1. #include <iostream>
   1. using namespace std;
   2. cout << “hello ..” << endl;
   3. system(“pause”)
   4. retrun 0
2. ::双冒号作用域运算符
   1. 全局作用域 直接加：：
3. namespace 命名空间
   1. 用途 解决名称冲突问题
   2. 必须在全局作用域下声明
   3. 命名空间下可以放入 函数、变量、结构体、类…
   4. 命名空间可以嵌套命名空间
   5. 命名空间是开放的，可以随时加入新的成员
   6. 匿名命名空间 static
   7. 可以起别名
4. using声明和using编译指令
   1. using LOL:: sunwukongID;
   2. 如果局部范围内还有 sunwukongID，会出现二义性问题，要注意避免
   3. 编译指令
   4. using namespace LOL
   5. 如果局部范围内还有 sunwukongID ,使用局部的ID
   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
   1. 全局变量检测增强
   2. 函数检测增强
      1. 参数类型检测
      2. 返回值检测
      3. 传参个数检测
   3. 类型转换检测增强
      1. malloc返回void\* ，C中可以不用强转，C++必须强转
   4. struct增强
      1. C中不许有函数 C++可以
      2. 使用C必须加关键字 struct ，C++可以不加
   5. bool数据类型增强
      1. C没有 C++有
      2. true 真 false假
      3. sizeof 1
   6. 三目运算符增强
      1. C中返回的是值
      2. C++中返回的是变量
   7. const增强
      1. C语言中const是伪常量，可以通过指针修改
      2. C++中const会放入到符号表中
      3. C语言中const默认是外部链接，C++中const默认是内部链接
      4. const分配内存情况
         1. 对变量取地址，会分配临时内存
         2. extern关键字下的const会分配内存
         3. 用普通变量初始化const变量
         4. 自定义数据类型会分配内存
      5. 尽量用const代替define
         1. define宏没有作用域概念
         2. define宏常量没有类型
6. 引用基本语法
   * 1. 用途起别名
     2. Type &别名 = 原名
     3. 引用必须初始化
     4. 一旦初始化后 不能修改
     5. 对数组建立引用
   1. 参数3种传递方式
      1. 值传递
      2. 地址传递
      3. 引用传递
   2. 注意事项，不要返回局部变量的引用
   3. 如果函数返回值是引用，那么函数的调用可以作为左值
   4. 引用的本质 就是一个指针常量
7. 指针的引用
   1. 用一级指针引用 可以代替二级指针
8. 常量引用
   1. 使用场景 修饰形参为只读
   2. const int &a = 10;会分配内存