**Report of C++ Experimental Programming**

Author: SHI Kunkun

Student ID: 3117007052

Xi’an Jiaotong University

2017.10

**Experimental objective:**

In the very first phase of C++ learning, we’ve got familiar with basic concepts and grammars of it. We have also studied the definitions of class and object, which are the essence of OOP C++ language. Besides, through the courses, we have compared the differences between procedure-oriented language (C language for example) and object-oriented language (C++ language especially) and have realized the advantages of OOP language. By means of hands on experiments, we can master the operations with computer programming and solve the questions encountered during this process, which will finally improve a lot our programming skills.

**Experimental contents:**

We have two small subjects to finish in this experiment. And the description of questions is given below in Chinese.

* 某次歌手比赛聘请了7名评委打分，每个人打的分数在1到100之间，7个评委必须对每个选手打分。对一个选手的得分做以下计算：去掉一个最高分，去掉一个最低分，然后对剩下的得分进行平均得到最后的选手成绩。要求编写程序模拟30名参赛选手的比赛情况，将他们的得分按从高到低的顺序显示输出，要求显示精度为两位小数。
* 建立一个包括30名学生的学生信息表，每个学生信息中包括：姓名、学号、专业、班级。编写一个程序，可以在以上的信息表中按名字或学号查询指定的学生信息，如果找到，则显示该学生的信息，如果找不到，则提示出错。

**Algorithm description:**

Following the fundamental thoughts of OOP, we should firstly extract the common points of our objects (undefined yet). We found that it would be appropriate to create a class of singers for the first subject and a class of students for the second.

In the first one, the member variables of object should contain his/her scores from seven judges. And the class should also include some methods to helps us get one object’s final score. Besides, in this case, a sort function is essential to order the singers by their grades. As for the display properties, it is very easy to format them using standard library functions. We can also overload some useful operators to simplify our operations with objects.

The main idea is the same for the second question, the 30 students’ basic information (including their name, student ID, major and class) is stored in a .txt file so that we could easily initialize our class array to avoid long initialization section in my codes. FILE I/O basic knowledge is required if we want to do it this way.

In a word, we need to use a series of skills to reduce our work in main function, which means the implementation should be well done so that the programmer can concentrate on top-level design.

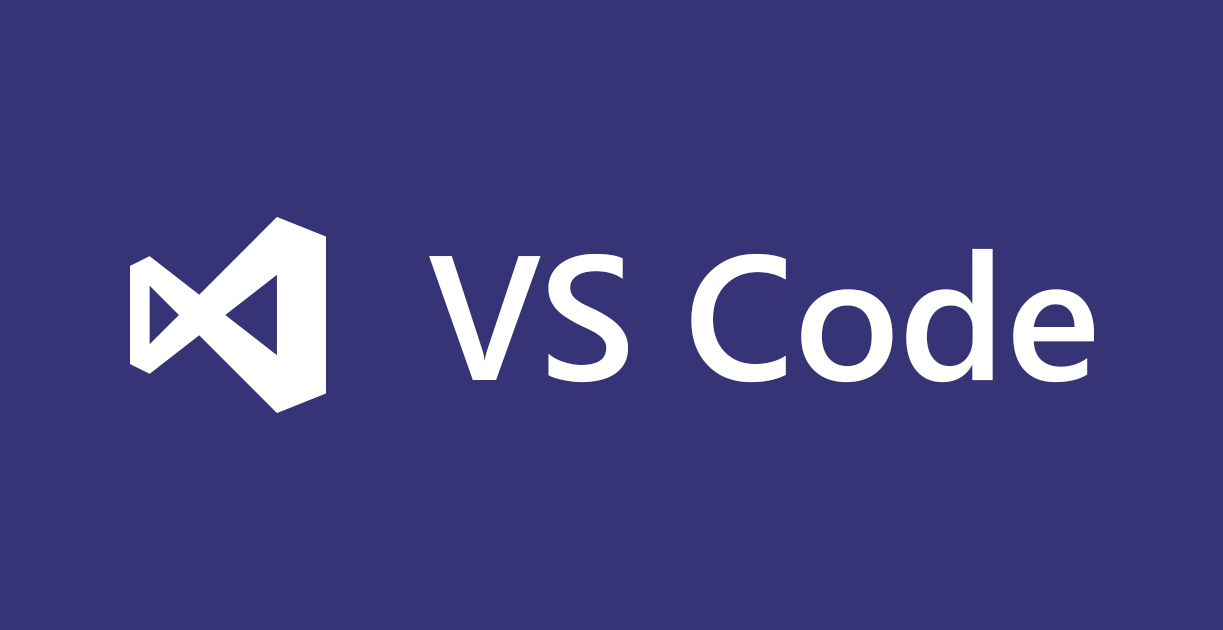
**Programming environment:**

Editor: Visual Studio Code(Microsoft)

