1. C中static有什么作用？

（1）隐藏。 只在当前文件中可见（静态全局变量和静态函数）（定义静态函数的好处：不被其他文件所使用，其他文件中可以出现同名函数，不会产生冲突）

（2）保持变量内容的持久。存储在静态数据区的变量会在程序刚开始运行时就完成初始化，也是唯一的一次初始化。共有两种变量存储在静态存储区：全局变量和static变量。（静态局部变量）

（3）static的第三个作用是默认初始化为0。其实全局变量也具备这一属性，因为全局变量也存储在静态数据区。在静态数据区，内存中所有的字节默认值都0×00，某些时候这一特点可以减少程序员的工作量。

补充说明：c++中static的作用：

（1）静态数据成员：所有对象只有一份拷贝，共享访问，它的值可以更新。静态数据成员存储在全局数据区，定义时需要分配空间，所以不能在类声明中定义

    静态数据成员初始化： <数据类型><类名>::<今天数据成员名> = <值>

    静态诗句成员有两种访问方式：

    <类对象名>.<静态数据成员名>  或  <类名>::<静态数据成员>

    静态数据成员与全局变量相比，有两个优势：

      静态数据成员没有进入程序的全局名字空间，因此不存在于程序中其它全局名字冲突的可能性。

      可以实现信息的隐藏，可以成为private成员，而全局变量不能

（2）静态成员函数：它为类的全部服务而不是某一个类的具体对象。不具有this指针，无法访问属于类的对象的非静态数据成员，也无法访问非静态成员函数，它只能调用其余的静态成员函数。

2．C++中const有什么用？

    const修饰的内容不可改变， 定义常量，还有const数据成员，const参数， const返回值， const成员函数等, 被const修饰的东西都受到强制保护，可以预防意外的变动，能提高程序的健壮性。

3. C与C++各自是如何定义常量的？有什么不同？

    C中是使用宏#define定义， C++使用更好的const来定义。

    区别：

    1）const是有数据类型的常量，而宏常量没有，编译器可以对前者进行静态类型安全检查，对后者仅是字符替换，没有类型安全检查，而且在字符替换时可能会产生意料不到的错误（边际效应）。

    2）有些编译器可以对const常量进行调试， 不能对宏调试。

4. 既然C++中有更好的const为什么还要使用宏？

    const无法代替宏作为卫哨来防止文件的重复包含。

5. C++中引用和指针的区别?

    引用是对象的别名, 操作引用就是操作这个对象, 必须在创建的同时有效得初始化(引用一个有效的对象, 不可为NULL), 初始化完毕就再也不可改变, 引用具有指针的效率, 又具有变量使用的方便性和直观性, 在语言层面上引用和对象的用法一样, 在二进制层面上引用一般都是通过指针来实现的, 只是编译器帮我们完成了转换. 之所以使用引用是为了用适当的工具做恰如其分的事, 体现了最小特权原则.

6. 说一说C与C++的内存分配方式？

   1）从静态存储区域分配。内存在程序编译的时候就已经分配好，这块内存在程序的整个运行期间都存在，如全局变量，static变量。

   2）在栈上创建。在执行函数时，函数内局部变量的存储单元都可以在栈上创建，函数执行结束时这些存储单元自动被释放。栈内存分配运算内置于处理器的指令集中，效率很高，但是分配的内存容量有限。

   3）从堆上分配（动态内存分配）程序在运行的时候用malloc或new申请任意多少的内存，程序员负责在何时用free或delete释放内存。动态内存的生存期自己决定，使用非常灵活。

7. new/delete 与 malloc()/free() 的区别?

   malloc() 与 free() 是C语言的标准库函数, new/delete 是C++的运算符, 他们都可以用来申请和释放内存, malloc()和free()不在编译器控制权限之内, 不能把构造函数和析构函数的任务强加给他们.

8. #include<a.h>和#include“a.h” 有什么区别?

   答：对于#include <a.h> ，编译器从标准库路径开始搜索 a.h对于#include “a.h” ，编译器从用户的工作路径开始搜索 a.h

9. 在C++ 程序中调用被 C编译器编译后的函数，为什么要加 extern “C”？       C++语言支持函数重载，C语言不支持函数重载。函数被C++编译后在库中的名字与C语言的不同。假设某个函数的原型为： void foo(int x, int y);该函数被C编译器编译后在库中的名字为\_foo，而C++编译器则会产生像\_foo\_int\_int之类的名字。C++提供了C连接交换指定符号extern“C”来解决名字匹配问题。

10. C++中的什么是多态性？ 是如何实现的？

    多态性是面向对象程序设计语言继数据抽象和继承之后的第三个基本特征。它是在运行时出现的多态性通过派生类和虚函数实现。基类和派生类中使用同样的函数名, 完成不同的操作具体实现相隔离的另一类接口，即把“ w h a t”从“h o w”分离开来。多态性提高了代码的组织性和可读性，虚函数则根据类型的不同来进行不同的隔离。

11. 什么是动态特性?

    在绝大多数情况下, 程序的功能是在编译的时候就确定下来的, 我们称之为静态特性. 反之, 如果程序的功能是在运行时刻才能确定下来的, 则称之为动态特性。C++中, 虚函数，抽象基类， 动态绑定和多态构成了出色的动态特性。

12．什么是封装?C++中是如何实现的？

    封装来源于信息隐藏的设计理念， 是通过特性和行为的组合来创建新数据类型让接口与具体实现相隔离。C++中是通过类来实现的, 为了尽量避免某个模块的行为干扰同一系统中的其它模块,应该让模块仅仅公开必须让外界知道的接口.

13. 什么是RTTI?

   RTTI事指运行时类型识别（Run-time type identification）在只有一个指向基类的指针或引用时确定一个对象的准确类型。

14. 什么是拷贝构造函数?

    它是单个参数的构造函数，其参数是与它同属一类的对象的(常)引用；类定义中，如果未提供自己的拷贝构造函数，C++提供一个默认拷贝构造函数，该默认拷贝构造函数完成一个成员到一个成员的拷贝

15. 什么是深浅拷贝?

