**第三部分 第2题：多关键字排序**

**实验报告**

题目：设计一个多关键字排序的程序处理系统。

班级：F1703407 姓名：赵伟基 学号：517021910883 完成日期：2018-12-16

1. **需求分析**

多关键字排序有一定的实用范围，在进行高考分数处理时，除了对总分进行排序外，不同专业对单科分数要求不同，需要在总分相同的情况下按照用户提出的单科分数次序要求排出考生录取次序。

1. 成绩记录汇总表的记录数不超过10000，考试科目共6们，每门成绩范围为0-100。成绩由随机数产生器产生。
2. 从高到低输入单科优先次序，先依照高考成绩综合从高到低排序，相同总分的按单科有限次序进行从高到低排序。
3. **概要设计**

题目要求分两部分，第一部分是产生指定学生数的成绩数据，第二部分是对这些数据进行排序。为此定义两个函数createdata()与BubbleSort()，为了提高算法的时间和空间效率，选择以有序链表来存储集合，为此需要定义学生成绩类Score，此外外加两个功能函数打印成绩表PrintScore()与释放空间函数clearscore()。

1. 主程序模块

int main()

{

初始化；

单科优先次序输入；

createdata();

BubbleSort();

PrintScore();

clearscore();

}

1. 学生数据结点结构体

Struct Score

{

成绩数据数组a[];

Score \*next；

}

1. 功能函数模块：
2. createdata（）：

初始条件：无

返回结果：创建空链表；返回一个学生成绩数据表的头指针；

1. BubbleSort ( ):

初始条件：输入一个学生成绩数据表的头指针；

操作过程：利用冒泡排序算法，依照先总分后单科优先次序的顺序对链表进行排序，返回排序完的学生成绩数据表的头指针；

1. PrintScore():

初始条件：输入一个学生成绩数据表的头指针；

返回结果：打印该学生成绩数据表；

1. clearscore()

初始条件：输入一个学生成绩数据表头指针；

返回结果：释放该表（链表）的空间。

1. **详细设计**

（1）main.cpp

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <iomanip>

using namespace std;

struct Score

{

int key[7]; //1-6为科目 0为总分

Score \*next;

};

void PrintScore(Score \*L); //打印函数

Score\* createdata(); //成绩表生成函数

Score\* BubbleSort(Score \*L,int a[]);//排序函数

void clearscore(Score \*L); //释放空间

int n; //数据条数

int main()

{

Score \*L,\*q;

int a[7];

a[0]=0;

double time;

cout<<"请输入单科优先次序"<<endl;

for(int i=1;i<7;i++)

{

cin>>a[i];

}

L=createdata();

BubbleSort(L,a);

cout<<endl;

cout<<"排序后的次序为"<<endl;

cout<<setw(8)<<"总分"<<setw(8)<<"科目1"<<setw(8)<<"科目2"<<setw(8)<<"科目3"<<setw(8)<<"科目4"<<setw(8)<<"科目5"<<setw(8)<<"科目6"<<endl;

PrintScore(L);

clearscore(L);

return 0;

}

Score\* createdata()

{

Score\*p,\*q,\*head;

p=head=new Score;

/\* cout<<p<<endl;

cout<<L->next<<endl;\*/

srand(time(0));//撒种

cout<<"请输入你所需要的数据个数，一个学生视为一条数据"<<endl;

cin>>n;

cout<<setw(8)<<"总分"<<setw(8)<<"科目1"<<setw(8)<<"科目2"<<setw(8)<<"科目3"<<setw(8)<<"科目4"<<setw(8)<<"科目5"<<setw(8)<<"科目6"<<endl;

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

{

p=p->next=new Score;

//p=new Score;

p->key[0]=0;

for (int j=1;j<7;j++)

{

p->key[j]=rand()%101;//生成0~100的随机数

p->key[0]=p->key[0]+p->key[j];

}

for(int k=0;k<7;k++)

{

cout<<setw(8)<<p->key[k];

}

cout<<endl;

}

p->next=NULL;

return head;

}

void clearscore(Score \*L)

{

Score \*p,\*temp;

p=L->next;

while(p!=NULL)

