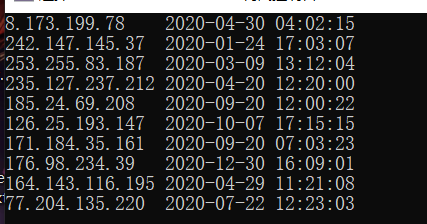
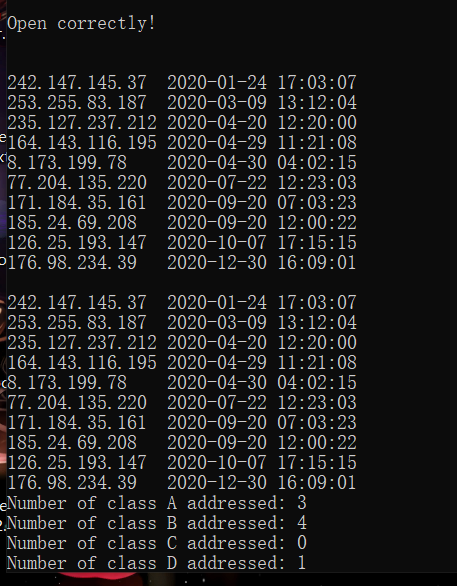
**EX1**





#define \_CRT\_SECURE\_NO\_WARNINGS

#include<iostream>

#include<fstream>

#include<string>

#include<sstream>

#include<cstdlib>

#include<algorithm>

using namespace std;

class Record

{

public:

string address;

string time\_stamp;

char str[100];

int day, month;

int hour, minute, second;

int start;

Record(const string& target)

{

int cnt = 0;

/\*for (char c : target)

if (c != 0) str[cnt++] = c;\*/

for (int i = 0; i < target.size(); i++)

if (target[i] != 0) str[cnt++] = target[i];

str[cnt] = 0;

start = 0;

for (int i = 0;i < target.size();i++)

{

if (target[i] == '.') { start /= 10; break; }

start += (target[i] - '0');

start \*= 10;

}

}

Record()

{

int temp;

string stemp;

stringstream sstemp;

for (int i = 1; i <= 4; i++)

{

temp = rand() % 256;

if (i == 1)start = temp;

sstemp << temp;

sstemp >> stemp;

address += stemp;

sstemp.clear();

if (i == 4) break;

address += '.';

}

time\_stamp += "2020-";

month = temp = rand() % 12 + 1;

if (temp <= 9)

time\_stamp += '0';

sstemp << temp;

sstemp >> stemp;

time\_stamp += stemp;

sstemp.clear();

time\_stamp += '-';

if (month == 2)

day = temp = rand() % 29 + 1;

else

day = temp = rand() % 30 + 1;

if (temp <= 9)

time\_stamp += '0';

sstemp << temp;

sstemp >> stemp;

time\_stamp += stemp;

time\_stamp += ' ';

sstemp.clear();

for (int i = 1; i <= 3; i++)

{

temp = rand() % 24;

if (i == 1) hour = temp;

else if (i == 2)minute = temp;

else second = temp;

if (temp <= 9)

time\_stamp += '0';

sstemp << temp;

sstemp >> stemp;

time\_stamp += stemp;

sstemp.clear();

if (i == 3) break;

time\_stamp += ':';

}

sprintf(str,"%-15s %-19s",address.c\_str(),time\_stamp.c\_str());

}

};

int classify(char type,Record col[],int size)

{

int numA = 0, numB = 0, numC = 0, numD = 0, numE = 0;

for (int i = 0; i < size; i++)

{

if (col[i].start <= 127)

numA++;

else if (col[i].start <= 191)//else if必大于127

numB++;

else if (col[i].start <= 223)

numC++;

else if (col[i].start <= 239)

numD++;

else if (col[i].start <= 247)

numE++;

}

switch(type)

{ //return 结束不需break

case 'A':

return numA;

case 'B':

return numB;

case 'C':

return numC;

case 'D':

return numD;

case 'E':

return numE;

default:return 0;

}

}

bool Mycompare(const Record& t1,const Record& t2)

