#include "Student.h"

#pragma once

class Group

{

public:

friend class Sort;

friend class Search;

typedef int Size;

typedef Student\* stucapc;

Group() {

length = 121;

\_arrStu = new Student[121 + 1];

}

Group(Size size): length(size) {

\_arrStu = new Student[size + 1];

}

void inputStuInfo(std::string number, std::string name, bool sex, double s1, double s2);

void printInfo() const;

void printScoreInfo() const;

~Group() {

delete[] \_arrStu;

}

private:

stucapc \_arrStu;

//std::pair<double, double> \_scoAvg;

std::pair<double, double> \_scoMax = { -1,-1 };

std::pair<double, double> \_scoMin = { 0x3f3f3f3f, 0x3f3f3f3f };

int length;

int nowStuCnt;

};

#include "Group.h"

void Group::inputStuInfo(std::string number, std::string name, bool sex, double s1, double s2) {

if (nowStuCnt >= length) {

std::cout << "FULL!" << std::endl;

return;

}

\_arrStu[++nowStuCnt].setName(name);

\_arrStu[nowStuCnt].setNumber(number);

\_arrStu[nowStuCnt].setSex(sex);

\_arrStu[nowStuCnt].setScore1(s1), \_arrStu[nowStuCnt].setScore2(s2);

\_arrStu[nowStuCnt].setSumScore();

\_scoMax = { std::max(\_scoMax.first, \_arrStu[nowStuCnt].getScore1()), std::max(\_scoMax.second, \_arrStu[nowStuCnt].getScore2()) };

\_scoMin = { std::min(\_scoMax.first, \_arrStu[nowStuCnt].getScore1()), std::min(\_scoMax.second, \_arrStu[nowStuCnt].getScore2()) };

}

void Group::printInfo() const {

for (int i = 1; i <= nowStuCnt; ++i) \_arrStu[i].printInfo();

puts("");

}

void Group::printScoreInfo() const {

double avg1 = 0, avg2 = 0;

for (int i = 1; i <= nowStuCnt; ++i) {

avg1 += \_arrStu[i].getScore1();

avg2 += \_arrStu[i].getScore2();

}

// avg1 / nowStuCnt, avg2 / nowStuCnt

std::cout << "There are " << nowStuCnt << " students in all. "

<< "The average score of score1 is " << avg1 / nowStuCnt << " \nand the average score of score2 is "

<< avg2 / nowStuCnt << "\n The max score1 is " << \_scoMax.first << " ,the min score1 is " << \_scoMin.first << "\n"

<< "The max score2 is " << \_scoMax.second << " ,the min score2 is " << \_scoMin.second << "\n";

}

#include "Sort.h"

#pragma once

class Search

{

public:

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

static bool lookupFailureIsTrue(Student stu, int Num, double inpNum, std::string inpStr) ;

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

static void printInfo(const Group& grp, int pos);

};

#include "Search.h"

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

int length = grp.nowStuCnt;

for (int i = 1; i <= length; ++i) {

if (!(lookupFailureIsTrue(grp.\_arrStu[i], Num, inpNum, inpStr))) return i;

}

return 0;

}

bool Search::lookupFailureIsTrue(Student stu, int Num, double inpNum, std::string inpStr) {

if (Num == NUMBER) {

return stu.\_stuNumber != inpStr;

}

else if (Num == NAME) {

return stu.\_stuName != inpStr;

}

else if (Num == SCORE\_ONE) {

return stu.score1 != inpNum;

}

else if (Num == SCORE\_TWO) {

return stu.score2 != inpNum;

}

else if (Num == TOTAL\_SCORE) {

return stu.totalScore != inpNum;

}

else {

// error

return 0;

}

}

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

int length = grp.nowStuCnt;

Sort::stableMerge(grp, Num);

//grp.printInfo();

int l = 1, r = length, mid;

if (Num == NUMBER) {

while (l < r) {

//mid = l + r >> 1;

mid = (l + r) / 2;

if (grp.\_arrStu[mid].\_stuNumber == inpStr) return mid;

else if (grp.\_arrStu[mid].\_stuNumber > inpStr) r = mid - 1;

else l = mid + 1;

}

return 0;

}

else if (Num == NAME) {

while (l < r) {

// mid = l + r >> 1;

mid = (l + r) / 2;

if (grp.\_arrStu[mid].\_stuName == inpStr) return mid;

else if (grp.\_arrStu[mid].\_stuName > inpStr) r = mid - 1;

else l = mid + 1;

}

return 0;

}

else {

// error!;

return 11;

}

}

void Search::printInfo(const Group& grp, int pos) {

grp.\_arrStu[pos].printInfo();

}

#include "Group.h"

const int NUMBER = 0;

const int NAME = 1;

const int SEX = 2;

const int SCORE\_ONE = 3;

const int SCORE\_TWO = 4;

const int TOTAL\_SCORE = 5;

#pragma once

class Sort

{

public:

static void stableMerge(const Group &grp, int Num);

static void divideEmerge(Student arr[], int l, int r, Student temp[], int Num);

static bool leftIsGreaterThanRight(Student arr[], int ii, int jj, int Num);

static void unstableQuick(const Group& grp, int Num);

static void quickStep(Student arr[], int l, int r, int Num);

private:

};

#include "Sort.h"

void Sort::stableMerge(const Group &grp, int Num) {

int length = grp.nowStuCnt;

Student \*temp = new Student[length\*2];

divideEmerge(grp.\_arrStu, 1, length, temp, Num);

delete[] temp;

//for (int i = 1; i <= length; ++i) grp.\_arrStu[i].printInfo();

}

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

if (l >= r) return;

int mid = (l + r) / 2;

divideEmerge(arr, l, mid, temp, Num), divideEmerge(arr, mid + 1, r, temp, Num);

int k = 0, i = l, j = mid + 1;

while (i <= mid && j <= r) {

if (leftIsGreaterThanRight(arr, i, j, Num)) temp[k++] = arr[j++];

else temp[k++] = arr[i++];

}

while (i <= mid) temp[k++] = arr[i++];

while (j <= r) temp[k++] = arr[j++];

for (i = l, j = 0; i <= r; ++i, ++j) {

arr[i] = temp[j];

//arr[i].printInfo();

}

}

bool Sort::leftIsGreaterThanRight(Student arr[], int ii, int jj, int Num) {

if (Num == NUMBER) {

return arr[ii].\_stuNumber > arr[jj].\_stuNumber;

}

else if (Num == NAME) {

return arr[ii].\_stuName > arr[jj].\_stuName;

}

else if (Num == SCORE\_ONE) {

return arr[ii].score1 > arr[jj].score1;

}

else if (Num == SCORE\_TWO) {

return arr[ii].score2 > arr[jj].score2;

}

else if (Num == TOTAL\_SCORE) {

return arr[ii].totalScore > arr[jj].totalScore;

}

else {

// error!;

return 0;

}

}

void Sort::unstableQuick(const Group& grp, int Num) {

int length = grp.nowStuCnt;

quickStep(grp.\_arrStu, 1, length, Num);

}

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

if (l >= r) return;

int i = l - 1, j = r + 1; arr[0] = arr[l + r >> 1];

while (i < j) {

do i++; while (leftIsGreaterThanRight(arr, 0, i, Num));

do j--; while (leftIsGreaterThanRight(arr, j, 0, Num));

if (i < j) std::swap(arr[i], arr[j]);

}

quickStep(arr, l, j, Num);

quickStep(arr, j + 1, r, Num);

}

#include <string>

#include <iostream>

#include <iomanip>

#pragma once

class Student

{

public:

friend class Sort;

friend class Search;

Student() = default;

Student(std::string number, std::string name, bool sex, double s1, double s2) :

\_stuNumber(number), \_stuName(name), \_stuSex(sex), score1(s1), score2(s2), totalScore(s1 + s2) {};

void printInfo() const;

void setNumber(std::string number);

std::string getNumber() const;

void setName(std::string name);

std::string getName() const;

void setSex(bool fg);

std::string getSex() const;

void setScore1(double s1);

double getScore1() const;

void setScore2(double s2);

double getScore2() const;

void setSumScore();

double getSumScore() const;

private:

std::string \_stuNumber;

std::string \_stuName;

bool \_stuSex;

double score1, score2, totalScore;

};

#include "Student.h"

void Student::setNumber(std::string number) {

\_stuNumber = number;

}

std::string Student::getNumber() const {

return \_stuNumber;

}

void Student::setName(std::string name) {

\_stuName = name;

}

std::string Student::getName() const {

return \_stuName;

}

void Student::setSex(bool fg) {

\_stuSex = fg;

}

std::string Student::getSex() const {

return \_stuSex ? "male" : "female";

}

void Student::setScore1(double s1) {

score1 = s1;

}

double Student::getScore1() const {

return score1;

}

void Student::setScore2(double s2) {

score2 = s2;

}

double Student::getScore2() const {

return score2;

}

void Student::setSumScore() {

totalScore = getScore1() + getScore2();

}

double Student::getSumScore() const {

return totalScore;

}

void Student::printInfo() const {

std::cout.width(14); ;

std::cout << std::setiosflags(std::ios\_base::fixed); std::cout << std::setprecision(2);

std::cout << std::setiosflags(std::ios\_base::left) << getName() << " ";

std::cout.width(10); std::cout << getNumber() << " ";

std::cout.width(7); std::cout << getSex() << " ";

std::cout.width(7); std::cout << getScore1() << " ";

std::cout.width(7); std::cout << getScore2() <<" ";

std::cout.width(7); std::cout << getSumScore() << std::endl << std::resetiosflags(std::ios\_base::left);

}

#include "Group.h"

#include "Sort.h"

#include "Search.h"

#pragma once

class StudentInfoSystem

{

public:

StudentInfoSystem() {

group.inputStuInfo("2021001", "golitter", 1, 98, 100);

group.inputStuInfo("2021003", "kerwin", 0, 65, 76);

group.inputStuInfo("2021002", "golemon", 1, 91, 89);

group.inputStuInfo("2021004", "yah", 1, 86, 89);

group.inputStuInfo("2021005", "golitter", 0, 34, 56);

group.inputStuInfo("2021006", "golitter", 0, 99, 87);

group.inputStuInfo("2021008", "杨昊", 1, 88, 87);

group.inputStuInfo("2021012", "Amy", 0, 97, 64);

group.inputStuInfo("2021013", "mike", 1, 23, 33);

group.inputStuInfo("2021015", "oten", 0, 67, 97);

