代码：（截图在代码后面）

#include <iostream>

#include <string>

using namespace std;

typedef struct student

{

int score;

string name;

} stu;

class Sqlist

{

private:

stu \*s;

int \_length;

int \_low;

int \_high;

int \_mid;

void Swap(stu &a , stu &b);

int Partition (int low, int high);//快速排序

public:

Sqlist(int length = 0);

//功能函数

void Show();

//排序算法函数

void InsertionSort ();//直接插入法

void BInsertSort ();//折半插入法

void ShellInsert ();//希尔排序

void BubbleSort();//冒泡排序

void QuickSort();//快速排序

void SelectSort ();//简单选择排序

};

void Sqlist::Swap(stu &a , stu &b)

{

stu t = a;

a = b;

b = t;

}

Sqlist::Sqlist(int length)

{

\_length = length;

s = new stu[length+1];

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

{

cout << "姓名：";

cin >> s[i].name;

cout << "分数：";

cin >> s[i].score;

}

this->\_low = 0;

this->\_high = 0;

}

void Sqlist::Show()

{

for(int i = 1 ; i < this->\_length + 1 ; ++i)

{

cout << "姓名：" << s[i].name << "分数：" << s[i].score << endl;

}

cout << endl;

}

void Sqlist::InsertionSort()//直接插入法

{

cout << "============" << endl;

cout << "=直接插入法=" << endl;

cout << "============" << endl;

int i = 0 , j = 0;

for(i = 2 ; i < this->\_length + 1 ; ++i)

{

if(s[i].score < s[i-1].score)

{

s[0] = s[i];

for(j = i-1 ; s[0].score < s[j].score ; --j)

{

s[j+1] = s[j];

}

s[j+1] = s[0];

}

cout << "第" << i-1 << "轮:" << endl;

this->Show();

}

}

void Sqlist::BInsertSort() //折半插入法

{

cout << "============" << endl;

cout << "=折半插入法=" << endl;

cout << "============" << endl;

int i = 0 , j = 0;

for(int i = 2 ; i < this->\_length + 1 ; ++i)

{

s[0] = s[i];

\_low = 1;

\_high = i-1;

while(\_low <= \_high)

{

\_mid = (\_low+\_high)/2;

if(s[0].score < s[\_mid].score)

{

\_high = \_mid - 1;

}

else

{

\_low = \_mid + 1;

}

}

for(j = i-1 ; j >= \_high + 1 ; --j)

{

s[j+1] = s[j];

}

s[\_high+1] = s[0];

cout << "第" << i-1 << "轮:" << endl;

this->Show();

}

}

void Sqlist::BubbleSort() //冒泡排序

{

cout << "============" << endl;

cout << "=冒泡排序法=" << endl;

cout << "============" << endl;

int i = 0 , j = 0;

for(i = 1 ; i < this->\_length + 1 ; ++i)

{

for(j = 1 ; j < this->\_length + 1 - i ; ++j)

{

if(s[j].score > s[j+1].score)

{

stu t = s[j];

s[j] = s[j+1];

s[j+1] = t;

}

}

cout << "第" << i << "轮:" << endl;

this->Show();

}

}

void Sqlist::SelectSort() //简单选择排序

{

cout << "==============" << endl;

cout << "=简单选择排序=" << endl;

cout << "==============" << endl;

int i = 0 , j = 0;

for(i = 1 ; i < this->\_length + 1 ; ++i)

{

for(j = i+1 ; j < this->\_length + 1 ; ++j)

{

if(s[i].score > s[j].score)

{

stu t = s[i];

s[i] = s[j];

s[j] = t;

}

}

cout << "第" << i << "轮:" << endl;

this->Show();

}

}

void Sqlist::ShellInsert() //希尔排序

{

cout << "============" << endl;

cout << "=希尔排序法=" << endl;

cout << "============" << endl;

int i = 0 , j = 0 , k = 0;

int dk = this->\_length;

while(dk > 1)

