C++面试题-1

1、C和C++的区别

1. C 是面向过程的语言，是一个结构化的语言，考虑如何通过一个过程对输入进行处理得到输出；C++ 是面向对象的语言，主要特征是“封装、继承和多态”。封装隐藏了实现细节，使得代码模块化；派生类可以继承父类的数据和方法，扩展了已经存在的模块，实现了代码重用；多态则是 “一个接口，多种实现”，通过派生类重写父类的虚函数，实现了接口的重用
2. C 和 C++ 动态管理内存的方法不一样，C 是使用 malloc/free，而 C++ 除此之外还有 new/delete 关键字。
3. C++ 中有引用，C 中不存在引用的概念

2、C++中指针和引用的区别

1. 指针有自己的一块空间，而引用只是一个别名
2. 使用sizeof看一个指针的大小为4字节（32位，如果是64位的话指针为8字节），而sizeof引用则是被引用对象的大小
3. 指针可以被初始化为NULL，而引用必须被初始化且必须是一个已有对象的引用
4. 作为参数传递时，指针需要被解引用才可以对对象进行操作，而直接对引用的修改会改变引用所指向的对象
5. 指针在使用中可以指向其他对象，但是引用只能是一个对象的引用，不能被改变
6. 指针可以是多级，而引用没有分级
7. 如果返回动态分配内存的对象或者内存，必须使用指针，引用可能会引起内存泄漏

> 引用占用内存空间吗?
>
> 对引用取地址，其实是取引用所对应的内存空间的地址，这让人觉得引用好像并非是一个实体，但是引用其实是占据内存空间的，而且其占用的内存和指针一样，因为引用的内部实现就是通过指针来完成的

3、结构体struct和共同体union的区别

结构体：将不同类型的数据组合成一个整体，是自定义类型

共同体：不同类型的几个变量共同占用一段内存

1. 结构体中的每个成员都有自己独立的地址，它们是同时存在的；共同体中的所有成员占用同一段内存，它们不能同时存在
2. sizeof(struct)是内存对齐后所有成员长度的综合，sizeof(union)是内存对齐后最长数据成员的长度

> 结构体为什么要内存对其呢?
>
> 1. 平台原因（移植问题）：不是所有的硬件平台都能访问任意地址上的任意数据，某些硬件平台只能在某些地址处取某些特定类型的数据，否则抛出硬件异常
> 2. 硬件原因：经过内存对齐之后，CPU的内存访问速度大大提升

4、struct和class的区别

1. 内部成员变量及成员函数的默认访问属性：struct默认是public的，而class默认是private的
2. 继承关系中默认访问属性的区别：在集成关系，struct默认是public的，而class是private的
3. class关键字还可以用于定义模板参数，等同于typename，而struct不用定义模板参数

5、#define和const的区别

1. #define定义的常量没有类型，所给出的是一个立即数；const定义的常量有类型名字，存放在静态区域
2. 处理阶段不同，#define定义的宏变量在预处理时进行替换，可能有多个拷贝，const所定义的变量在编译时确定其值，只有一个拷贝
3. #define定义的常量是不可以用指针去指向，const定义的常量可以用指针去指向该常量的地址
4. #define可以定义简单的函数，const不可以定义函数

6、重载overload，覆盖重写override，隐藏重定义overwrite这三者的区别

1. overload:将语义相近的几个函数用同一个名字表示，但是参数列表（参数的类型，个数，顺序不同）不同，这就是函数重载，返回值类型可以不同

   特征：相同范围（同一个类中）、函数名字相同、参数不同、virtual关键字可有可无

2. override，派生类覆盖基类的虚函数，实现接口的重用，返回值类型必须相同

   特征：不同范围（积累和派生类）、函数名字相同、参数相同、基类中必须有virtual关键字（必须是虚函数）

3. overwrite，派生类屏蔽了其同名的基类函数，返回值类型可以不同

   特征：不同范围（基类和派生类）、函数名字相同、参数不同或者参数相同且无virtual关键字

7、new和delete是如何实现的，与malloc和free有何异同?

