if (input[index] == '-' && input[index + 1] == 'l') {
            subcmd |= 2;
         }  //List all language Grades in Dec Order: 0x02

         else if (input[index] == '-' && input[index + 1] == 'm') {
          subcmd |= 1; }//List all Math Grades in Dec Order: 0x01}
      if (subcmd == 0)
         statistics_module(pstd_count, pstd_info, pstats_data, 1,
file_name);//Load Statistics Module with file output
      else
         output_generator(f_flag, file_name, std_name,
pstd_count, pstd_info, subcmd);//Initiating output
   } else {statistics_module(pstd_count, pstd_info, pstats_data, 0,
NULL);
}//Load Statistics Module without file output}
printf(" \n");
```

Description:

This part of code generates intended output.

If the command is “o” only, it will load statistic module with file flag = 0.

If detected file command, it will set file flag to 1 and save file name.

Then it will detect other user command and set corresponding data bits to 1 and then pass relative information to output generator.

2.2.10 Main

Input: none

Return: 0 (when quit normally)

```
int main() {
int std_count = 0;
float stats_data[4][4];//4X4 Statistical Data Array: AVG,
MAX, MIN, STDEV
std std_info[MAX_STD_NUM];
if (csv_reader(DB_NAME, &std_count, &std_info) == 1) {
   printf("%d students data successfully loaded\n",
std_count);
   system("CLS");
}
printf("q: 終了\nh: ヘルプ\n\n");
while (1) {
   if (input_parser(&std_count, &std_info, &stats_data))
      break;}
if (db_saver(&std_count, &std_info) == 9990)
   printf("DB save error\n");
return 0;
}
```

Description:

The main function is the program's entrence. It initializes global student count, statistic data matrix and students' information struct arry.

When executed, it will firstly print out initial information then calls input_parser and check for its return value with a dead loop. If detect a "1" as return, the program will end and return with code "0".

2.2.11 db_saver

Input: none

Return: 9990 (Database written error)

```
int db_saver(int *pstd_count, std *pstd_info) {
   FILE *fp;
   if ((fp = fopen(DB_NAME, "w+")) == NULL) {
      printf("DB writable Error (Permission problem
maybe?)");
      return 9990;}
   for (int i = 0; i < *pstd_count; i++) {
      fprintf(fp, "%s,%d,%d,%d\n", pstd_info[i].name,
pstd_info[i].eng_grade, pstd_info[i].lan_grade,
            pstd_info[i].math_grade);
   }
   fclose(fp);
   printf("Data saved");
}
```

Description:

This function save current data in student struct into db file

3. Operating results

課題 01-01

Grades input via dialog

```
q: 終了
h: ヘルプ

:i
1人目の成績を入力してください
名前:Taro
英語:87
国語:47
数学:81
1人目の成績を登録しました
```

Statistic result

```
:o
---  成績一覧  ---
登録者数  :  5  人
                英語        国語        数学        合計
   平均   :    68.0点    53.4点    62.8点    184.2点
  最高点  :      99点      99点      99点      297点
  最低点  :      22点      31点      23点       97点
標準偏差  :     33.54     27.67     27.32     67.98
```

Help menu

```
□試験結果入力オプション:
i                               対話式入力モード
i -f ファイル名                 ファイル入力(csvフォーマットのみ)
i 名前 英語成績 国語成績 数学成績  省力入力モード

出力オプション:
o                               集計結果を表示する
o -f ファイル名                 集計結果を表示するの上、ファイル出力
o -u (-f ファイル名)            登録されている学生の名前一覧表示(ファイル出力)
o -n 学生名 (-f ファイル名)     引数で与えた学生の成績のみを表示(ファイル出力)
o -e (-f ファイル名)            英語の点数が高い方から順に、学生名とその点数を表示(ファイル出力)
o -l (-f ファイル名)            国語の点数が高い方から順に、学生名とその点数を表示(ファイル出力)
o -m (-f ファイル名)            数学の点数が高い方から順に、学生名とその点数を表示(ファイル出力)

*: 1.出力ファイルはcsvフォーマットのみです、ファイル名の最後に".csv"を追加してください
   2.出力オプションとファイルオプションの位置を切り替えることができます
```

quit

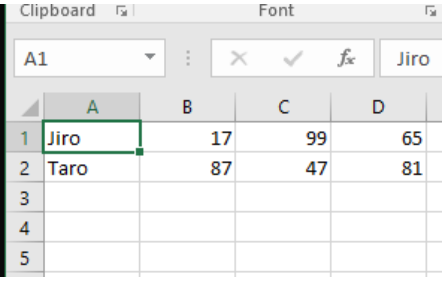
```
:q
Data saved
C:\Users\Konomi-Lab\Desktop\Projects\School-Work\System_Software\assignment1\cmake-build-debug>
```

課題 01-02

Express input function

```
:i Jiro 17 99 65
2人目の成績を登録しました
```

Statistic output with file saving



File input function

課題 01-03

5 output option

