1 目的

これまで習ってきた C 言語の知識を利用して、比較的大きなプログラムを書くことで、プログラミング言語の利用方法のまとめを行うと同時に、複雑なアルゴリズムの構築方法を学ぶことを目的とする。

2 原理

2.1 ファイルからの読み込み

ファイルの読み込み (または書き込み) には、まずファイルを利用できるようにする「ファイルのオープン」が必要であり、コード 2.1 のような形で可能となる。

コード 2.1 ファイルからの読み込み 1

```
1        FILE *file; // ファイルポインタの定義
2        file = fopen("file_name","r"); // ファイルのオープン  r はの意味 read
3         ( ファイルを使った処理を書く )
4        fclose(file);
```

また、一般的に数値をファイルから読み込むときは、以下のような「fscanf」を利用する。フォーマットはコード 2.2 のようであるが、基本的なフォーマットはファイルポインタ以外は、scanf と同じである。

コード 2.2 ファイルからの読み込み 2

```
1    fscanf(*file,"\%d",&a);
```

2.2 配列

配列は、同じ名前の (同じ機能を持つ) 変数を大量に作成するために用いられる変数である。また、配列は"番号"で管理することができ、この番号を出席番号などに対応させて使うこともできる。配列の宣言は、コード 2.3 のようになる。

コード 2.3 配列の宣言

```
1    int a[10];
2    float b[10];
```

ただし、前者の場合、用意される配列は a[0],a[1],.....,a[8],a[9] である。配列の宣言時は配列の個数を用いて行う、宣言によって作成される配列の番号を指定するときは、番号-1 番の数値を指定する必要がある。その理由は、配列は使用上 0 番目から格納されるためである。

2.3 並び替え (ソート)

並べ替え (ソート) は、データを大きい順や小さい順に並べることである。逐次比較して、並べ替えるようなアルゴリズムは、簡単であるが、時間のかかる方法である。また「クイックソート」「バブルソート」などのアルゴリズムは複雑であるが高速となる方法もある。

2.3.1 バブルソート

バプルソート (bubblesort) の原理は、配列の後ろから先頭に向かってスキャンしていき、もし隣り合う 2 つの要素の大小関係が逆であったら、それを入れ換えるという単純なものである。まず最初は、n-1 番目（最後）の要素から 0 番目（先頭）の要素までをスキャンする。その結果、最小の要素が 0 番目（先頭）に浮かび上がっていく。2 回目のスキャンでは、n-1 番目から 1 番目までの要素をスキャンして、結果として次に小さい要素が配列の 1 番目の位置に浮かび上がってくる。さらにスキャンを繰り返していき、最終的に n-1 回目のスキャンで整列が完了する。整列の過程が、ちょうど軽い要素が浮き上がっていくように見えるため、バブルソート（バブル：bubble とは泡のこと）と呼ばれている [1]。

2.3.2 クイックソート

クイックソート (quick. sort) の原理は、クイックソートの計算量は 0(nlogn) で、n 個の要素 a[O],a[1],...,a[n-1] をもった配列 a を整列することを考える。まず、ある適当な要素 x を 1 個選び出し、それを最終的に置かれるべき位置 a[v] に移動する。この要素 x を枢軸 (pivot) と呼びます。このとき同時に、x より大きな要素は a[v] より右に、x より小さな要素は a[v] より左へと振り分けを行います。この操作によって、配列 a を次の 3 つの部分 (1) a[v] よりも小さい要素、a[O], a[1],...,a[v-1] (2)a[v] (3)a[v] よりも大きな要素 a[v+l],a[v+2],...,a[n-l] にわけられる。

a[v] は最終的な位置に移動されていて，その It a[v] より小さい要素のみが、その右には a[v] より大きな要素のみが存在するので、2 つの部分配列 (1) と (3) をそれぞれ別個に整列すれば、配列 a 全体の整列が完了する。部分配列がただ 1 つの要素になるまで、この分割を繰り返す。

このように、大きな問題を複数の小さな問題に分割して確固撃破すると言うアプローチは、分割統治 (divide and rule または divid and conquer) と呼ばれる [1]。