{

bool ascending = 0;

if (t1.month > t2.month)

return ascending;

else if (t1.month < t2.month) return !ascending;

if ((t1.month == t2.month) && (t1.day > t2.day))

return ascending;

else if (t1.day < t2.day) return !ascending;

if ((t1.day == t2.day) && (t1.hour > t2.hour))

return ascending;

else if (t1.hour < t2.hour) return !ascending;

if ((t1.hour == t2.hour) && (t1.minute > t2.minute))

return ascending;

else if (t1.minute < t2.minute)return !ascending;

if ((t1.minute == t2.minute) && (t1.second > t2.second))

return ascending;

else return !ascending;

}

ostream& operator<<(ostream& cout, const Record& target)

{

cout << target.str;

//printf("%-15s %-19s",target.address.c\_str(),target.time\_stamp.c\_str());

return cout;

}

int main()

{

srand(time(0));

Record collection[10];

for (int i = 0; i < 10; i++)

cout << collection[i] << endl;

//for (Record& i : collection)

//{

// cout << i << endl;

//}

sort(collection, collection + 10, Mycompare);

//sort函数对于排序的检查

cout << endl << endl;

cout << endl << endl;

fstream target;

target.open("record.txt",ios::out);

if (target)

cout << "Open correctly!" << endl;

for (int i = 0; i < 10; i++)

target << collection[i] << endl;

/\*for (Record& i : collection)

target << i <<endl;\*/

target.close();

cout << endl << endl;

string temp;

target.open("record.txt", ios::in);

int cnt = 0;

while (!target.eof())

{

getline(target, temp);

cout << temp << endl;

if (!temp.empty())

collection[cnt++] = Record(temp);

}

for (int i = 0; i < 10; i++)

cout << collection[i] << endl;

/\*for (Record& i : collection)

cout << i << endl;\*/

for (char c = 'A'; c != 'F'; c++)

{

printf("Number of class %c addressed: %d\n", c, classify(c, collection, 10));

}

}

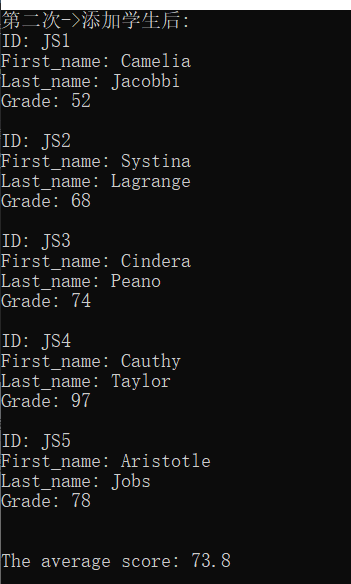
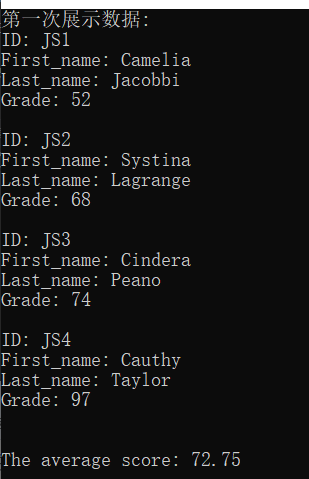
问题与思考：

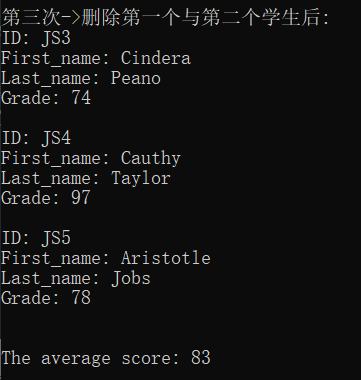
里面使用了Stringstream进行数据(实现数字转字符串)的转换，便于后续操作，sprintf也能实现数据格式标准化；

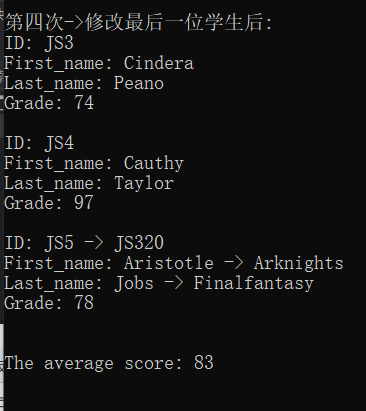
当然，如果不要求txt，完全可以使用二进制文件，通过read和write来进行读写，只需要重载读取输出的函数就行，不需要单独使用char[]来装要输出的字符串，不过写的时候没考虑到，这里仍采用此方式；

