C++概念题汇总

1. C 中 static 有什么作用？

（1）隐藏。 当我们同时编译多个文件时，所有未加 static 前缀的全局变量和函数都具有全局可见性,故使用 static 在不同的文件中定义同名函数和同名变量，而不必担心命名冲突。

（2）static 的第二个作用是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。

（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是 0x00，某些时候这一特点可以减少程序员的工作量。

2. C++中 const 有什么用？

不要一听到 const 就说是常量， 这样给考官一种在和一个外行交谈的感觉。应该说 const 修饰的内容不可改变就行了， 定义常量只是一种使用方式而已，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. free 不会调用对象的 destructor.

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. 什么是“引用”？申明和使用“引用”要注意哪些问题？
答：引用就是某个目标变量的“别名”(alias)，对应用的操作与对变量直接操作效果完全相同。申明一个引用的时候，切记要对其进行初始化。引用声明完毕后，相当于目标变量名有两个名称，即该目标原名称和引用名，不能再把该引用名作为其他变量名的别名。声明一个引用，不是新定义了一个变量，它只表示该引用名是目标变量名的一个别名，它本身不是一种数据类型，因此引用本身不占存储单元，系统也不给引用分配存储单元。不能建立数组的引用。

17. 将“引用”作为函数参数有哪些特点？
（1）传递引用给函数与传递指针的效果是一样的。这时，被调函数的形参就成为原来主调函数中的实参变量或对象的一个别名来使用，所以在被调函数中对形参变量的操作就是对其相应的目标对象（在主调函数中）的操作。
（2）使用引用传递函数的参数，在内存中并没有产生实参的副本，它是直接对实参操作；而使用一般变量传递函数的参数，当发生函数调用时，需要给形参分配存储单元，形参变量是实参变量的副本；如果传递的是对象，还将调用拷贝构造函数。因此，当参数传递的数据较大时，用引用比用一般变量传递参数的效率和所占空间都好。
（3）使用指针作为函数的参数虽然也能达到与使用引用的效果，但是，在被调函数中同样要给形参分配存储单元，且需要重复使用"*指针变量名"的形式进行运算，这很容易产生错误且程序的阅读性较差；另一方面，在主调函数的调用点处，必须用变量的地址作为实参。而引用更容易使用，更清晰。
4. 在什么时候需要使用“常引用”？
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。常引用声明方式：const 类型标识符 &引用名=目标变量名；引用名是目标变量名的一个别名，它本身不是一种数据类型，因此引用本身不占存储单元，系统也不给引用分配存储单元。不能建立数组的引用。

18. 将“引用”作为函数参数有哪些特点？
（1）传递引用给函数与传递指针的效果是一样的。这时，被调函数的形参就成为原来主调函数中的实参变量或对象的一个别名来使用，所以在被调函数中对形参变量的操作就是对其相应的目标对象（在主调函数中）的操作。
（2）使用引用传递函数的参数，在内存中并没有产生实参的副本，它是直接对实参操作；而使用一般变量传递函数的参数，当发生函数调用时，需要给形参分配存储单元，形参变量是实参变量的副本；如果传递的是对象，还将调用拷贝构造函数。因此，当参数传递的数据较大时，用引用比用一般变量传递参数的效率和所占空间都好。
（3）使用指针作为函数的参数虽然也能达到与使用引用的效果，但是，在被调函数中同样要给形参分配存储单元，且需要重复使用"*指针变量名"的形式进行

运算，这很容易产生错误且程序的阅读性较差；另一方面，在主调函数的调用点处，必须用变量的地址作为实参。而引用更容易使用，更清晰。

19. 在什么时候需要使用“常引用”？

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。常引用声明方式：const 类型标识符 &引用名=目标变量名；

20. 什么时候需要“引用”？

流操作符<<和>>、赋值操作符=的返回值、拷贝构造函数的参数、赋值操作符=的参数、其它情况都推荐使用引用。

21. 结构与联合有和区别？

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。

22. .h 头文件中的

ifndef/define/endif 的作用？

答：防止该头文件被重复引用。

23. #i nclude 与#i nclude "file.h"的区别？

答：前者是从

Standard Library 的路径寻找和引用

file.h，而后者是从当前工

作路径搜寻并引用

file.h。

24.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：实现 C++与 C 及其它语言的混合编程。

25. 面向对象的三个基本特征，并简单叙述之？

答：1. 封装：将客观事物抽象成类，每个类对自身的数据和方法实行保护(private,protected,public)

2. 继承：广义的继承有三种实现形式：实现继承（指使用基类的属性和方法而无需额外编码的能力）、可视继承（子窗体使用父窗体的外观和实现代码）、接口继承（仅使用属性和方法，实现滞后到子类实现）。前两种（类继承）和后一种（对象组合=>接口继承以及纯虚函数）构成了功能复用的两种方式。

3. 多态：是将父对象设置成为和一个或更多的他的子对象相等的技术，赋值之后，父对象就可以根据当前赋值给它的子对象的特性以不同的方式运作。简单的说，就是一句话：允许将

