

**C++实验报告**

学生姓名： 张子洋

指导老师： 阚世超

学 院： 计算机学院

专业班级： 计科2203班

2023年 12 月 12 日

**实验3 函数的声明、定义和调用**

**一、实验内容**

1、编写一个函数，用来求3个整数中的最大数。

① 用不带默认参数的函数实现。

② 用带默认参数的函数实现。

并编写主函数调用该函数，验证。

2、编写字符串查找函数，原型如下：   
 int mystrchr(char string[],char c );   
 该函数的功能：在字符串(参数 string)中查找指定字符（参数c），如果找到则返回该字符在字符串中的位置，否则返回-1。并编写主函数调用该函数，验证。

3、使用函数重载的方法定义两个重名函数，分别求出int型数的两个点间距离和浮点数的两点间距离。并编写主函数调用该函数，验证。

4、编写一个函数（void max(int a[][N],int m,int&row,int&col); //N为常量，m为数组二维数组a的行数），从一个二维整型数中查找具有最大值的元素，由引用参数row和col带回该元素的行号和列号。并编写主函数调用该函数，验证。

**二、实验过程**

**1、编写一个函数，用来求3个整数中的最大数。**

**① 用不带默认参数的函数实现。**

**② 用带默认参数的函数实现。**

**并编写主函数调用该函数，验证。**

**分析：**只需要对函数进行传参，使用三目运算符就可以解决，该问题旨在熟悉函数的操作。

**实现：**

#include<iostream>

using namespace std;

int compare1(int a,int b,int c){

return (a>b?a:b)>c?(a>b?a:b):c;

}//无默认参数

int compare2(int a=5,int b=99,int c=81){

return (a>b?a:b)>c?(a>b?a:b):c;

}//有默认参数

int main()

{

int a,b,c,n;

cin>>a>>b>>c;

n=compare1(a,b,c);

cout<<n<<endl;

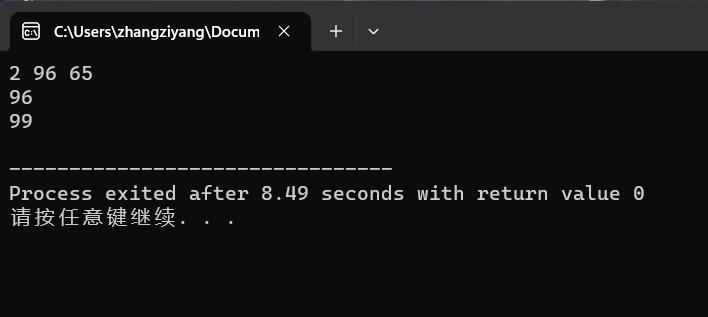
n=compare2();

cout<<n<<endl;

return 0;

}

**调试与运行：**



1. **编写字符串查找函数，原型如下：   
    int mystrchr(char string[],char c );   
    该函数的功能：在字符串(参数 string)中查找指定字符（参数c），如果找到则返回该字符在字符串中的位置，否则返回-1。并编写主函数调用该函数，验证。**

**分析：**使用简便的函数可以使代码十分精简，使用字符串的find函数，string1.find(char)，可以解决这个问题。

**实现：**

#include<iostream>

#include<algorithm>

#include<string>

#include<string.h>

using namespace std;

int mystrchr(string string1,char c ){

int n= string1.find(c);

return n;

}

int main()

{

cout<<"输入字符串"<<endl;

string string1;

cin>>string1;

char c;

cout<<"输入查询字母"<<endl;

cin>>c;

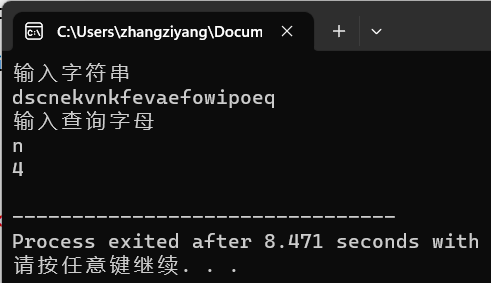
int n=mystrchr(string1,c)+1;

cout<<n<<endl;

return 0;

}

**调试与运行：**



**3、使用函数重载的方法定义两个重名函数，分别求出int型数的两个点间距离和浮点数的两点间距离。并编写主函数调用该函数，验证。**

**分析：**这个题就是用函数重载的方式，进行一个距离计算，使用开方函数即可。

**实现：**

#include<cstdio>

#include<iostream>

#include<cmath>

#include<string.h>

using namespace std;

double distance(int x1, int y1, int x2, int y2){

double ans = sqrt((x1 - x2) \* (x1 - x2) + (y1 - y2) \* (y1 - y2));

return ans;

}

double distance(double x1, double y1, double x2, double y2){

double ans = sqrt((x1 - x2) \* (x1 - x2) + (y1 - y2) \* (y1 - y2));

return ans;

}

int main(){

int a, b, c, d;

double a1, b1, c1, d1;

cout << "请输入两个点的坐标：";

cin >> a >> b >> c >> d;

double dis = distance(a, b, c, d);

cout << "两点之间距离为：" << dis << endl;

cout << "请输入两个点的坐标：";

cin >> a1 >> b1 >> c1 >> d1;

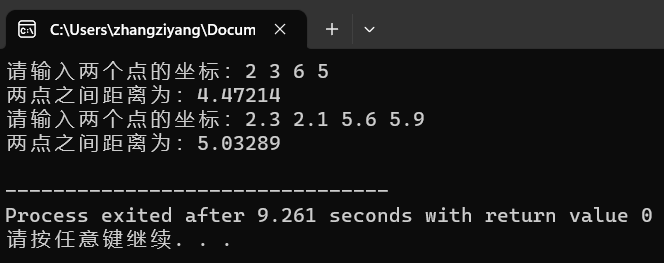
dis = distance(a1, b1, c1, d1);

cout << "两点之间距离为：" << dis << endl;

return 0;