    浅拷贝是创建了一个对象用一个现成的对象初始化它的时候只是复制了成员(简单赋值)而没有拷贝分配给成员的资源(如给其指针变量成员分配了动态内存); 深拷贝是当一个对象创建时，如果分配了资源，就需要定义自己的拷贝构造函数，使之不但拷贝成员也拷贝分配给它的资源.

16．面向对象程序设计的优点?

    开发时间短, 效率高, 可靠性高。面向对象编程的编码具有高可重用性，可以在应用程序中大量采用成熟的类库（如STL）,从而虽短了开发时间，软件易于维护和升级。

1. C和C++有什么区别？

C++是C语言的超集（我看网上很多文章说这是不对的），这意味着几乎所有的C程序都可以在C++编译器中编译和运行。然而，C++引入了许多新的概念和特性，使得两种语言在一些关键点上有显著的区别。

以下是C和C++的一些主要区别：

面向对象编程：C++支持面向对象编程（OOP），包括类、对象、继承、封装、多态等特性。这使得C++更适合大型软件项目，因为OOP可以提高代码的重用性和可读性。C语言是一种过程性语言，没有这些特性。

STL（Standard Template Library）：C++提供了STL，这是一套强大的模板类和函数的库，包括列表、向量、队列、栈、关联数组等。这些可以大大提高开发效率。C语言没有内置的数据结构库。

异常处理：C++提供了异常处理机制，可以更优雅地处理错误情况。C语言处理错误通常依赖于函数返回值。

构造函数和析构函数：C++支持构造函数和析构函数，这些特殊的函数允许对象在创建和销毁时执行特定的代码。C语言没有这个概念。

运算符重载：C++允许运算符重载，这意味着开发者可以更改已有运算符的行为，或者为用户自定义类型添加新的运算符。C语言不支持运算符重载。

例如，如果我们要创建一个复数类并对其进行算术运算，C++的面向对象和运算符重载特性就非常有用。我们可以定义一个复数类，然后重载+、-和\*运算符以执行复数的加法、减法和乘法。这样，我们就可以像处理内置类型一样处理复数对象。反观C语言，我们需要定义结构体来存储复数，并且需要写一堆函数来处理复数的加法、减法和乘法。

2. C语言的结构体和C++的有什么区别

C语言的结构体和C++的结构体在基本的使用上是相似的，都是用来封装多个不同或相同类型的数据。然而，C++中的结构体继承了C++面向对象的特性，与C语言中的结构体有一些关键性的区别：

成员函数：C++的结构体可以包含成员函数（包括构造函数和析构函数），而C语言的结构体不能。

访问控制：C++的结构体支持公有（public）、保护（protected）和私有（private）三种访问控制级别，而C语言的结构体中的所有成员都是公有的。

继承：C++的结构体可以从其他结构体或类继承，而C语言的结构体不能继承。

3. C 语言的关键字 static 和 C++ 的关键字 static 有什么区别

在C和C++中，static关键字有三个主要的用途，但其在C++中的用法更加丰富：

在函数内部：在C和C++中，static关键字可用于函数内部变量。此时，此变量的生命周期将贯穿整个程序，即使函数执行结束，这个变量也不会被销毁。每次调用这个函数时，它都不会重新初始化。这可以用于实现一些需要保持状态的函数。

在函数外部或类内部：在C和C++中，static关键字可以用于全局变量或函数。此时，此变量或函数的作用域被限制在定义它的文件内，无法在其他文件中访问。这可以防止命名冲突或不必要的访问。

在类内部：只有C++支持此用法。在C++中，static关键字可以用于类的成员变量或成员函数。对于静态成员变量，无论创建多少个类的实例，都只有一份静态成员变量的副本。静态成员函数则可以直接通过类名调用，而无需创建类的实例。

4. C++ 和 Java有什么核心区别？

C++和Java都是广泛使用的编程语言，但它们在设计理念、功能和用途上有很大的不同。以下是C++和Java的几个核心区别：

运行环境：Java是一种解释型语言，它的代码在JVM（Java虚拟机）上运行，这使得Java程序可以在任何安装有JVM的平台上运行，实现了“一次编写，到处运行”的理念。而C++是一种编译型语言，其代码直接编译成目标机器的机器码运行，因此需要针对特定平台编译。

内存管理：Java有自动内存管理和垃圾回收机制，程序员不需要直接管理内存。而在C++中，程序员需要手动进行内存的分配和释放，这提供了更大的控制力，但同时也增加了内存泄漏的风险。

面向对象编程：Java是一种纯面向对象的编程语言，所有的代码都需要包含在类中。与此不同，C++支持面向对象编程，但它也允许过程式编程。

错误处理：Java使用异常处理机制进行错误处理，而C++既支持异常处理，也支持通过返回值进行错误处理。

多线程：Java内置了对多线程的支持，而C++在C++11标准之后引入了对多线程的支持。

性能：因为C++的代码直接编译为机器码，所以它通常比Java程序运行得更快。但是，Java的跨平台能力和内置的垃圾回收机制使其在开发大型企业级应用时更具优势。

例如，如果你正在开发一个需要直接访问硬件，或者需要高性能数学计算的应用（比如游戏，图形渲染，科学计算），C++可能是一个更好的选择。而如果你正在开发一个大型的企业级web应用，Java的跨平台能力，内置的垃圾回收和强大的类库可能会更有优势。

5. C++中，a和&a有什么区别？

在C++中，a和&a表示的是两种完全不同的概念：

a：当你在代码中写a时，你正在引用变量a的值。例如，如果你之前写的int a = 10;，那么a的值就是10。

&a：&是一个地址运算符，它给出了变量a在内存中的位置。这被称作a的引用或者是指向a的指针。例如，如果你写int\* p = &a;，那么p就是一个指向a的指针，你可以通过\*p来访问或修改a的值。

这是C++中的一种基础概念，被称为指针和引用。通过指针和引用，你可以直接操作内存，这在很多情况下都非常有用，例如，动态内存分配，函数参数传递，数据结构（如链表和树）等等。