Sort中使用了自己写的排序谓词，发现挺有意思的，sort会对传进来的排序谓词有检查，如果不符合一些要求，vs2019这里会报错。

**EX2：**







#define \_CRT\_SECURE\_NO\_WARNINGS

#include<iostream>

#include<string>

#include<fstream>

#include<vector>

using namespace std;

class Student;//此处若不声明类，下方友元无效

class Record

{

friend ostream& operator<<(ostream& cout, Student& target);

friend class Student;

string id;

string first\_name;

string last\_name;

int grade;

public:

Record(const string& Id = "None", const string& f\_n = "None", const string& l\_n = "None", const int& score = 0)

{

setId(Id);

setFirst\_name(f\_n);

setLast\_name(l\_n);

setGrade(score);

}

void setId(const string& Id)

{

if (Id.empty())

{

printf("Error ID!!!\n");

this->id = "None";

}

else this->id = Id;

}

void setFirst\_name(const string& f\_n)

{

if (f\_n.empty())

{

printf("Error First\_name!!!\n");

this->first\_name = "None";

}

else this->first\_name = f\_n;

}

void setLast\_name(const string& l\_n)

{

if (l\_n.empty())

{

printf("Error Last\_name!!!\n");

this->last\_name = "None";

}

else this->last\_name = l\_n;

}

void setGrade(const int& score)

{

if (score < 0)

{

printf("Error Grade!!!\n");

this->grade = 0;

}

else this->grade = score;

}

};

class Student

{

friend ostream& operator<<(ostream& cout, Student& target);

Record data;

char str[100];

public:

Student(const string& Id = "None", const string& f\_n = "None", const string& l\_n = "None", const int& score = 0) :data(Id, f\_n, l\_n, score) {}

void display()

{

sprintf(str,"ID: %s\nFirst\_name: %s\nLast\_name: %s\nGrade: %d\n", data.id.c\_str(), data.first\_name.c\_str(), data.last\_name.c\_str(),data.grade);

cout << str;

}

void setId(const string& Id)

{

data.setId(Id);

}

void setFirst\_name(const string& f\_n)

{

data.setFirst\_name(f\_n);

}

void setLast\_name(const string& l\_n)

{

data.setLast\_name(l\_n);

}

void setGrade(const int& score)

{

data.setGrade(score);

}

string getId()

{

return this->data.id;

}

string getFirst\_name()

{

return this->data.first\_name;

}

string getLast\_name()

{

return this->data.last\_name;

}

int getGrade()

{

return this->data.grade;

}

void setStr()

{

sprintf(this->str, "ID: %s\nFirst\_name: %s\nLast\_name: %s\nGrade: %d\n", data.id.c\_str(), data.first\_name.c\_str(), data.last\_name.c\_str(), data.grade);

}

char\* getStr()

{

return str;

}

};

ostream& operator<<(ostream& cout, Student& target)

{

Record data = target.data;

sprintf(target.str, "ID: %s\nFirst\_name: %s\nLast\_name: %s\nGrade: %d\n", data.id.c\_str(), data.first\_name.c\_str(), data.last\_name.c\_str(), data.grade);

cout << target.str;

return cout;

}

void recordImport(vector<Student>& col,const char\* filename = "file.dat")

{

fstream target;

target.open(filename , ios::binary | ios::out);

/\*for (int i = 0; i < col.size(); i++)

target << col[i] << endl;\*/

for (int i = 0; i < col.size(); i++)

{

col[i].setStr();

target.write(col[i].getStr(), 100);

}

target.close();

}

void displayAll(vector<Student>& col, const char\* filename = "file.dat")

{

fstream target;

target.open("file.dat", ios::binary | ios::in);

/\*string temp;\*/

//while (!target.eof())

//{

// /\*getline(target, temp);

// cout << temp << endl;\*/

//

//}

for (int i = 0; i < col.size(); i++)

{

target.read(col[i].getStr(), 100);

cout << col[i].getStr() << endl;

