

- 1 `#include <iostream>`
 - 1.1 `using namespace std;`
 - 1.2 `cout << "hello .." << endl;`
 - 1.3 `system("pause")`
 - 1.4 `retrun 0`
- 2 `::`双冒号作用域运算符
 - 2.1 全局作用域 直接加`::`
- 3 `namespace` 命名空间
 - 3.1 用途 解决名称冲突问题
 - 3.2 必须在全局作用域下声明
 - 3.3 命名空间下可以放入 函数、变量、结构体、类...
 - 3.4 命名空间可以嵌套命名空间
 - 3.5 命名空间是开放的，可以随时加入新的成员
 - 3.6 匿名命名空间 `static`
 - 3.7 可以起别名
- 4 `using` 声明和 `using` 编译指令
 - 4.1 `using LOL:: sunwukongID;`
 - 4.2 如果局部范围内还有 `sunwukongID`，会出现二义性问题，要注意避免
 - 4.3 编译指令
 - 4.4 `using namespace LOL`
 - 4.5 如果局部范围内还有 `sunwukongID`，使用局部的 `ID`
 - 4.6 如果打开多个房间，那么也要注意二义性问题
- 5 C++对 C 语言增强
 - 5.1 全局变量检测增强
 - 5.2 函数检测增强
 - 5.2.1 参数类型检测
 - 5.2.2 返回值检测
 - 5.2.3 传参个数检测
 - 5.3 类型转换检测增强
 - 5.3.1 `malloc` 返回 `void*`，C 中可以不用强转，C++必须强转
 - 5.4 `struct` 增强
 - 5.4.1 C 中不许有函数 C++可以
 - 5.4.2 使用 C 必须加关键字 `struct`，C++可以不加
 - 5.5 `bool` 数据类型增强
 - 5.5.1 C 没有 C++有
 - 5.5.2 `true` 真 `false` 假
 - 5.5.3 `sizeof 1`
 - 5.6 三目运算符增强
 - 5.6.1 C 中返回的是值
 - 5.6.2 C++中返回的是变量
 - 5.7 `const` 增强
 - 5.7.1 C 语言中 `const` 是伪常量，可以通过指针修改
 - 5.7.2 C++中 `const` 会放入到符号表中
 - 5.7.3 C 语言中 `const` 默认是外部链接，C++中 `const` 默认是内部链接

- 5.7.4 const 分配内存情况
 - 5.7.4.1 对变量取地址，会分配临时内存
 - 5.7.4.2 extern 关键字下的 const 会分配内存
 - 5.7.4.3 用普通变量初始化 const 变量
 - 5.7.4.4 自定义数据类型会分配内存
- 5.7.5 尽量用 const 代替 define
 - 5.7.5.1 define 宏没有作用域概念
 - 5.7.5.2 define 宏常量没有类型
- 6 引用基本语法
 - 6.1.1 用途起别名
 - 6.1.2 Type &别名 = 原名
 - 6.1.3 引用必须初始化
 - 6.1.4 一旦初始化后 不能修改
 - 6.1.5 对数组建立引用
- 6.2 参数 3 种传递方式
 - 6.2.1 值传递
 - 6.2.2 地址传递
 - 6.2.3 引用传递
- 6.3 注意事项，不要返回局部变量的引用
- 6.4 如果函数返回值是引用，那么函数的调用可以作为左值
- 6.5 引用的本质 就是一个指针常量
- 7 指针的引用
 - 7.1 用一级指针引用 可以代替二级指针
- 8 常量引用
 - 8.1 使用场景 修饰形参为只读
 - 8.2 const int &a = 10;会分配内存