新标准 C++程序设计教材 11-20 章课后题答案

第 11 章：

1．简述结构化程序设计有什么不足，面向对象的程序如何改进这些不足。

答案：

结构化程序设计的缺点：

（1）用户要求难以在系统分析阶段准确定义，致使系统在交付使用时产生许多问题。

（2）用系统开发每个阶段的成果来进行控制，不适应事物变化的要求。

（3）系统的开发周期长。

面向对象的程序设计如何改进这些不足：

面向对象程序设计技术汲取了结构忧程序设计中好的思想，并将这些思想与一些新的、强大的理念相结台，从而蛤程序设计工作提供了一种全新的方法。通常，在面向对象的程序设计风格中，会将一个问题分解为一些相互关联的子集，每个子集部都包含了相关的数据和函数。同时会以某种方式将这些子集分为不同等级，而一个对象就是已定义的某个类型的变量。

2．以下说怯正确的是（ ）。

A. 每个对象部都有成员函数的实现代码

B. 一个类的私有成员函数部不能访问本类的私有成员变量

C. 类的成员函数之间可以互相调用

D. 编写一个类时，至少要编写一个成员函数

答案：C

3．以下对类 A的定义正确的是（ ）。

A ． class A{
private: int v;
public: void Func() {}
}

B. class A{
int v; A * next;
void Func() {}
};

C ． class A{
int v;
public :
void Func();
};
A: : void Func() { }

D. class A{
int v;
public:
A next ;
void Func() {}
};

答案：B

4．假设有以下类 A:

class A{
public:
int func(int a) { return a * a; }
};

以下程序段不正确的是（ ）。

A. A a; a.func(5);

B. A * p = new A; p->func(5);

C. A a； A&r =a ; r.func(5);

D. A a,b; if(a!=b) a.func(5);

答案：D

5．以下程序段不正确的是(A)。

A ． int main() {

class A { int v; }

A a; a.v= 3;　　return 0　;

B ． int main() {

class A { public: int v; A * p; };

A a;　a.p=&a;　return 0;

C ． int main() {

class A { public: int v; };

A * p = new A;

p-> v =4;　delete p;

return 0;

D. im main() {

class A { public:　int v; A * p; };

A a:　 a. p = new A;　delete a.p;

return 0;

答案：A

6．实现一个学生信息处理程序。输入:，年龄，学号（整数）。第一学年平均成绩·第二学年平均成绩，第三学年平均成绩，第四学年平均成绩。输出：,年龄,学号,4 年平均成绩。例如：

输入： Tom 18 7817 80 80 90 70

输出： Tom,18,7817,80

要现一个代表学生的类，并非所有成员变量都是私有的。

答案：

```
#include <iostream>
#include<cstring>
#include<cstdlib>
//#include<cstdio>
using namespace std;
class Student {
private:
    int age, score1, score2, score3, score4;
    char name[100], num[100];
    double average;
public:
    Student(char aname[], int aage, char anum[],int ascore1, int ascore2, int ascore3, int
ascore4) {
        strcpy(name, aname);
        age = aage;
        strcpy(num, anum);
        score1 = ascore1;
        score2 = ascore2;
        score3 = ascore3;
        score4 = ascore4;
```

```
    }
    double getAverage() {
        return (score1 + score2 + score3 + score4) / 4;
    }
    char * getName() {
        return name;
    }
    int getAge() {
        return age;
    }
    char * getNum() {
        return num;
    }
};
int main() {
    char name[100], a, num[100];
    int age, score1, score2, score3, score4;
    cin.getline(name, 100, ' ');
    cin >> age;
   // a = getchar();
    cin.getline(num, 100, ' ');
    cin >> score1 >> score2 >> score3  >> score4;
    Student s(name, age, num, score1, score2, score3, score4);

    return 0;
}
```