}

target.close();

double tempSum = 0;

for (int i = 0; i < col.size(); i++)

tempSum += col[i].getGrade();

cout << "The average score: ";

cout << tempSum / col.size() << endl << endl;

}

void recordDelete(vector<Student>& col,int index,const char\* filename = "file.dat")

{

fstream target;

target.open(filename, ios::binary | ios::out);

col.erase(col.begin()+index);

/\*for (int i = 0; i < col.size(); i++)

target << col[i] << endl;\*/

for (int i = 0; i < col.size(); i++)

{

col[i].setStr();

target.write(col[i].getStr(), 100);

}

target.close();

}

void recordAdd(const Student& student,vector<Student>& col, int index, const char\* filename = "file.dat")

{

col.insert(col.begin() + index, student);

fstream target;

target.open(filename, ios::binary | ios::out);

/\*for (int i = 0; i < col.size(); i++)

target << col[i] << endl;\*/

for (int i = 0; i < col.size(); i++)

{

col[i].setStr();

target.write(col[i].getStr(), 100);

}

target.close();

}

void recordModify(vector<Student>& col, int index, string id,string first\_name,string last\_name,int grade,const char\* filename = "file.dat")

{

col[index].setId(id);

col[index].setFirst\_name(first\_name);

col[index].setLast\_name(last\_name);

col[index].setGrade(grade);

fstream target;

target.open(filename, ios::binary | ios::out);

/\*for (int i = 0; i < col.size(); i++)

target << col[i] << endl;\*/

for (int i = 0; i < col.size(); i++)

{

col[i].setStr();

target.write(col[i].getStr(), 100);

}

target.close();

}

int main()

{

vector<Student> col(4);

col[0] = Student("JS1", "Camelia", "Jacobbi", 52);

col[1] = Student("JS2", "Systina", "Lagrange", 68);

col[2] = Student("JS3", "Cindera", "Peano", 74);

col[3] = Student("JS4", "Cauthy", "Taylor", 97);

//数据导入

recordImport(col);

//展示第一次导入数据

printf("第一次展示数据:\n");

displayAll(col);

//

recordAdd(Student("JS5", "Aristotle", "Jobs", 78),col,col.size());

printf("第二次->添加学生后:\n");

//

displayAll(col);

//

printf("第三次->删除第一个与第二个学生后:\n");

recordDelete(col, 1);

recordDelete(col, 0);

//

displayAll(col);

//

printf("第四次->修改最后一位学生后:\n");

recordModify(col, col.size() - 1, "JS5 -> JS320", "Aristotle -> Arknights", "Jobs -> Finalfantasy", 78);

displayAll(col);

}

问题与思考：

里面的str对于用二进制方式write和read是不必要的，转换Student类为char\*，后接sizeof可以解决问题，这里使用的原因是因为我原本尝试这样写：用流运算符输出，并用getline读取，用一个string接受并输出，然后成功了，结果正确，我觉得这种方式应该是不太行，但确实能成功，挺有意思，不妨尝试一下（下方代码）。（测试IDE：vs2019，标准：默认vs2019默认C++14）。

这里没有使用seekp()这样的函数进行文件中“随机访问”，而是选择修改储存数组中的内容，一并更新数据，来做到“随机访问”。

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<iostream>

#include<string>

#include<fstream>

#include<vector>

using namespace std;

class Student;//此处若不声明类，下方友元无效

class Record

{

friend ostream& operator<<(ostream& cout, Student& target);

friend class Student;

string id;

string first\_name;

string last\_name;

int grade;

public:

Record(const string& Id = "None", const string& f\_n = "None", const string& l\_n = "None", const int& score = 0)

{

setId(Id);

setFirst\_name(f\_n);

setLast\_name(l\_n);

setGrade(score);

}

void setId(const string& Id)

{

if (Id.empty())

{

printf("Error ID!!!\n");

this->id = "None";

}

else this->id = Id;

}

void setFirst\_name(const string& f\_n)

{

if (f\_n.empty())

{

printf("Error First\_name!!!\n");

this->first\_name = "None";

}

else this->first\_name = f\_n;

}

void setLast\_name(const string& l\_n)

{

if (l\_n.empty())

{

printf("Error Last\_name!!!\n");

this->last\_name = "None";

}

else this->last\_name = l\_n;