}

void mainFunc() const;

void homeScreen() const;

void viewInterface() const;

void viewScreen1() const;

void viewScreen2() const;

void selectSortMethod(int opt1, int opt2) const;

void searchInterface() const;

void searchScreen1() const;

void searchScreen2() const;

int selectSearchMethod(int opt1, int opt2, std::string content) const;

Group group;

};

#include "StudentInfoSystem.h"

void StudentInfoSystem::mainFunc() const {

while (true) {

system("cls");

homeScreen();

int opt; std::cin >> opt;

if (opt == 1) {

viewInterface();

}

else if (opt == 2) {

searchInterface();

}

else if (opt == 3) {

break;

}

else {

;

}

}

}

void StudentInfoSystem::homeScreen() const {

std::cout << "\t\t\tWelcome to SUXT Student Information Management System!\n";

std::cout << "\t\t\t1. View information\n";

std::cout << "\t\t\t2. Find student information\n";

std::cout << "\t\t\t3. exit program\n";

std::cout << std::endl;

std::cout << "Please enter what you want to execute.[Numerical numbering representation]: ";

}

void StudentInfoSystem::viewScreen1() const {

std::cout << "\t\t\tYou can sort the following information:\n";

std::cout << "\t\t\t1. student number\n";

std::cout << "\t\t\t2. student name\n";

std::cout << "\t\t\t3. score1\n";

std::cout << "\t\t\t4. score2\n";

std::cout << "\t\t\t5. total points\n";

std::cout << "\t\t\t6. exit\n";

std::cout << "Please enter what you want to execute.[Numerical numbering representation]: ";

}

void StudentInfoSystem::viewScreen2() const {

std::cout << "\t\t\tSelect sort type: \n";

std::cout << "\t\t\t1. stable sort -- merge sort\n";

std::cout << "\t\t\t2. unstable sort -- quick sort\n";

std::cout << "Please enter what you want to execute.[Numerical numbering representation]: ";

}

void StudentInfoSystem::viewInterface() const {

while (true) {

system("cls");

viewScreen1();

int opt1; std::cin >> opt1;

viewScreen2();

int opt2; std::cin >> opt2;

if (opt1 >= 1 && opt1 <= 5 && opt2 >= 1 && opt2 <= 2) {

selectSortMethod(opt1, opt2);

group.printInfo();

group.printScoreInfo();

std::cout << "Enter any string to roll back.";

std::string str; std::cin >> str;

}

else if (opt1 == 6) {

break;

}

}

}

void StudentInfoSystem::selectSortMethod(int opt1, int opt2) const {

if (opt2 == 1) {

if (opt1 == 1) Sort::stableMerge(group, opt1 - 1);

else if (opt1 == 2) Sort::stableMerge(group, opt1 - 1);

else {

Sort::stableMerge(group, opt1);

}

}

else {

if (opt1 == 1) Sort::unstableQuick(group, opt1 - 1);

else if (opt1 == 2) Sort::unstableQuick(group, opt1 - 1);

else {

Sort::unstableQuick(group, opt1);

}

}

}

void StudentInfoSystem::searchInterface() const {

while(true) {

system("cls");

searchScreen1();

int opt1; std::cin >> opt1;

searchScreen2();

int opt2; std::cin >> opt2;

if (opt1 >= 1 && opt1 <= 2 && opt2 >= 1 && opt2 <= 2) {

std::cout << "\t\t\tPlease enter what you want to search for: ";

std::string str; std::cin >> str;

int pos = selectSearchMethod(opt1, opt2, str);

if (pos == 0) {

std::cout << "\n\t\t\tunfound!\n";

}

else {

Search::printInfo(group, pos);

}

std::cout << "Enter any string to roll back.";

std::string str1; std::cin >> str1;

}

else if (opt1 == 3) {

break;

}

}

}

void StudentInfoSystem::searchScreen1() const {

std::cout << "\t\t\tSelect the keyword to look for.\n";

std::cout << "\t\t\t1. student number\n";

std::cout << "\t\t\t2. student name\n";

std::cout << "\t\t\t3. exit\n";

std::cout << "Please enter what you want to execute.[Numerical numbering representation]: ";

}

void StudentInfoSystem::searchScreen2() const {

std::cout << "\t\t\tChoose which algorithm to use for the search: \n";

std::cout << "\t\t\t1. general search\n";

std::cout << "\t\t\t2. binary search\n";

std::cout << "Please enter what you want to execute.[Numerical numbering representation]: ";

}

int StudentInfoSystem::selectSearchMethod(int opt1, int opt2, std::string content) const {

if (opt2 == 1) {

return Search::plainVersion(group, opt1 - 1, -1, content);

}

else {

return Search::binaryVersion(group, opt1 - 1, -1, content);

}

}