{

temp=p;

p=p->next;

delete temp;

}

L->next=NULL;

}

Score\* BubbleSort(Score \*L,int a[])

{

Score \*p;

int temp=0;

p=L->next;

//排序实现

for (int k=0;k<n;k++)

{

while(p->next!=NULL)

{

if(p->key[a[0]]<p->next->key[a[0]])//总分排序

{

for(int j=0;j<7;j++)

{

temp=p->key[j];

p->key[j]=p->next->key[j];

p->next->key[j]=temp;

}

p=p->next;

}

else if(p->key[a[0]]==p->next->key[a[0]]&&p->key[a[1]]<p->next->key[a[1]])//优先级最高的科目排序

{

for(int j=0;j<7;j++)

{

temp=p->key[j];

p->key[j]=p->next->key[j];

p->next->key[j]=temp;

}

p=p->next;

}

else if(p->key[a[0]]==p->next->key[a[0]]&&p->key[a[1]]==p->next->key[a[1]]&&p->key[a[2]]<p->next->key[a[2]])

{

for(int j=0;j<7;j++)

{

temp=p->key[j];

p->key[j]=p->next->key[j];

p->next->key[j]=temp;

}

p=p->next;

}

else if(p->key[a[0]]==p->next->key[a[0]]&&p->key[a[1]]==p->next->key[a[1]]&&p->key[a[2]]==p->next->key[a[2]]&&p->key[a[3]]<p->next->key[a[3]])

{

for(int j=0;j<7;j++)

{

temp=p->key[j];

p->key[j]=p->next->key[j];

p->next->key[j]=temp;

}

p=p->next;

}

else if(p->key[a[0]]==p->next->key[a[0]]&&p->key[a[1]]==p->next->key[a[1]]&&p->key[a[2]]==p->next->key[a[2]]&&p->key[a[3]]==p->next->key[a[3]]&&p->key[a[4]]<p->next->key[a[4]])

{

for(int j=0;j<7;j++)

{

temp=p->key[j];

p->key[j]=p->next->key[j];

p->next->key[j]=temp;

}

p=p->next;

}

else if(p->key[a[0]]==p->next->key[a[0]]&&p->key[a[1]]==p->next->key[a[1]]&&p->key[a[2]]==p->next->key[a[2]]&&p->key[a[3]]==p->next->key[a[3]]&&p->key[a[4]]==p->next->key[a[4]]&&p->key[a[5]]<p->next->key[a[5]])

{

for(int j=0;j<7;j++)

{

temp=p->key[j];

p->key[j]=p->next->key[j];

p->next->key[j]=temp;

}

p=p->next;

}

else if(p->key[a[0]]==p->next->key[a[0]]&&p->key[a[1]]==p->next->key[a[1]]&&p->key[a[2]]==p->next->key[a[2]]&&p->key[a[3]]==p->next->key[a[3]]&&p->key[a[4]]==p->next->key[a[4]]&&p->key[a[5]]==p->next->key[a[5]]&&p->key[a[6]]<p->next->key[a[6]])

{

for(int j=0;j<7;j++)

{

temp=p->key[j];

p->key[j]=p->next->key[j];

p->next->key[j]=temp;

}

p=p->next;

}

else p=p->next;

}

p=L->next;

}

return L;

}

void PrintScore(Score \*L)

{

Score \*q;

q=L->next;

while(q)

{

for(int k=0;k<7;k++)

{

cout<<setw(8)<<q->key[k];

}

cout<<endl;

q=q->next;

}

1. **调试分析**
2. 该程序将createdata()中的变量n（数据条数）设定为全局变量，数据的安全性缺乏保证
3. 该算法采取冒泡排序法对数据进行排序，效率较高，同时使用线性表作为数据存储的载体，节省空间，效率较高。
4. 算法复杂度分析

