**《面向对象编程（C++)》**

**实验指导**

**指导教师：刘嫔**

**电话：13755071484**

**QQ：24488083**

**中南大学**

**计算机学院**

**2020年**

### 实验目的

1、利用C++语言编写程序解决一些具体的问题，培养学生发现、分析和解决问题的能力；

2、了解程序错误的诊断和排除；

3、了解程序设计规范；

### 使用的工具软件及环境

Microsoft Visual Studio 6.0 以上

### 实验内容

**实验1 熟悉VC ++ 环境**

**一、实验目的**

1、熟悉VC++6.0集成开发环境，熟练掌握VC++6.0项目工作区、编辑器、菜单栏和工具栏的使用。

2、掌握如何编辑、编译、链接和运行一个C++程序。

3、学会使用输入输出标准流对象进行输入/输出。

**二、实验内容**

1．熟悉Visual C++集成开发环境；

2．使用Visual C++集成开发环境编写、调试3个简单的C++程序。记录每个程序的调试修改和运行过程。

参考问题：

（1）“Hello world”输出（使用输入输出标准流对象）；

（2）编写打印菱形的程序。

\*  
 \*\*\*  
 \*\*\*\*\*  
 \*\*\*\*\*\*\*  
 \*\*\*\*\*  
 \*\*\*  
 \*

（3）声明一个表示学生的结构体，学生信息包括：学号、姓名、专业、成绩；提示用户输入学生的信息，然后完整地输出出来。

**时间要求：第一次实验课(第三周)完成**

#include<iostream>

using namespace std;

void printHelloWorld(){

cout << "HelloWorld" << endl;

}

void printChar(int num, char pattern) {

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

cout << pattern;

}

}

void printLine(int size, int num, char pattern) {

printChar((size - num) / 2, ' ');

printChar(num, pattern);

printChar((size - num) / 2, ' ');

cout << endl;

}

int abs(int num) {

if (num < 0) {

return num \* -1;

}

return num;

}

void printPattern(int num, char pattern) {

int middle = (num + 1) / 2;

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

int blankNum = abs(middle - i);

int patternNum = num - 2 \* blankNum;

printLine(num, patternNum, pattern);

}

}

struct studentInfo {

long id = 0;

char name[10];

char major[10];

float grade = 0;

};

void inputStudentInfo(struct studentInfo\* student) {

cout << "Input student's id: ";

cin >> student->id;

cout << "\nInput student's name: " ;

cin >> student->name;

cout << "\nInput student's major: " ;

cin >> student->major;

cout << "\nInput student's grade: " ;

cin >> student->grade;

}

void printStudentInfo(struct studentInfo\* student) {

cout << student->id << " "

<< student->name << " "

<< student->major << " "

<< student->grade << endl;

}

int main() {

cout << "Question1: " << endl;

printHelloWorld();

cout << "Question2: " << endl;

printPattern(7,'\*');

cout << "Question3:" << endl;

studentInfo student1;

inputStudentInfo(&student1);

printStudentInfo(&student1);

return 0;

}

**实验2 C++简单程序设计**

**一、实验目的**

1、复习C++基本程序设计要素：常量、变量、运算符与表达式、三种基本控制结构、枚举类型，通过相关简单程序编写掌握这些基础。

2、进一步熟悉编辑、编译、连接和运行C++程序的方法。

3、进一步熟悉C++程序的结构和编程方法。

**二、实验内容**

在VC++集成开发环境中编写、调试解决以下问题的C++程序。记录每个程序的调试修改和运行过程。

1、求100之内的自然数中被13整除的最大数。

2、编写程序求斐波那契数列的第n 项和前n 项之和。斐波那契数列是形如

0, 1, 1, 2, 3, 5, 8, 13, ...

其通项为: F0 = 0; F1 = 1;

Fn = Fn-1+Fn-2 。

3、输入5个字符串，要求对它们按由小到大顺序排列。

4、（选做）有五本新书，要借给A，B，C三位同学，若每人每次只能借一本，则可以有多少种不同的借法？

5、（选做或课外练习）：习题2-18~2-29，2-32~2-35

**时间要求：第一次实验课 完成**

#include<iostream>

#include<String>

using namespace std;

int findLargestNum(int den, int num) {

int n = den / num;

return n \* num;

}

long fab(int n) {

if (n == 0 || n == 1) {

return n;

}

return fab(n - 1) + fab(n - 2);

}

int sumFab(int n) {

int sum = 0;

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

sum += fab(i);

}

return sum;

}

void arr\_add(int index, int arr[], int val) {

for (int i = sizeof(arr); i > index; i--) {

arr[i] = arr[i - 1];

}

arr[index] = val;

}

void sortString() {

const int str\_arr\_len = 5;

string str\_arr[str\_arr\_len];

cout << "请输入5个字符串： " << endl;

for (int i = 0; i < str\_arr\_len; i++) {

cin >> str\_arr[i];

}

int index\_arr[str\_arr\_len];

index\_arr[0] = 0;

for (int i = 1; i < str\_arr\_len; i++) {

bool if\_add = false;

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

if (str\_arr[i] <= str\_arr[index\_arr[j]]) {

arr\_add(j, index\_arr, i);

if\_add = true;

break;

}

}

if (!if\_add) {

arr\_add(i, index\_arr, i);

}

}

cout << "排序后： " << endl;

for (int i = 0; i < str\_arr\_len; i++) {

cout << str\_arr[index\_arr[i]] << endl;

}

}

int main() {

cout << "第一题： ";

cout << findLargestNum(100, 13) << endl;

cout << "第二题： ";

int n;

cout << "请输入第一个整数：";

cin >> n;

cout << "第" << n << "项斐波那契数： " << fab(n)

<< "，前" << n << "项斐波那契数列之和： " << sumFab(n) << endl;

cout << "第三题： ";

sortString();

return 0;

}

**实验3 函数的声明、定义和调用**

**一、实验目的**

1．掌握函数定义及函数调用方法；

2．掌握函数间参数传递方式（值方式和引用方式）；

3．掌握库函数的使用。

**二、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，否则返回零。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），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。

5、（选做或课外练习）：习题3-7~3-10，3-12~3-16

**三、实验报告（按要求撰写实验报告）**

**时间要求：第二次实验课 完成**

#include<iostream>

using namespace std;

int findLargestNum(int a, int b, int c = 0) {

if (a < b) {

a = b;

}

if (a < c) {

a = c;

}

return c;

}

int mystrchr(const char string[], char c) {

for (int i = 0; i < strlen(string); i++) {

if (string[i] == c) {

return i + 1;

}

}

return 0;

}

int distance(int a, int b) {

if (a > b) {

return a - b;

}

return b - a;

}

double distance(double a, double b) {

if (a > b) {

return a - b;

}

return b - a;

}

const int N = 5;

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

int maxNum = a[0][0];

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

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

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

maxNum = a[i][j];

row = i;

col = j;

}

}

}

}

int main() {

cout << "第一题：" << endl;

cout << findLargestNum(1, 2, 3) << endl;

cout << " 第二题： " << endl;

char str[5] = "asd";

cout << mystrchr(str, 'k') << endl;

cout << mystrchr("jkl", 'k') << endl;

cout << "第三题： " << endl;

cout << distance(1, 2);

cout << distance(2.0, 1.0) << endl;

cout << "第四题： " << endl;

int row = 0;

int col = 0;

int arr[1][N] = { 1,2,3,4,5 };

max(arr, 1, row, col);

cout << row << ' ' << col << endl;

return 0;

}

**实验4 类和对象（一）**

**一、实验目的**

1．掌握类的声明和使用方法；

2．掌握对象的创建和使用方法

**二、实验内容**

1、检查下面的程序，找出其中的错误，并改正之。然后上机调试，使之能正常运行。

#include <iostream>

using namespace std;

class Clock

{ void set\_clock(void);

void show\_clock(void);

int hour;

int minute;

int second;

};

Clock clock;

int main()

{

set\_clock();

show\_clock ();

}

int set\_clock(void)

{

cin>>t.hour;

cin>>t.month;

cin>>t.second;

}

int show\_clock(void)

{

cout<<t.hour<<":"<<t.minute<<":"<<t.second<<endl;

