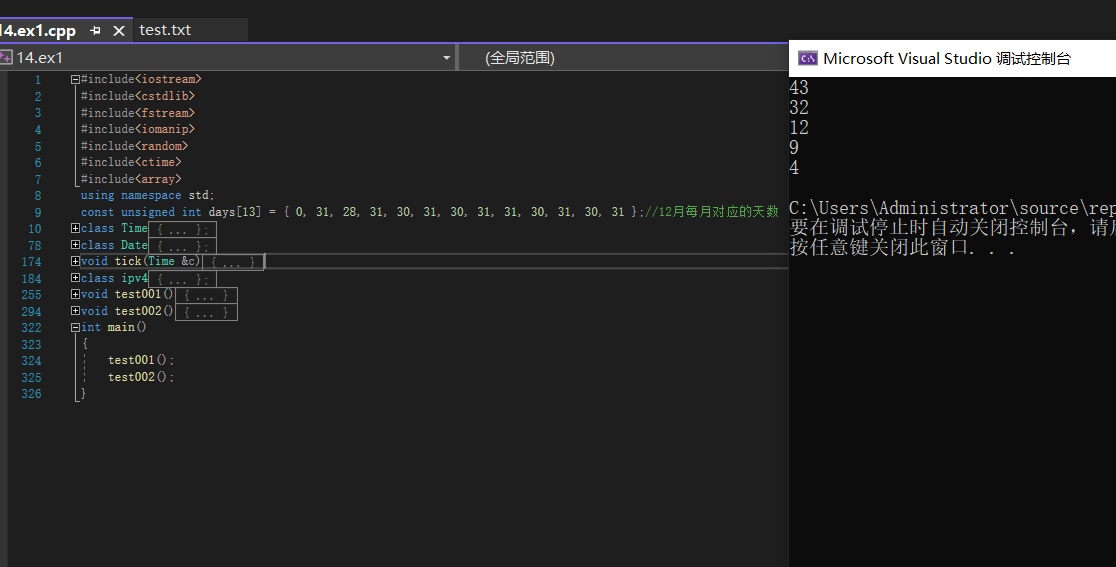
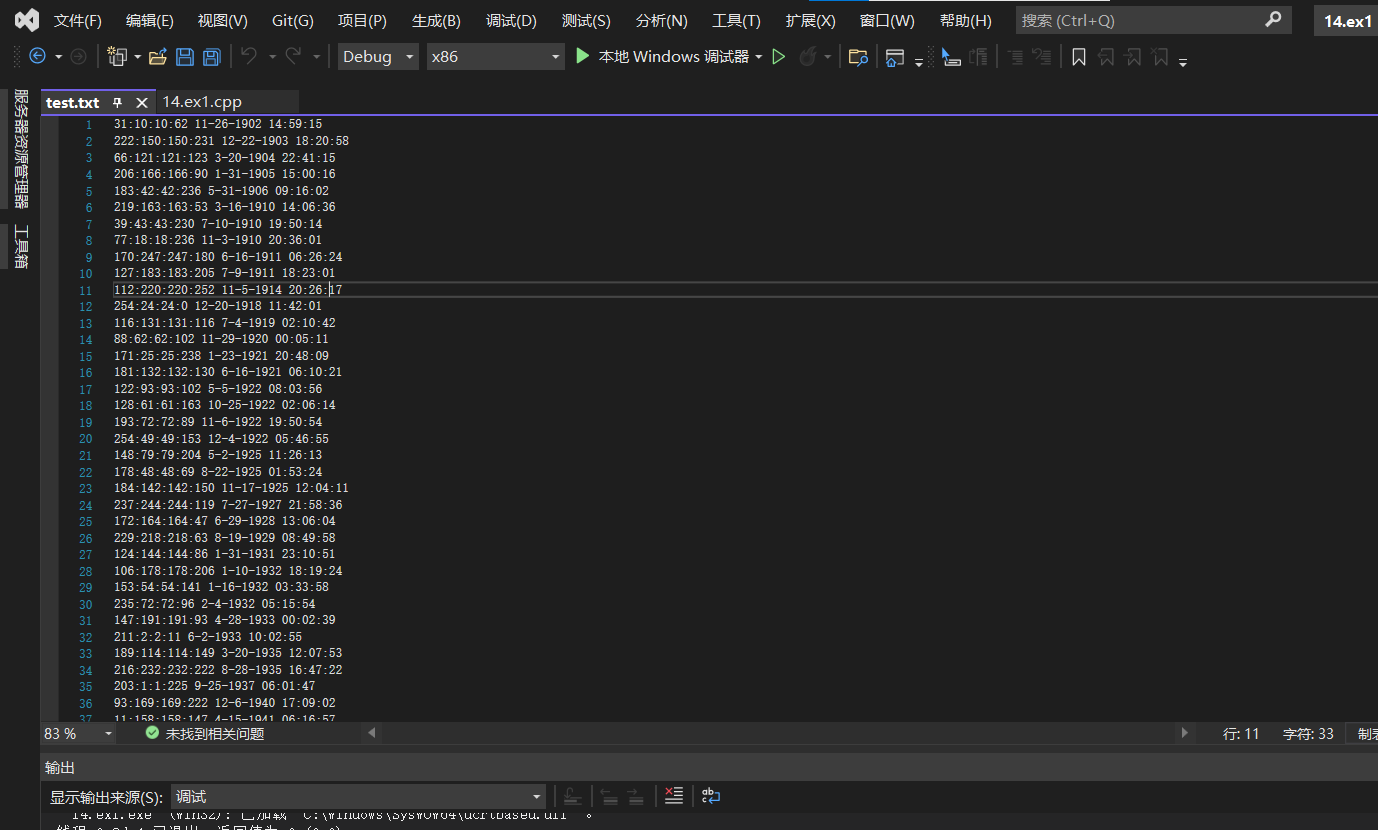
# 14.ex1



