

26 Destructors in C++

1. 析构函数

*Destructor*是在你销毁一个对象时运行，而*constructor*构造函数是在你创建一个对象实例时运行

析构函数同时适用于栈和堆分配的内存，因此如果你用*new*关键字创建一个对象（存在于堆上），然后你调用*delete*，析构函数就会被调用。

如果只有基于栈的对象，当跳出作用域的时候这个对象会被删除，所以这个时候析构函数也会被调用

析构函数：~类名


```
~Entity()
{
    std::cout << "Destroyed Entity!" << std::endl;
}

void Print()
{
    std::cout << X << "," << Y << std::endl;
}

};

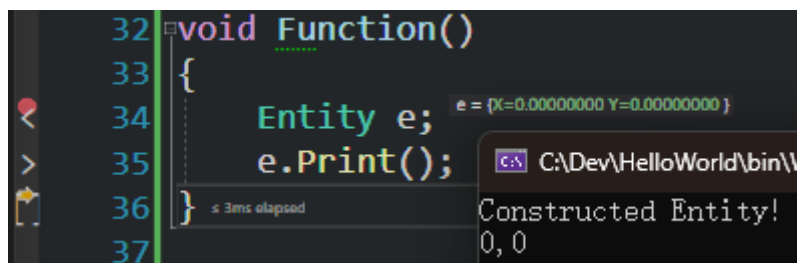
void Function()
{
    Entity e;
    e.Print();
}

int main()
{
    Function();
    std::cin.get();
}
```

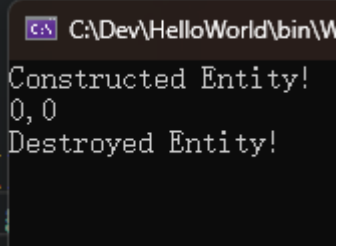


栈调用（离开作用域后销毁）：

```
32 void Function()
33 {
34     Entity e;
35     e.Print();
36 }
37
```



```
32 void Function()
33 {
34     Entity e;
35     e.Print();
36 }
37
38
39 int main()
40 {
41     Function()
42     std::cin.
```



2. 实际使用

为什么要用`destructor`? 若果你在构造函数中做了一些初始化工作, 你可能会想要在析构函数里进行释放或者销毁工作。如果不这么做的话, 可能会造成内存泄漏。

一个很好的例子就是`heap`(堆)分配对象, 如果你手动在堆上分配了任何类型的内存空间, 那么你也要手动地进行清除。如果使用了`Entity`或者在`Entity`的构造函数中进行分配, 那么你就要在析构函数中清除他们。因为析构函数调用后, 那个`Entity`对象就不存在了。

一般不手动调用`destructor`

```
Entity e;
e.~Entity(); // Destroyed Entity
```