}

**调试与运行：**



**4、编写一个函数（void max(int a[][N],int m,int&row,int&col); //N为常量，m为数组二维数组a的行数），从一个二维整型数中查找具有最大值的元素，由引用参数row和col带回该元素的行号和列号。并编写主函数调用该函数，验证。**

**分析：**这个代码就是在函数的使用中加入了指针，函数找到最大之后，返回最大值即可，指针指向的值可以被改变，只要保证函数中指针时刻指向最大值即可。代码定义每行最多5个元素。

**实现：**

#include<iostream>

#include<algorithm>

using namespace std;

#define N 5

int x,y;

void max(int a[][N],int m,int \*row,int \*col){

int maxnum=0;

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

for(int j=0;j<5;j++){

if( a[i][j]>maxnum){

x=i;

y=j;

maxnum=a[i][j];

}

}

}

cout<<"最大值为："<<maxnum<<" "<<endl;

}

int main()

{

int \*row=&x;

int \*col=&y;

Int a[][N]={55,561,65,56,52,6,8,523,3,56,1,856,56,56,89,54,89,6,9,8,88,

48,564};//23位

int n=sizeof(a)/sizeof(int);

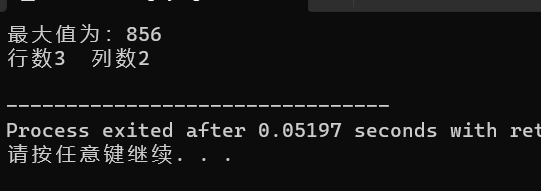
max(a,n/5,row,col);

cout<<"行数"<<\*row+1<<" "<<"列数"<<\*col+1<<endl;

return 0;

}

**调试与运行：**



**三、小结**

这次实验题目比较简单，主要是回顾之前已学内容和熟悉函数的使用。

**实验7 多态性和运算符重载**

1. **实验内容**

1、分析下面的程序，写出其运行时的输出结果。上机运行该程序，观察运行结果是否与你的分析相同。如有不同，试分析原因。

2、先建立一个点类Point，包含数据成员x，y（坐标点）。以它为基类，派生出圆类Circle，增加数据成员radius（半径），再以Cirlcle类为直接基类，派生出圆柱体类Cylinder，再增加数据成员height（高）。要求：

（1）每个类都有构造函数、用于从键盘获取数据的成员函数set()，用于显示数据的成员函数display()。

（2）用虚函数输出各类对象信息。

3（选做）、先建立一个职工类Employee，包含数据成员name（职工姓名），ID（职工编号）。以它为基类，派生出经理类Manager和技术人员类Technician，在经理类中增加数据成员salary（代表经理的月工资），在技术人员类中增加数据成员wage（代表每小时的工资数）和hours（月工作时数）。在定义类时，所有类中必须包含有构造函数、析构函数、修改和获取所有数据成员的函数，以及纯虚函数计算职工的工资，输出职工的信息。

4、设计并实现一个日期类Date，要求：

（1）可以建立具有指定日期（年、月、日）的Date对象，默认日期是2012.1.1。

（2）可以从输出流输出一个格式为“年-月-日”的日期，其中年是四位数据，月、日可以是一位也可以是两位数据。

（3）可以动态地设置年、月、日。

（4）可以用运算符= =、!=、<和>对两个日期进行比较。

（5）可以用运算符++、--、+=、-=等完成天数的加减一天或若干天的操作

（6）Date类必须能够正确表达日期，不会出现类似于13月，32日一类的情况。Date类还必须处理闰年的问题，闰年包括：所有能被400整除的年份，以及能被4整除同时又不能被100整除的年份。

**二、实验过程**

**1、分析下面的程序，写出其运行时的输出结果。上机运行该程序，观察运行结果是否与你的分析相同。如有不同，试分析原因。**

**（1）**

**分析：**这个代码就是考察虚函数和多态的使用，第一部分定义Derive类，调用构造函数，此时n=1，派生类赋值给基类，基类中仍n=1，基类调用g1,g1调用g4,输出第一部分。

第二部分使用基类指针指向派生类，g1因为在基类中写成了虚函数**，**派生类中重写了g1，调用派生类的g1，而g2，g3，g4则不具有多态性，调用基类中的，此时n值输出每次加一，为2，3，4，5。

第三部分同理，采用基类对象的引用，和类指针效用类似，g1调用派生类的，g2，g3，g4则不具有多态性，n值则输出6，7，8，9。

第四部分则通过点运算符，直接调用对象的g1函数，输出也是一样的，n的值输出10，11，12，13。

**实现：**

#include <iostream>

using namespace std;

class Base{

protected:

int n;

public:

Base (int m){ n=m++; }

virtual void g1(){cout<<"Base::g1()..."<<n<<endl; g4();}

virtual void g2(){cout<<"Base::g2()..."<<++n<<endl;g3();}

void g3(){cout<<"Base::g3()..."<<++n<<endl; g4();}

void g4(){cout<<"Base::g4()..."<<++n<<endl;}

};

class Derive:public Base{

int j;

public:

Derive(int n1,int n2):Base(n1){ j=n2; }

void g1(){cout<<"Deri::g1()..."<<++n<<endl;g2();}

void g3(){cout<<"Deri::g2()..."<<++n<<endl;g4();}

};

int main(){

Derive Dobj(1,0);//1 0

Base Bobj=Dobj;//1

Bobj.g1();//1 2 n=2

cout<<"------------------"<<endl;

Base \*bp=&Dobj;

bp->g1();// 2 3 4 5

cout<<"------------------"<<endl;

