机密★启用前

2006年9月全国计算机等级考试二级笔试试卷

C++语言程序设计

61

注意事项

一、考生应严格遵守考场规则，得到监考人员指令后方可作答。

二、考生拿到试卷后应首先将自己的姓名、准考证号等内容涂写在答题卡的相应位置上。

三、选择题答案必须用铅笔填涂在答题卡的相应位置上，填空题的答案必须用蓝、黑色钢笔或圆珠笔写在答题卡的相应位置上，答案写在试卷上无效。

四、注意字迹清楚，保持卷面整洁。

五、考试结束将试卷和答题卡放在桌上，不得带走。待监考人员收毕清点后，方可离场。

教育部考试中心

二00六年七月制

全国计算机等级考试二级 C++语言程序设计

2006 年 9 月笔试试卷

（考试时间 90 分钟，满分 100 分）

一、选择题（每小题2分，共70分）

（1）下列选项中不符合良好程序设计风格的是（　　）。

A）源程序要文档化　　B）数据说明的次序要规范化

C）避免滥用 goto 语句　　D）模块设计要保证高耦合、高内聚

（2）从工程管理角度看，软件设计一般分为两步完成，它们是（　　）。

A）概要设计与详细设计　　B）数据设计与接口设计

C）软件结构设计与数据设计　　D）过程设计与数据设计

（3）下列选项中不属于软件生命周期开发阶段任务的是（　　）。

A）软件测试　　B）概要设计

C）软件维护　　D）详细设计

（4）在数据库系统中，用户所见的数据模式为（　　）。

A）概念模式　　B）外模式

C）内模式　　D）物理模式

（5）数据库设计的四个阶段是：需求分析、概念设计、逻辑设计和（　　）。

A）编码设计　　B）测试阶段

C）运行阶段　　D）物理设计

（6）设有如下三个关系表：

R

A
m
n

S

B	C
1	3

T

A	B	C
m	1	3
n	1	3

下列操作中正确的是（　　）。

A）T=R∩S　　B）T=R∪S

C）T=R×S　　D）T=R/S

（7）下列描述中正确的是（　　）。

A）一个算法的空间复杂度大，则其时间复杂度也必定大

B）一个算法的空间复杂度大，则其时间复杂度必定小

C）一个算法的时间复杂度大，则其空间复杂度必定小

D）上述三种说法都不对

（8）在长度为 64 的有序线性表中进行顺序查找，最坏情况下需要比较的次数为（　　）。

A）63　　B）64

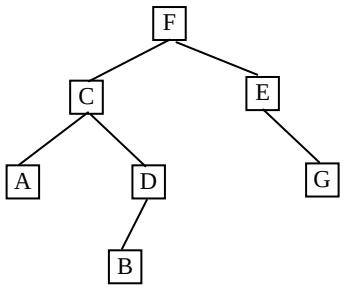
C）6　　D）7

（9）数据库技术的根本目标是要解决数据的（　　）。

A）存储问题　　B）共享问题

C）安全问题　　D）保护问题

（10）对下列二叉树进行中序遍历的结果是（　　）。



A）ACBDFEG　　B）ACBDFGE

C）ABDCGEF　　D）FCADBEG

（11）下列有关内联函数的叙述中，正确的是

A）内联函数在调用时发生控制转移　　B）内联函数必须通过关键字 inline 来定义

C）内联函数是通过编译器来实现的　　D）内联函数函数体的最后一条语句必须是 return 语句

（12）下列情况中，不会调用拷贝构造函数的是

A）用一个对象去初始化同一类的另一个新对象时

B）将类的一个对象赋值给该类的另一个对象时

C）函数的形参是类的对象，调用函数进行形参和实参结合时

D）函数的返回值是类的对象，函数执行返回调用时

（13）下列有关继承和派生的叙述中，正确的是

A）如果一个派生类私有继承其基类，则该派生类对象不能访问基类的保护成员

B）派生类的成员函数可以访问基类的所有成员

C）基类对象可以赋值给派生类对象

D）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

（14）下列运算符不能重载为友元函数的是

A）= () [ ] ->　　B）+ - ++ --

C）> < >= <=　　D）+= -= *= /=

（15）关于在调用模板函数时模板实参的使用，下列表述正确的是

A）对于虚拟类型参数所对应的模板实参，如果能从模板函数的实参中获得相同的信息，则都可以省略

B）对于虚拟类型参数所对应的模板实参，如果它们是参数表中的最后的若干个参数，则都可以省略

C）对于虚拟类型参数所对应的模板实参，若能够省略则必须省略

D）对于常规参数所对应的模板实参，任何情况下都不能省略

（16）下列关于输入流类成员函数 getline()的描述中，错误的是

A）该函数是用来读取键盘输入的字符串的　　B）该函数读取的字符串长度是受限制的

C）该函数读取字符串时，遇到终止符便停止　　D）该函数读取字符串时，可以包含空格

