研榜考研

整理 by 研榜考研丁老师 VX：18751931817

后续编订：

数据结构部分试题
数电模电部分试题

C++程序设计

百科园题库-整理版

1-5 章大题没有答案
6-10 章内容齐全
整理匆忙，难免存在错误

第一章

1.面向对象程序设计的主要优点是（）。
其他 3 个选项都是

2.面向对象方法中，实现对象的数据和操作结合于统一体中的是（）。
封装

3.下面选项中不属于面向对象程序设计特征的是（）。
类比性

4.下列有关类和对象的说法中，不正确的是（）。
一个类只能有一个对象

5.所谓数据封装就是将一组数据和与这组数据有关操作组装在一起，形成一个实体，这实体也就是（）。
类

6.在 C++中用类将数据和对数据操作的代码连接在一起称为（）
封装

7.下面概念中，不属于面向对象方法的是（）。
过程调用

第二章

1.字面常量 42、4.2、42L 的数据类型分别是（）。

int,double,long

2.有如下程序：

```
 Int x=3;
 Do {
  X = -2;
  Cout<<x;
}while(!(--x)) ;
```

执行这个程序的输出结果是()。

1 -2

3.设有定义 int i:double j=5:,则 10+i+j 值的数据类型是()。

Double

4.下列字符串中，不可以用做 C+标识符的是（）。

Switch

下面有关 for 循环的正确描述是（）。

在 for 循环的循环体语句中，可以包含多条语句，但必须用花括号括起来

6.判断 char 型变量 cl 是否小写字母的正确表达式为()。

(c1>='a)&&(c1<='z)

7.以下关于 C+语言标识符的描述中，正确的是()

Area 与 area 是不同的标识符

8.有如下程序：

```
#include <iostream>
using namespace std;
int main() {
      int f, f1=0, f2=1;
      for(int i=3; i<=6; i++) {
            f=f1+f2;
            f1=f2; f2=f;
}
cout<<f<<end1;
return 0;
}
```

运行时的输出结果是()

5

9.有如下程序：

```
#include <iostream>
using namespace std;
int main() {
 for(int i=1; i<=50; i++)  {
  if(i%3 != 0)
   continue;
  else
   if(i%5!=0)
    continue;
  cout<<i; break;
}
 return 0;
}
```

15

第三章

1.已知函数 func 的原型是：double func(double *pd, int &ri):

变量 x 和 y 的定义是：double x:inty:

把 x 和 y 分别作为第一参数和第二参数来调用 func,正确的调用语句是（）

func(&x,y);

2.在函数说明中，不必要的是（）

函数参数的名字

3.下列有关内联函数的叙述中，正确的是（）

内联函数是通过编译器来实现的

4.对 C++编译器区分重载函数无任何意义的信息是（）。

返回值类型

5.适合于实现功能不复杂但又要求有较快执行速度的函数是（

内联函数

6.有如下函数定义：

```
void func(int a, int& b) { a ++; b++; }
```

若执行代码段：

```
int x = 0, y=1; func(x, y);
```

则变量 x 和 y 的值分别是(　　) 。

0 和 2

7.在 C+中，下列关于设置参数默认值的描述中，正确的是（

设置参数默认值时，应该是先设置右边的再设置左边的

8.计算斐波那契数列第项(0 开始计数)的函数定义如下：

```
int fib(int n){
if(n==0)
return 1;
else if(n==1)
return 1;
else
return fib(n-1)+fib(n-2):
```

若执行函数调用 fib(4),则函数 fib 被调用的次数是（）。

9

9.必须用一对大括号括起来的程序段是（）。

函数的函数体

填空：

1.如下程序运行时输出的第一行到第三行分别是（），（）和（）•

```
#include <iostream>
```

```
using namespace std;
int fun(int *a, int *b);
int fun(int a, int b);
int fun(int*a, int &b);
int main(){
      int x=10,y=5;
      fun(&x,&y); cout<<x<<","<<y<<endl;
      fun(x,y); cout<<x<<", "<<y<<endl;
int fun(int*a, int&b);
int main(){
      int x=10,y=5;
      fun(&x,&y); cout<<x<<", "<<y<<endl;
      fun(x,y); cout<<x<<","<<y<<endl;
      fun(&x,y); cout<<x<<","<<y<<endl;
}
int fun(int*a, int *b) { int *temp=a;a=b; b=temp;}
int fun(int a, int b){ int temp=a;a=b;b=temp;}
int fun(int *a, int &b){ int *temp=a;*a=b;b=*temp;}
```

10 5
10 5
5 5

2.以下程序的输出结果是（

```
#include <iostream>
using namespace std;
long fun( int n) {
      long s;
      if(n==1||n==2) s=2;
      else s=n-fun(n-1);
      return s;}
int main(){
      cout << fun(3);
      return 0;
}
```

1

3.有如下程序：

```
#include<iostream>
```

```
void fun(int a, int b, int &c) {
  a=4,b=5, c=6;
}
int main(){
  int x=10,y=20,z=30;
  fun(x,y,z);
  cout<<x<<''<<y<<''<<z<<endl;
  return 0;
}
```

运行时的输出结果是()。

10 20 6

程序填空 1：

```
/*
```

fact 函数的功能是求 n 的阶乘(题目保证结果不会溢出)。

请将如下程序补充完整。

注意：仅允许在指定的下划线处填写代码，不得改动程序中的其他内容。

试题源程序如下：

```
*/
  #include<iostream>
  using namespace std;
  long long fact(int n)
  {
  /**********FILL**********/
      if(n==1) return _____;
  /**********FILL**********/
      else return ______;
  }
  int main(){
      int n;
      cin>>n;
  /**********FILL**********/
      cout<<_______<<endl;
      return 0;
  }
```

程序改错 1：

请改正程序中指定位置的错误，使程序的输出结果如下：

x=10, Previous=9, Next=11

注意：只允许修改注释“ERROR”的下一行，不得改动程序中的其他内容，也不允许增加或删减语句。

源程序清单：

```
*/
  #include<iostream>
  using namespace std;
  /**********ERROR**********/
  void prevnext(int x, int prev, int next)  {
  /**********ERROR**********/
    prev=x--;
    next=++x;
  }
  int main(){
    int x=10,y,z;
    prevnext(x,y,z);
    cout<<"x="<<x<<", Previous="<<y<<", Next="<<z<<endl;
    return 0;
  }
```

第四章

1.对于一个类定义，下列叙述中错误的是（）

如果没有定义构造函数，编译器将生成一个缺省的构造函数和一个拷贝构造函数

2. 在 C++中，类是一种（）。

自定义数据类型

3.()的功能是对对象进行初始化。

构造函数

4.能够实现类对象初始化任务的是（）。

构造函数

5.

```
#include<iostream>
using namespace std;
class MyClass {
  public:
    MyClass(int n){number=n;}
    MyClass(const IyClass &other){number=other.number:}
    MyClass（）{}
private:
  int number;
}:
void fun(IyClass p){
  MyClass temp(p);
  Return;
}
int main（）{
  MyC1assa(1):
  fun(a);
  return 0;
}
```

运行时 MyC1ass 类的复制构造函数被调用的次数是（）。

2

6.C++类成员的缺省访问属性是（

private

7.类的析构函数的作用是（

撤销对象前完成清理工作

8.已知类中一个成员函数说明如下：

```
void Set(A &a):
```

其中，A&a 的含义是（）。

a 是类 A 的对象引用，用来做函数 Set0 的形参

9.在下列关键字中，用以说明类中公有成员的是（）。

Public

填空题：

1.如下程序运行时输出的第一行和第二行分别是（）和（）。

```
#include <iostream>
using namespace std;
class shownumtype
{
    public:
          void show(int);
          void show(float):
};:
void shownumtype:show(int i)
{
        cout<<"Integer"<<endl;
{
void shownumtype::show(float f)
{
        cout<<"Float"<<endl;
}
int main(){
      int a=0;
      float f=1.0f;
      shownumtype snt
      snt.show(a);
      snt.show(⑤:
      return 0;
}
```

2.如下程序运行时的输出结果是（）.

```
#include <iostream>
using namespace std;
class Point
  int x,y:
public:
  Point(int x1=0,int y1=0):x(x1),y(y1){]
```

```
  int get0(return x+y:}
}:
class Circle{
  Point center;
  int radius;
public:
  Circle(int r=6):radius(r){)
  int get() (return center.get0+radius;
}:
int main(){
  Circle c;
  cout<<c.get0<<endl;
  return 0;
}
```

3.如下程序运行时输出的前两行分别是（）和（）.

```
#include <iostream>
using namespace std;
class test
  private:
    int a;
public:
  test（）{cout<<"constructor"<<endl;}
  test(int a){cout<<"constructor "<<a<<endl;}
  ~test（）cout<<"destructor"<<endl;}
}:
int main({
  test A(3);
  return 0;
}
```

程序设计 1：

```
/*
```

请将如下程序补充完整，使程序在运行时输出：

```
Computer ID: 2
Monitor Type: A
```

注意：仅在程序的下划线处填入所编写的若干表达式或语句，不得改动程序中的其他内容（需删除下划线）。

试题源程序如下：

```
*/
#include<iostream>
using namespace std;
class Monitor {
    public:
        Monitor(char t) { type=t; }
        void display() {
            cout<<"Monitor Type: "<<type<<endl;
        }
    private:
        char type;
};
class Computer {
    public:
/**********FILL**********/
        Computer(int i, char c):                { id=i;}
        void Display() {
            cout<<"Computer ID: "<<id<<endl;
/**********FILL**********/
                                ;
        }
    private:
        int id;
        Monitor mon;
};
int main() {
    Computer c1(2,'A');
/**********FILL**********/
                    ;
    return 0;
}
```

程序改错 1：

```
/*
```

请改正程序中指定位置的错误，使程序的输出结果如下：

```
2021
```

1020

注意：只允许修改注释"ERROR"的下一行，不得改动程序中的其他内容，也不允许增加或删减语句。

源程序清单：

```
*/
#include <iostream>
using namespace std;
class Test {
    private:
        int x,y;
/**********ERROR**********/
    private:
/**********ERROR**********/
        Test(int i,int j)
        { x=i,y=j; }
        int getx() { return x; }
        int gety() { return y; }
};
int main() {
    Test mt1;
    cout<<mt1.getx()<<mt1.gety()<<endl;
/**********ERROR**********/
    Test mt2;
    cout<<mt2.getx()<<mt2.gety()<<endl;
    return 0;
}
```

