温州大学（筹）期末考试试卷

2005—2006 学年第一学期

考试科目	面向对象程序设计 A	考试成绩	
试卷类型	AB 卷		
考试形式	开卷		
考试对象	04 计本 1、2 班		

学号 _____ 姓名 _____ 班级 _____ 学院 _____

1、找出以下程序中的语法错误，并作出解释。程序可能有多个错误。如果没有错误，请注明程序正确。（每小题 3 分，共 30 分）

a) 程序

```
#include <stdlib.h>
int main() {
    int *p1 = new int[10];
    int *p2 = malloc(sizeof(int)*10);
    free(p1);
    delete [] p2;
}
```

b) 程序

```
#include <iostream>
using namespace std;
class N {
    static int n = 0;
};
int main() {
    cout << N::n << endl;
}
```

c) 程序

```
class Date {
public:
    void Date(int y, int m, int d) { year = y; month = m; day = d; }
    int GetYear() const { return year++; }
    int GetMonth() const { return month; }
    int GetDay() const { return day; }
    int GetAnotherDay() const { return GetMonth();}
private:
    int day, month, year;
};
```

d) 程序

```
class A {
public:
    static int f() { return 0; }
    int m() { return 1; }
};
int main() {
    A *p = new A;
    A a();
    p.f();
    A::f();
    a.f();
    p.m();
    A::m();
    a.m();
}
```

e) 程序

```
#include <iostream>
using namespace std;
int main() {
    const int size;
    size = 10;
    int array[size];
    for( int i = 0; i < size; i++)
        cin >> array[i];
    int sum;
    for( int i = 0; i < size; i++) {
        cout << setw(10) << array[i];
        sum += array[i] * array[i];
    }
    cout << endl << "square sum of these numbers = " << sum << endl;
}
```

f) 程序

```
#include <string>
#include <iostream>
using namespace std;
class S {
public:
    S(int a) { cout << a << endl; }
    S(double a = 0.0, int b) { cout << a << b << endl; }
    S(char *p) { cout << p << endl;}
};
int main() {
    S a(1);
    S b;
    S c(0.0, 1);
    S d("hello");
    S e(true);
    S f(a);
    e = a;
}
```

g) 程序

```
#include <string>
using namespace std;
class S {
public:
    S(int &r) : a(0), b(1), c("abc"), d(r), p(new int) { }
private:
    int a;
    const int b;
    string c;
    int &d;
    int *p;
};
int main() {
    int r;
    S s(r);
}
```

h) 程序

```
#include <string>
#include <iostream>
using namespace std;
class Person{
public:
    Person(string name) : name_ (name) { }
protected:
    string name_;
};
class Student : public Person {
    Student(string name, string id) { name_ = name; id_ = id; }
    bool compare(Person p) { return p.name_ == name_; }
    void print() { cout << name_ << id_ << endl; }
private:
```

```
    string id_;
};
```

i) 程序

```
class B {
public:
    virtual void f() = 0;
};
class D : public B {
public:
    virtual int f(int a) { return a; }
};
int main() {
    B b;
    D d;
}
```

j) 程序

```
class Integer{
public:
    Integer(int a) : i(a) {}
    Integer operator+(int a, int b) { return a + b;}
    Integer operator-() { return -i; }
private:
    int i;
};
Integer operator-(Integer a, Integer b) {
    return a.i - b.i;
}
int main() {
    Integer i(-1);
    i = -i;
    i = i - 1;
```

```
    i = i + 10;
}
```

2、读程序，写出程序运行的结果（每小题3分，共30分）

k) 引用参数

```
#include <iostream>
using namespace std;
int f( int &a) {
    return ++a;
}
int g( int a) {
    return ++a;
}
int main() {
    int b = 10;
    cout << g(b) << endl;
    cout << f(b) << endl;
    cout << g(b) << endl;
}
```

结果:

l) 函数重载与函数缺省值

```
#include <iostream>
using namespace std;
int f( string a );
int f( int b , int c = 1);
int f( bool b);
int main() {
    cout << f(1, 2) << endl;
    cout << f(false) << endl;
```

```
    cout << f(0) << endl;
}
int f( string a) {
    return a.length();
}
int f( int b, int c) {
    return b * c + b + c + 1;
}
int f(bool b) {
    return b?1:-1;
}
```

m) 对象的构造和析构

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

class Z {
public:
    Z(int i) : id(i) { cout << id << " created" << endl; }
    ~Z() { cout << id << " destroyed" << endl; }
private:
    int id;
};
int main() {
    Z z1(1);
    Z z2(2);
    Z *z3 = new Z(3);
}
```

结果:

n) 拷贝构造函数、全局对象

