



R		
A	B	C
a	1	2
b	2	1
c	3	1

S		
A	B	C
d	3	2

T		
A	B	C
a	1	2
b	2	1
c	3	1
d	3	2

其中关系 T 由关系 R 和 S 通过某种操作得到，该操作为 ( )。

- A) 选择                      B) 投影                      C) 交                      D) 并

(11) 已知函数 FA 调用 FB，若要把这两个函数定义在同一个文件中，则 ( )。

- A) FA 必须定义在 FB 之前  
B) FB 必须定义在 FA 之前  
C) 若 FA 定义在 FB 之后，则 FA 的原型必须出现在 FB 的定义之前  
D) 若 FB 定义在 FA 之后，则 FB 的原型必须出现在 FA 的定义之前

(12) 有如下两个类定义：

```
class AA{};
class BB {
    AA v1,*v2;
    BB v3;
    int *v4;
};
```

其中有一个成员变量的定义是错误的，这个变量是 ( )。

- A) v1                      B) v2                      C) v3                      D) v4

(13) 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){}
};
```

YY 类的对象包含的数据成员的个数是 ( )。

- A) 1                      B) 2                      C) 3                      D) 4

(14) 下列关于运算符函数的描述中，错误的是 ( )。

- A) 运算符函数的名称总是以 operator 为前缀  
B) 运算符函数的参数可以是对象  
C) 运算符函数只能定义为类的成员函数  
D) 在表达式中使用重载的运算符相当于调用运算符重载函数

(15) 下列关于模板形参的描述中，错误的是 ( )。

- A) 模板形参表必须在关键字 template 之后  
B) 模板形参表必须用括弧()括起来  
C) 可以用 class 修饰模板形参  
D) 可以用 typename 修饰模板形参

(16) 在下列枚举符号中，用来表示“相对于当前位置”文件定位方式的是 ( )。

- A) ios\_base::cur B) ios\_base::beg  
C) ios\_base::out D) ios\_base::end
- (17) 下列字符串中可以用作 C++ 标识符的是 ( )。
- A) 2009var B) goto  
C) test-2009 D) \_123
- (18) 下列枚举类型的定义中, 包含枚举值 3 的是 ( )。
- A) enum test {RED, YELLOW, BLUE, BLACK};  
B) enum test {RED, YELLOW=4, BLUE, BLACK};  
C) enum test {RED=-1, YELLOW, BLUE, BLACK};  
D) enum test {RED, YELLOW=6, BLUE, BLACK};
- (19) 有如下程序段:
- ```
int i=1;
while(1) {
    i++;
    if(i==10) break;
    if(i%2==0) cout<<"*";
}
```
- 执行这个程序段输出字符\*的个数是 ( )。
- A) 10 B) 3 C) 4 D) 5
- (20) 已知数组 arr 的定义如下:
- ```
int arr[5]={1,2,3,4,5};
```
- 下列语句中输出结果不是 2 的是 ( )。
- A) cout << \*arr+1 << endl;  
B) cout << \* (arr+1) << endl;  
C) cout << arr[1] << endl;  
D) cout << \*art << endl;
- (21) 计算斐波那契数列第 n 项的函数定义如下:
- ```
int fib(int n){
    if(n==0) return 1;
    else if(n==1) return 2;
    else return fib(n-1)+fib(n-2);
}
```
- 若执行函数调用表达式 fib(2), 函数 fib 被调用的次数是 ( )。
- A) 1 B) 2 C) 3 D) 4
- (22) Sample 是一个类, 执行下面语句后, 调用 Sample 类的构造函数的次数是 ( )。
- ```
Sample a[2], *P = new Sample;
```
- A) 0 B) 1 C) 2 D) 3
- (23) 下列关于虚基类的描述中, 错误的是 ( )。
- A) 使用虚基类可以消除由多继承产生的二义性  
B) 构造派生类对象时, 虚基类的构造函数只被调用一次  
C) 声明 “class B : virtual public A” 说明类 B 为虚基类  
D) 建立派生类对象时, 首先调用虚基类的构造函数
- (24) 将运算符重载为类成员函数时, 其参数表中没有参数, 说明该运算符是 ( )。
- A) 不合法的运算符 B) 一元运算符  
C) 无操作数的运算符 D) 二元运算符
- (25) 有如下模板声明:
- ```
template < typename T1, typename T2> class A;
```

下列声明中，与上述声明不等价的是（ ）。

- A) template <class T1, class T2> class A;
- B) template <class T1, typename T2> class A;
- C) template <typename T1, class T2> class A;
- D) template <typename T1, T2> class A;

(26) 下列关于 C++流的描述中，错误的是（ ）。

- A) cout<<'A'表达式可输出字符 A
- B) eof()函数可以检测是否到达文件尾
- C) 对磁盘文件进行流操作时，必须包含头文件 fstream
- D) 以 ios\_base::out 模式打开的文件不存在时，将自动建立一个新文件

(27) 有如下程序：