（17）下列符号中，正确的 C++标识符是

A）enum　　B）2b

C）foo-9　　D）_32

（18）下列语句中，错误的是

A）const int buffer=256;　　B）const double *point;

C）int const buffer=256;　　D）double * const point;

（19）if 语句的语法格式可描述为：

格式 1：if(<条件>) <语句>

或

格式 2：if(<条件>) <语句 1> else <语句 2>

关于上面的语法格式，下列表述中错误的是

A）<条件>部分可以是一个 if 语句，例如 if(if(a==0) …) …

B）<语句>部分可以是一个 if 语句，例如 if(…) if(…) …

C）如果在<条件>前加上逻辑非运算符!并交换 <语句 1> 和 <语句 2> 的位置，语句功能不变

D）<语句>部分可以是一个循环语句，例如 if(…) while(…) …

（20）有如下说明

int a[10]={1,2,3,4,5,6,7,8,9,10},*p=a;

则数值为 9 的表达式是

A）*p+9　　B）*(p+8)

C）*p+=9　　D）p+8

（21）若有下面的函数调用：

fun(a+b, 3, max(n-1, b))

则 fun 的实参个数是

A）3　　B）4

C）5　　D）6

（22）以下关键字不能用来声明类的访问权限的是

A）public　　B）static

C）protected　　D）private

（23）在公有继承的情况下，允许派生类直接访问的基类成员包括

A）公有成员　　B）公有成员和保护成员

C）公有成员、保护成员和私有成员　　D）保护成员

（24）关于运算符重载，下列表述中正确的是

A）C++已有的任何运算符都可以重载

B）运算符函数的返回类型不能声明为基本数据类型

C）在类型转换符函数的定义中不需要声明返回类型

D）可以通过运算符重载来创建 C++中原来没有的运算符

（25）关于关键字 class 和 typename，下列表述中正确的是

A）程序中的 typename 都可以替换为 class
B）程序中的 class 都可以替换为 typename
C）在模板形参表中只能用 typename 来声明参数的类型
D）在模板形参表中只能用 class 或 typename 来声明参数的类型

（26）有如下程序

```
#include <iostream>
#include <iomanip>
using namespace std;
int main( ){
    cout<<setprecision(3)<<fixed<<setfill('*')<<setw(8);
    cout<<12.345<<_____<<34.567;
    return 0;
}
```

若程序的输出是：
12.34534.567
则程序中下划线处遗漏的操作符是
A）setprecision(3)　　B）fixed
C）setfill('*')　　D）setw(8)

（27）有如下程序

```
#include <iostream>
#include <iomanip>
using namespace std;
class MyClass{
public:
    MyClass( ){ cout<<'A'; }
     MyClass(char c){ cout<<c; }
    ~MyClass( ){ cout<<'B'; }
};
int main( ){
    MyClass p1,*p2;
    p2=new MyClass('X');
      delete p2;
      return 0;
}
```

执行这个程序屏幕上将显示输出
A）ABX　　B）ABXB
C）AXB　　D）AXBB

（28）有如下程序

```
#include <iostream>
using namespace std;
int i=1;
class Fun{
public:
    static int i;
    int value( ){ return i-1;}
    int value( )const{ return i+1;}
};
int Fun::i=2;
int main( ){
    int i=3;
    Fun fun1;
    const Fun fun2;
    _____
    return 0;
}
```