{

dk = (dk/3)+1;

for(i = dk+1 ; i < this->\_length + 1 ; ++i)

{

for(j = i - dk ; j > 0 ; j -= dk)

{

if(s[j].score > s[j+dk].score)

{

this->Swap(s[j] , s[j+dk]);

}

}

}

}

}

int Sqlist::Partition(int low , int high) //快速排序

{

if(low > high)

{

return 1;

}

this->\_low = low;

this->\_high = high;

s[0] = s[low];

while(this->\_low != this->\_high) //这里可以写成小于号，但是不能写小于等于号，因为等于的情况下，会进入死循环

{

while(s[0].score <= s[\_high].score && \_low < \_high)

{

--this->\_high;

}

while(s[0].score >= s[\_low].score && \_low < \_high)

{

++this->\_low;

}

if(\_low < \_high)

{

this->Swap(s[\_low] , s[\_high]);

}

}

this->Swap(s[low] , s[\_low]);

this->Partition(low , \_low-1);

this->Partition(\_low+1 , high);

}

void Sqlist::QuickSort()

{

cout << "============" << endl;

cout << "=快速排序法=" << endl;

cout << "============" << endl;

this->Partition(1 , this->\_length);

}

//aaa 87 bbb 76 ccc 92 ddd 64 eee 55 fff 78 ggg 100 hhh 43

int main()

{

Sqlist s(8);

s.InsertionSort();

s.Show();

s.BInsertSort();

s.Show();

s.BubbleSort();

s.Show();

s.SelectSort();

s.Show();

s.QuickSort();

s.Show();

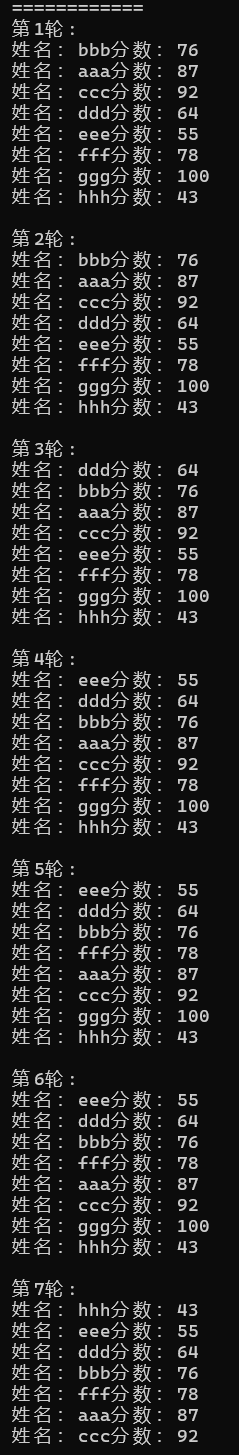
s.ShellInsert();

s.Show();

