【关键字】实现

C++面向对象程序设计教程课后题答案

1.1 什么是面向东西程序设计?

面向东西程序设计是一种新的程序设计范型.这种范型的主要特征是:

程序=东西+消息

面向东西程序的基本元素是东西。

主要结构特点是：

第一， 程序一般由类的定义和类的使用两部分组成;

第二， 程序中的一切操作都是通过向东西发送消息来实现的。

1.2 什么是东西?什么是类?东西与类之间的关系是什么?

东西是描述其属性的数据以及对这些数据施加的一组操作封装在一起构成的统一体。

类就是具有相同的数据和相同的操作的一组东西的集合，也就是说，类是对具有相同数据结构和相同操作的一类东西的描述。

类和东西之间的关系是抽象和具体的关系。类是多个东西进行综合抽象的结果，一个东西是类的一个实例。

1.3 现实世界中的东西有哪些特征？请举例说明。

现实世界中的东西具有以下特征：

1) 每一个东西必须有一个名字以区别于其他东西;

2) 用属性来描述东西的某些特征;

3) 有一组操作，每组操作决定东西的一种行为;

4)　　东西的行为可以分为两类：一类是作用于自身的行为，另一类是作用于其他东西的行为。

例如一个教师是一个东西。每个教师东西有自己的名字来和别的教师区别。教师具有编号，姓名，年龄，职称，专业等属性。教师拥有走路，吃饭，授课等行为操作。走路，吃饭是作用于自身的行为，授课是作用于其他东西的行为。

