尚德机构
SUNLANDS

密训·资料

C++程序设计（全国）

1904

MI XUN ZI LIAO

编前语

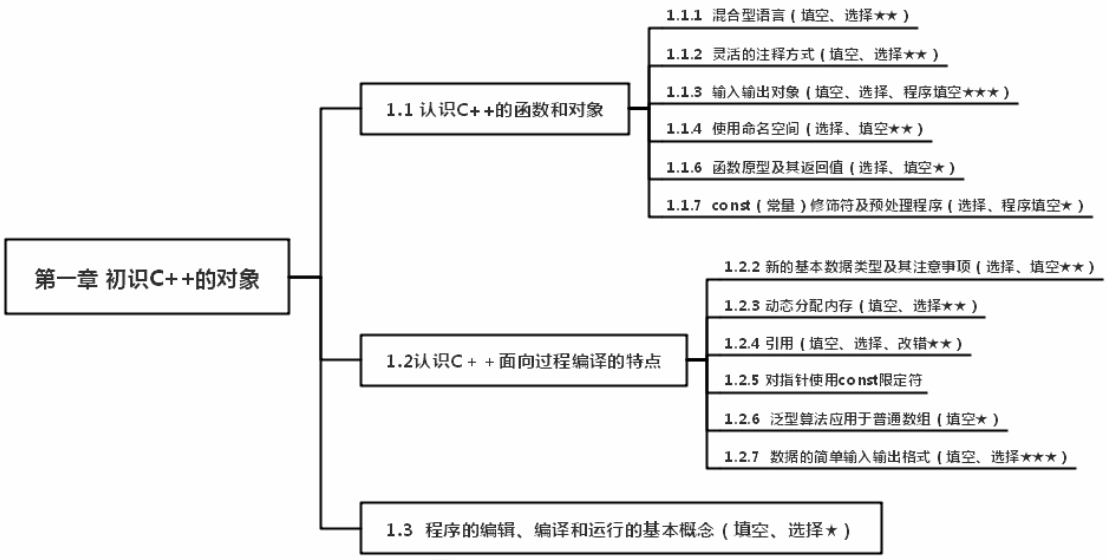
试卷说明：本科目考试题型为单选、填空、程序改错、程序填空、程序分析和程序设计题。各题型分值分布情况如下表所示：

题型	分值
单选题	1 分×20 道=20 分
填空题	1 分×20 道=20 分
程序改错题	4 分×5 道=20 分
程序填空题	4 分×5 道=20 分
运行结果题（分析题）	5 分×2 道=10 分
程序设计题	10 分×1 道=10 分

资料说明：本资料提供的都是考试重点，但是在重点中，按照知识点的重要程度，对知识点进行了标星。星标数目越多，表示知识点越重要，越可能考到。

目录

第一章 初识 C++的对象



1.1 认识 C++的函数和对象

1.1.0 认识 C++的函数和对象（填空、选择★★）	C++程序从主函数开始执行。 字符串中：\n 是转义符，表示回车符。\t 是转义符，表示跳格符。
1.1.1 混合型语言（填空、选择★★）	C++以 .cpp 为文件扩展名，有且只有一个名为 main 的主函数，因保留了这个面向过程的主函数，所以被称为混合型语言。
1.1.2 灵活的注释方式（填空、选择★★）	1.“/*”开始，到“*/”结束 2.从“//”开始到本行结束
1.1.3 输入输出对象（填空、选择、程序填空★★★）	1.用提取操作符“>>”从 cin 输入流中提取字符插入操作 2.用插入操作符“<<”向 cout 输出流中插入字符 3. 使用标准输入 cin 及标准输出 cout 前，要在主函数前使用 #include <iostream> C++标准输入输出库头文件 iostream 将其包括。 4.换行操作：用语句 cout<<endl；或 cout<<“\n”；实现，其中 endl 可以插在流的中间。
1.1.4 使用命名空间（选择、填空★★）	一般的程序都要具有下面的两条语句： #include <iostream> //包含头文件 using namespace std； //使用命名空间
1.1.6 函数原型及其返回值（选择、填空★）	1. C++使用变量和函数的基本规则都是：先声明，后使用。变量有时也可边声明边使用，但必须声明，否则出错。 2. 除构造函数与析构函数外，函数都需要有类型声明。如果函数确实不需要返回值，还可用 void 标识，一旦使用 void 标识，函数

	体内就不再需要使用 return 语句，否则会编译出错，但可使用 return；语句。 3. C＋＋函数有库函数（标准函数，引用时函数名外加＜ ＞）和自定义函数（引用时函数名外加 “ ” ）两类。
1.1.7 const(常量) 修饰符及预处理程序（选择、程序填空★）	const 修饰符：用于定义符号常量。C 中一般使用宏定义“＃define”定义常量，而 C＋＋中除此外，建议使用 const 代替宏定义，用关键字 const 修饰的标识符称为符号常量。

1.2 认识 C＋＋面向过程编译的特点

1.2.2 新的基本数据类型及其注意事项（选择、填空★★）		1.C++语言还比 C 语言多了 bool(布尔)型 2. C＋＋中的常量分三种，第一种为符号常量；第二种为整数常量，有 4 种类型，分别为十进制、长整型（后缀 L（l））、八进制(前缀 O)、十六进制(前缀 OX)，并用前缀和后缀进行分类标识；第三种为浮点常量，有三种类型，分别为 float 型、long float 型、double 型，并用后缀进行分类识别。double 精度高，有效数字 16 位，float 精度 7 位。 3.&&：与运算，两个条件为真时，结果为真。||或运算，有一个为真，结构为真。
1.2.3 动态分配内存(填空、选择★★)		在使用指针时，如果不使用对象地址初始化指针，可以自己给它分配地址。 对于只存储一个基本类型数据的指针，申请方式如下： new 类型名[size] //申请可以存储 size 个该数据类型的对象 不再使用时，必须使用 delete 指针名；来释放已经申请
1.2.4 引用（填空、选择、改错★★）		引用就是为现有的对象起个别名，别名的地址与引用对象的地址是一样的。 引用的声明方式为： 数据类型 & 别名＝对象名；，注意对象在引用前必须先初始化，另外声明中符号“&”的位置无关紧要
1.2.5 对指针使用 const 限定符	1.2.5.1 左值和右值（改错、选择、填空★★）	左值是指某个对象的表达式，必须是可变的。
	1.2.5.2 指向常量的指针（选择、填空、改错★）	指向常量的指针是在非常量指针声明前面使用 const，如：const int *p；
	1.2.5.4 指向常量的常量指针 （选择★）	可以声明指针本身和所指向的对象都不能改变的“指向常量的常量指针”，这时必须要初始化指针。如：int x＝2； const int * const p＝&x；

<table>
<tr><td>1.2.6　泛型算法应用于普通数组（填空★）</td><td>对数组内容进行升幂、输出、反转和复制等操作需要包含头文件<algorithm> ;对数组内容进行降幂和检索等操作需要包含头文件<functional>。
关键字：
反转：reverse
复制：copy，reverse_copy（逆向复制）
排序：sort（默认升幂，尾部加 greater<Type>() 为降幂）
检索：find
输出：copy(尾部必须加 ostream_iterator<Type>(cout,"字符串"))</td></tr>
<tr><td>1.2.7　数据的简单输入输出格式（填空、选择★★★）</td><td>
<table>
<tr><th>格式</th><th>含义</th><th>作用</th></tr>
<tr><td>dec</td><td>设置转换基数为十进制</td><td>输入/输出</td></tr>
<tr><td>oct</td><td>设置转换基数为八进制</td><td>输入/输出</td></tr>
<tr><td>hex</td><td>设置转换技术为十六进制</td><td>输入/输出</td></tr>
<tr><td>endl</td><td>输出一个换行符并刷新流</td><td>输出</td></tr>
<tr><td>Setprecision（n）</td><td>设置浮点数输出精度 n</td><td>输出</td></tr>
<tr><td>Setw（n）</td><td>设置输出数据字段宽度</td><td>输出</td></tr>
<tr><td>Setfill（'字符'）</td><td>设置 ch 为填充字符</td><td>输出</td></tr>
<tr><td>Setiosflags（flag）</td><td>设置 flag 指定的标志位</td><td>输出</td></tr>
<tr><td>resetiosflags（flag）</td><td>清除 flag 指定的标志位</td><td>输出</td></tr>
</table>
<table>
<tr><th>常量名</th><th>含义</th></tr>
<tr><td>ios_base::left</td><td>输出数据按输出域左边对齐输出</td></tr>
<tr><td>ios_base::right</td><td>输出数据按输出域右边对齐输出</td></tr>
<tr><td>ios_base::showpos</td><td>在正数前添加一个“+”号</td></tr>
<tr><td>ios_base::showpoint</td><td>浮点输出时必须带有一个小数点</td></tr>
<tr><td>ios_base::scientific</td><td>使用科学计数法表示浮点数</td></tr>
<tr><td>ios_base::fixed</td><td>使用定点形式表示浮点数</td></tr>
</table>
</td></tr>
</table>

