c++核心编程：

1. 内存分区模型（4 大区）

- 代码区：存放函数体的二进制代码，由操作系统进行管理
- 全局区：存放全局变量以及静态变量以及常量（字符串常量全局常量）
- 栈区(stack)：由编译器自动分配释放，存放函数的参数值，局部变量以及局部常量等。
- 堆区(heap)：由程序员分配和释放，若不释放，将在程序结束时由系统回收。

在程序运行前：程序编译后，会生成 exe 可执行程序。在没有执行该程序之前，分成两区域。

代码区：

1. 存放 CPU 执行的机器指令
2. 代码区是共享的，共享的目的是对于被频繁执行的程序吗，如函数之类的，只需在内存中有一份代码，即只占领一个内存空间，并不会重复占据内存空间。
3. 代码区是只读的，只读的目的是为了防止程序意外修改其指令。

全局区：

1. 全局变量和静态变量存放在此。
2. 全局区包含常量区，字符串常量和 const 修饰的全局常量也存放在此。
3. 该区域的数据在程序结束后由操作系统释放。
4. 静态变量:static，常量：const

在程序运行后：

栈区：

- 由编译器自动分配释放，存放函数的参数值，局部变量等要注意，不要返回局部变量的地址，即不要用指针的形式去接收局部变量的地址，在之后的解引用过程中会出错。栈区数据由编译器自动释放，在程序执行后那块内存会自动释放。因为系统设定，第一次打印时不会出错，第二次打印就是非法操作，会有问题。同时，函数中的形参数据也会放在栈区。

堆区：

- 由程序员分配释放，若程序员不释放，程序结束后由系统释放
- 在 c++中，主要利用 new 在堆区开辟内存

new 操作符：

用于开辟堆区数据，由程序员手动开辟。手动释放。手动释放：delete

语法：new 数据类型：利用 new 创建的数据，会返回该数据对应的类型的指针

```
int *a =new int (10);
delete a;
```
也可以用于开辟数组：`int *arr = new int[10]; delete[] arr;`
指针的本质也是局部变量，放在栈上，但是指针保存的数据是在堆区，指针表示的地址也是在堆区。因此在执行后并不会被释放。只有在整个程序结束后才会被释放。

引用
作用：给变量起别名，本质其实是指的同一块内存地址。引用本质是指针常量。在你使用的时候，系统其实会进行一些转换，原理就是指针常量。
语法：数据类型 &别名 = 原名

注意事项：
- 引用必须进行初始化
- 引用在初始化后，不可以改变，但是能够赋值。

引用做函数参数：
作用：函数传参时，可以利用引用技术让形参修饰实参，效果与地址传递类似
优点：能够简化指针修饰实参

引用做函数返回值：
- 作用：引用是可以作用函数的返回值存在
- 不要返回局部变量引用，可以返回比如静态变量引用
- 函数调用可以作为左值

常量引用：
常量引用主要用来修饰形参，防止误操作