（1）：时间复杂度

①O（n^2）: PrintScore()：依次遍历、打印数据。

Createdata():按照输入的数据条数进行划分空间

clearscore()：遍历删除

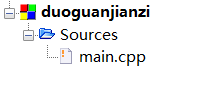
②O(n^3): BubbleSort() ：实现冒泡排序

（2）：空间复杂度

O(n^2):n条数据，每条数据有6个科目的内容。

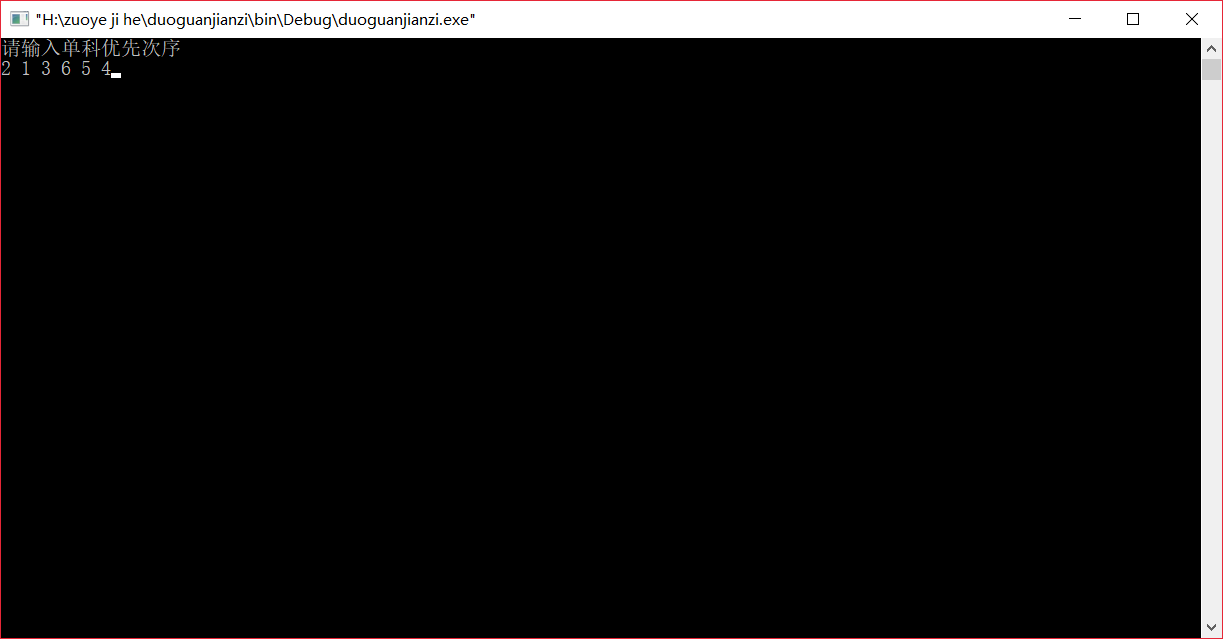
1. **用户手册**

1．本程序使用的 Code::Blocks 16.01 IDE，程序以项目（project）方式组织，如图 1 所示



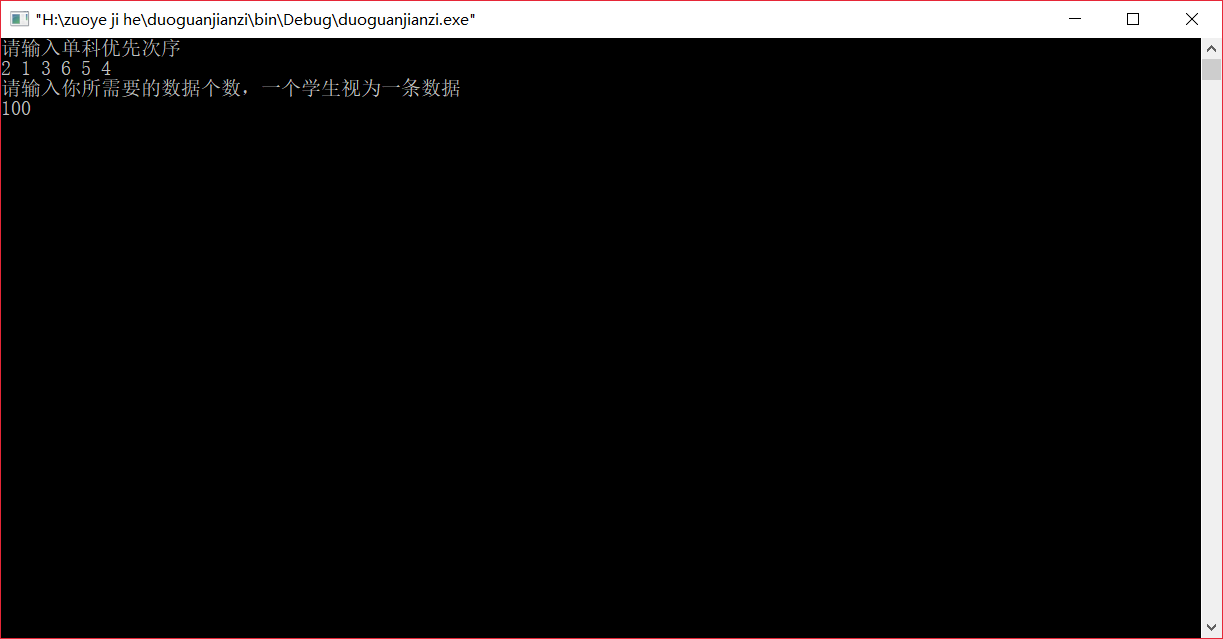
2． 用户点击main.cpp->build and run->依照交互提示输入数据->自动弹出按要求排序后的内容。

