哈尔滨工业大学计算机科学与技术学院

实验报告

课程名称： 数据结构与算法分析

课程类型：必修

实验项目名称： 查找结构与排序方法

实验题目：内部排序算法的实现及时间性能的比较

班级：11403102

学号：1143220116

姓名：匡盟盟

|  |  |  |
| --- | --- | --- |
| 设计成绩 | 报告成绩 | 指导老师 |
|  |  |  |

一、实验目的

本实验的目的是实现直接插入排序、折半插入排序，气泡排序、快速排序、选择排序、堆排序，基数排序七种排序方法，并对时间性能进行比较。

二、实验要求及实验环境

**实验要求：**

1.实现直接插入排序、折半插入排序，气泡排序、快速排序、选择排序、堆排序，基数排序等排序方法（基数排序时数据的最大位数不超过3 位）\*由于我多写了一个求figure的函数，所以这一项我没有考虑，我认为我的比要求的好，但我在最后附上题目要求的代码；

2. 用户输入待排序的数据规模N，利用随机函数产生N个随机整数：N=500，1000，1500，2000，2500，…,30000（数据规模可自行调整）；

3. 用户可以选择排序方式（直接插入排序、折半插入排序、气泡排序、快速排序、选择排序、堆排序、基数排序）进行排序，结果为由小到大的顺序；

4.对于每一种排序方法，统计该排序方法对不同数据规模的数据进行排序所耗费的时间。在报告中以图表坐标形式给出（横坐标为数据规模，纵坐标为时间），并与理论结果进行比较和分析。

三、设计思想（本程序中的用到的所有数据类型的定义，主程序的流程图及各程序模块之间的调用关系）

开始

提示用户选择生成元素方式！

根据输入选择生成程序

输入2

输入1

调用CreateNumber（）

自动生成元素

手动输入元素

输入其他

提示选择排序程序

根据用户选择进行不同排序

………………………………

排序7

排序1

输出排序后结果，并提示用户要不要继续

用户输入Y y

用户选择YorN

用户输入

N

n

结束

1．逻辑设计

待排序表使用顺序表的形式，堆排时采用最大堆，基数排序时，用了STL中的queue。各种算法与老师教案相同或相似，在此不再赘述。

2．物理设计

待排序表物理结构：typedef struct records {

keytype key;

}LIST[MAX];

