2002 级 C++面向对象程序设计试题（A 卷）

（满分：100 分）

一、单项选择（每题 1 分，共 10 分）

1. 下面列出的基类中的哪部分能被派生类自动继承：
   A. 基类中的构造函数　　B. 基类中的虚析构函数
   C. 基类中重载的赋值操作　　D. 基类中的私有成员
2. 对于全局函数 int f(void) ，与其等价的函数原型为：
   A. int& f( );　　B. int f( ) const;
   C. int* f( );　　D. const int f( );
3. 类 A 中有唯一的一个成员函数 f，且 f 是公有的静态或非静态成员函数，对于类 A 的一个对象 a,执行语句 a.f(100);成功，那么 f 的函数原型不可以是：
   A. A& f(int, int=50);　　B. void f(int&) ;
   C. const A * f(const int);　　D. A f(const int&);
4. 关于异常和 C++提供的异常处理机制不正确的说法是：
   A. 能够改变程序的执行顺序　　B. 异常可以是对象
   C. 用户不能自定义异常类型　　D. 异常可以用 catch 进行捕捉处理
5. 在不考虑强制类型转换的情况下，关于类中常量成员函数的下列说法不正确的是：
   A. 常量成员函数中不能修改本类中的非静态数据成员。
   B. 常量成员函数中可以调用本类中的任何静态成员函数。
   C. 常量成员函数的返回值只能是 void。
   D. 若常量成员函数中调用虚函数 f，那么函数 f 在本类中也一定是一个常量成员函数。
6. 任意一个类，析构函数的个数最多是：
   A. 不限个数　　B. 1
   C. 2　　D. 3
7. 在 C++程序中，对象之间的相互通信可以通过：
   A. 继承实现　　B. 调用成员函数实现
   C. 封装实现　　D. 函数重载实现
8. 下面模板定义中不正确的是：
   A. template<class Q> Q F(Q x) { return Q + x; }
   B. template<class Q> Q F(Q x) { return x + x; }
   C. template<class T> T F(T x) { return x * x; }
   D. template<class T> T F(T x) { return x > 1; }
9. 对类型转换函数说明正确的是：
   A．转换函数不能被派生类继承
   B．一个类中只能有一个类型转换函数，以免错误调用
   C．类型转换函数不能带参数，但可以指定返回类型
   D．转换函数能对所属类中对象进行类型转换
10. 下面关于类的成员函数描述不正确的是：
   A. 静态成员函数内可以直接访问类的非静态成员数据
   B. 静态成员函数内可以直接访问类的静态成员数据
   C. 非静态成员函数可以直接访问类的非静态成员数据
   D. 非静态成员函数可以直接访问类的静态成员数据

二、判断正误，对于你认为错误的论述，说明原因或举出反例。（每题 2 分，共 20 分）

1．在 public 继承方式下，基类中被说明为 protected 和 private 的成员只能被其派生类的成员函数访问，不能被其它的函数访问。

2．如果派生类的成员函数的原型与基类中被定义为虚函数的成员函数原型相同，那么，这个函数自动带有虚函数的特性。

3．一个类中的成员函数重载（overload）可以用 const 修饰符、函数原型的参数个数、对应位置的参数的类型和返回值类型为依据。

4．无法产生对象的类一定是抽象类。

5．在任何形式的继承方式（public、protected 和 private）下，基类类型的指针都可以指向其派生

类对象。

6．若在类 Base 和派生类 Derived 中，分别声明一个同名的私有整型成员变量 int x; ，那么通过 Derived 类的实例对象，无论如何也访问不到基类 Base 中的成员变量 x 的值。

7．在静态成员函数的实现代码体内不能使用 this 指针，在非静态的成员函数实现代码体内可以使用 this 指针。

8．在继承过程中，带有虚函数的类作为基类时，该类称作虚基类。

9．在不考虑出现运行时异常的情况下，若一个程序的 main 函数代码体是空，则这个程序不会有输入，也不会产生任何输出结果。

10．　构造函数的初始化列表中的内容，不会影响构造对象中成员数据的初始化顺序。

三、回答下列各题（共 20 分）

1．说明类中的 public、 protected 和 private 成员函数分别在 public、protected 和 private 继承方式下的在派生类中的可访问性。（4 分）

2．类的哪些成员在缺省时可以由编译器自动给出？（4 分）

3．如何判断一个类 A 的两个对象 a1 和 a2 是否是同一个对象？（3 分）

4．针对类 A 和类 B，如何设计使得它们的对象可以直接或间接地存放在同一个数组中？（3 分）

5．对运算符进行重载时，有的运算符只能重载为类的成员函数，有的只能重载为全局函数，还有的两者均可以，针对上三种情况各举一例。（3 分）

6．在哪些情况下实现一个类的构造函数时，必须使用初始化列表？至少给出三种不同情况。（3 分）

四、阅读下面两个类的定义和部分实现代码，完成 4 个问题。（每题 3 分，共 12 分）

```
class Base {
public:
    Base(int value=0):x(value) {}
    virtual ~Base() {}
    Base(const Base& rhs):x(rhs.x)
{}
    Base& operator =(const Base&
rhs)
    {x=rhs.x; return *this;}
private:
    int x;
};
```

```
class Derived:public Base {
public:
    Derived(int v):Base(v),y(v) { }
    virtual ~Derived() {}
    Derived(const Derived& rhs);
    Derived& operator=(const Derived&
rhs);
private:
    int y;
};
```

