深蓝学院
shenlanxueyuan.com

C++ 基础

第 8 章：动态内存管理

主讲人 李伟

微软高级工程师

《C++ 模板元编程实战》作者

目录

动态内存基础

- 栈内存 V.S. 堆内存
  - 栈内存的特点：更好的局部性，对象自动销毁
  - 堆内存的特点：运行期动态扩展，需要显式释放
- 在 C++ 中通常使用 new 与 delete 来构造、销毁对象
- 对象的构造分成两步：分配内存与在所分配的内存上构造对象；对象的销毁与之类似
- new 的几种常见形式
  - 构造单一对象 / 对象数组
  - nothrow new
  - placement new
  - new auto
- new 与对象对齐

动态内存基础（续）

- delete 的常见用法
  - 销毁单一对象 / 对象数组
  - placement delete
- 使用 new 与 delete 的注意事项
  - 根据分配的是单一对象还是数组，采用相应的方式销毁
  - delete nullptr
  - 不能 delete 一个非 new 返回的内存
  - 同一块内存不能 delete 多次
- 调整系统自身的 new / delete 行为
  - 不要轻易使用

智能指针

- 使用 new 与 delete 的问题：内存所有权不清晰，容易产生不销毁，多销毁的情况
- C++ 的解决方案：智能指针
  - auto_ptr （ C++17 删除）
  - shared_ptr / uniuqe_ptr / weak_ptr
- shared_ptr—— 基于引用计数的共享内存解决方案
  - 基本用法
  - reset / get 方法
  - 指定内存回收逻辑
  - std::make_shared
  - 支持数组（ C++17 支持 shared_ptr<T[]>； C++20 支持 make_shared 分配数组）
  - 注意： shared_ptr 管理的对象不要调用 delete 销毁

智能指针（续）

- unique_ptr—— 独占内存的解决方案
  - 基本用法
  - unique_ptr 不支持复制，但可以移动
  - 为 unique_ptr 指定内存回收逻辑
- weak_ptr—— 防止循环引用而引入的智能指针
  - 基于 shared_ptr 构造
  - lock 方法

动态内存的相关问题

- sizeof 不会返回动态分配的内存大小
- 使用分配器（ allocator ）来分配内存
- 使用 malloc / free 来管理内存
- 使用 aligned_alloc 来分配对齐内存
- 动态内存与异常安全
- C++ 对于垃圾回收的支持

深蓝学院
shenlanxueyuan.com

感谢聆听！

Thanks for Listening