8.9文字作业

一、堆区和栈区的区别

1. 分配方式

栈区(Stack)：由编译器自动分配和释放，存放函数的参数值、局部变量等。

堆区(Heap)：由程序员手动分配和释放，若程序员不释放，可能由OS回收。

2. 内存管理方式

栈区：系统自动管理，遵循后进先出(LIFO)原则。

堆区：程序员手动管理，使用类似链表的结构存储，需要用户调用相应函数(如malloc/free, new/delete)来管理。

3. 空间大小

栈区：大小固定，通常为几MB。

堆区：大小可动态调整，通常为数GB，受限于系统可用内存。

4. 碎片问题

栈区：不存在内存碎片问题。

堆区：频繁申请和释放内存可能导致内存碎片问题。

5. 存取效率

栈区：效率高，系统分配整块连续内存，调用开销小。

堆区：效率相对较低，系统需要维护空闲内存列表，有额外的管理开销。

6. 存储内容

栈区：存储局部变量、函数参数、返回地址等。

堆区：存储动态分配的对象、较大的数据结构等。

7. 生命周期

栈区：变量在超出其作用域时自动释放。

堆区：需要手动释放，否则会导致内存泄漏。

8. 安全性

栈区：较安全，但可能发生栈溢出。

堆区：较不安全，可能出现内存泄漏、悬挂指针等问题。

二、使用枚举而非#define的原因

1. 类型安全

枚举提供类型检查，而#define只是简单的文本替换，不提供类型安全。

2. 调试友好

枚举常量在调试时可以显示其符号名称，而#define常量在预处理后就消失了。

3. 命名空间管理

枚举可以在类作用域或命名空间内定义，有效避免命名冲突。#define是全局的，容易造成命名污染。

4. 结构化分组

枚举可以将相关的常量组织在一起，形成一个逻辑单元，提高代码可读性。

5. 编译器优化

编译器可以对枚举进行更好的优化，而#define是预处理指令，不参与编译优化。

6. 代码维护

枚举更有利于代码维护，可以方便地添加、删除或修改常量，而不会影响其他代码。

7. 强类型枚举(C++11)

C++11引入的强类型枚举(enum class)提供了更强的类型安全和作用域控制，进一步增强了枚举相对于#define的优势。