2.3.3 ソートの安定性について

ソートの安定性とは、あるデータをソートした時に、元の順番を保持するか、保持しないかである。ソートは上記のソート意外にも種類がたくさんあり、それらのソートも含めたソートの安定性についての表を表 2.1 に示す。

表 2.1 ソートの安定性

安定であるソート	安定でないソート
バブルソート	シェルソート
挿入ソート	ヒープソート
マージソート	選択ソート
	クイックソート

2.4 ファイルへの書き出し

一般的に数値をファイルへ書き込む時はまず読み込みと同様にファイルのオープンをしてから (ただし、"r" を、"w"は write の意味)、以下のような「fprintf」を利用する。フォーマットはコード 2.4 のようであるが、

基本的なフォーマットはファイルポインタ以外は printf と同じである。

コード 2.4　ファイルへの書き込み

```
1    fprintf(*file, "\%d",a)
```

2.5　表計算ソフトからの読み込み表計算ソフトへの書き出し

エクセルや LibreOffice の Cale などの表計算ソフトは、保存されているデータを 10 20 30 や「10,20,30」など、C 言語で簡単に読み込めるような、数値をスペースやカンマで区切るような形式に変換して保存することができる。また、C 言語で同様の形式でデータをファイルに書き込めば、表計算ソフトで読み込むことができる。これらを使い、表計算ソフトにあるデータをプログラミング言語で複雑な計算をし、それを表計算ソフトに改めて読み込ませてグラフ化をする、というようなことができる。

2.6　構造体

構造体 (structure) は、異なる型をもつデータをひとまとめにしたものである。配列は、同じ型のデータを集めたものなのに対して、構造体は異なる型のデータを集めたものである。構造体というのは、C 言語特有の呼び方で、一般の用語ではレコード構造と呼ばれる [1]。

3　設計

本実験では、品川高専入試プログラムを作成することを最終の目標とする。そのプログラムの仕様、設計を以下に示す。

3.1　仕様

本実験で作成するプログラムの仕様を以下に示す。

1. 受験時の国語の点数、数学の点数、英語の点数 (100 点満点)
2. 中学校 3 年次 2 学期の国語、社会、数学、理科、音楽、美術、保健・体育、技術・家庭、英語の成績

これらのデータ (ファイルに入っている) を使い、成績処理及び順位付けをするプログラムを作成する。作成には次のような行程を踏んで行う。

1. 成績が入ってるファイルから読み込を行い、そのまま処理結果ファイルに保存する。
2. それぞれの教科の点数を相当する変数に入れる。ただし、配列の添字と受験番号及び、教科に対応させる。
3. 単純に受験時の点の合計をを素点にいれ、中学校の成績の点をそれぞれ総合して、記録点として代入する。
4. 数学と理科の成績を 1.4 倍して、それらを使った中学校の成績の点を使って、中学校の成績の合計点の最高点の最高点 (7 点 $\times$ 2 教科 $+$ 5 教科 $\times$ 5 教科 $\bar{4}9$ 点) を 700 点に換算 (すなわち中学校の成績の合計点の最高点) した値を記録換算点に入れる。
5. 全ての総合点の平均をとり、平均点以上の受験生に対して合否に 1 を平均未満の受験生に対しては、0 を入力する。
6. 総合点を高い順番でソート (並べ替え) をファイル「sort.dat」に保存する。ただし、誰がどの点数だったかを記録しておく必要はない。
7. 総合点を使って、受験生の順位を決定し、順位に代入する。ただし、同点の場合は受験番号が若い (小さい) ものを上位とする。
8. 処理結果ファイルを使い、受験番号を入力したらその受験生の成績を表示するプログラムを作成する。

受検番号	国語点数	数学点数	英語点数	国語成績	社会成績	…	素点	記録点	記録換算点	総合点	合否	順位
1	60	70	80	5	2		210	30	428	638	1	10

