一、简答题

第一章 绪论（略）

第二章 简单程序设计

标识符

1、构成规则:①标识符由三类字符构成:英文大小写字母;数字 0……9;下划线。②必须由字母或下划线开头；后面可以跟随字母、数字或下划线。③C++语言区分大小写，即大小写字母有不同的含义，例如:num，Num，NUM 为 3 个不同的标识符。④标识符不能与关键字同名，不能与库函数或自定义函数同名。

基本数据类型

类型名	长度（字节）	取值范围
bool	1	false，true
char	1	-128~127
signed char	1	-128~127
unsigned char	1	0~255
short（signed short）	2	-32768~32767
unsigned short	2	0~65535
int（signed int）	4	-2^{31}~2^{31}-1
unsigned int	4	0~2^{32}-1
long（signed long）	4	-2^{31}~2^{31}-1
unsigned long	4	0~2^{32}-1
long long	8	-2^{63}~2^{63}-1
unsigned long long	8	0~2^{64}-1
float	4	绝对值范围3.4×10^{-38}~3.4×10^{38}
double	8	绝对值范围1.7×10^{-308}~1.7×10^{308}
long double	8	绝对值范围1.7×10^{-308}~1.7×10^{308}

注：表中各类型的长度和取值范围，以面向IA32处理器的msvc12和2 gcc4.8为准

运算符（书上都有）

真题：请说明 C++语言中以下的选项中，哪些是正确的常量？正确的请指出其数据类型，错误的请说明错误的理由？

（1）0679　　(2) .234　　(3) ‘\t’　　(4) ‘abcd’

参考答案：

正确的变量名选项是（2） double 类型（可以不带 0）、（3） char 类型

错误的包括(1)八进制不能含数字 9;(0 开头为 8 进制,0x 开头为 16 进制）（4）char 类型只能是单个字符

1.

第三章 函数

函数原型

1、把被调函数定义的位置放在主调函数之前，用这种方法也可以省去被调函数的原型说明;

2. 被调函数定义的位置放在主调函数之后，则必须在函数调用之前使用被调函数的原型说明。

内联函数

1、函数调用有一定的时间和空间开销，影响程序的执行效率。特别是对于一些函数体代码不是很大，但又频繁地被调用的函数，则引入内联函数。

2、在程序编译时编译系统将程序中出现内联函数调用的地方用函数体进行替换。引入内联函数可以提 高程序的运行效率，节省调用函数的时间开销，是一种以空间换时间的方案。

第四章 类与对象

构造函数

1、成员初始化顺序与它们在类定义中的出现顺序一致， 而不是在初始值中出现的顺序!

2、const 成员或引用成员必须将其初始化!

默认构造函数

1、不定义构造函数时，编译器自动产生默认构造函数。

2、不可以同时定义有参和无参的默认构造函数。

复制构造函数

1、复制构造函数被调用的三种情况：①定义一个对象时，以本类另一个对象作为初始值,发生复制构造;②如果函数的形参是类的对象，调用函数时，将使用实参对象初始化形参对象，发生复制构造；③如果函数的返回值是类的对象，函数执行完成返回主调函数时，将使用 return 语句中的对象初始化一个临时无名对象，传递给主调函数，此时发生复制构造。

析构函数

1、析构函数的功能是在对象的生存期即将结束的时刻，由编译系统自动调用来完成一些清理工作。它的调用完成之后,对象也就消失了,相应的内存空间也被释放。

2、析构函数也是类的一个公有成员函数，它的名称是由类名前 面加“~”构成，也不指定返回值类型。

3、和构造函数不同的是，析构函数不能有参数，因此不能重载。

移动构造函数

对持久存在变量的引用称为左值引用,用&表示(即第 3 章引用类型)

对短暂存在可被移动的右值的引用称之为右值引用，用&&表示

```
float n = 6;
float &lr_n = n; //左值引用
float &&rr_n = n; //错误，右值引用不能绑定到左值
```

float &&rr_n = n * n; //右值表达式绑定到右值引用

通过标准库中<utility>的 move 函数可将左值对象移动为右值

float n = 10;

float &&rr_n = std::move(n); //将 n 转化为右值

使用 move 函数承诺除对 n 重新赋值或销毁外，不以 rr_n 以外方式使用

基于右值引用，移动构造函数通过移动数据方式构造新对象，与复制构造函数类似，移动构造函数参数为该类对象的右值引用。示例如下

#include<utility>

class astring

{ public :

std::string s;

astring (astring&& o) noexcept: s(std::move(o.s)) //显式移动所有成员

{ 函数体 }

}

移动构造函数不分配新内存，理论上不会报错，为配合异常捕获机制，需声明 noexcept 表明不会抛出异常（将于 12 章异常处理介绍），被移动的对象不应再使用，需要销毁或重新赋值。

什么时候该触发移动构造？有可被利用的临时对象，例如函数返回局部对象时，为避免不必要的复制构造，可以使用移动构造。

default、delete 函数

1、=default:要求编译器生成默认或复制构造函数

2、=delete:要求编译器删除指定函数，除析构函数之外均可删除。

类的组合

1、**构造函数的设计：**①原则：不仅要负责对本类中的基本类型成员数据赋初值，也要对对象成员初始化。②声明形式：类名::类名(对象成员所需的形参，本类成员形参) :对象 1(参数)，对象 2(参数)，.....{ //函数体其他语句}

2、**成员对象构造函数调用顺序**：①先调用内嵌对象的构造函数，调用顺序按照内嵌对象在组合类的定义中出现的次序（与初始化列表中出现的顺序无关!）②执行本类构造函数的函数体。(P119 例 4-4)