1.3　程序的编辑、编译和运行的基本概念（填空、选择★）

程序的编辑、编译和运行的基本概念（填空、选择★）	掌握 Vistual C++的工程和文件的产生方法。1.编辑 2.编译 3.链接 4.执行 5.调试
	掌握程序的编辑、编译和运行的基本方法。1.创建项目 2.创建文件 3.程序的编辑、编译和运行。
	Windows 环境下 ,由 C++源程序文件编译而成的目标文件的扩展名是 obj。

【题目演练】

【单选题】

1.对使用关键字 new 所开辟的动态存储空间，释放时必须使用（　）

A:free

B:create

C:delete

D:realse

【解析】使用 delete 关键字释放空间。

2. 编写 C++程序一般需经过的几个步骤依次是（ ）

A:编辑、调试、编译、连接

B:编译、调试、编辑、连接

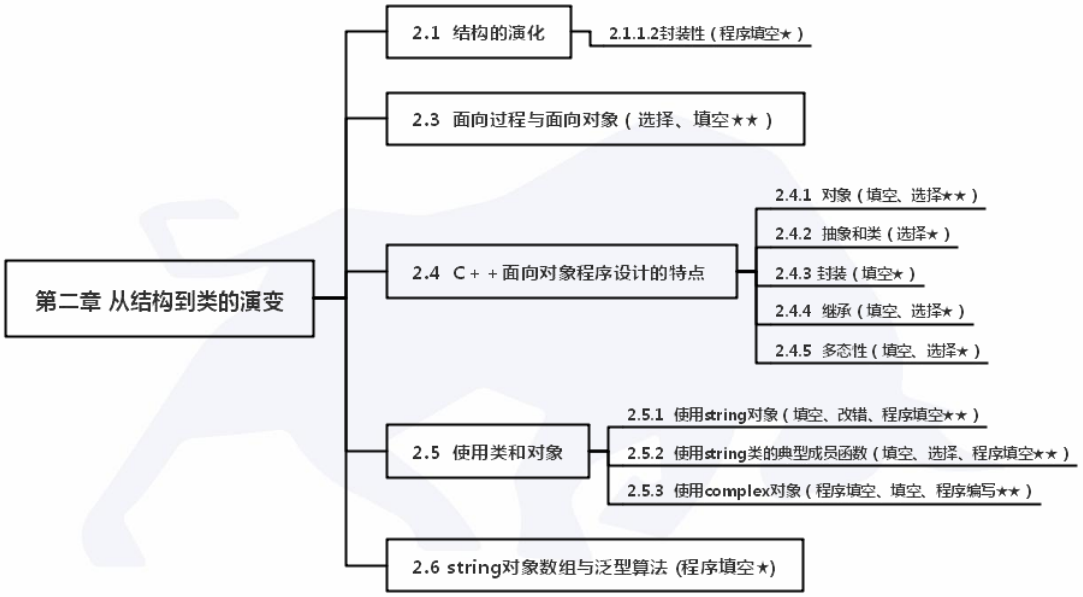
C:编译、编辑、连接、运行

D:编辑、编译、连接、运行

【正确答案】D

【解析】用 C++语言写成的程序称为源程序，源程序必须经过 C++编译程序翻译成机器语言才能执行。一般需要经过编辑、编译、连接、运行

第二章 从结构到类的演变



2.3 面向过程与面向对象（选择、填空★★）

2.3 面向过程与面向对象（选择、填空★★）	面向过程，就是不必了解计算机的内部逻辑，而把精力主要集中在对如何求解问题的算法逻辑和过程的描述上，通过编写程序把解决问题的步骤告诉计算机
	面向对象：是通过数据和代码建立分块的内存区域，以便提供对程序进行模块化的一种程序设计方法，这些模块可以被用作样板-类，在需要时将其实例化成对象

2.4 C++面向对象程序设计的特点

2.4.1 对象（填空、选择★★）	类实际上是一种抽象机制，它描述了一类问题的共同属性和行为。在 C++中，类的对象就是该类的某一特定实体（也称实例）
2.4.2 抽象和类（选择★）	抽象是一种从一般的观点看待事物的方法，即集中于事物的本质特征，而不是具体细节或具体实现。 类是某些对象共同特征的表示。类是创建对象的模板，对象是类的

	实例
2.4.3 封装（填空★）	所谓“封装”，就是把对象的属性和操作结合成一个独立的系统单位，并尽可能隐蔽对象的内部细节。
2.4.4 继承（填空、选择★）	继承是一个类可以获得另一个类的特性的机制，支持层次概念，通过继承，低层的类只需定义特定于它的特征，而共享高层的类中的特征。
2.4.5 多态性（填空、选择★）	不同的对象可以调用相同名称的函数，但可导致完全不同的行为的现象称为多态性。利用多态性，程序中只需进行一般形式的函数调用，函数的实现细节留给接受函数调用的对象，这大大提高了解决人们复杂问题的能力。

2.5 使用类和对象

2.5.1 使用 string 对象（填空、改错、程序填空★★）	在程序中可以使用 string 类定义存储字符串的对象，这些对象属于 string 类，因此还要使用#include <string>来包含这个类的头文件。
2.5.2 使用 string 类的典型成员函数（填空、选择、程序填空★★）	string 对象是通过调用成员函数实现操作，从而提供对象的行为或消息传递的，对象调用成员函数的语法为： 对象名称.成员函数（参数（可供选择的消息内容）） 常用的 string 类成员函数： substr： string newstr = str1.substr(3,3); find： int i = str1.find(“are”,0); getline： getline（cin，InputLline，’\n’）； swap： str1.swap(str2); = str2.swap(str1); begin 和 end：copy(str1.begin( ),str1.end( ),str2.begin( ));
2.5.3 使用 complex 对象（程序填空、填空、程序编写★★）	C++标准库提供 complex 类定义复数对象，在程序中包含这个类的头文件为：#include <complex> 复数(complex number)类需要两个初始值 :实部和虚部 ,complex 是一个模板类，利用构造函数初始化的格式为： complex <数据类型> 对象名（实部值，虚部值）；

2.6 string 对象数组与泛型算法（程序填空★）

2.6 string 对象数组与泛型算法（程序填空★）	泛型算法同样适合 string 类，但要注意不要将该节介绍的 find 函数与 string 本身的 find 函数混淆。 另外，string 类还有一个 swap 成员函数，用来交换两个对象的属性。假设有两个 string 类的对象 str1 和 str2，下面两种调用方式是等效的：str1.swap(str2);str2.swap(str1);

【题目演练】

【单选题】

1.不同对象调用同名函数，但导致完全不同行为的现象称为( )

A:抽象　　B:封装　　C:继承　　D:多态性

【正确答案】D

【解析】不同对象可以调用相同名称的函数，但可导致完全不同的行为的现象称为多态性

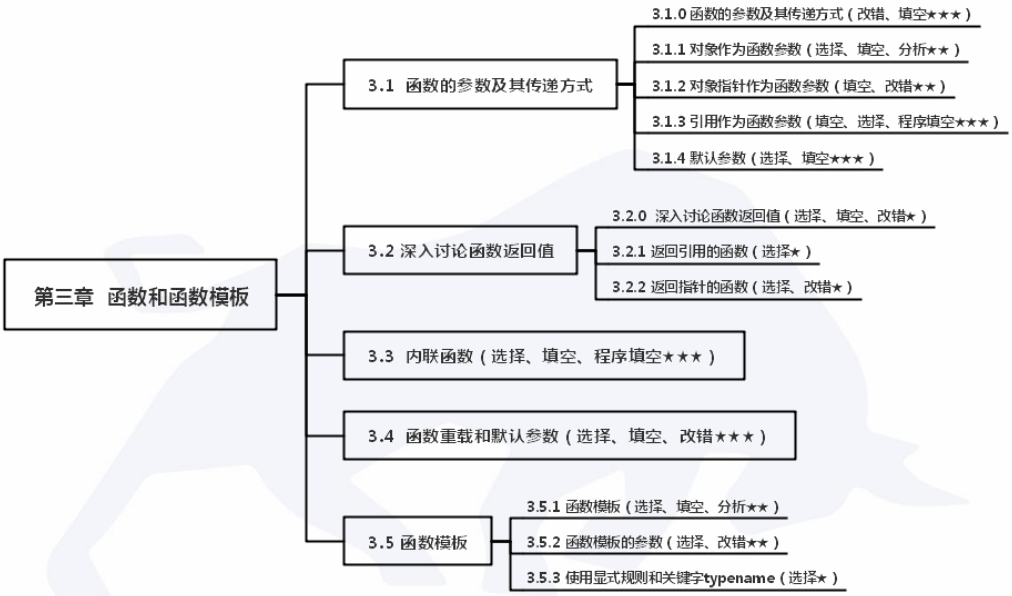
2.在 C++中，类与类之间的继承关系具有（ ）。

A:自反性　　B:对称性　　C:传递性　　D:反对称性

【正确答案】C

【解析】通过继承，低层的类只须定义特定于它的特征，而共享高层的类中的特征，即 C 选项的传递性

第三章　函数和函数模板



3.1　函数的参数及其传递方式

3.1.0 函数的参数及其传递方式（改错、填空★★★）	C 语言函数参数的传递方式只有传值一种 ,又分为传变量值和传变量地址值两种情况，而 C + + 的函数参数传递方式有两种：第一种和 C 语言一样，是传值；第二种是传引用，即传对象的地址，所以也称传地址方式。注意传地址值传递的是值，是以对象指针作为参数；而传地址传递的是地址，是以对象引用作为参数。
3.1.1 对象作为函数参数（选择、填空、分析★★）	使用对象作为函数参数，是将实参对象的值传递给形参对象，传递是单向的，形参具有实参的备份，当在函数中改变形参的值时，改变的是这个备份中的值，不影响原来实参的值。
3.1.2 对象指针作为函数参数（填空、改错★★）	将指向对象的指针作为函数参数，形参是对象指针，实参是对象的地址值。虽然参数传递方式仍然是传值方式，但因为形参传递的就是实参本身，所以当在函数中改变形参的值时，改变的就是原来实参的值。