図 3.1　成績の一例

3.2 入試プログラムの設計

品川高専入試プログラムを作成する時に、ファイルからの読み込み、内部処理の設計、ソート、ファイルへの書き出し、表計算ソフトでの読み込みファイルの作成、表計算ソフトでの書き出しファイルの作成、それらの段階ごとの設計について以下に示す。

3.2.1 ファイルからの書き込み

入試プログラムを作成するにあたり、入試テストの元のデータを読み取る必要がある、その機構の設計を以下に示す。ファイルからの書き込みに用いるファイルとして、可用性の高い、csv ファイルからデータを読み取るプログラムを設計することとした。入試プログラムに用いるファイルの読み込みは、c 言語の標準的な書き込みを用いる。まず fp というファイルポインタを定義し、前もって、空の配列を宣言しておく。fp という FILE ポインタから一時的に、csv の行の要素を配列の各要素として取り込み、構造体配列の各メンバに対して、それらの配列を代入することで、ファイルの読み取り処理を行う。

3.2.2 ソート

入試プログラムによって計算を行ったあと、得られた数値を高い順に並び替える必要がある、その機構の設計を以下に示す。テストの得点を降順にソートする。仕様にある通り、順位が同点の場合、出席番号が若いものを優先する必要がある。そのため、安定性のあるソートを使う必要があるもしくは、若い番号を優先するように作り替える必要があるため、今回は前者の安定性のあるソートによって実装することで、仕様に従った。安定性のあるソートは、実装も簡単に行えるバブルソートを採用する。

3.2.3 ファイルへの書き出し

入試プログラムを作成するにあたり、読み込みこんだファイルを計算したあと、ファイルを書き出す必要がある、その機構の設計を以下に示す。ファイルからの書き出しは、ファイルからの読み込みと同等にあらたなファイルポインタ fp_out を作成し、csv ファイルの形式で出力を行う。fprintf を用いて、一行目は各列のラベルを出力する。また、2 行目以降は、構造体配列の要素を for 文をもちいて、構造体配列の各要素を出力する。各構造体配列ごとに順位を割り振っていき順位も出力する。

3.2.4 読み込みファイルの表計算ソフトでの作成

入試プログラムを作成するにあたり、誰でもこのプログラムを使用できるように、表計算ソフトから読み込みようのファイルを作成することができるようにする必要がある、その設計を以下に示す。表計算ソフトの保存方法を csv 形式で保存させることで、プログラムからの読み込みを可能にする。

3.2.5 書き出しファイルの表計算ソフトでの読み取り

入試プログラムを作成するにあたり、誰でもこのプログラムを使用できるように、表計算ソフトによって、プログラムで出力されたファイルを閲覧できるようにする必要がある、その設計を以下に示す。プログラムによって出力されるファイルを csv ファイルとすることによって、閲覧を可能にする。

3.2.6　処理結果のグラフ化

表計算ソフトで出力されたデータのみでは、そのテスト事態の傾向や、その年の点数の分布を視覚的にわかりやすいとは言えない。そのため、表計算ソフトを用いて処理結果をグラフ化する。その設計を以下に示す。総合点の最大値は 1000 点であるため、階級値を 100 刻みの数値として、度数分布表を作成する。

4 実装

4.1 実装環境

実装に用いた環境の情報を 4.1 に示す。

表 4.1 実験環境

ツール	ツール名	バージョン
プログラミング言語	c	Apple clang version 12.0.5 (clang-1205.0.22.9)
OS	Mac OS Big Sur	11.2.3
CPU	Intel Core i5	2 GHz クアッドコア
メモリ	LPDDR4X	16 GB 3733 MHz
表計算ソフト	Micro Soft Excel	16.48