Base &bobj2=Dobj;

bobj2.g1();//6 7 8 9

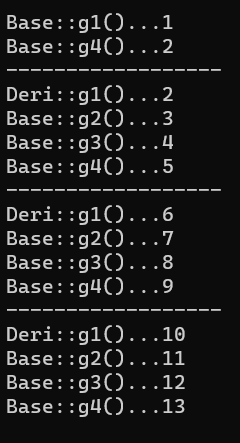
cout<<"------------------"<<endl;

Dobj.g1();

return 0;

}

**调试与运行：**



**（2）**

**分析：**

这段代码定义了一个名为ABC的类，该类具有私有成员变量a、b和c。它的构造函数使用传递的参数对这些成员变量进行初始化。在ABC类中，重载了输出流运算符<<，并利用友元函数调用类中的变量，使之可以直接输出ABC对象的内容。在main函数中，创建一个对象，传递了参数10、20和30。然后输出对象的值。

**实现：**

#include <iostream>

using namespace std;

class ABC {

int a, b, c;

public:

ABC(int x, int y, int z) :a(x), b(y), c(z) {}

friend ostream& operator<<(ostream& out, ABC& f);

};

ostream& operator<<(ostream& out, ABC& f)

{

out << "a=" << f.a << endl << "b=" << f.b << endl << "c=" << f.c << endl;

return out;

}

int main() {

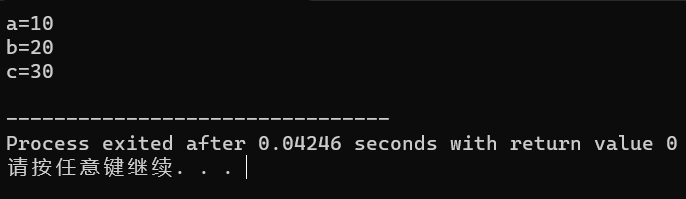
ABC obj(10, 20, 30);

cout << obj;

return 0;

}

**运行结果：**



**（3）**

**分析：**这段代码定义了一个类，私有成员变量n和构造函数。在Number类中，重载了前置自增运算符++和后置自增运算符++（使用一个int参数来区分前置和后置）。这些重载函数分别将n加加1，并返回修改后的对象的引用。前置自减运算符--和后置自减运算符--也是用同一方法进行重载。自减操作是定义两个友元函数operator--，这些函数也是将n减少1，并返回修改后的对象的引用。在main函数中，创建了对象N1，传参。然后进行自增和自减操作，最后输出对象的n值。

前置自增运算符++在输出前进行了3次连续的自增操作，使n的值增加到13。后置自增运算符++在输出后进行了1次自增操作，使n的值再增加1。前置自减运算符--在输出前进行了1次自减操作，使n的值减少到14。后置自减运算符--在输出后进行了3次连续的自减操作，使n的值减少到13。

**实现：**

#include<iostream>

using namespace std;

class Number {

int n;

public:

Number(int x) :n(x) {}

Number& operator++() { ++n; return \*this; }

Number& operator++(int) { n++; return \*this; }

friend Number& operator--(Number& o);

friend Number& operator--(Number o, int);

void display() { cout << "This Number is: " << n << endl; }

};

Number& operator--(Number& o) { --o.n; return o; }

Number& operator--(Number o, int) { o.n--; return o; }

int main() {

Number N1(10);

++ ++ ++N1;

N1.display();

N1++;

N1.display();

--N1;

N1.display();

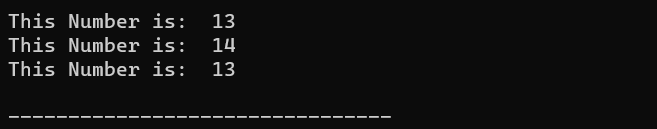
N1-- -- --;

N1.display();

return 0;

}

**运行结果：**



**2、先建立一个点类Point，包含数据成员x，y（坐标点）。以它为基类，派生出圆类Circle，增加数据成员radius（半径），再以Cirlcle类为直接基类，派生出圆柱体类Cylinder，再增加数据成员height（高）。要求：**

**（1）每个类都有构造函数、用于从键盘获取数据的成员函数set()，用于显示数据的成员函数display()。**

**（2）用虚函数输出各类对象信息。**

**分析：**代码定义了三个类：Point（点）、Circle（圆）和Cylinder（圆柱）。其中Circle和Cylinder是Point的派生类，继承Point的属性和方法。Point类定义了构造函数，set()方法，以及一个虚函数display()，输出这个点的位置。

Circle类继承Point类，并添加变量radius，并重写set()方法，从用户输入中得到半径，display()方法输出圆的半径。

Cylinder类继承Circle类，并添加了一个整型变量height，同样有一个set()方法，从用户输入中得到高，也有一个display()方法，输出圆柱的高度。

在main()函数中，我们创建对象p、c和r，并分别调用set()和display()方法，代码实现了一个基本的类继承关系，并通过虚函数实现了多态。

**实现：**

#include<iostream>

using namespace std;

class Point{

public:

int x;

int y;

Point() {

x = 0;

y = 0;

};

void set() {

cout << "输入X和Y：";

cin >> x >> y;

}

virtual void display(){

cout << "点的位置是：" << x << " " << y << endl;

}

};//基类

class Circle:public Point

{

public:

int radius;

Circle() { radius = 0; };

void set()

{

cout << "输入半径：";

cin >> radius;

}

void display() {

cout << "圆的半径是：" << radius << endl;

}

};

class Cylinder:public Circle

{

public:

int height;

Cylinder() {

height = 0;

};

void set(){

cout << "输入高：";

cin >> height;

}

void display () {

cout << "圆柱的高：" << height << endl;

}

};

int main()

{

Point p;

p.set();

p.display();