Compiler: g++(GNU) conformed to ANSI C++11

Debug tool: GDB

Operation System: Latest Windows 10.

**Source code:**

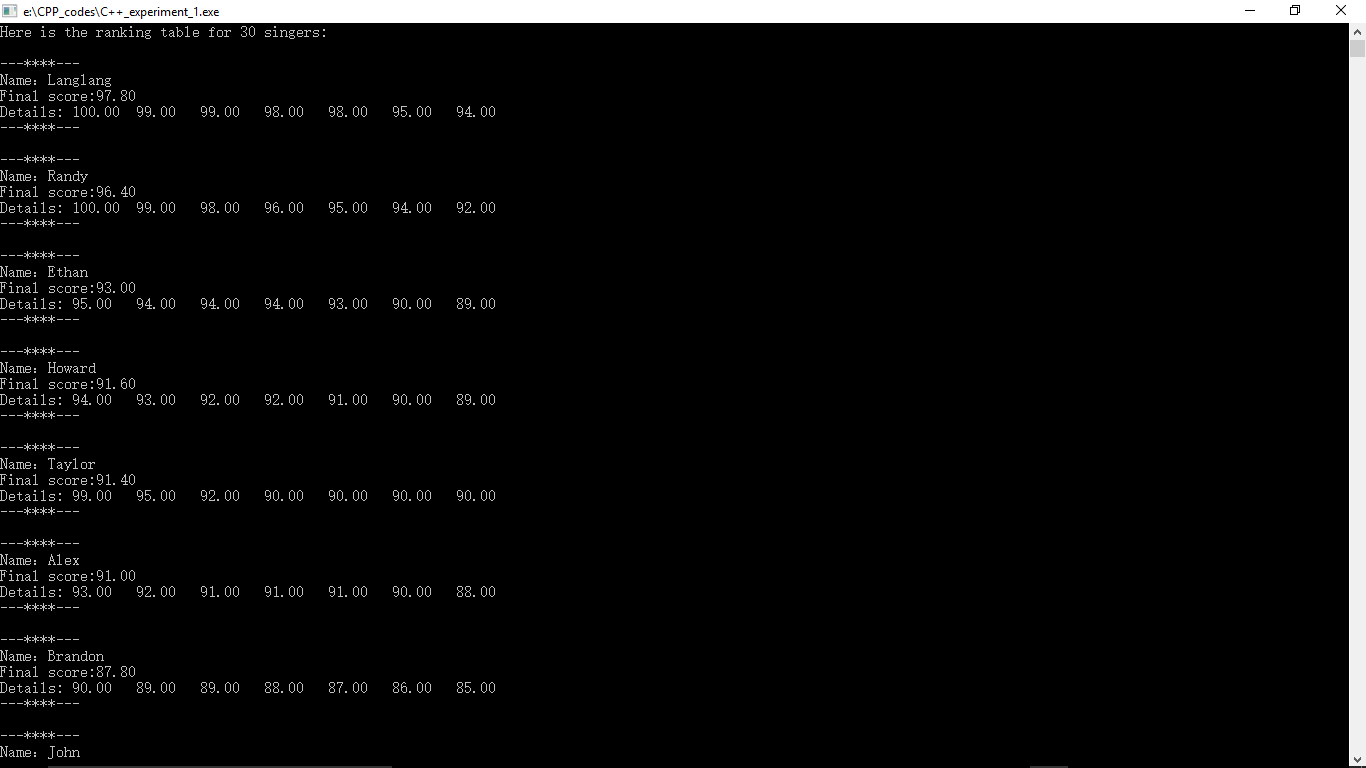
Exercise1:

Instream file ***singerslist.txt***

1. Jack    78 85 74 56 87 83 69
2. Bob     56 65 54 67 53 58 60
3. Tom     75 75 80 79 78 74 77
4. Lee     79 81 80 81 81 83 82
5. Hugo    58 60 61 60 59 63 62
6. Philip  65 64 67 68 59 60 60
7. Ethan   95 94 93 94 90 89 94
8. Jett    89 80 87 88 86 85 88
9. Randy   99 100 98 95 94 92 96
10. Juan    78 79 77 78 78 69 82
11. Bobby   82 81 83 81 81 82 81
12. Vincent 71 71 73 75 72 69 68
13. Jonny   87 89 85 86 86 79 81
14. Dylan   84 83 83 82 85 84 81
15. Craig   79 80 80 79 78 76 85
16. Jasper  74 69 68 72 72 71 75
17. Felix   65 64 63 65 68 69 70
18. Alex    92 91 90 91 93 91 88
19. Sam     89 85 87 86 89 89 85
20. Fred    71 76 70 71 67 72 71
21. Brandon 87 90 86 88 89 89 85
22. O'Neal  84 79 80 83 83 84 85
23. Taylor  90 90 90 99 95 90 92
24. Dirk    61 65 62 63 56 60 60
25. Mcgrady 73 72 76 79 78 70 73
26. John    89 88 86 85 90 87 88
27. Howard  92 93 91 90 89 94 92
28. Jenson  78 69 75 74 76 77 77
29. Kunkun  64 70 67 69 59 65 70
30. Langlang    100 99 98 95 94 99 98

***.cpp*** file:

1. #include <iostream>
2. #include <fstream>
3. #include <cstdio>
4. #include <cstdlib>
5. #include <cstring>
7. #define NUM\_JUDGES 7
8. **using** **namespace** std;
10. **class** Singer
11. {
12. **public**:
13. /\*default constructor with no parameter\*/
14. Singer();
16. /\*Copy constructor\*/
17. Singer(**const** Singer& singer\_object);
19. /\*get the final grade of a singer\*/
20. **double** get\_average();
22. /\*display an object of class Singer\*/
23. **void** display();
25. /\*overloading assignment operator = \*/
26. **void** operator = (Singer& right\_side);
28. /\*friend function to overload << and >> operator\*/
29. **friend** istream& operator >> (istream& ins, Singer& right\_side);
30. **friend** ostream& operator << (ostream& outs, **const** Singer& right\_side);
32. **private**:
33. **char** name[20];
34. **double** scores[NUM\_JUDGES];
35. };
37. /\* sort function for simple double type array and class array\*/
38. **void** sort(**double** \* array, **int** size);
39. **void** sort(Singer \* array, **int** size);
41. **int** main(**void**)
42. {
43. Singer singers[30];
44. ifstream ins;
46. ins.open("singerslist.txt");
47. **if**(ins.fail())
48. {
49. cout<<"Error when open singers list file!\n";
50. exit(1);
51. }
53. /\*Initialization section\*/
54. **for**(**int** i = 0; i < 30; i++)
55. {
56. ins >> singers[i];
57. }
59. ins.close();
61. sort(singers, 30);
63. cout.setf(ios::fixed);
64. cout.setf(ios::showpoint);
65. cout.precision(2);