程序设计 1：

```
/*
```

正方形（Square）类成员如下：

（1）公有成员：

Square(float xx, float yy, float len) // 构造函数，初始化所有数据成员，其中(xx,yy)为左下角位置，len 为边长

void resetSquare(float newX, float newY, float newLen) // 重置正方形左下角坐标，以及边长 newLen

float getLen() // 返回正方形的边长

double getArea() // 返回正方形的面积

double getCircumference() // 返回正方形的周长
bool isEqual(Square &s) // 判断与另一个正方形是否大小相等
（2）私有成员：
float x, y, length   // 正方形左下角的横坐标、纵坐标，边长
double area, circumference // 正方形的面积，周长
请根据上述说明，完成 Square 类的定义。

注意：部分源程序给出，仅允许在注释"Begin"和"End"之间填写内容，不得改动 main 函数和其他已有的任何内容。
试题程序：

```
*/
#include<iostream>
#include<fstream>
using namespace std;

/*******Begin*******/



/*******End*********/

int main() {
    float x, y, len;
    cin>>x>>y>>len;
    Square s1(x,y,len),s2(s1);
    cout<<"s1 边长: "<<s1.getLen()<<", 面积: "<<s1.getArea()<<", 周长:
"<<s1.getCircumference()<<endl;
    cout<<"s2 边长: "<<s2.getLen()<<", 面积: "<<s2.getArea()<<", 周长:
"<<s2.getCircumference()<<endl;
    cout<<"是否相等: "<<s2.isEqual(s1)<<endl;
    cin>>x>>y>>len;
    s2.resetSquare(x,y,len);
    cout<<"重置后: \ns1 边长: "<<s1.getLen()<<", 面积: "<<s1.getArea()<<", 周长:
"<<s1.getCircumference()<<endl;
    cout<<"s2 边长: "<<s2.getLen()<<", 面积: "<<s2.getArea()<<", 周长:
"<<s2.getCircumference()<<endl;
```

```
    cout<<"是否相等: "<<s2.isEqual(s1)<<endl;

    ifstream in1("4.1.1.4_2-s1_in.dat");
    ofstream out1("4.1.1.4_2-s1_out.dat");
    while(in1>>x>>y>>len)
    {
        Square s1(x,y,len),s2(s1);
        out1<<"s1 边长: "<<s1.getLen()<<", 面积: "<<s1.getArea()<<", 周长:
"<<s1.getCircumference()<<endl;
        out1<<"s2 边长: "<<s2.getLen()<<", 面积: "<<s2.getArea()<<", 周长:
"<<s2.getCircumference()<<endl;
        out1<<"是否相等: "<<s2.isEqual(s1)<<endl;
        in1>>x>>y>>len;
        s2.resetSquare(x,y,len);
        out1<<"重置后: \ns1 边长: "<<s1.getLen()<<", 面积: "<<s1.getArea()<<", 周
长: "<<s1.getCircumference()<<endl;
        out1<<"s2 边长: "<<s2.getLen()<<", 面积: "<<s2.getArea()<<", 周长:
"<<s2.getCircumference()<<endl;
        out1<<"是否相等: "<<s2.isEqual(s1)<<endl<<endl;
    }
    in1.close();
    out1.close();
    return 0;
}
```

程序设计 2：

```
/*
```

点（Point）类成员如下：

（1）公有成员：

```
Point(float xx, float yy)   // 构造函数，初始化点的 x，y 坐标
void moveTo(float newX, float newY)  // 将点的 x，y 坐标移动到 newX，newY
```

（2）私有成员：

```
float x, y   // 点的横坐标，纵坐标
```

在此基础上，定义正方形（Square）类，其成员如下：

（1）公有成员：

Square(float x=0.0,float y=0.0,float len=1.0) // 构造函数，初始化所有数据成员，其中(x,y)为左下角位置，len 为边长

void resetSquare(float newX, float newY, float newLen) // 重置正方形左下角坐标为(newX, newY)，边长为 newLen

double getLen() // 返回正方形的边长

double getCircumference() // 返回正方形的周长

bool isEqual(Square &s) // 判断与另一个正方形是否大小相等

（2）私有成员：

Point p // 正方形左下角位置

float length // 正方形的边长

double circumference // 正方形的面积，周长

请根据上述说明，完成 Point，Square 两个类的定义。

注意：部分源程序给出，仅允许在注释“Begin”和“End”之间填写内容，不得改动 main 函数和其他已有的任何内容。

试题程序：

```
*/
#include<iostream>
#include<fstream>
using namespace std;

/*******Begin*******/



/*******End*********/

int main() {
    float x,y,len;
    cin>>x>>y>>len;
    Square s1(x,y,len),s2;
    cout<<"s1 边长: "<<s1.getLen()<<", 周长: "<<s1.getCircumference()<<endl;
    cout<<"s2 边长: "<<s2.getLen()<<", 周长: "<<s2.getCircumference()<<endl;
    cout<<"是否相等: "<<s2.isEqual(s1)<<endl;
```

```
    cin>>x>>y>>len;
    s2.resetSquare(x,y,len);
    cout<<"重置后:\ns1 边长: "<<s1.getLen()<<", 周长:
"<<s1.getCircumference()<<endl;
    cout<<"s2 边长: "<<s2.getLen()<<", 周长: "<<s2.getCircumference()<<endl;
    cout<<"是否相等: "<<s2.isEqual(s1)<<endl;
    ifstream in1("4.2.3_2-2_in.dat");
    ofstream out1("4.2.3_2-2_out.dat");
    while(in1>>x>>y>>len)
    {
        Square s1(x,y,len),s2;
        out1<<"s1 边长: "<<s1.getLen()<<", 周长: "<<s1.getCircumference()<<endl;
        out1<<"s2 边长: "<<s2.getLen()<<", 周长: "<<s2.getCircumference()<<endl;
        out1<<"是否相等: "<<s2.isEqual(s1)<<endl;
        in1>>x>>y>>len;
        s2.resetSquare(x,y,len);
        out1<<"重置后:\ns1 边长: "<<s1.getLen()<<", 周长:
"<<s1.getCircumference()<<endl;
        out1<<"s2 边长: "<<s2.getLen()<<", 周长: "<<s2.getCircumference()<<endl;
        out1<<"是否相等: "<<s2.isEqual(s1)<<endl<<endl;
    }
    in1.close();
    out1.close();
    return 0;
}
```

第五章

1.f()函数是类的一个常成员函数，它有一个 int 型参数，并且返回类型为 int。下列对该常成员函数进行声明的选项中，正确的是(　　)。

int f(int) const;

2. 下列关于友元函数的说法中，不正确的是(　　)。

友元函数可以直接访问类的所有成员

3. 下面对友元函数的描述正确的是(　　)。

友元函数破坏了类的封装性和隐藏性

4. class A{}

```
publie:
  A(){data=0;}
  ~A0{}
  int GetData() const { return data;}
  void SetData(int n) {data=n;}
private:
  int data;
};
const A a;
A b;
```

下列函数调用中错误的是(　　)。

a. SetData(10);

5. 有如下类定义：

```
class Point {
private:
static int how_many;
};
      how_many =0;
```

要初始化 Point 类的静态成员 how_many，下划线处应填入的内容是(　　)。

static int Point::

6. 有如下类和对象的定义：

```
class Constants {
  public:
    static double getPI() { return 3.1416; }
};
Constants constants;
```

下列各组语句中，能输出 3.1416 的是(　　)。

cout<<constants. getPI();和 cout<<Constants::getPI();

7．关于局部变量的说法中，不正确的是(　　)。

不同的函数中可以定义同名的局部变量

8．局部变量可以隐藏全局变量，在有同名全局变量和局部变量的情形时，可以用(　)提供对全局变量的访问。

域运算符

9．下列有关类成员的叙述中，正确的是(　　)。

类成员的默认访问权限是私有的

填空题 1：

有如下程序：

```
#include <iostream >
using namespace std;
class Sample{
    friend long fun(Sample s);
  public:
    Sample(long a) {x = a;}
  private:
    long x;
};
long fun(Sample s) {
  if(s. x<2) return 1;
  return s. x*fun(Sample(s. x-1));
}
int main(){
  int sum=0;
  for(int i=0;i<6;i++)
  { sum += fun(Sample(i));}
  cout<<sum;
  return 0;
}
```

执行这个程序的输出结果是(　)。

填空题 2：

如下程序运行时输出的前两行分别是(　　) 和(　　)。

```
#include <iostream>
```

```
using namespace std;
class MyClass {
  public:
    MyClass(int x):val(x) {}
    void Print() const { cout<<"const::val="<<val<<endl;}
    void Print() { cout<<"val="<<val<<endl;}
  private:
  int val;
};
int main() {
  const MyClass obj1(10);
  MyClass obj2(20);
  obj1. Print();
  obj2. Print();
  return 0;
}
```

填空题 3：

如下程序运行时输出的第一行到第三行分别是(　　)，(　　)和 (　　)。

```
#include <iostream>
using namespace std;
int a;
int f1(int a){
    int b=0;
    static int c=2;
    a++;b++; c++;
    return a+b+c;
}
int main() {
    cout<<f1(a++)<<endl;
    cout<<f1(a++)<<endl;
    cout<<f1(a++)<<endl;
    return 0;
}
```

填空题 4：

有如下程序：

```
#include <iostream>
```

```
using namespace std;
class A {
  public:
  static int a;
  void init() { a=1;}
  A(int a=2) { init(); a++;}
};
int A::a = 0;
int main() {
  A obj1,obj2(4);
  cout<<obj1. a<<A::a<<endl;
  return 0;
}
```

运行时输出的结果是（　）。

填空题 5：

如下程序运行时输出的前两行分别是(　　)和(　　)。

```
#include <iostream>
using namespace std;
class MyClass {
  public:
  MyClass(int x=5):val(x)  { }
  void Print() const { cout<<"const::val="<<val<<endl; }
  void Print()  { cout<<"val="<<val<<endl; }
  private:
  int val;
};
int main()  {
  const MyClass obj1(10);
  objl. Print();
  MyClass obj2;
  obj2. Print();
  return 0;
}
```

程序改错 1：

```
/*
请改正程序中指定位置的错误，使程序的输出结果如下：
x=0/ y=0
x=0, y=1
x=0, y=3
x=0; y=6

注意：只允许修改注释“ERROR”的下一行，不得改动程序中的其他内容，也不允许增加或删减语句。
源程序清单：
*/
#include <iostream>
using namespace std;

class Test {
    public:
/**********ERROR**********/
        Test(int i) { y+=i; }
/**********ERROR**********/
        void display();
        void display() { cout<<"x="<<x<<", y="<<y<<endl; }
        static void show();
    private:
        static const int x=0;
/**********ERROR**********/
        int y;
};

int Test::y=0;

void Test::display() const
{ cout<<"x="<<x<<"; y="<<y<<endl; }

/**********ERROR**********/
static void Test::show()
{ cout<<"x="<<x<<"/ y="<<y<<endl; }
```

```
int main() {
    Test::show();
    Test t1; t1.display();
    Test t2(2); t2.display();
    const Test t3(3);
    t3.display();
    return 0;
}
```