3.1.3 引用作为函数参数（填空、选择、程序填空★★★）	使用引用作为函数参数，在函数调用时，实参对象名传给形参对象名，形参对象名就成为实参对象名的别名。实参对象和形参对象代表同一个对象，所以改变形参对象的值就是改变实参对象的值。
3.1.4 默认参数（选择、填空★★★）	默认参数就是不要求程序员设定该参数， 而由编译器在需要时给该参数赋默认值。当程序员需要传递特殊值时，必须显式的指明。默认参数必须在函数原型中说明，默认参数可以多于 1 个，但必须放在参数序列的后部。

3.2 深入讨论函数返回值

3.2.0 深入讨论函数返回值（选择、填空、改错★）	c++函数的返回值类型可以是除数组和函数以外的任何类型，非 void 类型的函数必须向调用者返回一个值，数组只能返回地址。
3.2.1 返回引用的函数（选择★）	函数可以返回一个引用，这样的目的是为了将该函数用在赋值运算符的左边，因为其他情况下，一个函数是不能直接用在赋值运算符左边的。 返回引用的函数原型的声明方式为： 数据类型 & 函数名（参数列表）；
3.2.2 返回指针的函数（选择、改错★）	指针函数：返回值是存储某种类型数据的内存地址的函数。 返回指针的函数原型的声明方式为： 数据类型 *函数名（参数列表）；

3.3 内联函数（选择、填空、程序填空★★★）

3.3 内联函数（选择、填空、程序填空★★★）	使用关键字 inline 说明的函数称为内联函数，内联函数必须在程序中第一次调用此函数的语句出现之前定义，在 C++中，除具有循环语句、switch 语句的函数不能说明为内联函数外，其他函数都可以说明为内联函数。使用内联函数可以提高程序执行速度，但如果函数体语句多，则会增加程序代码的大小。

3.4 函数重载和默认参数（选择、填空、改错★★★）

3.4 函数重载和默认参数(选择、填空、改错★★★)	深刻理解函数重载的概念和方法。	函数重载可使一个函数名具有多种功能，即具有“多种形态”，称这种特性为多态性。从函数原型可见，它们的区别一是参数类型不同，二是参数个数不同，所以仅凭返回值不同或仅凭参数个数不同，均不能区分重载函数。在函数重载时，源代码只指明函数调用，而不说明调用具体调用哪个函数，直到程序运行时才确定调用哪个函数，编译器的这种连接方式称为动态联编或迟后联编。
	熟练掌握构造函数默认参数	函数在定义时可以预先声明默认的形参值。调用时如果给出实参，则用实参初始化形参；如果没有给出实参，则采用预先声明的默认形参值

	的方法。	

3.5 函数模板

3.5.1 函数模板（选择、填空、分析★★）	C++中规定模板以关键字 template 和一个形参表开头，形参表中以 class 表示“用户定义的或固有的类型”，一般选用 T 作为标识符来标识类型参数。 由于函数在设计时没有使用实际的类型，而是使用虚拟的类型参数，故其灵活性得到加强。当用实际的类型来实例化这种函数时，就好像按照模板来制造新的函数一样，所以称这种函数为函数模板。将函数模板与某个具体数据类型连用，就产生了模板函数，又称这个过程为函数模板实例化，这种形式就是类型参数化。
3.5.2 函数模板的参数（选择、改错★★）	显示比较准则形式为：函数模板名<模板参数> （参数列表） 每次调用都显式的给出比较准则，会使人厌烦，一般可使用下面的默认方式：函数模板名（参数列表）
3.5.3 使用显式规则和关键字 typename（选择★）	C + + 中，关键字 typename 仅用于模板中，其用途之一就是代替类模板 template 参数列表中的关键字 class。

【题目演练】

1. 一个函数功能不太复杂，但要求被频繁调用，应选用（ ）。

A:内联函数　　B:重载函数　　C:递归函数　　D:嵌套函数

【正确答案】A

【解析】使用内联函数能加快程序执行速度，但如果函数体语句多，则会增加程序代码的大小。当一个函数功能不太复杂，但要求被频繁调用，应选用它。

2.函数模板 template<typename T>void Func(T，T)不能具有哪种实例化形式（ ）。

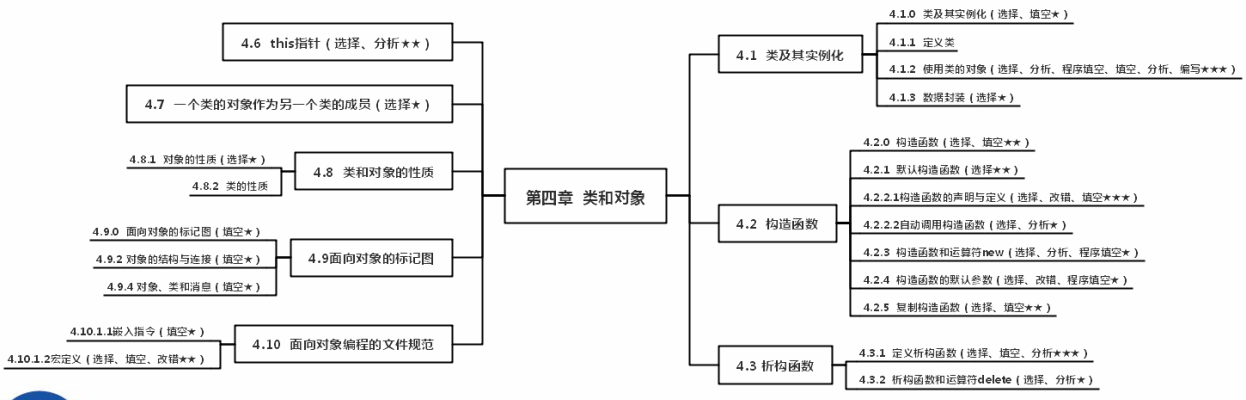
A:void Func(int，int)　　B:void Func(bool，bool)

C:void Func(double，int)　　D:void Func(char，char)

【正确答案】C

【解析】template<typename T>void Func(T，T)需保持两类型一致。因此选择 C 选项。

第四章　类和对象



4.1 类及其实例化

<table>
<tr><td colspan="2">4.1.0 类及其实例化（选择、填空★）</td><td>对象就是一类物体的实例，将一组对象的共同特征抽象出来，从而形成“类”的概念。</td></tr>
<tr><td rowspan="3">4.1.1 定义类</td><td>4.1.1.1 声明类（选择、填空、改错、程序填空、编写★★★★）</td><td>C++中声明类的一般形式为：
class 类名{
private：
私有数据和函数
public：
公有数据和函数
protected：
保护数据和函数
};</td></tr>
<tr><td>4.1.1.2 定义成员函数（选择、程序填空★★）</td><td>定义成员函数的一般形式为：
返回类型 类名：：成员函数名（形参列表）
{
成员函数的函数体 //内部实现
}</td></tr>
<tr><td>4.1.1.3 数据成员的赋值和初始化（改错★）</td><td>不能在类体内或类体外面给数据成员赋值，当然在类体外面就更不允许了。数据成员的具体值是用来描述对象的属性的。只有产生了一个具体的对象。这些数据值才有意义。</td></tr>
<tr><td colspan="2" rowspan="2">4.1.2 使用类的对象（选择、分析、程序填空、填空、分析、编写★★★）</td><td>对象的定义格式为：类名 对象名(参数)；</td></tr>
<tr><td>对象和引用使用'.'运算符访问对象的成员，而指针使用"->"运算符访问对象的成员</td></tr>
<tr><td colspan="2">4.1.3 数据封装（选择★）</td><td>封装就是将抽象得到的数据和行为（或功能）相结合，形成一个有机的整体，也就是将数据与操作数据的函数代码进行有机地结合，形成“类”，其中的数据和函数都是类的成员</td></tr>
</table>