67. cout << "Here is the ranking table for 30 singers:\n " << endl;
68. **for**(**int** i = 0; i <30; i++)
69. singers[i].display();
71. getchar();
72. **return** 0;
73. }
75. Singer::Singer()
76. {
77. /\* initialize private member variable \*/
78. name[0] = '\0';
79. **for**(**int** i = 0; i < NUM\_JUDGES; i++)
80. scores[i] = 0;
81. }
83. Singer::Singer(**const** Singer& singer\_object)
84. {
85. strcpy(name, singer\_object.name);
86. **for**(**int** i = 0; i < NUM\_JUDGES; i++)
87. scores[i] = singer\_object.scores[i];
89. }
91. **double** Singer::get\_average()
92. {
93. **double** sum = 0;
94. **double** result;
95. sort(scores,NUM\_JUDGES);
96. **for**(**int** i = 1; i < (NUM\_JUDGES - 1); i++)
97. sum += scores[i];
98. result = sum / (NUM\_JUDGES - 2);
100. **return** result;
101. }
103. istream& operator >> (istream& ins, Singer& right\_side)
104. {
105. ins >> right\_side.name;
106. **for**(**int** i = 0; i < NUM\_JUDGES; i++)
107. ins >> right\_side.scores[i];
109. **return** ins;
110. }
112. ostream& operator << (ostream& outs, **const** Singer& right\_side)
113. {
114. outs << right\_side.name;
115. **for**(**int** i = 0; i < NUM\_JUDGES; i++)
116. outs << right\_side.scores[i];
118. **return** outs;
119. }
121. **void** Singer::display()
122. {
123. cout << "---\*\*\*\*---" << endl;
124. cout << "Name：" << name << endl;
125. cout << "Final score:" << get\_average() << endl;
126. cout << "Details: ";
127. **for**(**int** i = 0 ; i < NUM\_JUDGES; i++) // debug
128. printf("%-8.2lf",scores[i]);
129. cout << "\n---\*\*\*\*---"<< endl;
130. printf("\n");
131. }
133. **void** Singer::operator = (Singer& right\_side)
134. {
135. **double** \* pt = right\_side.scores;
136. strcpy(name,right\_side.name);
137. **for**(**int** i = 0; i < NUM\_JUDGES; i++)
138. {
139. scores[i] = pt[i];
140. }
142. }
144. **void** sort(**double** \* array, **int** size)
145. {
146. **double** temp;
147. **for**(**int** i = 0; i < size; i++)
148. {
149. **for** (**int** j = i; j < size; j++)
150. {
151. **if**(array[j] > array[i])
152. {
153. temp = array[j];
154. array[j] = array[i];
155. array[i] = temp;
156. }
157. }
158. }
159. }
161. **void** sort(Singer \* singers, **int** size)
162. {
163. Singer temp;
164. **for**(**int** i = 0; i < size; i++)
165. {
166. **for** (**int** j = i; j < size; j++)
167. {
168. **if**(singers[j].get\_average() > singers[i].get\_average())
169. {
170. temp = singers[j];
171. singers[j] = singers[i];
172. singers[i] = temp;
173. }
174. }
175. }
176. }

