1. C++概述
   1. C++两大编程思想
      1. 面向对象
      2. 泛型编程
   2. 移植性和标准
      1. ANSI 在1998制定出C++第一套标准
2. c++初识
   1. 引入头文件 #include <iostream> 标准输入输出流
   2. 使用标准命名空间 using namespace std;
   3. 标准输出流对象 cout << “..” << 1234 << 3.14 << endl;
   4. 面向对象三大特性
      1. 封装、继承、多态
3. 双冒号作用域运算符
   1. ::代表作用域 如果前面什么都不添加 代表全局作用域
4. namespace命名空间
   1. 命名空间用途：解决名称冲突
   2. 命名空间下可以存放 ： 变量、函数、结构体、类…
   3. 命名空间必须要声明在全局作用域
   4. 命名空间可以嵌套命名空
   5. 命名空间是开放的，可以随时将新成员添加到命名空间下
   6. 命名空间可以匿名的
   7. 命名空间可以起别名
5. using声明以及using编译指令
   1. using声明
      1. using KingGlory::sunwukongId
      2. 当using声明与 就近原则同时出现，出错，尽量避免
   2. using编译指令
      1. using namespace KingGlory;
      2. 当using编译指令 与 就近原则同时出现，优先使用就近
      3. 当using编译指令有多个，需要加作用域 区分
6. C++对C语言增强以及扩展
   1. 全局变量检测增强
      1. int a ;
      2. int a = 10; C下可以，C++重定义
   2. 函数检测增强
      1. 函数的返回值
      2. 形参类型
      3. 函数调用参数个数
   3. 类型转换检测增强
      1. char \* p = (char \*)malloc(64) C++下必须等号左右一致类型
   4. struct 增强
      1. C++可以在结构体中放函数
      2. 创建结构体变量 可以简化关键字struct
   5. bool数据类型扩展
      1. C++才有bool类型
      2. 代表真 --- 1 true 假 ---- 0 false
      3. sizeof = 1
   6. 三目运算符增强
      1. C语言下返回的是值
      2. C++语言下返回的是变量
   7. const增强
      1. C语言下
         1. 全局const 直接修改 失败 间接修改 语法通过，运行失败
         2. 局部 const 直接修改 失败 间接修改 成功
      2. C++语言下
         1. 全局 const 和C结论一样
         2. 局部 const 直接修改失败 间接修改 失败
         3. C++const可以称为常量
7. const 链接属性
   1. C语言下const修饰的全局变量默认是外部链接属性
   2. C++下const修饰的全局变量默认是内部链接属性，可以加extern 提高作用域
8. const分配内存情况
   1. 对const变量 取地址 ，会分配临时内存
   2. 使用普通变量 初始化 const变量
   3. 对于自定义数据类型
9. 尽量用const代替define
   1. define出的宏常量，没有数据类型、不重视作用域
10. 引用
    1. 目的：起别名
    2. 语法： 类型（与原名类型必须一致） &别名 = 原名
    3. 引用必须要初始化
    4. 引用一旦初始化后，就不可以引向其他变量
    5. 建立对数组引用
       1. 直接建立引用
       2. int arr[10];
       3. int(&pArr)[10] = arr;
    6. 先定义出数组类型，再通过类型 定义引用
       1. typedef int(ARRAY\_TYPE)[10];
       2. ARRAY\_TYPE & pArr2 = arr;
11. 参数的传递方式
    1. 值传递
    2. 地址传递
    3. 引用传递
12. 注意事项
    1. 引用必须引一块合法内存空间
    2. 不要返回局部变量的引用
    3. 当函数返回值是引用时候，那么函数的调用可以作为左值进行运算
13. 指针的引用
    1. 利用引用可以简化指针
    2. 可以直接用同级指针的 引用 给同级指针分配空间
14. 常量的引用
    1. const int &ref = 10;
    2. // 加了const之后， 相当于写成 int temp = 10; const int &ref = temp;
    3. 常量引用的使用场景 修饰函数中的形参，防止误操作