- The **return values** of functions are passed to the calling function via a **temporary object.** When the flow of the program sees the return statement, the compiler first creates a temporary object of the return value type. Then it assigns the return statement to this temporary object. When we use the return value, we actually use that temporary object. This temporary object is automatically destroyed by the compiler when the function call ends.

For example:

```
int foo(void)
{
/* ... */
return expression ;
}
...
x = foo() * 2;
```

- Here, in fact, the following processes are taking place in the background:

```
int temp =expression ; // when the flow comes to the return statement
...
x = temp * 2;  // temp is being destroyed
```

So the type of the return value of the function actually specifies the type of the temporary object that will be created by the return operation. The return operation is like an assignment operation to this temporary object. So the return operation is actually a hidden assignment operation. Compilers generally create temporary objects in CPU registers during the return operation whenever possible. Note that in void functions such a temporary object is never created.