程序填空 1:

```
/*
```

请将如下程序补充完整，使得程序运行时的输出结果为：

```
0
50
150
```

注意：仅允许在指定的下划线处填写代码，不得改动程序中的其他内容（需删除下划线）。

试题源程序如下：

```
*/
#include <iostream>
using namespace std;
class goods {
    public:
        goods(int w) { weight=w; totalweight+=weight; }
        goods(goods &gd) { weight=gd.weight; totalweight+=weight; }
        ~goods() { totalweight-=weight; }
/**********FILL**********/
        ___【1】___ { return totalweight; }
    private:
        static int totalweight;
        int weight;
};
/**********FILL**********/
___【2】___;
int main() {
/**********FILL**********/
```

```
    cout<<___【3】___<<endl;
    goods g1(50);
    cout<<g1.gettotal()<<endl;
    goods g2(100);
    cout<<g2.gettotal()<<endl;
    return 0;
}
```

程序填空 2：

```
/*
```

请将如下程序补充完整，使得程序运行时的输出结果为：

```
a=10, b=2
a=20; b=4
```

注意：仅允许在指定的下划线处填写代码，不得改动程序中的其他内容（需删除下划线部分）。

试题源程序：

```
*/
#include <iostream>
using namespace std;

class A {
    public:
/**********FILL**********/
        A(int i):___【1】___{ b+=2; }
        void print() const;
        void print() { cout<<"a="<<a<<", b="<<b<<endl; }
    private:
        const int a;
/**********FILL**********/
        ___【2】___;
};

int A::b=0;

/**********FILL**********/
___【3】___ { cout<<"a="<<a<<"; b="<<b<<endl; }
```

```
int main() {
    A a1(10); a1.print();
    const A a2(20);
/**********FILL**********/
    ___【4】___;
    return 0;
}
```

第六章

1.定义函数指针的是（）。

int (*p)（）；

2. 类 MyClass 的定义如下：

```
class IIyClass
{
    public:
        MyClass(){value=0;}
        SetValue(int i){value=i;}
    private:
        int value;
};
```

则对语句：MyClass *p, my; p=&my;正确的描述是(　　)。

语句 p->SetValue(5)与 my. SetValue(5)等价

3.有如下头文件：

```
int f1（）；
static int f2（）：
class MA {
public:
int f3（）；
static int f4（）；
}:
```

在所描述的函数中，具有隐含的 this 指针的是（）。

f3

4.若有声明语句“int a:int* const p=&a:”，则（）。

P 是一个指针型的常量，指针 p 本身的值不能被改变

5.下列关于对象数组的描述中，()是错误的。

对象数组只能赋初值，而不能在定义后赋值

6.已知函数 f1oat fun(float)是类 A 的成员函数，fp 是指向该函数类型的指针，但 fp 不是类 A 的成员，则下列操作正确的是（

fp=A::fun;

7.下面关于数组的初始化正确的是（）。

char str[]={'a','b','c'}

8.若有定义：int *p=new int:,则下列说法中不正确的是（）。

系统为指针变量 p 分配了一个整型数据的存储空间

9.有如下程序：

```
#include <iostream>
```

```
using namespace std;
int main() {
  char s[6];
  for(int i=0; i<5; ++i) s[i]=’ a’ +i;
  s[i] =’ \0’ ;
  cout<<s<<end1;
  return 0;
}
```

编译运行的情况是(　　)。

编译出错

填空题：

1.有如下程序：

```
#include <iostream>
using namespace std;
class CD {
 public:
    CD() { cout<<' B'; }
    ~CD() { cout<<'C'; }
 private:
  char name[10];
};
int main() {
  CD a, *b, d[2];
  return 0;
}
```

运行时的输出结果是(　　)。

BBBCCC

2.如下程序运行时的输出结果是（）。

```
#include<iostream>
using namespace std;

class XCF{
  int a;
public:
  XCF(int aa=0):a(aa){cout<<"1";}
  XCF(XCF & x){a=x. a; cout<<"2";}
```

```
  ~XCF(){cout<<a;}
  int Geta(){return a;}
};
int main(){
  XCF d1(5), d2(d1);
  XCF *pd=new XCF(8);
  cout<<pd->Geta();
  delete pd;
  return 0;
}
```

1218855

3.如下程序运行时输出的第一行和第二行分别是（）和（）。

```
#include <string>
#include <iostream>
using namespace std;
int main(){
      strings1 ="面向对象", s2="程序设计";
      cout<<s1<<s2<<endl;
      s1+=s2;
      cout<<s1<<endl;
      return 0;
}
```

面向对象程序设计

面向对象程序设计

程序填空 1：

```
/*
完善程序，使该程序运行时的输出结果是"066"。
源程序如下：
*/
#include <iostream>
using namespace std;
class myClass{
/**********FILL**********/
  ______【1】____a;
public:
  myClass(){a++;}
  ~myClass(){ a--;}
/**********FILL**********/
  ______【2】____{return a;}
};
int myClass::a=0;
int main(){
  cout<<myClass::getA();
  myClass a,b[4]; myClass*c=new myClass;
  cout<<a. getA()<<c->getA()<<endl;
  return 0;
}
```

解：

```
static int
static int getA( )
```

程序设计 1：

```
/*
课程(Course) 类主要用于处理某门课程的成绩，主要成员如下：
(1) 私有成员：
  int number  //课程编号
  string name //课程名称
  int credit  //课程学分
  int n;     //选修课程的学生数量
  float *score; //指向保存学生成绩的数组
  int max, min //课程的最高分、最低分
```

float average //课程的平均分

int count //不及格(<60)的学生数量

(2) 公有成员:

Course(int n); //构造函数,动态创建一个长度为 n 的数组,返回值赋给 score，并将 credit，count，average 初始化为 0，n 为学生数量

~Course(); //析构函数，删除成绩数组

void input( ) //依次输入课程编号、课程名称、学分以及 n 名学生的成绩

void process( ) //计算课程的最高分、最低分、平均分，以及不及格的学生数量

void print( ) //输出课程编号、课程名称、最高分、最低分、平均分以及不及格学生数

请根据上述说明，完成 Course 类的定义。

注意：部分源程序给出，仅允许在注释"Begin"和"End"之间填写内容，不得改动 main 函数和其他已有的任何内容。

```
#include<iostream>
#include<fstream>
#include<string>
using namespace std;

class Course {
      int number;  //课程编号
      string name; //课程名称
      int credit ; //课程学分
      int n;    //选修课程的学生数量
      float*score; //指向保存学生成绩的数组
      int max, min: //课程的最高分、最低分
      float average://课程的平均分
      int count;  //不及格(<60) 的学生数量
public:
      Course(int n); //构造函数,动态创建一个长度为 n 的数组,返回值赋给 score，并将 credit，count，average 初始化为 0，n 为学生数量
      ~Course():   //析构函数,删除成绩数组
      void input(): //依次输入课程编号、课程名称、课程学分以及 n 名学生的成绩
      void process(): //计算课程的最高分、最低分、平均分，以及不及格的学生数量
      void print(){ //输出课程编号、名称、学分、最高分、最低分、平均分，以及不及格学生数
          cout<<"课程编号:"<<number<<endl;
```

```
        cout<<"课程名称:"<<name<<"课程学分:"<<credit<<endl;
        cout<<"最高分:"<<max<<"最低分:"<<min<<"平均分:"<<average<<"不及格
        数:"<<count<<endl<<endl:
        }
};
/*******Begin*******/



/*******End*********/
int main() {
   int n;
   cin>>n;
   Course c(n);
   c. input();c. process();c. print();

   ifstream in1("6.6.2_2_in. dat");
   ofstream out1("6.6.2_2_out. dat");
   streambuf *cinbackup;
   streambuf*coutbackup;
   cinbackup=cin. rdbuf(in1. rdbuf());

int main(){
   int n;
   cin>>n;
   Course c(n);
   c. input();c. process();c. print();

   ifstream in1("6.6.2_2_in. dat");
   ofstream out1("6.6.2_2_out. dat");
   streambuf*cinbackup;
   streambuf*coutbackup;
   cinbackup=cin. rdbuf(in1. rdbuf());
   coutbackup=cout. rdbuf(out1. rdbuf());
   while(cin>>n){
      Course c(n);
      c. input();c. process();c. print();
```

```
    }
    cin. rdbuf(cinbackup);
    cout. rdbuf(coutbackup);
    in1. close();
    out1. close();
    return 0;
}
```

试题程序 1：

```
Course::Course(int n) {
    this->n=n;
    score=new float[n];
    credit=0; count=0; average=0;
}
Course::~Course(){ delete[] score; }
void Course::input() {
    cin>>number>>name>>credit;
    for(int i=0; i<n; i++) cin>>score[i];
}
void Course::process() {
    max=score[0]; min=score[0];
    for(int i=0; i<n; i++) {
        if(max<score[i]) max=score[i];
        if(min>score[i]) min=score[i];
        if(score[i]<60) count++;
        average+=score[i];
    }
    average/=n;
}
```

程序设计 2.

/*

成绩(Score) 类用于保存某门课程的考试成绩，并统计该课程的最高分、最低分、平均分和不及格的学生人数。

Score 类的成员声明已给出，请参照注释，完成 Score 类的定义。

注意：部分源程序给出，仅允许在注释"Begin"和"End"之间填写内容，不得改动 main 函数和其他已有的任何内容。

试题程序：

```
*/
#include<iostream>
#include<fstream>
#include<string>
using namespace std;
class Score {
   public:
      Score(int n); //构造函数，动态创建一个长度为 n 的数组，返回值赋给 score，
      n 为学生数量
      Score(con st Score &s);//实现对象的深复制
      ~Score();    //析构函数,删除成绩数组
      void input0: //依次输入课程编号、课程名称，以及 n 个学生的成绩
      void process(); //计算课程的最高分、最低分、平均分，以及不及格人数
      void print() { //输出课程编号、课程名称、最高分、最低分、平均分以及不及格
      人数
         cout<<"课程编号:"<<cNum<<"课程名称:"<<cName<<endl;
         cout<<"最高分:"<<max<<"最低分:"<<min<<"平均分:"<<average<<" 不及格
         数:"<<num_fail<<endl<<endl;
      }
   private:
      string cNum,cName;//课程编号、课程名称
      int n;        //选修课程的学生数量
      float *score;   //指向保存学生成绩的数组
      int max, min, num_fail; //课程的最高分、最低分、不及格人数
      float average;   //课程的平均分
};
/*******Begin*******/


/*******End*********/
int main(){
   int n;
   cin>>n;
   Scores1(n);
       s1. input();   s1. process(); s1. print();
   Score s2(s1); s2. print();
```

```
    ifstream in1("6.6.3_3_in. dat");
    ofstream out1("6.6.3_3_out. dat");
    streambuf*cinbackup;
    streambuf*coutbackup;

int main(){
    int n;
    cin>>n;
    Score s1(n);
        s1. input();   s1. process(); s1. print();
    Scores2(s1); s2. print();
    ifstream in1("6.6.3_3_in. dat");
    ofstream out1("6.6.3_3_out. dat");
    streambuf *cinbackup;
    streambuf*coutbackup;
    cinbackup=cin. rdbuf(in1. rdbuf());
    coutbackup=cout. rdbuf(out1. rdbuf());
    while(cin>>n){
       Scores1(n);
              s1. input();   s1. process(); s1. print();
       Scores2(s1); s2. print();
    }
    cin. rdbuf(cinbackup);
    cout. rdbuf(coutbackup);

    in1. close();
    out1. close();
    return 0;
}
```