4.2 构造函数

<table>
<tr><td colspan="2">4.2.0 构造函数（选择、填空★★）</td><td>建立一个对象时，对象的状态（数据成员的取值）是不确定的。为了使对象的状态确定，必须对其进行正确的初始化。C++有称为构造函数的特殊成员函数，它可自动进行对象的初始化。</td></tr>
<tr><td colspan="2">4.2.1 默认构造函数（选择★★）</td><td>当没有为一个类定义任何构造函数的情况下，C++编译器会自动建立一个不带参数的构造函数。
如果程序中定义了有参数的构造函数，又存在需要先建立对象数组后进行赋值操作的情况，则必须为它定义一个无参数的构造函数。</td></tr>
</table>

4.2.2.1 构造函数的声明与定义（选择、改错、填空★★★）	假设数据成员为 x1、x2、...xn，则有以下两种形式：
	类名：：类名（形参 1，形参 2,...形参 n）：x1（形参 1），x2（形参 2）,...xn（形参 n）{ }
	类名：：类名（形参 1，形参 2，...形参 n）
	{
	x1 = 形参 1；
	x2 = 形参 2；
	.....
	xn=形参 n；
	}
	构造函数的参数在排列时无顺序要求，只要保证相互对应即可。
	假设类 F 的对象 f 是类 A 的成员对象，则“A a”语句执行时，先调用类 F 的构造函数。

4.2.2.2 自动调用构造函数（选择、分析★）	程序员不能在程序中显式地调用构造函数，构造函数是自动调用的。可以设计多个构造函数，编译系统根据对象产生的方法自动调用相应的构造函数，构造函数将在产生对象的同时初始化对象。
4.2.3 构造函数和运算符 new（选择、分析、程序填空★）	1.运算符 new 用于建立生存期可控的动态对象，new 返回这个对象的指针,使用运算符 new 建立的动态对象只能用运算符 delete 删除，以便释放所占空间。 2.当使用 new 建立一个动态对象时，new 首先分配足以保存 Point 类的一个对象所需要的内存，然后自动调用**构造函数**来初始化这块内存，再返回这个动态对象的地址。
4.2.4 构造函数的默认参数（选择、改错、程序填空★）	如果程序定义了有参数构造函数，又想使用无参数形式的构造函数，解决的方法是将相应的构造函数全部使用默认参数设计。
4.2.5 复制构造函数（选择、填空★★）	复制构造函数的作用，就是通过拷贝方式使用一个类已有的对象来建立该类的一个新对象，所以又直译为拷贝构造函数。通常情况下，由编译器建立一个默认复制构造函数 ,其原形为 :类名：:类名(const 类名&），或不加 const 修饰如构造函数一样，复制构造函数也可自定义，如果自定义了，则编译器只调用自定义的复制构造函数，而不再调用默认的复制构造函数。拷贝构造函数使用**引用**作为参数初始化创建中的对象。

4.3 析构函数

4.3.1 定义析构函数（选择、填空、分析★★★）	①析构函数在类体里的声明形式：~类名（ ）； ②析构函数的定义形式：类名：：~类名（ ）{ }。 析构函数没有参数,也没有返回值，而且不能重载。因此在一个类中只能有一个析构函数。

4.3.2 析构函数和运算符 delete（选择、分析★）	运算符 delete 与析构函数一起工作，当使用运算符 delete 删除一个动态对象时，它首先为这个对象调用析构函数，然后再释放这个动态对象占用的内存，这和使用运算符 new 建立动态对象的过程刚好相反。

4.6 this 指针（选择、分析★★）

4.6 this 指针（选择、分析★★）	C++规定，当一个成员函数被调用时，系统将自动向它传递一个隐含的参数，该参数是一个指向调用该函数的对象的指针，名为 this 指针，从而使成员函数知道该对哪个对象进行操作。
	this 指针是 C++实现封装的一种机制，它将对象和该对象调用的成员函数连接在一起，从而在外部看来，每个对象都拥有自己的成员函数。当一个成员函数被调用时，该成员函数的 **this 指针**指向调用它的对象。

4.7 一个类的对象作为另一个类的成员（选择★）

4.7 一个类的对象作为另一个类的成员（选择★）	因为类本身就是一种新的数据类型，所以一个类的对象可以作为另一个类的成员。例如有 A、B 两个类，可以通过在 B 类里定义 A 的对象作为 B 的数据成员，或者定义一个返回类型为 A 的函数作为 B 的成员函数。

4.8 类和对象的性质

4.8.1 对象的性质（选择★）		1、同一类的对象之间可以相互赋值 2、可以使用对象数组 3、可以使用指向对象的指针 4、对象可以用作函数参数 5、对象作为函数参数时，可以使用对象、对象引用和对象指针三种方式 6、一个对象可以作为另一个类的成员。
4.8.2 类的性质	4.8.2.1 使用类的权限(改错、选择★)	1、类本身的成员函数可以使用类的所有成员（私有和公有成员）。2、类的对象只能访问公有成员函数。3、其他函数不能使用类的私有成员，也不能使用公有成员函数，它们只能通过定义类的对象为自己的数据成员，然后通过类的对象使用类的公有成员函数。4、虽然一个类可以包含另外一个类的对象，但这个类也只能通过被包含的类的对象使用那个类的成员函数，通过成员函数使用数据成员。
	4.8.2.2 不完全的类的声明（选择★）	类不是内存中的物理实体，只有当使用类产生对象时，才进行内存分配，这种对象建立的过程称为实例化。不完全声明的类不能实例化，否则会编译出错；不完全声明仅用于类和结构，企图存取没有完全声明的类成员，也会引起编译错误。

4.9 面向对象的标记图

4.9.0 面向对象的标记图（填空★）	UML 语言是一种典型的面向对象建模语言，主要用于面向对象分析和建模。在 UML 语言中用符号描述概念，概念间的关系描述为连接符号的线

4.9.2 对象的结构与连接	4.9.2.2 对象组成关系及其表示(填空★)	组成关系说明的是整体与部分的关系，C++中最简单的是包含关系，如线段由两个点组成。
	4.9.2.3 实例连接及其表示（填空★）	实例连接反映对象之间的静态关系，例如车和驾驶员的关系，实例连接有一对一、一对多和多对多 3 种连接方式，用一条实线表示连接关系。简单的实例连接是对象实例之间的一种二元关系。
	4.9.2.4 消息连接及其表示（填空★）	消息连接描述对象之间的动态关系，即若一个对象在执行自己的操作时，需要通过消息请求另一个对象为它完成某种服务，则说第 1 个对象与第 2 个对象之间存在着消息连接。消息连接是有方向的，使用一条带箭头的实线表示，从消息的发送者指向消息的接收者。
4.9.4 对象、类和消息(填空★)		对象的属性和行为是对象定义的组成要素，分别代表了对象的静态和动态特征，无论对象是简单的或是复杂的，对象一般具有以下特征：①有一个状态，由与其相关联的属性值集合所表征；②有唯一标识名，可以区别于其他对象；③有一组操作方法，每个操作决定对象的一种行为；④对象的状态只能被自己的行为所改变；⑤对象的操作包括自身操作（施加于自身）和施加于其他对象的操作；⑥对象之间以消息传递的方式进行通信；⑦一个对象的成员仍可以是一个对象。
		消息是向对象发出的服务请求，它是面向对象系统中实现对象间的通信和请求任务的操作。消息传递是系统构成的基本元素，是程序运行的基本处理活动。
		对象传送的消息一般由 3 部分组成：接收对象名、调用操作名和必要的参数。

4.10 面向对象编程的文件规范

4.10.1.1 嵌入指令（填空★）	嵌入指令#include 指示编译器将一个源文件嵌入到该指令所在的位置。尖括号或双引号中的文件名可含有路径信息。
4.10.1.2 宏定义（选择、填空、改错★★）	# define 指令定义一个标识符及串，在源程序中每次遇到该标识符时，编译器将自动用后面的串代替它。该标识符称为宏名，替换过程称为宏替换。宏定义的一般形式为：# define 宏名 替换正文。

【题目演练】

【单选题】

1.用运算符 delete 删除一个动态对象时（ ）。

A:首先为该动态对象调用构造函数，再释放其占用的内存　　B:首先释放该动态对象占用的内存，再为其调用构造函数

C:首先为该动态对象调用析构函数,再释放其占用的内存　　D:首先释放该动态对象占用的内存，再为其调用析构函数

【正确答案】C

【解析】运算符 delete 与析构函数一起工作。当使用运算符 delete 删除一个动态对象时，它首先为这个动态对象调用析构函数,然后再释放这个动态对象占用的内存,这和使用 new 建立动态对象的过程正好相反。因此选择 C 选项