}

void setGrade(const int& score)

{

if (score < 0)

{

printf("Error Grade!!!\n");

this->grade = 0;

}

else this->grade = score;

}

};

class Student

{

friend ostream& operator<<(ostream& cout, Student& target);

Record data;

char str[100];

public:

Student(const string& Id = "None", const string& f\_n = "None", const string& l\_n = "None", const int& score = 0) :data(Id, f\_n, l\_n, score) {}

void display()

{

sprintf(str,"ID: %s\nFirst\_name: %s\nLast\_name: %s\nGrade: %d\n", data.id.c\_str(), data.first\_name.c\_str(), data.last\_name.c\_str(),data.grade);

cout << str;

}

void setId(const string& Id)

{

data.setId(Id);

}

void setFirst\_name(const string& f\_n)

{

data.setFirst\_name(f\_n);

}

void setLast\_name(const string& l\_n)

{

data.setLast\_name(l\_n);

}

void setGrade(const int& score)

{

data.setGrade(score);

}

string getId()

{

return this->data.id;

}

string getFirst\_name()

{

return this->data.first\_name;

}

string getLast\_name()

{

return this->data.last\_name;

}

int getGrade()

{

return this->data.grade;

}

};

ostream& operator<<(ostream& cout, Student& target)

{

Record data = target.data;

sprintf(target.str, "ID: %s\nFirst\_name: %s\nLast\_name: %s\nGrade: %d\n", data.id.c\_str(), data.first\_name.c\_str(), data.last\_name.c\_str(), data.grade);

cout << target.str;

return cout;

}

void recordImport(vector<Student>& col,const char\* filename = "file.dat")

{

fstream target;

target.open(filename , ios::binary | ios::out);

for (int i = 0; i < col.size(); i++)

target << col[i] << endl;

target.close();

}

void displayAll(vector<Student>& col, const char\* filename = "file.dat")

{

fstream target;

target.open("file.dat", ios::binary | ios::in);

string temp;

while (!target.eof())

{

getline(target, temp);

cout << temp << endl;

}

target.close();

double tempSum = 0;

for (int i = 0; i < col.size(); i++)

tempSum += col[i].getGrade();

cout << "The average score: ";

cout << tempSum / col.size() << endl << endl;

}

void recordDelete(vector<Student>& col,int index,const char\* filename = "file.dat")

{

fstream target;

target.open(filename, ios::binary | ios::out);

col.erase(col.begin()+index);

for (int i = 0; i < col.size(); i++)

target << col[i] << endl;

target.close();

}

void recordAdd(const Student& student,vector<Student>& col, int index, const char\* filename = "file.dat")

{

col.insert(col.begin() + index, student);

fstream target;

target.open(filename, ios::binary | ios::out);

for (int i = 0; i < col.size(); i++)

target << col[i] << endl;

target.close();

}

void recordModify(vector<Student>& col, int index, string id,string first\_name,string last\_name,int grade,const char\* filename = "file.dat")

{

col[index].setId(id);

col[index].setFirst\_name(first\_name);

col[index].setLast\_name(last\_name);

col[index].setGrade(grade);

fstream target;

target.open(filename, ios::binary | ios::out);

for (int i = 0; i < col.size(); i++)

target << col[i] << endl;

target.close();

}

int main()

{

vector<Student> col(4);

col[0] = Student("JS1", "Camelia", "Jacobbi", 52);

col[1] = Student("JS2", "Systina", "Lagrange", 68);

col[2] = Student("JS3", "Cindera", "Peano", 74);

col[3] = Student("JS4", "Cauthy", "Taylor", 97);

//数据导入

recordImport(col);

//展示第一次导入数据

printf("第一次展示数据:\n");

displayAll(col);

//

recordAdd(Student("JS5", "Aristotle", "Jobs", 78),col,col.size());

printf("第二次->添加学生后:\n");

//

displayAll(col);

//

printf("第三次->删除第一个与第二个学生后:\n");

recordDelete(col, 1);

recordDelete(col, 0);

//

displayAll(col);

//

printf("第四次->修改最后一位学生后:\n");

recordModify(col, col.size() - 1, "JS5 -> JS320", "Aristotle -> Arknights", "Jobs -> Finalfantasy", 78);

displayAll(col);

}