Class CS\_UC211 Number 202115030121 Name Hao Yang

Machine number Lab3 Experiment date 2022/12/18 Report date 2022/12/23

**Experiment topic:**

Student Information Management System Based on Search and Sorting

1. **Overview**

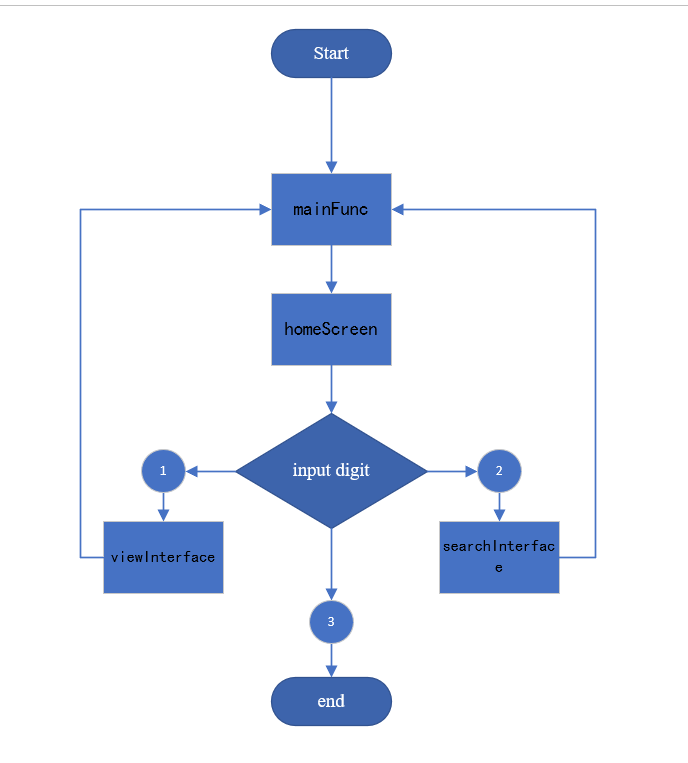
Make up groups of students and string the whole together by searching and sorting classes and interface management classes.

**2. Experimental scheme**

2.1 Design scheme

Each module is represented by a class and designed as an information management system starting from one class.

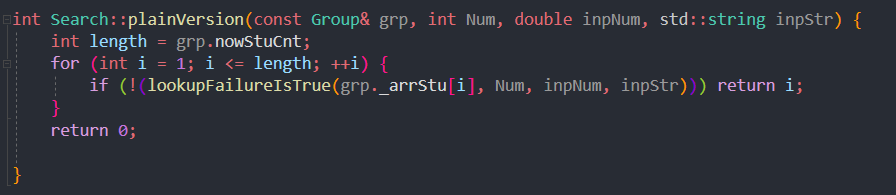
2.2 Function call relation



2.3 Key algorithm implementation

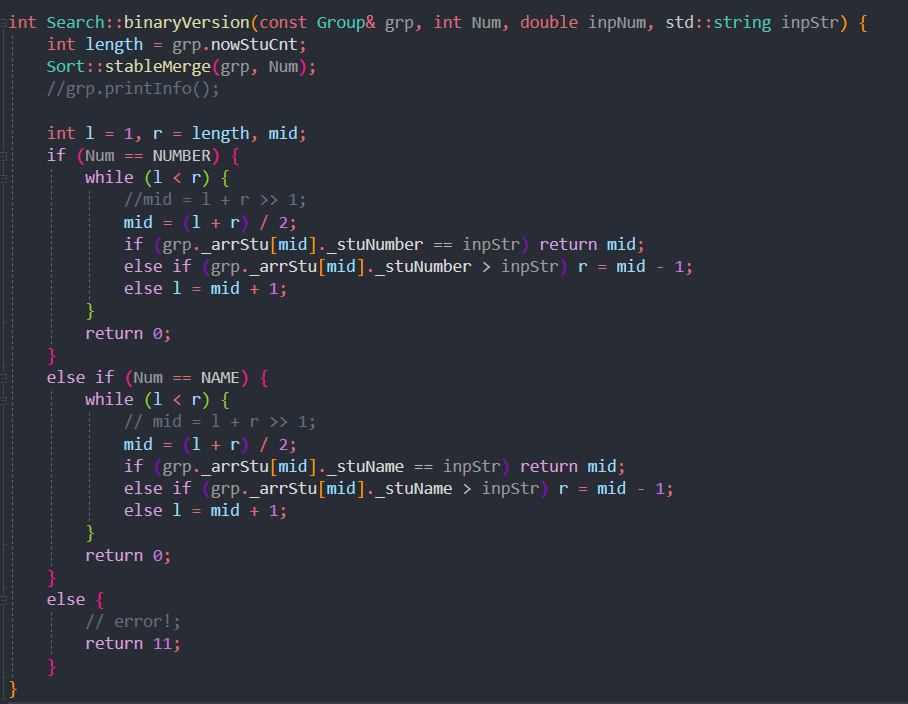
* int Search::plainVersion(const Group& grp, int Num, double inpNum, std::string inpStr)