4.2 実装したプログラム

設計を元にして本実験で最終的に実装した入試プログラムを 4.1 に示す。

コード 4.1 成績処理のプログラム

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define NUM 13
   struct score {
     int number; // 受験番号
     int japanese_score; // 国語のテスト点
     int math_score; // 数学のテスト点
     int english_score; // 英語のテスト点

     int japanese_grade; // 国語の成績
     int ss_grade; // 社会の成績 (social studies)
     int math_grade; // 数学の成績
     int science_grade; // 理科の成績
     int music_grade; // 音楽の成績
     int art_grade; // 美術の成績
     int pe_grade; // 体育の成績
     int helt_grade; // 技術家庭 (home economics life technique)
     int english_grade; // 英語の成績

     int total_score; // テスト点数の合計
     int total_grade; // 内申点の合計
     float base_score; // 記録換算点内申点を点数化したもの()

     float score; // 全てを考慮した点数

     int result; // 合否
     int rank; // 順位
   };
```

```
30
31  void score_sort(struct score scores[],int ninzu){
32     int s,t;
33     struct score tmp;
34      for(s=0 ; s<ninzu-1 ; s++){
35          for(t=s+1;t<ninzu;t++){
36              if(scores[t].score>scores[s].score){
37                  tmp = scores[t];
38                  scores[t] = scores[s];
39                  scores[s] = tmp;
40                  // bubble sort sort の安定性が高いため採用
41              }
42          }
43      }
44  }
45
46  int avg_score(struct score scores[],int ninzu){
47    int i;
48    float avg;
49    for(i=0;i<ninzu;i++){
50      avg += scores[i].score;
51    }
52    avg = avg/i;
53    return avg;
54  }
55
56  int main(){
57    FILE *fp;
58    FILE *fp_out;
59
60    char *fname = "input.csv";
61    int ret , i=0 ,avg;
62    char buf[NUM][100];
63    int data[NUM];
64    char fname_out[24];
65
66    // ファイルを開く
67    fp = fopen( fname, "r" );
68    if( fp == NULL ){
69      printf( "%ファイルが開けませんs\n", fname );
70      return -1;
71    }
72
73
74    fscanf(fp,
          "%[^,],%[^,],%[^,],%[^,],%[^,],%[^,],%[^,],%[^,],%[^,],%[^,],%[^,],%[^,],%s",
75    buf[0], buf[1], buf[2], buf[3],buf[4],buf[5],buf[6],buf[7],buf[8],buf[9],buf[10],buf
          [11],buf[12]);
76
77    struct score scores[500];
78
79    while((ret=fscanf(fp, "%d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d ,%d",
80    &data[0], &data[1], &data[2], &data[3], &data[4], &data[5], &data[6],
81    &data[7], &data[8], &data[9], &data[10], &data[11], &data[12])) != EOF){
82      scores[i].number =data[0];
83      scores[i].japanese_score =data[1];
84      scores[i].math_score = data[2];
```

```
85        scores[i].english_score =data[3];
86
87        scores[i].japanese_grade =data[4];
88        scores[i].ss_grade =data[5];
89        scores[i].math_grade =data[6];
90        scores[i].science_grade =data[7];
91        scores[i].music_grade =data[8];
92        scores[i].art_grade =data[9];
93        scores[i].pe_grade =data[10];
94        scores[i].helt_grade =data[11];
95        scores[i].english_grade =data[12];
96
97        scores[i].total_score = data[1]+data[2]+data[3];
98        scores[i].total_grade =data[4]+data[5]+data[6]+data[7]+data[8]+data[9]+data[10]+
              data[11]+data[12];
99        scores[i].base_score =700*((data[4]+data[5]+(data[6]*1.4)+data[7]+data[8]+data[9]+
              data[10]+data[11]+(data[12]*1.4))/49);
100
101       scores[i].score = scores[i].total_score + scores[i].base_score;
102       ++i;
103     }
104
105     int ninzu = i;
106
107     avg = avg_score(scores,ninzu);
108     score_sort(scores,ninzu);
109
110     // ファイル出力
111     int year = 2021 ,month = 4;
112     sprintf( fname_out, "%04d%02dresult.csv", year, month);
113
114     // ファイルのオープン
115     if( (fp_out = fopen( fname_out, "w" )) == NULL ) {
116         printf( 結果ファイルがオープンできませんでした"\n");
117         exit( 1 );
118     }
119
120     // ヘッダ行を出力する
121     fprintf( fp_out,
122     "number , japanese_score , math_score,English_score, japanese_grade,ss_grade,
            math_grade, science_grade, music_grade , art_grade , pe_grade , helt_grade,
            english_grade , total_score , total_grade , base_score, score , result,rank\n"
123     );
124
125     // 各データをカンマ区切りで出力する
126      for( i = 0; i < ninzu; i++ ) {
127        if(scores[i].score>=avg){
128            scores[i].result = 1;
129        }else{
130            scores[i].result = 0;
131        }
132           scores[i].rank = i+1;
133
134          fprintf( fp_out, "%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%4.0f,%4.0f,%d,%
                d\n",
135            scores[i].number,
136            scores[i].japanese_score,
```