3、析构函数与构造函数调用顺序正好相反。

UML 类图

1、类和对象：数据成员在上，函数成员在下，Public、Private、Protected 分别用+、-、#表示。

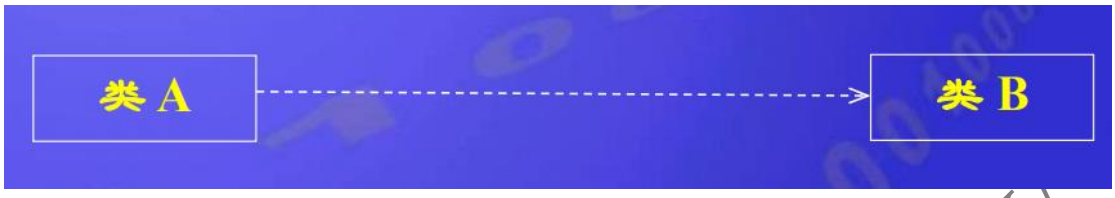
2、几种关系的图形标识

①重数：指的是一个关系的多重性，表示一个类与另一个类之间的关系的数量或重复性。

标记	说明
*	任何数目的对象（包括 0）
1	恰好 1 个对象
n	恰好 n 个对象
0..1	0 个或 1 个对象（表面关联是可选的）
n..m	最少为 n、最多为 m 个对象（m、n 都是整数）

2,4	离散的结合（如 2 个或者 4 个）

②依赖关系：重数为 1，表示一个类使用了另一个类中的某些成员。在类图中，用虚线表示。例如，汽车类使用引擎类的某些方法。

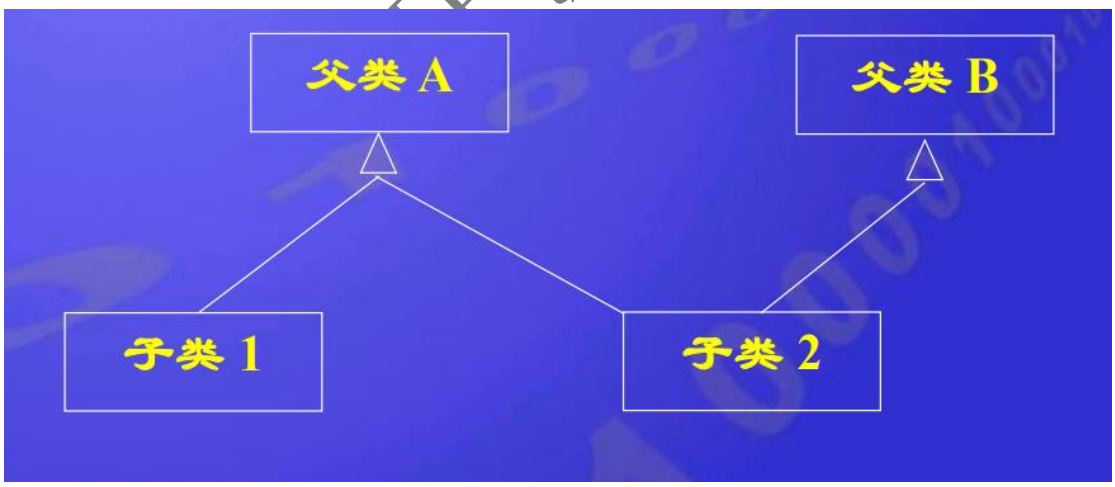


③关联关系的重数可以为 0、1、、*1..* 等，表示两个类之间的对象之间的数量和重复性。在类图中，用实线表示，可以用箭头表示方向。例如，一个订单类和一个客户类之间可能有一个关联关系，因为一个客户可以拥有多个订单，而一个订单只能属于一个客户。

④包含关系的重数可以为 1 或 0..1，表示一个类包含另一个类的对象，被包含的类通常是被多个类共享的。在类图中，用实线和带实心菱形的连线表示，菱形指向被包含的类。例如，一个学校类包含多个班级类的对象，每个班级只属于一个学校。

聚集表示类之间的关系是整体与部分的关系，“包含”、“组成”、“分为……部分”等都是聚集关系。共享聚集：部分可以参加多个整体；组成聚集（组合）：整体拥有各个部分，整体与部分共存，如果整体不存在了，那么部分也就不存在了。

⑤继承关系表示一个类从另一个类继承了它的属性和方法，并可以在其基础上添加或修改自己的属性和方法。



第五章 数据的共享与保护

作用域

1、函数原型作用域：函数原型声明时形式参数的作用范围。

2、局部作用域（其中变量为局部变量）：①函数形参的作用域：从声

明处到整个函数体结束之时。②函数体内声明变量：从声明处到所在块结束的大括号为止。

3、类作用域：①如果在 X 的成员函数中没有声明同名的局部作用域标识符，那么可以在函数体中直接访问。②通过 x.m 或者 X::m 来访问。③通过 ptr->m 来访问。

4、命名空间作用域（其中变量为全局变量）

静态生存期

1、全局变量拥有这种生存期，该生存期与程序的运行期相同。

2、函数内部中定义，要用 static，具有全局寿命和局部可见性（函数内外输出的值可能会不同——参考课本 P151 例 5-2)。

3、静态生存期的变量未被初始化时，会被以 0 值初始化。

动态生存期

1、动态生存期也称作局部生存期，该对象诞生于声明点，结束于其所在块执行完毕之时。

2、局部生存期变量不指定初值时，初值将不确定。

类的静态成员

1、静态数据成员：①用 static 关键字声明。②不管产生多少个对象，类的静态数据成员拥有单一的存储空间， 为该类的所有对象共享。静态数据成员具有静态生存期。③必须在类外定义和初始化，用(::)来指明所属的类。④静态数据成员可以是 public、private、protected 权限之一。(P155 例 5-4)

2、静态函数成员：①类外代码可以使用类名和作用域操作符来调用

静态成员函数。②静态成员函数只能引用属于该类的静态数据成员或静态成员函数。③静态成员函数属于整个类，不属于类中的某个对象，能在类的范围内共享。

类的友元

1、友元函数（独立函数或者类成员函数）是在类声明中由关键字 friend 修饰说明的非成员函数。

```
friend 返回值类型 函数名(参数表);
```

2、友元类：若一个类为另一个类的友元，则此类的所有成员函数都能访问对方类的私有成员。

```
friend class 类名B ;
```

3、友元的特性：①友元关系不能传递。②友元关系是单向的。③友元关系是不被继承的。

共享数据的保护

1、常对象：const 类型 对象名——必须进行初始化，不能更新。

2、常成员：const 修饰的类成员：常数据成员和常函数成员——同上。

3、常引用：const 类型 &引用名——被引用的对象不能被更新。

4、常数值：类型 const 数组名[大小]：数组元素不能被更新。

5、常指针：类型 const &引用名——指向的值不能变。

外部变量

1、如果一个变量除了在定义它的源文件中可以使用外，还 能被其它文件使用，那么就称这个变量是外部变量。

2、文件作用域中定义的变量，缺省情况下都是外部变量， 但在其它

文件中如果需要使用这一变量，需要用 extern 关键字加以声明。

外部函数

1、在所有类之外声明的函数（也就是非成员函数），都是 具有文件作用域的。

2、这样的函数都可以在不同的编译单元中被调用，只要在 调用之前进行引用性声明（即声明函数原型）即可。也 可以在声明函数原型或定义函数时用 extern 修饰，其效 果与不加修饰的缺省状态是一样的

