

# Lesson 13. Variable Modifiers (变量修饰符)

## — Auto & Extern

变量修饰符是指用于修改变量的行为或属性的附加关键字或标记

### Auto Modifier

#### Auto means Automatic

Variables declared inside a scope by default are automatic variables

(默认情况下，在作用域内声明的变量是自动变量)

自动变量的生命周期与函数的执行周期相匹配，也就是说，它们在函数被调用时创建，在函数执行结束后销毁，该变量不会浪费任何内存

Syntax: `auto int some_variable_name;`

自C99标准以后，C语言引入了 `auto` 关键字的新用法，用于显式声明存储类别为自动的局部变量。然而，这种用法几乎不再使用，因为局部变量的默认存储类别就是自动的，所以通常不需要显式地使用 `auto`

#### Take Aways (要点)

- If you won't initialize auto variable, by default it will be initialized with some garbage (random) value

```
1 | #include <stdio.h>
2 |
3 | int main()
4 | {
5 |     auto int var;
6 |     printf("%d", var);
7 |     return 0;
8 | }
9 |
```



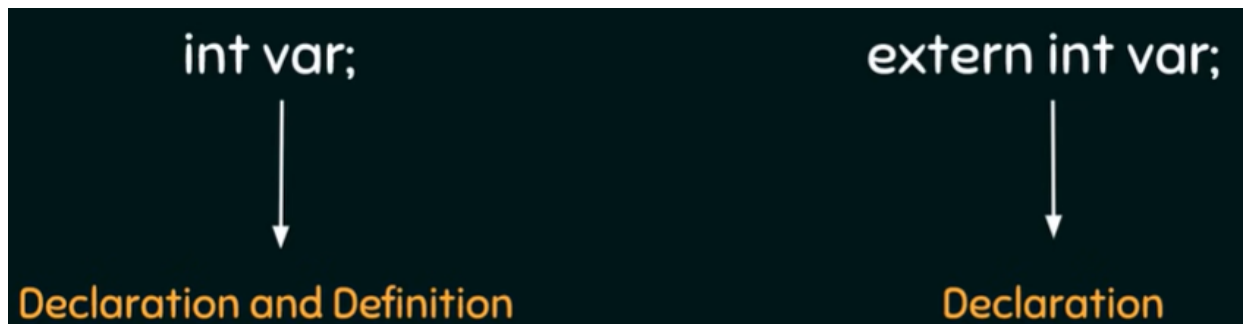
- On the other hand, global variable by default(默认) initialized to 0

```
1 | #include <stdio.h>
2 |
3 | int var;
4 |
5 | int main()
6 | {
7 |     printf("%d", var);
8 |     return 0;
9 | }
10 |
```

0  
Process returned 0 (0x0) execution time : 0.394 s  
Press any key to continue.

## Extern Modifier

Extern is short name for external



Used when a particular file needs to access a variable from another file

`extern` 关键字用于声明一个全局变量或函数，以指示它们实际上是在其他源文件中定义的，而不是在当前源文件中定义的。这允许在不同的源文件之间共享变量或函数，以便它们在整个程序中可用

```
main.c x other.c x
1 | #include <stdio.h>
2 |
3 | extern int a;
4 |
5 | int main()
6 | {
7 |     printf("%d", a);
8 |     return 0;
9 | }
10 |
```

```
main.c x other.c x
1 | int a = 5;
2 |
```

```
"E:\C Programming & Data St  X  +  v
5
Process returned 0 (0x0)    execution time : 0.579 s
Press any key to continue.
```

```
1  #include <stdio.h>
2
3  int a = 9;
4
5  int main()
6  {
7      extern int a;
8      printf("%d", a);
9      return 0;
10 }
11
```

```
"E:\C Programming & Data St  X  +  v
9
Process returned 0 (0x0)    execution time : 0.351 s
Press any key to continue.
```

## Take Aways

1. When we write `extern some_data_type some_variable_name;` no memory is allocated. Only property of variable is announced.
2. Multiple declarations of extern variable is allowed within the file. This is not the case with automatic variables.
3. Extern variable says to compiler "go outside from my scope and you will find the definition of the variable that I declared".
4. Compiler believes that whatever the extern variable said is true and produce no error. Linker throws an error when he finds no such variable exist.
5. When an extern variable is `initialized`, then memory for this variable is allocated and it will be considered `defined`.

- 1.当我们写`extern some_data_type some_variable_name;`没有分配内存。仅声明变量的属性。
- 2.文件内允许多次声明extern变量。自动变量的情况并非如此。
- 3.Extern 变量对编译器说：“从我的作用域之外出去，你将找到我声明的变量的定义”。
- 4.编译器认为外部变量所说的都是正确的，不会产生错误。当链接器发现不存在这样的变量时，会抛出错误。
- 5.当一个外部变量被初始化时，该变量的内存被分配，并且它将被视为已定义。