Results:

The name list was rearranged by singers’ scores. To obtain final scores of singers, I eliminated the max and the min grades in object’s scores[ ] array, which is mentioned in description of question. And then the final score is the average of remained grades.

Exercise 2:

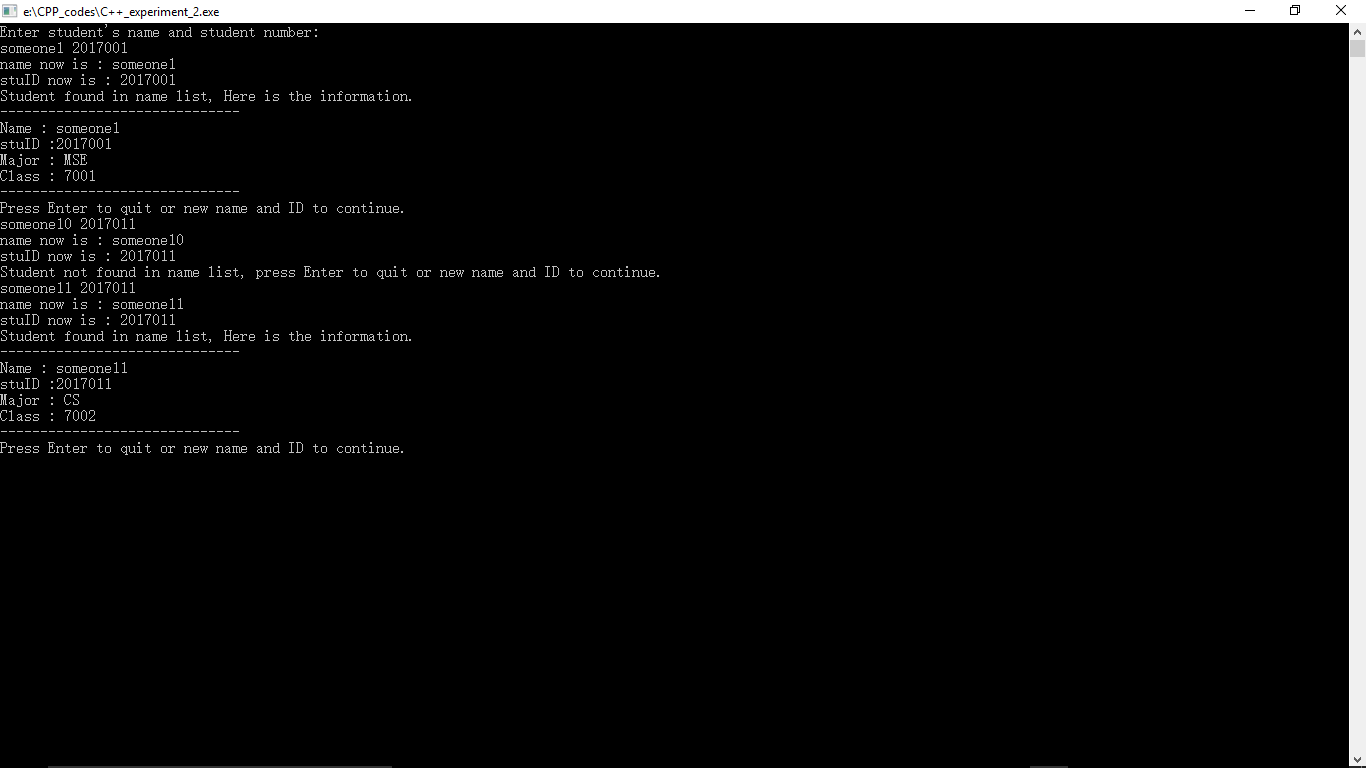
Instream file ***stulist.txt***

1. someone1    2017001 MSE 7001
2. someone2    2017002 MSE 7001
3. someone3    2017003 MSE 7001
4. someone4    2017004 MSE 7001
5. someone5    2017005 MSE 7001
6. someone6    2017006 MSE 7001
7. someone7    2017007 MSE 7001
8. someone8    2017008 MSE 7001
9. someone9    2017009 MSE 7001
10. someone10    2017010 MSE 7001
11. someone11    2017011 CS  7002
12. someone12    2017012 CS  7002
13. someone13    2017013 CS  7002
14. someone14    2017014 CS  7002
15. someone15    2017015 CS  7002
16. someone16    2017016 CS  7002
17. someone17    2017017 CS  7002
18. someone18    2017018 CS  7002
19. someone19    2017019 CS  7002
20. someone20    2017020 CS  7002
21. someone21    2017021 SE  7003
22. someone22    2017022 SE  7003
23. someone23    2017023 SE  7003
24. someone24    2017024 SE  7003
25. someone25    2017025 SE  7003
26. someone26    2017026 SE  7003
27. someone27    2017027 SE  7003
28. someone28    2017028 SE  7003
29. someone29    2017029 SE  7003
30. someone30    2017030 SE  7003

And here is the main implementation file. To improve user-friendliness of application interface, we chose to have a main loop in our code so that the user can search student all the time unless he wants to quit by pressing [Enter] on keyboard.