6. C++中，static关键字有什么作用？

在C++中，static关键字有多个用途，它的作用主要取决于它在哪里被使用：

在函数内部：如果static被用于函数内部的变量，那么它会改变该变量的生命周期，使其在程序的整个运行期间都存在，而不是在每次函数调用结束时被销毁。这意味着，这个变量的值在函数调用之间是保持的。

在函数外部：如果static被用于函数外部的全局变量或函数，那么它会将这个变量或函数的链接范围限制在它被定义的文件内。换句话说，这个变量或函数不能在其他文件中被直接访问。这可以帮助减少命名冲突，而且能提供一种控制变量和函数可见性的方式。

在类中：如果static被用于类的成员变量，那么该变量将会成为这个类的所有实例共享的变量，也就是说，类的每个实例都能访问到这个同样的变量。如果static被用于类的成员函数，那么这个函数可以直接通过类来调用，而不需要创建类的实例。

7. C++中，#define和const有什么区别？

#define和const都可以用来定义常量，但它们在实现方式和使用上有一些区别。

预处理器与编译器：#define是预处理器指令，在编译前会被预处理器替换，它只是简单的文本替换，不进行类型检查，也不会分配内存。而const是编译器处理的，它会在编译时进行类型检查，确保你不会意外地改变它的值。

作用域：#define没有作用域的概念，一旦定义，到文件结束都有效。而const常量有作用域，它的作用范围限制在定义它的块或者文件中。

调试：在调试时，#define定义的宏常量无法查看，因为在预处理阶段就已经被替换掉了。而const定义的常量在调试过程中是可以查看的。

9. 静态链接和动态链接有什么区别？

静态链接和动态链接是两种不同的程序链接方式，它们主要的区别在于链接的时间和方式。

静态链接：在静态链接中，所有代码（包括程序本身的代码和它依赖的库的代码）都会在编译时期被合并为一个单一的可执行文件。这个可执行文件包含了程序运行所需的所有信息，因此它不依赖于任何外部的库文件。静态链接的优点是部署简单，因为不需要额外的依赖，只需要一个文件就可以运行。缺点是可执行文件通常会比动态链接的大，因为它包含了所有需要的代码，而且如果库更新，程序需要重新编译和链接。

动态链接：在动态链接中，程序的代码和它依赖的库的代码被分开。程序的可执行文件只包含了程序本身的代码和一些标记，这些标记表示程序在运行时需要链接到哪些库。当程序运行时，操作系统会负责加载这些库并进行链接。动态链接的优点是可执行文件更小，因为它不包含库的代码，而且多个程序可以共享同一份库，节省内存。此外，如果库更新，只需要替换库文件，程序无需重新编译和链接。缺点是部署稍微复杂一些，因为需要确保运行环境中有所需的库文件。

例如，假设我们有一个程序，它使用了一个数学库。如果我们静态链接这个库，那么所有的数学函数都会被包含在我们的可执行文件中，我们可以将这个文件复制到任何地方运行。如果我们动态链接这个库，那么我们的可执行文件就会小得多，但如果我们想在另一台机器上运行这个程序，我们就需要确保那台机器上也安装了这个数学库。

10. 变量的声明和定义有什么区别

在C++中，变量的声明和定义是两个不同的概念。

声明是告诉编译器某个变量的存在，以及它的类型。声明并不分配存储空间。例如，外部变量的声明extern int a;，这里只是告诉编译器有一个类型为int的变量a存在，具体的a在哪里定义的，编译器此时并不知道。

定义是声明的延伸，除了声明变量的存在和类型以外，还分配了存储空间。例如，int a;就是一个定义，编译器在这里为a分配了足够的存储空间来存储一个整数。

在C++中，一个变量可以被声明多次，但只能被定义一次。例如，我们可以在多个文件中声明同一个变量，但只能在一个文件中定义它。如果在多个地方定义同一个变量，编译器会报错。

举个例子，假设我们正在编写一个大型程序，这个程序有一个全局变量需要在多个文件中使用。我们可以在一个文件中定义这个变量，然后在其他需要使用这个变量的文件中声明它。这样，所有的文件都可以访问到这个变量，但只有一个文件负责管理它的存储空间。

11. typedef 和define 有什么区别

typedef和#define都是C++中用于定义别名的关键字，但它们的用途和行为有所不同。

typedef是C++的一个关键字，用于为现有的类型创建一个新的名称（别名）。例如，如果我们想要为unsigned long int创建一个更简单的别名，我们可以写typedef unsigned long int ulong;，然后在代码中就可以使用ulong来代替unsigned long int。typedef只能为类型定义别名，不能为值定义别名。

#define是预处理器的一个指令，用于创建宏。宏可以是一个值，也可以是一段代码。例如，#define PI 3.14159就定义了一个名为PI的宏，它的值是3.14159。#define的作用范围更广，它不仅可以为类型定义别名，也可以为值定义别名，甚至可以定义一段代码。

两者的主要区别在于：

typedef仅作用于类型，而#define可以定义类型、值或者代码。

typedef是由编译器解析的，而#define是由预处理器处理的。因此，typedef的作用范围是局部的，只在定义它的文件或作用域内有效，而#define的作用范围是全局的，一旦定义即在整个源代码中有效。

typedef定义的别名会受到类型检查，而#define定义的宏不会。例如，如果你试图使用typedef为一个函数类型定义别名，然后使用这个别名定义一个整数，编译器会报错。但是如果你使用#define定义一个函数类型的宏，然后使用这个宏定义一个整数，预处理器会默默地接受。

typedef可以处理模板化的类型，而#define不能。例如，typedef std::vector int\_vector;是合法的，但是使用#define来做同样的事情就会出现问题。

typedef定义的别名在调试时更友好。因为它是编译器处理的，所以在调试时可以看到别名。而#define定义的宏在预处理阶段就被替换掉了，所以在调试时看不到宏的名称，只能看到宏的值。