四、时间性能

**1、总体测试：**

N=500，1000，1500，2000，2500，10000，30000

（1）7种排序时间比较（总）：

分析：

由图可见，记录少时时间基本为0，记录大时，直接插入和气泡排序明显较耗时。

**（2）分项排序：**

分析：

随着数据量增大，每种排序时间均有所增长；

（3）分记录大小时间比较：

分析：

通过7种排序在不用记录数下时间分析，发现：

①堆排序时间起伏不大；

②在数据量小时，前6种排序时间基本没什么差别，而基数排序时间较多；

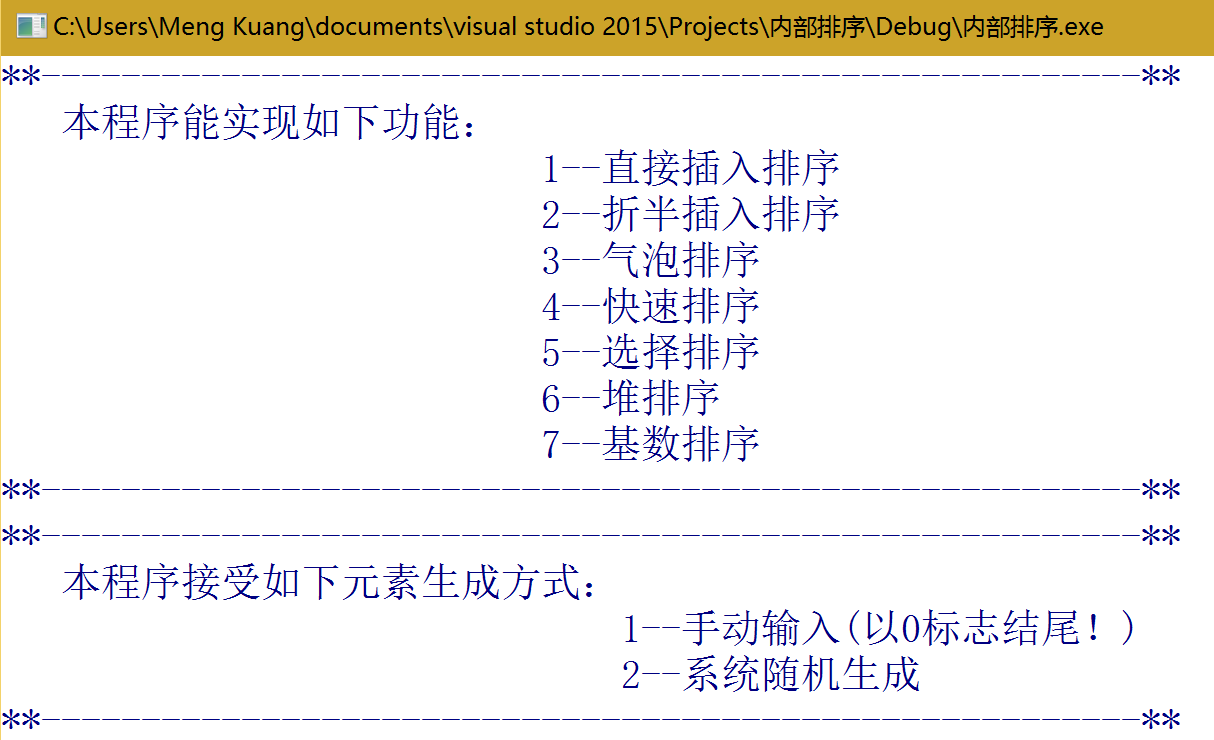
③当数据量较大时，折半插入，快速，选择，堆这几种排序时间相对较少；

④当数据量进一步增大时，折半插入，快速，选择，堆，基数这几种排序时间明显较少。

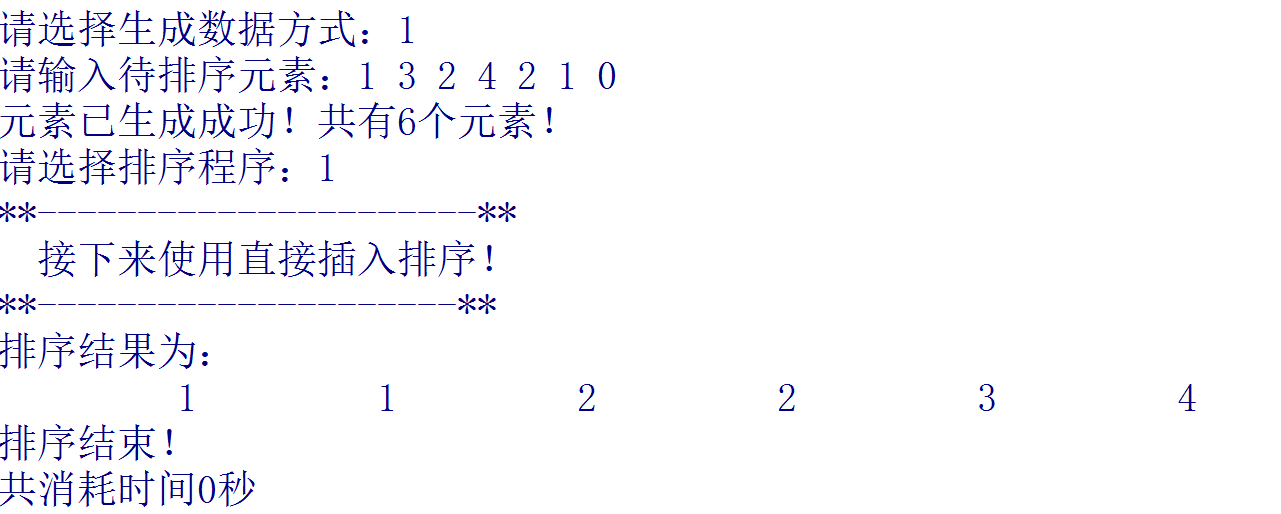
五、测试结果

测试结果截图如下：

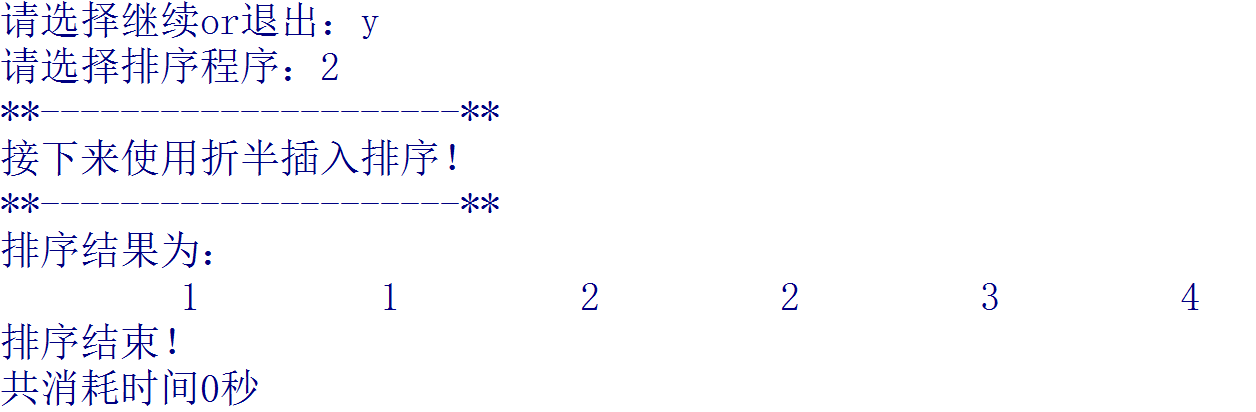
首先是程序介绍：



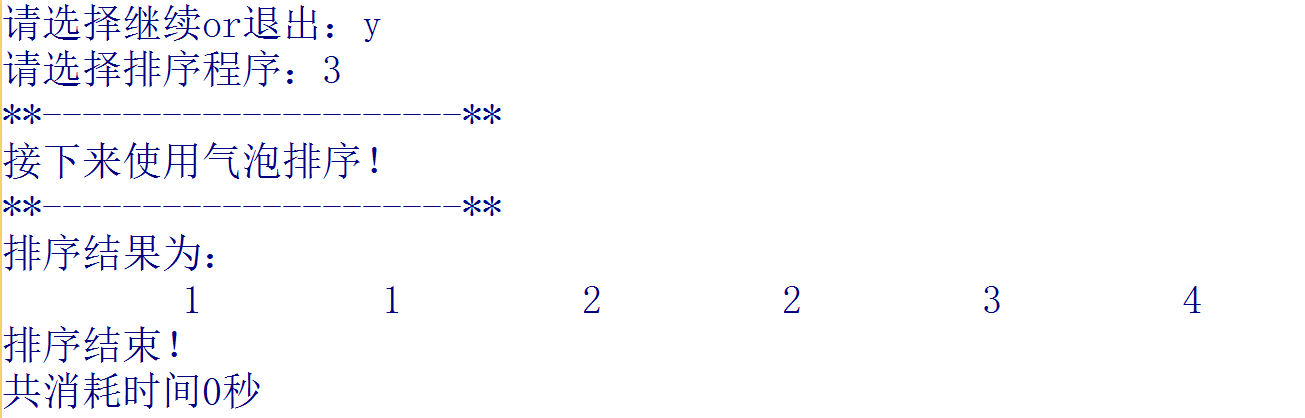
先以第一种方式生成待排序元素并进行直接插入排序：



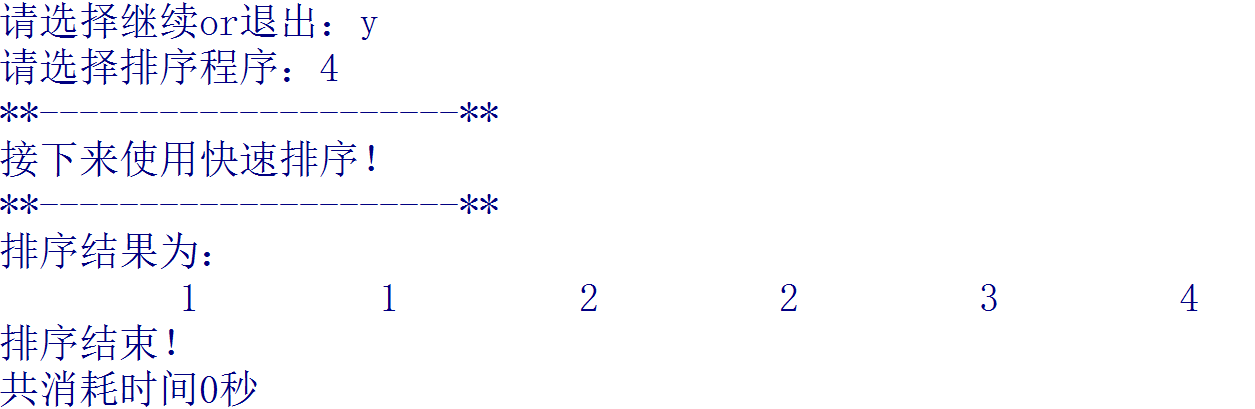
继续选择折半插入排序：



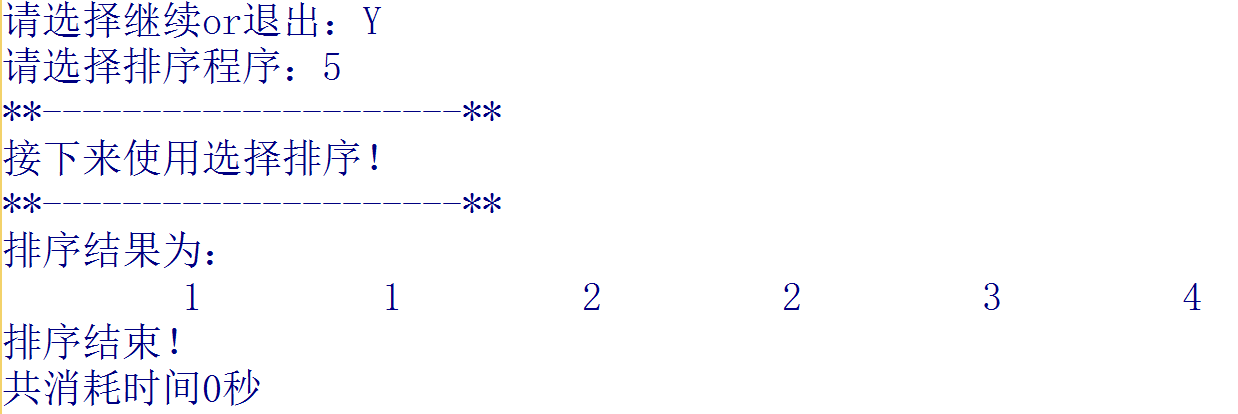
接着选择气泡排序：



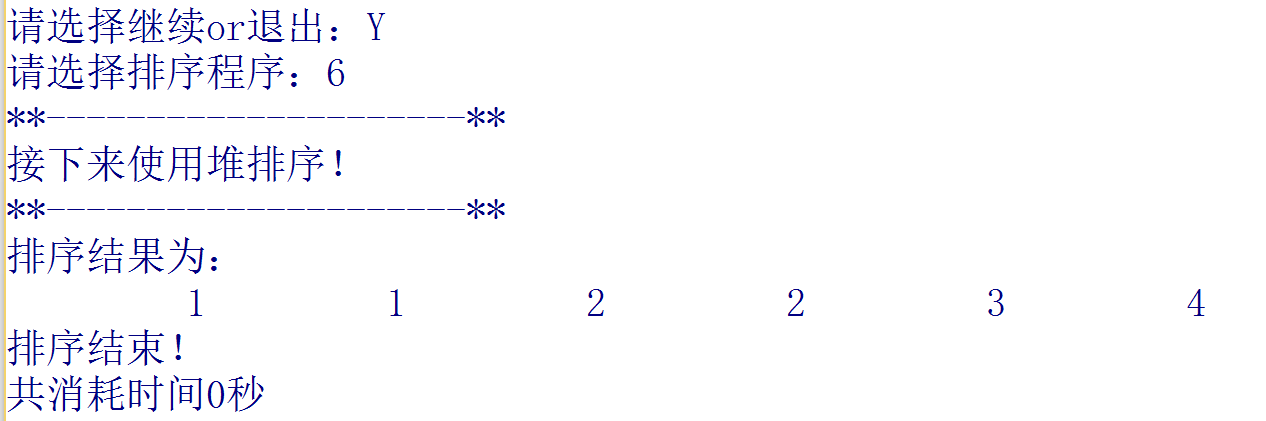
再使用快速排序：



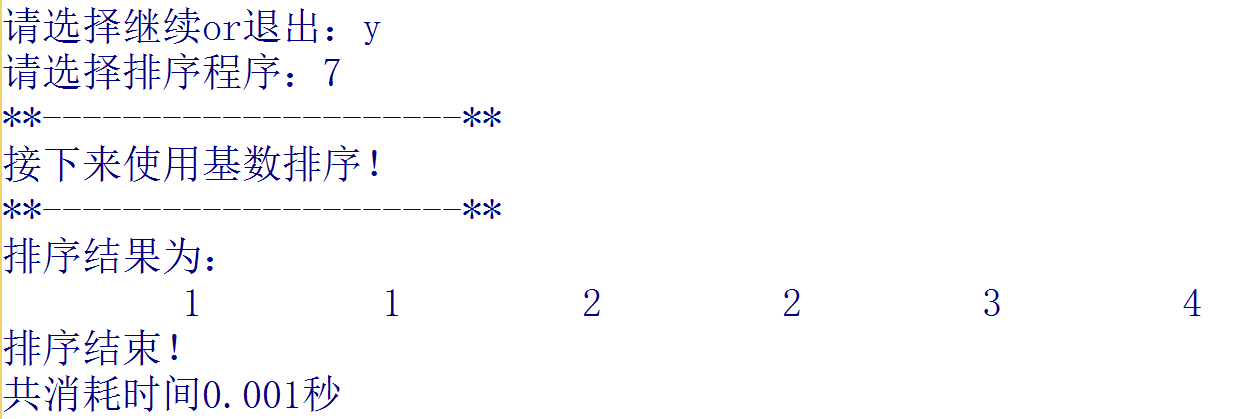