Circle c;

c.set();

c.display();

Cylinder r;

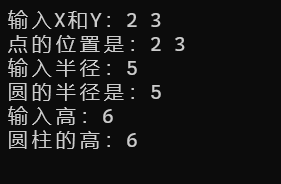
r.set();

r.display();

return 0;

}

**运行结果：**



**3（选做）、先建立一个职工类Employee，包含数据成员name（职工姓名），ID（职工编号）。以它为基类，派生出经理类Manager和技术人员类Technician，在经理类中增加数据成员salary（代表经理的月工资），在技术人员类中增加数据成员wage（代表每小时的工资数）和hours（月工作时数）。在定义类时，所有类中必须包含有构造函数、析构函数、修改和获取所有数据成员的函数，以及纯虚函数计算职工的工资，输出职工的信息。**

**分析：**

这段代码实现了一个简单的职员薪资计算系统，通过使用类和继承来实现不同职员的不同薪资计算方法。在主函数中，创建一个Manager对象m，然后调用Set()方法设置经理的月薪为2000元，再调用Change()方法将经理的月薪修改为5000元，最后调用Sum()方法计算并输出经理的薪资信息。然后，输入一个技术人员的姓名、ID、工时数和时薪，创建Technician对象T，和对象m操作类似。

在Employee类中定义了一个纯虚函数Sum()，该函数在Manager和Technician两个派生类中都被重载，具有多态性，所以可以实现不同职员的薪资计算方法。最后，由于在各个类中使用了析构函数，当程序结束时，会先调用派生类的析构函数，再调用基类的析构函数。因此输出的结果中，先输出派生类的析构函数，在输出基类的析构函数。

**实现：**

#include<iostream>

using namespace std;

class Employee{

public:

string name;

int ID;

virtual void Sum() = 0; //纯虚函数

Employee(string s,int a){

name = s;

ID = a;

}

~Employee(){

cout << "基类 Employee" << endl;

}

void Change(string s, int x){

this->name = s;

this->ID = x;

}

void Set(string s,int x){

this->name = s;

this->ID = x;

}

};

class Manager:public Employee{

public:

int salay;

virtual void Sum() {

cout << "经理的名字是：" << name << endl;

cout << "经理的ID是：" << ID << endl;

cout << "经理的工资是：" << salay << endl;

}

Manager(string s,int a ,int sa):Employee(s,a){

salay = sa;

}

~Manager(){

cout << "派生类 Manager" << endl;

}

void Change(int a){

this->salay = a;

}

void Set(int a){

this->salay = a;

}

};

class Technician:public Employee{

public:

int wage;//表示时工资

int hours;//月工作小时数

virtual void Sum() {

cout << "技术工的名字是：" << name << endl;

cout << "技术工的ID是：" << ID << endl;

cout << "技术工的工资是：" << wage\*hours\*4 << endl;

}

Technician(string s,int a,int w,int h):Employee(s,a){

wage = w;

hours = h;

}

~Technician(){

cout << "派生类 Technician" << endl;

}

void Change(int wage, int hours){

this->wage = wage;

this->hours = hours;

}

void Set(int wage,int hours){

this->wage = wage;

this->hours = hours;

}

};

int main(){

string name;

int ID,w;

cout<<"请输入经理名称：";cin>>name;

cout<<"请输入ID：";cin>>ID;

cout<<"请输入月工资：";cin>>w;

Manager m(name,ID,w);

m.Set(2000);

m.Change(5000);

m.Sum();//工资已被修改

cout<<"-----------------------"<<endl;

int time;

cout<<"请输入技术人员名称：";cin>>name;

cout<<"请输入ID：";cin>>ID;

cout<<"请输入工时：";cin>>time;

cout<<"请输入月工资：";cin>>w;

Technician T(name,ID,time,w);

T.Set(30,100);

T.Change(50, 100);

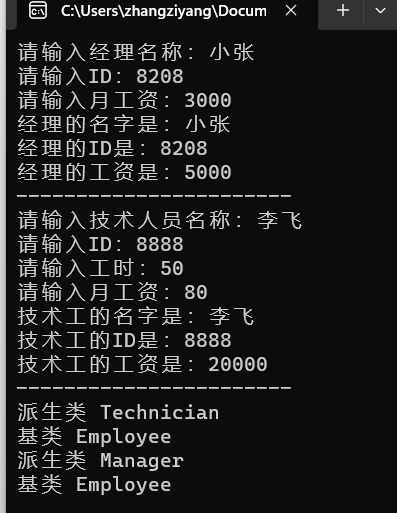
T.Sum();

cout<<"-----------------------"<<endl;

return 0;

}

**运行结果：**



**4、设计并实现一个日期类Date，要求：**

**（1）可以建立具有指定日期（年、月、日）的Date对象，默认日期是2012.1.1。**

**（2）可以从输出流输出一个格式为“年-月-日”的日期，其中年是四位数据，月、日可以是一位也可以是两位数据。**

**（3）可以动态地设置年、月、日。**

**（4）可以用运算符= =、!=、<和>对两个日期进行比较。**

**（5）可以用运算符++、--、+=、-=等完成天数的加减一天或若干天的操作**

**（6）Date类必须能够正确表达日期，不会出现类似于13月，32日一类的情况。Date类还必须处理闰年的问题，闰年包括：所有能被400整除的年份，以及能被4整除同时又不能被100整除的年份。**

**分析：**

这个题代码很长，但具体实现并不难。首先实现Data类运用构造函数初始化，然后定义输出函数display（），输出年时，**如果不够四位要在前面加0来补位，用setw(4) +setfill('0') 可以实现**。动态设置就用输入流修改即可，然后就是运算符的重载，确定传入的参数后（比较简单），就可以根据步骤进行了，需要注意的是当你在进行日期的加减时，需要考虑月的进位和年的进位（**注意：每月的时间不一样，闰年和平年的2月日期差一天，所以在每次月份改变时要查看该月的天数，每一次进行年份的改变时，一定要判断是否为闰年，否则特殊样例时代码结果会有细微的错误**），月份使用两个数组，分别储存平年和闰年每月的天数，而check()函数用来检查输入的日期是否规范。运算符重载就是按部就班，具体代码实现就比较简单了。