1.4 什么是消息？消息具有什么性质？

一个东西向另一个东西发出的请求成为“消息”。

消息具有以下 3 个性质：

1)　　同一个东西可以接收不同形式的多个消息，做出不同的相应；

2)　　相同形式的消息可以传递给不同的东西，所做出的响应可以是不同的；

3)　　对消息的响应并不是必须的，东西可以响应消息，也可以不响应。

1.5 什么是抽象和封装？请举例说明。

抽象是将有关事物的共性归纳、集中的过程。

例如：把所有具有大学生学籍的人归为一类，成为“大学生”，这就是一个抽象。

封装是指把数据和实现操作的代码集中起来放在东西内部，并尽可能隐藏东西的内部细节。

例如：每一台洗衣机都有出厂日期、机器编号等属性，也有启动、暂停、选择等操作。人们在使用洗衣机的时候只需要按下对应的按钮，而不用关心具体的内部实现。这就是封装。

1.6 什么是继承？请举例说明。

继承就是允许派生类使用基类的数据和操作，同时，派生类还可以增加新的操作和数据。

例如：哺乳动物是一种热血、有毛发、用奶哺育幼崽的动物； 狗是有犬牙、食肉、特定的骨骼结构、群居的哺乳动物。 狗就继承了哺乳动物。

1.7 若类之间具有继承关系，则他们之间具有什么特征？

若类之间具有继承关系，则他们之间具有下列几个特征：

1) 类间具有共享特征（包括数据和操作代码的共享）；

2) 类间具有差别或新增部分（包括非共享的数据和操作代码）；

3) 类具有层次结构。

1.8 什么是单继承、多继承？请举例说明。

单继承是指每个派生类只直接继承了一个基类的特征。例如狗继承自哺乳动物。

多继承是指多个基类派生出一个派生类的继承关系。比如玩具车同时继承自玩具和车。

1.9 什么是多态？请举例说明。

多态是指不同的对象收到相同的消息时执行不同的操作。

例如，有一个窗口类对象，还有一个棋子类对象。当我们发出“移动”消息时，两个对象的行为不同。

1.10 面向对象程序设计的主要优点是什么？

1. 可提高程序的重用性；
2. 可控制程序的复杂性；
3. 可改善程序的可维护性；
4. 能够更好地支持大型程序设计；
5. 增强了计算机处理信息的范围；

能够很好地适应新的硬件环境。

2.1 简述 C++的主要特点。

1) C++是 C 的超集，保持与 C 的兼容。

2) 保持了 C 的简洁、高效和接近汇编语言等特点，并对 C 的功能作了不少扩充。用 C++编写的程序比 C 更安全，可读性更好，代码结构更为合理。

3) 程序质量高。

4)　　增加了面向对象机制。

2.2

```
#include <iostream>
using namespace std;

int main()
{
    int a, b, d, min;
    cout << "Enter two numbers:";
    cin >> a >> b;
    min = a > b ? b : a;
    for(d = 2; d < min; d++)
    {
        if(((a % d) == 0) && ((b % d ) == 0)) break;
    }
    if (d == min)
    {
        cout << "No common denominators" << endl;
        return 0;
    }
    cout << "The lowest common denominator is" << d << endl;
    return 0;
}
```

2.3 有效

2.4 没有函数声明；

函数定义没有写返回值类型。

2.5 （1）等价，函数声明可以省略参数的名字。

（2）不等价，第二个的函数定义不能省略参数的名字。

2.6-2.10 CDAAB

2.11-2.15 ACBDC

2.16-2.17 DC

2.18

101

2.19

10 10

2.20

10

20

2.21 举例说明可以使用 const 替代#define 以消除#define 的不安全性。

```
#include <iostream>
using namespace std;

int main()
{
    int a = 1;
#define T1 a+a
#define T2 T1-T1
    cout << "T2 is " << T2 <<endl;
```

```
    return 0;
}
```

上面这个程序，初看应该输出 T2 is 0

但是实际上，得出 T2 is 2

如果把#define 换成 const， 则可以输出想要的结果。

2.22 用动态分配空间的方法，计算 Fibonacci 数列的前 20 项，并存储到动态分配的空间中。

```
#include <iostream>
using namespace std;

int main()
{
    int *pi = new int[20];
    *pi = 1;
    pi[1] = 1;
    for(int i = 2; i < 20; i++)
    {
        pi[i] = pi[i - 2] + pi[i - 1];
    }
    return 0;
}
```

2.23 重载 sroot 函数，输出一个数的二次方根。

```
#include <iostream>
using namespace std;

double sroot(int num)
{
    return (double)sqrt((double)num);
```

```
}

double sroot(long num)
{
    return (double)sqrt((double)num);
}

double sroot (double num)
{
    return (double)sqrt(num);
}
int main()
{
    return 0;
}
```

2.24 解决百钱问题。将一元人民币换成 1、2、5 分的硬币，有多少种换法？

```
#include <iostream>
using namespace std;

int main()
{
    int num = 0; //总共换法的总数。初始化为 0。
    for(int i = 0; i <= 100; i++)
    {
        for(int j = 0; j <= 50; j++)
        {
            if((i + 2*j) > 100)
            {
                break;
            }
            for(int k = 0; k <= 20; k++)
            {
                if((i + 2*j + 5*k) == 100)
                {
                    num++;
                    cout << "1 分" << i << "个;" << "2 分" << j << "个;"
<< "5 分" << k << "个;" << endl;
                }
                if ((i + 2*j + 5*k) > 100)
                {
```

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

2.25 输入两个整数，按由小到大的顺序输出。要求使用变量的引用。

```
#include <iostream>
using namespace std;

void swap(int &a, int &b)
{
    a = a + b;
    b = a - b;
    a = a - b;
}

int main()
{
    int a, b;
    cin >> a >>b;
    if(a > b)
    {
        swap(a, b);
    }
    cout << a << "," << b << endl;
    return 0;
}
```