```
137             scores[i].math_score,
138             scores[i].english_score,
139             scores[i].japanese_grade,
140             scores[i].ss_grade,
141             scores[i].math_grade,
142             scores[i].science_grade,
143             scores[i].music_grade,
144             scores[i].art_grade,
145             scores[i].pe_grade,
146             scores[i].helt_grade,
147             scores[i].english_grade,
148             scores[i].total_score,
149             scores[i].total_grade,
150             scores[i].base_score,
151             scores[i].score,
152             scores[i].result,
153             scores[i].rank
154             );
155      }
156
157     printf( "%s に結果が出力されました\n", fname_out );
158
159     printf("%d",avg_score(scores,ninzu));
160     fclose( fp_out );
161   }
```

5　結果

実装したプログラムの実行結果を以下に示す。成績の計算が行われる前の表計算ソフトを使用して作成した csv ファイルを 5.1 に示す。

number	japanese_sco	math_score	English_score	japanese_gra	ss_grade	math_grade	science_grad	music_grade	art_grade	pe_grade	helt_grade	english_grade
1000	74	7	84	2	5	5	1	4	3	2	4	3
1001	35	35	61	1	2	2	3	4	4	1	1	5
1002	26	93	88	4	4	2	2	4	3	5	4	4
1003	63	97	56	5	3	1	2	1	4	3	1	4
1004	62	8	29	3	1	4	3	4	4	5	4	1
1005	82	29	32	2	2	2	5	4	2	1	3	2
1006	7	21	15	2	1	1	5	5	3	5	1	4
1007	64	47	58	2	3	2	3	1	2	5	2	1
1008	49	36	87	1	5	4	1	2	2	4	3	2
1009	21	71	99	4	4	1	1	2	2	3	1	4
1010	54	29	71	1	5	3	3	4	5	4	1	3
1011	34	56	55	3	5	4	2	4	4	1	2	3
1012	31	58	94	3	1	3	2	1	2	3	4	4
1013	11	60	87	1	3	2	5	4	1	4	1	2
1014	10	82	12	1	3	4	2	2	2	1	1	5
1015	98	44	92	5	4	4	2	3	1	4	2	3
1016	15	38	86	5	4	2	1	3	2	1	4	2
1017	49	57	95	1	1	1	5	1	5	5	4	4
1018	4	35	31	3	4	1	2	2	4	4	2	4
1019	51	48	7	5	1	5	5	4	5	1	4	4
1020	47	19	56	1	3	2	4	3	2	5	2	5
1021	47	34	70	1	5	4	1	2	3	2	2	1
1022	85	65	51	5	2	4	1	2	3	3	5	5
1023	19	10	46	3	1	2	2	4	3	1	1	3
1024	86	39	54	1	3	2	5	1	3	2	5	4
1025	65	5	22	1	1	3	1	4	4	1	1	2
1026	92	53	88	3	5	4	5	1	5	4	5	1
1027	21	95	80	3	1	1	2	1	1	2	5	3
1028	43	63	95	2	2	3	2	3	5	2	5	2
1029	54	86	14	2	2	5	5	3	4	3	4	5
1030	79	46	9	3	5	4	4	5	1	3	3	2
1031	3	81	25	3	4	3	2	4	5	2	5	3