**代码：**

#include<iostream>

#include <iomanip>

using namespace std;

class Date{

public:

int year;

int month;

int day;

Date() {

year = 2012;

month = 1;

day = 1;

};

Date(int year, int month, int day){

if (check(year,month,day)){//要对日期进行检查

this->year = year;

this->month = month;

this->day = day;

}else{

cout << "格式错误" << endl;

}

}

bool check(int y,int m,int d){

int num1[12] = { 31,29,31,30,31,30,31,31,30,31,30,31 };

int num2[12] = { 31,28,31,30,31,30,31,31,30,31,30,31 };//存储每月的天数

if (m > 12 || m < 0){

return false;

}//检查月份

else{

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

//year = y;

//month = m;

if (d <= num2[m-1] && d >= 1){

return true;

//day = d;

}

else return false;

}else{

//year = y;

if (d <= num1[m - 1] && d >= 1){

return true;

}else{

//cout << "格式错误" << endl;

return false;

}

}

}

}

void setdate(int year, int month, int day){

if (check(year,month,day)){//要对日期进行检查

this->year = year;

this->month = month;

this->day = day;

}else{

cout << "格式错误" << endl;

}

}

void display(){

cout << setw(4) << setfill('0') << year;//setw(位数)，和setfill(指定字符),确保年份为四位数

cout << "-"<< month <<"-" << day << endl;

}

//下面对运算符进行重载

bool operator == (Date& data){

if (year == data.year && month == data.month && day == data.day){

return true;

}else{

return false;

}

};

bool operator!=(Date& data){

if (year == data.year && month == data.month && day == data.day){

return false;

}else{

return true;

}

};

bool operator<(Date& data){

check(data.year, data.month, data.day);

if (year > data.year){

return true;

}

else if(year < data.year){

return false;

}

else{

if (month > data.month){

return true;

}

else if (month < data.month){

return false;

}

else {

if (day >= data.day){

return true;

}

else{

return false;

}

}

}

};

bool operator >(Date& data){

check(data.year, data.month, data.day);

if (year < data.year){

return true;

}else if (year > data.year){

return false;

}else {

if (month < data.month){

return true;

}

else if (month > data.month){

return false;

}else {

if (day <= data.day){

return true;

}

else{

return false;

}

}

}

};

Date operator+=(int x){//还要考虑进位

Date d(year, month, day);

day = day + x;

int num1[12] = { 31,29,31,30,31,30,31,31,30,31,30,31 };

int num2[12] = { 31,28,31,30,31,30,31,31,30,31,30,31 };

while(true){

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

if (day > num2[month - 1]){

day = day - num2[month - 1];

month++;

if (month > 12){

year++;

month = 1;

}

}else{

break;

}

} else{

if (day > num1[month - 1]){

day = day - num1[month - 1];

month++;

if (month > 12){

year++;

month = month;

}

}else{

break;

}

}

}

return d;

};

Date operator -=(int x){

Date d(year, month, day);

day = day - x;

int num1[12] = { 31,29,31,30,31,30,31,31,30,31,30,31 };

int num2[12] = { 31,28,31,30,31,30,31,31,30,31,30,31 };

while(true){

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

if(day <= 0){

month--;

if (month <= 0){

month = 12;

day = day + num2[month - 1];

year--;

}

else{

day = day + num2[month - 1];

}

}else{

break;

}

}

else{

if (day <= 0){

month--;

if (month <= 0){

month = 12;

day = day + num1[month - 1];

year--;

}else{

day = day + num1[month - 1];

}

}else break;

}

}

return d;

};

Date operator -- (){

Date d(year,month,day);

day = day - 1;

int num1[12] = { 31,29,31,30,31,30,31,31,30,31,30,31 };

int num2[12] = { 31,28,31,30,31,30,31,31,30,31,30,31 };

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

while (day <= 0){

month--;

if (month <= 0){

month = 12;

day = day + num2[month-1];

year--;

}else{

day = day + num2[month - 1];

}

}

}

else{

while(day <= 0){

month--;

if (month <= 0){

month = 12;

day = day + num1[month - 1];

year--;

}else{

day = day + num1[month - 1];

}

}

}

return d;

};

Date operator ++(){

Date d(year,month,day);

day = day + 1;

int num1[12] = { 31,29,31,30,31,30,31,31,30,31,30,31 };

int num2[12] = { 31,28,31,30,31,30,31,31,30,31,30,31 };

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

while (day > num2[month - 1]){

day = day - num2[month - 1];

month++;

if (month > 12){

year++;

month = 1;

}

}

}

else{

while (day > num1[month - 1]){

day = day - num1[month - 1];

month++;

if (month > 12){

year++;

month = 1;

}

}

}

return d;

};

};

int main(){

int a,b,c;

cout<<"请分别输入年份，月份，日期：";

cin>>a>>b>>c;

Date d(a,b,c);cout<<"日期为：";

d.display();

++d;cout<<"日期++后的值：";

d.display();

--d;cout<<"日期--后的值：";

d.display();

cout<<"日期增加天数：";

cin>>a;

d += (a);cout<<"日期+后的值：";

d.display();

cout<<"日期减少天数：";

cin>>a;

d -= (a);cout<<"日期-后的值：";

d.display();

cout<<"请分别输入第二个年份，月份，日期：";

cin>>a>>b>>c;