第六章 数组、指针与字符串

数组的声明与使用

1、数组的声明方法：类型说明符 数组名[常量表达式][常量表达式]……;

2、数组名是地址常量。

3、数组元素的访问：数组名[下标 1][下标 2]……

4、数组元素下标从 0，必须先声明，后使用。

5、只能逐个引用数组元素，而不能一次引用整个数组

6、越界访问存储单元：C++编译系统不检查。

7、数组名作参数，形参被转换为指针，实参应是数组名。（P193 例 6-2）

对象数组

1、声明：类名 数组名[元素个数]

2、访问方法：数组名[下标].成员名

3、只能引用单个数组元素，每个数组都是一个对象，初始化过程即调用构造函数对每个元素进行初始化。

指针和指针变量

1、指针：一个变量的地址

2、指针变量：专门存放变量地址的变量

3、若静态指针变量，系统自动初始化为 NULL，此时指针指向对象不存在。

与地址相关的运算“*”与“&”

1、&：若 int a;则&a 表示变量 a 在内存中的起始地址。

2、*：若 int *p；则表示 p 是一个 int 型指针。

指针的赋值

1、指针名=地址，如 p=&a；(则*p=a)

2、“地址”中存放的数据类型与指针类型必须相符。

3、向指针变量赋的值必须是地址常量或变量，不能是普通整数。但可以赋值为整数 0，表示空指针。

4、指针的类型是它所指向变量的类型，而不是指针本身数据值的类型。

5、允许声明指向 void 类型的指针。该指针可以被赋予任何类型对象的地址。

6、指向常量的指针：const int *p=&a，指针可以改变指向的对象，但不能通过指针来改变所指对象的值。

7、指针类型的常量：int *const p =&a，不能改变指针指向的对象。

指针运算

1、指针与整数的加减运算：①指针 p 加上或减去 n，其意义是指针当前指向位置的前方或后方 n 个数据的地址。②这种运算的结果值取决于指针指向的数据类型。③p1[n1]等价于*(p1 + n1)

2、指针加一减一运算：指向下一个或前一个数据 y=*p++（*与++优先级相同）

3、指针运算：p1-p2——两个指针必须类型相同，结果为整数（p1 与 p2 指向的对象之间间隔的数据个数),它们通常指向同一个数组空间。

4、关系运算：必须是指向同数据类型之间的指针才有意义，也可以和 NULL 之间做关系运算。

用指针处理数组元素

1、若 int a[10], *pa; pa=&a[0], 则*pa 就是 a[0]， *(pa+1)就是 a[1], … ，*(pa+i)就是 a[i] 。a[i]、 *(pa+i)、 *(a+i),、pa[i]都是等效的。（不能写 a++因为 a 是数组首地址是常量）

用指针作为函数参数

1、以地址方式传递数据,可以用来返回函数处理结果。(P209 例 6-10)

2、实参是数组名时形参可以是指针或者数组名。

3、形参是数组名时，编译系统将其转换为对应的指针类型。

指向函数的指针

1、函数指针：存放函数代码首地址的变量，可以像使用函数名一样使用指向函数的指针来调用函数。

2、声明：数据类型 (*函数指针名)（形参表）

3、调用：函数指针名=函数名（P212 例 6-11）

对象指针

1、声明形式：类名*指针对象名

2、通过指针访问对象：对象指针名->成员名

ptr->getx()相当于(*ptr).getx()

this 指针

1、隐含于每一个类的成员函数中的特殊指针。

2、明确地指出了成员函数当前所操作的数据所属的对象。

3、当通过一个对象调用成员函数，系统先将该对象 的地址赋给 this 指针，然后调用成员函数，成员函数 对对象的数据成员进行操作时，就隐含使用了 this 指针。

申请和释放动态数组

1、分配：new 类型名[数组长度]（数组长度可以是任意值）

2、释放：delete[]数组名 p：p 必须是用 new 分配的数组首地址。

深拷贝与浅拷贝

1、浅拷贝：实现对象间数据元素的一一对应复制。

2、深拷贝：当被复制的对象数据成员是指针类型时，不是复 制该指针成员本身，而是将指针所指对象进行复制。

用字符数组存储和处理字符串

1、字符串常量（例：“program”）：①各字符连续、顺序存放，每个字符占一个字节，以‘\0’结尾，相当于一个隐含创建的字符常量数

组。

p	r	o	g	r	a	m	\0

②“program”出现在表达式中，表示这一 char 数组的首地址。③首地址可以赋给 char 常量指针： const char *STRING1 = "program";④若中间有空格，也要占用一个字节。

2、字符串变量:可以显式创建字符数组来表示字符串变量,例如： char str[8] = "program";字符指针模拟字符数组，表示字符串变量，例如: char *str= "program"