}

2、建立一个名为Student的类，该类有以下几个私有成员变量：学生姓名、学号、性别、年龄。还有以下两个成员函数：

（1）SetStu：用于初始化学生姓名、学号、性别和年龄的构造函数；

（2）PrintStu：用于输出学生信息的函数。

编写一个主函数，创建一个学生对象，然后调用成员函数输入学生对象数据，并将该学生对象的信息在屏幕上输出。

3、定义一个名为Time 的时间类，包含时、分、秒等属性，对这些属性操作的方法有：

（1）printMilitary()：输出24小时制时间（如13:25:39）；

（2）printStandard()：输出标准格式时间（如 1:25:39 PM）

实现并测试这个类。

4、（选做）设计一个职工类Employee。数据成员包括字符串型的name（姓名）、street（街道地址）、city（市）、province（省）、zip（邮政编码）。要求用成员函数实现以下功能：

（1）可动态地设置职工对象信息。

（2）在屏幕上打印职工信息。

实现并测试这个类。

**三、实验报告（按要求撰写实验报告）**

**时间要求：第三次实验课 完成**

/\* 4.cpp :

1、检查下面的程序，找出其中的错误，并改正之。然后上机调试，使之能正常运行。

#include <iostream>

using namespace std;

class Clock

{

void set\_clock(void);

void show\_clock(void);

int hour;

int minute;

int second;

};

Clock clock;

int main()

{

set\_clock();

show\_clock();

}

int set\_clock(void)

{

cin >> t.hour;

cin >> t.month;

cin >> t.second;

}

int show\_clock(void)

{

cout << t.hour << ":" << t.minute << ":" << t.second << endl;

}

2、建立一个名为Student的类，该类有以下几个私有成员变量：学生姓名、学号、性别、年龄。还有以下两个成员函数：

（1）SetStu：用于初始化学生姓名、学号、性别和年龄的构造函数；

（2）PrintStu：用于输出学生信息的函数。

编写一个主函数，创建一个学生对象，然后调用成员函数输入学生对象数据，并将该学生对象的信息在屏幕上输出。

3、定义一个名为Time 的时间类，包含时、分、秒等属性，对这些属性操作的方法有：

（1）printMilitary()：输出24小时制时间（如13:25 : 39）；

（2）printStandard()：输出标准格式时间（如 1 : 25 : 39 PM）

实现并测试这个类。

\*/

#include<iostream>

#include<string>

using namespace std;

class Clock{

private:

int hour;

int minute;

int second;

public:

void set\_clock() {

cin >> hour;

cin >> minute;

cin >> second;

}

void show\_clock() {

cout << hour << ":" << minute << ":" << second << endl;

}

};

class Student {

private:

string name;

string id;

string gender;

int age;

public:

void Setstu() {

cin >> name >> id >> gender >> age;

}

void PrintStu() {

cout << name << ' ' << id << ' ' << gender << ' ' << age << endl;

}

};

class Time {

private:

int hour;

int minute;

int second;

public:

void setTime() {

cin >> hour;

cin >> minute;

cin >> second;

}

void printMilitary() {

cout << hour << ':' << minute << ':' << second << endl;

}

void printStandard() {

cout << hour << ':' << minute << ':' << second;

if (hour < 12) {

cout << "AM" << endl;

}

else {

cout << "PM" << endl;

}

}

};

int main() {

cout << "第一题: " << endl;

Clock clock;

clock.set\_clock();

clock.show\_clock();

cout << "第一题： " << endl;

Student stu;

stu.Setstu();

stu.PrintStu();

cout << "第三题： " << endl;

Time time;

time.setTime();

time.printMilitary();

time.printStandard();

}

**实验5 类和对象（二）**

**一、实验目的**

1．掌握类的构造函数和析构函数的概念和使用方法。

2．初步掌握用类和对象编制基于对象的程序。

3．学习检查和调试基于对象的程序。

4．学习类的组合使用方法。

5．掌握对象数组、对象的指针及其使用方法。

6．掌握共用数据的保护方法

**二、实验内容**

1、检查下面的程序，找出其中的错误，并改正之。然后上机调试，使之能正常运行。

（1）

#include <iostream>

using namespace std;

class A

{public:

void A(int i=0){m=i;}

void show(){cout<<m;}

void ~A(){}

private:

int m;

};

int main()

{ A a(5);

a.m+=10;

a.show();

return 0;

}

（2）

class X

{ private:

int a=0;

int &b;

const int c;

void setA(int i){a=i;}

X(int i){ a=i;}

public:

int X(){ a=b=0;}

X(int i, int j, int k){ a=i; b=j; c=k; }

setC(int k) const { c=c+k;}

};

int main()

{

X x1;

X x2(2) ;

X x3(1,2,3) ;

x1.setA(3) ;

return 0 ;

}

2、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）

#include <iostream>

using namespace std;

class A {

public:

A();

A(int i,int j);

void print( );

private:

int a,b;

};

A::A( )

{

a=b=0;

cout<<”Default constructor called.\n”;

}

A::A(int i,int j)

{

a=i;

b=j;

cout<<”Constructor called.\n”;

}

void A::print()

{

cout<<”a=”<<a<<”,b=”<<b<<endl;

}

void main()

{

A m,n(4,8);

m.print();

n.print();

}

（2）

#include <iostream>

using namespace std;

class Date

{public:

Date(int,int,int);

Date(int,int);

Date(int);

Date( );

void display( );

private:

int month;

int day;

int year;

};

Date::Date(int m,int d,int y):month(m),day(d),year(y){ }

Date::Date(int m,int d):month(m),day(d) {year=2005;}

Date::Date(int m):month(m)

{ day=1;

year=2005;

}

Date::Date( )

{ month=1;

day=1;

year=2005;

}

void Date::display( )

{cout<<month<<"/"<<day<<"/"<<year<<endl;}

int main( )

{ Date d1(10,13,2005);

Date d2(12,30);

Date d3(10);

Date d4;

d1.display( );

d2.display( );

d3.display( );

d4.display( );

return 0;

}

（3）

#include <iostream>

using namespace std;

class test

{ public:

test( );

int getint( ){return num;}

float getfloat( ){return fl;}

~test( );

private:

int num;

float fl;

};

test::test( )

{ cout << "Initalizing default" << endl;

num=0;

fl=0.0;

}

test::~test( )

{

cout << "Destructor is active" << endl;

}

int main( )

{

test array[2];

cout << array[1].getint( )<< " " << array[1].getfloat( ) <<endl;

return 0;

}

（4）

#include<iostream>

using namespace std;

#include<string>

class X

{public:

X(int x1,char \*x2,float x3):a(x1),c(x3)

{ b=new char[sizeof(x2)+1];

strcpy(b,x2);

}

X():a(0),b("X::X()"),c(10){ }

X(int x1,char \*x2="X::X(....)",int x3=10):a(x1),b(x2),c(x3){}

X(const X&other)

{ a=other.a;

b="X::X(const X &other)";

c=other.c;

}

void print()

{ cout<<"a="<<a<<"\t"<<"b="<<b<<"\t"<<"c="<<c<<endl; }

private :

int a;

char \*b;

float c;

};

int main(){

X \*A=new X(4,"X::X(int,char,float)",32);

X B,C(10),D(B);

A->print();

B.print();

C.print();

D.print();

return 0;

}

3、某单位的职工工资包括基本工资Wage，岗位津贴Subsidy，房租Rent，水费WaterFee，电费ElecFee。设计并实现工资管理的类Salary，该类的形式如下：

class Salary

{public:

Salary(){初始化工资数据的各分项数据为0}

Salary(……) {初始化工资数据的各分项数据}

void setXX(double f){ XX=f; }

double getXX(){ return XX; }

double RealSalary(){ …… }//计算实发工资

……

Private:

Bouble Wage, Subsidy, Rent, WaterFee, ElecFee;

};

其中，成员函数setXX()用于设置工资的各分项数据，成员函数getXX()用于获取工资的各分项数据，XX代表Wage、Subsidy等数据成员，如Wage对应的成员函数则为setWage()和getWage()。

