# 内存管理(Memory Pool)

# 一，介绍

使用内存池进行管理，内存池采用双向链表结构。由内存池向系统申请一块连续内存，使用者仅需向内存池进行申请内存。内存池进行内存分配和回收，并且会对内存碎片进行合并。有效的减少了频繁向系统申请内存的时间消耗，已经产生过多的内存碎片。

# 功能：

1.可以按照任意顺序进行释放已经申请的内存，无需按照申请顺序释放

2.每次释放可以打印信息查看当前的内存是增长还是减少

3.系统结束前可以打印信息查看内存池是否恢复初始状态

4. 每次释放内存，内存池自动合并可以合并的内存碎片

5.内存池是线程安全的

# 二，使用方法

## 1.初始化内存池：

#include “memory.h”  
  
/\* 申请1M大小的内存池 \*/  
size\_t pool\_size = 1024\*1024;  
  
/\* 方法1 \*/  
CMemoryPool::GetInstance().InitPool(pool\_size);  
  
/\* 方法2 \*/  
auto &mem\_pool = CMemoryPool::GetInstance();  
mem\_pool.InitPool(pool\_size);

## 2.申请内存：

/\* 初始化申请对象类型指针 \*/  
int \*p\_int = nullptr;  
/\* 申请对象内存为1K大小 \*/  
size\_t alloc\_size = 1024;  
  
/\* 方法1 \*/  
p\_int = CMemoryPool::GetInstance().Pnew(p\_int, alloc\_size);  
  
/\* 方法2 \*/  
p\_int = mem\_pool.Pnew(p\_int, alloc\_size);

## 3.释放内存：

/\* 方法1 \*/  
CMemoryPool::GetInstance().Pdelete(p\_int);  
  
/\* 方法2 \*/  
mem\_pool.Pdelete(p\_int);

## 4.查询内存池当前信息:

/\* 方法1 \*/  
CMemoryPool::GetInstance().GetMemoryInfo();  
  
/\* 方法2 \*/  
mem\_pool.GetMemoryInfo();

## 5.释放内存池内存：

CMemoryPool::GetInstance().FreePool();

# 三，内存池信息说明

Pool Size: 内存池大小

Max Used Size: 内存池最大使用大小，可能会大于Pool Size

Current Used Size: 当前内存池占用代销

Memory Increasing: 内存使用是否是增长状态

All Memory Are Available: 内存池中所有内存是否都空闲

All Used Memory Free: 内存池中占用的内存是否全部释放完