String 类

1、常用构造函数:

string(); //缺省构造函数，建立一个长度为 0 的串

string(const char *s); //用指针 s 所指向的字符串常量初始化 string 类的对象

string(const string& rhs); //拷贝构造函数

例:

string s1; //建立一个空字符串

string s2 = “abc”; //用常量建立一个初值为”abc”的字符串

string s3 = s2;//执行拷贝构造函数，用 s2 的值作为 s3 的初值

2、string 类操作符（详见课本 P231）

多级指针

定义	含义

int i;	定义整型变量 i
int *p;	p 为指向整型数据的指针变量
int a[n]	定义含 n 个元素的整型数组 a
int *p[n];	n 个指向整型数据的指针变量组成的指针数组 p
int (*p)[n];	p 为指向含 n 个元素的一维整型数组的指针变量
int f();	f 为返回整型数的函数
int *p();	p 为返回指针的函数，该指针指向一个整型数据
int (*p)();	p 为指向函数的指针变量，该函数返回整型数
int **p;	p 为指针变量，它指向一个指向整型数据的指针变量
int (*p)(int);	指向函数的指针，函数返回 int 型变量
int *(*p)(int);	指向函数的指针，函数返回 int 型变量
int (*p[3])(int);	函数指针数组，函数返回 int 型变量
int *(*p[3])(int);	函数指针数组，函数返回 int 型指针

第七章 继承与派生

类的继承与派生

1、**类的继承**：保持已有类的特性而构造新类的过程称为继承。

2、**类的派生**：在已有类的基础上新增自己的特性而产生新类的过程称为派生。

3、**基类**：被继承的已有类。

4、**派生类**：派生出的新类。

5、**直接基类与间接基类**：直接参与派生出某类的基类称为直接基类，基类的基类甚至更高层的基类称为间接基类。

派生类的声明

1、**单继承**：一个派生类只有一个直接基类。

2、**多继承**：一个派生类同时拥有多个基类。

3、**派生类成员**：指除了从基类继承来的所有成员之外，新增的数据和成员函数。

4、**派生类不能继承基类的构造函数和析构函数。**

5、**同名隐藏**：如果派生类声明了一个和某基类成员同名的新成员，派生类的新成员就隐藏了外层同名成员。

6、**派生类的生成过程**：吸收基类成员、改造基类成员、添加新的成员。

访问控制

1、**不同继承方式的影响主要体现在**：派生类成员对基类成员的访问权限、通过派生类对象对基类成员的访问权限。

2、**公有继承**：基类的公有成员和保护成员的访问属性在派生类中不变，而基类的私有成员不可直接访问。

3、**私有继承**：基类中的公有成员和保护成员都以私有成员身份出现在派生类中，而基类的私有成员在派生类中不可直接访问。

4、**保护继承**：基类的公有成员和保护成员都已保护成员身份出现在

派生类中，而基类的私有成员不可直接发访问。

类型兼容规则

1、**类型兼容规则**：在需要基类对象的任何地方，都可以使用公有派生类的对象来替代，反之则禁止。替代后，派生类对象可以作为基类的对象使用，但只能使用从基类继承的成员。

2、**类型转换（兼容）的情况**：①派生类的对象可以隐含转换为基类对象。即用派生类对象中从基类继承来的成员，逐个赋值给基类对象的成员。②派生类的对象可以初始化基类的引用。③派生类的指针可以隐含转换为基类的指针。**（这些情况下，调用能使用从基类继承的成员。）**

派生类的构造函数

1、构造派生类对象时，就要对基类的成员对象和新增成员对象进行初始化。

2、由于基类的构造函数和析构函数不能被继承，在派生类中，如果**对派生类新增成员进行初始化**，就必须为派生类添加新的构造函数。

3、派生类对于基类的很多成员函数是不能直接访问的，因此要完成对基类成员对象的初始化工作，需要调用基类的构造函数。

4、定义构造函数时，只需要对本类中新增成员进行初始化对继 承来的基类成员的初始化，**自动调用基类构造函数完成**。

5、派生类的构造函数需要给基类的构造函数传递参数。

6、**什么时候需要声明派生类的构造函数**：如果对基类初始化时，需要调用基类的带形参表的构造函数时，派生类就必须声明构造函数。

7、**构造函数的执行顺序**：①**调用基类构造函数（只调用直接基类）**，调用顺序按照它们被继承时声明的顺序（从左向右）。②对本类成员初始化列表中的基本类型成员和**对象成员**进行初始化，初始化顺序按照它们在类中声明的顺序对象成员初始化是自动调用对象所属类的构造函数完成的。③执行派生类的**构造函数体中**的内容。

派生类的析构函数

1、析构函数也不被继承，派生类自行声明。

2、声明方法与一般（无继承关系时）类的析构函数相同。

3、不需要显式地调用基类的析构函数，系统会**自动隐式调用**。

4、**析构函数的执行顺序：**析构函数的调用次序与构造函数相反。

作用域限定

1、隐藏规则：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，而内层没用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。

2、若未特别限定，则通过派生类对象使用的是派生类中的同名成员。

3、如要通过派生类对象访问基类中被隐藏的同名成员，应使用基类名和作用域操作符（::）来限定。

二义性问题

1、在多继承时，基类与派生类之间，或基类之间出现同名成员时，将出现访问时的二义性（不确定性)——采用虚函数或同名隐藏来解决。

2、当派生类从多个基类派生，而这些基类又从同一个基类派生，则在访问此共同基类中的成员时，将产生二义性——采用虚基类来解决。

虚基类

1、将共同基类设置为虚基类，这时从不同的路径继承过来的同名数据成员在内存中就只有一个副本，同一个函数名也只有一个映射。

2、**虚基类的作用**：①主要用来解决多继承时可能发生的对同一基类继承多次而产生的二义性问题。②为最远的派生类提供唯一的基类成员，而不重复产生多次拷贝。

3、第一级继承时就要将共同基类设计为虚基类。

4、最（远）派生类：建立对象时所指定的类。

5、虚基类的成员是由最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。

6、在整个继承结构中，直接或间接继承虚基类的所有派生类，都必须在构造函数的成员初始化表中给出对虚基类的构造函数的调用。如果未列出，则表示调用该虚基类的默认构造函数。

7、在建立对象时，**只有最派生类的构造函数调用虚基类的构造函数**，该派生类的其他基类对虚基类构造函数的调用被忽略。

第八章 多态性

多态性概述

1、**多态性的定义**：**多态是指操作接口具有表现多种形态的能力。**即能根据操作环境的不同采用不同的处理方式。多态性是面向对象系统的主要特性之一，在这样的系统中，一组具有相同基本语义的方法能在同一接口下为不同的对象服务。