第 12 章

1．以下说法中正确的是(　)。

A. 一个类一定会有无参构造函数

B. 构造函数的返回值类型是 void

C. 一个类只能定义一个析构函数，但可以定义多个构造函数

D. 一个类只能定义一个构造函数，但可以定义多个析构函数

答案：C

2．对于强过 new运算符牛成的对象(　)。

A. 程程序完毕时自动析构

B. 执行 delete　操作时才能析构

C. 在包含 new语句的函数返回时自动析构

D. 在执行 delete　操作时会析构，如果没有执行 delete　操作，则在程序完毕时自动析构

答案：D

3．如果某函数的返回值是个对象，则该函数被调用时，返回的对象(　)。

A. 是通过复制构造函数初始化的

B. 是通过无参构造函数初始化的

C. 用哪个构造函数初始化取决于函数的 return　语句是怎么写的

D. 不需要初始化

答案：C

4．以下说法 LE确的是（ ）。

A. 在静态成员函数中可以调用同类的其他任何成员函数

B. const 成员函数不能作用于非 const 对象

C. 在静态成员函数中不能使用 this 指针

D. 静态成员变量每个对象有各自的一份

答案：C

5 以下关于 this 指针的说法中不正确的是（ ）。

A. const 成员函数部不可以使用 this 指针

B. 成员函数的 this 指针，指向成员函数所作用的对象

C. 在构造函数部可以使用 this 指针

D. 在析构函数部可以使用 this 指针

答案：A

6．请写出下面程序的输出结果。

```
class CSample {
        int x;
public:
        CSample() { cout<<”C1”<<endl; }
        CSample(int n) {
                x=n;
                cout<<”C2,x=”<<n<<endl;
        }
};
int main(){
        CSample array1[2];
        CSample array2[2]={6,8};
        CSample array3[2]={12};
        CSample *array4=new Csample[3];
        Return 0;
}
```

答案：

C1

C1

C2,x=6

C2,x=8;

C2,x=12

C1

C1

C1

C1

7．下面程序的输出结果是。

```
#include <iostream>
using namespace std;
class Sample {
```

```
public:
 int v;
 Sample() {}
 Sample(int n) :v(n) { };
 Sample(const Sample & x) { v = 2 + x.v; }
} ;
Sample& PrintAndDouble(Sample o)
{
 cout << o.v;
 o.v = 2 * o.v;
 return o;
}
int main()
{
 Sample a(5);
 Sample b = a;
 Sample c = PrintAndDouble(b);
 cout << endl;
 cout << c.v << endl;
 //cout << hex << c.v << endl;
 Sample d;
 d = a;
 cout << d.v;
 return 0;
}
```

答案：

9

20

5

8．请写出下面程序的运行结果:4,6

请填空：

```
#include<iostream>
using namespace std;
class A{
	int val;
	public:
		A(int n){
			val=n;
		}
		int GetVal(){
			return val;
		}
```

```
};
class B:public A{
        private:
                int val;
                public:
                        B(int n):_____ {

                        };
                        int GetVal(){
                                return val;
                        }
};
int main(){
        B b1(2);

        return 0;

}
```

答案：

Val(4),A(6)

9．下面程序的输出结果是

0

5

请填空：

```
#include<iostream>
using namespace std;
class A{
        public:
                int val;
                A(_______){
                        val=n;
                }
                ______ GetObj(){
                        return _________;
                }
};

int main(){
        A a;
        cout<<a.val<<endl;
        a.GetObj()=5;
        cout<<a.val<<endl;
}
```

答案：

int n=0

A&

*this

10．下面程序的输出结果是：10

请补充 Sample 类的成员函数，不能增加成员变量

```
#include<iostream>
using namespace std;
class Sample{
        public:
                int v;
                Sample(int n):v(n){
                };
                _____________________
                Sample(Sample &obj){
                        this->v = 2 * obj.v;
                };
};
int main(){
        Sample a(5);
        Sample b=a;
        cout<<b.v;
        return 0;
}
```

答案：

```
Sample(Sample &obj){
        This->v=2*obj.v;
};
```

11. 下面程序的输出结构是：

5，5

5，5

请填空：

```
#include<iostream>
using namespace std;
class Base{
        public:
        int k;
        Base(int n):k(n){
        }
};
class Big {
        public:
                int v;
                Base b;
```

```
        Big _____ {
        };
        Big_____ {
        };
};
int main(){
    Big a1(5);
    Big a2=a1;


    return 0;
}
```

答案：

(int n):v(n),b(n)

(Big &x):v(x.v),b(x.b.k)

12．完成附录“魔静世界大作业”中提到的第一阶段作业

(省略)

第 13 章

1 如果将运算符“[]”重载为某个类的成员运算符（也即成员函数），则该成员函数的参数个数是(　)。

A．0 个　B．1　个　C．2　个　D．3 个

答案：B

2．如果将运算符“*”重载为菜个类的成员运算符（也即成员函数），则该成员函数的参数个数是(　)。

A．0 个　B．1　个　C．2　个　D．0　个或 1 个均可

答案：D

3．下面程序的输出是

```
3+4i
  5+6i
```

请补充 Complex 类的成员函数，不能加成员变量。

```
#include<iostream>
#include<cstring>
using namespace std;
class Complex{
private:
    double r,i;
public:
    void Print(){

    }
//在这里补充
```

```
//在这里补充
};
int main(){
	Complex a;

	a.Print();

	a.Print();
	return 0;
}
```