12. final和override关键字

final和override是C++11引入的两个关键字，主要用于类的继承和虚函数的覆盖。

final：如果一个类被声明为final，那么它不能被继承。例如，class Base final { … };，此时任何试图继承Base的类都会导致编译错误。此外，如果一个虚函数被声明为final，那么它不能在派生类中被覆盖。例如，virtual void fun() final;，此时任何派生类试图覆盖fun()函数都会导致编译错误。

override：如果一个虚函数被声明为override，那么编译器会检查这个函数是否真的覆盖了基类中的一个虚函数。如果没有，编译器会报错。这个关键字可以帮助我们避免因为拼写错误或者函数签名错误而导致的错误。例如，void fun() override;，如果基类中没有一个函数的签名和fun()完全匹配，那么编译器就会报错。

例如，假设我们有一个基类Animal和一个派生类Dog。Animal有一个虚函数make\_sound()，Dog需要覆盖这个函数。如果我们在Dog的make\_sound()函数声明中加上了override关键字，那么如果我们不小心将函数名拼写成了mkae\_sound()，编译器就会因为找不到对应的基类函数而报错，帮助我们及时发现错误。

13. 宏定义和函数有何区别？

宏定义（#define）和函数是两种常见的在C++中编写代码的方式，但它们有一些重要的区别：

编译阶段：宏定义是在预处理阶段展开的，而函数是在编译阶段处理的。这意味着使用宏定义的代码在编译前就已经被预处理器替换掉了，而函数在编译阶段会生成对应的函数调用。

类型检查：函数在编译时会进行类型检查，而宏定义不会。这可能会导致宏定义在使用时出现错误，而在编译阶段并不会被发现。

效率：由于宏定义在预处理阶段就被替换，因此它没有函数调用的开销（如堆栈操作），所以在某些情况下可能更快。然而，过度使用宏定义可能会导致编译后的代码体积增大，因为每次使用宏都会插入一份宏的代码副本。

封装：函数提供了更好的封装，使得代码更易于阅读和维护。而宏定义由于其替换性质，可能会在复杂的表达式中产生不易察觉的错误。

14. sizeof 和strlen 的区别

sizeof和strlen是两个在C++中常用的函数，但它们的功能和用途有所不同：

sizeof 是一个编译时操作符，它返回一个对象或者类型所占用的字节数。例如，sizeof(int)将返回4（在大多数现代系统中），sizeof(char)将返回1。如果你对一个数组使用sizeof，它将返回整个数组的大小，而不是数组中的元素个数。例如，int arr[10]; sizeof(arr);将返回40（在大多数现代系统中，因为一个int通常占用4个字节，所以10个int占用40个字节）。

strlen 是一个运行时函数，它返回一个以空字符终止的字符串的长度。该长度不包括终止的空字符。例如，strlen(“hello”)将返回5。注意，strlen只能用于字符数组，且该字符数组必须以空字符(‘\0’)终止，否则strlen会继续读取内存，直到遇到一个空字符，这可能会导致未定义的行为。

举个例子，如果我们有一个字符数组char arr[10] = “hello”;，那么sizeof(arr)将返回10（因为arr是一个10个字符的数组，每个字符占用1个字节），而strlen(arr)将返回5（因为字符串"hello"的长度是5，不包括终止的空字符）。

15. 简述strcpy、sprintf 与memcpy 的区别

strcpy, sprintf, 和 memcpy 都是 C/C++ 标准库中的函数，用于处理字符串和内存，但它们的作用是不同的：

strcpy：这个函数用于将源字符串复制到目标字符串。它会复制源字符串的所有字符，直到遇到终止的空字符，并在目标字符串的末尾添加一个空字符。例如，char dest[10]; strcpy(dest, “hello”);，这将会把字符串 “hello”（包括终止的空字符）复制到 dest 中。需要注意的是，如果源字符串的长度（包括终止的空字符）超过了目标字符串的大小，那么会导致缓冲区溢出，这可能会引发安全问题。

sprintf：这个函数用于将格式化的数据写入字符串。它可以接受多个输入参数，并按照指定的格式将这些参数格式化为一个字符串，然后将这个字符串写入目标字符串。例如，char str[50]; sprintf(str, “Hello, %s!”, “world”);，这将会把 “Hello, world!” 写入 str。同样，如果格式化后的字符串长度（包括终止的空字符）超过了目标字符串的大小，那么会导致缓冲区溢出。

memcpy：这个函数用于复制内存区域。它会从源内存区域复制指定数量的字节到目标内存区域。例如，char dest[10]; char src[10] = “hello”; memcpy(dest, src, 6);，这将会把 src 的前6个字节（包括终止的空字符）复制到 dest。memcpy 不会因为遇到空字符而停止复制，它总是复制指定的字节数，因此，如果指定的字节数超过了目标内存区域的大小，那么会导致缓冲区溢出。

1. [面向对象](https://zhida.zhihu.com/search?content_id=118356713&content_type=Article&match_order=1&q=%E9%9D%A2%E5%90%91%E5%AF%B9%E8%B1%A1&zhida_source=entity)的程序设计思想是什么？

答：把数据结构和对数据结构进行操作的方法封装形成一个个的对象。

2. 什么是类？

答：把一些具有共性的对象归类后形成一个集合，也就是所谓的类。

3. 对象都具有的两方面特征是什么？分别是什么含义？

答：对象都具有的特征是：静态特征和动态特征。  
静态特征是指能描述对象的一些属性（成员变量），动态特征是指对象表现出来的行为（[成员函数](https://zhida.zhihu.com/search?content_id=118356713&content_type=Article&match_order=1&q=%E6%88%90%E5%91%98%E5%87%BD%E6%95%B0&zhida_source=entity" \t "_blank)）