1. **测试结果**
2. 输入单科优先次序

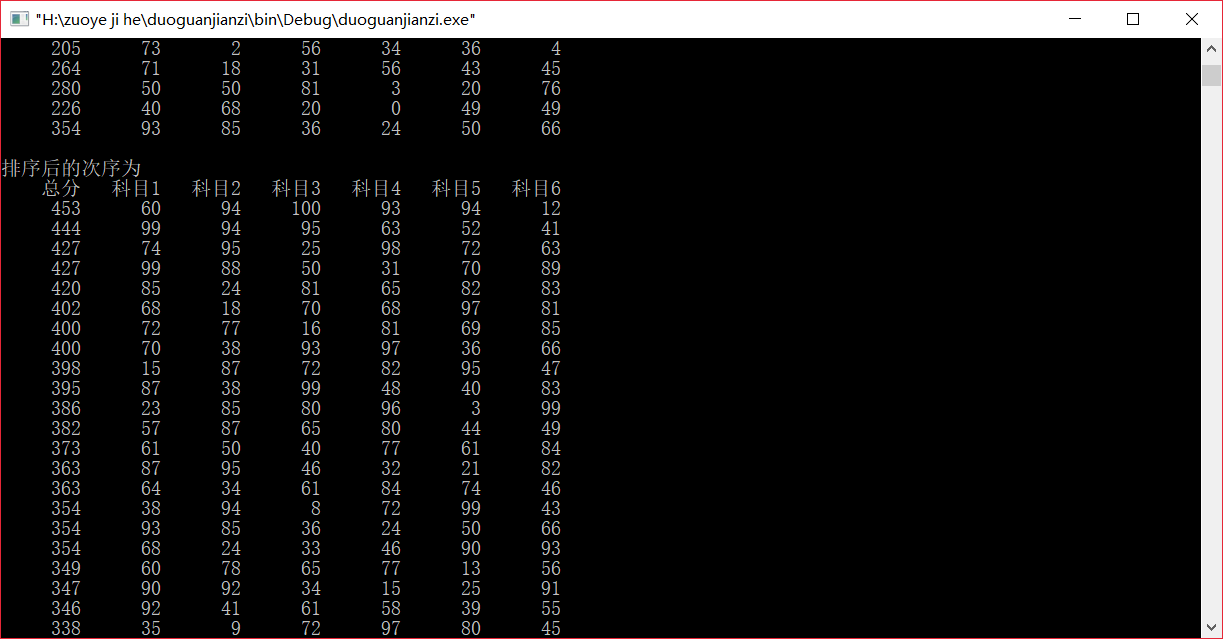


如图所示，表示优先级为：科目2>科目1>科目3>科目6>科目5>科目4

（2） 输入所需的学生数据数



（3） 按下回车，自动排序



1. **附录**

main.cpp //主函数 功能函数模块