2.26 用二分法求解 f(x)=0的根。

```
#include <iostream>
using namespace std;

double Fun(double x)
{
    return 35*x +25; //假设 f(x)=35x+25
```

```
}

int main()
{
    double a, b;
    cin >> a;
    if(Fun(a) == 0)
    {
        cout << "x = " << a << endl;
        return 0;
    }
    do
    {
        cin >> b;
    }
    while ((Fun(a) * Fun(b)) >= 0);

    if(Fun(b) == 0)
    {
        cout << "x = " << b << endl;
        return 0;
    }

    if(a > b)
    {
        a = a + b;
        b = a - b;
        a = a - b;
    }

    while(1)
    {
        if(Fun((a + b)/2) == 0)
        {
            cout << "x = " << (a + b)/2 << endl;
            return 0;
        }
        if(Fun(a) * Fun((a + b)/2) < 0)
        {
            b = (a + b)/2;
        }
        if(Fun(b) * Fun((a + b)/2) < 0)
        {
            a = (a + b)/2;
```

```
        }
    }
    return 0;
}
```

3.1 类声明的一般格式是什么?

```
class 类名
{
    [private:]
        私有数据成员和成员函数
    public:
        公有数据成员和成员函数
}
```

3.2 构造函数和析构函数的主要作用是什么？它们各自有什么特性？

构造函数是一种特殊的成员函数，它主要用于为对象分配空间，进行初始化。

构造函数的名字必须与类名相同，而不能由用户任意命名。它可以有任意类型的参数，但不能具有返回值类型。

析构函数通常用于执行一些清理任务，如释放分配给对象的内存空间等。

析构函数名与类名相同，但它前面必须加一个波浪号。不能有返回值，也不能有参数。

3.3 什么是对象数组？

所谓对象数组，是指每一个数组元素都是对象的数组。

3.4 什么是 this 指针？它的主要作用是什么？

C++为成员函数提供了一个名为 this 的指针，这个指针称为自引用指针。每当创建一个对象时，系统就把 this 指针初始化为指向该对象。

一个类的所有对象合用一份成员函数，this 指针可以帮助对象辨别出当前调用的是自己的那个对象的数据成员和函数。

3.5 友元函数有什么作用？

友元函数可以在类的外部访问类的私有成员或保护成员。

3.6

（1）声明并定义了 P2， P3， 并用默认无参构造函数初始化。

（2）声明并定义了 P2， 并调用 Point 类的拷贝构造函数用 P1 对 P2 进行初始化。

（3）声明并定义了 P2， 并调用 Point 类的拷贝构造函数用 P1 对 P2 进行初始化。

（4）调用拷贝构造函数，将 P1 的成员值赋值给 P4 的成员。

3.7-3.10 BCCB

3.11-3.15 BAABA

3.16-3.17 BB

3.18

10, 20

30, 48

50, 68

70, 80

90,16

11,120

3.19

Constructing

10

100

Destructing

3.20

3objects in existence

4objects in existence after allocation

3objects in existence after deletion

3.21

Counting at0

Counting at9

3.22

Default constructor called.

Default constructor called.

Default constructor called.

Construcotor:a=1,b=2

Construcotor:a=3,b=4

Construcotor:a=5,b=6

3.23

Con.

Copy con.

default.

Copy con.

3.24

A=5

B=14

A=9

B=14

3.25

5,7

22.25

3.26

Constructing

Constructing

A=5

B=15

A=10

B=15

Destructing

Destructing

3.27

void pintStu();函数只有声明，没有定义。

age 是私有成员，不能用对象直接调用。

3.28

void printStu() 和 void setSno(int s) 没有加限定符 Student::

void setAge(int a)在类中没有声明

3.29

构造函数不能定义为私有。否则无法创建对象。

3.30 下面是一个计算器类的定义，请完成该类成员函数的实现。

```
class counter
{
public:
    counter(int number);
    void increment();                   //给原始值加 1
    void decrement();                   //给原始值减 1
    int getvalue();                     //取的计数器值
    int print();                        //显示计数
private:
    int value;
};
```

```
counter::counter(int number)
{
    value = number;
}
void counter::increment()
{
    ++value;
}
void counter::decrement()
{
    --value;
}
int counter::getvalue()
{
    return value;
}
int counter::print()
{
    cout << value <<endl;
    return value;
}
```