4. 在头文件中进行类的声明，在对应的实现文件中进行类的定义有什么意义？

答：这样可以提高编译效率，因为分开的话只需要编译一次生成对应的.obj文件后，再次应用该类的地方，这个类就不会被再次编译，从而大大的提高了编译效率。

5. 在类的内部定义成员函数的函数体，这种函数会具备那种属性？

答：这种函数会自动为内联函数，这种函数在函数调用的地方在编译阶段都会进行代码替换。

6. 成员函数通过什么来区分不同对象的成员数据？为什么它能够区分？

答：通过this指针指向对象的首地址来区分的。

7. C++编译器自动为类产生的四个缺省函数是什么？

答：默认构造函数，[拷贝构造函数](https://zhida.zhihu.com/search?content_id=118356713&content_type=Article&match_order=1&q=%E6%8B%B7%E8%B4%9D%E6%9E%84%E9%80%A0%E5%87%BD%E6%95%B0&zhida_source=entity)，析构函数，赋值函数。

8. 拷贝构造函数在哪几种情况下会被调用？

答：  
1.当类的一个对象去初始化该类的另一个对象时；  
2.如果函数的形参是类的对象，调用函数进行形参和实参结合时；  
3.如果函数的返回值是类对象，函数调用完成返回时。

9. 构造函数与普通函数相比在形式上有什么不同？（构造函数的作用，它的声明形式来分析）

答：构造函数是类的一种特殊成员函数，一般情况下，它是专门用来初始化对象成员变量的。  
构造函数的名字必须与类名相同，它不具有任何类型，不返回任何值。

10. 什么时候必须重写拷贝构造函数？

答：当构造函数涉及到动态存储分配空间时，要自己写拷贝构造函数，并且要深拷贝。

11. 构造函数的调用顺序是什么？

答：  
1.先调用基类构造函数  
2.按声明顺序初始化数据成员  
3.最后调用自己的构造函数。

12. 哪几种情况必须用到初始化成员列表？

答：类的成员是常量成员初始化；  
类的成员是对象成员初始化，而该对象没有无参构造函数。  
类的成员为引用时。

13. 什么是常对象？

答：常对象是指在任何场合都不能对其成员的值进行修改的对象。

14. 静态函数存在的意义？

答：静态私有成员在类外不能被访问，可通过类的静态成员函数来访问；  
当类的构造函数是私有的时，不像普通类那样实例化自己，只能通过静态成员函数来调用构造函数。

15. 在类外有什么办法可以访问类的非公有成员？

答：[友元](https://zhida.zhihu.com/search?content_id=118356713&content_type=Article&match_order=1&q=%E5%8F%8B%E5%85%83&zhida_source=entity)，继承，公有成员函数。

16. 什么叫抽象类？

答：不用来定义对象而只作为一种基本类型用作继承的类。

17. [运算符重载](https://zhida.zhihu.com/search?content_id=118356713&content_type=Article&match_order=1&q=%E8%BF%90%E7%AE%97%E7%AC%A6%E9%87%8D%E8%BD%BD&zhida_source=entity)的意义？

答：为了对用户自定义数据类型的数据的操作与内定义的数据类型的数据的操作形式一致。

18. 不允许重载的5个运算符是哪些？

答：  
1. .\*（成员指针访问运算符号）  
2. ：：域运算符  
3. Sizeof 长度运算符号  
4. ？：条件运算符号  
5. .（成员访问符）

19. 运算符重载的三种方式？

答：普通函数，友元函数，类成员函数。

20. 流运算符为什么不能通过类的成员函数重载？一般怎么解决？

答：因为通过类的成员函数重载必须是运算符的第一个是自己，而对流运算的重载要求第一个参数是流对象。所以一般通过友元来解决。

21. 赋值运算符和拷贝构造函数的区别与联系？

答：  
相同点：都是将一个对象copy到另一个中去。  
不同点：拷贝构造函数涉及到要新建立一个对象。

22. 在哪种情况下要调用该类的析构函数？

答：对象生命周期结束时。

23. 对象间是怎样实现数据的共享的？

答：通过类的静态成员变量来实现对象间的数据共享。静态成员变量占有自己独立的空间不为某个对象所私有。

24. 友元关系有什么特性？

答：单向的，非传递的，不能继承的。

25. 对对象成员进行初始化的次序是什么？

答：它的次序完全不受它们在初始化表中次序的影响，只有成员对象在类中声明的次序来决定的。

26. 类和对象之间的关系是什么？

答：类是对象的抽象，对象是类的实例。

27. 对类的成员的访问属性有什么？

答：public，protected，private。

28．const char \*p和char \* const p; 的区别

答：  
如果const位于星号的左侧，则const就是用来修饰指针所指向的变量，即指针指向为常量；  
如果const位于星号的右侧，const就是修饰指针本身，即指针本身是常量。

29. 是不是一个父类写了一个virtual 函数，如果子类覆盖它的函数不加virtual ,也能实现多态?

答：  
virtual修饰符会被隐形继承的。  
virtual可加可不加,子类覆盖它的函数不加virtual ,也能实现多态。

30. 函数重载是什么意思？它与虚函数的概念有什么区别？

答：函数重载是一个同名函数完成不同的功能，编译系统在编译阶段通过函数参数个数、参数类型不同，函数的返回值来区分该调用哪一个函数，即实现的是静态的多态性。但是记住：不能仅仅通过函数返回值不同来实现函数重载。而虚函数实现的是在基类中通过使用关键字virtual来申明一个函数为虚函数，含义就是该函数的功能可能在将来的派生类中定义或者在基类的基础之上进行扩展，系统只能在运行阶段才能动态决定该调用哪一个函数，所以实现的是动态的多态性。它体现的是一个纵向的概念，也即在基类和派生类间实现。

31. 构造函数和析构函数是否可以被重载,为什么?

答：构造函数可以被重载，析构函数不可以被重载。因为构造函数可以有多个且可以带参数，而析构函数只能有一个，且不能带参数。

32. 如何定义和实现一个类的成员函数为回调函数？

答：

所谓的回调函数，就是预先在系统的对函数进行注册，让系统知道这个函数的存在，以后，当某个事件发生时，再调用这个函数对事件进行响应。  
定义一个类的成员函数时在该函数前加CALLBACK即将其定义为回调函数，函数的实现和普通成员函数没有区别

33. 虚函数是怎么实现的？

答：简单说来使用了[虚函数表](https://zhida.zhihu.com/search?content_id=118356713&content_type=Article&match_order=1&q=%E8%99%9A%E5%87%BD%E6%95%B0%E8%A1%A8&zhida_source=entity).