解：

```
Score::Score(int n) {
        this->n=n;
        score=new float[n];
    }
    Score::Score(const Score &s){
        cNum=s.cNum; cName=s.cName;
```

```
        n=s.n; max=s.max; min=s.min; num_fail=s.num_fail; average=s.averag
e;
        score=new float[n];
        for(int i=0;i<n;i++)
            score[i]=s.score[i];
    }

    Score::~Score(){ delete[] score; }
    void Score::input() {
        cin>>cNum>>cName;
        for(int i=0; i<n; i++) cin>>score[i];
    }
    void Score::process() {
        num_fail=0,average=0;
        max=score[0]; min=score[0];
        for(int i=0; i<n; i++) {
            if(max<score[i]) max=score[i];
            if(min>score[i]) min=score[i];
            if(score[i]<60) num_fail++;
            average+=score[i];
        }
        average/=n;
    }
```

第七章

1.设置虚基类的目的是（

消除二义性

2.有如下类定义：

```
class Music
  public:
    void setTitle(char*str){strcpy(title,str):
  protected:
    char type[10];
  private:
    char title[20]:
};
class Jazz:public Iusic
  public:
    void set(char*str){
      strcpy(type,"Jazz"):/①
      strcpy(title,str);//2
  }
};
```

下列叙述中正确的是（）。

程序编译时语句②出错

3.继承具有（），即当基类本身也是某一个类的派生类时，底层的派生类也会自动维承间接基类的成员。

传递性

4．有如下类定义：

```
class AA
  int a;
Public:
    AA(int n=0): a(n){}
}:
class BB:public AA
  public:
    BB(int n)_______
};
```

其中横线处的缺失部分是（）。

:AA(n){}

5.有如下类定义：

```
class XX{
```

```
    int xdata;
  public:
    XX(int n=0):xdata (n){}
class YY : public XX
    int ydata:
  public:
    YY(int m=0,int n=0):XX(m),ydata(n){}
}
```

则 YY 类的对象包含的数据成员的个数是（）。

2

6.C++中的类有两种用法：一种是类的实例化，即生成类的对象，并参与系统的运行：另一种是通过（),派生出新的类。

继承

7.下列关于派生类构造函数和析构函数的说法中，错误的是（）

在销毁派生类对象时，先调用基类的析构函数，再调用派生类的析构函数

8 在公有派生情祝下，有关派生类对象和基类对象的关系，下列叙述不正确的是（）。

派生类的对象可以直接访问基类中的成员

9.下列叙述中错误的是（）

基类成员的访问能力在派生类中保持不变

10. 基类的（）在派生类中的性质与继承的性质一样

公有成员

第八章

1．下列有关维承和派生的叙述中，正确的是（

如果派生类没有实现基类的一个纯虚函数，则该派生类是一个抽象类

2.下列运算符中，()运算符在 C+中不能重载。

::

3.关于析构函数的叙述中，不正确的是（

基类的析构函数可以被派生类继承

4.朕于动态联编的下列描述中，()是错误的。

动态联编是在编译时确定操作函数的

5.以下基类中的成员函数表示纯虚函数的是()。

virtual void tt（）=0

6.下列关于运算符函数的描述中，错误的是（）•

运算符函数只能定义为类的成员函数

7.静态成员函数不能说明为（

虚函数

8.有关多态性说法不正确的是（）

运行时的多态性可通过模板和虚函数实现

填空题：

1.如下程序运行时输出的第一行到第三行分别是（),()和（

```
#include<iostream>
using namespace std;
class Base
    public:
        virtual Base（）[cout<<"Base Destructor"<<endl:
        virtual void fun（）{cout<<"Base:fun"<<endl;
}:
class Derived:public Base
    public:
        Derived（）{cout<<"Derived Destructor"<<endl:
        void fun（）[cout<<"Derived:fun"<<endl:
};
int main（）{
    Base +p=new Derived:p->fun（）;
    delete p:
    return 0;
)
```

Derived::fun

Derived Destructor

Base Destructor

2.如下程序运行时的输出结果是（）.

```
#include〈iostream>
using namespace std;
class A
  public:
    virtual void func1（）cout <"A1";
    void func20 cout <"A2";
}:
class B:public A
  public:
    void func1（）{cout <"B1";
    void func2（）{cout <"B2";
};
int main（）{
  A *p=new B:
  p->func10);
  p->func2（）;
  delete p:
  return 0;
}
```

B1A2

3.如下程序运行时的输出结果是（

```
#include〈iostream>
using namespace std;
class MyClass
  public:
    MyClass(int i=0){cout<<1;
    MyClass(const IyClass &x){cout<<2:
    MyClass&operator =(const IyClass &x){
      cout<<3;
      return *this;
  }
  MyClass（）{ cout<<4: }
};
int main（）{
```

```
  MyClass objl(1),obj2(2),obj3(obj1);
  return 0;
}
```

112444

程序改错：

```
/*
```

请改正程序中指定位置的错误，使程序的输出结果如下：

```
f1 function of base
f2 function of base
f1 function of derive
f2 function of base
```

注意：只允许修改注释“ERROR”的下一行，不得改动程序中的其他内容，也不允许增加或删减语句。

源程序清单：

```
*/
#include <iostream>
using namespace std;

class base {
/**********ERROR**********/
    protected:
/**********ERROR**********/
        void f1()
        { cout<<"f1 function of base"<< endl; }
        void f2() { cout<<"f2 function of base "<< endl; }
};

/**********ERROR**********/
class derive::base
{   public:
        void f1() { cout<<"f1 function of derive"<<endl; }
        void f2() { cout<<"f2 function of derive "<<endl; }
};

/**********ERROR**********/
void fun(base p)
```

```
{ p->f1(); p->f2(); }

int main() {
    base obj1; fun(&obj1);
    derive obj2; fun(&obj2);
    return 0;
}
public:
virtual void f1()
class derive:public base
void fun(base *p)
```

程序填空：

```
/*
```

下面程序在定义分数（fract）类的基础上，重载复合赋值运算符"+="。请将程序补充完整，使程序运行时的输出结果为 41/28。

注意：仅允许在指定的下划线处填写代码，不得改动程序中的其他内容（需删除下划线部分）。

试题源程序：

```
*/
#include <iostream>
using namespace std;

class fract {
        int den; //分子
        int num; //分母
    public:
        fract(int d=0,int n=1):den(d),num(n) {}
        fract &operator +=(const fract&);
/**********FILL**********/
        ___【1】___{ cout<<den<<'/'<<num<<endl; }
};

/**********FILL**********/
___【2】___ {
    this->den=this->den*f.num+this->num*f.den;
    this->num*=f.num;
/**********FILL**********/
```

```
    return ___【3】___;
}

int main() {
    fract fr(3,4),fr2(5,7);
/**********FILL**********/
    ___【4】___fr2;
    fr.show();
    return 0;
}
void show(){ cout<<den<<'/'<<num<<endl; }
fract  &fract::operator +=(const fract& f){
return *this;
fr += fr2;
```

程序设计：

```
/*
```

复数（Complex）类定义如下：

```
class Complex {
    public:
        Complex(int r = 0, int i = 0): real(r), imag(i) { }  // 构造函数
    private:
        int real;  // 复数实部
        int imag;  // 复数虚部
};
```

基于上述定义，请完成以下运算符重载：

（1）以成员函数形式重载“+”运算符，实现两个复数相加。

（2）将运算符“==”重载为非成员函数形式，实现判断两个复数是否相等（若相等则返回 true，否则返回 false）。

（3）重载“<<”运算符，用于复数对象的输出（格式为“(real, imag)”）。

注意：部分源程序已给出，仅允许在注释“Begin”和“End”之间补全代码，不得改动其他已有的任何内容。

测试样例：

输入：

```
1 2 3 4
```

输出：

```
c1 的值为：(1, 2)
c2 的值为：(3, 4)
c1!=c2
c3=c1+c2 的值为：(4, 6)
```

试题程序：

```
*/
#include <iostream>
#include<fstream>
using namespace std;

/*******Begin*******/



/*******End*********/

ostream & operator <<(ostream &out, const Complex &c) {  //运算符<<重载函数实现
    out << "(" << c.real << ", " << c.imag << ")";
    return out;
}

int main() {
    int r1,i1,r2,i2;
    cin>>r1>>i1>>r2>>i2;
    Complex c1(r1,i1),c2(r2,i2),c3;
    cout<<"c1 的值为：<<c1<<endl;
    cout<<"c2 的值为：<<c2<<endl;
    if(c1==c2) cout<<"c1==c2"<<endl;
    else cout<<"c1!=c2"<<endl;
    c3=c1+c2;
    cout<<"c3=c1+c2 的值为：<<c3<<endl;

    ifstream in1("8.2.1.1-s1_in.dat");
    ofstream out1("8.2.1.1-s1_out.dat");
    streambuf *cinbackup;
    streambuf *coutbackup;
    cinbackup=cin.rdbuf(in1.rdbuf());
    coutbackup=cout.rdbuf(out1.rdbuf());
    while(cin>>r1>>i1>>r2>>i2) {
        Complex c1(r1,i1),c2(r2,i2),c3;
        cout<<"c1 的值为：<<c1<<endl;
        cout<<"c2 的值为：<<c2<<endl;
        if(operator ==(c1,c2)) cout<<"c1==c2"<<endl;
        else cout<<"c1!=c2"<<endl;
        c3=c1.operator +(c2);
```

```
        cout<<"c3=c1+c2 的值为："<<c3<<endl<<endl;
    }
    cin.rdbuf(cinbackup);
    cout.rdbuf(coutbackup);
    in1.close();
    out1.close();
    return 0;
}
class Complex {
    public:
        Complex(int r=0,int i=0): real(r),imag(i) { } //构造函数
        Complex operator +(const Complex &) const; //运算符+重载
        friend bool operator ==(const Complex &,const Complex &); //判断两个
复数是否相等
        friend ostream & operator <<(ostream &,const Complex &); //运算符<<
重载
    private:
        int real; //复数实部
        int imag; //复数虚部
};

Complex Complex::operator +(const Complex &c) const { //运算符+重载函数实现
    return Complex(this->real+c.real, this->imag+c.imag);
}

bool operator ==(const Complex &c1,const Complex &c2) { //运算符==重载函数实
现
    if(c1.real==c2.real && c1.imag==c2.imag) return true;
    else return false;
}
```

