内存四区

2023年9月9日

17:40

内存四区：

1）代码区：源文件二进制代码，由操作系统管理

a）存放CPU执行的机器指令

b）代码区是共享和只读的

2）全局区：存放全局变量和静态变量以及常量（字符串常量和其他常量）

全局区里的数据在程序结束后由操作系统释放

注意：

C++程序运行前分为全局区和代码区

代码区的特点是共享和可读

全局区中存放全局变量、静态变量和常量

常量区中存放const修饰的全局常量和字符串常量

3）栈区：由编译器自动分配，存放函数的参数值、局部变量等

注意：不要返回局部变量的地址，栈区开辟的地址生命周期结束后由编译器自动释放

4）堆区：由程序员分配和释放，若不释放，程序结束操作系统自行回收

四区的意义：不同区域的数据，赋予不同生命周期，给我们更大的灵活编程

new操作符

new+数据类型+（数据大小）表示在堆区开辟一个整型变量

释放时只需要释放指向该地址的指针即可，delete p；

new+数据类型+[数据个数]，表示在堆区开辟一个数组

释放时，需要加[],表示释放数组，delete[] p;

引用

数据类型 &别名=原名；

引用注意事项：

* 引用必须初始化
* 引用初始化之后，就不可以改变

引用做函数参数

作用：函数传参时，可以利用技术让形参修饰实参

引用做函数的返回值

（不返回局部函数的引用）

函数的调用可以做左值，左值返回即是变量的别名，对其便可进行赋值

引用的实质：就是一个指针常量

int\* const ref=&a;

常量引用，用来修饰形参，防止误操作

int a=10;

int &ref=10;//是不被允许的，引用必须引用一块合法的内存空间

const int &ref=10;//加上const便可以了，编译器会将代码改成int temp; const int &ref=temp;

函数默认参数  
1.如果一个位置已经有了默认参数值，那么从这个位置往后，从左向右都必须有默认参数值

2.如果函数声明有默认参数，函数实现就不能有默认参数，声明和实现只能有一个默认参数

函数的占位参数

例如：

void func(int a,int)  
{

//语句

}

第二个int，就起占位参数的作用

占位参数还可以有默认参数

例如void fuc(int a,int =10);

函数重载

条件：

* 在同一作用域下
* 函数名称相同
* 函数参数类型不同或者个数不同或者顺序不同

注意：函数的返回值不可以作为函数重载的条件

函数重载注意事项

* 引用作为重载条件
* 函数重载碰到函数默认参数

例如：

void func(int &a);

void func(const int &a);

int a=10;

语句func(a)调用第一句

语句func(10)调用第二句

例如：

void func(int a,int b=10);

void func(int a);

语句func（10）调用时，会出现二义性，两种函数都可以调用