1032	47	41	63	4	4	4	5	4	4	5	5	4
1033	17	56	83	1	2	5	2	2	5	4	5	4
1034	95	62	49	1	3	1	2	3	5	5	1	1

図 5.1　プログラムが実行される前の csv ファイル

図 5.1 の csv ファイルをプログラムを実行し、計算の処理を行った。処理された後の csv ファイルを図 5.2 に示す。

number	japanese_sc	math_score	English_score	japanese_gr	ss_grade	math_grade	science_grad	music_grade	art_grade	pe_grade	helt_grade	english_grad	total_score	total_grade	base_score	score	result	rank
1032	47	41	63	4	4	4	5	4	4	5	5	4	151	39	603	754	1	1
1026	92	53	88	3	5	4	5	1	5	4	5	1	233	33	500	733	1	2
1002	26	93	88	4	4	2	2	4	3	5	4	4	207	32	491	698	1	3
1029	54	86	14	2	2	5	5	3	4	3	4	5	154	33	529	683	1	4
1022	85	65	51	5	2	4	1	2	3	3	5	5	201	30	480	681	1	5
1015	98	44	92	5	4	4	2	3	1	4	2	3	234	28	440	674	1	6
1019	51	48	7	5	1	5	5	4	5	1	4	4	106	34	537	643	1	7
1033	17	56	83	1	2	5	2	2	5	4	5	4	156	30	480	636	1	8
1000	74	7	84	2	5	5	1	4	3	2	4	3	165	29	460	625	1	9
1017	49	57	95	1	1	1	5	1	5	5	4	4	201	27	414	615	1	10
1010	54	29	71	1	5	3	3	4	5	4	1	3	154	29	449	603	1	11
1028	43	63	95	2	2	3	2	3	5	2	5	2	201	26	400	601	1	12
1030	79	46	9	3	5	4	4	5	1	3	3	2	134	30	463	597	1	13
1003	63	97	56	5	3	1	2	1	4	3	1	4	216	24	371	587	1	14
1031	3	81	25	3	4	3	2	4	5	2	5	3	109	31	477	586	1	15
1011	34	56	55	3	5	4	2	4	4	1	2	3	145	28	440	585	1	16
1024	86	39	54	1	3	2	5	1	3	2	5	4	179	26	406	585	1	17
1012	31	58	94	3	1	3	2	1	2	3	4	4	183	23	369	552	0	18
1008	49	36	87	1	5	4	1	2	2	4	3	2	172	24	377	549	0	19
1020	47	19	56	1	3	2	4	3	2	5	2	5	122	27	426	548	0	20
1004	62	8	29	3	1	4	3	4	4	5	4	1	99	29	443	542	0	21
1009	21	71	99	4	4	1	1	2	2	3	1	4	191	22	343	534	0	22
1034	95	62	49	1	3	1	2	3	5	5	1	1	206	22	326	532	0	23
1013	11	60	87	1	3	2	5	4	1	4	1	2	158	23	351	509	0	24
1016	15	38	86	5	4	2	1	3	2	1	4	2	139	24	366	505	0	25
1001	35	35	61	1	2	2	3	4	4	1	1	5	131	23	369	500	0	26
1005	82	29	32	2	2	2	5	4	2	1	3	2	143	23	351	494	0	27
1027	21	95	80	3	1	1	2	1	1	2	5	3	196	19	294	490	0	28
1007	64	47	58	2	3	2	3	1	2	5	2	1	169	21	317	486	0	29
1021	47	34	70	1	5	4	1	2	3	2	2	1	151	21	329	480	0	30
1018	4	35	31	3	4	1	2	2	4	4	2	4	70	26	400	470	0	31
1006	7	21	15	2	1	1	5	5	3	5	1	4	43	27	414	457	0	32
1014	10	82	12	1	3	4	2	2	2	1	1	5	104	21	351	455	0	33