使用选择排序：



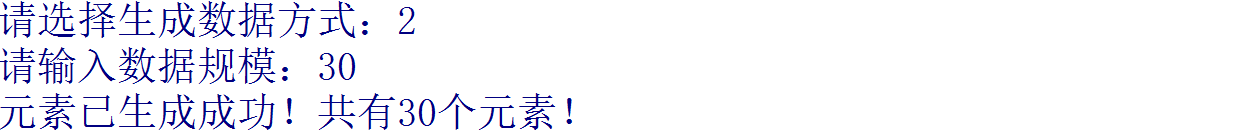
然后使用堆排序：



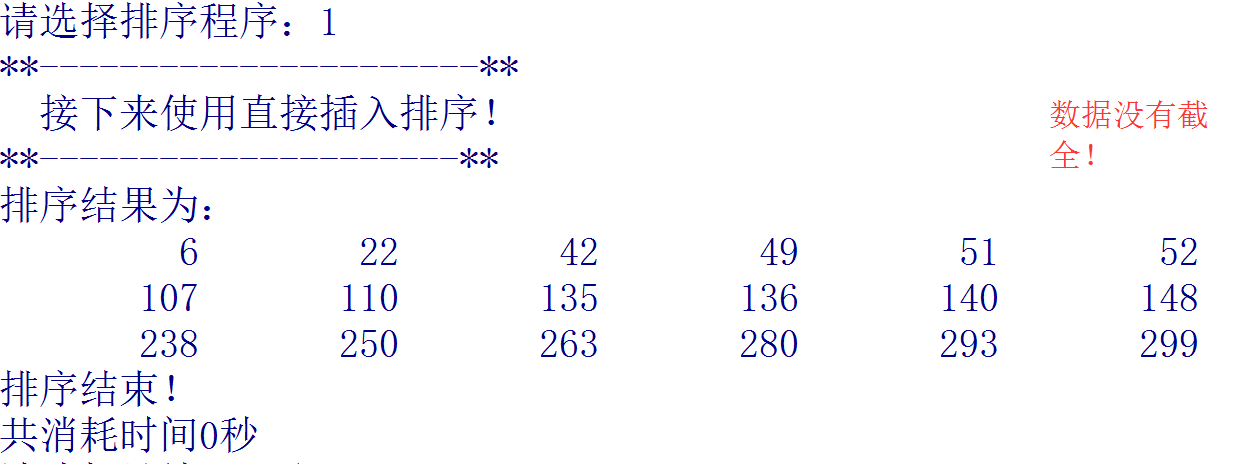
最后使用基数排序：



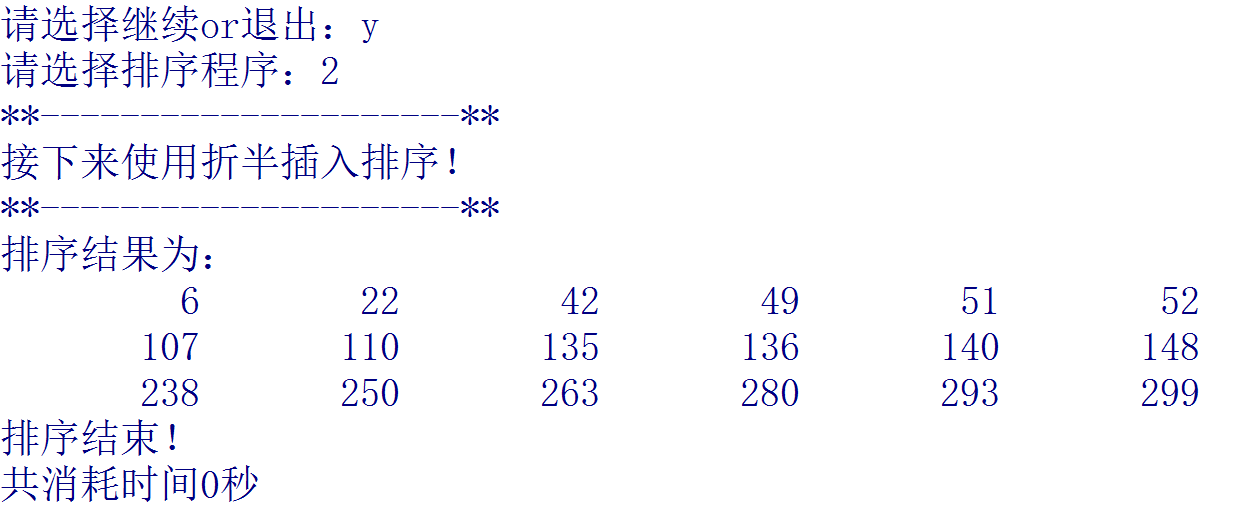
接下来使用第二种生成数据方式：



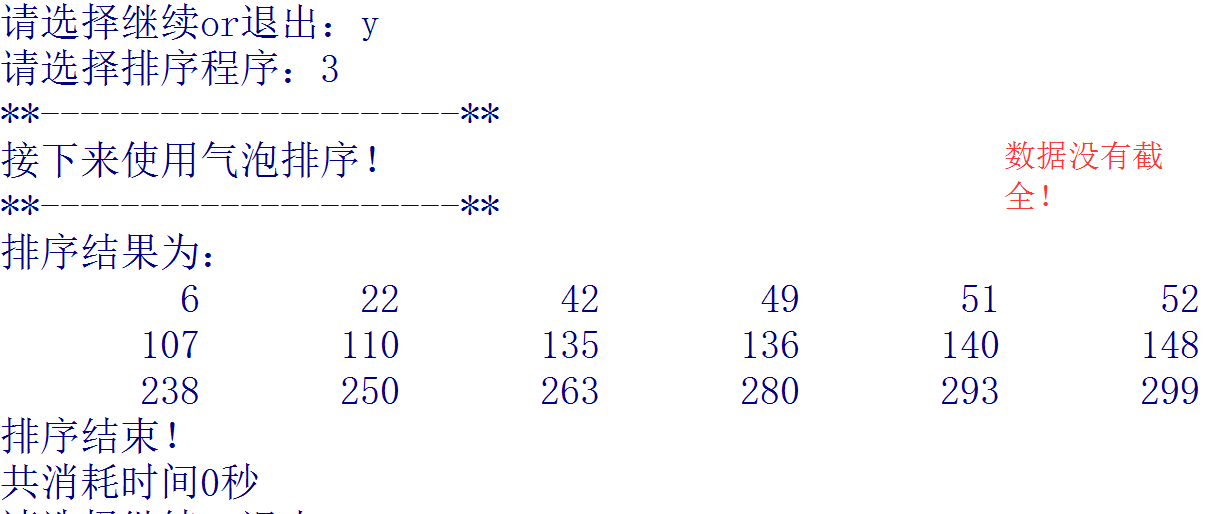
接下来使用直接插入排序：



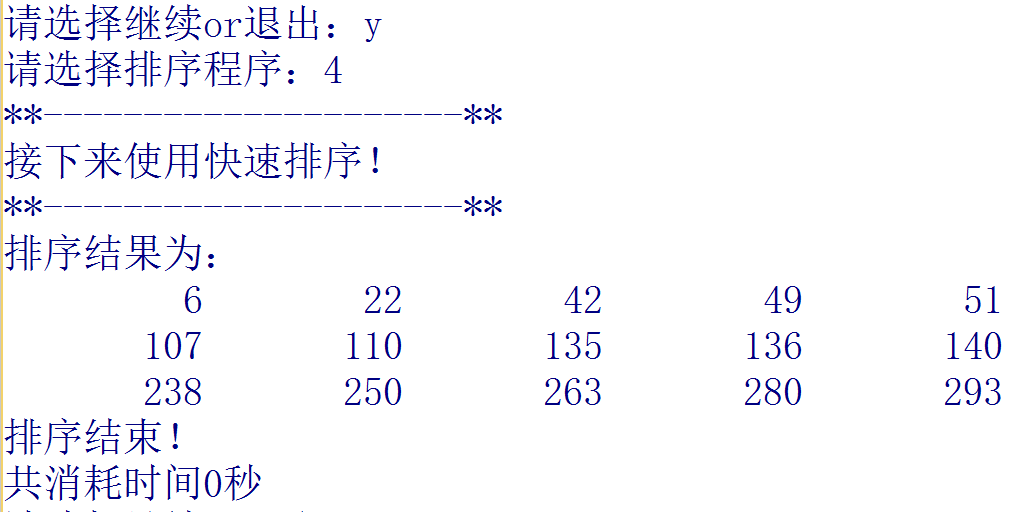
接下来使用折半插入排序：



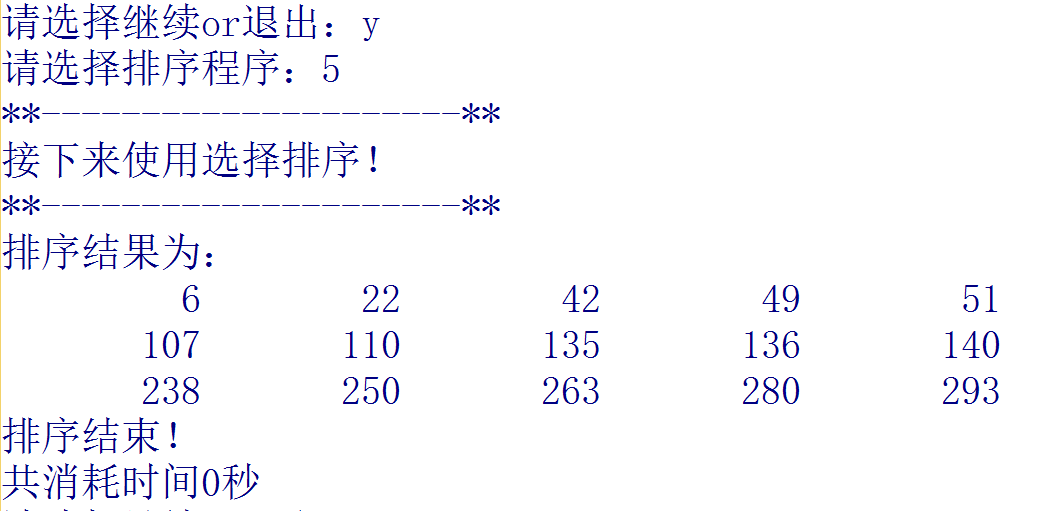
再使用气泡排序：



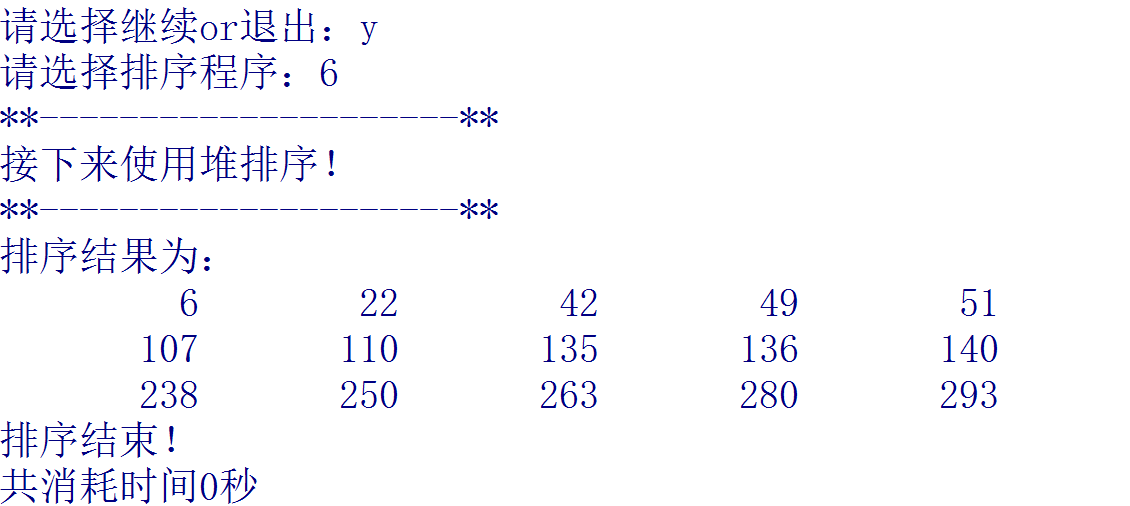
快速排序如下：



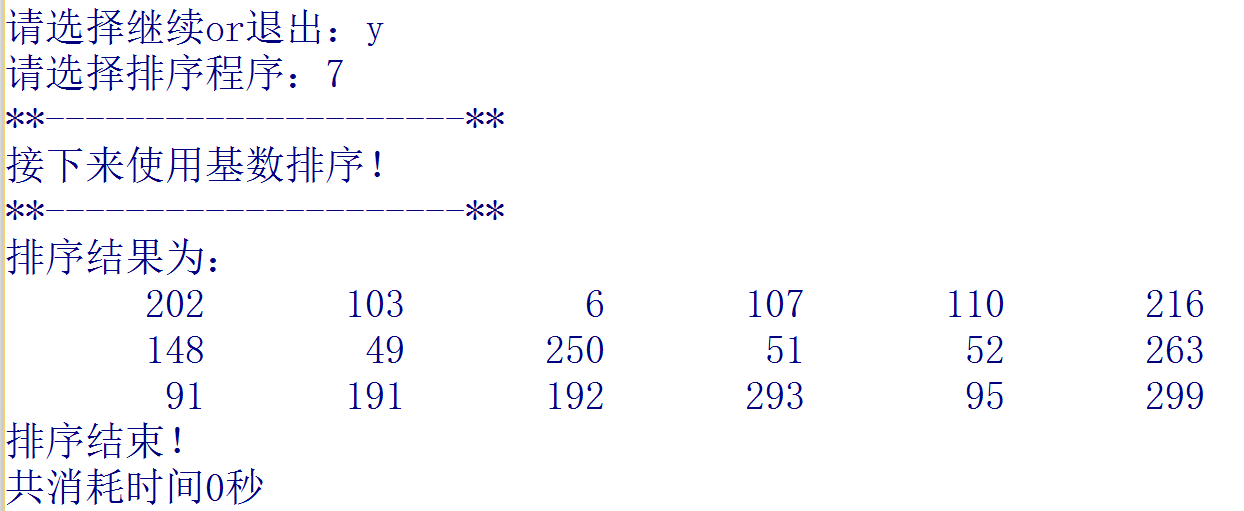
选择排序如下：



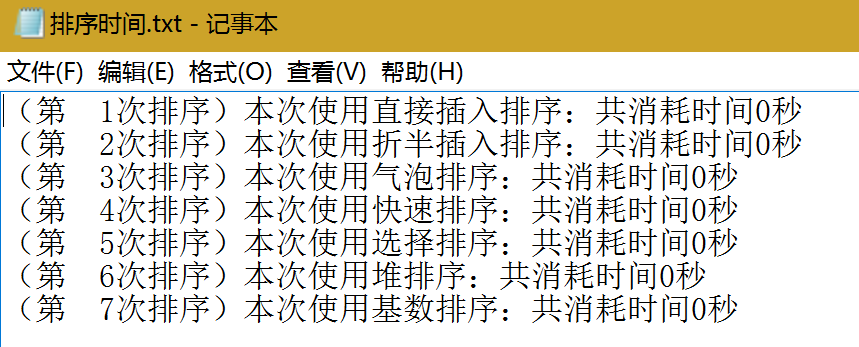
接下来使用堆排序：



最后使用基数排序：



（以上7个截图数据均不全！！只是为了表示我的程序有这个功能，由于屏幕大小限制只能截取部分，望老师谅解）

生成文件：  


由于数据量小，时间均为0

六、系统不足与经验体会

系统不足：

（1）界面不够友好；

（2）实现的功能太少；

（3）只能排大于0的数；

（4）在基数排序时最大位数没有按要求。

经验体会：

（1）程序只有自己写，才能体会美妙；

（2）程序只会越写越通。

七、附录：源代码（带注释）

