# 西 安 电 子 科 技 大 学

考试时间__120__分钟

# 试　　题

| 题号 | 一 | 二 | 三 | 四 | 总分 |
|---|---|---|---|---|---|
| 分数 | | | | | |

1. 考试形式：闭卷； 2. 本试卷共 四 大题，满分 100 分。

班级________学号____________姓名___________任课教师____________

## Part I　There is one error in each code paragraph. Find out the error and write down the error statement on your answer sheet. (20 points)

(1)
```
int f(const int x, int y){
   int temp ;
   x += y;
   return x;
}
```

(2)
```
int f(double x, int i = 0, char c);
void g(){
      cout << f(23.5, 10) << endl;
}
```

(3)
```
namespace a{
   float   x;
}
namespace b{
   int i;
   float   x;
};
using namespace a::x=1;
```

(4)
```
class C {
   friend C operator+ (const C&, const C&);
   /* …… */
};
C C::operator+ (const C& c1, const C& c2)
{   /* …… */   }
```

(5)
```
class Base{
public:
   virtual void f( ){ }
   virtual int g( ) =0;
};
void f(){
   Base a;
}
```

(6)
```
class C{
   int x;
   void setx(int a) { /* … */}
};
void f() {
   C c1;
   c1.setx(3);
}
```

(7)

```
template <class T, int x>
class Array {
public:
   void m();
   // ……
};

void f() {
   int a;
   Array<double, a> ar;
}
```

(8)

```
class Base {
protected:
   int x;
public:
   Base(int xx){ x = xx; }
};
class Sub : public Base {
   char c;
   Sub(int x1, char c1) {
     x = x1; c = c1;}
};
```

(9)

```
class C {
public:
   void m() {/* … */}
   static void s() {/* … */}
};
void   f() {
   C c1;
   c1.m();
   C::m();
   c1.s();
   C::s();
}
```

(10)

```
class Parent {
   int x;
public:
   int a;
   int b;
};
class Son: public Parent {
public:
   int f () const {
      int c = a+b;
      return x;
   }
};
```

## **Part II  Write the following programs' output.** ( 30 points )

**1. (6 points)**

```
#include <iostream>
using namespace std;
void func(int x, int& y, int *jia){
    y *= x + 2;
    *jia = x + y;
}
int main( ){
   int i = 10, j = 4, x1 = 1;
   func(i, j, &x1);
   cout << i << "," << j << "," << x1 << endl;
   return 0;
}
```

**2. (6 points)**

```
#include <iostream>
using namespace std;
class Point {
private:
        int x, y;
public:
         Point(int i, int j) { x = i;    y = j; }
         void Print() { cout << '(' << x << ','<< y << ')' << endl; }
         void operator += (Point p) { x += p.x; y += p.y; }
         void operator -= (Point p) { x -= p.x; y -= p.y; }
};
int main() {
      Point P1(9, 8), P2(4, 6);
      P1.Print();
      P2.Print();
      P1 += P2;
      P1.Print();
      P2 -= P1;
      P2.Print();
      return 0;
}
```

**3. (6 points)-**

```
#include <iostream>
using namespace std;
class A {
    static int obj_count;
public:
    A()      { obj_count++; }
    ~A()    { obj_count--; }
    int get_num_of_objects()    { return obj_count; }
};
int A::obj_count = 0;
A a;
int main() {
    A    b, *p, *q;
    p = new A;
    q = new A[5];
    cout << a. get_num_of_objects() << '\t';
    delete []q;
    cout << p->get_num_of_objects() << '\t';
    for(int i = 0; i < 2; i++)    {
        A c;
        cout << c.get_num_of_objects() << '\t';
    }
    delete p;
    cout << b.get_num_of_objects() << endl;
    return 0;
}
```

**4. (6 points)**

```
#include <iostream>
using namespace std;
int main() {
        try {
                int a = 9;
                throw a;
                float f = 0.5F;
                throw f;
        }
        catch (float k) {
                cout << "Exception occured here -- float!\n";
        }
        catch (int k) {
                cout << "Exception occured here -- int!\n";
        }
        cout << "Succeed!\n";
        return 0;
}
```