答案：

```
Complex &operator=(const char t[]) {
		if(!t) {
			r=0.0; i=0.0;
		} else {
			r = t[0] - '0';
			i = t[2] - '0';
		} return *this;
	}
```

4．下面的 MyInt 类只有一个成员变量。MyInt 类部的部分代码被隐藏了。假设下面的程序能编译通过，且输出结果是：

4,1

请写出被隐藏的部分（要求编写的容必须是能全部救进 MyInt 类部的，MyInt 的成员函数不允许使用静态变量）。

```
#include<iostream>
using namespace std;
class MyInt{
	int nVal;
	public:
		MyInt(int n){
			nVal=n;
		}
		int ReturnVal(){
			return nVal;
		}
	// 在这里补充

//在这里补充
};
int main(){
	MyInt objInt(10);
	objInt-2-1-3;
	cout<<objInt.ReturnVal();
```

```
        objInt-2-1;
        cout<<objInt.ReturnVal();
        return 0;
}
```

答案：

```
MyInt& operator - (int i){
        nVal -= i;
        return *this;
        }
```

5．下面的程序输出结果是

(4,5)

(7,8)

请填空：

```
#include<iostream>
using namespace std;
class Point{
        private:
                int x;
                int y;
        public:
                Point(int x_,int y_):x(x_),y(y_){
                };
                //
                ____________________________
};
____________operator<<(_________,const Point & p){
        _____________________________;
        Return_____;
}
```

```
int main(){
        cout<<Point(4,5)<<Point(7,8);
        return 0;
}
```

答案：

```
friend ostream& operator << (ostream &o,const Point &p);
friend ostream &
ostream &o
o
```

6．编写一个二维数组类 Array2，使得程序的输出结果是

4,5,6,7,

8,9,10,11,

next

0,1,2,3,

4,5,6,7,

8,9,10,11,

答案:

```
#include <iostream>
#include <cstring>
using namespace std;
class Array2
{
private:
    int row;//        数组行数
    int column;//        数组列数
    int* ptr;//        指向二维数组的指针
public:
    Array2()
        {
        ptr = NULL;
        }
    Array2(int paraRow, int paraColumn):row(paraRow),column(paraColumn)
        {
        ptr = new int[row * column];
        }
    Array2(Array2& a):row(a.row),column(a.column)
        {
        ptr = new int[row * column];
        memcpy(ptr, a.ptr, sizeof(int)*row*column);
        }
    Array2& operator= (const Array2 &a)
        {
        if (ptr) delete[] ptr;
        row = a.row;
        column = a.column;
        ptr = new int[row * column];
        memcpy(ptr, a.ptr, sizeof(int)*row*column);
        return *this;
        }
    ~Array2()
        {
        if (ptr) delete[] ptr;
```

```
    int* operator [] (int i)
        {
        return ptr + i*column;
        }
    int& operator() (int i, int j)
        {
        return ptr[i*column + j];
        }
} ;
int main()
{
    Array2 a(3,4);
    int i,j;
    for(  i = 0;i < 3; ++i )
        for(  j = 0; j < 4; j ++ )
            a[i][j] = i * 4 + j;
    for(  i = 0;i < 3; ++i ) {
        for(  j = 0; j < 4; j ++ ) {

            }
        cout << endl;
        }

    Array2 b;      b = a;
    for(  i = 0;i < 3; ++i ) {
        for(  j = 0; j < 4; j ++ ) {

            }
        cout << endl;
        }
    return 0;
}
```

7．编写一个 Mystring 类，使得程序的输出结果是

1. abcd-efgh-abcd-
2. abcd-
3.
4. abcd-efgh-
5. efgh-
6. c
7. abcd-
8. ijAl-
9. ijAl-mnop
10. qrst-abcd-

about

big

me

take

abcd

qrst-abcd-

答案:

```
#include <cstdlib>
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;
class MyString : public string {
public:
        MyString():string() {};
        MyString( const char * s):string(s){};
        MyString( const string & s ): string(s){};
        MyString operator() ( int s, int l) {
                return this->substr(s,l);
        }
} ;
int main()
{
        s4 = s3;
        s3 = s1 + s3;
        s2 = s1;
        s1[2] = 'A' ;
```

```
        sort(SArray,SArray+4);
        for( int i = 0;i < 4;i ++ )
        cout << SArray[i] << endl;
        //s1  的从下标 0 开始长度为 4 的子串
        cout << s1(0,4) << endl;
        //s1  的从下标 5 开始长度为 10 的子串
        cout << s1(5,10) << endl;
        return 0;
}
```