Date d2(a,b,c);cout<<"日期为：";

d2.display();

bool x = d > d2;

if(x) cout<<"原日期大于新日期"<<endl; else cout<<"原日期不大于新日期"<<endl;

x = d == d2;

if(x) cout<<"原日期等于新日期"<<endl; else cout<<"原日期不等于新日期"<<endl;

x= d != d2;

if(x) cout<<"原日期不等于新日期"<<endl; else cout<<"原日期等于新日期"<<endl;

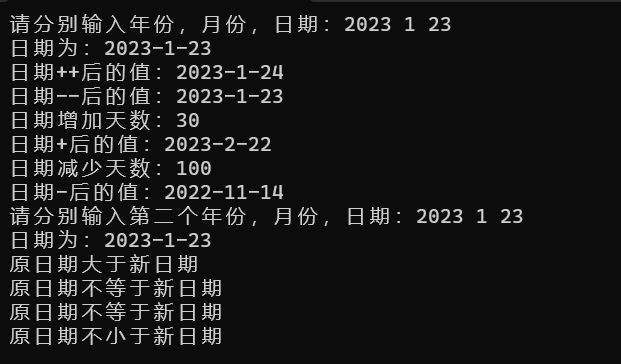
x = d < d2;

if(x) cout<<"原日期小于新日期"<<endl; else cout<<"原日期不小于新日期"<<endl;

return 0;

}

**运行结果：**



**三、小结**

这次实验主要就是考察对虚函数，纯虚函数实现多态的理解以及运算符重载的代码重写。第四个代码需要考虑的细节较多，但是捋清思路还是比较容易上手的。写了这几个题让我对虚函数和多态有了更深的理解。

**实验8 流类库与输入/输出**

**一、实验内容**

1．阅读下面的程序，写出程序运行的结果。

2、已知数据文件IN.DAT中存有20个整数，每个整数间用空格分隔。有一类Array的结构如下：

class Array{

int A[20];

public:

Array(){

for(int i=0;i<20;i++) A[i]=0;

}

int getNumberA(Array &a,int k){return a.A[k];}

void getdata();//读数据函数

void max\_min(int &,int &);//排序函数

void putdata(int &,int &);//写结果数据函数

};

其中：

成员函数getdata()的功能为：从数据文件IN.DAT中把20个数据读出来存入数据成员A[]中。

成员函数max\_min(int &,int &)的功能为：求数据成员A[]中20个整数的最大值和最小值。

成员函数putdata(int &,int &)的功能为：把求得的数据成员A[]中20个整数的最大值和最小值输出到数据文件OUT.DAT。

要求：在类外写出上述三个成员函数的实现代码，并在main函数中对该类进行测试。

3、假设有学生类Student，包括姓名、学校、专业、班级、电话号码、通信地址、邮政编码等数据成员。编程完成Student类的设计，从键盘输入10个同学的通信录信息，并将这10个同学的信息写入磁盘文件address.dat中。然后从address.dat文件中读取各同学信息并显示在屏幕上。

**二、实验过程**

**1．阅读下面的程序，写出程序运行的结果。**

**（1）**

**分析：**

先定义两个文件流对象，然后打开一个名为"a.dat"的文件，以输出模式打开（ios::out）。

将三行字符串写入到文件中。然后关闭文件流out。又声明了一个字符数组buffer，用于存储从文件中读取的文本。以输入模式打开文件"a.dat"，从文件中读取一行文本，并存储到buffer中并输出到控制台。

这段代码的主要功能是将一些文本写入一个文件中，并从该文件中读取文本并输出到控制台。需要注意的是，这段代码并没有对读取的文本进行任何处理或修改，只是直接将其输出到控制台。

**代码：**

#include<iostream>

#include<fstream>

using namespace std;

int main(){

fstream out,in;

out.open("a.dat",ios::out);

out<<"on fact\n";

out<<"operating file \n";

out<<"is the same as inputing/outputing data on screen...\n";

out.close();

char buffer[80];

in.open("a.dat",ios::in);

while(!in.eof())

{

in.getline(buffer,80);

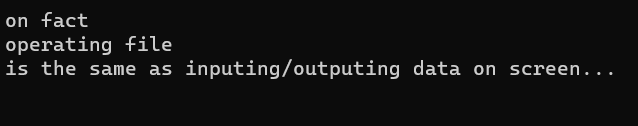
cout<<buffer<<endl;

}

return 0;

}

**运行结果：**



**（2）**

**分析：**

这段代码实现了将 Worker类写入二进制文件，以及从二进制文件中读取出 Worker 类型的数据并显示。先是定义了一个 Worker 类，输入 6 个员工的信息。打开 Worker.dat 文件并将文件流对象 out 与该文件关联，使用二进制模式以写入的方式打开文件。使用 seekg() 方法移动文件读取指针到文件中的第 3 个 Worker 对象的位置，即第二个元素之后。其中，第一个参数为偏移量，第二个参数为偏移的基准位置（ios::beg 表示文件开头，ios::cur 表示当前位置，ios::end 表示文件结尾）。使用 read() 方法从文件中读取一个 Worker 对象的信息，并将读取的信息存储到 s1 对象中。其中，read() 方法的第一个参数需要传入一个 char\* 类型的指针，因此需要将 s1 转换成 char\* 类型的指针，再传入 read() 方法中。使用 seekg() 方法将文件读取指针移动到文件开头。再次使用 read() 方法从文件中读取一个 Worker 对象的信息，并将读取的信息存储到 s1 对象中。再次调用 s1 的 display() 方法，将读取的信息在屏幕上显示出来。

**代码：**

#include<iostream>

#include<cstring>

#include<fstream>

using namespace std;

class Worker{

private:

int number ,age;

char name[20];

double sal;

public:

Worker(){}

Worker(int num, const char\* Name,int Age, double Salary):number(num),age(Age),sal(Salary)

{strcpy\_s(name,Name); }

void display() {cout<<number<<"\t"<<name<<"\t"<<age<<"\t"<<sal<<endl; }

};

int main(){

ofstream out("Worker.dat",ios::out|ios::binary);//ios::out | ios::binary 表示二进制模式以写入的方式打开

//ios::binary 可以和其他模式标记组合使用

Worker man[]={Worker(1,"张三",23,2320),Worker(2,"李四",32,2321),

Worker(3,"王五",34,2322),Worker(4,"刘六",27,2324),

Worker(5,"晓红",23,2325),Worker(6,"黄明",50,2326)};

for(int i=0;i<6;i++) out.write((char\*)&man[i],sizeof(man[i]));

out.close();

Worker s1;

ifstream in("Worker.dat",ios::in|ios::binary); //表示二进制模式以读取的方式打开

in.seekg(2\*(sizeof(s1)),ios::beg); //seekg从文件中 get（获取）信息 ； seekp可用于将信息 put（放入 写入）到文件中

in.read((char\*)&s1,sizeof(s1));

s1.display();

in.seekg(0,ios::beg);

in.read((char\*)&s1,sizeof(s1));

s1.display();

in.close();

return 0;

}

**输出结果：**



**2、已知数据文件IN.DAT中存有20个整数，每个整数间用空格分隔。有一类Array的结构如下：**

**class Array{**

**int A[20];**

**public:**

**Array(){**

**for(int i=0;i<20;i++) A[i]=0;**

**}**

**int getNumberA(Array &a,int k){return a.A[k];}**

**void getdata();//读数据函数**

**void max\_min(int &,int &);//排序函数**

**void putdata(int &,int &);//写结果数据函数**

**};**

**其中：**

**成员函数getdata()的功能为：从数据文件IN.DAT中把20个数据读出来存入数据成员A[]中。**

**成员函数max\_min(int &,int &)的功能为：求数据成员A[]中20个整数的最大值和最小值。**

**成员函数putdata(int &,int &)的功能为：把求得的数据成员A[]中20个整数的最大值和最小值输出到数据文件OUT.DAT。**

**要求：在类外写出上述三个成员函数的实现代码，并在main函数中对该类进行测试。**

**分析：**

构造函数初始化数组A。

getNumberA 功能：返回k 处的值。

getdata 功能： 从“IN.DAT”中读取 20 个整数。文件放入数组A。

max\_min 功能：查找数组中的最大值和最小值A。

putdata 功能： 将最大值和最小值写入“OUT.DAT”文件中。文件。。

文件处理：程序从“IN.DAT”读取数据。并将结果写入“OUT.DAT”。如果文件操作失败，则会显示错误消息。

**代码：**

#include <iostream>

#include <fstream>

using namespace std;

class Array {

int A[20];

public:

Array() {

for (int i = 0; i < 20; i++)

A[i] = 0;

}

int getNumberA(Array &a, int k) {

return a.A[k];

}

void getdata() {

ifstream inFile("IN.DAT");

if (inFile.is\_open()) {

for (int i = 0; i < 20; i++)

inFile >> A[i];

inFile.close();

} else {

cerr << "Error: Unable to open IN.DAT" << endl;

}

}

void max\_min(int &maxValue, int &minValue) {

maxValue = A[0];

minValue = A[0];

for (int i = 1; i < 20; i++) {

if (A[i] > maxValue)

maxValue = A[i];

if (A[i] < minValue)

minValue = A[i];

}

}

void putdata(int &maxValue, int &minValue) {

ofstream outFile("OUT.DAT");

if (outFile.is\_open()) {

outFile << "Max Value: " << maxValue << endl;

outFile << "Min Value: " << minValue << endl;

outFile.close();

} else {

cerr << "Error: Unable to open OUT.DAT" << endl;

}

}

};

int main() {

Array myArray;

myArray.getdata();

int maxValue, minValue;

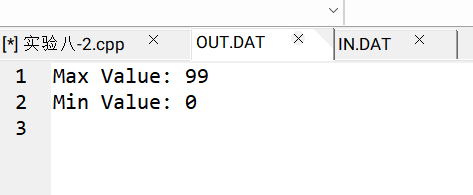
myArray.max\_min(maxValue, minValue);

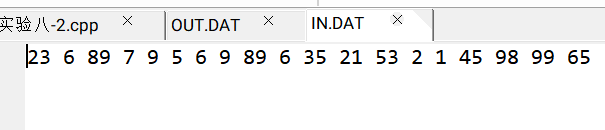
myArray.putdata(maxValue, minValue);

return 0;

}

**输出结果：**





**3、假设有学生类Student，包括姓名、学校、专业、班级、电话号码、通信地址、邮政编码等数据成员。编程完成Student类的设计，从键盘输入10个同学的通信录信息，并将这10个同学的信息写入磁盘文件address.dat中。然后从address.dat文件中读取各同学信息并显示在屏幕上。**

**分析：**

inputInfo()方法通过 getline(cin, ...) 函数从用户那里获取学生的信息。owriteToFile将学生的信息写入到文件 address.dat 中。oreadFromFile(ifstream &file) 方法从 address.dat 文件中读取学生的信息。初始化一个名为 students 的数组，用于存储 Student 对象。将所有学生的信息写入到 address.dat 文件中，再从文件中读取信息并逐个显示出来。总体就是对文件的处理。