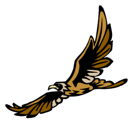
**5. (6 points)**

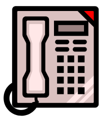
```
#include <iostream>
using namespace std;
class BASE{
protected:
    int id;
public:
    BASE() : id(0) {}
    int   update(int n)   { id += n; return id; }
    virtual void hello(){ cout << "BASE" << endl; }
};
class DERIVED : public BASE {
public:
    DERIVED () { id = 1;}
    int update(int n) { id += 2*n; return id;}
    void hello() { cout << "DERIVED " << endl; }
};
int main ()   {
    BASE* objs[2];
    objs[0] = new BASE();      objs[1] = new DERIVED ();
    for(int i=0; i<2; i++) {
        objs[i]->hello();
        cout << objs[i]->update(10) << endl;
    }
    return 0;
}
```

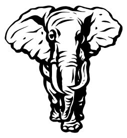
## **Part III   Object-Oriented Analyzing and Designing** ( 30 points )

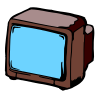
**1. (15 points)**

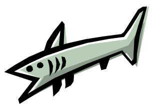
**From following named pictures, please analyze and design the class hierarchies.**

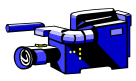

Eagle


Telephone


Elephant


Television


Shark


Camera

**2. (15 points)**

Please define a class **DoubleValue** that wraps*(包装)* a value of primitive type ***double*** and satisfies the following requirements:

(1) it has a default constructor which sets the value to 0.0;

(2) it has a constructor with one argument of type ***double*** that is wrapped;

(3) by overloading the operator "==", it can compare ***this*** object against another

specified **DoubleValue** object, and return true if and only if both DoubleValue represent the same double value;
(4) it can return a string representation of the wrapped double value;
(5) it can return the value of this DoubleValue as an ***int*** type after a narrowing primitive conversion.

## **Part IV　Programming** ( 20 points )

**1. (10 points)**

Implement a class ***Integer*** that can substitute the basic int type in C++. The interfaces of the class ***Integer*** SHOULD output the messages or input data shown in the following program's comments.

```
#include <iostream>
using namespace std;
int main() {
   Integer   a, b = 10, c(b);
   cout << "a=" << a << endl;       // Display:   a=0
   cout << "b=" << b << endl;       // Display:   b=10
   cout << "c=" << c << endl;       // Display:   c=10
   cin >> c;                        // input 2 from keyboard
   cout << "c=" << c << endl;       // Display: c=2
   c = b + 90;
   cout << "b=" << b << " c=" << c << endl;     // Display: b=10 c=100
   a = b - 100;
   cout << "a=" << a << " b=" << b << endl;     // Display: a=-90 b=10
   c = a / b;
   cout << "a=" << a << " b=" << b << " c=" << c   << endl;
   //Display: a=-90 b=10 c=-9
   c = b *= a;
   cout << "a=" << a << " b=" << b << " c=" << c << endl;
   //Display: a=-90 b=-900 c=-900
   return 0;
}
```

***Hint:*** Operator "<<" and ">>" can be overloaded as followings：

```
ostream& operator<< ( ostream& out,   Integer& I ){
      out << I.value;       return out;
}
```

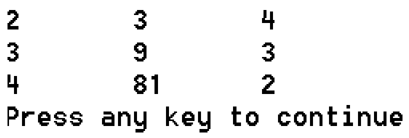
```
istream&    operator>> ( istream& in , Integer& I)    {
        in >> I.value;  return in;
}
```

**2. (10 points)**

According to the main function and the output below, implement a class hierarchy with ***Sequence*** as the base class with a method *print* which output the value of a data member named *number*. Derived classes are ***Increment***, ***Power***, and ***Decrement***.

```
int main() {
        Sequence *spi = new Increment(2);
        Sequence *spp = new Power(3);
        Sequence *spd = new Decrement(4);
        for(int i = 0; i < 3; i++) {
                spi->print();
                spp->print();
                spd->print();
                cout<<endl;
        }
        return 0;
}
```

*Output:*



```
2       3       4
3       9       3
4       81      2
Press any key to continue
```