子类类型的指针赋值给父类类型的指针。

26. 重载（overload)和重写(overried，有的书也叫做 “覆盖”）的区别?
答：常考的题目。从定义上来说：
重载：是指允许存在多个同名函数，而这些函数的参数表不同（或许参数个数不同，或许参数类型不同，或许两者都不同)。

重写：是指子类重新定义复类虚函数的方法。

从实现原理上来说：

重载：编译器根据函数不同的参数表，对同名函数的名称做修饰，然后这些同名函数就成了不同的函数（至少对于编译器来说是这样的)。如，有两个同名函数：function func(p:integer):integer;和 function func(p:string):integer;。那么编译器做过修饰后的函数名称可能是这样的：int_func、str_func。对于这两个函数的调用，在编译器间就已经确定了，是静态的。也就是说，它们的地址在编译期就绑定了（早绑定)，因此，重载和多态无关!

重写：和多态真正相关。当子类重新定义了父类的虚函数后，父类指针根据赋给它的不同的子类指针，动态的调用属于子类的该函数，这样的函数调用在编译期间是无法确定的（调用的子类的虚函数的地址无法给出)。因此，这样的函数地址是在运行期绑定的（晚绑定)。

27. 多态的作用?
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。

28. New delete 与 malloc free 的联系与区别 ?
答案：都是在堆(heap)上进行动态的内存操作。用 malloc 函数需要指定内存分配的字节数并
且不能初始化对象，new 会自动调用对象的构造函数。delete 会调用对象的 destructor，而 free 不会调用对象的 destructor.

29. 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)。C#是类型安全的。

30. 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。

31. 描述内存分配方式以及它们的区别?
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。例如全局变量，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。
3）从堆上分配，亦称动态内存分配。程序在运行的时候用
malloc 或
new 申请任意多少的
内存，程序员自己负责在何时用
free 或
delete 释放内存。动态内存的生存期由程序员决定，
使用非常灵活，但问题也最多。

32.struct 和 class 的区别
答案：struct 的成员默认是公有的，而类的成员默认是私有的。struct 和 class 在其他方面
是功能相当的。

从感情上讲，大多数的开发者感到类和结构有很大的差别。感觉上结构仅仅象一堆缺乏封装和功能的开放的内存位，而类就象活的并且可靠的社会成员，它有智能服务，有牢固的封装屏障和一个良好定义的接口。既然大多数人都这么认为，那么只有在你的类有很少的方法并且有公有数据(这种事情在良好设计的系统中是存在的 !)时，你也许应该使用 struct 关键字，否则，你应该使用 class 关键字。

33.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。

34. 在 8086 汇编下，逻辑地址和物理地址是怎样转换的？ (Intel)
答案：通用寄存器给出的地址，是段内偏移地址，相应段寄存器地址*10H+通用寄存器内地址，就得到了真正要访问的地址。

35.请说出 const 与#define 相比，有何优点？
答案：1） const 常量有数据类型，而宏常量没有数据类型。编译器可以对前者进行类型安全检查。而对后者只进行字符替换，没有类型安全检查，并且在字符替换可能会产生意料不到的错误。
2）有些集成化的调试工具可以对 const 常量进行调试，但是不能对宏常量进行调试。

36.简述数组与指针的区别？
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。

37.类成员函数的重载、覆盖和隐藏区别？
答案：

a.成员函数被重载的特征：
（1）相同的范围（在同一个类中）；
（2）函数名字相同；
（3）参数不同；
（4）virtual 关键字可有可无。
b.覆盖是指派生类函数覆盖基类函数，特征是：
（1）不同的范围（分别位于派生类与基类）；
（2）函数名字相同；
（3）参数相同；
（4）基类函数必须有
virtual 关键字。
c.“隐藏”是指派生类的函数屏蔽了与其同名的基类函数，规则如下：
（1）如果派生类的函数与基类的函数同名，但是参数不同。此时，不论有无
virtual 关键字，
基类的函数将被隐藏（注意别与重载混淆）。
（2）如果派生类的函数与基类的函数同名，并且参数也相同，但是基类函数没有
virtual 关
键字。此时，基类的函数被隐藏（注意别与覆盖混淆）

38. There are two int variables: a and b, don’t use “if”, “? :”, “switch”
or other judgement
statements, find out the biggest one of the two numbers.
答案：((a+ b)+ abs(a - b)) / 2

39. 如何判断一段程序是由
C 编译程序还是由
C++编译程序编译的？
答案：

```
#ifdef __cplusplus
cout<<"c++";
#else
cout<<"c";
#endif
```

45. 如何判断一个单链表是有环的？（注意不能用标志位，最多只能用两个额外指针）

```
struct node { char val; node* next;}

bool check(const node* head) {} //return false : 无环；true: 有环
```

一种 O（n）的办法就是（搞两个指针，一个每次递增一步，一个每次递增两步，如果有环的话两者必然重合，反之亦然）：

```
bool check(const node* head)
{
if(head==NULL) return false;
node *low=head, *fast=head->next;
while(fast!=NULL && fast->next!=NULL)
{
low=low->next;
fast=fast->next->next;
if(low==fast) return true;
}
return false;
}
```