34. 抽象类不会产生实例，所以不需要有构造函数。

答： 错

35. 从一个模板类可以派生新的模板类，也可以派生非模板类。

答：对

36. main 函数执行以前，还会执行什么代码？

答案：全局对象的构造函数会在main 函数之前执行。

37. 当一个类A 中没有生命任何成员变量与成员函数,这时sizeof(A)的值是多少，如果不是零，请解释一下编译器为什么没有让它为零。（Autodesk）

答案：肯定不是零。举个反例，如果是零的话，声明一个class A[10]对象数组，而每一个对象占用的空间是零，这时就没办法区分A[0],A[1]…了。

38. delete与 delete []区别：

答：delete只会调用一次析构函数，而delete[]会调用每一个成员的析构函数。

39．子类析构时要调用父类的析构函数吗？

答：会调用。析构函数调用的次序是先派生类的析构后基类的析构，也就是说在基类的的析构调用的时候,派生类的信息已经全部销毁了

40. 继承的优缺点。

答：

优点：  
1、类继承是在编译时刻静态定义的，且可直接使用，  
2、类继承可以较方便地改变父类的实现。  
缺点：  
1、因为继承在编译时刻就定义了，所以无法在运行时刻改变从父类继承的实现  
2、父类通常至少定义了子类的部分行为，父类的任何改变都可能影响子类的行为  
3、如果继承下来的实现不适合解决新的问题，则父类必须重写或被其他更适合的类替换。这种依赖关系限制了灵活性并最终限制了复用性。

41. 解释堆和栈的区别。

答：栈区（stack）— 由编译器自动分配释放 ，存放函数的参数值，局部变量的值等。  
堆（heap）一般由程序员分配释放， 若程序员不释放，程序结束时可能由OS回收 。

42. 一个类的构造函数和析构函数什么时候被调用,是否需要手工调用?

答：构造函数在创建类对象的时候被自动调用，析构函数在类对象生命期结束时，由系统自动调用。

43. 何时需要预编译：

答：总是使用不经常改动的大型代码体。  
程序由多个模块组成，所有模块都使用一组标准的包含文件和相同的编译选项。在这种情况下，可以将所有包含文件预编译为一个预编译头。

44. 多态的作用？

答：主要是两个：  
1. 隐藏实现细节，使得代码能够模块化；扩展代码模块，实现代码重用；  
2. 接口重用：为了类在继承和派生的时候，保证使用家族中任一类的实例的某一属性时的正确调用

45. 虚拟函数与普通成员函数的区别？内联函数和构造函数能否为虚拟函数？

答案：  
区别：虚拟函数有virtual关键字，有虚拟指针和虚函数表，虚拟指针就是虚拟函数的接口，而普通成员函数没有。内联函数和构造函数不能为虚拟函数。

46. 构造函数和析构函数的调用顺序? 析构函数为什么要虚拟?

答案：构造函数的调用顺序：基类构造函数—对象成员构造函数—派生类构造函数；析构函数的调用顺序与构造函数相反。析构函数虚拟是为了防止析构不彻底，造成内存的泄漏。

47. C++中类型为private的成员变量可以由哪些函数访问?

答：只可以由本类中的成员函数和友元函数访问

48. 请说出类中private，protect，public三种访问限制类型的区别

答：private是私有类型，只有本类中的成员函数访问;protect是保护型的，本类和继承类可以访问;public是公有类型，任何类都可以访问.

49. 类中成员变量怎么进行初始化？

答：可以通过构造函数的初始化列表或构造函数的函数体实现。

50. 在什么时候需要使用"常引用"？

答：如果既要利用引用提高程序的效率，又要保护传递给函数的数据不在函数中被改变，就应使用常引用。

51. 引用与指针有什么区别？

答 ：  
1) 引用必须被初始化，指针不必。  
2) 引用初始化以后不能被改变，指针可以改变所指的对象。  
3) 不存在指向空值的引用，但是存在指向空值的指针。

52. 描述实时系统的基本特性

答 、在特定时间内完成特定的任务，实时性与可靠性。

54. 全局变量和局部变量在内存中是否有区别？如果有，是什么区别？

答 、全局变量储存在静态数据区，局部变量在堆栈中。

55. [堆栈溢出](https://zhida.zhihu.com/search?content_id=118356713&content_type=Article&match_order=1&q=%E5%A0%86%E6%A0%88%E6%BA%A2%E5%87%BA&zhida_source=entity)一般是由什么原因导致的？

答 、没有回收垃圾资源

56. 什么函数不能声明为虚函数？

答 ：构造函数（constructor）

57. IP地址的编码分为哪俩部分？

答： IP地址由两部分组成，网络号和主机号。

58. 不能做switch()的参数类型是：

答 ：switch的参数不能为实型。

59. 如何引用一个已经定义过的全局变量？

答 ：可以用引用头文件的方式，也可以用extern关键字，如果用引用头文件方式来引用某个在头文件中声明的全局变理，假定你将那个变写错了，那么在编译期间会报错，如果你用extern方式引用时，假定你犯了同样的错误，那么在编译期间不会报错，而在连接期间报错

60. 对于一个频繁使用的短小函数,在C语言中应用什么实现,在C++中应用什么实现?