2、**多态性的分类**：C++语言支持的多态性可以按其实现的时机分为**编译时多态**（例如，运算符重载等）和**运行时多态**（例如虚函数）两类。

运算符重载

1、不能重载的运算符：“.”、“.*”、“::”、“?:”等。

2、重载之后运算符的优先级和结合性都不会改变。

3、运算符重载是针对新类型数据的实际需要，对原有运算符进行适当的改造。

4、**两种重载方式**：重载为类的非静态成员函数和重载为非成员函数。

虚函数

1、用 virtual 关键字说明的函数

2、虚函数是实现运行时多态性的基础

3、C++中的虚函数是动态绑定的函数

4、虚函数必须是非静态的成员函数，虚函数经过派生之后，就可以实现运行过程中的多态。

虚析构函数

1、为什么需要虚析构函数？①可能通过基类指针删除派生类对象；②如果你打算允许其他人通过基类指针调用对象的析构函数（通过 delete 这样做是正常的），就需要让基类的析构函数成为虚函数，否则执行 delete 的结果是不确定的

C++中不能定义为虚函数的？

1、普通函数（非成员函数）：只能被 overload，不能被 override

2、构造函数

3、内联成员函数：inline 函数在编译时被展开，虚函数在运行时 才能动态的绑定函数

4、静态成员函数：对于每个类来说只有一份代码，所有的对象 都共享这一份代码，所以没有要动态绑定的必要性

5、友元函数：没有继承特性的函数，就没有虚函数之说

纯虚函数

1、纯虚函数是一个在基类中声明的虚函数，它在该基类中没有定义具体的操作内容，要求各派生类根据实际需要定义自己的版本。

2、带有纯虚函数的类称为抽象类。

二、分析改错题

常见错误：

1、变量类型错误

2、变量初始值错误（常见*=0）

3、结构、算法等条件错误

4、没有定义拷贝复制函数而进行深拷贝处理。

5、类外访问私有成员

6、动态内存未释放

7、越界访问存储单元

8、改变常量的值（如对数组首地址 a 使用 a++）

三、程序分析题

1、派生类中虚函数的覆盖（课本母题：P317 例 8-4）

真题：（10 分）请仔细阅读以下程序，写出程序的输出结果

```
1.  #include <iostream>
2.  using namespace std;
```

```
class A
{ public:
    virtual void print(void)  {    cout<<"A::print()"<<endl;  }
};
class B:public A
{public:
    virtual void print(void)  {    cout<<"B::print()"<<endl; };
};
class C:public B
{public:
    virtual void print(void) {  cout<<"C::print()"<<endl; }
};
void print(A a)
{    a.print();
}
int main( )
{  A a, *pa,*pb,*pc;
   B b;
   C c;
   pa=&a;    pb=&b;    pc=&c;
   pa->print();
   pb->print();
   pc->print();
   cin.ignore ();
   return 0;
}
```

【参考答案】(目标 3) 得分点 3,3,4

A::print()

B::print()

C::print()

2、类的继承中构造函数的调用（课本母题：P266 例 7-4）

8. 真题：请仔细阅读以下程序，简述构造函数调用顺序，说明程序运行结果。

```
#include <iostream>
using namespace std;
class A
{   public:        A() {cout<<"Constructing A "<<endl;}
};
class B
{   private:       int a;
    public:            B(int j) :a(j){a += j; cout<<"Constructing B: "<<a<<endl;}
```

```
};
class C:public A, public B
{   private: int b;
    public:     C(int x, int y):B(y),b(x){cout<<"Constructing C #"<<b<<endl;}
};
class D: public C
{
public:
    D(int a, int b, int c, int d) : C(a+b,c+d),memberB2(a+b+c+d),memberB1(c-d,a-b){  }
private:
    A memberB3;
    B memberB2;
    C memberB1;
};
int main()
{     D obj(1,3,5,7);
      cin.ignore();
      return 0;
}
```

【参考答案】

Constructing A

Constructing B: 24

Constructing C #4

Constructing A

Constructing B: 32

Constructing A

Constructing B: -4

Constructing C #-2

四、编程实现题

1、 合并数组，剔除相同数据

（通过 for 循环遍历数组，若遇到相同数据则跳出循环并给 bool 数据赋值）

```
void xorIntArrayMerge(int A[], int lenA, int B[], int lenB, int C[])
{
  int currentElement = 0;
  int indexC = 0;
```

```
bool equalFlag = false;
int i = 0, j = 0;
//从数组A中剔除和数组B中重复的元素
for (i = 0; i < lenA; i++) {
  currentElement = *(A + i);
  for (j = 0; j < lenB; j++) {
    if (currentElement == *(B + j)) {
      equalFlag = true;
      break;
    }
  }
  if (!equalFlag) {
    *(C + indexC) = currentElement;
    indexC++;
  }
  equalFlag = false;
}
equalFlag = false;
//从数组B中剔除和数组A中重复的元素
for (j = 0; j < lenB; j++) {
  currentElement = *(B + j);
  for (i = 0; i < lenA; i++) {
    if (currentElement == *(A + i)) {
      equalFlag = true;
      break;
    }
  }
  if (!equalFlag) {
    *(C + indexC) = currentElement;
    indexC++;
  }
  equalFlag = false;
}
return;
}
```

2、计算位数+逆序打印

```
//处理负整数
if (inputN < 0) {
  inputN = -inputN;
  cout << "-";
}
```

```
    do {
      right_digit = inputN % 10;
      cout << right_digit;
      inputN /= 10;
      length++;
    } while (inputN != 0);
    cout << endl;
    cout << "数据位数为：" << length << endl;
```

3、数组逆序排放

请编程完成函数 reverseArray(a,n,m,k)的代码，该函数的功能是给定一组原始数据数组（n 为数据个数），将其中从第 m 个数据开始的 k 个数据逆序存放。

例如，有如下定义

float a[10] = {2.1, 5.2, 6.0, ***1.5, 6.3, 9.5, 10.0, 12.14****, 1.1, 23.1};*

则函数调用 **reverseArray(a, 10, 4,5);**结束后，数组 a 中数据存储顺序为*{2.1, 5.2, 6.0,****12.14, 10.0,9.5,6.3,1.5,****1.1,23.1}.*

