Class CS\_UC211 Number 202115030121 Name Hao Yang Data 2022/12/22

**Experiment topic：**

Student Information Management System

**1. Problem analysis**

1.1 Statement

Experimental objectives

Master various storage structures and algorithms for search operation and for sort operation.

Use search algorithms and sort algorithms to solve practical problems.

Can analyze the efficiency performance of different search and sorting algorithms.

1.2 Abstraction

The sequential storage structure is used to store information, and the search and sorting algorithm is abstractly encapsulated as a tool class, while the student information management system is also encapsulated as a class.

* 1. ADT design
* Student

According to the requirements of this experiment, set the student number, name, score, total score, and other attributes for the student type. Set the constructor to encapsulate the property.

* Group

Set multiple students as a group in the information management system. In this group, there are the highest score, the lowest score, and the average score. Member methods include adding members, displaying information, etc.

* Sort

Static member method to convert this class into a tool class. See the name and meaning, and know that this class is sorted.

* Search

Same as Sort

* StudentInfoSystem

Contains member methods for student groups and interfaces.

**2.** **Experimental scheme**

2.1 Storage Scheme

Because a large number of search queries may be required, and students may also be sorted, the storage structure is selected for sequential storage.

2.2 Algorithm design

* 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.

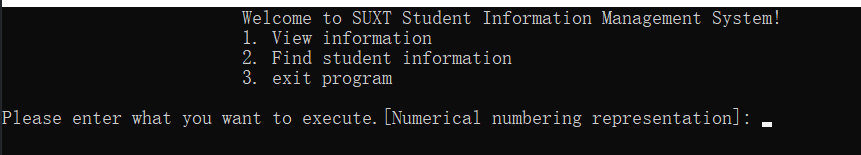
* void Sort::quickStep(Student arr[], int l, int r, int Num)

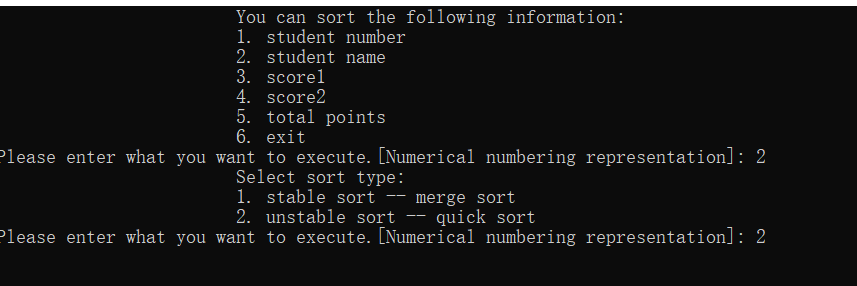
Set the middle number to keypov. If l is less than r, loop: compare the subscript array of l and r with keypov, change l or r, and then exchange the two numbers of l and r. Repeat in sequence. Finally, use recursion to perform the next quick sort.

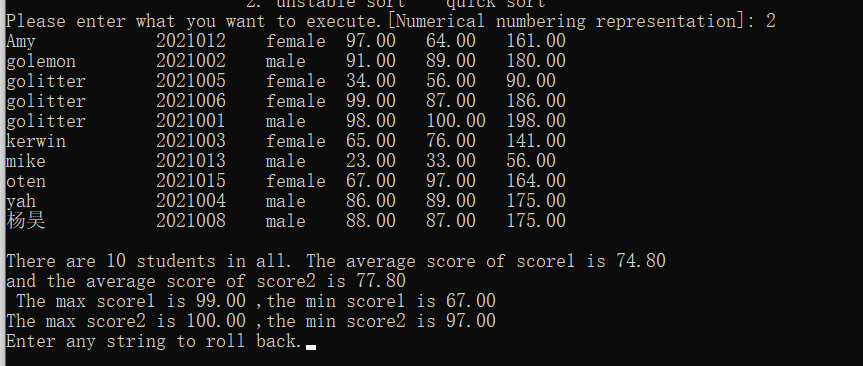
* void StudentInfoSystem::mainFunc() const

