简述堆区和栈区的区别

堆区：由程序员手动放入，需要专门找到相应的区域储存，空间大，容易产生碎片问题，需要动态共享或全局共享时使用，会被项目程序启动时或垃圾回收器调用，由new/malloc产生，在delete/free结束

栈区：由编译器自动生成，与作用域绑定，离开及被销毁，空间小，不会产生碎片化问题，在调用指针时使用。

简述出现野指针的可能原因和相应的避免方法。

1.未初始化即使用定义指针但未赋初值，其值随机

避免：定义指针时立即置 nullptr，使用前确保已指向合法对象。

对象已释放仍继续访问

2. free/delete 后仍保留原指针。

避免：释放后立即将指针置 nullptr，或改用智能指针（unique\_ptr、shared\_ptr）。

返回局部变量地址

3.函数返回指向栈上局部变量的指针，函数结束后栈帧被回收。

避免：不返回局部变量地址；改为返回堆上对象、静态对象或按值返回。

重复释放同一块内存

4对同一块内存多次 free/delete，第二次及以后形成野指针。

避免：释放后置 nullptr，或利用智能指针自动管理。

5.数组/对象指针运算后指向非法地址。

避免：严格检查边界，使用 std::array、std::vector 等容器替代裸指针。

6.强制类型转换后指针与指向对象类型不匹配。

避免：使用 static\_cast、reinterpret\_cast 前确保类型安全，或避免不必要的强转

分析该程序运行后的输出结果。

arr 的类型是 int[5]，在大多数表达式中会退化为 int\* 指向首元素。&arr 的类型是 int (\*)[5]，即“指向含 5 个 int 的数组的指针”。&arr + 1 的步长是整个数组的大小（5 \* sizeof(int) = 20 字节）。因此 ptr1 指向 arr 末尾之后 20 字节的位置。ptr1[-1] 等价于 \*(ptr1 - 1)，即回到 arr 最后一个元素 5（0x00000005）。arr 先被强制转换为 int（整数地址值），然后加 1 字节。ptr2 指向 arr+1，解引用 \*ptr2 会尝试读取从 arr+1 开始的 4 字节：小端序下，读取的字节是 00 00 00 02，即 0x02000000。最后在小端系统中输出52000000.

我们可以用#define定义常量，为什么要用枚举？

#define 只是宏替换，enum 提供了类型、作用域、调试、自动递增等额外好处，代码更安全、可读、易维护。