（1） 实现类 Derived 的赋值运算 Derived& operator=(const Derived&)

（2） 实现类 Derived 的拷贝构造函数 Derived(const Derived&)

（3） 拷贝构造函数原型的参数类型说明中，const 修饰符的作用是什么？为什么使用引用的形式？

（4） 通常在什么情况下，一个类需要自定义并实现拷贝构造函数和赋值运算？

五、写出下面程序的运行结果（5 分）

```
#include <iostream.h>
class A {
public:
    A( )  { cout << "A" << endl; }
    ~A( ) { cout << "~A" << endl; }
};
```

```
class B  {
public:
    B( )     { f( );  cout << "B" <<
endl;}
    virtual void f( )  { cout << "B::f"
<< endl; }
    virtual ~B( )    { cout << "~B" <<
endl; }
};
```

```
Class D: public B {
public:
   D( )       {  cout  <<  "D"  <<
endl;  }
   void f( )  { cout << "D::f" <<
endl;  }
   virtual ~D( ) { cout << "~D" <<
endl; }
private:
   A  a;
};
```

```
void main( )
{
   B* pB = new D;
   pB->f( );
   delete pB;
}
```

六、写出下面程序的运行结果（5 分）

```
Class Myclass
{
   public:
      Myclass(int a,int b,int c);
      void PrintNumber( );
      void PrintSum( );
   private:
      int A,B,C;
      static int Sum;
};
```

```
void main( )
{
   Myclass  M(3,7,10),  N(14,9,1);
   M.PrintNumber( );
N.PrintNumber( );
   M.PrintSum(                    );
N.PrintSum( );
}
```

```
#include <iostream.h>
int Myclass::Sum=10;
 Myclass::Myclass(int a,int b,int c)
 {
   A=a; B=b; C=c;
   Sum+=A+B+C;
 }
 void Myclass::PrintNumber()
 {
   cout <<"Number="<<A
      <<","<<B<<","<<C<<endl;
 }
 void Myclass::PrintSum( )
 {
   cout<<"Sum="<<Sum<<endl;
 }
```

七、下面是对类 A、类 B 和类 C 的部分定义和函数 add 的完整定义及实现。其中，void add(A& obj1,A& obj2)函数实现体中的语句 obj1+obj2 的含义为：obj1 和 obj2 中对应的同名成员变量 va、vb、vc 成员分别相加，若对象中无相应的成员变量,则按 0 处理,即对于 add(a1,a2)含义为 a1.va=a1.va+a2.va, a1.vb=a1.vb+a2.vb, a1.vc=a1.vc+a2.vc，若其中的某数据分量不存在则按 0 计算。为保证 add 函数功能，必须对这三个类进行必要的补充，请完整实现（包括定义声明和实现）它们。（10 分）

```
class A{
private:
   int va;
};
```

```
class   B:public
A{
private:
   int vb;
};
```

```
class   C:public
B{
private:
   int vc;
};
```

```
void  add(A&  obj1,A&
obj2)
{
  obj1=obj1+ obj2;
}
```

八、下面是一段完整的程序，其中类 D 由类 B1 和类 B2 经过多重继承派生，请你只改变类 D 的定义和实现，使其去掉多重继承，其它代码不作任何变化，并保持程序原有功能不变。（8 分）

```
class B1 {
public:
   virtual void f( )
const{ }
};
```

```
class B2 {
public:
   virtual void
g( )const { }
};
```

```
class D : public B1,public
B2 {
public:
   void func( ) const{ f
( ); g( ); }
};
```

```
void F(const B1& aB1) {
   aB1.f( );
}
```

```
void FF(const B2& aB2)
{
   aB2.g( );
}
```

```
void main( )  {
   D d;   d.func( );
   F(d);  FF(d);
}
```

九、类 L 的对象 objL 和类 R 的对象 objR 构成一个二元组(objL,objR)，把其中的一个对象称作左元，如 objL；另一个称作右元，如 objR。一个简单字典(Dictionary)由若干个这样的二元组按一定条件构成，其条件是：所有二元组的左元类型都是 L，所有二元组的右元类型都是 R；任何两个二元组的左元的值(对象内容)不能相同，右元的值可以相同；一个字典具有的基本操作是：在满足前述条件约束下，可将任何两个对象构成的二元组加入字典中；任给一对象，可知它是否作为左元出现在字典中，若在，可返回与其对应的右元。现请你定义并实现这样的字典类，在

main 函数中给出使用这个字典的例子代码。不用考虑模板，允许将二元组中的元素设计成指针或引用形式 (10 分)