答 、c用宏定义，c++用inline

61. C++是不是类型安全的？

答案：不是。两个不同类型的指针之间可以强制转换（用reinterpret cast)

62. 当一个类A 中没有生命任何成员变量与成员函数,这时sizeof(A)的值是多少，请解释一下编译器为什么没有让它为零。

答案：为1。举个反例，如果是零的话，声明一个class A[10]对象数组，而每一个对象占用的空间是零，这时就没办法区分A[0],A[1]…了。

63. 简述数组与指针的区别？

答：数组要么在静态存储区被创建（如全局数组），要么在栈上被创建。指针可以随时指向任意类型的内存块。  
(1)修改内容上的区别  
char a[] = "hello";  
a[0] = 'X';  
char \*p = "world"; // 注意p 指向常量字符串  
p[0] = 'X'; // 编译器不能发现该错误，运行时错误  
(2) 用运算符sizeof 可以计算出数组的容量（字节数）。sizeof(p),p 为指针得到的是一个指针变量的字节数，而不是p 所指的内存容量。

64. C++函数中值的传递方式

答：有三种方式：值传递、指针传递、引用传递

65. 内存的分配方式

答：分配方式有三种，  
1、 静态存储区，是在程序编译时就已经分配好的，在整个运行期间都存在，如全局变量、常量。  
2、 栈上分配，函数内的局部变量就是从这分配的，但分配的内存容易有限。  
3、 堆上分配，也称动态分配，如我们用new,malloc分配内存，用delete,free来释放的内存。

66. extern"C"有什么作用？

答：Extern "C"是由Ｃ＋＋提供的一个连接交换指定符号，用于告诉Ｃ＋＋这段代码是Ｃ函数。这是因为C++编译后库中函数名会变得很长，与C生成的不一致，造成Ｃ＋＋不能直接调用C函数，加上extren "c"后，C++就能直接调用C函数了。  
Extern "C"主要使用正规DLL函数的引用和导出 和 在C++包含C函数或C头文件时使用。使用时在前面加上extern "c" 关键字即可。可以用一句话概括extern "C"这个声明的真实目的：实现C++与C及其它语言的混合编程。

67. 用什么函数开启新进程、线程。

答案：

线程：CreateThread/AfxBeginThread等  
进程：CreateProcess等

68. SendMessage和PostMessage有什么区别

答案：SendMessage是阻塞的，等消息被处理后，代码才能走到SendMessage的下一行。PostMessage是非阻塞的，不管消息是否已被处理，代码马上走到PostMessage的下一行。

69. CMemoryState主要功能是什么

答案：查看内存使用情况，解决内存泄露问题。

70. #include <filename.h> 和 #include "filename.h" 有什么区别？

答：对于#include <filename.h> ，编译器从标准库路径开始搜索 filename.h  
对于#include "filename.h" ，编译器从用户的工作路径开始搜索 filename.h

71. 处理器标识#error的目的是什么？

答：编译时输出一条错误信息，并中止继续编译。

72. #if!defined(AFX\_…\_HADE\_H)

#define(AFX\_…\_HADE\_H)

……

#endif作用？

答：防止该头文件被重复引用。

73. 在定义一个宏的时候要注意什么？

答：定义部分的每个形参和整个表达式都必须用括号括起来，以避免不可预料的错误发生

74. 数组在做函数实参的时候会转变为什么类型？

答：数组在做实参时会变成指针类型。

75. 系统会自动打开和关闭的3个标准的文件是？

答：  
(1) 标准输入----键盘---stdin  
(2) 标准输出----显示器---stdout  
(3) 标准出错输出----显示器---stderr

76. .在Win32下 char, int, float, double各占多少位？

答：  
(1) Char 占用8位  
(2) Int 占用32位  
(3) Float 占用32位  
(4) Double 占用64位

77. strcpy()和memcpy()的区别？

答：strcpy()和memcpy()都可以用来拷贝字符串，strcpy()拷贝以'\0'结束，但memcpy()必须指定拷贝的长度。

78. 说明define和const在语法和含义上有什么不同？

答：(1) #define是C语法中定义符号变量的方法，符号常量只是用来表达一个值，在编译阶段符号就被值替换了，它没有类型；  
(2) Const是C++语法中定义常变量的方法，常变量具有变量特性，它具有类型，内存中存在以它命名的存储单元，可以用sizeof测出长度。

79. 说出字符常量和字符串常量的区别，并使用运算符sizeof计算有什么不用？

答：字符常量是指单个字符，字符串常量以'\0'结束，使用运算符sizeof计算多占一字节的存储空间。

80. 简述全局变量的优缺点？

答：全局变量也称为外部变量，它是在函数外部定义的变量，它属于一个源程序文件，它保存上一次被修改后的值，便于数据共享，但不方便管理，易引起意想不到的错误。

81. 总结static的应用和作用？

答：  
（1）函数体内static变量的作用范围为该函数体，不同于auto变量，该变量的内存只被分配一次，因此其值在下次调用时仍维持上次的值；  
（2）在模块内的static全局变量可以被模块内所用函数访问，但不能被模块外其它函数访问；  
（3）在模块内的static函数只可被这一模块内的其它函数调用，这个函数的使用范围被限制在声明它的模块内；  
（4）在类中的static成员变量属于整个类所拥有，对类的所有对象只有一份拷贝；  
（5）在类中的static成员函数属于整个类所拥有，这个函数不接收this指针，因而只能访问类的static成员变量。

82. 总结const的应用和作用？

答：  
（1）欲阻止一个变量被改变，可以使用const关键字。在定义该const变量时，通常需要对它进行初始化，因为以后就没有机会再去改变它了；  
（2）对指针来说，可以指定指针本身为const，也可以指定指针所指的数据为const，或二者同时指定为const；  
（3）在一个函数声明中，const可以修饰形参，表明它是一个输入参数，在函数内部不能改变其值；  
（4）对于类的成员函数，若指定其为const类型，则表明其是一个常函数，不能修改类的成员变量；  
（5）对于类的成员函数，有时候必须指定其返回值为const类型，以使得其返回值不为"左值"。

83. 什么是指针？谈谈你对指针的理解？

答：指针是一个变量，该变量专门存放内存地址；  
指针变量的类型取决于其指向的数据类型，在所指数据类型前加\*  
指针变量的特点是它可以访问所指向的内存。

84. 什么是常指针，什么是指向常变量的指针？

答：常指针的含义是该指针所指向的地址不能变，但该地址所指向的内容可以变化，使用常指针可以保证我们的指针不能指向其它的变量。  
指向常变量的指针是指该指针的变量本身的地址可以变化，可以指向其它的变量，但是它所指的内容不可以被修改。指向长变量的指针定义。

85. 函数指针和指针函数的区别？

答：函数指针是指向一个函数入口的指针；  
指针函数是函数的返回值是一个指针类型。

87. 简述Debug版本和Release版本的区别？

答：Debug版本是调试版本，Release版本是发布给用户的最终非调试的版本，

88. 指针的几种典型应用情况？

答：

int \*p[n];-----指针数组，每个元素均为指向整型数据的指针。  
int (\*)p[n];---p为指向一维数组的指针，这个一维数组有n个整型数据。  
int \*p();------函数带回指针，指针指向返回的值。  
int (\*)p();----p为指向函数的指针。

89. static函数与普通函数有什么区别?