若程序的输出结果是：
123
则程序中下划线处遗漏的语句是
A）cout<<fun1.value( )<<Fun::i<<fun2.value( );
B）cout<<Fun::i<<fun1.value()<<fun2.value( );
C）cout<<fun1.value( )<<fun2.value( )<<Fun::i;
D）cout<<fun2.value( )<<Fun::i<<fun1.value( );

（29）有如下程序：

```
#include <iostream>
using namespace std;
class Obj{
    static int i;
public:
    Obj( ){ i++; }
    ~Obj( ){ i--; }
     static int getVal( ){ return i;}
};
int Obj::i=0;
void f(){Obj ob2; cout<<ob2.getVal( ); }
int main( ){
     Obj ob1;
     f( );
     Obj *ob3=new Obj; cout<<ob3->getVal( );
     delete ob3; cout<<Obj::getVal( );
     return 0;
}
```

程序的输出结果是

A）232　　B）231

C）222　　D）221

（30）有如下程序

```
#include <iostream>
using namespace std;
class Base {
protected:
     Base( ){ cout<<'A'; }
     Base(char c){ cout<<c; }
};
class Derived: public Base{
public:
     Derived( char c ){ cout<<c; }
};
int main( ){
     Derived d1('B');
     return 0;
}
```

执行这个程序屏幕上将显示输出

A）B　　B）BA

C）AB　　D）BB

（31）有如下类定义：

```
class MyBase{
    int k;
public:
    MyBase(int n=0):k(n){ }
    int value( )const{ return k;}
};
class MyDerived: MyBase{
    int j;
public:
    MyDerived(int i): j(i) {}
    int getK( )const{ return k; }
    int getJ( )const{ return j; }
};
```

编译时发现有一处语法错误，对这个错误最准确的描述是

A）函数 getK 试图访问基类的私有成员变量 k

B）在类 MyDerived 的定义中，基类名 MyBase 前缺少关键字 public、protected 或 private

C）类 MyDerived 缺少一个无参的构造函数

D）类 MyDerived 的构造函数没有对基类数据成员 k 进行初始化

（32）在一个派生类对象结束其生命周期时

A）先调用派生类的析构函数后调用基类的析构函数

B）先调用基类的析构函数后调用派生类的析构函数

C）如果基类没有定义析构函数，则只调用派生类的析构函数

D）如果派生类没有定义析构函数，则只调用基类的析构函数

（33）有如下的运算符重载函数定义：

```
double operator +(int i, int k){ return double(i+k); }
```

但定义有错误，对这个错误最准确的描述是

A）+ 只能作为成员函数重载，而这里的 + 是作为非成员函数重载的

B）两个 int 型参数的和也应该是 int 型，而这里将 + 的返回类型声明为 double

C）没有将运算符重载函数声明为某个类的友元

D）C++已经提供了求两个 int 型数据之和的运算符 +，不能再定义同样的运算符

（34）语句 ofstream f("SALARY.DAT", ios_base::app);的功能是建立流对象 f，并试图打开文件 SALARY.DAT 与 f 关联，而且

A）若文件存在，将其置为空文件；若文件不存在，打开失败

B）若文件存在，将文件指针定位于文件尾；若文件不存在，建立一个新文件

C）若文件存在，将文件指针定位于文件首；若文件不存在，打开失败

D）若文件存在，打开失败；若文件不存在，建立一个新文件

（35）有如下程序

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

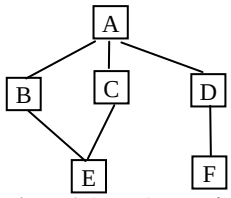
运行此程序，屏幕上将显示输出

A）B1B2　　B）A1A2

C）B1A2　　D）A1B2

二、填空题（每空2分，共30分）

（1）下列软件系统结构图的宽度为_____。



（2）_____的任务是诊断和改正程序中的错误。

（3）一个关系表的行称为_____。

（4）按“先进后出”原则组织数据的数据结构是_____。

（5）数据结构分为线性结构和非线性结构，带链的队列属于_____。

（6）若有定义语句：int* a, b;，则变量 b 的数据类型是_____。

（7）已知数组 a 中的元素个数为 n，下列语句的作用是将下标为 i 的元素移动到下标为 i－1 的单元，其中 1≤i<n。例如，当 n＝4，a 中原有数据为 1，2，3，4 时，则移动后 a 中元素变为 2，3，4，4。请将语句补充完整：

```
for (int i=0; i<n-1; i++)    a[i] = a[_____ ];
```