资源txt



#include<iostream>

#include<cstdlib>

#include<fstream>

#include<iomanip>

#include<random>

#include<ctime>

#include<array>

using namespace std;

const unsigned int days[13] = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };//12月每月对应的天数

class Time//时间类 时：分：秒

{

public:

Time(const int h = 0, const int m = 0, const int s = 0)

{//默认构造函数，自动赋值为0：0：0

setTime(h, m, s);//调用构造函数

}

void setTime(const int h, const int m, const int s)

{//赋值函数（三个一起）

setHour(h);//时赋值函数

setMinute(m);//分赋值函数

setSecond(s);//秒赋值函数

}

void setHour(const int h)//时赋值函数

{

if (h >= 0 && h < 24)//属于正常的时间（0-23）

hour = h;

else//弹出报错

throw invalid\_argument("hour must be 0-23");//返回报错

}

void setMinute(const int m)

{

if (m >= 0 && m < 60)//属于正常的时间（0-59）

minute = m;

else//弹出报错

throw invalid\_argument("minute must be 0-59");//返回报错

}

void setSecond(const int s)

{

if (s >= 0 && s < 60)//属于正常的时间（0-59）

second = s;

else//弹出报错

throw invalid\_argument("second must be 0-59");

}

unsigned int getHour() const

{//返回值函数，返回小时

return hour;

}

unsigned int getMinute() const

{//返回值函数，返回分钟

return minute;

}

unsigned int getSecond() const

{//返回值函数，返回秒

return second;

}

virtual void print() const

{//标准-输出函数。输出各项信息(可重载）

cout << setfill('0') << setw(2) << getHour() << ":" << setw(2) << getMinute() << ":" << setw(2) << getSecond();

}

virtual void print1(ofstream &a) const

{//标准-输出函数。输出各项信息（可重载）

a << setfill('0') << setw(2) << getHour() << ":" << setw(2) << getMinute() << ":" << setw(2) << getSecond();

}

void printStandard() const

{//12小时制-输出函数

cout << ((getHour() == 0 || getHour() == 12) ? 12 : getHour() % 12) << ":" << setfill('0') << setw(2) << getMinute() << ":" << setw(2) << getSecond() << (hour < 12 ? " AM" : " PM") << endl;

}

virtual int getcompare()