```
:o -u
登録学生一覧
Jiro
Taro

:o -n Taro
名前     英語     数学     国語      合計
Taro     87点     47点     81点     215点

:o -e
--- 英語の成績 ---
登録者数  :  2人
名前        点数
Taro        87点
Jiro        17点

:o -l
--- 国語の成績 ---
登録者数  :  2人
名前        点数
Jiro        99点
Taro        47点

:o -m
--- 数学の成績 ---
登録者数  :  2人
名前        点数
Taro        81点
Jiro        65点
```

5 output option with file output function

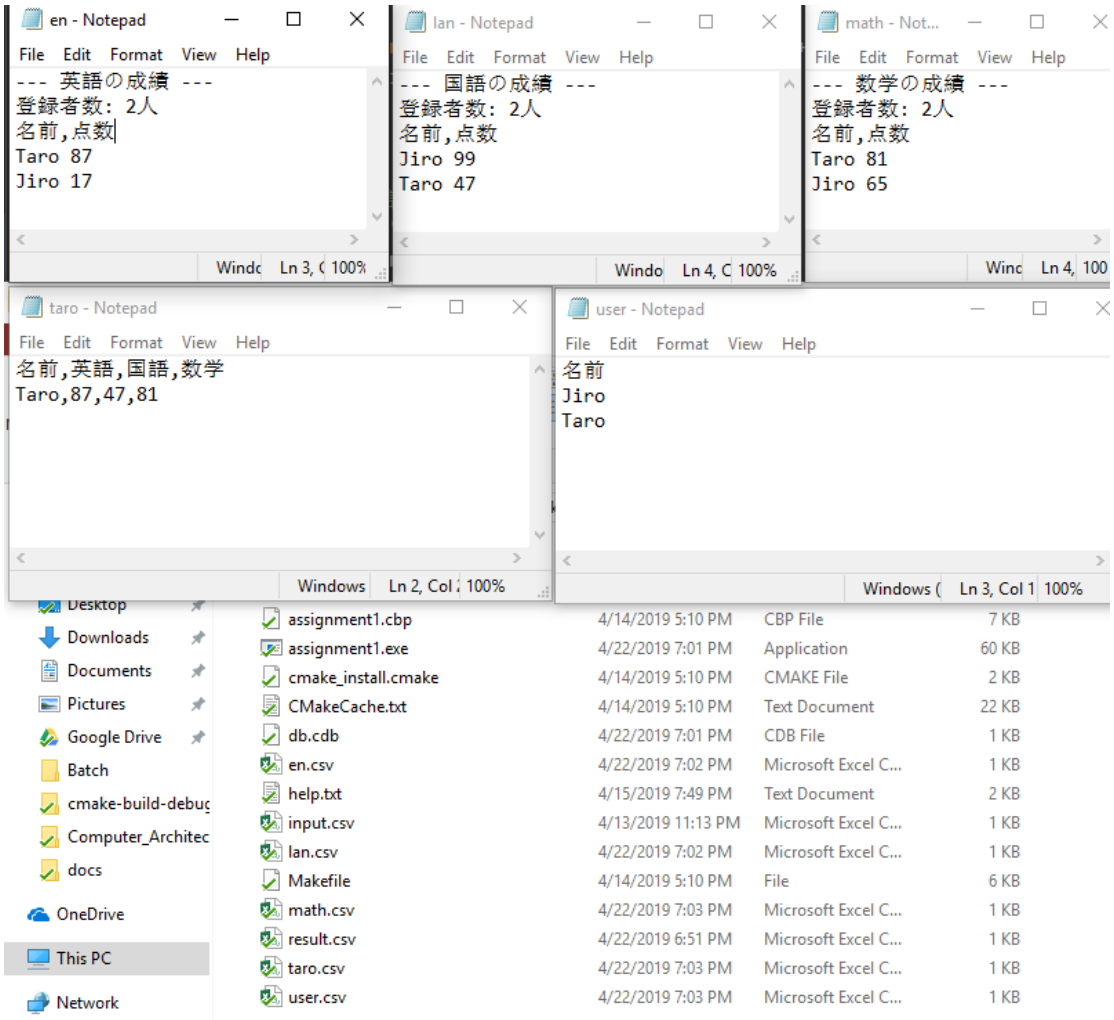
```
:o -f en.csv -e
en.csv  ファイルに出力しました

:o -f lan.csv -l
lan.csv  ファイルに出力しました

:o -f math.csv -m
math.csv  ファイルに出力しました

:o -n Taro -f taro.csv
taro.csv  ファイルに出力しました

:o -u -f user.csv
user.csv  ファイルに出力しました
```


en - Notepad
File Edit Format View Help
--- 英語の成績 ---
登録者数: 2人
名前,点数
Taro 87
Jiro 17
lan - Notepad
File Edit Format View Help
--- 国語の成績 ---
登録者数: 2人
名前,点数
Jiro 99
Taro 47
math - Not...
File Edit Format View Help
--- 数学の成績 ---
登録者数: 2人
名前,点数
Taro 81
Jiro 65
taro - Notepad
File Edit Format View Help
名前,英語,国語,数学
Taro,87,47,81
user - Notepad
File Edit Format View Help
名前
Jiro
Taro
Desktop
Downloads
Documents
Pictures
Google Drive
Batch
cmake-build-debug
Computer_Architec
docs
OneDrive
This PC
Network
assignment1.cbp 4/14/2019 5:10 PM CBP File 7 KB
assignment1.exe 4/22/2019 7:01 PM Application 60 KB
cmake_install.cmake 4/14/2019 5:10 PM CMAKE File 2 KB
CMakeCache.txt 4/14/2019 5:10 PM Text Document 22 KB
db.cdb 4/22/2019 7:01 PM CDB File 1 KB
en.csv 4/22/2019 7:02 PM Microsoft Excel C... 1 KB
help.txt 4/15/2019 7:49 PM Text Document 2 KB
input.csv 4/13/2019 11:13 PM Microsoft Excel C... 1 KB
lan.csv 4/22/2019 7:02 PM Microsoft Excel C... 1 KB
Makefile 4/14/2019 5:10 PM File 6 KB
math.csv 4/22/2019 7:03 PM Microsoft Excel C... 1 KB
result.csv 4/22/2019 6:51 PM Microsoft Excel C... 1 KB
taro.csv 4/22/2019 7:03 PM Microsoft Excel C... 1 KB
user.csv 4/22/2019 7:03 PM Microsoft Excel C... 1 KB