答：static函数在内存中只有一份，普通函数在每个被调用中维持一份拷贝

90. struct(结构) 和 union(联合)的区别？

答：1. 结构和联合都是由多个不同的数据类型成员组成, 但在任何同一时刻, 联合中只存放了一个被选中的成员（所有成员共用一块地址空间）, 而结构的所有成员都存在（不同成员的存放地址不同）。  
2. 对于联合的不同成员赋值, 将会对其它成员重写, 原来成员的值就不存在了, 而对于结构的不同成员赋值是互不影响的。

91. class 和 struct 的区别？

答：struct 的成员默认是公有的，而类的成员默认是私有的。

92. 简述枚举类型？

答：枚举方便一次定义一组常量，使用起来很方便；

93. ASSERT()的作用？

答：ASSERT()是一个调试程序时经常使用的宏，在程序运行时它计算括号内的表达式，如果表达式为FALSE (0), 程序将报告错误，并终止执行。如果表达式不为0，则继续执行后面的语句。这个宏通常原来判断程序中是否出现了明显非法的数据，如果出现了终止程序以免导致严重后果，同时也便于查找错误。

94. 局部变量和全局变量是否可以同名？

答：能。局部会屏蔽全局。要用全局变量，需要使用"::"(域运算符)。

95. 程序的不同变量的不同存储位置

答：局部变量存在于（堆栈）中，全局变量存在于（静态区 ）中，动态申请数据存在于（ 堆）中。

96. 在什么时候使用常引用？

答：如果既要利用引用提高程序的效率，又要保护传递给函数的数据不在函数中被改变，就应使用常引用。

97. 类的声明和实现的分开的好处？

答：  
1. 起保护作用；  
2. 提高编译的效率。

98. windows消息系统由哪几部分构成？

答：由一下3部分组成：  
1. 消息队列：操作系统负责为进程维护一个消息队列，程序运行时不断从该消息队列中获取消息、处理消息；  
2. 消息循环：应用程序通过消息循环不断获取消息、处理消息。  
3. 消息处理：消息循环负责将消息派发到相关的窗口上使用关联的窗口过程函数进行处理。

99. 什么是消息映射？

答：消息映射就是让程序员指定MFC类（有消息处理能力的类）处理某个消息。然后由程序员完成对该处理函数的编写，以实现消息处理功能。

100.什么叫智能指针？

答：当一个类中，存在一个指向另一个类对象的指针时，对指针运算符进行重载，那么当前类对象可以通过指针像调用自身成员一样调用另一个类的成员。

101. winsock建立连接的主要实现步骤?

答：  
服务器端：socket()建立套接字，绑定（bind）并监听（listen），用accept（）等待客户端连接, accept（）发现有客户端连接，建立一个新的套接字，自身重新开始等待连接。该新产生的套接字使用send()和recv（）写读数据，直至数据交换完毕，closesocket()关闭套接字。  
客户端：socket()建立套接字，连接（connect）服务器，连接上后使用send()和recv（），在套接字上写读数据，直至数据交换完毕，closesocket()关闭套接字。

102.多态的两个必要条件

答：1.一个基类的指针或引用指向一个派生类对象 2.虚函数

103. 构成Win32 API 函数的三个动态链接库是什么？

答：内核库，用户界面管理库，图形设备界面库。

104.什么时候需要用虚析构函数？

答：当基类指针指向用new运算符生成的派生类对象时，delete基类指针时，派生类部分没有释放掉而造成释放不彻底现象，需要虚析构函数。 补充：虚函数就是让派生类调用基类的虚函数。

105. 什么是平衡二叉树？

答：左右子树都是平衡二叉树，而且左右子树的深度差值的约对值不大于1。

106. 派生新类的过程要经历三个步骤

答：1.吸收基类成员 2.改造基类成员 3.添加新成员

107. 简单介绍GDI？

答： GDI是Graphics Device Interface 的缩写，译为：图形设备接口；是一个在Windows应用程序中执行与设备无关的函数库，这些函数在不同的输出设备上产生图形以及文字输出。

108.在C++ 程序中调用被 C 编译器编译后的函数，为什么要加 extern "C"？

答：C++语言支持函数重载，C 语言不支持函数重载。函数被C++编译后在库中的名字  
与C 语言的不同。假设某个函数的原型为： void foo(int x, int y);该函数被C 编译器编译后在库中的名字为\_foo ， 而C++编译器则会产生像\_foo\_int\_int 之类的名字。C++提供了C 连接交换指定符号extern"C"来解决名字匹配问题。

109. 怎样定义一个纯虚函数？含有纯虚函数的类称为什么？

答：在虚函数的后面加=0，含有虚函数的类称为抽象类。

110. 简述Visual C++ 、Win32 API和MFC之间的关系？

答：(1) Visual C+是一个以C++程序设计语言为基础的、集成的、可视化的编程环境；  
(2) Win32 API是32位Windows操作系以C/C++形式提供的一组应用程序接口；  
(3) MFC是对Win32 API的封装，简化了开发过程。

111. 怎样消除多重继承中的二义性？

答： 1．成员限定符 2．虚基类

112. 什么叫静态关联，什么叫动态关联

答：在多态中，如果程序在编译阶段就能确定实际执行动作，则称静态关联，  
如果等到程序运行才能确定叫动态关联。