{//这是个用于获取总秒数的比对函数

return hour \* 3600 + minute \* 60 + second;

}

friend void tick(Time &);//友元函数

private:

unsigned int hour; //0 - 23

unsigned int minute; //0- 59

unsigned int second; //0- 59

};

class Date :public Time//包含时间的日期类

{

friend std::ostream &operator<<(ostream &, const Date &);//经典输出友元函数

public:

Date(int m = 1, int d = 1, int y = 1900,const int h = 0, const int min = 0, const int s = 0)

{//初始化函数，初始天分与时间

setTime(h, min, s);//时分秒初始化函数

setDate(m, d, y);//年月天初始化函数

}//default constructor

void setDate(int mm, int dd, int yy)//年月天初始化函数

{

if (mm >= 1 && mm <= 12)//

month = mm;

else

throw invalid\_argument("Month must be 1-12");

if (yy >= 1900 && yy <= 2100)

year = yy;

else

throw invalid\_argument("Year must be>=1900 and<-2100");

// test for a leap year

if ((month == 2 && leapYear(year) && dd >= 1 && dd <= 29) || (dd >= 1 && dd <= days[month]))

day = dd;

else

throw invalid\_argument("Day is out of range for current month and year");

}// set month,day,year

Date &operator++()

{

helpIncrement();// increment date

return \*this;// reference return to create an lvalue

}//prefix increment operator

Date operator++(int)

{

Date temp = \*this;//hold current state of object

helpIncrement();//postfix increment operator

return temp;

}

Date &operator+=(unsigned int additiona1Days)

{

for (int i = 0; 1 < additiona1Days;++i)

helpIncrement();// add days,modify object

return \*this;

}

static bool leapYear(int testYear)

{

if (testYear % 400 == 0 || (testYear % 100 != 0 && testYear % 4 == 0))

return true; // a leap year

else

return false; // not a leap year

}// is date in a leap year?

bool endofMonth(int testDay)const

{

if (month == 2 && leapYear(year))

return testDay == 29;// last day of Feb. in leap year

else

return testDay == days[month];

}// is date at the end of month?

void helpIncrement()

{

//day is not end of month

if (!endofMonth(day))

++day; // increment day

else

if (month < 12)// day is end of month and month < 12

{

++month;// increment month

day = 1; // first day of new month

}//end if

else // 1ast day of year

{

++year; // increment year

month = 1; // first month of new year

day = 1; // first day of new month

}

}

int getmonthdate()const

{//返回月份

return month;

}

virtual void print()const

{//重载后的标准-输出函数

cout << month << "-" << day << "-" << year;

}

virtual void print1(ofstream &a)const

{//重载后的文件操作-输出函数

a << month << "-" << day << "-" << year;

}

virtual int getcompare()

{//重载后的返回比对值函数，不同的是，此值无意义

return year \* 10000 + month \* 100 + day;

}

private:

unsigned int month;

unsigned int day;

unsigned int year;

};

void tick(Time &c)//已经提前声明

{

int t;

t = 3600 \* c.hour + 60 \* c.minute + c.second;//将其转化为一天中的总秒数，无需if

t++;

t = t % 86400;//一天只有86400秒

c.hour = t / 3600;

c.second = t % 60;

c.minute = t % 3600 / 60;

}

class ipv4 :public Date//一个ipv4类

{

public:

ipv4(int i1=0,int i2=0,int i3=0,int i4=0, int m = 1, int d = 1, int y = 1900, const int h = 0, const int min = 0, const int s = 0)

{//初始构造函数

setTime(h, min, s);

setDate(m, d, y);

setipv4(i1, i2, i3, i4);

}

void setipv4(int i1 = 0, int i2 = 0, int i3 = 0, int i4 = 0)

{//设置IP，仅当0-255时可用

if (i1 >= 0 && i1 <= 255)

ip1 = i1;

else

ip1 = 0;

if (i2 >= 0 && i2 <= 255)

ip2 = i3;

else

ip2 = 0;

if (i4 >= 0 && i3 <= 255)

ip3 = i3;

else

ip1 = 0;

if (i4>= 0 && i4 <= 255)

ip4 = i4;

else

ip4 = 0;