3.31 根据注释语句提示，实现类 Date 的成员函数

```
#include <iostream>
using namespace std;

class Date
{
public:
    void printDate();
    void setDay(int d);
    void setMonth(int m);
    void setYear(int y);
private:
    int day, month, year;
};
void Date::printDate()
{
    cout << "今天是" << year << "年" << month << "月" << day << "日" <<
endl;
```

```
}
void Date::setDay(int d)
{
    day = d;
}
void Date::setMonth(int m)
{
    month = m;
}
void Date::setYear(int y)
{
    year = y;
}
int main()
{
    Date testDay;
    testDay.setDay(5);
    testDay.setMonth(10);
    testDay.setYear(2003);
    testDay.printDate();
    return 0;
}
```

3.32 建立类 cylinder，cylinder的构造函数被传递了两个 double值，分别表示圆柱体的半径和高度。用类 cylinder计算圆柱体的体积，并存储在一个 double变量中。在类 cylinder中包含一个成员函数 vol，用来显示每个 cylinder对象的体积。

```
const int PI = 3.14;

class cylinder
{
private:
    double radius, height, volume;
public:
    cylinder(int rad, int hei);
    double getVolume();
    void vol();
};

cylinder::cylinder(int rad, int hei)
{
```

```
    radius = rad;
    height = hei;
}

double cylinder::getVolume()
{
    volume = PI * radius * radius *height;
    return volume;
}

void cylinder::vol()
{
    cout << "圆柱体的体积是:  " << volume <<endl;
}
```

3.33 构建一个类 book，其中包含有两个私有数据成员 qu 和 price，将 qu 初始化为 1~5，将 price初始化为 qu 的 10 倍，建立一个有 5 个元素的数组对象。显示每个对象数组元素的 qu*price值。

```
class book
{
private:
    int qu, price;
public:
    book(int qu);
    int mult();
};

book::book(int q)
{
    if(q < 1 || q > 5)
    {
        qu = 1;
    }
    else
    {
        qu = q;
    }
    price = 10 * qu;
}
```

```
int book::mult()
{
    return qu * price;
}

int main()
{
    book books[5] = {1,2,3,4,5};
    for(int i = 0; i < 5; i++)
    {
        cout << books[i].mult() << " ";
    }
}
```

3.34 修改 3.33，通过对象指针访问对象数组，使程序以相反的顺序显示每个对象数组元素的 qu*price值。

```
class book
{
private:
    int qu, price;
public:
    book(int qu);
    int mult();
};

book::book(int q)
{
    if(q < 1 || q > 5)
    {
        qu = 1;
    }
    else
    {
        qu = q;
    }
    price = 10 * qu;
}

int book::mult()
{
```

```
    return qu * price;
}

int main()
{
    book books[5] = {1,2,3,4,5};
    book *p = books;
    p += 4;
    for(int i = 0; i < 5; i++)
    {
        cout << p->mult() << " ";
        --p;
    }
    return 0;
}
```

3.35 构建一个类 Stock，含字符数组 stockcode及整型数组成员 quan、双精度型数据成员 price。构造函数含 3 个参数：字符数组 na[及 q、p。当定义 Stock的类对象时，将对象的第一个字符串参数赋给数据成员 stockcode。第 2 和第 3 个参数分别赋给 quan、price。未设置第 2 和第 3 个参数时，quan 的值为 1000，price的值为 8.98成员函数 print没有形参，需使用 this指针，显示对象数据成员的内容。编写程序显示对象数据成员的值。

```
#include <iostream>
using namespace std;

class Stock
{
private:
    char stockcode[25];
    int quan;
    double price;
public:
    Stock(char na[], int q = 1000, double p = 8.98);
    Stock(char na[]);
    void print();
};

Stock::Stock(char na[], int q = 1000, double p = 8.98)
{
    strcpy(stockcode, na);
    quan = q;
```

```
    price = p;
}

void Stock::print()
{
    cout << "stockcode: " << this->stockcode << " quan: " << this->quan
<< " price: " << this->price << endl;
}

int main()
{
    Stock stock1( "600001", 3000, 5.67);
    Stock stock2("600002");
    stock1.print();
    stock2.print();
    return 0;
}
```