实发工资=Wage+Subsidy-Rent-WaterFee-ElecFee

编程完善该类的设计，并在主函数中测试该类的各成员函数。

4、设计一个时钟类Clock。数据成员包括hour（小时）、minute（分钟）、second（秒）。要求用成员函数实现以下功能：

（1）创建具有指定时钟（小时、分钟、秒）的Clock对象，默认时钟为00:00:00。

（2）动态地设置时、分、秒。

（3）在屏幕上按“时:分:秒”的格式显示时钟。

（4）在主函数中测试该类。

5、需要求3个三角形的体积，请编写一个基于对象的程序。数据成员包括三角形的三边长a、b、c。要求用成员函数实现以下功能：

(1) 定义构造函数完成三角形的初始化；

(2) 求三角形的周长；

(3) 求三角形的面积；

(4) 输出三角形信息。

6、建立一个对象数组，内放5个学生的数据（学号、成绩），设立一个函数max，用指向对象的指针作函数参数，在max函数中找出5个学生中成绩最高者，并输出其学号、成绩。

7、设计一个点类Point，再设计一个矩形类，矩形类使用Point类的两个坐标点作为矩形的对角顶点。并可以输出4个坐标值和面积。使用测试程序验证程序。

8、下面是一个整型数组类intArray的声明，请给出该类所有数据成员的类外定义。

class intArray

{public:

intArray(int size);//构造函数

intArray(const intArray &x);//复制构造函数

~intArray();//析构函数

bool Set(int i, int elem);//设置第i个数组元素的值，设置成功返回true，失败返回false

bool Get(int i, int &elem); //获取第i个数组元素的值，获取成功返回true，失败返回false

int Length( ) const;//获取数组的长度

void ReSize ( int size ); //重置数组

void Print();//输出数组

private:

int \*element; //指向动态数组的指针

int arraysize; //数组的当前长度

};

#### 9、声明一个复数类，complex，使用友元函数add实现复数加法。

10、（选做或课外练习）：习题4-8~4-14，4-19~4-20，5-3，5-7，5-13~5-14

**三、实验报告（按要求撰写实验报告）**

**时间要求：第四、五次实验课 完成**

**第一题：**

**（1）**

#include <iostream>

using namespace std;

class A

{

public:

A(int i = 0) { m = i; }

void show() { cout << m; }

~A() {}

private:

int m;

};

int main()

{

A a(5);

a.show();

return 0;

}

**（2）**

#include <iostream>

using namespace std;

class X

{

private:

int a = 0;

int\* b;

int c;

public:

X() { a = 0; if (b == nullptr) b = new int, \* b = 0; }

X(int i) { a = i; }

X(int i, int j, int k) { a = i; c = k; if (b == nullptr) b = new int, \* b = j; }

void setA(int i) { a = i; }

void setC(int k) { c = c + k; }

};

int main()

{

X x1;

X x2(2);

X x3(1, 2, 3);

x1.setA(3);

return 0;

}

**第二题：**

（1） 符合预期：m,n分别调用了类的有参和无参构造函数。

（2） 符合预期：d1,d2,d3,d4分别调用了类的有参和无参构造函数。

只不过构造函数的赋值语句没有写在函数体而是写在函数头上。

（3） 符合预期：main 函数里的test array[2] 相当于声明了两个test对象存放在array这个数组里，所以会出现两次构造和析构函数的调用。

（4）需将给出代码中的char\*替换成string否则编译不通过。原因是字符串形式为const char\*不能赋值给 char\* 修改过后则符合预期。

**第三题：**

#include<iostream>

using namespace std;

class Salary {

public:

Salary() {

Wage = Subsidy = Rent = WaterFee = ElecFee = 0;

}

Salary(double W,double S,double R,double WF,double EF):Wage(W),Subsidy(S),Rent(R),WaterFee(WF),ElecFee(EF){}

void setWage(double W) { Wage = W; }

void setSubsidy(double S) { Subsidy = S; }

void setRent(double R) { Rent = R; }

void setWaterFee(double WF) { WaterFee = WF; }

void setElecFee(double EF) { ElecFee = EF; }

double getWage(){ return Wage; }

double getSubsidy() { return Subsidy; }

double getRent() { return Rent; }

double getWaterFee() { return WaterFee; }

double getElecFee() { return ElecFee; }

double RealSalary() { return Wage + Subsidy - Rent - WaterFee - ElecFee; }

private:

double Wage, Subsidy, Rent, WaterFee, ElecFee;

};

int main() {

Salary s1;

Salary s2(10000, 2000, 4500, 50, 500);

s1.setWage(s2.getWage());

s1.setSubsidy(s2.getSubsidy());

s1.setRent(s2.getRent());

s1.setWaterFee(s2.getWaterFee());

s1.setElecFee(s2.getElecFee());

cout << s1.RealSalary() << endl;

cout << s2.RealSalary() << endl;

return 0;

}

**第四题：**

#include<iostream>

using namespace std;

class Clock {

private:

int hour;

int minute;

int second;

public:

Clock() { hour = minute = second = 0;}

Clock(int h,int m, int s):hour(h),minute(m),second(s){}

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

this->hour = h;

this->minute = m;

this->second = s;

}

void setHour(int h) { this->hour = h; }

void setMinute(int m) { this->minute = m; }

void setSecond(int s) { this->second = s; }

void showTime() { printf("%.2d:%.2d:%.2d\n", hour, minute, second); }

};

int main() {

Clock c1;

Clock c2;

c2.setHour(8);

c2.setMinute(6);

c2.setSecond(59);

c1.showTime();

c2.showTime();

return 0;

}

第五题：

#include<iostream>

using namespace std;

class Triangle {

private:

double a, b, c;

public:

Triangle() { a = b = c = 0; }

Triangle(double a, double b, double c) {

this->a = a;

this->b = b;

this->c = c;

}

double perimeter() { return a + b + c; }

double area() {

double p = (a + b + c) / 2;

return sqrt(p \* (p - a) \* (p - b) \* (p - c));

}

void printMessage() {

cout << "三角形边长为：" << a << " " << b << " " << c << endl;

cout << "三角形周长为：" << perimeter() << endl;

cout << "三角形面积为：" << area() << endl;

}

};

int main() {

Triangle t1(3, 4, 5);

t1.printMessage();

return 0;

}

**第六题：**

#include<iostream>

using namespace std;

class Student {

private:

long id;

double grade;

public:

Student() {

this->grade = 0;

this->id = 0;

}

Student(long id, double grade) {

this->id = id;

this->grade = grade;

}

void setGrade(double grade) { this->grade = grade; }

void setId(long id) { this->id = id; }

double getGrade() { return grade; }

long getId() { return id; }

};

void max(Student\* student\_array) {

int n = sizeof(student\_array);

double max\_grade = 0;

int max\_index = 0;

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

if(max\_grade < student\_array[i].getGrade()){

max\_grade = student\_array[i].getGrade();

max\_index = i;

}

}

cout << "学号：" << student\_array[max\_index].getId() << endl;

cout << "成绩：" << student\_array[max\_index].getGrade();

}

int main() {

double grade[5] = { 90.8,88.6,95.0,96.9,99.8 };

Student students[5];

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

students[i].setGrade(grade[i]);

students[i].setId(i);

}

max(students);

return 0;

}

**第七题：**

#include<iostream>

using namespace std;

class Point {

private:

double x;

double y;

public:

Point() {

this->x = 0;

this->y = 0;

}

Point(double x, double y) {

this->x = x;

this->y = y;

}

void setX(double x) { this->x = x; }

void setY(double y) { this->y = y; }

double getX() { return x; }

double getY() { return y; }

};

class Rectangle {

private:

Point p1, p2;

public:

Rectangle(Point p1, Point p2) {

this->p1 = p1;

this->p2 = p2;

}

double area() {

double x = abs(p1.getX() - p2.getX());

double y = abs(p1.getY() - p2.getY());

return x \* y;

}

void printMessage() {

double x1 = p1.getX();

double x2 = p2.getX();

double y1 = p1.getY();

double y2 = p2.getY();

cout << "(" << x1 << "," << y1 << ")" << " ";

cout << "(" << x1 << "," << y2 << ")" << " ";

cout << "(" << x2 << "," << y1 << ")" << " ";

cout << "(" << x2 << "," << y2 << ")" << " ";

cout << endl << "矩形面积为：" << area() << endl;

}

};

int main() {

Point p1(3, 5);

Point p2(2, 4);

Rectangle rectangle(p1, p2);

rectangle.printMessage();

return 0;