程序设计 2：

```
/*
```

日期（Date）类成员声明如下：

```
class Date {
    public:
        Date(int yy=2023,int mm=1,int dd=1); //构造函数，yy，mm，dd 分别用于初
始化日期的年、月、日
        void setDate(int newY,int newM,int newD); // 设置日期的年、月、日为 newY,
newM, newD
    protected:
        int year, month, day;  // 日期的年、月、日
};
```

请参照注释，完成以下任务：

（1）完成 Date 类的定义。
（2）以成员函数形式重载“==”运算符，用于判断两个日期对象是否相等（对应数据值相等），若相等则返回 true，否则返回 false。
（3）重载运算符“<<”，用于输出日期对象，输出格式为“年-月-日”。

注意：部分源程序已给出，仅允许在注释“Begin”和“End”之间补全代码，不得改动其他已有的任何内容。

测试样例：
输入：
2023 4 10
2023 4 10
输出：
d1 的值为：2023-4-10
d2 的值为：2023-1-1
d1 和 d2 不是同一天
重置后 d2 的值为：2023-4-10
d1 和 d2 是同一天

试题程序：

```
*/
#include <iostream>
#include<fstream>
using namespace std;

/*******Begin*******/



/*******End*********/

ostream & operator <<(ostream &out, const Date &d) { // 运算符<<重载
    out << d.year << '-' << d.month << '-'<< d.day;
    return out;
}

int main()
{
    int y,m,d;
    cin>>y>>m>>d;
    Date d1(y,m,d),d2;
    cout<<"d1 的值为："<<d1<<endl;
    cout<<"d2 的值为："<<d2<<endl;
```

```
    if(d1==d2) cout<<"d1 和 d2 是同一天"<<endl;
    else cout<<"d1 和 d2 不是同一天"<<endl;
    cin>>y>>m>>d;
    d2.setDate(y,m,d);
    cout<<"重置后 d2 的值为："<<d2<<endl;
    if(d1.operator ==(d2)) cout<<"d1 和 d2 是同一天"<<endl;
    else cout<<"d1 和 d2 不是同一天"<<endl;

    ifstream in1("8.2.2.5_2-s1_in.dat");
    ofstream out1("8.2.2.5_2-s1_out.dat");
    streambuf *cinbackup;
    streambuf *coutbackup;
    cinbackup=cin.rdbuf(in1.rdbuf());
    coutbackup=cout.rdbuf(out1.rdbuf());
    while(cin>>y>>m>>d) {
        Date d1(y,m,d),d2;
        cout<<"d1 的值为："<<d1<<endl;
        cout<<"d2 的值为："<<d2<<endl;
        if(d1==d2) cout<<"d1 和 d2 是同一天"<<endl;
        else cout<<"d1 和 d2 不是同一天"<<endl;
        cin>>y>>m>>d;
        d2.setDate(y,m,d);
        cout<<"重置后 d2 的值为："<<d2<<endl;
        if(d1.operator ==(d2)) cout<<"d1 和 d2 是同一天"<<endl;
        else cout<<"d1 和 d2 不是同一天"<<endl;
        cout<<endl;
    }
    cin.rdbuf(cinbackup);
    cout.rdbuf(coutbackup);
    in1.close();
    out1.close();
    return 0;
}
class Date {
    public:
        Date(int yy=2023,int mm=1,int dd=1); //构造函数，yy，mm，dd 分别用于初始化日期的年、月、日
        void setDate(int newY,int newM,int newD); // 设置日期的年、月、日为 newY，newM，newD
        bool operator ==(const Date &) const; // 判断两个日期是否相等
        friend ostream & operator <<(ostream &, const Date &); //运算符<<重载
    protected:
        int year, month, day;  // 日期的年、月、日
```

```
};

Date::Date(int yy,int mm,int dd) { //构造函数实现
    this->year=yy; this->month=mm; this->day=dd;
}
void Date::setDate(int newY,int newM,int newD){ // 设置日期的年、月、日为 newY, newM, newD
    this->year=newY; this->month=newM; this->day=newD;
}
bool Date::operator == (const Date &d) const { //  运算符==重载
    if(this->year==d.year && this->month==d.month && this->day==d.day) return true;
    else return false;
}
```

第九章

1.某类中有一个无参且无返回值的常成员函数 Show,则正确的 Show 函数原型是（）。

void Show（）const;

2.模板对类型的参数化提供了很好的支持，因此（）。

类模板实例化时，编译器将根据给出的模板实参生成一个类

3.有如下类定义：

```
class AA
    int a;
  Public:
    AA(int n=0):a(n)O
};
class BB:public AA
  public:
    BB(int n)_________
```

其中横线外的缺失部分是（

:AA(n){}

4.下列关于派生类和基类的描述中，正确的是（

派生类成员函数只能访问基类的公有和保护成员

5.在 C++程序中，对象之间的相互通信是通过（)实现的。

调用成员函数

6.若有变量定义：itx=5:,则将 rx 定义为变量 x 的引用的是（

int &rx=x;

7.友元的作用是（）

提高程序的运行效率

8.有如下语句序列：

```
char str[10];cin>>str:
```

当从键盘输入"I love this game"时，str 中的字符串是（）。

"I"

9.下列语句中，错误的是（

const int temp;

10.对类成员访问权限的控制，是通过设置成员的访问控制属性实现的，下列不是访问控制属性的是（）。

友元类型

11.关于动态内存分配，对 delete 运算符的下列说法中，错误的是（

对一个指针变量可任意多次使用该运算符

12.执行下列语句段后，输出字符“*“的个数是（

```
for(int i=50;i>1;i-=2)cout <<'*'
```

25

13.下列关于虚函数的表述中，正确的是（

虚函数都是成员函数

14.下列关于函数的描述中，错误的是（

函数不能被定义为模板

15.下列关于多态性的描述，错误的是（

运行时的多态性可通过模板和虚函数实现

16.有如下类定义：

```
class Point{
private:
static int how_many;
}:
_____how_many =0:
```

要初始化 Pointi 类的静态成员 how_many,下划线处应填入的内容是（）。

int Point::

17.下列关于流类库的描述中，错误的是（）。

流类库中总共定义了 3 个类

18.内了提高函数调用的实际运行速度，可以将较简单的函数定义为（

内联函数

19.一个类可以被描述为（

其他选项都正确

20.不是构造函数的特征。

构造函数必须指定类型说明

填空题：

1.有如下程序：

```
#include〈iostream>
using namespace std;
void fun(int *a,int b){
    int *k;
    k=a;
    *a=b:
    b=*k:
]
int main（）{
  int a=3,*x=a;
  int y=5;
  fun(x,y);
  cout<<a<''<<y<<endl:
  return 0;
}
```

运行时的输出结果是（）。

5 5

2.如下程序运行时输出的第一行和第二行分别是（)和（）

```
#include〈iostream>
using namespace std;
void fun(int&x,int&y){int z=x:x=y:y=z:}
void fun(int *x,int *y){int *z=x:x=y;y=z;]
int main（）{
    int x=5,y=10;
    fun(x,y):
    cout <<"x="<<x<<",y="<<y<<endl;
    fun(&x,&y):
    cout <<"x="<<x<<",y="<<y<<endl;
    return 0;
}
```

x=10,y=5

x=10,y=5

3.如下程序运行时输出的第一行到第三行分别是（),()和（

```
#include〈iostream>
using namespace std;
class A {
    public:
        int a:
        A(int x=0){a=x;cout<<"A:a="<<a<<end1;}
}:
:class B:public A
{
    public:
        int a.
    B(int x,int y=5):A(x){a=y:cout<<"B:a="<<a<<endl:}
};
Int main（）{
  B b(1,2):
  A *p=&b;
  cout<<"a="<<p->a<<endl;
  return 0;
}
```

A::a=1

B::a=2

a=1

4.如下程序运行时输出的第一行到第三行分别是（

```
#include〈iostream>
using namespace std;
int a=4;
int main（）{
    int b=7,c=10;
    cout<<a<<","<<b<<","<<c<<end1;
    {
        int b=6;
        float c=8.8;
        cout<<a<<","<<b<<","<<c<<end1:
        a=b;
    }
    cout<<a<<","<<b<<","<<c<<end1;
    return 0;
}
```

4,7,10

4,6,8.8

6,7,10

5.如下程序运行时的输出结果是（）.

```
#include〈iostream>
using namespace std;
class Part
  public:
    Part(int x=0):val(x)cout<<val;
    Part（）cout<<val;
private:
  int val;
};
Class Whole
  public:
    Whole(int x,int y,int z=0):p2(x),p1(y),val(z)cout<<val:
    Whole（）cout<<val;
private:
  Part p1,p2;
```

```
  Int val;
};
int main（）{
  Who1e obj(1,2,3):
  return 0;
}
```

213312

程序改错 1：

```
/*
```

请改正程序中指定位置的错误，使程序的输出结果如下：

```
x=0, y=1
x=1, y=3
x=3; y=7
```

注意：只允许修改注释"ERROR"的下一行，不得改动程序中的其他内容，也不允许增加或删减语句。

源程序清单：

```
*/
#include <iostream>
using namespace std;

/**********ERROR**********/
class
{

/**********ERROR**********/
        const int x;
        static int y;
    public:
        Test() { y+=1; }

/**********ERROR**********/
        Test(int i,int j):x={i}
        { y+=j; }
        void display() const;
        void display() { cout<<"x="<<x<<", y="<<y<<endl; }
};

int Test::y=0;

/**********ERROR**********/
void Test::display()
```

```
{ cout<<"x="<<x<<"; y="<<y<<endl; }

int main() {
    Test t1; t1.display();
    Test t2(1,2); t2.display();
    const Test t3(3,4);
    t3.display();
    return 0;
}
class Test
   x=0
   Test(int i,int j):x(i)
   void Test::display() const
```

程序改错 2：

```
/*
```

请改正程序中指定位置的错误，使程序的输出结果如下：

```
Base::display
Derive::display
Derive::display const
Base::display
Derive::display const
Derive::display const
```

