### C++面试题

1. **const的所有用法**

修饰变量：表示变量值不可变

修饰指针：

int arr[] = {1, 2, 3, 4, 5};

const int\* arr\_a = arr; 表示指针指向的内容不可改变

int\* const arr\_b = arr; 表示指针不可指向其他地方，但指针内容可变

const int\* const arr\_c = arr; 指针指向与被指内容都不可变

修饰引用：

void test\_const(const int& a) { return; }

表示形参内容不可改变，避免了对a的复制及函数对a的修改

修饰类成员函数：

class TestConst {

private:

const int a; // 只有在初始化的时候才能赋值，其他时候不能变

public:

int geta() const; //常成员函数，不得修改类中的任何数据成员的值

int geta(); //普通成员函数

};

1. **static关键字**

修饰普通变量：存储在静态区，main执行之前就分配了存储空间。

修饰普通函数：表明函数的定义范围，尽在定义文件内可以使用。

修饰类成员变量：成为类变量。

修饰成员函数：不需要生成对象，就可以访问，静态函数内不能访问非静态变量。

1. **this指针**

i 隐含于非静态成员函数内的特殊指针，指向调用该函数的对象。

ii当对一个对象调用成员函数时，编译程序先将对象的地址赋给 this 指针，然后调用成员函数，每次成员函数存取数据成员时，都隐含使用 this 指针

iii当一个成员函数被调用时，自动向它传递一个隐含的参数，该参数是一个指向这个成员函数所在的对象的指针

iv this 指针被隐含地声明为: ClassName \*const this，这意味着不能给 this 指针赋值；在 ClassName 类的 const 成员函数中，this 指针的类型为：const ClassName\* const，这说明不能对 this 指针所指向的这种对象是不可修改的（即不能对这种对象的数据成员进行赋值操作）；

v this 并不是一个常规变量，而是个右值，所以不能取得 this 的地址（不能 &this）。

在以下场景中，经常需要显式引用 this 指针：为实现对象的链式引用；为避免对同一对象进行赋值操作；在实现一些数据结构时，如 list。

1. **inline内联函数**

相当于把内联函数里面的内容写在调用内联函数处；不用执行进入函数的步骤，直接执行函数体；比宏多了类型检查，真正具有函数特性；编译器一般不内联包含循环、递归、switch 等复杂操作的内联函数；在类声明中定义的函数，除了虚函数的其他函数都会自动隐式地当成内联函数。

1. **volatile**
2. **assert()**
3. **sizeof()**
4. **#pragma pack(n)**
5. **位域**
6. **extern “C”**
7. **struct和 typedef struct**
8. **struct 和class**
9. **c实现c++类**
10. **explicit关键字**
11. **friend友元类和友元函数**
12. **using**
13. **::范围解释符**
14. **enum枚举类型**
15. **decltype**
16. **引用**
17. **宏**
18. **成员初始化列表**
19. **initializer\_list初始化列表**
20. **面向对象**
21. **虚析构函数**
22. **纯虚函数**
23. **虚继承与虚函数**
24. **模版类、成员模版、虚函数**
25. **抽象类、接口类、聚合类**
26. **内存分配和管理**
27. **delete this合法么**
28. **如何定义一个只在堆上或栈上生成的对象类**
29. **智能指针**
30. **强制类型转换运算符**
31. **运行时类型信息RTTI**
32. **-91.effective c++**

**92.STL各容器实现**