}

**实验6 类的继承**

**一、实验目的**

1．了解继承在面向对象程序设计中的重要作用。

2．理解继承与派生的概念。

3．掌握通过继承派生出一个新的类的方法。

**二、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。

(1)

#include <iostream>

using namespace std;

class A

{public:

A(){cout<<"A::A() called.\n";}

virtual ~A(){cout<<"A::~A() called.\n";}

};

class B:public A

{ public:

B(int i)

{ cout<<"B::B() called.\n";

buf=new char[i];

}

virtual ~B()

{ delete []buf;

cout<<"B::~B() called.\n";

}

private:

char \*buf;

};

void fun(A \*a)

{ cout<<"May you succeed!";

delete a;

}

int main()

{

A \*a=new B(15);

fun(a);

return 0;

}

(2)

#include<iostream>

using namespace std;

class A{

public:

A(int a):x(a){ cout<<"A constructor..."<<x<<endl; }

int f(){return ++x;}

~A(){cout<<"destructor A..."<<endl;}

private:

int x;

};

class B:public virtual A{

private:

int y;

A Aobj;

public:

B(int a,int b,int c):A(a),y(c),Aobj(c){ cout<<"B constructor..."<<y<<endl;}

int f(){

A::f();

Aobj.f();

return ++y;

}

void display(){ cout<<A::f()<<"\t"<<Aobj.f()<<"\t"<<f()<<endl; }

~B(){cout<<"destructor B..."<<endl;}

};

class C:public B{

public:

C(int a,int b,int c):B(a,b,c),A(0){ cout<<"C constructor..."<<endl;}

};

class D:public C,public virtual A{

public:

D(int a,int b,int c):C(a,b,c),A(c){ cout<<"D constructor..."<<endl;}

~D(){cout<<"destructor D...."<<endl;}

};

int main()

{

D d(7,8,9);

d.f();

d.display();

return 0;

}

(3)

#include<iostream>

using namespace std;

class A{

public:

A(int a,int b):x(a),y(b){ cout<<"A constructor..."<<endl; }

void Add(int a,int b){ x+=a;y+=b;}

void display(){ cout<<"("<<x<<","<<y<<")";}

~A(){cout<<"destructor A..."<<endl;}

private:

int x,y;

};

class B:private A{

private:

int i,j;

A Aobj;

public:

B(int a,int b,int c,int d):A(a,b),i(c),j(d) ,Aobj(1,1){ cout<<"B constructor..."<<endl;}

void Add(int x1,int y1,int x2,int y2)

{

A::Add(x1,y1);

i+=x2; j+=y2;

}

void display(){

A::display();

Aobj.display();

cout<<"("<<i<<","<<j<<")"<<endl;

}

~B(){cout<<"destructor B..."<<endl;}

};

int main()

{

B b(1,2,3,4);

b.display();

b.Add(1,3,5,7);

b.display();

return 0;

}

(4)

#include <iostream>

using namespace std;

class Base1

{

public:

Base1()

{

cout<<"class Base1!"<<endl;

}

};

class Base2

{

public:

Base2()

{

cout<<"class Base2!"<<endl;

}

};

class Level1:public Base2,virtual public Base1

{

public:

Level1()

{

cout<<"class Level1!"<<endl;

}

};

class Level2: public Base2,virtual public Base1

{

public:

Level2()

{

cout<<"class Level2!"<<endl;

}

};

class TopLevel:public Level1,virtual public Level2

{

public:

TopLevel()

{

cout<<"class TopLevel!"<<endl;

}

};

int main()

{

TopLevel obj;

return 0;

}

2、某出版系统发行图书和磁带，利用继承设计管理出版物的类。要求如下：建立一个基类Publication存储出版物的标题title、出版物名称name、单价price及出版日期date。用Book类和Tape类分别管理图书和磁带，它们都从Publication类派生。Book类具有保存图书页数的数据成员page，Tape类具有保存播放时间的数据成员playtime。每个类都有构造函数、析构函数，且都有用于从键盘获取数据的成员函数inputData()，用于显示数据的成员函数display()。

4、按下列要求编写程序。

（1）定义一个分数类Score。它有3个数据成员：

Chinese //语文课成绩

English //英语课成绩

Mathematics //数学课成绩

2个构造函数：无参的和带参数的

3个成员函数：是否带参数根据需要自定

sum() //计算三门课总成绩

print() //输出三门课成绩和总成绩

modify() //修改三门课成绩

（2）定义一个学生类Student。它有3个数据成员：

Num //学号

Name //姓名

MyScore //成绩

2个构造函数：无参的和带参数的

3个成员函数：是否带参数根据需要自定

sum() //计算某学生三门课总成绩

print() //输出某学生学号、姓名和成绩

modify() //修改某学生学号、姓名和成绩

（3）在主函数中，先定义一个学生类对象数组，再通过for循环给对象数组赋上实际值，最后输出对象数组个元素的值。

4、编写一个程序实现小型公司的人员信息管理系统。该公司雇员（employee）包括经理（manager），技术人员（technician）、销售员（salesman）和销售部经理（salesmanager）。要求存储这些人员的姓名、编号、级别、当月薪水，计算月薪并显示全部信息。

程序要对所有人员有提升级别的功能。为简单起见，所有人员的初始级别均为1，然后进行升级，经理升为4级，技术人员和销售部经理升为3级，销售员仍为1级。

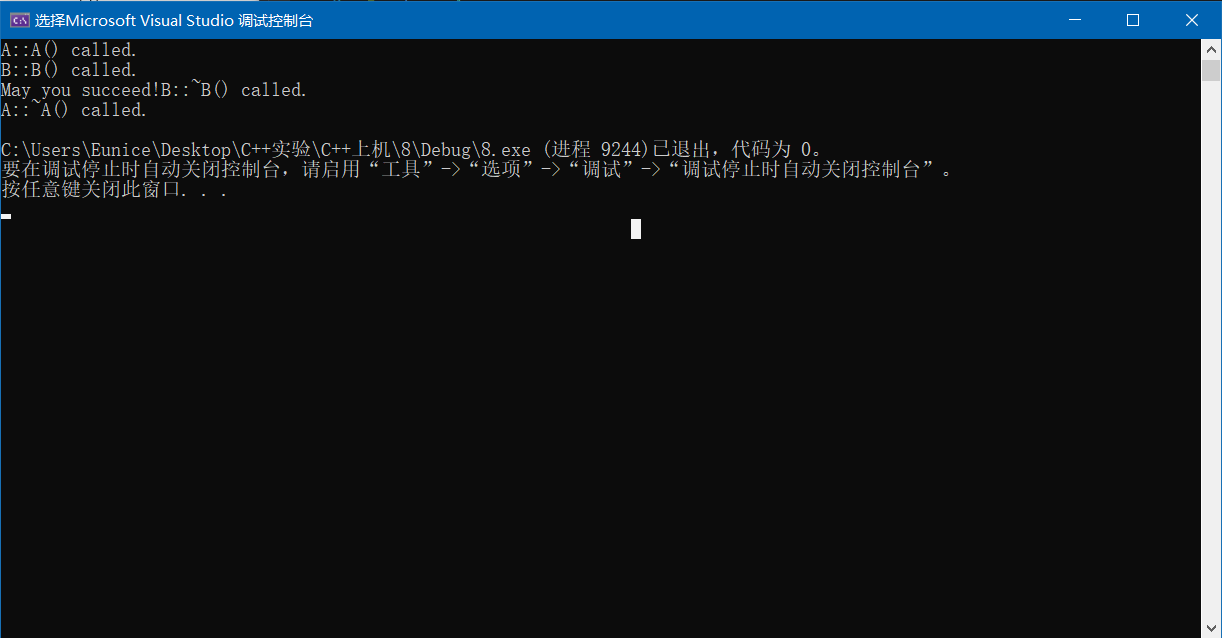
月薪计算办法是：经理拿固定月薪8000元，技术人员按每小时100元领取月薪，销售员按该当月销售额4%提成，销售经理既拿固定月工资也领取销售提成，固定月工资为5000元，销售提成为所管辖部门当月销售额的5‰。