}

void setip(int i1 = 0, int i2 = 0, int i3 = 0, int i4 = 0, int m = 1, int d = 1, int y = 1900, const int h = 0, const int min = 0, const int s = 0)

{//将其中所蕴含的12个数据全初始化一边

setTime(h, min, s);

setDate(m, d, y);

setipv4(i1, i2, i3, i4);

}

virtual void print()const

{//重载的标准-输出函数，输出该相对应的全部信息

cout << ip1 << ":" << ip2 << ":" << ip3 << ":" << ip4<<" ";

Date::print();

cout << " ";

Time::print();

cout << endl;

}

virtual void print1(ofstream &a)const

{//重载的文件操作-输出函数，输出该相对应的全部信息

a << ip1 << ":" << ip2 << ":" << ip3 << ":" << ip4 << " ";

Date::print1(a);

a << " ";

Time::print1(a);

a << endl;

}

bool operator>(ipv4& ip)

{//比对函数

int a1, a2;

a1 = Date::getcompare();

a2 = ip.Date::getcompare();

if (a1 != a2)//如果日期更大则直接结束判定

return ((a1 - a2) / fabs(a1 - a2)+1)/2;

else

{

a1 = Time::getcompare();//进行时间判定

a2 = ip.Time::getcompare();

if (a1 != a2)

return ((a1 - a2) / fabs(a1 - a2) + 1) / 2;

else

return false;

}

}

private:

int ip1, ip2, ip3, ip4;

};

void test001()

{

srand(time(0));//重载随机数

array<ipv4, 101>i\_p;//IP

int sor[101];

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

sor[i] = 0;

sor[1] = 1;

int side = 1;

int k;

ofstream here("test.txt", ios::out);//文件操作

for (int j = 1;j <= 100;j++)

{

const unsigned int num[11] = { 0,256,256,256,256,12,10000,120,24,60,60 };//对应的IP数据上界

unsigned int num1[11] = { 0,0,0,0,0,1,0,1901,0,0,0 };//对应的日期测试数据

int i;

for (i = 1;i <= 10;i++)

num1[i] += rand() % num[i];//随机生成日期，IP

num1[6] = num1[6] % ::days[num1[5]]+1;//月份日期，较为特殊

i\_p[j].setip(num1[1], num1[2], num1[3], num1[4], num1[5], num1[6], num1[7], num1[8], num1[9], num1[10]);//将其载入IP

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

{

if (i\_p[sor[k]] > i\_p[j])//把这一个以升序顺序载入IP数组

{

for (int p = j; p >k;p--)

sor[p] = sor[p-1];

sor[k] = j;

k = j + 1;

}

}

if (k == j)

sor[k] = k;

}

for (k = 1; k <= 100; k++)

{

i\_p[sor[k]].print1(here);//输出去

}

here.close();

}

void test002()

{

ifstream there("test.txt", ios::in);//读入

int sum[6] = { 0,0,0,0,0,0 };

int i = 1;

int ip1;

for (i = 0;i < 100;i++)

{

there >> ip1;//通过重载得到读入来读入并分类

there.ignore();//忽略 换行符

if (ip1 <= 127)//根据IP数据范围来统计数据

sum[1]++;

else if (ip1 <= 191)

sum[2]++;

else if (ip1 <= 223)

sum[3]++;

else if (ip1 <= 239)

sum[4]++;

else sum[5]++;

for (int j = 1; j <= 9; j++)

{

there >> ip1;

there.ignore();//再次读入一个IP

}

}

for (i = 1; i <= 5; i++)

cout << sum[i] << endl;

}

int main()