说明：

（1）请自行补充完善所需的主函数或辅助函数；

（2）在主函数中调用reverseArray(a,10,m,k))；数组a、变量m、k的取值均由运行时输入。

【参考答案】

```
#include <iostream>
using namespace std;
void reverseArray(double *a,int n, int m, int k)
{   double temp;
    if(m<n-1&&m-2+k<n-1)
    for(int i=m-1,j=m-2+k;i<j;i++;j--)
    {   temp=a[i];   a[i]=a[j];   a[j]=temp;}
}
int main()
{    int m,k;
     double a[10];
     for(int i=0; i<10;i++)   cin>>a[i];
     cin>>m>>k;
     reverseArray(a,10,m,k);
   for (int i=0;i<10;i++)   cout<<a[i]<<endl;
   cin.ignore();
   return 0;
}
```

4、取最值

假设有如下定义
float arr[10] = {2.1, 5.2, 6.0, 1.5, 6.3, 9.5, 10.0, 12.14, 1.1, 23.1};
int num;
float min;
则函数调用 find(arr, 10, min,num);结束后，变量 min 中保存了数组 arr 中最小值 1.1，变量 num 保存了最小值 1.1 的下标 8。
请编写函数 find()。 说明：主函数不用编写

参考答案：

```
void find(float *a,int n, float &min,int &m){
    min = a[0];m=0;
    for (int i=0;i<n;i++){
        if(a[i]<min){
            min=a[i];
            m=i;
        }
    }
}
```

5、类的定义

请完整定义和实现一个采用分数表示的有理数类 Rational，要求 Rational 类支持如下的操作：
Rational x; //x 的 up 为 1，x 的 down 为 1
Rational y(2,4) //将 2 和 4 进行约分处理（即消除最大公约数）后，y 的 up 为 1，y 的 down 为 2
x*y //计算两个有理数 x 和 y 的乘积，返回 Rational 对象
x==y //判断两个有理数是否相等，如相等，返回 true；不相等，返回 false
说明：Rational 类中，有两个 private 权限的数据成员，分别是用整数表示的分子 up 和分母 down（分母不为 0）。成员函数 Minmultiple、Maxdivisor 分别用于求解最小公倍数和最大公约数，这里略去成员函数 Minmultiple、Maxdivisor 的实现，假设已经实现并可直接调用。

```
class Rational
{   private:
        int up,down;          //分子、分母
    int Minmultiple(int a,int b); //最小公倍数函数，函数假设已实现，函数体略
        int Maxdivisor(int a,int b);//最大公约数，函数假设已实现，函数体略
  public:
      //请补充 Rational 类的的定义
};
```

【参考答案】

```
class Rational
{
public:
    Rational(int num1=1,int num2=1);
    Rational operator*(Rational r1);
  bool operator==(Rational rhs);
```

```
private:
      int up;
      int down;
      int Minmultiple(int a,int b); //最小公倍数
      int Maxdivisor(int a,int b);//最大公约数
};
Rational Rational::operator*(Rational r1)
{
   int Ndown,Nup,i;
   Ndown=down*r1.down;
   Nup=up*r1.up;
   i=Maxdivisor(Nup,Ndown);
   return Rational(Nup/i,Ndown/i);
}
bool Rational::operator==(Rational rhs)
{   if(rhs.up/rhs.down == up/down)    return true;
else    return false;
}
```

6、运算符的重载（课本母题 P310 例 8-1,P311 例 8-2）

真题：有一个 Time 类，包含数据成员 minute(分)和 sec(秒)，模拟秒表，每次走一秒， 满 60 秒进一分钟，此时秒又从 0 开始算，Time 类的声明如下所示：

```
class Time{
public:
      Time(){minute=0; sec=0;}
      Time( int m, int s):minute(m),sec(s){}
      void display(){ cout<<minute<<”;”<<sec<<endl;
private: int minute,sec;
};
```

请为 Time 类增加前缀++和后缀++运算符函数：

（1）给出增加前缀++和后缀++运算符函数之后类的声明；

（2）给出 Time 类的前缀++和后缀++运算符函数实现。

参考答案：

```
class Time{
public:
      Time(){minute=0; sec=0;}
      Time( int m, int s):minute(m),sec(s){}
      void display(){ cout<<minute<<”;”<<sec<<endl;
      Time &operator++();
      Time operator++(int);
private: int minute,sec;
};
Time & Time::operator(){
   if(++sec>=60)
   {   sec-=60;   ++minute;}
   return *this;
}
```

```
Time Time::operator(int){
   Time temp(*this);
   if(++sec>=60)
   {   sec-=60;   ++minute;}
   return temp;
}
```