1. #include <iostream>
2. #include <fstream>
3. #include <cstdio>
4. #include <cstdlib>
5. #include <cstring>
6. **using** **namespace** std;
8. **class** Student\_info
9. {
10. **public**:
11. /\*default constructor\*/
12. Student\_info();
14. /\*destructor\*/
15. ~Student\_info();
17. /\*method to get the dynamic string's pointer\*/
18. **char** \* get\_name() **const**;
20. /\*method to access student number\*/
21. **long** get\_stuID() **const**;
23. /\*formatted output for object\*/
24. **void** display() **const**;
26. /\*overload extract operator\*/
27. **friend** istream& operator >> (istream& ins, Student\_info& right\_side);
29. /\*overload insertion operator\*/
30. **friend** ostream& operator << (ostream& outs, **const** Student\_info& right\_side);
32. **private**:
33. **char** \*name, \*major;
34. **long** stuID;
35. **int** class\_num;
36. **int** max\_size; // the max size for dynamic string
37. };
39. /\*auxiliary function\*/
40. **void** new\_line();
42. **int** main(**void**)
43. {
44. Student\_info students[30]; // declare a class array which contains 30 elements
45. **char** name[20];
46. **long** ID;
47. **bool** isfound;
48. **int** index;
49. **char** ch = 'a'; // declare a char and initialize it
51. ifstream ins;
53. ins.open("stulist.txt");
54. **if**(ins.fail())
55. {
56. cout<<"Error when open student list file!\n";
57. exit(1);
58. }
60. /\* read the list and store students' information in class array\*/
61. **for**(**int** i = 0; i < 30; i++)
62. {
63. ins >> students[i];
64. }
66. /\*debug: check if read operation is done properly\*/
67. // for(int i = 0; i < 30; i++)
68. // {
69. //     cout << students[i] << endl;
70. // }
72. /\*serach the student in name list\*/
73. cout << "Enter student's name and student number:\n";
74. **while**( ch != '\n')
75. {
76. isfound = **false**;
77. cin >> name >> ID;
79. printf("name now is : %s\n",name); //debug
80. printf("stuID now is : %ld\n",ID);
82. new\_line(); // discard extra characters
83. **for**(**int** i = 0 ; i < 30; i++)
84. {
85. **if**(!strcmp(name, students[i].get\_name()) && (ID == students[i].get\_stuID())) // strcmp() return zero if name is the same as get\_name()
86. {
87. isfound = **true**;
88. index = i;
89. **break**;
90. }
91. }
93. **if**(isfound)
94. {
95. cout << "Student found in name list, Here is the information.\n";
96. students[index].display();
97. cout << "Press Enter to quit or new name and ID to continue.\n";
98. }
100. **else**
101. {
102. cout << "Student not found in name list, press Enter to quit or new name and ID to continue.\n";
103. }
105. cin.get(ch);
106. cin.putback(ch); // Put back the char got into stream
107. }
109. cout << "Byebye!" <<endl;
110. ins.close();
112. getchar();
113. **return** 0;
114. }
116. Student\_info::Student\_info(): stuID(0),class\_num(0),max\_size(30)
117. {
118. name = **new** **char**[max\_size];
119. name[0] = '\0';
120. major = **new** **char**[max\_size];
121. major[0] = '\0';
122. }
124. Student\_info::~Student\_info()
125. {
126. **delete** [] name;
127. **delete** [] major;
128. }
130. **char** \* Student\_info::get\_name() **const**
131. {
132. **return** name;
133. }
135. **long** Student\_info::get\_stuID() **const**
136. {
137. **return** stuID;
138. }
140. **void** Student\_info::display() **const**
141. {
142. **char** temp1[max\_size],temp2[max\_size];
143. strcpy(temp1,name);
144. strcpy(temp2,major);
145. cout << "------------------------------\n";
146. cout << "Name : " << temp1 <<endl;
147. cout << "stuID :" << stuID << endl;
148. cout << "Major : " << temp2 << endl;
149. cout << "Class : " << class\_num << endl;
150. cout << "------------------------------\n";
151. }
153. istream& operator >> (istream& ins, Student\_info& right\_side)
154. {
155. **char** temp[right\_side.max\_size];
156. ins >> temp ;
157. strcpy(right\_side.name,temp);
158. ins >>  right\_side.stuID;
159. ins >> temp ;
160. strcpy(right\_side.major,temp);
161. ins >> right\_side.class\_num;
162. **return** ins;
163. }
165. ostream& operator << (ostream& outs, **const** Student\_info& right\_side)
166. {
167. **char** temp1[right\_side.max\_size],temp2[right\_side.max\_size];
168. strcpy(temp1,right\_side.name);
169. strcpy(temp2,right\_side.major);
170. outs << temp1 << " " << right\_side.stuID << " " << temp2 << " " << right\_side.class\_num;
171. **return** outs;
172. }
174. **void** new\_line()
175. {
176. **char** symbol;
177. **do** {
178. cin.get(symbol);
179. } **while**(symbol != '\n');
180. }

Results:



**Reflection:**

By experimental programming for real questions, I have mastered the basic ideas to build a class and the implementation of different functions. However, during this process, I realized that I still need more practices to clear up some basic concept. It is worth mentioning that I have spent a lot of time to figure out how to initialize the class array in the first question. I have planned to define a random double type array generator but I failed. The srand() function and rand() function did not give me the answer I want, so I chose to define a initialization .txt file my self by tapping my keyboard, which is not the style of programmer. I will try to find another solution for this part.