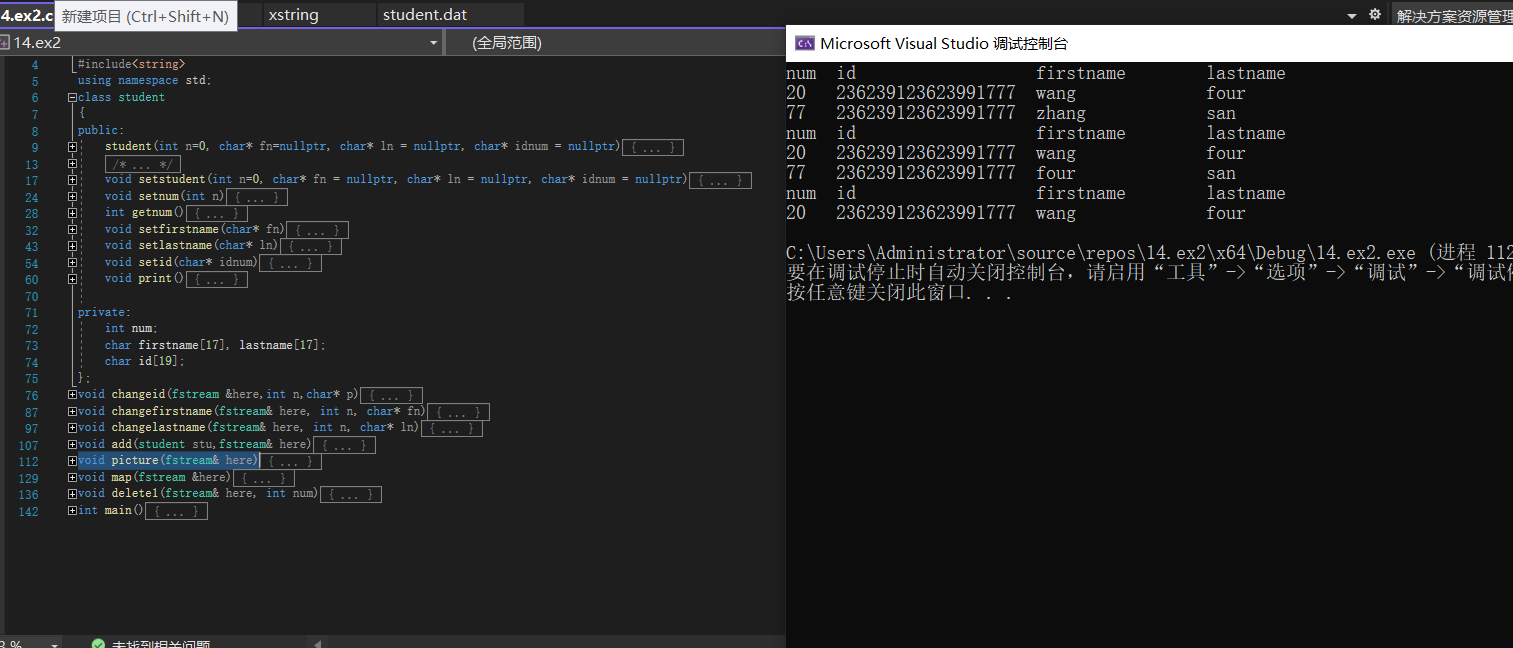
{

test001();