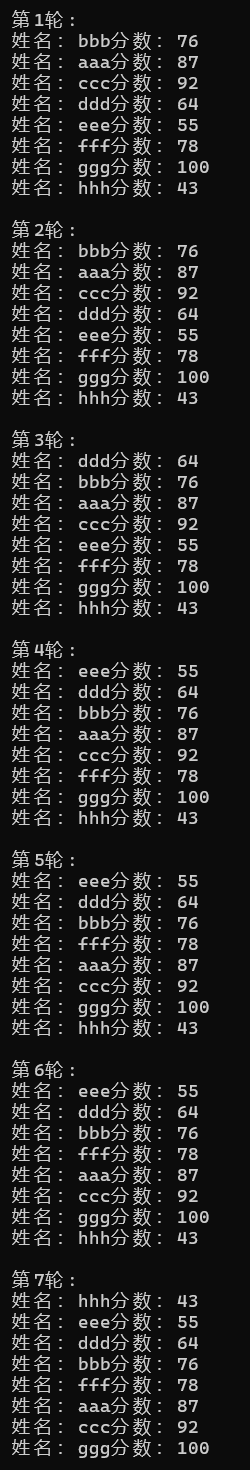
}

截图：

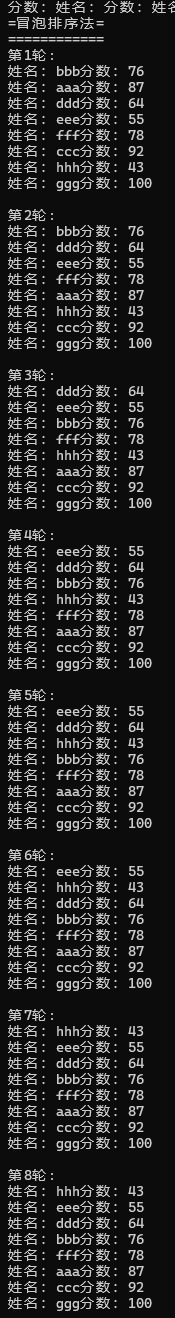
直接插入法：



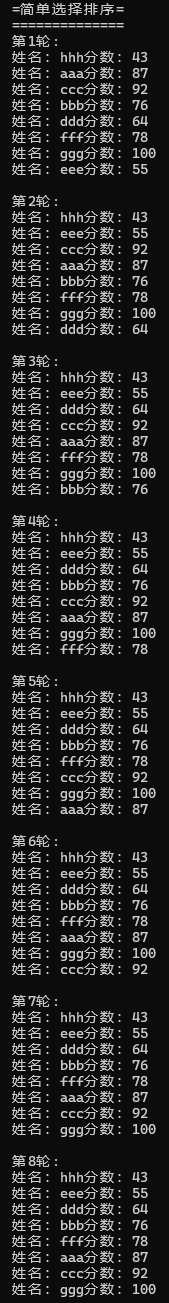
折半插入法：



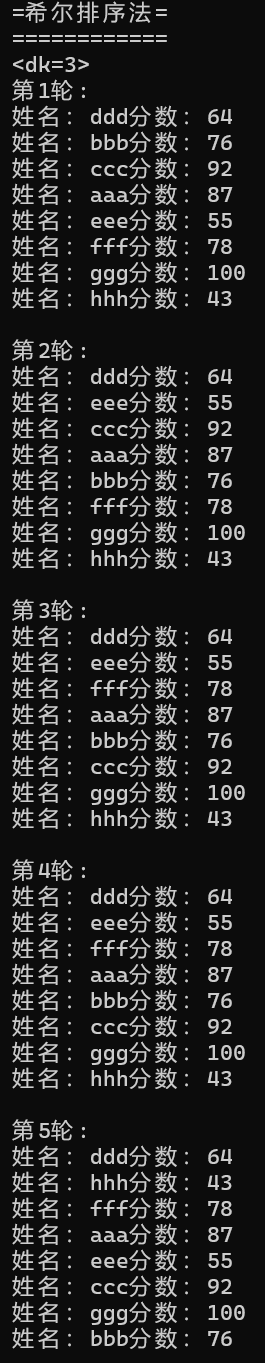
冒泡排序：



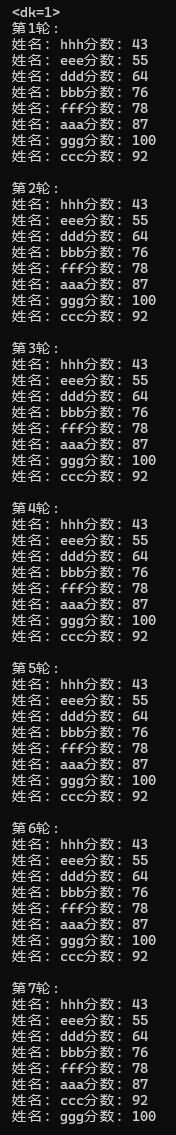
简单选择排序：



希尔排序：







快速排序：