5、（选做或课外练习）：习题7-5~7-11，7-13

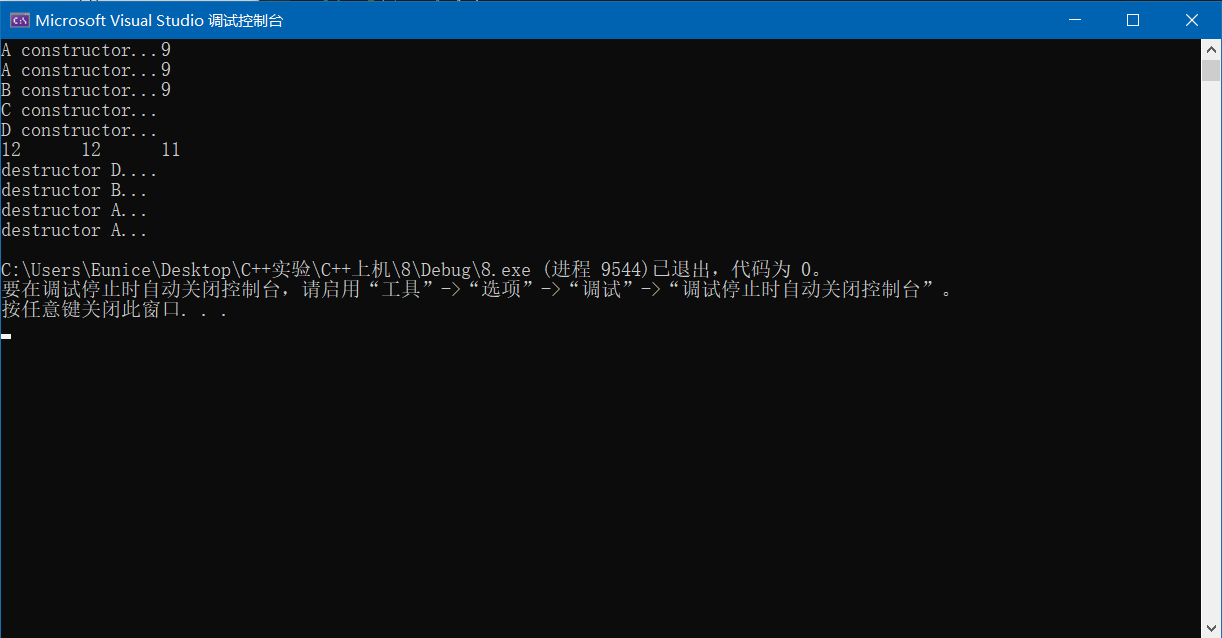
**三、实验报告（按要求撰写实验报告）**

**第一题：**

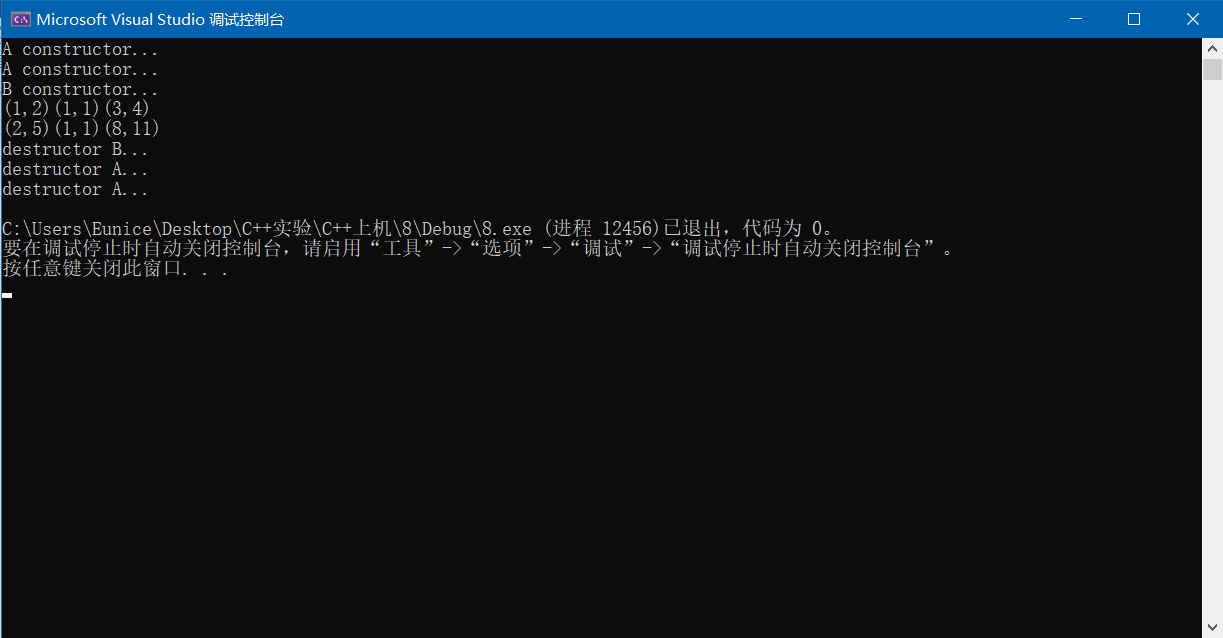
**(1)**

****

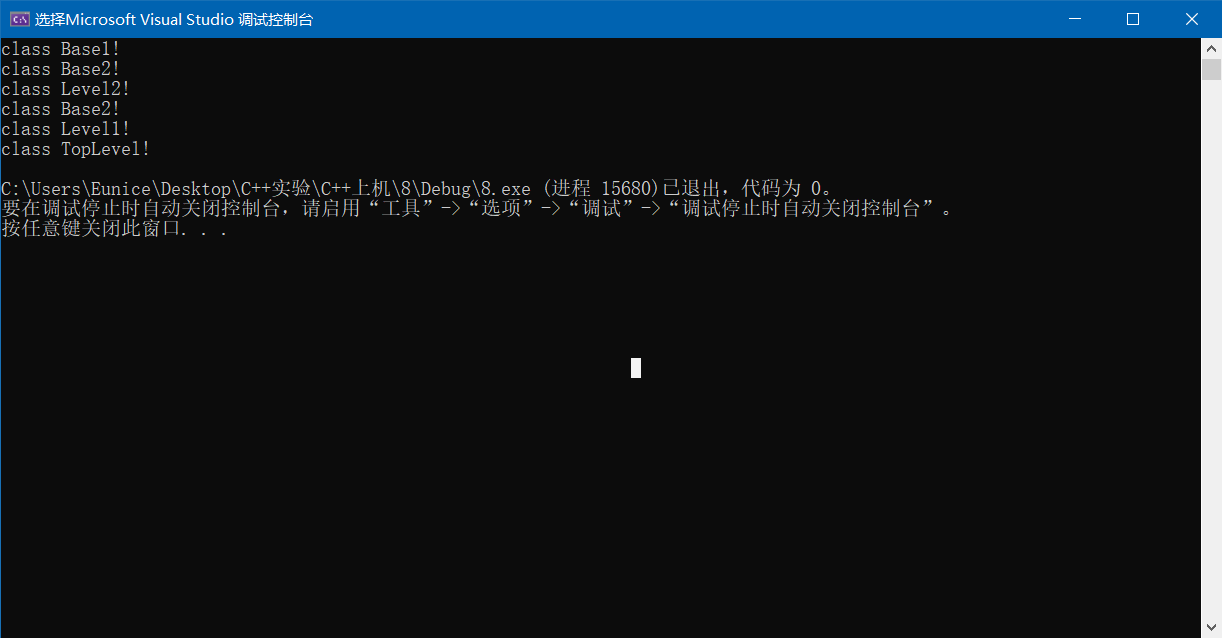
**（2）**

****

**（3）**

****

**（4）**



**第二题：**

#include<iostream>

#include<string.h>

using namespace std;

class Publication {

private:

char title[8];

char name[8];

int price;

int date;

public:

Publication() {};

void inputData();

void display();

~Publication() {};

};

class Book :public Publication {

private:

int page;

public:

Book() {};

void inputData();

void display() {

cout << endl;

Publication::display();

cout << page << endl;

}

~Book() {};

};

class Tape :public Publication {

private:

int playtime;

public:

Tape() {};

void inputData();

void display() {

Publication::display();

cout << playtime << endl;

}

~Tape() {};

};

void Tape::inputData() {

cout << "请输入保存时间、出版物标题、名称、单价及出版日期" << endl;

cin >> playtime;