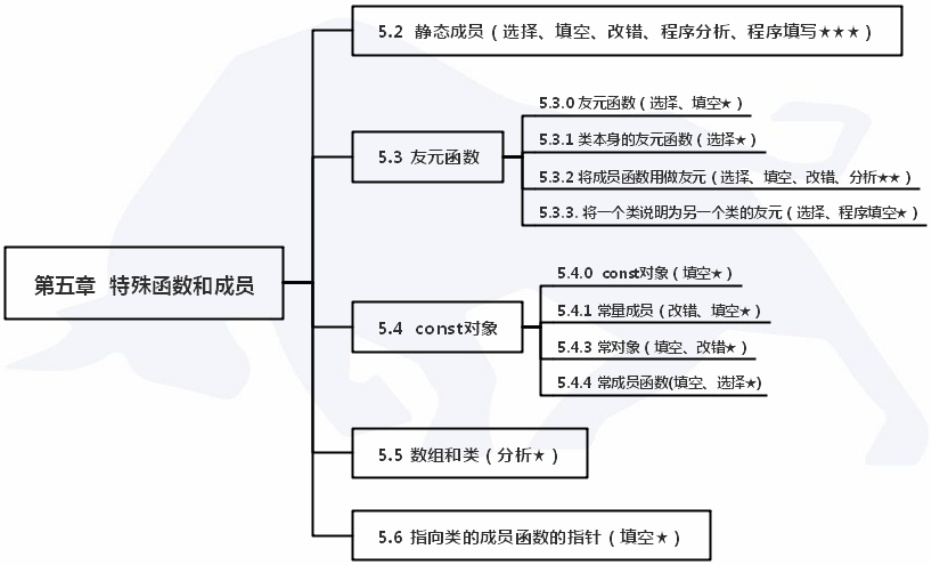
2.在定义类成员时，为产生封装性，则需使用哪个关键字（ ）。

A:public　　B:publish　　C:protected　　D:private

【正确答案】D

【解析】private 限制了类之外的函数的访问权。只要将数据成员和成员函数使用 private 限定，就设定了一道防线，也就是封装性。

第五章　特殊函数和成员



5.2　静态成员（选择、填空、改错、程序分析、程序填写★★★）

5.2　静态成员(选择、填空、改错、程序分析、程序填写★★★）	如果类的数据成员或成员函数使用关键字 static 修饰，这样的成员称为静态数据成员或静态成员函数,统称为静态成员。静态成员函数没有 this 指针。
	静态数据成员必须进行初始化，其一般格式为：数据类型 类名：：静态数据成员名=初始值；
	对静态数据成员初始化时应注意问题： 1.静态数据成员只能说明一次,如果在类中仅是对静态数据成员进行声明，则必须在文件作用域的某个地方对静态数据成员进行定义并初始化，则应在类**外**对静态数据成员进行初始化 2.在类体外对静态成员初始化时不使用 static 关键字

	3.由于静态数据成员是类的成员，因此在初始化必须使用作用域运算符"::"限定它所属的类

5.3 友元函数

5.3.0 友元函数（选择、填空★）	友元函数不是当前类的成员函数，而是独立于当前类的外部函数（包括普通函数和其他类的成员函数），但它可以访问该类的所有成员，包括私有成员、保护成员和公有成员
5.3.1 类本身的友元函数（选择★）	友元函数要在定义时声明，声明时要在其函数名前加上关键字 friend。该声明可以放在公有部分，也可以放在私有部分或是保护部分。友元函数的定义通常在类外部
5.3.2 将成员函数用做友元（选择、填空、改错、分析★★）	除了可以将普通函数声明为类的友元函数以外，也可以将另外一个类的成员函数声明为一个类的友元函数，使用时要通过相应的类和对象名进行访问
5.3.3. 将一个类说明为另一个类的友元（选择、程序填空★）	可以将一个类 One 说明为另一个类 Two 的友元，这时，整个类 One 的成员函数均是类 Two 的友元函数，声明语句简化为：“friend class 类名；”。

5.4 const 对象

5.4.0 const对象（填空★）	可以在类中使用 const 关键字定义数据成员和成员函数或修饰一个对象。一个 const 对象只能访问 const 成员函数，否则将产生编译错误。
5.4.1 常量成员（改错、填空★）	常量成员包括静态常数据成员、常数据成员和常引用，其中前者仍保留静态成员特征，需要在类外初始化，后两者则只能通过初始化列表来获得初值。
5.4.3 常对象（填空、改错★）	在对象名前使用 const 声明的对象就是常对象，常对象必须在声明的同时进行初始化，而且不能被更新，形式为： 类名 const 对象名（参数表）； //必须在声明的同时进行初始化
5.4.4 常成员函数(填空、选择★)	声明常成员函数的格式：类型标识符 函数名（参数列表） const； 定义格式：类型标识符 类名::函数名（参数列表） const {……//函数体} 在类中用内联函数定义 const 函数：类型标识符 函数名（参数列表）const {……//函数体}

5.5 数组和类（分析★）

5.5 数组和类（分析★）	类的引入产生了除 C 语言可以声明的数组类型外的一系列新的数组类型，简单介绍类对象数组和类对象指针数组。ANSI C 可以声明的数字类型都适用于 C++，但类的引入产生了一系列新的数组类型。

5.6 指向类的成员函数的指针（填空★）

5.6 指向类的成员函数的指针（填空★）	C++既包含指向类数据成员的指针，又包含指向成员函数的指针。它提供一种特殊的指针类型，指针指向类的成员，而不是指向该类的一个对象中该成员的一个实例，这种指针称为指向类成员的指针。

【题目演练】

【单选题】

1.设类 A 中有静态数据成员 x，两个 A 类对象 a 和 b，若 a．x=10，则 b．x 的值为（ ）。

A:9　　B:10　　C:11　　D:不能确定

【正确答案】B

【解析】使用关键字 static 进行修饰，这样的成员称为静态数据成员或静态成员函数。没有建立对象，但静态成员已经存在。因此 b.x 为 10。

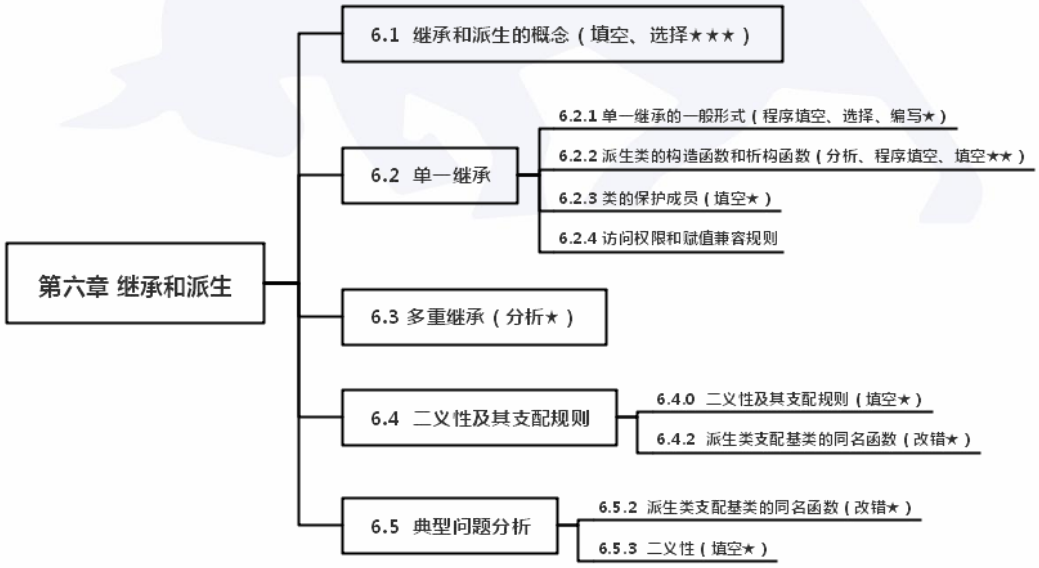
2.下列哪个函数不是类的成员函数（ ）。

A:构造函数　　B:析构函数　　C:友元函数　　D:拷贝构造函数

【正确答案】C

【解析】友元函数其实就是一般普通的函数，仅有的不同点是：它在类中说明，可以访问该类所有对象的私有成员。一个类的成员函数包括构造函数(包括拷贝构造函数)和析构函数。因此 C 选项不是类的成员函数。

第六章 继承和派生



6.1 继承和派生的概念（填空、选择★★★）

6.1 继承和派生的概念（填空、选择★★★）	类的派生是指从一个或多个以前定义的类产生新类的过程，原有的类称为基类，新产生的类称为派生类，派生类继承了基类所有的数据成员和成员函数。 派生类可以有如下几种变化：增加新的成员(数据成员或成员函数)；重新定义已有的成员函数；改变基类成员的访问权限。