#include <iostream>

#include <ctime>

#include <iomanip>

#include <cmath>

#include <queue>

#include <fstream>

#define MAX 1000000

typedef int keytype;

using namespace std;

//排序表的数据结构

typedef struct records {

keytype key;

}LIST[MAX];

LIST A;

clock\_t start, stop;

double duration;

//实现交换

void swap(records &x, records &y) {

records temp;

temp = x;

x = y;

y = temp;

}

//排序程序1，直接插入排序

void InsertSort(int n, LIST A) {

int i, j;

for (i = 1; i <= n; i++) {

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

if (A[j].key<A[j - 1].key) swap(A[j], A[j - 1]);

}

}

}

//折半查找待插入位置

int BinSearch(LIST A, int low, int high, keytype k) {

if (low>high) return low;

else {

int mid = (low + high) / 2;

if (k<A[mid].key) return BinSearch(A, low, mid - 1, k);

else if (k>A[mid].key) return BinSearch(A, mid + 1, high, k);

else return mid;

}

}

//排序程序2，折半插入排序

void BinInsertSort(int n, LIST A) {

A[0].key = -1;

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

int j = BinSearch(A, 1, i, A[i].key);

keytype temp;

temp = A[i].key;

for (int k = i; k>j; k--) {

A[k].key = A[k - 1].key;

}

A[j].key = temp;

}

}

//排序程序3，气泡排序

void BubbleSort(int n, LIST A) {

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

for (int j = n; j >= i + 1; j--) {

if (A[j].key<A[j - 1].key) {

swap(A[j], A[j - 1]);

}

}

}

}

int FindPivot(int i, int j) {

keytype firstkey = A[i].key;

int k;

for (k = i + 1; k <= j; k++) {

if (A[k].key>firstkey) return k;

else if (firstkey>A[k].key) return i;

}

return 0;

}

int Partition(int i, int j, keytype pivot) {

int x, y;

do {

for (x = i; A[x].key<pivot; x++);

for (y = j; A[y].key >= pivot; y--);

if (x<y) swap(A[x], A[y]);

} while (x <= y);

return x;

}

//排序程序4，快排

void QuickSort(int i, int j) {

keytype pivot;

int k, pivotindex;

pivotindex = FindPivot(i, j);

if (pivotindex != 0) {

pivot = A[pivotindex].key;

k = Partition(i, j, pivot);

QuickSort(i, k - 1);

QuickSort(k, j);

}

}

//排序程序5，选择排序

void SelectSort(int n, LIST A) {

keytype lowkey;

int i, j, lowindex;

for (i = 1; i<n; i++) {

lowindex = i;

lowkey = A[i].key;

for (j = i + 1; j <= n; j++) {

if (A[j].key<lowkey) {

lowkey = A[j].key;

lowindex = j;