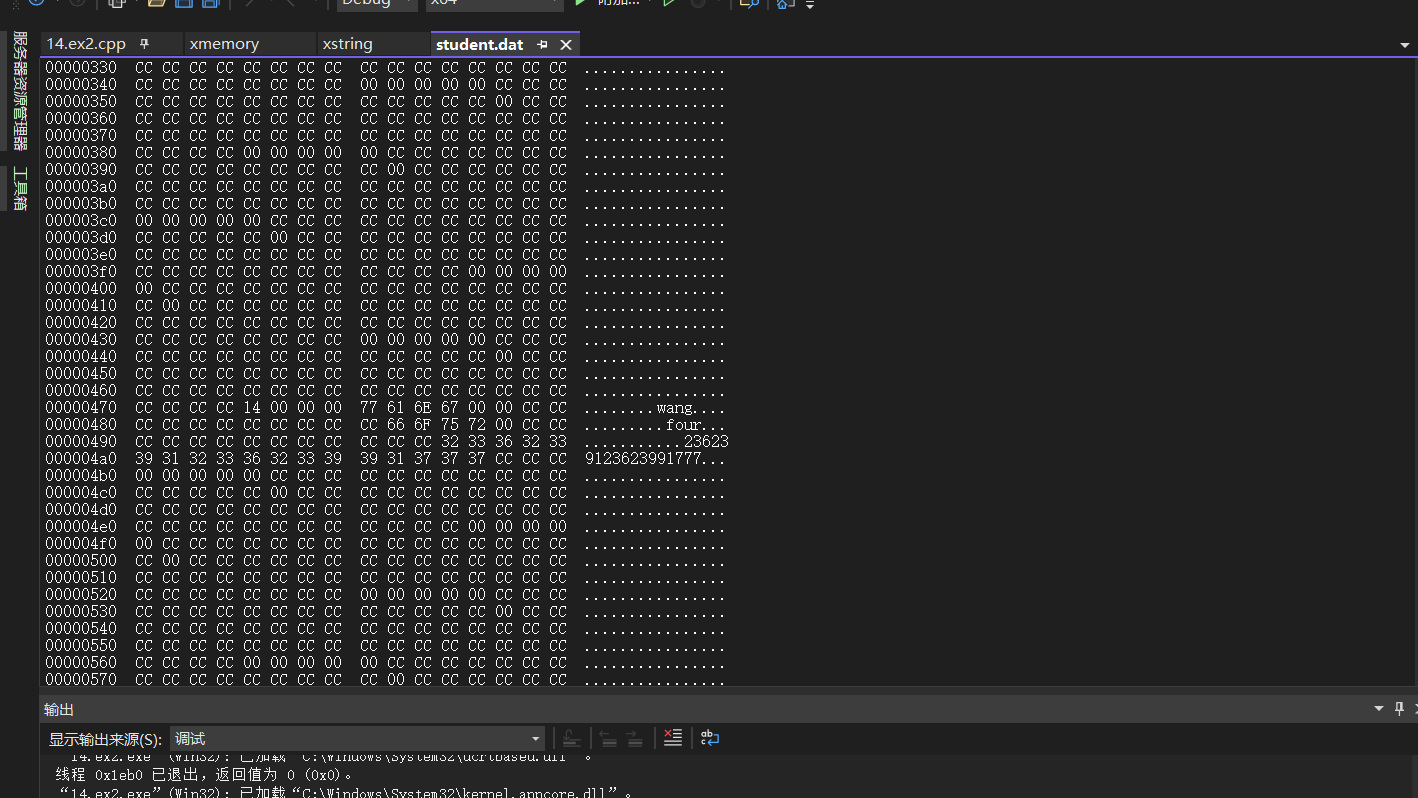
test002();

}

# 14.ex2



资源dat



#include<iostream>

#include<fstream>

#include<iomanip>

#include<string>

using namespace std;

class student

{

public:

student(int n=0, char\* fn=nullptr, char\* ln = nullptr, char\* idnum = nullptr)

{//构造函数，依次是学号，fistname，lastname，id

setstudent(n, fn, ln, idnum);

}

/\*~student()

{

}\*/

void setstudent(int n=0, char\* fn = nullptr, char\* ln = nullptr, char\* idnum = nullptr)

{//student的批量赋值函数

setnum(n);//依次调用各自的赋值函数

setfirstname(fn);

setlastname(ln);

setid(idnum);

}

void setnum(int n)//学号赋值函数

{

num = n;

}

int getnum()//学号返回函数

{

return num;

}

void setfirstname(char\* fn)//firstname赋值函数

{

int i = 0;

if(fn!=nullptr)//如果说有可复制的

while (i <= 16 && \*(fn + i) != '\0')

{//最多复制位

firstname[i] = \*(fn + i);

i++;

}

firstname[i] = '\0';

}

void setlastname(char\* ln)

{//lastname赋值函数

int i = 0;

if (ln != nullptr)//如果说有可复制的

while (i <= 16 && \*(ln + i) != '\0')//做多赋值16位

{

lastname[i] = \*(ln + i);

i++;

}

lastname[i] = '\0';

}

void setid(char\* idnum)

{//初始化18位的身份证号

if (idnum != nullptr)

for (int i = 1; i <= 18; i++)

id[i] = \*(idnum + i);