注意：只允许修改注释"ERROR"的下一行，不得改动程序中的其他内容，也不允许增加或删减语句。

试题源程序：

```
*/
#include <iostream>
using namespace std;

class Base {
    public:
/**********ERROR**********/
        void display() {
            cout<<"Base::display"<< endl;
        }
};

class Derive:public Base {
    public:
/**********ERROR**********/
        void display();
        void display() { cout<<"Derive::display"<<endl; }
```

```
};

void Derive::display() const {
    cout<<"Derive::display const"<<endl;
}

/**********ERROR**********/
void Display(Base p) {
    p.display();
}

int main() {
    Base b1; Derive d1;
/**********ERROR**********/
    Derive d2;
    b1.display(); d1.display(); d2.display();
    Display(b1); Display(d1); Display(d2);
    return 0;
}
virtual void display() const
   void display() const;
   void Display(const Base &p){
   const Derive d2;
```

程序设计 1：

```
/*
课程（Course）类主要用于处理某门课程的成绩，主要成员如下：
（1）私有成员：
int number    // 课程编号
string name   // 课程名称
int credit    // 课程学分
int n;        // 选修课程的学生数量
float *score;  // 指向保存学生成绩的数组
int max,min   // 课程的最高分、最低分
float average  // 课程的平均分
int count     // 不及格（<60）的学生数量
（2）公有成员：
Course(int n);   // 构造函数，动态创建一个长度为 n 的数组，返回值赋给 score，并将 credit，count，average 初始化为 0，n 为学生数量
~Course();     // 析构函数，删除成绩数组
void input()    // 依次输入课程编号、课程名称、学分以及 n 名学生的成绩
void process()   // 计算课程的最高分、最低分、平均分，以及不及格的学生数量
void print()    // 输出课程编号、课程名称、最高分、最低分、平均分以及不及格学生数
```

请根据上述说明，完成 Course 类的定义。

注意：部分源程序给出，仅允许在注释"Begin"和"End"之间填写内容，不得改动 main 函数和其他已有的任何内容。

试题程序：

```
*/
#include<iostream>
#include<fstream>
#include<string>
using namespace std;

class Course {
        int number;    // 课程编号
        string name;   // 课程名称
        int credit;    // 课程学分
        int n;         // 选修课程的学生数量
        float *score;  // 指向保存学生成绩的数组
        int max,min;   // 课程的最高分、最低分
        float average; // 课程的平均分
        int count;     // 不及格（<60）的学生数量
    public:
        Course(int n);   // 构造函数，动态创建一个长度为 n 的数组，返回值赋给 score，并将 credit，count，average 初始化为 0，n 为学生数量
        ~Course();       // 析构函数，删除成绩数组
        void input();    // 依次输入课程编号、课程名称、课程学分以及 n 名学生的成绩
        void process();  // 计算课程的最高分、最低分、平均分，以及不及格的学生数量
        void print() {   // 输出课程编号、名称、学分、最高分、最低分、平均分，以及不及格学生数
            cout<<"课程编号: "<<number<<endl;
            cout<<"课程名称: "<<name<<" 课程学分: "<<credit<<endl;
            cout<<"最高分: "<<max<<" 最低分: "<<min<<" 平均分: "<<average<<" 不及格数: "<<count<<endl<<endl;
        }
};

/*******Begin*******/
```

```
/*******End*********/

int main() {
    int n;
    cin>>n;
    Course c(n);
    c.input();c.process();c.print();

    ifstream in1("6.6.2_2_in.dat");
    ofstream out1("6.6.2_2_out.dat");
    streambuf *cinbackup;
    streambuf *coutbackup;
    cinbackup=cin.rdbuf(in1.rdbuf());
    coutbackup=cout.rdbuf(out1.rdbuf());
    while(cin>>n) {
        Course c(n);
        c.input();c.process();c.print();
    }
    cin.rdbuf(cinbackup);
    cout.rdbuf(coutbackup);
    in1.close();
    out1.close();
    return 0;
}
Course::Course(int n) {
    this->n=n;
    score=new float[n];
    credit=0; count=0; average=0;
}
Course::~Course(){ delete[] score; }
void Course::input() {
    cin>>number>>name>>credit;
    for(int i=0; i<n; i++) cin>>score[i];
}
void Course::process() {
    max=score[0]; min=score[0];
    for(int i=0; i<n; i++) {
        if(max<score[i]) max=score[i];
        if(min>score[i]) min=score[i];
        if(score[i]<60) count++;
        average+=score[i];
    }
```

```
    average/=n;
}
```

程序设计 2：

```
/*
```

点（Point）类成员如下：

（1）公有成员：

```
Point(float xx, float yy)  // 构造函数，初始化点的 x，y 坐标
float getX() const  // 返回横坐标 x
float getY() const  // 返回纵坐标 y
void setX(float newX)  // 重设横坐标为 newX
void setY(float newY)  // 重设纵坐标为 newY
```

（2）私有成员：

```
float x, y  // 点的横坐标，纵坐标
```

由 Point 类公有派生出圆（Circle）类，基类 Point 的 x，y 成员作为圆心的坐标，并新增如下成员：

（1）公有成员：

```
Circle(float x=0.0, float y=0.0, float r=1.0)  // 构造函数，其中(x,y)为圆心位置，r 为圆的半径
void moveTo(float newX, float newY)  // 平移操作，将圆心移动到 newX，newY
float getRadius() const  // 返回圆的半径
double getCircumference() const  // 计算并返回圆的周长，π=3.14159
double getArea() const  // 计算并返回圆的面积，π=3.14159
double dist(const Circle &c) const  // 计算并返回到另一个圆的距离（圆心之间的距离）
bool isEqual(const Circle &c) const  // 判断与另一个圆是否大小相等
```

（2）私有成员：

```
float radius  // 圆的半径
```

请根据上述说明，完成 Point，Circle 两个类的定义。

注意：部分源程序已给出，仅允许在注释“Begin”和“End”之间填写内容，不得改动其他已有的任何内容。

测试样例：

输入：

```
0 0 1
3 4
```

输出：

```
初始:
c1：圆心：(0，0)，半径：1，周长：6.28318，面积：3.14159
c2：圆心：(0，0)，半径：1，周长：6.28318，面积：3.14159
c1 与 c2 圆心之间距离：0
c1 与 c2 大小是否相等：1
```

平移后：
c2：圆心：(3，4)，半径：1，周长：6.28318，面积：3.14159
c1 与 c2 圆心之间距离：5
c1 与 c2 大小是否相等：1

试题程序：
*/
```
#include<iostream>
#include<fstream>
#include<cmath>
using namespace std;
const double PI=3.14159;

/*******Begin*******/



/*******End*********/

int main() {
    float x,y,r;
    cin>>x>>y>>r;
    Circle c1(x,y,r),c2;
    cout<<"初始: "<<endl;
    cout<<"c1: 圆心: ("<<c1.getX()<<", "<<c1.getY()<<"), "<<"半径:
"<<c1.getRadius()<<", 周长: "<<c1.getCircumference()<<", 面积:
"<<c1.getArea()<<endl;
    cout<<"c2: 圆心: ("<<c2.getX()<<", "<<c2.getY()<<"), "<<"半径:
"<<c2.getRadius()<<", 周长: "<<c2.getCircumference()<<", 面积:
"<<c2.getArea()<<endl;
    cout<<"c1 与 c2 圆心之间距离："<<c1.dist(c2)<<endl;
    cout<<"c1 与 c2 大小是否相等："<<c1.isEqual(c2)<<endl<<endl;

    cin>>x>>y;
    c2.moveTo(x,y);
    cout<<"平移后: "<<endl;
    cout<<"c2: 圆心: ("<<c2.getX()<<", "<<c2.getY()<<"), "<<"半径:
"<<c2.getRadius()<<", 周长: "<<c2.getCircumference()<<", 面积:
"<<c2.getArea()<<endl;
    cout<<"c1 与 c2 圆心之间距离："<<c1.dist(c2)<<endl;
    cout<<"c1 与 c2 大小是否相等："<<c1.isEqual(c2)<<endl<<endl;

    ifstream in1("7.1.2_1-s2_in.dat");
```

```
    ofstream out1("7.1.2_1-s2_out.dat");
    while(in1>>x>>y>>r) {
        Circle c1(x,y,r),c2;
        out1<<"初始: "<<endl;
        out1<<"c1: 圆心: ("<<c1.getX()<<", "<<c1.getY()<<"), "<<"半径:
"<<c1.getRadius()<<", 周长: "<<c1.getCircumference()<<", 面积:
"<<c1.getArea()<<endl;
        out1<<"c2: 圆心: ("<<c2.getX()<<", "<<c2.getY()<<"), "<<"半径:
"<<c2.getRadius()<<", 周长: "<<c2.getCircumference()<<", 面积:
"<<c2.getArea()<<endl;
        out1<<"c1 与 c2 圆心之间距离: "<<c1.dist(c2)<<endl;
        out1<<"c1 与 c2 大小是否相等: "<<c1.isEqual(c2)<<endl<<endl;

        in1>>x>>y;
        c2.moveTo(x,y);
        out1<<"平移后: "<<endl;
        out1<<"c2: 圆心: ("<<c2.getX()<<", "<<c2.getY()<<"), "<<"半径:
"<<c2.getRadius()<<", 周长: "<<c2.getCircumference()<<", 面积:
"<<c2.getArea()<<endl;
        out1<<"c1 与 c2 圆心之间距离: "<<c1.dist(c2)<<endl;
        out1<<"c1 与 c2 大小是否相等: "<<c1.isEqual(c2)<<endl<<endl;
    }
    in1.close();
    out1.close();
    return 0;
}
class Point {
    public:
        Point(float xx,float yy): x(xx),y(yy) { }
        float getX() const { return this->x; }
        float getY() const { return this->y; }
        void setX(float newX){ this->x=newX; }
        void setY(float newY){ this->y=newY; }
    private:
        float x,y;
};

class Circle:public Point {
    public:
        Circle(float x=0.0,float y=0.0,float r=1.0):Point(x,y) {
            this->radius=r;
        }
        void moveTo(float newX,float newY) {
```

```
            setX(newX); setY(newY);
        }
        float getRadius() const { return this->radius; }
        double getCircumference() const { return 2*PI*radius; }
        double getArea() const { return PI*radius*radius; }
        double dist(const Circle &c) const {
            double x=this->getX()-c.getX(),y=this->getY()-c.getY();
            return sqrt(x*x+y*y);
        }
        bool isEqual(const Circle &c) const {
            if(this->radius==c.radius) return true;
            return false;
        }
    private:
        float radius;
};
```