<table>
<tr><td colspan="2"></td><td>C++中有两种继承方式：单一继承和多重继承，对于前者，派生类只能有一个基类；对于后者，派生类可以有多个基类。</td></tr>
<tr><td colspan="3">6.2 单一继承</td></tr>
<tr><td colspan="2">6.2.1 单一继承的一般形式（程序填空、选择、编写★）</td><td>class 派生类名:访问控制 基类名{
private:
成员声明列表
protected:
成员声明列表
public:
成员声明列表
}</td></tr>
<tr><td colspan="2" rowspan="2">6.2.2 派生类的构造函数和析构函数（分析、程序填空、填空★★）</td><td>定义派生类构造函数的一般定义形式：
派生类名::派生类名（参数表 0）:基类名（参数表）{..... //函数体}</td></tr>
<tr><td>派生类析构函数的一般定义形式：派生类名::~派生类名（ ）{.....//函数体}</td></tr>
<tr><td colspan="2">6.2.3 类的保护成员（填空★）</td><td>类的保护成员是指在类声明中以关键字 protceted 声明的成员，保护成员具有私有成员和公有成员的双重角色 :对派生类的成员函数而言，它是公有成员，可直接访问；而对其他函数而言，它是私有成员，不能直接访问，只能通过基类的对象访问。</td></tr>
<tr><td rowspan="4">6.2.4 访问权限和赋值兼容规则</td><td>6.2.4.1 公有派生和赋值兼容规则（选择、填空★★）</td><td>在公有派生情况下，基类成员的访问权限在派生类中保持不变。赋值兼容规则是指在公有派生情况下，一个派生类的对象可以作为基类的对象来使用的情况。</td></tr>
<tr><td>6.2.4.2 “isa”和“has-a”的区别（选择★）</td><td>所谓“isa”，就是公有继承“就是一个”的含义，所谓“has-a”，就是分层时“有一个”的含义</td></tr>
<tr><td>6.2.4.3 公有继承存取权限表（选择、填空★★）</td><td>
<table>
<tr><td>基类成员</td><td>派生类成员函数</td><td>基类和派生类对象</td><td>外部函数</td></tr>
<tr><td>private 成员</td><td>不可访问</td><td>不可访问</td><td>不可访问</td></tr>
<tr><td>protected 成员</td><td>protected</td><td>不可访问</td><td>不可访问</td></tr>
<tr><td>public 成员</td><td>public</td><td>可访问</td><td>可访问</td></tr>
</table>
在公有继承的情况下，基类数据成员在派生类中的访问权限保持不变。</td></tr>
<tr><td>6.2.4.4 私有派生(填空★）</td><td>通过私有派生，基类的私有和不可访问成员在派生类中是不可访问的，而公有和保护成员这时就成了派生类的私有成员，派生类的对象不能访问继承的基类成员，必须定义公有的成员函数作为接口。</td></tr>
</table>

	6.2.4.5 保护派生(填空★)	派生也可使用 protected 定义，这种派生使原来的权限都降一级使用，即 private 变为不可访问；protected 变为 private；public 变为 protected。因为限制了数据成员和成员函数的访问权限，所以用的也很少。

6.3 多重继承（分析★）

6.3 多重继承（分析★）	多重继承可视为单一继承的扩展，一个类从多个基类派生的一般形式为：class 类名 1：访问控制 类名 2，访问控制 类名 3，…，访问控制 类名 n{ //类体}。

6.4 二义性及其支配规则

6.4.0 二义性及其支配规则（填空★）	对基类成员的访问必须是无二义性的，如使用一个表达式的含义能解释为可以访问多个基类中的成员，则这种对基类成员的访问就是不确定的，称这种访问具有二义性。
6.4.2 派生类支配基类的同名函数(改错★)	基类的成员和派生类新增的成员都具有作用域，基类在外层，派生类在内层。如果此时派生类定义了一个和基类成员函数同名的新成员函数（因参数不同属于重载，所以这里是指具有相同参数表的成员函数），派生类的新成员函数就覆盖了外层的同名成员函数，在这种情况下，直接使用成员名就只能访问派生类的成员函数，只有使用作用域分辨，才能访问基类的同名成员函数。

6.5 典型问题分析

6.5.2 派生类支配基类的同名函数(改错★)	对象一定先调用自己的同名成员函数，只有使用了作用域分辨运算符，才会调用直接基类的同名成员函数。
6.5.3 二义性(填空★)	程序设计过程中，一定要注意避免定义的二义性，为此可通过使用作用域分辨运算符“：：”和成员名限定来消除二义性。

【题目演练】

【单选题】

1.基类中的 public 成员，通过 public 派生，其在派生类中的访问权限为( )

A:不可访问　　B:private　　C:protected　　D:public

【正确答案】D

【解析】在公有派生情况下，基类的公有成员在派生类中仍然是公有的

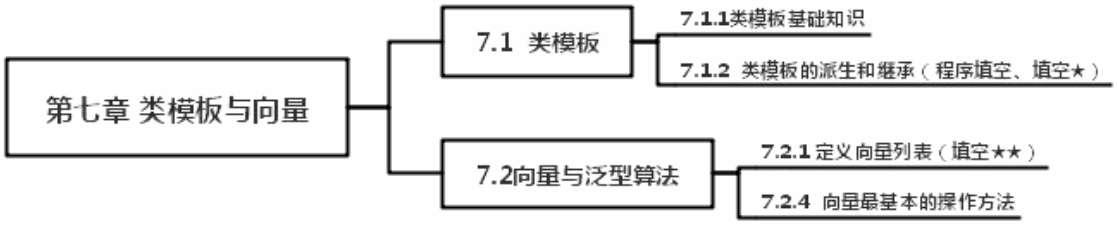
2.派生类继承了基类的（　）

A:所有成员　　B:部分成员　　C:数据成员　　D:成员函数

【正确答案】A

【解析】派生类自动地将基类的所有成员作为自己的成员。派生类继承了基类所有的数据成员和成员函数

第七章 类模板与向量



7.1 类模板

7.1.1 类模板基础知识	7.1.1.1 类模板的成分及语法（程序填空、改错、填空、分析★★★）	类模板声明的一般形式 :template <类模板参数> class 类名 { //类体}; 类模板参数中的 class 意为“任意内部类型或用户定义类型”，T 是结构或类。 通用的数据类型 T 并不是类，而是对类的描述，常称之为类模板。在编译时，由编译器将类模板与某种特定数据类型联系起来，就产生一个特定的类（模板类）。
	7.1.1.2 类模板的对象（填空、选择、程序填空★）	类模板声明的一般形式 :template <类模板参数> class 类名 { //类体}; 类模板参数中的 class 意为“任意内部类型或用户定义类型”，T 是结构或类。
7.1.2 类模板的派生和继承（程序填空、填空★）		类模板也可以继承，继承的方法和普通的类一样。声明模板继承之前，必须重新声明类模板。模板类的基类和派生类都可以是模板类或非模板类。

7.2 向量与泛型算法

7.2.1 定义向量列表（填空★★）	如果用 length 表示长度，用 type 表示数据类型，用 name 表示对象名，则有：
	vector <type> name; //定义元素类型为 type 的向量空表
	vector <type> name(length); //定义具有 length 个类型为 type 的元素的向量，元素均初始化为 0
	vector <type> name（length,a）; //定义具有 length 个类型为 type 的元素的向量，元素均初始化为 a
	vector <type> name1（name）; //用已定义的向量 name1 构造向量 name

7.2.4 向量最基本的操作方法	7.2.4.1 访问向量容量信息的方法（填空★）	size（）：返回当前向量中已经存放的对象个数； max_size（）：返回向量最多可以容纳的对象个数，此参数取决于硬件结构，不由用户指定； capacity（）：返回无需再次分配内存就能容纳的对象个数，其初始值为程序员最初申请的元素个数，即已申请的空间。实际操作时，当存放空间已满，又增加一个元素时，其值在原来的基础上自动翻倍，扩充空间以便存放更多的元素。这前三者关系：max_size( )>=capacity( )>=size( ) empty（）：当前向量为空时，返回 true 值，内容不空时，返回 0 值。
	7.2.4.2 访问向量中对象的方法（填空★）	front（）：返回向量中的第一个对象； back（）：返回向量中的最后一个对象； operator[]（size_type，n）：返回向量中下标为 n（第 n+1 个）的元素。
	7.2.4.3 在向量中插入对象的方法（改错★）	push_back（const T&）：向向量尾部插入一个对象。 insert(iterator it,const T&):向 it 所指的向量位置前插入一个对象。 insert(iterator it,size_type n,const T&X):向 it 所指向量位置前插入 n 个值为 x 的对象。
	7.2.4.4 在向量中删除对象的方法（填空★）	pop_back（const T&）:删除向量中最后一个对象。 erase(iterator it):删除 it 所指向的容器对象 clear():删除向量中的所有对象，empty()返回 true 值。

【题目演练】

【单选题】

1.关于类模板的说法正确的是（ ）。

A:类模板的主要作用是生成抽象类　　B:类模板实例化时，编译器将根据给出的模板实参生成一个类

C:在类模板中的数据成员具有同样类型　　D:类模板中的成员函数没有返回值

【正确答案】B

【解析】类模板也称为参数化类。初始化类模板时，只要传给它指定的数据类型（例如 double 或 int），编译器就用指定类型替代模板参数产生相应的模板类。即类模板实例化时，编译器将根据给出的模板实参生成一个类。因此选择 B 选项

