C++程序设计第四节课官方笔记

目录

一、课件下载及重播方法

二、教材节构图

三、本章知识点及考频总结

（一）选择题（共 5 道）

1. 一个完整的 C++程序包括以下几部分。

- 一个主函数，可以调用其他函数，但不能被调用，也称为主程序。
- 用户定义的任意多个的类及全局函数。
- 全局说明。在所有函数和类定义之外的变量说明及函数原型。
- 注释。
- 头文件。

对于比较大的程序，根据主函数和各用户定义的类及全局函数的功能及相互关系，可以把类及全局函数划分为几个程序文件，包括.cpp 文件和.h 文件。**.cpp 文件是源程序文件，.h 文件是头文件。**

2. 实现成员函数时要指明类的名称，在类体外定义的一般格式如下：

```
返回值类型 类名::函数成员名(参数表)
{
    函数体
}
```

成员函数并非每个对象各自存有一份。成员函数和普通函数一样，在内存中只有一份，它可以作用于不同的对象，为类中各对象共享。

通常，因为函数体代码较长，所以在类体内仅给出成员函数的原型，然后在类体外给出对应的函数体。如果函数体定义在类体内，则系统将其视为内联函数。

类中定义的成员函数允许重载。

3. 创建类对象的基本形式

方法一的基本格式如下：

```
类名 对象名;
```

或是

```
类名 对象名(参数);
```

或是

```
类名 对象名=类名(参数);
```

可以扩展为多个对象，如下所示：

```
类名 对象名 1，对象名 2，…;
```

或是

```
类名 对象名 1(参数 1)，对象名 2(参数 2)，…;
```

方法二的基本格式如下：

```
类名 *对象指针名 = new 类名;
```

或是

```
类名 *对象指针名 = new 类名();
```

或是

```
类名 *对象指针名 = new 类名(参数);
```

用 new 创建对象时返回的是一个对象指针，这个指针指向本类刚创建的这个对象。C++分配给指针的仅仅是存储指针值的空间，而对象所占用的空间分配在堆上。使用 new 创建的对象，必须用 delete 来撤销。

使用“类名 *对象指针名 = new 类名;”创建对象时，调用无参的构造函数。如果这个构造函数是由编译器为类提供的，则类中成员变量不进行初始化。

使用“类名 *对象指针名 = new 类名( );”创建对象时，也调用无参的构造函数。如果这个构造函数是由编译器为类提供的，则对类中的成员变量进行初始化。

与基本数据类型一样，还可以声明对象的引用、对象的指针及对象的数组。

声明对象引用，即变量别名的基本格式如下：

```
类名 &对象引用名 = 对象;
```

声明对象指针，即指向对象的指针的基本格式如下：

```
类名 *对象指针名 = 对象的地址;
```

声明对象数组的格式如下：

```
类名 对象数组名[数组大小];
```

同类型的对象之间可以相互赋值。对象和对象指针都可以用作函数参数。函数的返回值可以是对象或指向对象的指针。

例如，定义了类 C 后，可以有如下的声明：

```
C a1,b1;          //定义了 C 类的对象 a1 和 b1
C *p = &a1;       //定义了指向对象 a1 的 C 类类型的指针 p
C &R = b1;        //定义了 C 类类型对象 b1 的引用 R
C A[3];           //定义了 C 类类型对象的数组 A，含 3 个元素
```

程序运行中，创建对象就是为对象分配内存，此时使用类作为模板，也就是按照类定义中声明的成员变量为对象分配内存。

通过对象访问成员变量的一般格式如下：

```
对象名.成员变量名
```

调用成员函数的一般格式如下：

```
对象名.成员函数名(参数表)
```

除了“对象名.成员名”的格式外，还可以使用指针或引用的方式来访问类成员。如果是通过指针访问成员变量，则点运算符.换为箭头运算符->，即使用“指针->成员名”的方式来访问对象的成员。

使用访问范围说明符可以说明类成员的访问限制，这些访问范围说明符也称为**访问修饰符**，包括 public（公有）、private（私有）和 protected（保护），它们都是 C++的关键字。

4. 访问范围说明符的含义

访问范围说明符	含义	作用
public	公有的	使用它修饰的类的成员可以在程序的任何地方被访问。
private	私有的	使用它修饰的类的成员仅能在本类内被访问。
protected	保护的	它的作用介于 public 与 private 之间，使用它修饰的类的成员能在本类内及子类中被访问。

5. 隐藏的作用

设置私有成员的机制叫作“隐藏”。“隐藏”的一个目的就是强制对私有成员变量的访问一定要通过公有成员函数进行。这样做的好处是：如果以后修改了成员变量的类型等属性，只需要更改成员函数即可；否则，所有直接访问成员变量的语句都需要修改。

“隐藏”机制还可以避免对对象的不正确操作。

如果存在两个或多个具有包含关系的作用域，外层声明了一个标识符，而内层没有再次声明同名标识符，那么外层标识符在内层仍然可见。如果在内层声明了同名标识符，则外层标识符在内层不可见，这时称内层标识符隐藏了外层同名标识符，这种机制称为隐藏规则。

（二）主观题（共 1 道）

分析程序的执行结果。

```
#include <iostream>
#include <string>
```

```
using namespace std;
class CEmployee
{
private:
    string szName;          //姓名
    int salary;             //工资
public:
    void setName(string);   //设置姓名
    string getName();       //获取姓名
    void setSalary(int);    //设置工资
    int getSalary();        //获取工资
    int averageSalary(CEmployee);   //计算两人的平均工资
};
void CEmployee::setName(string name)
{   szName = name;
}
string CEmployee::getName()
{   return szName;
}
void CEmployee::setSalary(int mon)
{   salary = mon;
}
int CEmployee::getSalary()
{   return salary;
}
int CEmployee::averageSalary(CEmployee e1)
{   return ( salary + e1 .getSalary() )/2;
}
int main()
{   CEmployee eT, eY;
    //eT.szName ="Tom1234567";      //编译错误，不能直接访问私有成员
    eT.setName("Tom1234567");       //需要通过公有成员函数访问
    //eT.salary=5000;               //编译错误，不能直接访问私有成员
    eT.setSalary(5000);             //需要通过公有成员函数访问
    cout<<eT.getName()<<endl;       //输出 Tom1234567
    eY.setName("Yong7654321");
    eY.setSalary(3500);
    cout<<eY.getName()<<endl;       //输出 Yong7654321
    cout<<"aver="<<eT.averageSalary(eY)<<endl; //输出 aver=4250
    return 0;
}
```

四、配套练习题

1、局部变量可以隐藏全局变量，那么在有同名全局变量和局部变量的情形时，可以用下列哪一项提供对全局变量的访问（　　）。

A:作用域运算符

B:指针运算符

C:提取运算符

D:插入运算符

2、下列关于类和对象的说法中，正确的是（　　）。

A:编译器为每个类和类的对象分配内存

B:类的对象具有成员函数的副本

C:类的成员函数由类来调用

D:编译器为每个对象的成员变量分配内存

3、下列关于对象数组的描述中，错误的是（　　）。

A:对象数组的下标是从 0 开始的

B:对象数组的数组名是个常量指针

C:对象数组的每个元素是同一个类的对象

D:对象数组只能赋初值，不能被赋值

[参考答案] ADD

五、其余课程安排