1023	19	10	46	3	1	2	2	4	3	1	1	3	75	20	314	389	0	34
1025	65	5	22	1	1	3	1	4	4	1	1	2	92	18	286	378	0	35

図 5.2　プログラムが実行された後の csv ファイル

図 5.2 から、各行で素点、記録点、換算記録点、総合点が出力されていることがわかる。総合点 (score) の平均点を求めた。平均点 564.45 点より高い総合点を取っている行の結果 (result) は 1 になっていること、総合点が低い行の結果が 0 になっていることが確認できた。また、その総合点の上位から順位が振られていることがわかった。

また、図 5.2 を元にして度数分布表を作成した。その結果を図 5.3 に示す。

以上	未満	階級値	度数	相対度数	累積相対度数
50	100	75	0	0	0
100	150	125	0	0	0
150	200	175	0	0	0
200	250	225	0	0	0
250	300	275	0	0	0
300	350	325	0	0	0
350	400	375	0	0	0
400	450	425	2	0.05714286	0.05714286
450	500	475	0	0	0.05714286
500	550	525	8	0.22857143	0.28571429
550	600	575	7	0.2	0.48571429
600	650	625	6	0.17142857	0.65714286
650	700	675	6	0.17142857	0.82857143
700	750	725	4	0.11428571	0.94285714
750	800	775	1	0.02857143	0.97142857
800	850	825	1	0.02857143	1
850	900	875	0	0	1
900	950	925	0	0	1
950	1000	975	0	0	1
			0		

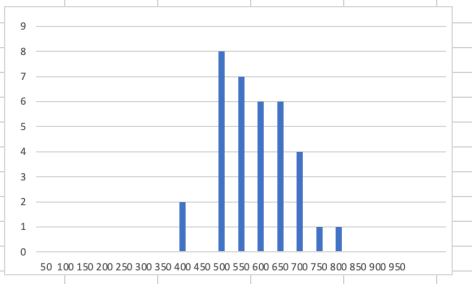

図 5.3　図 5.2 を元にして作成した度数分布表とヒストグラム

6　考察

6.1　構造体の採用について

本プログラムでは、各生徒の点数を扱うときに、配列ではなく構造体を用いて計算を行った。構造体を用いるメリットとして、扱っているデータに名前をつけることができるため、プログラムの可読性が向上する、直感的にデータを操作できることなどの点が挙げられた。デメリットとして、構造体のデータを他の関数などに渡すときに、構造体をひとまとめで渡す必要があることなどが挙げられた。上記のメリット、デメリットを加味し、プログラム上で扱う数値が、テストの点数 3 つ、成績の評定が 9 つ存在するため、点数や、評定をみたときに、なんの科目の数値を扱っているのかがわかりやすくするために、今回実装したプログラムでは、配列ではなく構造体を利用した。また、上記のことから本実験で実装する際には構造体を用いた方が実装が容易であることが考えられる。

6.2　バブルソートの採用について

設計においても示した通り、ソートを実装する際に、バブルソートを用いて、実装を行った。バブルソートはソートの安定性があるソートなので、総合点が同じ行が二つ以上あった場合、点数が若い方が優先されるようになっている。図 5.2 の 585 点の生徒をみてわかるように、番号が若い方が順位が高くなっていることがわかった。また、バブルソートの計算量は $\mathrm{O}(N^2)$ であるため、N の数がさらに増えたときに実行の速度が遅くなることが考えられる。しかし、実装に対するコストが低くすむこと、高専の入試プログラムとして扱う場合はテストの受験者が多くとも 500 人前後であること、を考えると、さほど差はないように考えられる。もし、このプログラムを統一テストであったり、センター試験のテストであったり、より複雑な計算をしたりする場合がある時は、マージソートやより早いソートアルゴリズムを採用した方がより良いプログラムになると考えられる。

6.2.1　度数分布表について

表計算ソフトを用いて、度数分布表を作成した。中央の値に対して山なりなグラフになっていることから、このグラフは正規分布であると言える。その理由として、テストの点数 3 つ、成績 9 つの数値を乱数を用いて生成しているため、であると考えられる。

7　結論

これまで習ってきた C 言語の知識を利用して、比較的大きなプログラムを書くことで、プログラミング言語の利用方法のまとめを行うと同時に、複雑なアルゴリズムの構築方法を学ぶことができた。

参考文献

[1] 近藤嘉雪.2018. 定本 C プログラマのためのアルゴリズムとデータ構造.SB クリエイティブ株式会社.