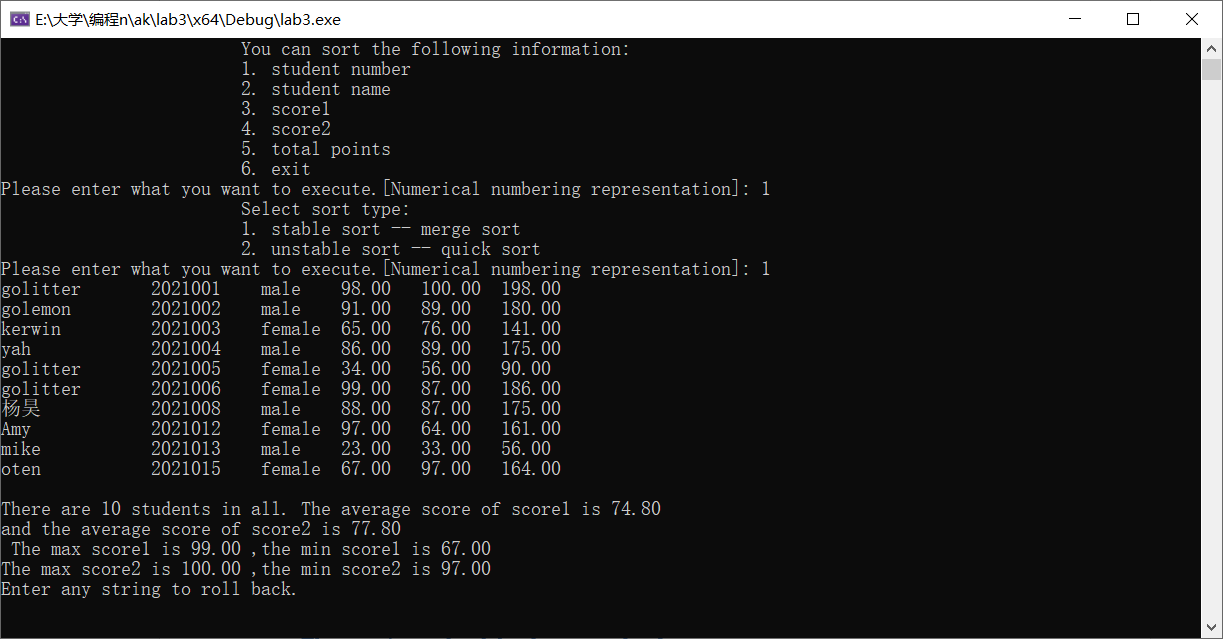
While (1) cycle, after entering, execute the text printing of the introduced steps, input the module to enter, and enter after judgment.

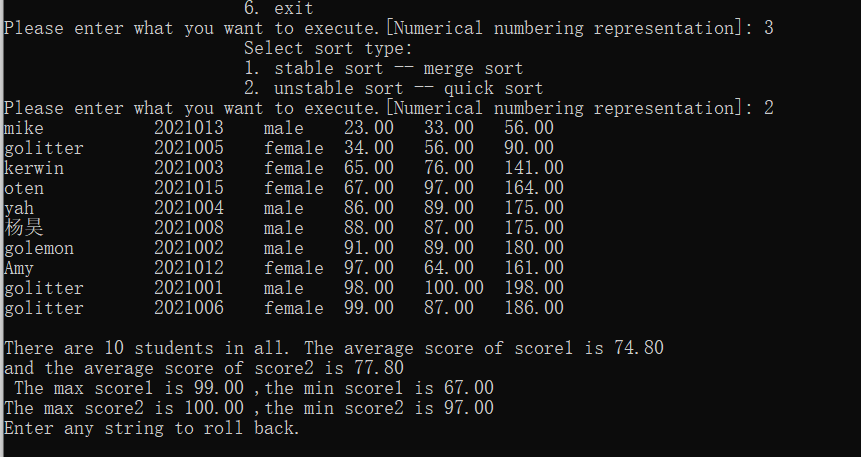
2.3 Test scheme











**3. Task solutions**

**A student information management system is designed, which should include at least student number, name, gender, score 1, score 2, total score, etc.**

**There are at least 10 student records in the file (at least two records of the same student).**

**The total score can be calculated automatically.**

**Search operation: Find student information based on a given student number or name. (Use at least two search algorithms to do this).**

**Sort operation: Sort by student number, name, grade 1, grade 2 and total score, respectively. Stable merge sort and unstable quick sort were selected.**

**Select Sort: This is another simple sort algorithm that sorts by looking for the smallest (or largest) elements. Its time complexity is also O (n^2), so it is not suitable for large datasets.**

**Insert Sort: Sort by continually removing elements from unsorted sections and inserting them into the appropriate positions in the sorted sections. Its time complexity is O (n^2), but it may be faster in some cases than bubble and select sorting.**

**Merge Sort: This is a divide-and-conquer algorithm that sorts data recursively by dividing it into smaller parts. Its time complexity is O (nlogn), so it is suitable for large datasets.**