Start traversing from 1, and search the name, number and score by judging Num.



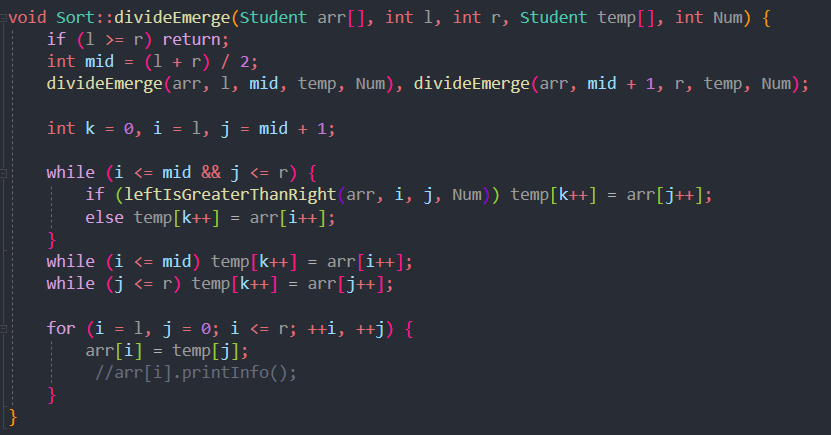
* int Search::binaryVersion(const Group& grp, int Num, double inpNum, std::string inpStr)

The name and student number are C++string types. You can compare the size of strings. Therefore, the two are searched by dichotomy. Compare the value to be searched with the mid value in a while loop. If it is greater than, r = mid - 1; otherwise, l = mid+1.



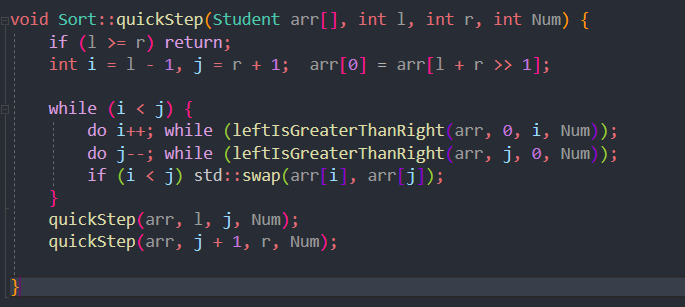
* void Sort::divideEmerge(Student arr[], int l, int r, Student temp[], int Num)

Create a new student array space with the size of nowStuCnt, reduce the array to be sorted one by one, and then merge the small arrays. Finally, we get an ordered array.



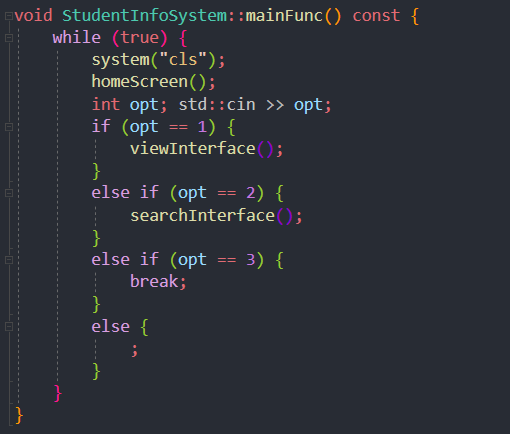
* void Sort::quickStep(Student arr[], int l, int r, int Num)

Set the middle number to keypov. If l is less than r, loop: compare the subscript array of l and r with keypov, change l or r, and then exchange the two numbers of l and r. Repeat in sequence. Finally, use recursion to perform the next quick sort.