```
# include <iostream>
using namespace std;
class Toy {
public:
    Toy(char* _n){ strcpy(name, _n); count++;}
    ~Toy() { count--;}
    char* GetName(){ return name; }
    static int getCount(){ return count; }
private:
    char name[10];
    static int count;
};
int Toy::count=0;
int main(){
    Toy t1("Snoopy"), t2("Mickey"), t3("Barbie");
    cout<<t1.getCount()<<endl;
    return 0;
}
```

运行时的输出结果是（ ）。

- A) 1
- B) 2
- C) 3
- D) 运行时出错

(28) 有如下程序：

```
# include <iostream>
using namespace std;
class A {
public:
    A(int i) : r1(i) {}
    void print() {cout<<'E'<<r1<<'-';}
    void print() const {cout<<'C'<<r1*r1<<'-';}
private:
    int r1;
};
int main(){
    A a1(2);    const A a2(4);
    a1.print(); a2.print();
    return 0;
}
```

```
}
```

运行时的输出结果是 ( )。

- A) 运行时出错      B) E2-C16-      C) C4-C16-      D) E2-E4-

(29) 有如下程序:

```
#include <iostream>
using namespace std;
class Name{
    char name[20];
public:
    Name(){
        strcpy(name, "");      cout<<'?';
    }
    Name(char *fname){
        strcpy(name,fname);    cout<<'?';
    }
};
int main(){
    Name names[3]={Name("张三"), Name("李四")};
    return 0;
}
```

运行此程序输出符号?的个数是 ( )。

- A) 0      B) 1      C) 2      D) 3

(30) 有如下程序:

```
#include <iostream>
using namespace std;
class AA{
public:
    AA(){ cout<<'1'; }
};
class BB: public AA{
    int k;
public:
    BB(): k(0) { cout<<'2'; }
    BB(int n): k(n) { cout<<'3'; }
};
int main(){
    BB b(4), c;
    return 0;
}
```

运行时的输出结果是 ( )。

- A) 1312      B) 132      C) 32      D) 1412

(31) 有如下程序:

```
#include <iostream>
using namespace std;
class C1{
public:
```

```

~C1(){ cout<<1; }
};
class C2: public C1{
public:
    ~C2(){ cout<<2; }
};
int main(){
    C2 cb2;
    C1 *cb1;
    return 0;
}

```

运行时的输出结果是 ( )。

- A) 121                      B) 21                      C) 211                      D) 12

(32) 有如下程序:

```

#include <iostream>
using namespace std;
class Publication{ //出版物类
    char name[30];
public:
    Publication(char *name="未知名称"){
        strcpy(this->name,name);
    }
    const char * getName()const{ return name; }
    virtual const char * getType()const{ return "未知类型";}
};
class Book: public Publication{ //书类
public:
    Book(char *name): Publication(name){ }
    virtual const char * getType()const{ return "书";}
};
void showPublication( Publication &p){
    cout<<p.getType()<<": "<<p.getName()<<endl;
}
int main(){
    Book book("精彩人生");
    showPublication(book);
    return 0;
}

```

运行时的输出结果是 ( )。

- A) 未知类型: 未知名称                      B) 未知类型: 精彩人生  
C) 书: 未知名称                      D) 书: 精彩人生

(33) 下列关于运算符重载的描述中, 错误的是 ( )。

- A) ::运算符不能重载  
B) 类型转换运算符只能作为成员函数重载  
C) 将运算符作为非成员函数重载时必须定义为友元  
D) 重载[]运算符应完成“下标访问”操作

(34) 有如下程序:

```
#include <iostream>
#include <iomanip>
using namespace std;
int main(){
    int s[]={ 123, 234};
    cout<<right<<setfill('*')<<setw(6);
    for(int i=0; i<2; i++){ cout<<s[i]<<endl; }
    return 0;
}
```

运行时的输出结果是 ( )。

- A) 123                      B) \*\*\*123                      C) \*\*\*123                      D) \*\*\*123  
       234                      234                      \*\*\*234                      234\*\*\*

(35) 有如下类定义:

```
class A {
    char *a;
public:
    A(): a(0){}
    A(char *aa){ //把 aa 所指字符串拷贝到 a 所指向的存储空间
        a=_____ ;
        strcpy(a,aa);
    }
    ~A() {delete []a;}
};
```

横线处应填写的表达式是 ( )。

- A) new char[strlen(aa)+1]                      B) char[strlen(aa)+1]  
 C) char[strlen(aa)]                      D) new char[sizeof(aa)-1]

## 二、填空题 (每空 2 分, 共 30 分)

- (1) 某二叉树有 5 个度为 2 的结点以及 3 个度为 1 的结点, 则该二叉树中共有 【1】 个结点。
- (2) 程序流程图中的菱形框表示的是 【2】。
- (3) 软件开发过程主要分为需求分析、设计、编码与测试四个阶段, 其中 【3】 阶段产生“软件需求规格说明书”。
- (4) 在数据库技术中, 实体集之间的联系可以是一对一或一对多或多对多的, 那么“学生”和“可选课程”的联系为 【4】。
- (5) 人员基本信息一般包括: 身份证号, 姓名, 性别, 年龄等。其中可以作为主关键字的是 【5】。
- (6) 若表达式  $(x+(y-z)*(m/n))+3$  中的变量均为 double 型, 则表达式值的类型为 【6】。
- (7) 有如下循环语句:

```
for(int i=50; i>20; i-=2) cout<<i<<',';
```

运行时循环体的执行次数是 【7】 次。

- (8) 利用表达式  $a[i]$  可以访问 int 型数组 a 中下标为 i 的元素。在执行了语句  $\text{int } *p=a;$  后, 利用指针 p 也可访问该元素, 相应的表达式是 【8】。
- (9) 下面是一个递归函数, 其功能是使数组中的元素反序排列。请将函数补充完整。

```
void reverse(int *a, int size){
    if(size<2) return;
    int k=a[0];
    a[0]=a[size-1];
```

```

    a[size-1]=k;
    reverse(a+1, 【9】);
}

```

- (10) 类 Sample 的构造函数将形参 data 赋值给数据成员 data。请将类定义补充完整。

```

class Sample{
public:
    Sample(int data=0);
private:
    int data;
};
Sample::Sample(int data){
    【10】
}

```

- (11) 有如下类定义:

```

class Sample{
public:
    Sample();
    ~Sample();
private:
    static int data;
};

```

将静态数据成员 data 初始化为 0 的语句是 **【11】**。

- (12) “图形”类 Shape 中定义了纯虚函数 CalArea(), “三角形”类 Triangle 继承了类 Shape, 请将 Triangle 类中的 CalArea 函数补充完整。

```

class Shape{
public:
    virtual int CalArea()=0;
};
class Triangle: public Shape{
public:
    Triangle(int s, int h) : side(s), height(h){ }
    【12】。 { return side * height/2;}
private:
    int side;
    int height;
};

```

- (13) 有如下程序:

```

#include <iostream>
using namespace std;
class GrandChild{
public:
    GrandChild(){ strcpy(name,"Unknown"); }
    const char * getName() const { return name;}
    virtual char * getAddress() const=0;
private:
    char name[20];
}

```



```

};
class GrandSon : public GrandChild{
public:
    GrandSon(char *name){}
    char * getAddress() const { return "Shanghai"; }
};
int main(){
    GrandChild *gs=new GrandSon ("Feifei");
    cout<<gs->getName()<<"住在"<<gs->getAddress()<<endl;
    delete gs;
    return 0;
}

```

运行时的输出结果是 **【13】**。

- (14) 如下程序定义了“单词”类 Word，类中重载了<运算符，用于比较“单词”的大小，返回相应的逻辑值。程序的输出结果为：After Sorting: Happy Welcome，请将程序补充完整。

```

#include <iostream>
#include <string>
using namespace std;
class Word{
public:
    Word(string s) : str(s){}
    string getStr(){ return str; }
    bool operator < (Word w) const {return(str<w.str);}
    friend ostream& operator<<(ostream& output, const Word &w)
        { output<<w.str; return output; }
private:
    string str;
};
int main(){
    Word w1("Happy"),w2("Welcome");
    cout<<"After sorting: ";
    if(w1<w2) cout<<w1<<' '<<w2;
    else cout<<w2<<' '<<w1;
    return 0;
}

```

- (15) 请将下列模板类 Data 补充完整。

```

template <typename T>
class Data{
public:
    void put(T v) { val=v; }
    【15】 get() //返回数据成员 val 的值，返回类型不加转换
        { return val; }
private:
    T val;
};

```