第 14 章

1．以下说法不正确的是（假设在公有派生情况下）（ ）。

A. 可以将基类对象赋值给派生类对象

B. 可以将派生类对象的地址赋值给基类指针

c．可以将派生类对象赋值给基类的引用

D. 可以将派生类对象赋值给基类对象

答案：A

2．写出下面程序的输出结果。

```
#include<iostream>
using namespace std;
class B{
public:
        B(){

        }
        ~B(){

        }
};
class C:public B{
public:
        C(){

        }
        ~C(){

        }
};
int main(){
        C * pc = new C;
        delete pc;
        return 0;
}
```

答案：

B_Con

C_Con

C_Des

B_Des

3．写出下面程序的输出结果。

```
#include<iostream>
using namespace std;
class Base{
        public:
                int val;
                Base(){
                }
                ~Base(){

                }
};
class Base1:virtual public Base{
};
class Base2:virtual public Base{
};
class Derived:public Base1,public Base2{
};
int main(){
        Derived d;
        return 0;
}
```

答案：

Base Constructor

Base Destructor

4．按照第 13 章的第 7 题的要求编写 MyString 类，但 MyString 类必须是从 stnng 类派生而来。

提示 1□□□□□□□□□”MyString”□“ string “替换，那么题目的程序除了最后两条语句无法编译通过外，其他语句都没有问题，而且输出和前面的结果吻合。也就是说，MyString 类对 string 类的功能扩充只能表达在最后两条语句上面。

提示 2：string 类有一个成员函数 string substr(int start,int length), 能够求从 start 开始位置，长度为 length 的字串

答案（疑惑）

```
class MyString : public string {
public:
        MyString():string() {};
        MyString( const char * s):string(s){};
```

```
        MyString( const string & s ): string(s){};
        MyString operator() ( int s, int l) {
                return this->substr(s,l);
        }
};
```

5. 完成附魔兽世界作业中的的二阶段。

(省略)

第 15 章

1．以下说确的是(　)

A.在虚函数中不能使用 this　指针

B.在构造甬数中调用虚函数，不是动态联编

C.抽象类的成员函数都是纯虚函数

D.构造函数和析构函数都不是虚构函数

答案：B

2．写出下面程序的输出结果

```
#include<iostream>
using namespace std;
class A{
        public:
        A(){
        }
        virtual void func(){

        }
        ~A(){
        }
        virtual void fund(){

        }
};
class B:public A{
        public:
        B(){
                func();
        }
        void fun(){
                func();
        }
        ~B(){
                fund();
        }
};
class C:public B{
        public:
```

```
    C(){}
    void func(){

    }
    ~C(){
        fund();
    }
    void fund(){

    }
};
int main(){
    C c;
    return 0;
}
```

答案：

A::func

C::fund

A::fund

3．写出下面程序的输出结果。

```
#include<iostream>
using namespace std;
class A{
    public:
        virtual ~A(){

        }
};
class B:public A{
    public:
        virtual ~B(){
        }
};
class C:public B{
    public:
        virtual ~C(){

        }
};
int main(){
    A * pa = new C;
    delete pa;
    A a;
```

```
        return 0;
}
```

答案：

DestructC

DestructB

DestructA

DestructA

4．写出下面程序的输出结果：

```
#include<iostream>
using namespace std;
class A{
        public:
                virtual void func(){
                }
                virtual void fund(){

                }
                void fun(){

                }
};
class B:public A{
        public:
        B(){
                func();
        }
        void fun(){
                func();
        }
};
class C:public B{
        public:
                C(){
                }
                void func(){

                }
                void fund(){

                }
};
int main(){
        A * pa = new B();
```

```
        pa->fun();
        B * pb = new C();
        pb->fun();
        return 0;
}
```

答案：

A::func

A::fun

A::func

C::func

5．下面程序的输出结果是

```
#include<iostream>
using namespace std;
class A{
        private:
                int nVal;
        public:
                void Fun(){

                }
                void Do(){

                }
};
class B:public A{
        public:
                virtual void Do(){

                }
};
class C:public B{
        public:
                void Do(){

                }
                void Fun(){

                }
};
void Call(B& p){
        p.Fun();
        p.Do();
}
int main(){
```