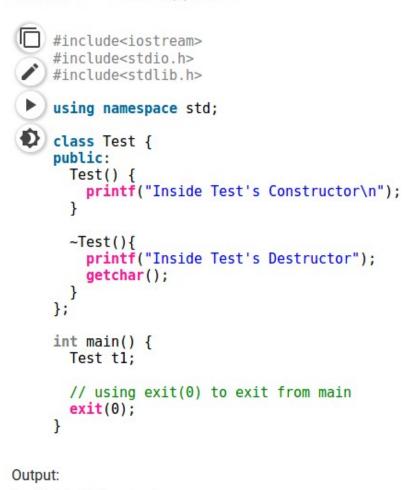
return statement vs exit() in main()

In C++, what is the difference between exit(0) and return 0?

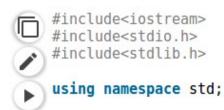
When *exit(0)* is used to exit from program, destructors for locally scoped non-static objects are not called. But destructors are called if return 0 is used.

Program 1 - - uses exit(0) to exit



Program 2 – uses return 0 to exit

Inside Test's Constructor



```
class Test {
public:
    Test() {
        printf("Inside Test's Constructor\n");
    }
    ~Test(){
        printf("Inside Test's Destructor");
    }
};

int main() {
    Test t1;

    // using return 0 to exit from main
    return 0;
}
```

Output:

Inside Test's Constructor

Inside Test's Destructor

Calling destructors is sometimes important, for example, if destructor has code to release resources like closing files.

Note that static objects will be cleaned up even if we call exit(). For example, see following program.

```
#include<iostream>
   #include<stdio.h>
   #include<stdlib.h>
   using namespace std;
class Test {
   public:
     Test() {
       printf("Inside Test's Constructor\n");
     ~Test(){
       printf("Inside Test's Destructor");
       getchar();
   };
   int main() {
     static Test t1; // Note that t1 is static
     exit(0);
   }
```

Output:

Inside Test's Constructor

Inside Test's Destructor