2.关于函数模板，描述错误的是（ ）。

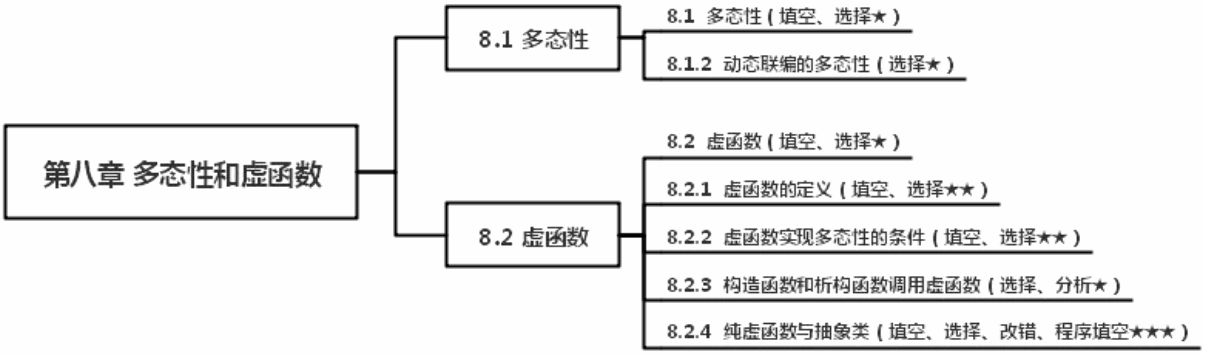
A:函数模板必须由程序员实例化为可执行的函数模板　　B:函数模板的实例化由编译器实现

C:一个类定义中，只要有一个函数模板，则这个类是类模板　　D:类模板的成员函数都是函数模板，类模板实例化后，成员函数也随之实例化

【正确答案】C

【解析】类模板声明的一般方法为：template<类模板参数>class 类名{//类体}；，因此类中有函数模板不能说明是类模板，因此 C 选项错误

第八章 多态性和虚函数



8.1 多态性

8.1 多态性(填空、选择★)	静态联编所支持的多态性称为编译时的多态性；动态联编所支持的多态性称为运行时的多态性，由虚函数实现。C++中要实现动态联编，调用虚函数时必须使用基类指针。
8.1.2 动态联编的多态性（选择★）	虚函数的地址翻译取决于对象的内存地址，虚函数的调用规则是：根据当前对象，优先调用对象本身的虚成员函数。派生类能继承基类的虚函数表，而且只要是和基类同名（参数相同）的成员函数，无论是否使用 virtual 声明，它们都自动成为虚函数。

8.2 虚函数

8.2 虚函数(填空、选择★)	一旦基类定义了虚函数，该基类的派生类中的同名函数也自动成为虚函数
8.2.1 虚函数的定义(填空、选择★★)	虚函数只能是类中的一个成员函数，但不能是静态成员，关键字 virtual 用于类中该函数的声明中。
8.2.2 虚函数实现多态性的条件（填空、选择★★）	产生运行时的多态性有 3 个条件：1.类之间的继承关系满足赋值兼容性规则 2.改写了同名虚函数 3.根据赋值兼容性规则使用指针(或引用)
8.2.3 构造函数和析构函数调用虚函数（选择、分析★）	构造函数不得声明为虚函数，在构造函数和析构函数中调用虚函数采用静态联编，即被调用的虚函数是自己的类或基类中定义的虚函数
8.2.4 纯虚函数与抽象类(填空、选择、改错、程序填空★★★)	析构函数可以通过 virtual 修饰为虚函数。一个类中只能有一个虚析构函数。只要基类的析构函数被声明为虚函数，那么由它派生的所有派生类的析构函数，无论是否使用 virtual 进行说明，都自动地成为虚函数

	（1）说明纯虚函数的一般形式为：virtual 函数类型 函数名（参数列表）=0；
	（2）如果通过同一个基类派生一系列的类，则将这些类总称为类族。抽象类的这一特点保证了进入类族的每个类都具有提供纯虚函数所要求的行为，进而保证了围绕这个类族所建立起来的软件能正常运行。
	（3）抽象类至少含有一个虚函数，而且至少有一个虚函数是纯虚函数，以便将它与空的虚函数区分开来，如下：virtual void area( )=0; //纯虚函数 virtual void area( ) { }; //空的虚函数
	（4）在成员函数内可以调用纯虚函数，因为没有为纯虚函数定义代码，所以在构造函数或析构函数内调用一个纯虚函数将导致程序运行错误。

【题目演练】

【单选题】

1. C++中要实现动态联编，调用虚函数时必须使用（ ）调用虚函数。

A:基类指针 B:类名 C:派生类指针 D:对象名

【正确答案】A

【解析】联编是指一个计算机程序自身彼此关联的过程。按照联编所进行的阶段不同，可分为两种不同的联编方法：静态联编和动态联编。编译程序在编译阶段并不能确切知道将要调用的函数，只有在程序执行时才能确定将要调用的函数，为此要确切知道该调用的函数，要求联编工作要在程序运行时进行，这种在程序运行时进行联编工作被称为动态联编，或称动态束定，又叫晚期联编。C++规定动态联编是在虚函数的支持下实现的。C++中要实现动态联编，调用虚函数时必须使用基类指针。

2.下面函数原型声明中，声明了 fun 为虚函数的是（ ）。

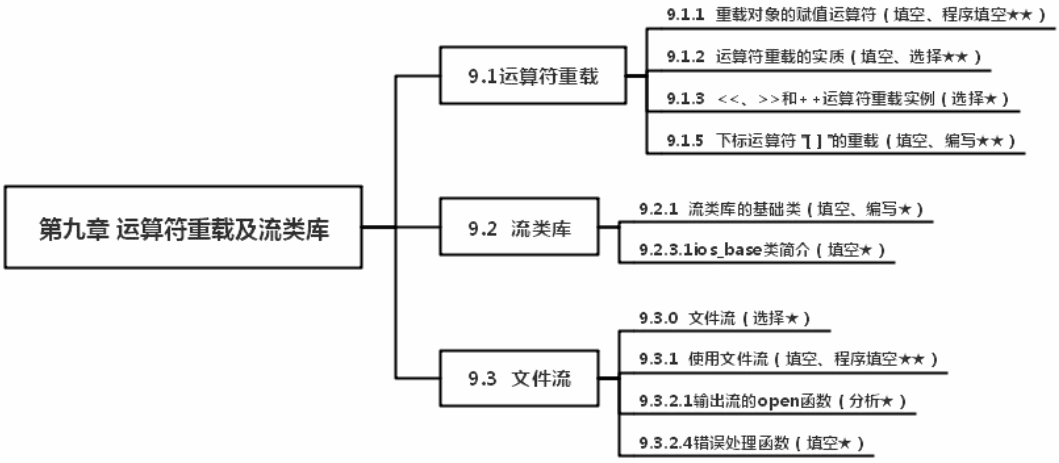
A:void fun()=0 B:virtual void fun()=O C:virtual void fun() D:virtual void fun(){}

【正确答案】C

【解析】虚函数只能是类中的一个成员函数，但不能是静态成员，关键字 virtual 用于类中该函数的声明中。例如：

```
class A{
        public:
          virtual void fun( );//声明虚函数
} ;
void A::fun( ){...//}   //定义虚函数
```

第九章 运算符重载及流类库

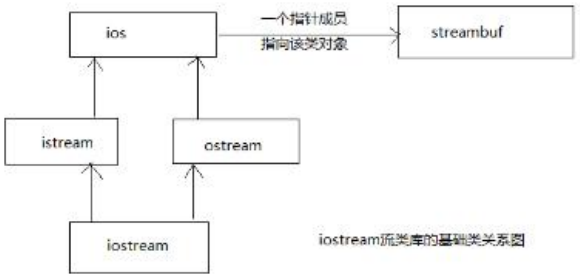


9.1 运算符重载

9.1.1 重载对象的赋值运算符（填空、程序填空★★）	默认情况下，编译器为每个类生成一个默认的赋值操作，即使没有重载赋值运算符，仍然可以使用运算符“=”，用于同类的两个对象之间相互赋值。在这种情况下，默认的赋值操作就是同类对象之间，对象之间对应成员的逐一赋值。很多情况下，这样的赋值方式对于有些类可能是不正确的，如类中包含指向动态空间的指针。赋值运算符只能作为成员函数重载。
9.1.2 运算符重载的实质（填空、选择★★）	运算符重载的实质就是函数重载，要重载某个运算符，只要重载相应的函数就可以了。 运算符的重载形式有两种，重载为类的成员函数和重载为类的友元函数。operator 是定义运算符重载函数的关键字,C++的运算符大部分都可以重载，除了“.”、“::”、“*”、“?:”和 sizeof 五种运算符，另外，sizeof 和#不是运算符，故不能重载；而 = （）[] ->这四个运算符只能用类运算符来重载。
9.1.3 <<、>>和++运算符重载实例（选择★）	插入符"<<"和提取符">>"运算符重载只能作为类的友元重载。++运算符既可以是前缀运算符，也可以是后缀运算符。为了区别这两种情况，重载这两个运算符时必须在格式上有所区别：重载后缀++时必须多一个虚拟参数：int，因此从形式上看像是一个二元运算符重载。
9.1.5 下标运算符"[]"的重载（填空、编写★★）	C++中，运算符[]只能用类运算符来重载。