Publication::inputData();

cout << endl;

}

void Book::inputData() {

cout << "请输入页数、出版物标题、名称、单价及出版日期" << endl;

cin >> page;

Publication::inputData();

cout << endl;

}

void Publication::inputData()

{

cout << "请输入出版物标题、名称、单价及出版日期" << endl;

cin >> title >> name >> price >> date;

cout << endl;

}

void Publication::display()

{

cout << title << " " << name << " " << price << " " << date << " ";

}

void main()

{

Publication p;

p.inputData();

p.display();

cout << endl;

Book b;

b.inputData();

b.display();

Tape t;

t.inputData();

t.display();

}

**第三题：**

#include<iostream>

using namespace std;

#include<string>

class Score {

public:

Score() {};

Score(double a, double b, double c)

{

Chinese = a; English = b; Mathematics = c;

}

double sum();

void print();

void modify(double a, double b, double c);

private:

double Chinese, English, Mathematics;

};

double Score::sum()

{

return (Chinese + English + Mathematics);

}

void Score::print()

{

cout << "语文课成绩为:" << Chinese << endl;

cout << "英语课成绩为:" << English << endl;

cout << "数学课成绩为:" << Mathematics << endl;

cout << "总成绩为:" << sum() << endl;

}

void Score::modify(double a, double b, double c)

{

Chinese = a; English = b; Mathematics = c;

}

class Student {

public:

Student() {};

Student(string a, string b, double c, double d, double e) :MyScore(c, d, e)

{

Num = a; Name = b;

}

double sum();

void print();

void modify(string a, string b, double c, double d, double e);

private:

string Num, Name;

Score MyScore;

};

double Student::sum()

{

return (MyScore.sum());

}

void Student::print()

{

cout << "学号为:" << Num << endl;

cout << "姓名为:" << Name << endl;

MyScore.print();

}

void Student::modify(string a, string b, double c, double d, double e)

{

Num = a; Name = b; MyScore.modify(c, d, e);

}

int main()

{

int size;

cout << "人数:";

cin >> size;

string Num, Name;

double Chinese, English, Mathematics;

Student s[50];

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

{

cout << "请输入第" << i + 1 << "位同学的信息" << endl;

cout << "学号:";

cin >> Num;

cout << "姓名:";

cin >> Name;

cout << "语文课成绩:";

cin >> Chinese;

cout << "英语课成绩:";

cin >> English;

cout << "数学课成绩:";

cin >> Mathematics;

s[i].modify(Num, Name, Chinese, English, Mathematics);

}

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

s[i].print();

return 0;

}

**第四题：**

//编写一个程序实现小型公司的人员信息管理系统。

//该公司雇员（employee）包括经理（manager），技术人员（technician）、销售员（salesman）和销售部经理（salesmanager）。

//要求存储这些人员的姓名、编号、级别、当月薪水，计算月薪并显示全部信息。

//程序要对所有人员有提升级别的功能。为简单起见，所有人员的初始级别均为1，

//然后进行升级，经理升为4级，技术人员和销售部经理升为3级，销售员仍为1级。

//月薪计算办法是：经理拿固定月薪8000元，

//技术人员按每小时100元领取月薪，销售员按该当月销售额4% 提成，

//销售经理既拿固定月工资也领取销售提成，固定月工资为5000元，

//销售提成为所管辖部门当月销售额的5‰

#include<iostream>

#include<string>

using namespace std;

class Employee { //虚基类

public:

Employee() {

cout << "职工编号";

cin >> ID;

cout << "职工姓名";

cin >> name;

salary = 0; //月薪

}

virtual void pay() = 0;//月薪计算函数

virtual void show() = 0;

protected:

string name; //姓名

int ID; //职工编号

double salary; //月薪

};

class Technician :virtual public Employee {

public:

Technician() { perhour = 20; } //每小时附加酬金

void pay() {

cout << "请输入技术工人本月工作时数：\n";

cin >> hours;

salary = perhour \* hours;

}

void show() {

cout << "技术人员" << name << "(编号为" << ID << ")" << "本月工资：" << salary << endl;

}

protected:

double hours; //月工作时数

double perhour; //月每小时附加时数

};

class manager :virtual public Employee {

public:

manager() { monthpay = 8000; }

void pay() { salary = monthpay; }

void show() { cout << "经理" << name << "(编号为" << ID << ")" << "本月工资：" << salary << endl; }

protected:

double monthpay; //固定工资

};

class salesman :virtual public Employee {

public:

salesman() { slfactor = 0.004; }//提成比例

void pay() {

cout << "请输入销售员本月销售额：\n";

cin >> amount;

salary = amount \* slfactor;

}

void show() { cout << "销售员" << name << "(编号为" << ID << ")" << "本月工资：" << salary << endl; }

protected:

double amount; //月销售额

double slfactor; //提成比例

};

class salesmanager :public manager, public salesman {

public:

salesmanager() {

slfactor = 0.005; //提成比例

monthpay = 5000;

}

void pay() {

cout << name << "请输入销售经理所管部门本月销售额：";

cin >> amount;

salary = monthpay + amount \* slfactor;

}

void show() {

cout << "销售经理" << name << "(编号为" << ID << ")" << "本月工资：" << salary << endl;

}

};

int main() {

Employee\* p = NULL;

cout << "经理的";

manager magl;

cout << "技术人员的";

Technician tecl;

cout << "销售人员的";

salesman sall;

cout << "销售经理的";

salesmanager saml;

p = &magl;

p->pay();

p->show();

p = &tecl;

p->pay();

p->show();

p = &sall;

p->pay();

p->show();

p = &saml;

p->pay();

p->show();

return 0;

}

**时间要求：第六、七次实验课 完成**

**实验7 多态性和运算符重载**

**一、实验目的**

1．掌握使用虚函数实现多态性；

2．掌握运算符重载的方法。

**五、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。

（1）

#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);

Base Bobj=Dobj;

Bobj.g1();

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

Base \*bp=&Dobj;

bp->g1();

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

Base &bobj2=Dobj;

bobj2.g1();

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

Dobj.g1();

return 0;

}

（2）

#include<iostream.h>

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;

}

(3)

#include<iostream.h>

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;

}

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。

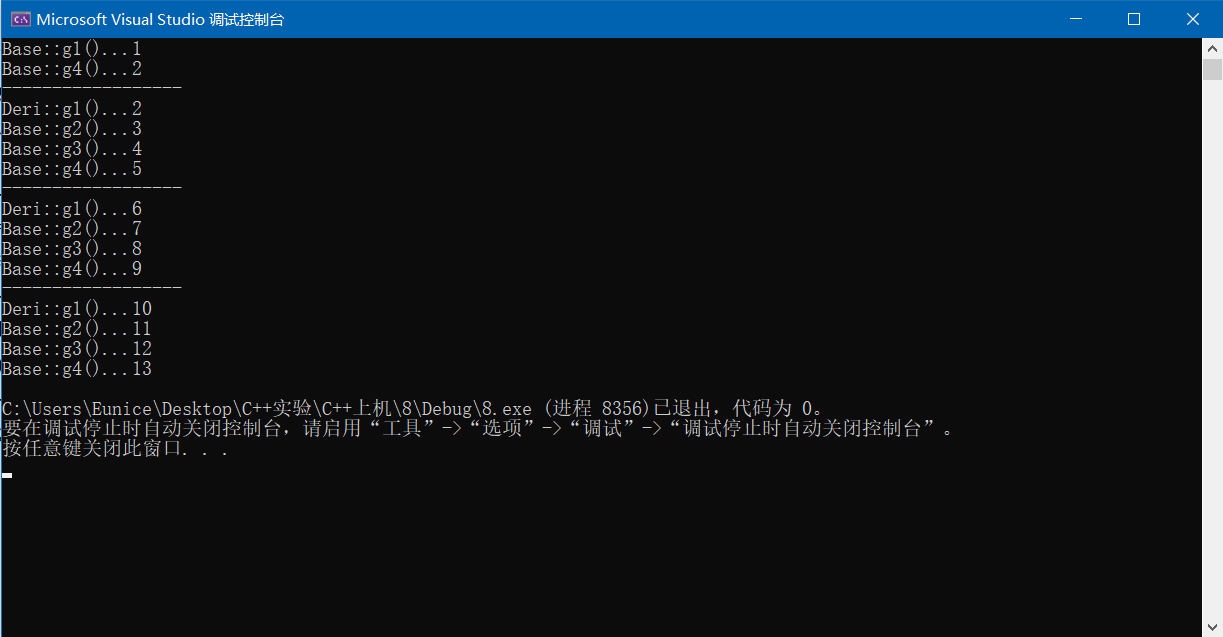
5、（选做或课外练习）：习题8-4~8-11

**三、实验报告（按要求撰写实验报告）**

**时间要求：第八次实验课 完成**

**第一题：**

**（1）**

****

**（2）**

#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;