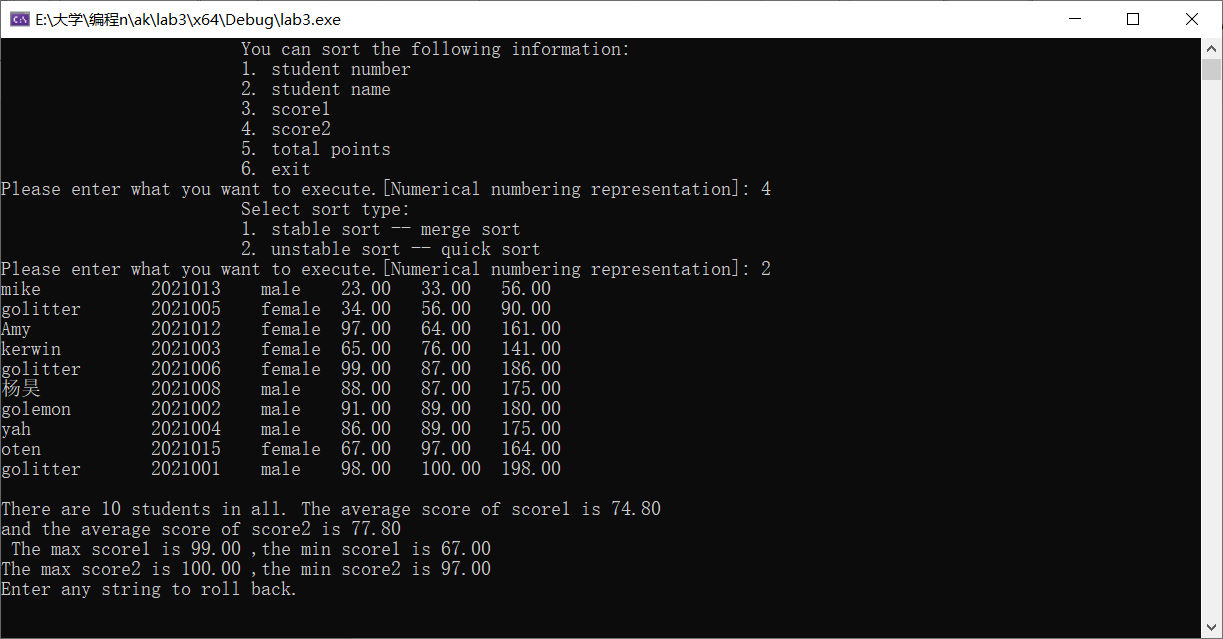
* void StudentInfoSystem::mainFunc() const

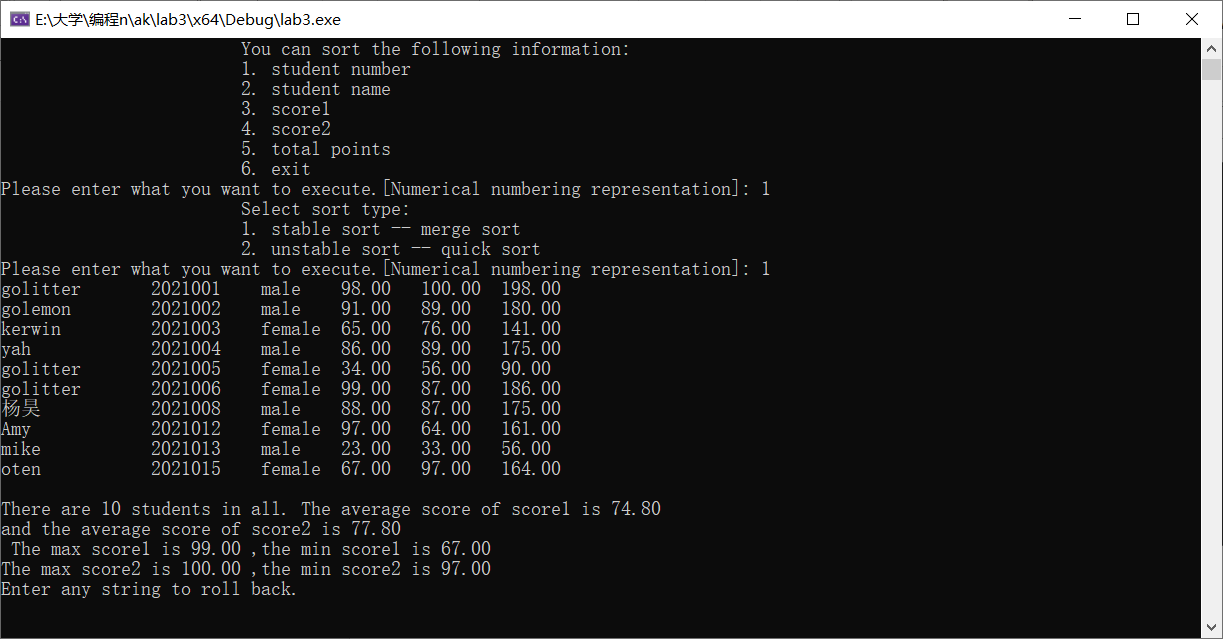
While (1) cycle, after entering, execute the text printing of the introduced steps, input the module to enter, and enter after judgment.