new操作针对数据类型的处理，分为两种情况：

1. 简单数据类型（包括基本数据类型和不需要构造函数的类型）简单类型直接调用operator new分配内存；可以通过new_handler来处理new失败的情况;new分配失败的时候不像malloc那样返回NULL，它直接抛出异常(bad_alloc)。要判断是否分配成功应该用异常捕获的机制
2. 复杂数据类型（需要由构造函数初始化对象）new复杂数据类型的时候先调用operator new，然后在分配的内存上调用构造函数

delete也分为两种情况：

1. 简单数据类型（包括基本数据类型和不需要析构函数的类型）delete简单数据类型默认只是调用free函数
2. 复杂数据类型（需要由析构函数销毁对象）delete复杂数据类型先调用析构函数再调用operator delete

> 与malloc和free的区别：
>
> 1. 属性上：new/delete是C++关键字，需要编译器支持，malloc/free是库函数，需要C的头文件支持
> 2. 参数：使用new操作符申请内存分配时无须指定内存块的大小，编译器会根据类型信息自行计算，而malloc则需要显式地指出所需内存的尺寸
> 3. 返回类型：new操作符内存分配成功时，返回的是对象类型的指针，类型严格与对象匹配，故new是符合类型安全性的操作符，而malloc内存成功分配返回的是void *，需要通过类型转换将其转换为我们需要的类型
> 4. 分配失败时：new内存分配失败时抛出bad_alloc异常，malloc分配内存失败时返回NULL
> 5. 自定义类型：new会先调用operator new函数，申请足够的内存(通常底层使用malloc实现)，然后调用类型的构造函数，初始化成员变量，最后返回自定义类型指针。delete先调用析构函数，然后调用operator delete函数释放内存（通常底层使用free实现）。malloc/free是库函数，只能动态的申请和释放内存，无法强制要求其做自定义类型对象构造和析构工作
> 6. 重载：C++允许重载new/delete操作符，而malloc为库函数不允许重载
> 7. 内存区域：new 操作符从自由存储区（free store）上为对象动态分配内存空间，而 malloc 函数从堆上动态分配内存。其中自由存储区为：C++ 基于 new 操作符的一个抽象概念，凡是通过 new 操作符进行内存申请，该内存即为自由存储区。而堆是操作系统中的术语，是操作系统所维护的一块特殊内存，用于程序的内存动态分配，C 语言使用 malloc 从堆上分配内存，使用 free 释放已分配的对应内存。自由存储区不等于堆，如上所述，布局 new 就可以不位于堆中。、

> 既然有了 malloc/free，C++ 中为什么还需要 new/delete 呢?
>
> **运算符是语言自身的特性，有固定的语义，编译器知道意味着什么，由编译器解释语义，生成相应的代码。库函数是依赖于库的，一定程度上独立于语言的。编译器不关心库函数的作用，只保证编译，调用函数参数和返回值符合语法，生成 call 函数的代码。 对于非内部数据**

> **类型而言，光用 malloc/free 无法满足动态对象都要求。new/delete 是运算符，编译器保证调用构造和析构函数对对象进行初始化 / 析构。但是库函数 malloc/free 是库函数，不会执行构造 / 析构。**

8、delete和delete[]的区别

delete只会调用一次析构函数，而deleet[]会调用每个成员的析构函数用new分配的内存用delete释放，用new[]分配的内存用delete[]释放

9、const知道吗？解释一下其作用

const修饰类的成员变量，表示常量不可能被修改

const修饰类的成员函数，表示该函数不会修改类中的数据成员，不会调用其他非const的成员函数

const函数只能调用const函数，非const函数可以调用const函数

10、关键字static的作用

1. 函数体内：static修饰的局部变量作用范围为该函数体内，不同于auto变量，其内存只被分配一次，因此其值在下次调用的时候维持了上次的值
2. 模块内：static修饰全局变量或全局函数，可以被模块内的所有函数访问，但是不能被模块外的其他函数访问，使用范围限制在声明它的模块内
3. 类中：修饰成员变量，表示该变量属于整个类所有，对类的所有对象只有一份拷贝
4. 类中：修饰成员函数，表示该函数术语整个类所有，不接受this指针，只能访问类中的static成员变量 注意其和const的区别！

   const强调值不能被修改，而static强调唯一的拷贝，对所有类的对象。