9.2 流类库

9.2.1 流类库的基础类（填空、编写★）



C++的流类库预定义了 4 个流：cin(标准输入)、cout（标准输出）、cerr（标准出错信息输出）和 clog（带缓冲的标准出错信息输出）。

9.2.2 默认输入输出格式控制（填空★）

因为字符串没有结束位，所以将字符串作为整体输出时，有效字符串的后面将出现乱码。不过，可以用手工增加表示字符串的结束符“\0” 来消除乱码。

9.2.3.1ios_base 类简介（填空★）

常用常量名	含义
left/right	输出数据按输出域左边/右边对其
dec/oct/hex	转换基数为十进制/八进制/十六进制形式
scientific/fixed	使用科学计数法/定点形式表示浮点数
skipws	跳过输入中的空白
internal	在指定任何引导标志或基之后填充字符
showbase	输出带有一个表示制式的字符
showpoint	浮点输出时必须带有一个小数点和尾部的 0
showpos	在正数前添加一个“+”号
uppercase	十六进制数值输出使用大写 A~F，科学计数显示使用大写字母 E
untibuf	每次插入之后，ostream::osfx 刷新该流的缓冲区，默认缓冲单元为 cerr
boolalpha	把逻辑值输出为 true 和 false（否则输出 1 和 0）
adjustfield	对齐方式域（与 left、right、internal 配合使用，如 ios_base::adjustfield）
basefield	数字方式域（与 dec、oct、hex 配合使用，如 ios_base::basefield）
floatfield	浮点方式域（与 fix、scientific，如 ios_base::floatfield）

成员函数	作用
long flags(long)	允许程序员设置标志字的值，并返回以前所设置的标志字
long flags()	仅返回当前的标志字
long setf(long,long)	用于设置标志字的某一位，第 2 个参数指定所要操作的位，第 1 个参数指定为该位所设置的值
long setf(long)	用于设置参数指定的标志位
long unsetf(long)	用于清除参数指定的标志位
int width(int)	返回以前设置的显示数据的域宽
int width()	只返回当前域宽（默认宽度为 0）
char fill(char)	设置填充字符，设置的宽度小时，空余的位置用填充字符来填充，默认条件下是空格。该函数返回以前设置的填充字符。
char fill()	获得当前的填充字符
int precision(int)	返回以前设置的精度（小数点后的小数位数）
int precision()	返回当前的精度

9.3 文件流

9.3.0 文件流（选择★）	在 C++中，文件操作通过流来完成。C++有输入文件流、输出文件流和输入输出文件流 3 种，ifstream 是文件输入流类，ofstream 是文件输出流类，fstream 是文件输入输出流类。要利用这些类来定义文件流对象，必须在程序的开始部分包含如下的预处理命令： #include <fstream>
9.3.1 使用文件流（填空、程序填空★★）	1.打开一个相应的文件流，如“ofstream myStream;”打开一个输出文件流。 2.把这个流和相应的文件关联起来，如“myStream.open(“myText.txt”);”，3.操作文件流，想对文件进行什么操作，对相应的文件流进行相应操作即可。4.及时关闭不再使用的文件，格式为：“文件流名.close();”
9.3.2.1 输出流的 open 函数(分析★）	open 函数的原型： void open(char const *,int filemode,int=filebuf::openprot)。 当使用 ofstream 流类定义一个流对象并打开一个磁盘文件时，文件的隐含打开方式为 iOS::out。
9.3.2.4 错误处理函数（填空★）	在对一个流对象进行 I/O 操作时，可能会产生错误

【题目演练】

【单选题】

1. C++中定义标准输入的库为（ ）。

A:stdio　　B:math　　C:ifstream　　D:stdlib

【正确答案】C

【解析】要打开一个输入文件流，需要定义一个 ifstream 类型的对象。

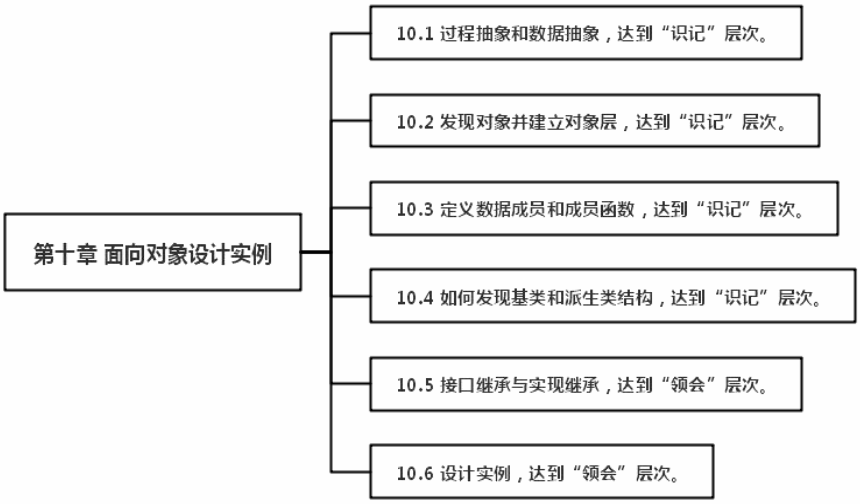
2.进行文件操作时需要包含头文件（ ）。

A:iostream　　B:fstream　　C:stdio　　D:stdlib

【正确答案】B

【解析】要打开一个输入文件流，需要定义一个 ifstream 类型的对象；要打开一个输出文件流，需要定义一个 ofstream 类型的对象；如果要打开输入输出文件流，则要定义一个 fstream 类型的对象。这三种类型都定义在头文件<fstream>里。

第十章 面向对象设计实例



10.1 过程抽象和数据抽象，达到“识记”层次。	抽象是形成概念的必要手段，它抽取事物的本质特征。对于分析而言，抽象原则有两方面的意义:1、分析员只需要分析研究与系统目标有关的事物及其本质特征，对那些鱼系统目标无关的特征则不需要了解 2、通过舍弃个体事物在细节上的差异，抽取其共同特征得到一批事物的抽象概念
	（1）过程抽象：是指任何一个完成确定功能的操作序列，其使用者都可以把它看作一个单一的实体；（2）数据抽象：是根据施加于数据之上的操作来定义数据类型，并限定数据的值只能由这些操作来修改和观察。
10.2 发现对象并建立对象层，达到“识记”层次。	步骤：1、将问题域和系统责任作为出发点 2、正确运用抽象原则 3、寻找候选对象的基本方法 4、审查和筛选对象 5、异常情况的检查和调整
10.3 定义数据成员和成员函数，达到“识记”层次。	发现对象的数据成员，应首先考虑是否可以借鉴以往的面向对象分析结果，并尽可能复用以往已开发模型中同类对象数据成员的定义。然后重点研究当前的问题域和系统责任，根据本系统的实际情况正确地进行抽象，从而找出每一类对象的数据成员
10.4 如何发现基类和派生类结构，达到“识记”层次。	发现基类和派生类结构同样可以借鉴以往的面向对象分析结果，尽可能复用以往的系统成分。发现基类和派生类结构可按如下方法进行 :1、学习当前领域的分类学知识 2、按照常识考虑事物的分类 3、构建基类与派生类 4、考察类的成员
10.5 接口继承与实现继承，达到“领会”层次。	类的成员函数提供了与外界的接口。成员函数有实函数、虚函数和纯虚函数 3 种。 公有继承有两部分组成，即函数接口的继承和函数实现的继承

【题目演练】

【单选题】

1.下面说法正确的是（ ）

A:继承的总是虚函数　　B:继承的总是成员函数　　C:继承的总是成员函数接口

D:继承的总是公有成员函数

【正确答案】C

【解析】公有继承有两部分组成，即函数接口的继承和函数实现的继承。可归纳成如下几点：（1）继承的总是成员函数的接口（2）声明纯虚函数的目的是使派生类仅仅继承函数接口，而纯虚函数的实现则由派生类去完成（3）声明虚函数的目的是使派生类既能继承基类对此虚函数的实现，又能继承虚函数提供的接口（4） 声明实函数的目的是使派生类既能继承基类对此实函数的实现，又能继承实函数提供的接口

2.声明实函数的目的是使（ ）

A:派生类既继承函数的实现，也继承函数接口　　B:派生类只继承函数接口

C:派生类只继承函数的实现　　D:派生类改写实函数

【正确答案】A

【解析】对于一个实成员函数，无论派生类如何变化，它都不会变化。即实成员函数在派生类中保持不变。声明实函数的目的是使派生类既继承该函数的实现，也继承其函数接口。