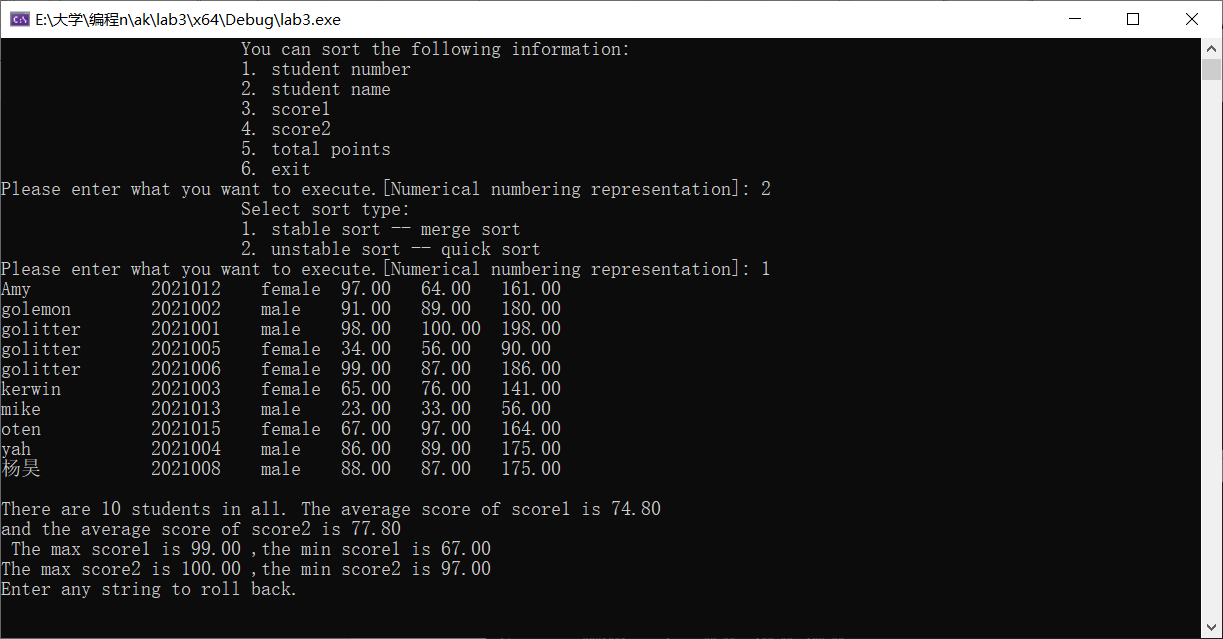


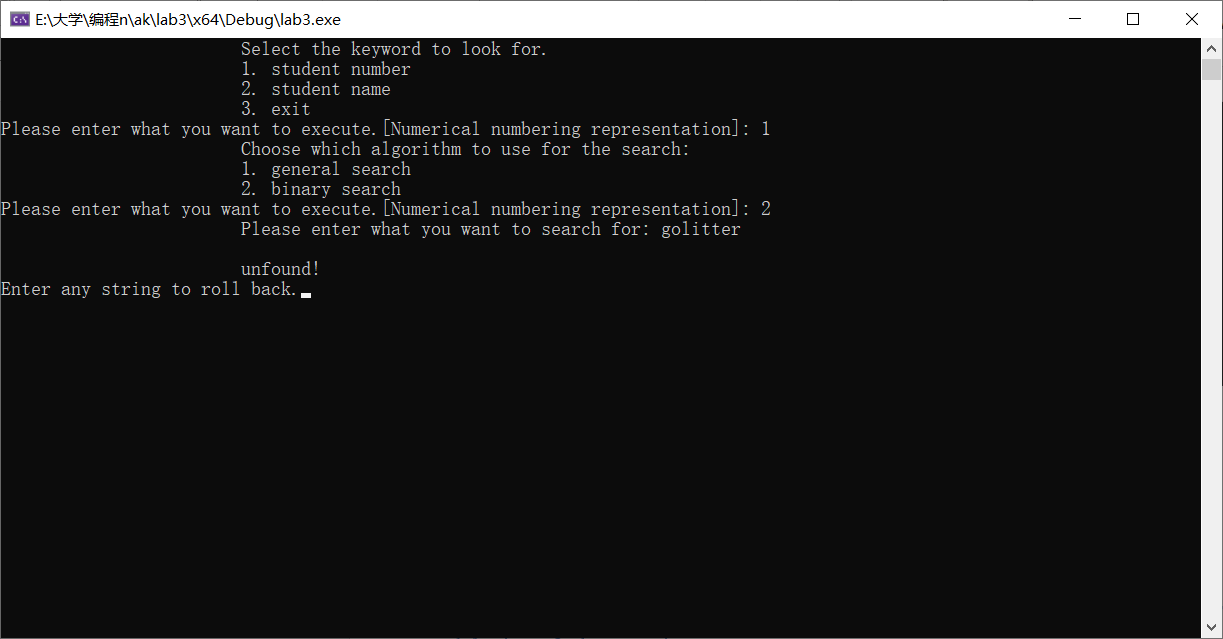
**3. Experiment process**

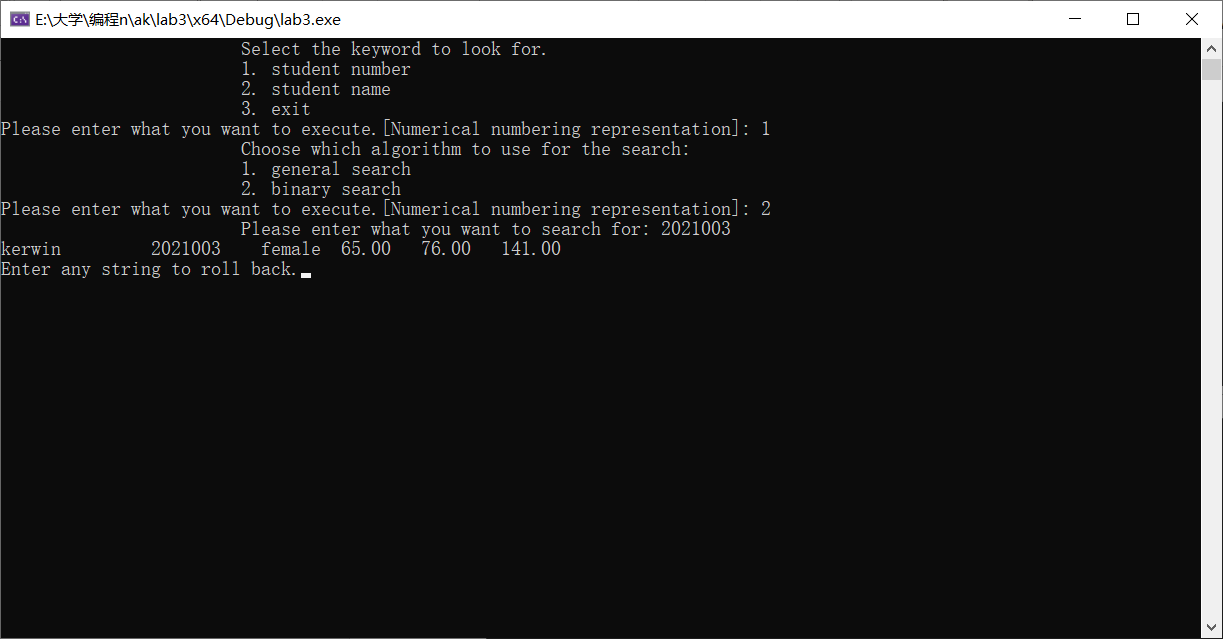
3.1 Test process











3.2 Debugging analysis

When operating on a user-defined type, no reference and const are added, resulting in consistent error reports at the beginning.

**4. Evaluation analysis**

4.1 Analysis of experimental results

Merging sorting, quick sorting and searching can correctly express the effect to be expressed after testing. The interface system code is also very simple and beautiful.

4.2 Algorithm performance evaluation

Improved the quick sorting of textbooks. Keypivot starts from the middle value. Other algorithms are relatively efficient in the student group.

**5. Summary and experience**.

Completely finished a project, from the beginning to the final interface effect.

I have a deeper understanding of the use of classes and friends.

Have a deep understanding of project construction and logical problem-solving.