C++ 11 语言特性举例分析

# 保持稳定性和兼容性

## 保持与C99兼容

1. C99预定义宏
2. \_\_func\_\_预定义标识符
3. pragma操作符 #pragma once 防止头文件被重复包含 \_Pragma使用
4. 不定参数宏定义以及\_\_VA\_ARGS\_\_ 用于打印日志 #define PR(...) printf(\_\_VA\_ARGS\_\_)
5. 宽窄字符串连接 支持

## long long整型

支持

## \_\_cplusplus

C++ 11 定义\_\_cplusplus为201103LL 小于此值说明不支持c++11

## 静态断言

static\_assert用于编译时断言 用法static\_assert(表达式， “出错语句”);

assert用于运行时断言

## noexceptt修饰符与noexcept操作符

利用noexcept代替throw()表示不抛出异常

## 快速初始化成员变量

## sizeof可以对类中成员进行大小计算

## 扩展的firend语法

这部分用处不大吧， 一般用法满足需求

## final/override控制

final 用于阻止重写

override用于虚函数重写

## 模板函数的默认模板函数

不必传类型，根据参数来判断typename的类型

## 外部模板

# 通用为本， 专用为末

## 继承构造函数:

使用using

## 委托构造函数

在构造函数初始化列表中调用默认构造函数

## 右值引用：移支语义和完美转发

移动语义，实现类似拷贝构造函数，把引用&换成&&， nullptr是空指针的问题

std::move强制一个左值变成右值

太难XX, 后续补上

## 初始化：

vector 可以用{}来初始化了, map也可以了

## POD类型

pod可以用来memcpy, 没有构造函数的类为pod类型

# 新手易学， 老兵易用

1. 右尖括号的改进
2. auto类型推导, 写模板时更加简单， 代码更少
3. decltype 用于定义类型
4. 追踪返回类型 无用
5. 基于范围的for循环

# 提高类型安全

## 强类型枚举

声明: enum class EnumType {};， 强作用域， 不可以与整型隐式转换，可以指定底层类型

## 堆内存管理： 智能指针与垃圾回收

野指针，重复释放，内存泄露

C++98中auto\_ptr废弃，不能调用delete[], C++11 unique\_ptr, shared\_ptr, weak\_ptr智能指针来解决

1. unique\_ptr:

指定指针唯一， 不能再赋值到其它指针， 可以用move来复制，同时原来指针失效， reset失效

1. weak\_ptr:引用计数型指针，可以指向shared\_ptr， 不能拥有该内存，lock可以返回一个shared\_ptr指针，
2. shared\_ptr: 引用计数型指针，可以重复制定

垃圾回收的几种方法：基于引用计数， 跟踪处理

最小垃圾回收支持，不支持吧，这个还需要更加努力

# 提高性能及操作硬件的能力

## 常量表达式函数的使用

无用

## 变长模板

不会用到

## 原子类型与原子操作

互斥锁，粗粒度， 现有原始类型及模板

## 线程局部存储

\_\_pthread errCode 使用的时候还是会用pthread

# 为改变思考方式而改变

## 指针空值nullptr

## 默认函数的控制

=default 来指定默认 =deleted

## lambda函数

定义一些匿名函数, 就地可见性， 简单，比传统意义上的要好

auto add= [=]()->int {return a + b };

# 融入实际应用

## 对齐支持

offsetof 看某个字段在结构体中的偏移

alignof alignas关键字使用

## 通用属性

## unicode支持

## 原生字符串支持，加入R