C++内存管理

C++的内存的划分

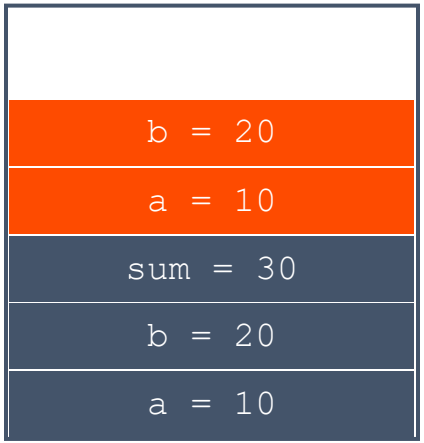
- 栈区：由编译器自动分配与释放，存放为运行时函数分配的局部变量、函数参数、返回数据、返回地址等。其操作类似于数据结构中的栈。
- 堆区：一般由程序员自动分配，如果程序员没有释放，程序结束时可能有OS回收。
- 全局区（静态区）：存放全局变量、静态数据。程序结束后由系统释放。
- 常量区（文字常量区）：存放常量值，如字符串常量，不允许修改。程序结束后有系统释放。
- 代码区：存放函数体（类成员函数和全局区）的二进制代码。

C++管理数据内存的方式

- C++ 根据用于分配内存的方法，有 3 种管理数据内存的方式：自动存储、静态存储和动态存储。这 3 种方式分配的数据对象在存在时间的长短方面各不相同。
- 1.自动存储：在函数内部定义的常规变量使用自动存储空间，被称为自动变量，它们在所属的函数被调用时自动产生，在该函数结束时消亡。自动变量通常存储在栈内存中。

```
int getSum(int a, int b) {
    return a + b;
}

int main() {
    int a = 10;
    int b = 20;
    int sum = getSum(a, b);
}
```



C++管理数据内存的方式

- 2.静态存储：静态存储是整个程序执行期间都存在的存储方式。使变量成为静态的方式有两种：一种是在函数外面定义它；另一种是在声明变量时使用关键字static：

```
static int num = 10;
```

- 3.动态存储：new 和 delete 运算符提供了一种比自动变量和静态变量更灵活的方法。它们管理了一个内存池，这在 C++ 中被称为自由存储空间或堆。该内存池同用于静态变量和自动变量的内存是分开的。在栈中，自动添加和删除机制使得占用的内存总是连续的，但 new 和 delete 的相互影响可能导致占用的自由存储区不连续，这使得跟踪新分配内存的位置更困难。

内存泄漏

- 如果使用 new 运算符在自由存储空间（或堆）上创建变量后，没有调用 delete ，将发生什么情况呢?
- 如果没有调用 delete，即使指针变量的内存被释放，但在自由存储空间上动态分配的内存也将继续存在。因为指向这些内存的指针无效，这样将无法访问自由存储空间中的这些内存。从而导致内存泄漏。
- 被泄漏的内存在程序整个生命周期内都不可使用；这些内存被分配出去但无法收回。内存泄漏可能会非常严重，以致于应用程序可用的内存被耗尽，导致程序崩溃。
- 要避免内存泄漏，最好是养成这样一种习惯，即同时使用 new 和 delete 运算符，在自由存储空间上动态分配内存，随后便释放它。

Thanks