```
#include <iostream>
```

```
using namespace std;

class Z {
public:
    Z(int i) : id(i) { cout << id << " created" << endl; }
    ~Z() { cout << id << " destroyed" << endl; }
    Z(Z &z) : id(z.id) { cout << id << " created a copy" << endl; }
private:
    int id;
};

Z z1(1);

int main() {
    cout << "function main start" << endl;
    Z z2(2);
    Z z4(z2);
}
```

o) 组合对象的构造与析构

```
#include <iostream>
using namespace std;
class X {
public:
    X() { cout << "create X" << endl; }
    ~X() { cout << "destroy X" << endl; }
};
class Y {
public:
    Y() { cout << "create Y" << endl; }
    ~Y() { cout << "destroy Y" << endl; }
private:
    X x;
};
int main() {
    Y y;
}
```

结果:

p) 派生类对象的构造与析构

```
#include <iostream>
using namespace std;
class X {
public:
    X() { cout << "create X" << endl; }
    ~X() { cout << "destroy X" << endl; }
};
class Y : public X{
public:
    Y() { cout << "create Y" << endl; }
    ~Y() { cout << "destroy Y" << endl; }
};
int main() {
    Y y;
}
```

结果:

q) 虚函数

```
#include <iostream>
using namespace std;
class B{
public:
    virtual void f() { cout << "B::f()" << endl; }
    void m() { cout << "B::m()" << endl; }
};
class D : public B {
public:
    void f() { cout << "D::f()" << endl; }
    void m() { cout << "D::m()" << endl; }
};
```

```
int main() {
    B *p = new D;
    p->f();
    p->m();
    B &r = *p;
    r.f();
    r.m();
    delete p;
};
```

r) 虚函数覆盖与重载

```
#include <iostream>
using namespace std;
class B{
public:
    virtual void f() { cout << "B::f()" << endl; }
    virtual void f(int a) { cout << "B::f(int)" << endl; }
};
class D : public B {
public:
    virtual void f() { cout << "D::f()" << endl; }
    virtual void f(bool a) { cout << "D::f(bool)" << endl; }
};
int main() {
    B *p = new B;
    p->f();
    p->f(0);
    delete p;
    p = new D;
    p->f();
    p->f(0);
    delete p;
    D *d = new D;
    d->f();
    d->f(0);
    delete d;
};
```

结果:

s) 虚析构函数

```
#include <iostream>
using namespace std;
class Z{
public:
    Z () { cout << "create Z" << endl; }
    virtual ~Z() { cout << "destroy Z" << endl; }
};
class Y : public Z {
public:
    Y () { cout << "create Y" << endl; }
    virtual ~Y() { cout << "destroy  Y" << endl; }
};
int main() {
    Z *p = new Z;
    delete p;
    p = new Y;
    delete p;
}
```

结果

t) 赋值运算符重载

```
#include <iostream>
using namespace std;
class D {
public:
    D() { cout << "D created" << endl; }
    D(D&) { cout << "D created a copy" << endl; }
    ~D() { cout << "D destroyed" << endl; }
    D& operator=(D&) { cout << "D assigned" << endl; return *this; }
};
```

```
int main() {
    D d1;
    D d2(d1);
    d1 = d2;
}
```

结果

3、程序填空题，按要求填写程序（每空 2 分，共 20 分）

u) 学校的人事关系

学校里面有四类人：人（Person），学生（Student），教师（Teacher），助教（Assistant）。其中助教是一类特殊的人，他们是教师，同时也是学生。请完成以下程序，使之能得到正确运行结果。

```
#include <iostream>
#include <string>
using namespace std;
class Person {
public:
    Person(string name) : name_(name) { }
    virtual void print() const { cout << name_; }
protected:
    string name_;
};
class Student : virtual public Person {
public:
    Student(string name, string id) : ①_____  { }
    void print() const { cout << name_<< "\t" << id_; }
protected:
    string id_;
};
class Teacher : virtual public Person {
public:
    Teacher(string name, string title) : ②_____{ }
    void print() const { cout << name_ << "\t" << title_;}
protected:
    string title_;
```

```
};
class Assistant : ③_____________________{
public:
    Assistant(string name, string id, string title)
    : ④_____________________________
    {
    }
    void print() const {
        cout << ⑤_____________________________________________;
    }
};
int main() {
    Person *p[4];
    p[0] = new Person("jack");
    p[1] = new Student("rose", "9710073");
    p[2] = new Teacher("czk", "professor");
    p[3] = new Assistant("oldbig", "9815015", "assistant");
    for(int i= 0; i < 4; i++) {
        p[i] -> print();
        cout << endl;
    }
}
```

程序输出结果:

```
jack
rose    9710073
czk     professor
oldbig  9815015 assistant
```

v) 实现 Stack 类

栈 Stack 是一种存放数据（int）的容器。它提供 push（把一个元素放入栈中），pop（把栈中最后一个元素删去），top（取栈中最后一个元素）。栈是一种后进先出的数据结构，即后入栈（push）的元素在出栈（pop）的时候会先被删掉。Stack 类的实现如下，请补充完整，使其能够得到正确的运行结果。

```
#include <iostream>
using namespace std;
class Stack {
public:
    Stack(int capacity) : capacity_ (capacity), size_(0) {
        array_ = new int[ capacity ];
    }
```

```
    ~Stack() { ⑥_____________________;  }
    int size() const { return size_; }
    void push(int i) { array_[ ⑦_____________ ] = i; }
    void pop() { --size_; }
    int &top() { return array_ [ size_ - 1]; }
    Stack(Stack &s) { copy(s); }
    Stack &operator=(Stack &s) {
        if(this == &s) return *this;

        ⑧_____________________;

        copy(s);
        return *this;
    }
private:

    int capacity_; //栈的容量（能存放的最大元素个数）

    int size_; //元素个数

    int *array_;  //存放元素的空间

    void copy(Stack &s) {
        capacity_ = s.capacity_;
        size_ = s.size_;

        ⑨_____________________;

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

            ⑩_____________________;

    }
};
int main() {
    Stack s1(5);
    for(int i = 0; i < 3; i++) s1.push(i);
    Stack s2(s1);
    while(s1.size() > 0) {
        cout << s1.top() << endl;
        s1.pop();
    }
    s1 = s2;
    while(s1.size() > 0) {
        cout << s1.top() << endl;
        s1.pop();
    }
}
```

程序输出结果

```
2
1
0
2
1
0
```

4、编程题，按要求编写程序（每小题 10 分，共 20 分）

w) 编写 Clock 类

编写一个时钟类 Clock，并实现运算符++（时间增加 1 秒），-（计算两个时间差多少秒），+（计算过一定秒数以后的新时间），使得如下主程序可以产生后面的输出。Clock 类的声明写在 clock.h 中，Clock 类成员函数的定义写在 clock.cpp 中。

以下程序的输出结果:

```
20:0:0
20:0:2
8
20:8:8
```

main.cpp 内容:

```
#include <iostream>
using namespace std;
#include "clock.h"
int main() {
    Clock c1 ( 20, 8, 8 ); //20 点 8 分 8 秒

    Clock c2 ( 20, 8 ); //20 点 8 分 0 秒

    Clock c3(20);  // 20 点整
    cout  << c3++ << endl;
    cout <<  ++c3 << endl;
    cout << c1 - c2 << endl;
    cout << c2 + 8 << endl;
    return 0;
```

}

clock.h 的内容:

//请在这里给出 Clock 类的声明

clock.cpp 的内容:

//请在这里给出 Clock 类成员函数的实现:

x) 编写一个计算各种图形面积的程序。

下列 shape 类是一个表示图形的抽象类，area( )为求图形面积的函数，total( )则是一个通用的用以求不同形状的图形面积总和的函数。请从 shape 类派生三角形类(triangle)、矩形类（rectangle）、圆形类（circle），并给出具体的求面积函数，使以下程序能够运行并得到正确结果。每个类都写在不同的文件里面，类的成员函数的定义写在类的声明里面。

以下程序的运行结果为：

40.2743

shape.h 的内容：

```
#ifndef SHAPE_H
#define SHAPE_H
class shape{
public:
    virtual double area( ) = 0;
};
#endif
```

main.cpp 的内容：

```
#include <iostream>
using namespace std;
#include "shape.h"
#include "triangle.h"
#include "circle.h"
#include "rectangle.h"

double total(shape *s[ ],int n) {
    double sum = 0.0;
    for(int i=0;i<n;i++)
        sum += s[i]->area();
    return sum;
```

```
}
int main() {
   shape *shapes[3];
   shapes[0] = new circle(3.0); //半径3.0
   shapes[1] = new rectangle(2.0, 3.0); //长和宽分别为2.0和3.0
   shapes[2] = new triangle(2.0, 3.0, 3.14159/2.0);
   //三角形两边长分别为2.0和3.0，这两边的夹角的弧度为3.14159/2.0
   cout << total(shapes, 3) << endl; //输出这三个图形的面积总和
   return 0;
}
```

rectangle.h 的内容：

//请在下面写出 rectangle.h

circle.h 的内容：

//请在下面写出 circle.h

triangle.h 的内容：

//请在下面写出 triangle.h