（8）已知递归函数 f 的定义如下：

```
int f(int n)
{
    if (n <= 1) return 1;    //递归结束情况
    else return n * f(n-2); //递归}
```

则函数调用语句 f(5)的返回值是_____。

（9）创建对象数组时，对数组的每一个元素都将调用一次构造函数，如果没有显式给出数组元素的初值，则调用缺省构造函数。下列程序涉及到对象数组的创建和单个对象的创建，其输出结果是_____。

```
#include <iostream>
```

```
using namespace std;
class Foo {
  public:
    Foo(int x) { cout << 'A'; }
    Foo() {}
};
int main()
{
    Foo f[3], g(3);
    return 0;
}
```

（10）已知下列程序的输出结果是 42，请将画线处缺失的部分补充完整。

```
#include <iostream>
using namespace std;
class Foo {
      int value;
  public:
    Foo( ) : value(0) {}
    void setValue(int value)
    { _____ = value;      //给 Foo 的数据成员 value 赋值}
    void print( ) { cout << value; }
};
int main( )
{
    Foo f;
    f.setValue(42);
    f.print();
    return 0;
}
```

（11）如果不使用多态机制，那么通过基类的指针虽然可以指向派生类对象，但是只能访问从基类继承的成员。下列程序没有使用多态机制，其输出结果是_____。

```
#include <iostream>
using namespace std;
class Base {
  public:
    void print ( ) { cout << 'B'; }};
  class Derived : public Base {
  public:
    void print ( ) { cout << 'D'; }};
int main( )
{
    Derived* pd = new Derived( );
    Base* pb = pd;
    pb->print( );
    pd->print( );
    delete pd;
    return 0;
}
```

（12）在声明派生类时，如果不显式地给出继承方式，缺省的类继承方式是私有继承 private。已知有如下类定义：

```
class Base {
  protected:
    void fun( ) {}
};
class Derived : Base { };
```

则 Base 类中的成员函数 fun()，在 Derived 类中的访问权限是_____（注意：要求填写 private、protected 或 public 中的一项）。

（13）在 MyClass 类的定义中，对赋值运算符=进行重载。请将画线处缺失的部分补充完整。

```
_____ MyClass::operator=(const MyClass& rhs)
{
    if (this == &rhs) return *this;
      value = rhs.value;
      return *this;
}
```

（14）插入排序算法的主要思想是：每次从未排序序列中取出一个数据，插入到已排序序列中的正确位置。下面类的成员函数 sort()实现了插入排序算法。请将画线处缺失的部分补充完整。

```
class InsertSort{
public:
InsertSort(int* a0, int n0) :a(a0), n(n0) {} //参数 a0 是某数组首地址，n 是数组元素个数
void sort( )
{//此函数假设已排序序列初始化状态只包含 a[0]，未排序序列初始为 a
[1]...a[n-1]
      for (int i=1; i<n; ++i){
         int t=a[i];
         int j;
         for (_____; j>0; --j){
            if (t>=a[j-1]) break;
            a[j]=a[j-1];}
         a[j]=t;}}
protected:
    int *a, n; //指针 a 用于存放数组首地址，n 用于存放数组元素个数
};
```

（15）下列程序的输出结果是_____。

```
#include <iostream>
using namespace std;
class A {
      int a;
   public:
      A():a(9){}
      virtual void print() const { cout<<a;};
};
class B : public A {
      char b;
   public:
      B( ){b='S';}
void print( ) const { cout <<b;}
};
void show(A &x){ x.print();}
int main()
{     A d1,*p;
      B d2;
      p=&d2;
      d1.print();
      d2.print();
      p->print();
      show(d1);
      show(d2);
      return 0;}
```