C++八股文

叶茂林 2024年1月12日星期五

**C和C++有什么区别？**

C和C++在一般语句上没有什么区别，但是基本上属于两种编程语言，C++有新增的语法和关键字，增加了命名空间，管理内存方面增加了new和delete，在指针的基础上增加了引用，C++的不同版本还增加了一些新的关键字，C++还增加了函数重载

相对于C的结构体，C++增加了用于实现面向对象的类，出现了成员函数，并且提供了访问权限的控制，并在此基础上通过继承和虚函数实现了多态

最后就是C++中增加了模板，提供了强大的STL标准容器库

**C++和Java有什么区别？**

Java没有指针，而C++允许通过指针操纵内存

Java不支持多重继承，但Java有接口，可以通过一个类继承多个接口来达到C++多重继承的效果

Java是完全面向对象的语言，所有的函数和变量都必须是某个类的成员

Java有垃圾回收机制可以自动回收无用内存，C++则要通过智能指针才能实现内存自动回收

**C++多态的实现**

C++多态分成静态多态和动态多态

静态多态就是函数重载，这是在编译的时候可以根据函数参数列表来决定调用哪个函数

动态多态则是通过类继承+虚函数实现的

**虚函数实现原理**

虚函数是通过虚函数指针和虚函数表实现的，当一个类声明了虚函数的时候，编译器就会为这个类生成一个虚函数指针，这个虚函数指针会指向一个虚函数表，这个虚函数表是一个指向各个虚函数的指针数组，每个指针指向一个虚函数，当类继承的时候会同时继承这个虚函数表，如果重写虚函数那么将会覆盖这个虚函数表中对应的虚函数指针，当这个类生成对象实例的时候，这个对象会多一个虚函数指针指向类的虚函数表，在通过基类指针调用虚函数的时候就会根据这个虚函数指针调用对应的虚函数。

**析构函数为什么要写成虚函数**

这是因为当使用指向派生类对象的基类指针来销毁对象的时候，如果析构函数不是虚函数，那么就会调用基类的析构函数，而不是派生类的，那这就不对了

**New、delete和malloc、free的区别？**

New和delete是C++的运算符，malloc和free是C的标准库里面的函数，malloc和free用来分配和释放内存，new和delete除了完成分配和释放内存还会自动调用对象的构造函数和析构函数。

**多次delete同一个对象会怎么样？**

Delete对象会调用对象的析构函数进行内存的释放，指向该对象的指针会变成野指针，如果不将野指针赋值为空指针的话，再次delete会出现未定义的行为。

**Delete空指针会怎么样？**

Delete空指针是安全的，delete内部有对指针判空，不会怎么样。

**C++什么时候会出现内存泄漏？**

申请的内存用完没有释放，发生异常跳过释放内存的代码，容器用完没有释放，delete与delete[]用混，还有就是share指针相互引用导致析构的时候引用无法降为0出现的内存泄露。

**说一下C++的四种智能指针？**

C++的四种智能指针包括C++11新增的unique\_ptr、share\_ptr、weak\_ptr和C++11抛弃使用的auto\_ptr。

auto和unique\_ptr一样是独占所有权的，保证同一时刻只有一个智能指针可以指向该对象，auto在转换资源所有权的时候是通过reset和release来重载赋值运算符实现的，先是release将auto本指针赋值为了0，然后reset将资源转换给新的指针，这个时候如果再通过原来的指针进行资源的访问就会出现访问空指针的问题，会触发未定义行为，而unique不允许直接复制，需要move移动赋值。

Share\_ptr是共享式的，允许多个智能指针指向相同的资源，每次构造和拷贝和赋值的时候引用计数会加一，当析构的时候会引用减一，引用为0的时候会释放资源。

然而当share指针相互引用的时候会出现析构的时候引用无法降为0的问题，weak\_ptr就是用来解决这个问题的，weak指针是弱引用，不会增加资源的引用计数。

**指针和引用有什么区别？**

指针本质是一个变量，这个变量里面存储的是某个变量的内存地址，而引用是一个已经存在的变量的别名，具体反映在编译的时候它们两个的符号表的值是不一样的，指针符号对应的值是这个指针变量的地址，而引用符号对应的是引用变量的地址，所以这个指针是可以为空的，因为它本质是一个变量，但是引用不能为空，它的存在必须依赖于已经存在的变量。

**指针传参和引用传参的区别？**

先解释指针和引用的区别，虽然两种传参都可以修改实参，但是指针传参实际还是值传递，将变量的地址传递到参数上，而引用传参直接传递参数本身，并且这个指针传参可以传空指针，但是引用传参不能传空引用，而且在函数内部修改指针参数不会对外部指针造成影响。