（全卷完毕）

2002 级 C++面向对象程序设计试题（A 卷）

（满分：100 分）

一、单项选择（每题 1 分，共 10 分）

1． D

2． D

3． B

4． C

5． C

6． B

7． B

8． A

9． A 答案可能有误，我认为应该是 D

10． A

二、

11． 错误， private 的成员不能被其派生类的成员函数访问。

12． 正确。

13． 错误，返回值类型不作为依据。

14． 错误，构造函数在私有或保护权限修饰下时是反例。

15． 错误， public 继承方式下的基类类型的指针都可以指向其派生类对象。

16． 错误，可通过 Base 类中访问其 x 的方法见解访问到。

17． 正确。

18． 错误，继承方式中用 virtual 修饰的是。

19． 错误，静态说明的对象初始化时，其构造函数的执行可能有结果输出。

20．正确，顺序与列表顺序无关。

三、回答下列各题（共 20 分）

1.

继承方式\访问权限	Public	Protected	private
Public	可访问，相当于 public	可访问，相当于 protected	不可访问
protected	可访问，相当于 protected	可访问，相当于 protected	不可访问
private	可访问，相当于 private	可访问，相当于 private	不可访问

2．不带参数的构造函数、拷贝构造函数、析构函数、赋值运算，还可以有相关的 new、delete 运算等。

3．a1 与 a2 的地址值是否相等。

4．让 A 与 B 均从同一个类 X 派生，元素类型是 X*或 X&的树组可存放 A 及 B 的对象地址或引用。

5．只能为成员函数的，如：赋值运算；只能为全局函数的，如：插入运算符<<；均可的，如：+运算。

6．（1）基类中无不带参数的构造函数；（2）类中有常量成员；（3）类中有引用成员。

四、

（5）
```
Derived& operator=(const Derived& aD)
{
    Base::operator=(aD);
    y = aD.y;
    return *this;
}
```

（6）
```
Derived(const Derived& aD)
{
   operator=(aD);
```

```
    }
```

（7） const 修饰符的作用是使实际参数既可以是变量，又可以是常量。使用引用的目的是避免拷贝，避免无限递归循环调用。

（8） 有指针或引用类型的数据成员时。

五、

```
B::f
B
A
D
D::f
~D
~A
~B
```

六、

```
Number=3,7,10
Number=14,9,1
Sum=54
Sum=54
```

七、考查以虚函数形式定义的运算符重载，各类中实现相似，略。

```
class A {
private:
   int va;
public:
   A(int n):va(n) {}
   virtual int get_va(){return va;}
   virtual int get_vb(){return 0;}
   virtual int get_vc(){return 0;}
   virtual A& operator+(A& obj){va+=obj.get_va();return *this;}
   void printf(){cout << get_va() <<','<<get_vb()<<','<<get_vc()<<endl;}
};
class B: public A {
private:
   int vb;
public:
   B(int n1,int n2):A(n1),vb(n2){}
   virtual int get_vb(){return vb;}
   virtual A& operator+(A& obj){
      A::operator+(obj);
   vb+=obj.get_vb();return *this;}
};
class C: public B {
private:
   int vc;
public:
   C(int n1,int n2,int n3):B(n1,n2),vc(n3){}
   virtual int get_vc(){return vc;}
   virtual A& operator+(A& obj){B::operator+(obj);vc+=obj.get_vc();return *this;}
};
```

八、

```
class D: public B1
    { public:
        D(const &B2 aB2): m_B2(aB2);
        void func( ) const { f( ); m_b2.g( ); }
     private:
        const B2& m_B2;
  };
```

九、

```
class Pair
    { public:
        Pair(L*, R*);
```

```
    R* getR( ) { reuturn aR; }
    L* getL( ) { return aL;}
  private:
    R* aR;
    L* aL;
};
class Dictionary
{
  public:
   Dictionary( ) { lines=0; }
   void addARow(Pair * aPair)
   { //先查是否有重复的，代码类似下面 getRBy,代码略
     Rows[lines++] = aPair
   }
   R* getRBy(L* aL)
   { R* pR;
     int found = 0;
     int i = 0;
     while (!found)
     { if  ( i >= line)
         break;
       if (rows[i]->getL( ) == aL)
       //或者 if(rows[i]->getL( )->getVal( ) == aL->getVal( )
          {  found = 1; pR = new R(*(rows[i]->getR( ))); }
        else
           i++;
     }
    if (found)
        return pR;
    else
         return 0;
   }
 L* getLBy(R* aR){ 代码与上类似，略}
private:
   Pair* Rows[200];
   int lines;
 };
```