**代码：**

#include <iostream>

#include <fstream>

using namespace std;

class Student {

private:

string name;

string school;

string major;

string className;

string phoneNumber;

string address;

string postalCode;

public:

void inputInfo() {

cout << "Enter student information:" << endl;

cout << "Name: ";

getline(cin, name);

cout << "School: ";

getline(cin, school);

cout << "Major: ";

getline(cin, major);

cout << "Class: ";

getline(cin, className);

cout << "Phone Number: ";

getline(cin, phoneNumber);

cout << "Address: ";

getline(cin, address);

cout << "Postal Code: ";

getline(cin, postalCode);

}

void displayInfo() const {

cout << "Name: " << name << endl;

cout << "School: " << school << endl;

cout << "Major: " << major << endl;

cout << "Class: " << className << endl;

cout << "Phone Number: " << phoneNumber << endl;

cout << "Address: " << address << endl;

cout << "Postal Code: " << postalCode << endl;

cout << "------------------------" << endl;

}

void writeToFile(ofstream &file) const {

file << name << endl;

file << school << endl;

file << major << endl;

file << className << endl;

file << phoneNumber << endl;

file << address << endl;

file << postalCode << endl;

file << "------------------------" << endl;

}

void readFromFile(ifstream &file) {

getline(file, name);

getline(file, school);

getline(file, major);

getline(file, className);

getline(file, phoneNumber);

getline(file, address);

getline(file, postalCode);

string separator;

getline(file, separator); // Read and discard separator line

}

};

int main() {

const int numStudents = 10;

Student students[numStudents];

// Input student information

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

students[i].inputInfo();

}

// Write information to file

ofstream outFile("address.dat");

if (outFile.is\_open()) {

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

students[i].writeToFile(outFile);

}

outFile.close();

cout << "Student information has been written to address.dat." << endl;

} else {

cerr << "Error: Unable to open address.dat for writing." << endl;

return 1;

}

// Read information from file and display

ifstream inFile("address.dat");

if (inFile.is\_open()) {

Student student;

while (!inFile.eof()) {

student.readFromFile(inFile);

if (!inFile.eof()) {

student.displayInfo();

}

}

inFile.close();

} else {

cerr << "Error: Unable to open address.dat for reading." << endl;

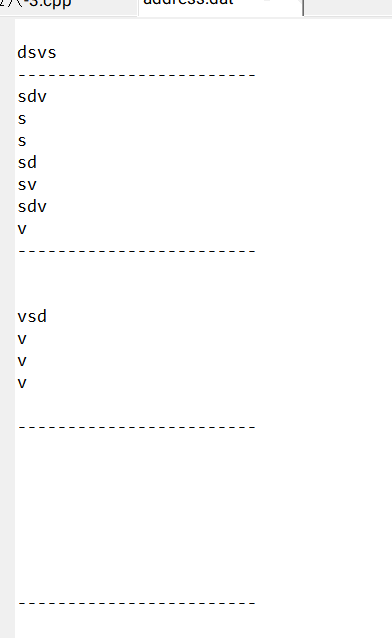
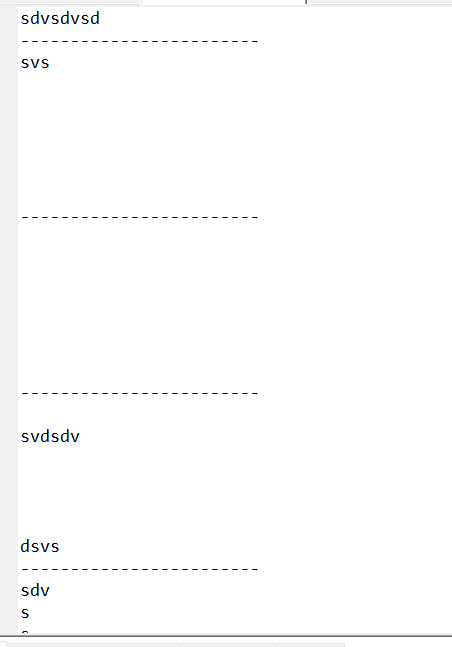
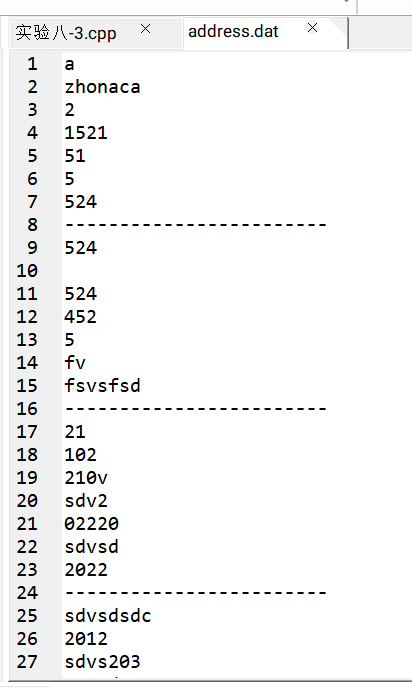
return 1;

}

return 0;

}

**输出结果：**



**三、小结**

这次实验是熟悉深入理解C++的输入输出的含义与其实现方法，掌握对文件的输入输出操作。和之前的一样，都是根据题目进行编程，熟悉文件流的操作。

**写在最后：**

一学期的C++学习让我收获颇丰，学习C++的人有很多，有趣的是很少人敢于将精通C++写在自己的简历上。我也有C++并不难掌握的错觉，越往后学习越发现自己只是学了一点皮毛，到了STL标准库更是很难去熟悉和使用。在学习了汇编语言和同为高级语言的JAVA后，我对C++的理解也发生了变化，汇编语言频繁地进行地址的访问和值的传递，进行计算机很基本的数据处理，而指针、内存分配等都说明C++在偏底层和硬件方面有更多的控制，C++多继承，多态，指针引用也提高了代码的复杂性。C++学习任重道远，希望之后能够真正精通这门语言。