3.36 编写一个程序，已有若干学生的数据，包括学号、姓名、成绩，要求输出这些学生的数据并计算出学生人数和平均成绩（要求将学生人数和总成绩用静态数据成员表示）。

```
#include <iostream>
using namespace std;

class student
{
private:
    char name[25], studentNo[10];
    int score;
    static int sum;
    static int totalScore;
public:
    student(char na[], char stuNo[], int sc);
    void show();
    static void showTotal();
};

student::student(char na[], char stuNo[], int sc)
{
    strcpy(name, na);
    strcpy(studentNo, stuNo);
```

```
    score = sc;
    ++sum;
    totalScore += sc;
}

void student::show()
{
    cout << "姓名:  " << name <<endl;
    cout << "学号:  " << studentNo << endl;
    cout << "成绩:  " << score << endl;
}

void student::showTotal()
{
    cout << "总人数:  " << sum << endl;
    cout << "平均成绩:  " << (double)totalScore/sum <<endl;
}

int student::sum = 0;
int student::totalScore = 0;

int main()
{
    student s1("张无忌", "111254", 75);
    student s2("李莫愁", "254114", 60);
    student s3("小龙女", "112587", 88);
    s1.show();
    s2.show();
    s3.show();
    student::showTotal();
return 0;
}
```

4.1 有哪几种继承方式?每种方式的派生类对基类成员的继承性如何?

公有继承,私有继承和保护继承。

基类的私有成员，无论哪种继承方式都不能访问。

公有继承不改变基类的公有和保护成员的访问限制。

私有继承将基类的公有和保护成员都变成私有。

保护继承将基类的公有和保护成员都变成保护。

4.2 派生类能否直接访问基类的私有成员？若否，应如何实现？

不能。可以在基类里添加一个公有成员函数来访问私有成员，派生类就能继承这个公有成员函数，实现对基类私有成员的访问。

4.3 保护成员有哪些特性？保护成员以公有方式或私有方式继承后的访问特性如何？

保护成员只能被本类或本类的派生类所访问，在类或派生类外是不能被访问的。

后面一问见第一题的答案。

4.4 派生类构造函数和析构函数的执行顺序是怎样的？

构造函数是先执行父类的构造函数，再执行类中其他类对象的构造函数，再执行本类的构造函数。如果同级有多个构造函数，则按声明顺序执行。

析构函数与构造函数的执行顺序刚好相反。

4.5 派生类构造函数和析构函数的构造规则是怎样的？

派生类名（参数总表）：　基类名（参数总表）

{

派生类新增数据成员的初始化语句

}

派生类中的析构函数与基类无关。

4.6 什么是多继承？多继承时，构造函数与析构函数的执行顺序是怎样的？

多继承是指一个类同时继承自多个不同的基类。

执行顺序同 4.4

4.7 在类的派生中为何要引入虚基类？虚基类构造函数的调用顺序是如何规定的？

如果一个类有多个直接基类，而这些直接基类又有一个共同的基类，则在最底层的派生类中会保留这个简介的共同基类数据成员的多份同名成员。在访问这些同名成员的时候，会产生二义性。为了解决二义性，引入了虚基类。

1) 如果虚基类中定义有带形参的构造函数，并且没有定义默认形式的构造函数，则整个继承结构中，所有直接或间接的派生类都必须在构造函数的成员初始化表中列出对虚基类构造函数的调用，以初始化在虚基类中定义的数据成员。

2) 建立一个对象时，如果这个对象中含有从虚基类继承来的成员，则虚基类的成员是由最远派生类的构造函数通过调用虚基类的构造函数进行初始化的。该派生类的其他基类对虚基类构造函数的调用都自动被忽略。

3) 若同一层次中同时包含虚基类和非虚基类，应先调用虚基类的构造函数，再调用非虚基类的构造函数，最后调用派生类的构造函数。

4.8-4.11 ACCB

4.12

Constructor B1.

Constructor B2.

Constructor A.

3

2

1