}

}

swap(A[i], A[lowindex]);

}

}

//大数上排

void PushUp(int first, int last) {

int r = first;

while (r <= last / 2) {

if (r == last / 2 && last % 2 == 0) {

if (A[r].key<A[2 \* r].key) {

swap(A[r], A[2 \* r]);

}

r = last;

}

else if (A[r].key<A[2 \* r].key && A[2 \* r].key >= A[2 \* r + 1].key) {

swap(A[r], A[2 \* r]);

r = 2 \* r;

}

else if (A[r].key<A[2 \* r + 1].key && A[2 \* r + 1].key>A[2 \* r].key) {

swap(A[r], A[2 \* r + 1]);

r = 2 \* r + 1;

}

else {

r = last;

}

}

}

//排序程序6，堆排

void HeapSort(int n, LIST A) {

int i;

for (i = n / 2; i >= 1; i--) {

PushUp(i, n);

}

for (i = n; i >= 2; i--) {

swap(A[1], A[i]);

PushUp(1, i - 1);

}

}

int Radix(int k, int p) {

int power = 1;

for (int i = 1; i <= p - 1; i++) {

power = power \* 10;

}

return (k % (power \* 10)) / power;

}

//找figure

int FindFigure(int n) {

int figure = 1;

for (int i = 1; i<100; i++) {

if (pow(10, i) <= n && pow(10, i + 1)>n) return i + 1;

}

return figure;

}

//排序程序7，基数排序

void RadixSort(int n, LIST A) {

int figure = FindFigure(n);

queue<records> P;

queue<records> Q[10];

records data;

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

P.push(A[i]);

}

for (int pass = 1; pass <= figure; pass++) {

while (!P.empty()) {

data = P.front();

P.pop();

int r = Radix(data.key, pass);

Q[r].push(data);

}

for (int i = 0; i <= 9; i++) {

while (!Q[i].empty()) {

data = Q[i].front();

Q[i].pop();

P.push(data);

}

}

}

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

A[i] = P.front();

P.pop();

}

}

//随机生成A[max]

void CreateNumber(int n) {

srand(n);

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

A[i].key = rand() % (10 \* n);

}

}

//输出A[max]

void PrintList(int n, LIST A) {

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

cout << right << setw(10) << A[i].key;

}

}

int main(int argc, const char \* argv[]) {

system("color f1");

cout << "\*\*-------------------------------------------------------\*\*" << endl;

cout << " 本程序能实现如下功能：" << endl;

cout << " 1--直接插入排序" << endl;

cout << " 2--折半插入排序" << endl;

cout << " 3--气泡排序" << endl;

cout << " 4--快速排序" << endl;

cout << " 5--选择排序" << endl;

cout << " 6--堆排序" << endl;

cout << " 7--基数排序" << endl;

cout << "\*\*-------------------------------------------------------\*\*" << endl;

cout << "\*\*-------------------------------------------------------\*\*" << endl;

cout << " 本程序接受如下元素生成方式：" << endl;

cout << " 1--手动输入(以0标志结尾！)" << endl;

cout << " 2--系统随机生成" << endl;

cout << "\*\*-------------------------------------------------------\*\*" << endl;

int size,way,num,i=0;

cout << "请选择生成数据方式：";

cin >> way;

while (way != 1 && way != 2) {

cout << "只接受1、2两种生成元素方式！" << endl;

cout << "请重新选择：";

cin >> way;

}

switch (way) {

case 1:

cout << "请输入待排序元素：";

cin >> num;

try {

while (num != 0) {

i++;

A[i].key = num;

cin >> num;

}

if (i == 0) throw 0;

}

catch (int e) {

cout << "无法对"<<e<<"个元素排序！" << endl;

system("pause");

return 0;

}

size = i;

break;

case 2:

cout << "请输入数据规模：";

cin >> size;

while (size == 0) {

cout << "本程序无法实现对0个元素排序！" << endl;

cout << "请重新输入数据规模：";

cin >> size;

}

CreateNumber(size);

break;

default:

break;

}

cout << "元素已生成成功！共有" << size << "个元素！" << endl;

ofstream output("A.txt",ios::ate);

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

output <<right<<setw(10)<<A[i].key;

if (i % 15 == 0) output << endl;

}

output << endl<<"排序前元素已全部写入！"<<endl;

output.close();

bool flag = false;

int times=0;

ofstream output3("排序时间.txt",ios::app);

do {

cout << "请选择排序程序：";

int choose;

cin >> choose;

output << "本次排序元素个数"<<size<<"个";

output3 << "（第"<<setw(3)<<++times<<"次排序）";

switch (choose) {

case 1:

cout << "\*\*----------------------\*\*" << endl;

cout << " 接下来使用直接插入排序！" << endl;

cout << "\*\*---------------------\*\*" << endl;

output3 << "本次使用直接插入排序：";

start = clock();

InsertSort(size, A);

stop = clock();

break;

case 2:

cout << "\*\*---------------------\*\*" << endl;

cout << "接下来使用折半插入排序！" << endl;

cout << "\*\*---------------------\*\*" << endl;

output3 << "本次使用折半插入排序：";

start = clock();

BinInsertSort(size, A);

stop = clock();

break;

case 3:

cout << "\*\*---------------------\*\*" << endl;

cout << "接下来使用气泡排序！" << endl;

cout << "\*\*---------------------\*\*" << endl;

output3 << "本次使用气泡排序：";

start = clock();

BubbleSort(size, A);

stop = clock();

break;

case 4:

cout << "\*\*---------------------\*\*" << endl;

cout << "接下来使用快速排序！" << endl;

cout << "\*\*---------------------\*\*" << endl;

output3 << "本次使用快速排序：";

start = clock();

QuickSort(1, size);

stop = clock();

break;

case 5:

cout << "\*\*---------------------\*\*" << endl;

cout << "接下来使用选择排序！" << endl;

cout << "\*\*---------------------\*\*" << endl;

output3 << "本次使用选择排序：";

start = clock();

SelectSort(size, A);

stop = clock();

break;

case 6:

cout << "\*\*---------------------\*\*" << endl;

cout << "接下来使用堆排序！" << endl;

cout << "\*\*---------------------\*\*" << endl;

output3 << "本次使用堆排序：";

start = clock();