五、编程实现题

1、继承、派生与多态——张量、向量、标量、矩阵类

```
#include <iostream>
using namespace std;
//定义 Tensors 抽象类类
class Tensors
{
public:
  virtual  ~Tensors() {};
  virtual int getSize() = 0;          //纯虚函数，取数据个数
  virtual int getDimension() = 0;     //纯虚函数，取数据维数
  double* data = nullptr; //数据
protected:
  int size = 0;           //数据个数
  int dimension = 0;      //维数
  string name = "Tensor";
};

//定义 Vector 类
class Vector :public Tensors
{
public:
  Vector(const double* initData, int size);
  Vector(const Vector& v);  //拷贝构造
  ~Vector();
  int getSize();
  int getDimension();
  Vector& operator+(const Vector& v1);
  Vector& operator=(const Vector& from);
};
```

```
//构造Vector
Vector::Vector(const double* initData, int size)
{
  double* ptr;
  this->size = size;
  this->dimension = 2;
  this->name = "Vector";
  this->data = new double[size];
  ptr = this->data;

  cout << "构造Vector : " << endl;
  //初始化数据
  for (int i = 0; i < size; i++) {
    *(ptr + i) = *(initData + i);
    cout << "*(ptr + i) =" << *(ptr + i) << ", *(initData + i)=" << *(initData + i) <<
endl;
  }
}
//拷贝构造
Vector::Vector(const Vector& v) {
  size = v.size;
  dimension = v.dimension;
  data = new double[v.size];
  name = "copy构造";
  for (int i = 0; i < size; i++) {
    *(data + i) = *(v.data + i);
  }
}
//析构Vector
Vector::~Vector()
{
  cout << endl << "析构Vector : " << this->name << ",  " << this->data << endl;
  delete[] this->data;
}
//+运算符重载
Vector& Vector::operator+(const Vector& v1) {
  if (this->size == v1.size) {
    cout << endl << " Vector()+ 函数,name:" << this->name << " | this->size: "
      << this->size << " | v1.size: " << v1.size << endl;
    for (int i = 0; i < size; i++) {
      *(this->data + i) += *(v1.data + i);
      cout << endl << " *(this->data + i):" << *(this->data + i) << " | *(v1.data + i) :"
        << *(v1.data + i) << endl;
    }
```

```
    }
    return *(this);
}
//=运算符重载
Vector& Vector::operator=(const Vector& from) {
    Vector reVector(from.data, from.size);
    return reVector;

}

int Vector::getSize() {
    return size;
}
int Vector::getDimension() {
    return dimension;
}

//定义 Scalar 类
class Scalar :public Tensors
{
public:
    Scalar(double* data);
    ~Scalar();
    int getSize();
    int getDimension();
    Scalar& operator+(const Scalar& vv);
    Scalar& operator=(const Scalar& from);
};
Scalar::Scalar(double* initData)
{
    this->data = initData;
    this->size = 1;
    this->dimension = 1;
}
Scalar::~Scalar() {}
int Scalar::getSize() {
    return size;
}
int Scalar::getDimension() {
    return dimension;
}

//定义 Matrix 类
class Matrix : public Tensors
```

```
{
public:
  Matrix(double* data, int size, int cols, int rows);
  ~Matrix();
  int getSize();
  int getDimension();
  Matrix& operator+(const Matrix& m1);
  Matrix& operator=(const Matrix& from);
  };
Matrix::Matrix(double* data, int size, int cols, int rows) {
  //初始化Matrix ....
}
Matrix::~Matrix() {}
int Matrix::getSize() {
  return size;
}
int Matrix::getDimension() {
  return dimension;
}

int main() {
  double dataVector1[] = { 2.0,3.0 };
  int sizeVector1 = 2;
  double dataVector2[] = { 4.0,5.0 };
  int sizeVector2 = 2;
  Vector v1(dataVector1, sizeVector1);
  Vector v2(dataVector2, sizeVector2);
  v1 = v1 + v2;
  cout << "v1中的数据为: " << endl;
  for (int i = 0; i < sizeVector1; i++)  {
    cout << *(v1.data + i) << "  ";
  }
  cin.ignore();
  return 0;
}
```

2、类的设计

某高校有四类员工：教师、实验员、行政人员，教师兼行政人员；共有的信息包括：编号、姓名、性别、年龄等。其中，教师还包含的信息有：所在系部、专业、职称；实验员还包含的信息由：所在实验室、职务；行政人员还包含的信息有：政治面貌、职称等。

2、功能要求

（1）添加功能：程序能够任意添加上述四类人员的记录，可提供选择界面供用户选择所要添加的人员类别，要求员工的编号要唯一，如果添加了重复编号的记录时，则提示数据添加重复并取消添加。

（2）查询功能：可根据编号、姓名等信息对已添加的记录进行查询，如果未找到，给出相应的提示信息，如果找到，则显示相应的记录信息。
（3）显示功能：可显示当前系统中所有记录，每条记录占据一行。
（4）编辑功能：可根据查询结果对相应的记录进行修改，修改时注意编号的唯一性。
（5）删除功能：主要实现对已添加的人员记录进行删除。如果当前系统中没有相应的人员记录，则提示“记录为空!”并返回操作；否则，输入要删除的人员的编号或姓名，根据所输入的信息删除该人员记录，如果没有找到该人员信息，则提示相应的记录不存。
（6）统计功能：能根据多种参数进行人员的统计。能统计四类人员数量以及总数，统计男、女员工的数量。
（7）保存功能：可将当前系统中各类人员记录存入文件中，存入方式任意。
（8）读取功能：可将保存在文件中的人员信息读入到当前系统中，供用户进行使用。

```
/*-------------------------------------共 有 类-------------------------------------*/

class CCommon//公有类
{
protected:
int number;//编号
string name;//姓名
string sex;//性别
int age;//年龄
};
/*----------------------教 师 类-------------------------------------*/
class Cteacher:public CCommon//教师类
{
protected:
string department;//所在系部
string profession;//专业
 string position;//职称
public:
void set(int _number,string _name,string _sex,int_age,string _department,string
_profession,string _position)
{
number = _number;
name = _name;
sex = _sex;
age = _age;
department = _department;
profession = _profession;
position = _position;
}//输入教师数据
void output();
~Cteacher()
{
}//析构函数
```

```
friend int find_nu(Cteacher teacher[], int _nu, int&t);
friend int find_na(Cteacher teacher[], string _na, int&t);
friend void add(Cteacher teacher[], int _fa);
friend void dele(Cteacher teacher[], int _fa);
friend void closefile(Cteacher teacher[] ,int &t);
friend void _statistics(Cteacher teacher[], int &t);
};
void teacher_manage();
/*----------------------------实 验 员 类-----------------------------------------*/
class Cassistant:public CCommon//实验员类
{
protected:
string laboratory;//所在实验室
string position;//职称
public:
void set(int _number,string _name,string _sex,int_age,string _laboratory,string
_position)
{
number = _number;
name = _name;
sex = _sex;
age = _age;
laboratory = _laboratory;
position = _position;
}//输入实验员数据
void output();
~Cassistant()
{
}//析构函数
friend int find_nu(Cassistant assistant[], int _nu, int&t);
friend int find_na(Cassistant assistant[], string _na,int &t);
friend void add(Cassistant assistant[], int _fa);
friend void dele(Cassistant assistant[], int _fa);
 friend voidclosefile(Cassistant assistant[] ,int &t);
friend void _statistics(Cassistant assistant[], int&t);
};
void assistant_manage();
/*----------------------------行 政 人 员 类-----------------------------------------*/
class Cexecution:public CCommon//行政人员类
{
protected:
string politics;//政治面貌
string position;//职称
public:
```

```
void set(int _number,string _name,string _sex,int_age,string _politics,string_position)
{
number = _number;
name = _name;
sex = _sex;
age = _age;
politics = _politics;
position = _position;
}//输入行政人员数据
void output();
~Cexecution()
{
}//析构函数
friend int find_nu(Cexecution execution[], int _nu, int&t);
friend int find_na(Cexecution execution[], string _na,int &t);
friend void add(Cexecution execution[], int _fa);
friend void dele(Cexecution execution[], int _fa);
friend void closefile(Cexecution execution[] ,int&t);
friend void _statistics(Cexecution execution[], int&t);
};
void execution_manage();
/*--------------------------行 政 人 员 兼 教 师 类-----------------------------------*/
class Ctea_exe:public CCommon//行政人员兼教师类
{
protected:
string politics;//政治面貌
string department;//所在系部
string profession;//专业
string position;//职称
public:
void set(int _number,string _name,string _sex,int_age,string _politics,string
_department,string _profession,string _position)
{
number = _number;
name = _name;
sex = _sex;
age = _age;
politics = _politics;
department = _department;
  profession =_profession;
position = _position;
}//输入行政人员兼教师数据
void output();
~Ctea_exe()
```