程序填空 1：

```
/*
```

请将如下程序补充完整，使得程序运行时的输出结果为：

```
1, 3
5
7
9
```

注意：仅允许在指定的下划线处填写代码，不得改动程序中的其他内容（需删除下划线）。

试题源程序：

```
*/
#include <iostream>
using namespace std;

class A {
        int x,y;
    public:
        A(int a,int b) { x=a, y=b; }
        void show() { cout<<x<<", "<<y<<endl; }
};

class B: virtual protected A {
        int k;
    public:
        B(int a,int b,int c):A(a,b) { k=c; }
        void show() { cout<<k<<endl; }
```

```
};


/**********FILL**********/
class C: ___【1】___ {
        int m;
    public:
        C(int a,int b,int c):A(a,b) { m=c; }
        void show() { cout<<m<<endl; }
};


/**********FILL**********/
class D: ___【2】___ {
        int n;
    public:
/**********FILL**********/
        D(int a,int b,int c,int d, int e): ___【3】___, B(a,b,c), C(a,b,d) { n=e; }
        void show() { cout<<n<<endl; }
};

int main() {
    D d(1,3,5,7,9);
/**********FILL**********/
    ___【4】___;
    d.B::show();
    d.C::show();
    d.show();
    return 0;
}
virtual public A
   class D: public B,public C,virtual public A{
   D(int a,int b,int c,int d, int e): A(a,b), B(a,b,c), C(a,b,d) { n=e; }
   d.A::show();
```

程序填空 2：

```
/*
```

下面程序在日期（Date）类定义的基础上，重载“==”运算符，用于判断两个日期对象是否相等（对应数据值相等），若相等则返回 true，否则返回 false；重载运算符“>>”，用于以“年 月 日”的顺序输入日期对象。但程序不完整。

请将程序补充完整，使得程序运行时输出正确结果。例如，输入：

2022 12 28

2022 12 28

则输出结果为：

日期 d1：2022-12-28
日期 d2：2022-10-24
d1 和 d2 不是同一天
重置后日期 d2：2022-12-28
d1 和 d2 是同一天

注意：仅允许在指定的下划线处填写代码，不得改动程序中的其他内容（需删除下划线）。
试题源程序：

```
*/
#include <iostream>
using namespace std;

class Date {
    public:
        Date(int yy=2022,int mm=10,int dd=24);  // 构造函数，yy,mm,dd 分别用于初始化日期的年、月、日
        void setDate(int newY,int newM,int newD);  // 设置日期的年、月、日为 newY, newM, newD
        void showDate() const {  // 显示（输出）当前日期，输出格式为"年-月-日"
            cout<<this->year<<"-"<<this->month<<"-"<<this->day<<endl;
        }
        bool operator ==(const Date &) const;  // 运算符==重载

/**********FILL**********/
                【1】          (istream &, Date &);  // 运算符>>重载
    private:
        int year,month,day;  // 日期的年、月、日
};

/**********FILL**********/
         【2】          {
    this->year=yy; this->month=mm; this->day=dd;
}

void Date::setDate(int newY,int newM,int newD) {
    this->year=newY; this->month=newM; this->day=newD;
}

/**********FILL**********/
         【3】          {
    if(this->year==d.year && this->month==d.month && this->day==d.day) return true;
    else return false;
```

```
}

istream &operator >>(istream &in, Date &d) {
    in>>d.year>>d.month>>d.day;

/**********FILL**********/
            【4】        ;
}

int main() {
    int y,m,d;
    Date d1,d2;
    cin>>d1;
    cout<<"日期 d1： "; d1.showDate();
    cout<<"日期 d2： "; d2.showDate();
    if(d1==d2) cout<<"d1 和 d2 是同一天"<<endl;
    else cout<<"d1 和 d2 不是同一天"<<endl;
    cin>>y>>m>>d;
    d2.setDate(y,m,d);
    cout<<"重置后日期 d2： "; d2.showDate();
    if(d1==d2) cout<<"d1 和 d2 是同一天"<<endl;
    else cout<<"d1 和 d2 不是同一天"<<endl;
    return 0;
}
friend istream &operator >>
   Date::Date(int yy,int mm,int dd){
   bool Date::operator ==(const Date &d) const{
```

第十章

1.关于 this 指针使用说法正确的是（

保证每个对象拥有自己的数据成员，但共享处理这些数据的代码

2.为了取代 C 语言中带参数的宏，在 C++中使用（）。

内联函数

3.下列代码段中声明了 3 个类：

```
class Person {}
class Student public Person {}
class Undergraduate Student }
```

下列关于这些类之间关系的描述中，错误的是（）。

类 Undergraduate 从类 Student 公有继承

4.下列有关 C++流的表述中，错误的是（

包含头文件 iostream 后，就可以利用 c++流的任何操作符了

5.设有定义 intx:float y:,则 10+z+y 值的数据类型是（）。

float

6.以私有方式派生时，基类中的公有成员和保护成员在派生类中（

均成为私有成员

7.下列运算符中，()运算符在 C++中不能重载。

?:

8.以下关键字不能用来声明类成员的访问权限的是（

static

9.如下程序运行时的输出结果是

```
#include〈iostream>
using namespace std;
class A
public:
int a;
A(int x=0){a=x;cout<<"A:a="<<a<<endl;
}:
ass B:public A
public:
int a;
B(int x,int y=5):A(x){a=y:cout<<"B:a="<<a<<endl;
}:
int main（）{
Bb(3):
A &r=b;
cout<<"a="<<r.a<<endl;
return 0;
```

10 30

40 -20

10.引用调用的调用方式是指（）。

形参是引用，实参是变量

11.以私有方式派生时，基类中的公有成员和保护成员在派生类中（）。

均成为私有成员

12.关于动态内存分配，对 delete?运算符的下列说法中，错误的是（）。

对一个指针变量可任意多次使用该运算符

13.在类的定义中，用于为对象分配内存空间，对类的数据成员进行初始化并执行其他内部管理操作的函数是（）。

构造函数

14.如下程序运行时的输出结果是（

```
#include〈iostream>
using namespace std;
class A
  public:
    int a:
    A(int x=0){a=x:cout<<"A:a="<<a<<endl;
};
class B:public A {
  public:
    int a:
    B(int x,int y=5):A(x)a=y;cout<<"B:a="<<a<<end1;
}:
int main（）{
    B b (3):
    A &r=b;
  cout<<"a="<<r.a<<end1;
  return 0;
}
```

A:a=3

B:a=5

a=3

15.如下程序的输出结果是(

```
#include〈iostream>
using namespace std;
int main（）{
    int k=1,n=26;
    do{k*=n%10:n/=10:}whi1e(n):
```

```
    cout<<k<<end1;
    return 0;
}
```

12

16.若有定义：int x[5],*p=x; 则 p 的值为（）。

数组 x 的首元素

17.设有定义 intx:f1oaty:,则 10+x+y 值的数据类型是（）。

float

18.关于在调用模板函数时模板实参的使用，下列表述中不正确的是（）。

对于常规参数所对应的模板实参，任何情况下都不能省略

19.若 A 为一个类，a 为该类的非静态数据成员，在该类的一个成员函数定义中访问 a 时，其书写格式为（）。

a

20.下列代码段中声明了 3 个类：

```
class Person {}
Class Student public Person {}
Class Undergraduate Student }
```

下列关于这些类之间关系的描述中，错误的是（

类 Undergraduate,从类 Student 公有继承

21.类的析构函数的作用是（）。

删除类创建的对象

22.已知类 A 中的一个成员函数说明为 void fun(A&a):,则 A 是 a 的含义是（）。

a 是类 A 的对象引用，用来做函数 fun0 的形参

23.为了取代 C 语言中带参数的宏，在 C++中使用（）。

内联函数

24.如下程序运行时输出的前两行分别是（

```
#include〈iostream>
using namespace std;
class test
  private:
    int a;
public:
  test（）{cout<<"constructor"<<endl;
  test(int a){cout<<"constructor "<<a<<endl:
  ~test（）cout<<"destructor"<<endl:
}
int main（）{
  test A(3)
  return 0;}
```

```
}
```

constructor 3

destructor

25.下面程序运行时的输出结果是（）.

```
#include〈iostream>
using namespace std;
int i=10;
int f1(int j){
    static int i=20;
    j=i--:
    return j;
}
int f2(int i){
    int j=15;
    return i=j+=i;
}
int main（）{
    for (int j=1;j<3;j++)
        cout<<f1(i)<<","<<f2(i+j)<<end1;
    return 0;
}
```

20 26

19 27

程序改错 1：

```
/*
```

请改正程序中指定位置的错误，使程序的输出结果如下：

```
The computer id is 101
The type of monitor is A
```

注意：只允许修改注释"ERROR"的下一行，不得改动程序中的其他内容，也不允许增加或删减语句。

源程序清单：

```
*/
#include<iostream>
using namespace std;

class Monitor {
    public:
```

```
        Monitor(char t) { type=t; }

/**********ERROR**********/
        void display()
        { cout<<"The type of monitor is "<<type<<endl; }
    private:
        char type;
};

class Computer {
    public:

/**********ERROR**********/
        Computer(int i, char c):mon=c
        { id=i; }

/**********ERROR**********/
        void Display()
        {   cout<<"The computer id is "<<id<<endl;

/**********ERROR**********/
            display();
        }
    private:
        int id;
        Monitor mon;
};

int main() {
    const Computer myComputer(101, 'A');
    myComputer.Display();
    return 0;
}
void display() const
   Computer(int i, char c):mon(c)
   void Display() const
   mon.display();
```

程序改错 2：

/*

请改正程序中指定位置的错误，使程序的输出结果如下：

45123

65123

注意：只允许修改注释“ERROR”的下一行，不得改动程序中的其他内容，也不允许增加或删减语句。

源程序：

*/

```
#include<iostream>
using namespace std;

class BaseClass {
    public:
        int x;
};

/**********ERROR**********/
class ClassA::BaseClass
{   protected:
        int y;
};

class ClassB:public BaseClass {
    protected:
        int z;
};


/**********ERROR**********/
class Derived::public ClassA; ClassB
{   public:
        int x;
        Derived() {
            x=1,y=2,z=3;
/**********ERROR**********/
```

```
            BaseClass::x=4;
        }
    void Display() {
        cout<<ClassA::x<<ClassB::x<<x<<y<<z<<endl;
    }
};

int main() {
    Derived d;
    d.Display();
/**********ERROR**********/
    d.x=6;
    d.Display();
    return 0;
}
class ClassA:public BaseClass
   class Derived:public ClassA,public ClassB
   ClassA::x=4, ClassB::x=5
   d.ClassA::x=6
```

程序填空 1：

```
/*
```

请将如下程序补充完整，使得程序运行时的输出结果为：

```
BCA
4, 2, 6
4; 2; 6
```

注意：仅允许在指定的下划线处填写内容，并删除下划线及其编号，不允许增加或删除语句，也不得改动程序中的其他部分。

试题源程序：

```
*/
#include <iostream>
using namespace std;

class Base1 {
    public:
        Base1(int x) { b1=x; cout<<"A"; }
        int b1;
```

```
};

class Base2 {
    public:
        Base2() { b2+=2; cout<<"B"; }

/**********FILL**********/
              [1]      ;
};

int Base2::b2=0;

class Base3 {
    public:
        Base3(int y):b3(y) { cout<<"C"; }
        int b3;
};

/**********FILL**********/
class Derived       [2]       {
    public:

/**********FILL**********/
        Derived(int x,int y):       [3]       { }

/**********FILL**********/
        void       [4]       {
            cout<<b1<<", "<<b2<<", "<<b3<<endl; }
};

int main() {
    const Derived d(4,6);
    cout<<endl;
    d.Display();
    cout<<d.b1<<"; "<<d.b2<<"; "<<d.b3<<endl;
    return 0;
}
```

```
virtual void display()
   void display() const
   p.display()
   const Derive
```