HeapSort(size, A);

stop = clock();

break;

case 7:

cout << "\*\*---------------------\*\*" << endl;

cout << "接下来使用基数排序！" << endl;

cout << "\*\*---------------------\*\*" << endl;

output3 << "本次使用基数排序：";

start = clock();

RadixSort(size, A);

stop = clock();

break;

default:

break;

}

cout << "排序结果为：" << endl;

PrintList(size, A);

duration = (static\_cast <double>(stop - start)) / CLK\_TCK;

cout << endl << "排序结束！" << endl;

cout << "共消耗时间" << setprecision(20) << duration << "秒" << endl;

output3<< "共消耗时间" << setprecision(20) << duration << "秒" << endl;

output.open("A.txt",ios::app);

output << "（第" << setw(3) << times << "次排序）"<<endl;

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

output << right << setw(10) << A[i].key;

if (i % 15 == 0) output << endl;

}

output << endl << "排序后元素写入完毕！" << endl;

output.close();

records number;

ifstream output2;

output2.open("A.txt",ios::in);

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

output2 >> A[i].key;

}

output2.close();

cout << "请选择继续or退出：" ;

char ch;

cin >> ch;

while (ch!='Y'&&ch!='y' && ch!='N' && ch!='n') {

cout << "只接受‘Y’‘y’‘N’‘n’四个字母！" << endl;

cout << "请重新选择：";

cin>>ch;

}

if (ch == 'y' || ch == 'Y') flag = true;

else if (ch == 'N' || ch == 'n') flag = false;

} while (flag);

system("pause");

return 0;

}

**题目要求的基数排序.cpp**

**#include <iostream>**

**#include <ctime>**

**#include <iomanip>**

**#include <cmath>**

**#include <queue>**

**#include <fstream>**

**#define MAX 1000000**

**typedef int keytype;**

**using namespace std;**

**//排序表的数据结构**

**typedef struct records {**

**keytype key;**

**}LIST[MAX];**

**LIST A;**

**clock\_t start, stop;**

**double duration;**

**//实现交换**

**void swap(records &x, records &y) {**

**records temp;**

**temp = x;**

**x = y;**

**y = temp;**

**}**

**int Radix(int k, int p) {**

**int power = 1;**

**for (int i = 1; i <= p - 1; i++) {**

**power = power \* 10;**

**}**

**return (k % (power \* 10)) / power;**

**}**

**//排序程序7，基数排序**

**void RadixSort(int n, LIST A) {**

**int figure = 3;**

**queue<records> P;**

**queue<records> Q[10];**

**records data;**

**for (int i = 1; i <= n; i++) {**

**P.push(A[i]);**

**}**

**for (int pass = 1; pass <= figure; pass++) {**

**while (!P.empty()) {**

**data = P.front();**

**P.pop();**

**int r = Radix(data.key, pass);**

**Q[r].push(data);**

**}**

**for (int i = 0; i <= 9; i++) {**

**while (!Q[i].empty()) {**

**data = Q[i].front();**

**Q[i].pop();**

**P.push(data);**

**}**

**}**

**}**

**for (int i = 1; i <= n; i++) {**

**A[i] = P.front();**

**P.pop();**

**}**

**}**

**//随机生成A[max]**

**void CreateNumber(int n) {**

**srand(n);**

**for (int i = 1; i <= n; i++) {**

**A[i].key = rand() % 1000;**

**}**

**}**

**//输出A[max]**

**void PrintList(int n, LIST A) {**

**for (int i = 1; i <= n; i++) {**

**cout << right << setw(10) << A[i].key;**

**}**

**}**

**int main(int argc, const char \* argv[]) {**

**system("color f1");**

**cout << "\*\*-------------------------------------------------------\*\*" << endl;**

**cout << " 本程序接受如下元素生成方式：" << endl;**

**cout << " 1--手动输入(以0标志结尾！仅接受1--999)" << endl;**

**cout << " 2--系统随机生成" << endl;**

**cout << "\*\*-------------------------------------------------------\*\*" << endl;**

**int size, way, num, i = 0;**

**bool flag = false;**

**do {**

**cout << "请选择生成数据方式：";**

**cin >> way;**

**while (way != 1 && way != 2) {**

**cout << "只接受1、2两种生成元素方式！" << endl;**

**cout << "请重新选择：";**

**cin >> way;**

**}**

**switch (way) {**

**case 1:**

**cout << "请输入待排序元素：";**

**cin >> num;**

**try {**

**while (num != 0) {**

**i++;**

**A[i].key = num;**

**cin >> num;**

**}**

**if (i == 0) throw 0;**

**}**

**catch (int e) {**

**cout << "无法对" << e << "个元素排序！" << endl;**

**system("pause");**

**return 0;**

**}**

**size = i;**

**break;**

**case 2:**

**cout << "请输入数据规模：";**

**cin >> size;**

**while (size == 0) {**

**cout << "本程序无法实现对0个元素排序！" << endl;**

**cout << "请重新输入数据规模：";**

**cin >> size;**

**}**

**CreateNumber(size);**

**break;**

**default:**

**break;**

**}**

**cout << "元素已生成成功！共有" << size << "个元素！" << endl;**

**cout << "\*\*---------------------\*\*" << endl;**

**cout << "接下来使用基数排序！" << endl;**

**cout << "\*\*---------------------\*\*" << endl;**

**start = clock();**

**RadixSort(size, A);**

**stop = clock();**

**cout << "排序结果为：" << endl;**

**PrintList(size, A);**

**duration = (static\_cast <double>(stop - start)) / CLK\_TCK;**

**cout << endl << "排序结束！" << endl;**

**cout << "共消耗时间" << setprecision(20) << duration << "秒" << endl;**

**cout << "请选择继续or退出：";**

**char ch;**

**cin >> ch;**

**while (ch != 'Y'&&ch != 'y' && ch != 'N' && ch != 'n') {**

**cout << "只接受‘Y’‘y’‘N’‘n’四个字母！" << endl;**

**cout << "请重新选择：";**

**cin >> ch;**

**}**

**if (ch == 'y' || ch == 'Y') flag = true;**

**else if (ch == 'N' || ch == 'n') flag = false;**

**} while (flag);**

**system("pause");**

**return 0;**

**}**