}

void print()//统一制表输出

{

cout.setf(ios::left);

cout << setw(5) << num;//编号

for (int i = 1; i <= 18; i++)

cout << id[i];//身份证号

cout << " ";

cout <<setw(17) << firstname<<setw(17)<<lastname;//主要是为了对齐

cout << endl;

}//num++id++++++++++++++++++firstname++++++++

private:

int num;

char firstname[17], lastname[17];

char id[19];

};

void changeid(fstream &here,int n,char\* p)//更改id

{

student stu(n);//声明一个空student

here.seekg((n - 1) \* sizeof(student));//跳转到n-1对应的学生开头

here.read(reinterpret\_cast<char\*>(&stu), (sizeof(student)));//读入对应长度的身份证号（注：这个是当时纠错用的）

stu.setid(p);//设置

here.seekp((n - 1) \* sizeof(student));//将put指针移动到哪里

here.write(reinterpret\_cast<char\*>(&stu), sizeof(student));//覆盖重写

}

void changefirstname(fstream& here, int n, char\* fn)

{

student stu(n);//声明一个空student

here.seekg((n - 1) \* sizeof(student));//跳转到n-1对应的学生开头

here.read(reinterpret\_cast<char\*>(&stu), (sizeof(student)));//读入对应长度的firstname

stu.setfirstname(fn);//设置

here.seekp((n - 1) \* sizeof(student));//将put指针移动到哪里

here.write(reinterpret\_cast<char\*>(&stu), sizeof(student));//覆盖重写

}

void changelastname(fstream& here, int n, char\* ln)

{

student stu(n);//声明一个空student

here.seekg((n - 1) \* sizeof(student));//跳转到n-1对应的学生开头

here.read(reinterpret\_cast<char\*>(&stu), (sizeof(student)));//读入对应长度的lastname

stu.setlastname(ln);//设置

here.seekp((n - 1) \* sizeof(student));//将put指针移动到哪里

here.write(reinterpret\_cast<char\*>(&stu), sizeof(student));//覆盖重写

}

void add(student stu,fstream& here)//添加

{

here.seekp((stu.getnum() - 1) \* sizeof(student));//移动put指针

here.write(reinterpret\_cast<char\*>(&stu), sizeof(student));//将整个student写上去

}

void picture(fstream& here)

{

student stu;//空置student

int i;

i = 1;

cout << "num id firstname lastname " << endl;//初始制表（标题）

here.seekg(0);

here.read(reinterpret\_cast<char\*>(&stu), sizeof(student));//初始读入

while (i<100)

{

i++;//1至100遍历

if(stu.getnum()!=0)

stu.print();//打印

here.read(reinterpret\_cast<char\*>(&stu), sizeof(student));//向下挪动一位

here.seekg((i - 1) \* sizeof(student));

}

}

void map(fstream &here)

{//把每一个点位都创建出来，并且添加终止符

student stu;

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

here.write(reinterpret\_cast<char\*>(&stu), sizeof(student));

here << EOF;

}

void delete1(fstream& here, int num)

{//将某个点位对应的数据洗掉

student stu;

here.seekp((num - 1) \* sizeof(student));

here.write(reinterpret\_cast<char\*>(&stu), sizeof(student));

}

int main()

{//快乐测试

char p[19] = {'1','2','3','6','2','3','9','1','2','3','6','2','3','9','9','1','7','7','7'};

char fn1[19] = { 'z','h','a','n','g','\0' };

char ln1[19] = { 's','a','n','\0' };

char fn2[19] = { 'w','a','n','g','\0' };

char ln2[19] = { 'f','o','u','r','\0'};

student stu(77,fn1,ln1,p);

fstream here("student.dat", ios::in | ios::out | ios::binary);

//fstream there("window.txt", ios::in | ios::out | ios::binary);

map(here);

add(stu, here);

stu.setstudent(20, fn2, ln2, p);

add(stu,here);

picture(here);

changefirstname(here, 77, ln2);

picture(here);

delete1(here,77);

picture(here);

here.close();

//there.close();

}