}

**（3）**

#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;

}

**第二题：**

#include<iostream>

using namespace std;

class Point

{

public:

Point(float x = 0, float y = 0);

void setPoint(float, float);

float getX() const

{

return x;

}

float getY() const

{

return y;

}

friend ostream& operator<<(ostream&, const Point&);

protected:

float x, y;

};

Point::Point(float a, float b)

{

x = a;

y = b;

}

void Point::setPoint(float a, float b)

{

x = a;

y = b;

}

ostream& operator<<(ostream& output, const Point& p)

{

output << "[" << p.x << "," << p.y << "]" << endl;

return output;

}

class Circle :public Point

{

public:

Circle(float x = 0, float y = 0, float r = 0);

void setRadius(float);

float getRadius() const;

float area() const;

friend ostream& operator<<(ostream&, const Circle&);

protected:

float radius;

};

Circle::Circle(float a, float b, float r) :Point(a, b), radius(r)

{

}

void Circle::setRadius(float r)

{

radius = r;

}

float Circle::getRadius() const

{

return radius;

}

float Circle::area() const

{

return float(3.14159 \* radius \* radius);

}

ostream& operator<<(ostream& output, const Circle& c)

{

output << "Center=[" << c.x << "," << c.y << "],r=" << c.radius << ",area=" << c.area() << endl;

return output;

}

class Cylinder :public Circle

{

public:

Cylinder(float x = 0, float y = 0, float r = 0, float h = 0);

void setHeight(float);

float getHeight() const;

float area() const;

float volume() const;

friend ostream& operator<<(ostream&, const Cylinder&);

protected:

float height;

};

Cylinder::Cylinder(float a, float b, float r, float h) :Circle(a, b, r), height(h)

{

}

void Cylinder::setHeight(float h)

{

height = h;

}

float Cylinder::getHeight() const

{

return height;

}

float Cylinder::area() const

{

return float(2 \* Circle::area() + 2 \* 3.14159 \* radius \* height);

}

float Cylinder::volume() const

{

return Circle::area() \* height;

}

ostream& operator<<(ostream& output, const Cylinder& cy)

{

output << "Center=[" << cy.x << "," << cy.y << "],r=" << cy.radius << ",h=" << cy.height << "\narea=" << cy.area() << ",volume=" << cy.volume() << endl;

return output;

}

int main()

{

Cylinder cy1(3.5f, 6.4f, 5.2f, 10);

cout << "original circle:\nx=" << cy1.getX() << ",y=" << cy1.getY() << ",r=" << cy1.getRadius() << ",h=" << cy1.getHeight() << "\narea=" << cy1.area() << ",volume=" << cy1.volume() << endl;

cy1.setHeight(15);

cy1.setRadius(7.5);

cy1.setPoint(5, 5);

cout << "\nnew circle:\n" << cy1;

Point& pRef = cy1;

cout << "\npRef as a point:" << pRef;

Circle& cRef = cy1;

cout << "\ncRef as a Circle:" << cRef;

return 0;

}

**第四题：**

#include <iostream>

using namespace std;

class Date

{

public:

Date():year(2012), month(1), day(1) {}

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

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

this->year = year;

this->month = month;

this->day = day;

}

void setYear(int year) {

this->year = year;

}

void setMonth(int month) {

this->month = month;

}

void setDay(int day) {

this->day = day;

}

void printDate() const {

cout << "当前日期为：";

printf("%d年%2d月%2d日\n", year, month, day);

}

bool operator==(const Date& A) const {

return date\_to\_code() == A.date\_to\_code();

}

bool operator!=(const Date& A) const {

return date\_to\_code() != A.date\_to\_code();

}

bool operator>(const Date& A) const {

return date\_to\_code() > A.date\_to\_code();

}

bool operator<(const Date& A) const {

return date\_to\_code() < A.date\_to\_code();

}

void operator++() {

day++;

if (!is\_day\_legal()) {

day = 1;

if (month == 12) {

year++;

month = 1;

}

else {

month++;

}

}

}

void operator--() {

day--;

if (day == 0) {

day = get\_max\_day(month-1);

if (month == 1) {

year--;

month = 12;

}

else {

month--;

}

}

}

void operator+=(int num) {

if (num < 0) {

operator-=(-num);

return;

}

while (num > 365) {

if (month > 2 || month == 2 && day == day\_list[2]) {

year++;

if (is\_leap\_year()) {

num -= 366;

}

else {

num -= 365;

}

}

else {

if (is\_leap\_year()) {

num -= 366;

}

else {

num -= 365;

}

year++;

}

}

while (num > 31) {

num -= get\_max\_day(month);

if (month == 12) {

month = 1;

year++;

}

else {

month++;

}

}

while (num > 0) {

operator++();

num--;

}

}

void operator-=(int num) {

if (num < 0) {

operator+=(-num);

return;

}

while (num > 365) {

if (month > 2 || month == 2 && day == day\_list[2]) {

if (is\_leap\_year()) {

num -= 366;

}

else {

num -= 365;

}

year--;

}

else {

year--;

if (is\_leap\_year()) {

num -= 366;

}

else {

num -= 365;

}

}

}

while (num > 31) {

num -= get\_max\_day(month-1);

if (month == 0) {

month = 12;

year--;

}

else {

month--;

}

}

while (num > 0) {

operator--();

num--;

}

}

private:

int year, month, day;

int day\_list[13] = {31, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

int date\_to\_code() const {

return day + month \* 32 + year \* 385;

}

bool is\_leap\_year() const {

return year % 400 == 0 || year % 4 == 0 && year % 100 != 0;

}

int get\_max\_day(int month) {

if (is\_leap\_year()) {

day\_list[2] = 29;

}

else {

day\_list[2] = 28;

}

return day\_list[month];

}

bool is\_day\_legal() {

return day <= get\_max\_day(month);

}

};

void testOperatorEqual() {

cout << endl << ">>test operator==() and operator!=()" << endl;

bool flag = true;

for (int i = 0; i < 9999; i++) {

int year = rand() % 2019;

int month = rand() % 12 + 1;

int day = rand() % 28 + 1;

Date date1(year, month, day);

Date date2(year, month, day);

if (date1 != date2 && !(date1 == date2)) {

cout << "Error!";

flag = false;

}

}

for (int i = 0; i < 9999; i++) {

int year1 = rand() % 2019;

int year2 = rand() % 2019;

int month = rand() % 12 + 1;

int day = rand() % 28 + 1;

Date date1(year1, month, day);

Date date2(year2, month, day);

if (year1 != year2 && date1 == date2 && !(date1 != date2)) {

cout << "Error!";

flag = false;

}

}

for (int i = 0; i < 9999; i++) {

int year = rand() % 2019;

int month1 = rand() % 12 + 1;

int month2 = rand() % 12 + 1;

int day = rand() % 28 + 1;

Date date1(year, month1, day);

Date date2(year, month2, day);

if (month1 != month2 && date1 == date2 && !(date1 != date2)) {

cout << "Error!";

flag = false;

}

}

for (int i = 0; i < 9999; i++) {

int year = rand() % 2019;

int month = rand() % 12 + 1;

int day1 = rand() % 28 + 1;

int day2 = rand() % 28 + 1;

Date date1(year, month, day1);

Date date2(year, month, day2);

if (day1!= day2 && date1 == date2 && !(date1 != date2)) {

cout << "Error!";

flag = false;

}

}

if (flag) {

cout << ">>test passed!" << endl;

}

}

void testOperatorAdd1() {

cout << endl << ">>test operator++() and operator--():" << endl;

Date date(2019, 12, 1);

for (int i = 0; i < 366; i++) {

++date;