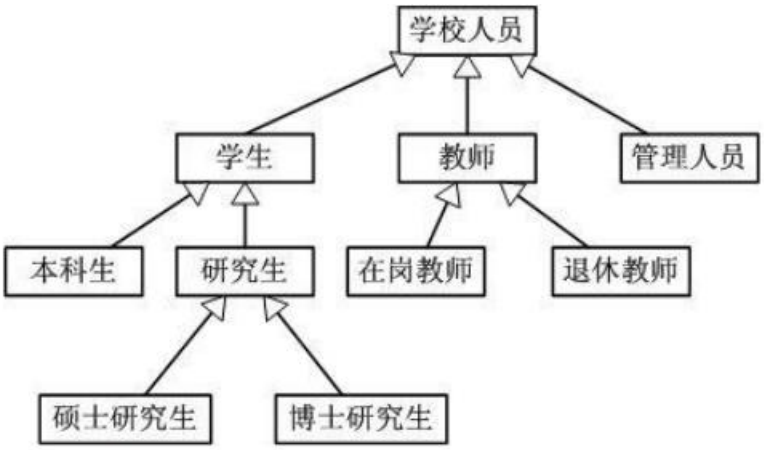
```
{
}//析构函数
friend int find_nu(Ctea_exe tea_exe[], int _nu, int&t);
friend int find_na(Ctea_exe tea_exe[], string _na, int&t);
friend void add(Ctea_exe tea_exe[], int _fa);
friend void dele(Ctea_exe tea_exe[], int _fa);
friend void closefile(Ctea_exe tea_exe[] ,int &t);
friend void _statistics(Ctea_exe tea_exe[], int &t);
};
void tea_exe_manage();
/*---------------------------------统 计 总 人 数--------------------------------------*/
void statistics_all();//数据统计数
void statistics_teacher();
void statistics_assistant();
void statistics_execution();
void statistics_tea_exe();
void order();
```

3、类与对象真题

分析设计出健康信息登记管理模块：下图是某大学人员的分类图，现需要对各类人员的信息 进行登记，每日收集和分析人员的健康状态。

此模块能够实现的业务功能包括： 1) 登记各类人员的基本信息,如：编号、姓名、性别、手机号码、证件类型（工作证、学生证、退休证）、 证件号码等；2) 登记居住信息，如：当前所在国家/地区、当前省份、当前城市、当前社区、详细住址等； 3) 登记每日健康信息，如：登记时间、体温、健康状态（健康、发热、疑似、确诊、治愈、其它）等； 4) 统计分析功能，包括：统计全体人员各种健康状态人数的日报和周报；统计每类人员各种健康状态 人数的日报和周报；统计每周居住地发生变化的人数。

需要设计完成的任务有以下三项： 1) 设计出此管理模块所需的类，可用 UML 类图说明所设计的类间关系，并请简要说明你给出的类设 第 5 页 共 5 页 计方案的理由和优缺点。 2) 请用规范的 C++类定义语法，写出上述所设计类的定义语句。根据所需情况写出类的数据成员、函 数成员、构造函数和析构函数的声明，以及成员访问权限。所有函数的实现（函数体）语句不必写出来。3) 设计出能完成上述信息登记、统计分析等业务功能所需的函数。你的设计是把这些函数定义为类的成员函数，还是非成员函数呢？请简要说明你的设计理由。对所设计的这些函数请写出函数的声明语句，所有函数的实现（函数体）语句不必写出来，但请注 释说明函数功能、形式参数的含义和类型、返回值类型等必要信息。




```
class Member
{
protected:
    string number, name, phoneNumber, IDNumber;
    bool gender;
    int idType;
    string country, province, city, community, detailedAddr;
    bool isAddrChanged;
    int time, temperature;
    vector<int> state;
public:
    Member(){}
    void setInfo(string num, string name, string pNum, string IDNum, bool gender, int idType);
    void setAddr(string country, string province, string city, string community, string detailedAddr);
    void setTemperature(int time, int temperature, int stateNum, int i);
    friend void totalHealthState(Member p[], int size, int i);
    friend int addrChangedNumber(Member p[]);
};
class Student :public Member{};
void totalHealthState(Member p[], int size, int i);
int addrChangedNumber(Member p[]);
```

第 3 问，信息登记设置为成员函数，统计设置为非成员函数（但设为友元函数），是一种便于实现的方法，每次对给定群体调用这 个非成员函数即可得到结果 第 1 问 学生一个类吧，另外两个老师和管理人员就按照那个图来。具体的研究生和本科生就在构造对象时分开吧 可以把硕士归为一个类。 这样的优点：结构清晰，便于管理，而且长远考虑，具有更强的可扩展性，比如本科生类可能以后会增加字段，函数等等。缺 点：代码更繁杂，对于当前需求来说是溢出的。