程序填空 2：

```
/*
请将如下程序补充完整，使得程序运行时的输出结果为：
Base::display
Derive::display
Derive::display const

注意：仅允许在指定的下划线处填写内容，并删除下划线及其编号，不允许增加或删除语句，
也不得改动程序中的其他部分。
试题源程序如下：
*/
#include <iostream>
using namespace std;

class Base {
    public:

/**********FILL**********/
              [1]       {
           cout<<"Base::display"<< endl;
       }
};

class Derive:public Base {
    public:

/**********FILL**********/
               [2]       ;
       void display() { cout<<"Derive::display"<<endl; }
};
void Derive::display() const {
    cout<<"Derive::display const"<<endl; }
```

```
void Display(Base &p)

/**********FILL**********/
{        [3]       ; }

int main() {
    Base b1; Display(b1);
    Derive d1; Display(d1);

/**********FILL**********/
           [4]        d2;
    d2.display();
    return 0;
}
static int b2;
   :public Base2,public Base3,public Base1
   Base1(x),Base3(y){}
   Display() const
```

程序设计 1：

```
/*
```

点（Point）类成员如下：

（1）公有成员：

```
Point(float xx, float yy)   // 构造函数，初始化点的 x，y 坐标
float getX()   // 返回横坐标 x
float getY()   // 返回纵坐标 y
void moveTo(float newX, float newY)  // 将点的 x，y 坐标移动到 newX，newY
```

（2）私有成员：

```
float x, y   // 点的横坐标 x，纵坐标 y
```

在此基础上，定义矩形（Rectangle）类，其成员如下：

（1）公有成员：

```
Rectangle(float x1=0.0,float y1=0.0,float x2=3.0,float y2=4.0) // 构造函数，
```

初始化矩形的数据成员，其中(x1,y1)为左下角坐标，(x2,y2)为右上角坐标

```
void resetRect(float newX1,float newY1,float newX2,float newY2) // 重置矩形
```

左下角 p1(newX1,newY1)，右上角 p2(newX2,newY2)

```
double getArea() // 返回矩形的面积
double getCircumference() // 返回矩形的周长
bool isSquare() // 判断是否为正方形
```

（2）私有成员：

Point p1,p2　// 矩形左下角 p1 和右上角 p2

double area, circumference　// 矩形的面积，周长

请根据上述说明，完成 Point，Rectangle 两个类的定义。

注意：部分源程序给出，仅允许在注释“Begin”和“End”之间填写内容，不得改动 main 函数和其他已有的任何内容。

试题程序：

```
*/
#include<iostream>
#include<fstream>
using namespace std;

/*******Begin*******/




/*******End*********/

int main() {
    float x1,y1,x2,y2;
    cin>>x1>>y1>>x2>>y2;
    Rectangle rect1(x1,y1,x2,y2),rect2;
    cout<<"rect1 面 积 : "<<rect1.getArea()<<" 周 长 :
"<<rect1.getCircumference()<<endl;
    cout<<"是否正方形: "<<rect1.isSquare()<<endl;
    cout<<"rect2 面 积 : "<<rect2.getArea()<<" 周 长 :
"<<rect2.getCircumference()<<endl;
    cout<<"是否正方形: "<<rect2.isSquare()<<endl;
    cin>>x1>>y1>>x2>>y2;
    rect2.resetRect(x1,y1,x2,y2);
    cout<<" 重 置 后 :\nrect2 面 积 : "<<rect2.getArea()<<" 周 长 :
"<<rect2.getCircumference()<<endl;
    cout<<"是否正方形: "<<rect2.isSquare()<<endl;
```

```
    ifstream in1("4.2.1_4-2_in.dat");
    ofstream out1("4.2.1_4-2_out.dat");
    while(in1>>x1>>y1>>x2>>y2) {
        Rectangle rect1(x1,y1,x2,y2),rect2;
        out1<<"rect1 面 积 : "<<rect1.getArea()<<" 周 长 :
"<<rect1.getCircumference()<<endl;
        out1<<"是否正方形: "<<rect1.isSquare()<<endl;
        out1<<"rect2 面 积 : "<<rect2.getArea()<<" 周 长 :
"<<rect2.getCircumference()<<endl;
        out1<<"是否正方形: "<<rect2.isSquare()<<endl;
        in1>>x1>>y1>>x2>>y2;
        rect2.resetRect(x1,y1,x2,y2);
        out1<<" 重 置 后 :\nrect2 面 积 : "<<rect2.getArea()<<" 周 长 :
"<<rect2.getCircumference()<<endl;
        out1<<"是否正方形: "<<rect2.isSquare()<<endl<<endl;
    }
    in1.close();
    out1.close();
    return 0;
}
class Point {
    public:
        Point(float xx,float yy): x(xx), y(yy) { }
        float getX() { return x; }
        float getY() { return y; }
        void moveTo(float newX,float newY) { x=newX,y=newY; }
    private:
        float x,y;
};

class Rectangle {
    public:
        Rectangle(float x1=0.0,float y1=0.0,float x2=3.0,float
y2=4.0):p1(x1,y1),p2(x2,y2) {
            area=(p2.getX()-p1.getX())*(p2.getY()-p1.getY());
            circumference=2*(p2.getX()-p1.getX()+p2.getY()-p1.getY());
        }
```

```
        void resetRect(float newX1,float newY1,float newX2,float newY2) {
            p1.moveTo(newX1, newY1);
            p2.moveTo(newX2, newY2);
            area=(p2.getX()-p1.getX())*(p2.getY()-p1.getY());
            circumference=2*(p2.getX()-p1.getX()+p2.getY()-p1.getY());
        }
        double getArea() { return area; }
        double getCircumference() { return circumference; }
        bool isSquare() {
            if(p2.getX()-p1.getX()==p2.getY()-p1.getY()) return true;
            return false;
        }
    private:
        Point p1,p2;
        double area,circumference;
};
```

程序设计 2：

```
/*
日期（Date）类成员声明如下：
class Date {
    public:
        Date(int yy=2023,int mm=1,int dd=1); //构造函数，yy，mm，dd 分别用于初始化日期的年、月、日
        void setDate(int newY,int newM,int newD); // 设置日期的年、月、日为 newY，newM，newD
        void showDate() const; // 显示（输出）当前日期，输出格式为"年-月-日"
    protected:
        int year,month,day;  // 日期的年、月、日
};
```

请参照注释，完成以下任务：

（1）完成 Date 类的定义。

（2）以成员函数形式重载“==”运算符，用于判断两个日期对象是否相等（对应数据值相等），若相等则返回 true，否则返回 false。

（3）重载“>>”运算符，用于日期对象的输入（以“年 月 日”的格式输入）。

注意：部分源程序已给出，仅允许在注释“Begin”和“End”之间补全代码，不得改动其他已有的任何内容。

测试样例：

输入：

2023 4 10

2023 5 10

2023 4 10

输出：

d1 的值为：2023-4-10

d2 的值为：2023-5-10

d1 和 d2 不是同一天

重置后 d2 的值为：2023-4-10

d1 和 d2 是同一天

试题程序：

```
*/
#include <iostream>
#include<fstream>
using namespace std;

/*******Begin*******/



/*******End*********/

void Date::showDate() const { // 显示（输出）当前日期，输出格式为"年-月-日"
    cout<<this->year<<"-"<<this->month<<"-"<<this->day<<endl;
}

int main() {
    int y,m,d;
    Date d1,d2;
    cin>>d1>>d2;
    cout<<"d1 的值为："; d1.showDate();
    cout<<"d2 的值为："; d2.showDate();
    if(d1==d2) cout<<"d1 和 d2 是同一天"<<endl;
    else cout<<"d1 和 d2 不是同一天"<<endl;
```

```
    cin>>y>>m>>d;
    d2.setDate(y,m,d);
    cout<<"重置后 d2 的值为："; d2.showDate();
    if(d1.operator ==(d2)) cout<<"d1 和 d2 是同一天"<<endl;
    else cout<<"d1 和 d2 不是同一天"<<endl;

    ifstream in1("8.2.2.5_2-s2_in.dat");
    ofstream out1("8.2.2.5_2-s2_out.dat");
    streambuf *cinbackup;
    streambuf *coutbackup;
    cinbackup=cin.rdbuf(in1.rdbuf());
    coutbackup=cout.rdbuf(out1.rdbuf());
    while(cin>>d1>>d2) {
        cout<<"d1 的值为："; d1.showDate();
        cout<<"d2 的值为："; d2.showDate();
        if(d1==d2) cout<<"d1 和 d2 是同一天"<<endl;
        else cout<<"d1 和 d2 不是同一天"<<endl;
        cin>>y>>m>>d;
        d2.setDate(y,m,d);
        cout<<"重置后 d2 的值为："; d2.showDate();
        if(d1.operator ==(d2)) cout<<"d1 和 d2 是同一天"<<endl;
        else cout<<"d1 和 d2 不是同一天"<<endl;
        cout<<endl;
    }
    cin.rdbuf(cinbackup);
    cout.rdbuf(coutbackup);
    in1.close();
    out1.close();
    return 0;
}
class Date {
    public:
        Date(int yy=2023,int mm=1,int dd=1); //构造函数，yy,mm,dd 分别用于初始化日期的年、月、日
        void setDate(int newY,int newM,int newD); // 设置日期的年、月、日为 newY, newM, newD
        void showDate() const; // 显示（输出）当前日期，输出格式为"年-月-日"
```

```
        bool operator ==(const Date &) const; // 判断两个日期是否相等
        friend istream &operator >>(istream &, Date &); //运算符>>重载
    protected:
        int year,month,day;  // 日期的年、月、日
};

Date::Date(int yy,int mm,int dd) { //构造函数实现
    this->year=yy; this->month=mm; this->day=dd;
}
void Date::setDate(int newY,int newM,int newD) { // 设置日期的年、月、日为 newY, newM, newD
    this->year=newY; this->month=newM; this->day=newD;
}
bool Date::operator ==(const Date &d) const {
    if(year==d.year && month==d.month && day==d.day) return true;
    else return false;
}

istream &operator >>(istream &in, Date &d){ // 运算符>>重载函数实现
    in>>d.year>>d.month>>d.day;
    return in;
}
```