}

Date supposedDate(2020, 12, 1);

if (date == supposedDate) {

cout << ">>operator++() passed!" << endl;

}

for (int i = 0; i < 366 + 365; i++) {

--date;

}

supposedDate.setDate(2018, 12, 1);

if (date == supposedDate) {

cout << ">>operator--() passed!" << endl;

}

}

void testOperatorAddMany() {

cout << endl << ">>test operator+=() and operator-=():" << endl;

bool flag = true;

for (int i = 0; i < 99; i++) {

int rand\_num1 = rand();

Date actualDate(2019, 12, 1);

Date supposedDate(2019, 12, 1);

for (int j = 0; j < rand\_num1; j++) {

++actualDate;

}

supposedDate += rand\_num1;

if (actualDate != supposedDate) {

flag = false;

}

}

for (int i = 0; i < 99; i++) {

int rand\_num1 = rand();

Date actualDate(2019, 12, 1);

Date supposedDate(2019, 12, 1);

for (int j = 0; j < rand\_num1; j++) {

--actualDate;

}

supposedDate -= rand\_num1;

if (actualDate != supposedDate) {

flag = false;

}

}

if (flag) {

cout << "test passed!" << endl;

}

}

void testOperatorLarger() {

cout << endl << ">>test operator>() and operator<()" << endl;

bool flag = true;

for (int i = 0; i < 9999; i++) {

int rand\_num1 = rand();

int rand\_num2 = rand();

bool suppose = rand\_num1 > rand\_num2;

Date date1 = Date(2019, 12, 1);

Date date2 = Date(2019, 12, 1);

date1 += rand\_num1;

date2 += rand\_num2;

bool actual = date1 > date2;

if (suppose != actual) {

flag = false;

}

}

for (int i = 0; i < 9999; i++) {

int rand\_num1 = rand();

int rand\_num2 = rand();

bool suppose = rand\_num1 < rand\_num2;

Date date1 = Date(2019, 12, 1);

Date date2 = Date(2019, 12, 1);

date1 += rand\_num1;

date2 += rand\_num2;

bool actual = date1 < date2;

if (suppose != actual) {

flag = false;

}

}

if (flag) {

cout << "test passed!" << endl;

}

}

int main()

{

testOperatorEqual();

testOperatorAdd1();

testOperatorAddMany();

testOperatorLarger();

## }

**实验8 流类库与输入/输出**

### 一、实验目的

1．深入理解C++的输入输出的含义与其实现方法。

2．掌握标准输入输出流的应用，包括格式输入输出。

3．掌握对文件的输入输出操作。

### 二、实验内容

1．阅读下面的程序，写出程序运行的结果。

（1）

#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;

}

(2)

#include<iostream>

#include<string>

#include<fstream>

using namespace std;

class Worker{

private:

int number ,age;

char name[20];

double sal;

public:

Worker(){}

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

{strcpy(name,Name); }

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

};

int main(){

ofstream out("Worker.dat",ios::out|ios::binary);

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("Employee.dat",ios::in|ios::binary);

in.seekg(2\*(sizeof(s1)),ios::beg);

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;

}

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。

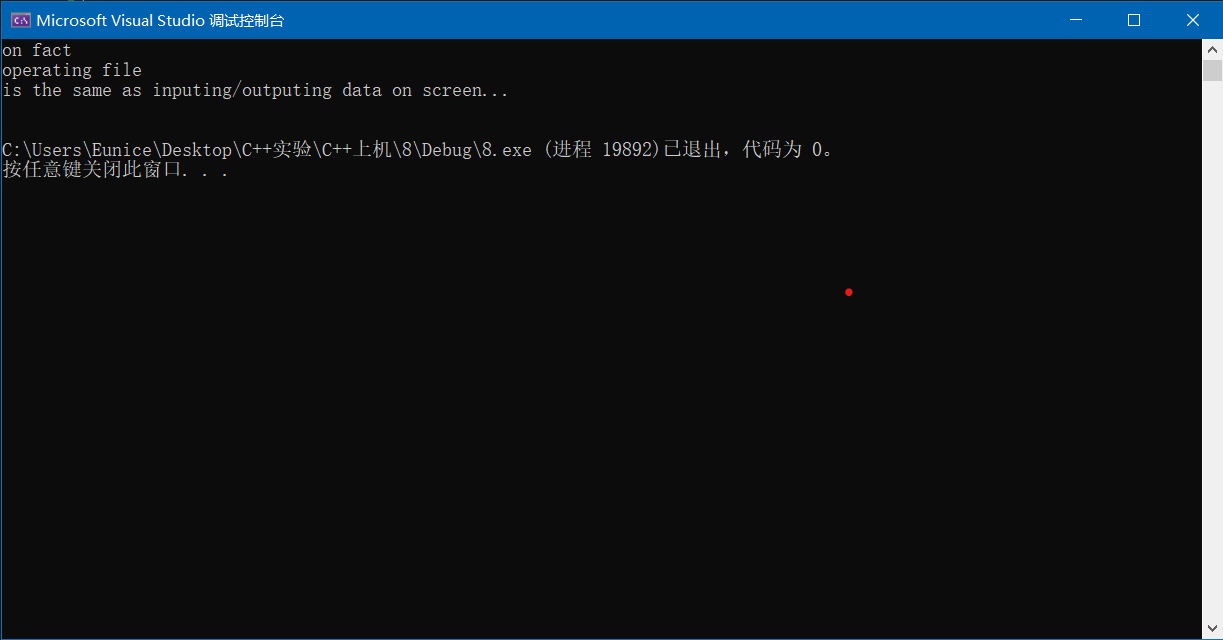
成员函数putdata(int &,int &)的功能为：把求得的数据成员A[]中20个整数的最大值和最小值输出到数据文件OUT.DAT。

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。

**第一题：**

**（1）**



**(2)**

3 "王五" 34 2322  
1 "张三" 23 2320

**第二题：**

// 8-2.cpp : 此文件包含 "main" 函数。程序执行将在此处开始并结束。

//

#include<iostream>

#include<string>

#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();//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

};

void Array::getdata()

{

fstream scan;

scan.open("IN.DAT", ios::in);

for (size\_t i = 0; i < 20; i++)

{

scan >> A[i];

}

scan.close();

}

void Array::max\_min(int& max, int& min)//排序函数

{

max = INT\_MIN;

min = INT\_MAX;

for (size\_t i = 0; i < 20; i++)

{

max = max > A[i] ? max : A[i];

min = min < A[i] ? min : A[i];

}

}

void Array::putdata(int& min, int& max)//写结果数据函数

{

fstream outfile;

outfile.open("OUT.DAT", ios::out);

max = INT\_MIN;

min = INT\_MAX;

for (size\_t i = 0; i < 20; i++)

{

max = max > this->A[i] ? max : A[i];

min = min < A[i] ? min : A[i];

}

outfile << max << " " << min;

outfile.close();

}

int main() {

Array arr;

arr.getdata();

int maxn, minn;

cout << arr.getNumberA(arr, 1);

arr.putdata(minn, maxn);

system("pause");

return 0;

}

**第三题：**

#include<iostream>

#include<string>

#include<fstream>

using namespace std;

class Student

{

private:

string name;

string school;

string major;

int class\_num;

string phone\_num;

string address;

string email\_num;

public:

void logging();

void output();

void scan();

};

void Student::logging()

{

cout << "请输入学生姓名、学校、专业、班级、电话号码、通信地址、邮政编码" << endl;

cin >> name >> school >> major >> class\_num >> phone\_num >> address >> email\_num;

}

void Student::output()

{

fstream outfile;

outfile.open("address.dat", ios::app);

outfile << name << " " << school << major << class\_num << endl;

outfile << "电话号码为：" << phone\_num << endl;

outfile << "通信地址为：" << address << endl;

outfile << "邮政编码为：" << email\_num << endl;

outfile.close();

}

void Student::scan()

{

fstream infile;

string str;

infile.open("address.dat", ios::in);

for (int j = 0; j < 10; j++)

{

for (int i = 0; i < 5; i++)

{

infile >> str;

cout << str << endl;

}

}

infile.close();

}

int main()

{

Student str[10];

for (int i = 0; i < 10; i++)

{

str[i].logging();

str[i].output();

}

str[0].scan();

system("pause");

}