概念：

C语言中所有的标识符（变量/名字）都有可见范围。为了搞清楚这些标识符的可见范围我们来研究一下，而可见分为也被称为作用域。

在软件开发的过程中应该尽可能缩小标识符的可见范围，可以尽可能的降低同名冲突的问题。

函数声明的作用域：

概念： 一般写在头文件内部， 或者源文件的头部。用来告诉编译器函数的模型。

```
1
2 main.c
3
4 // 函数声明
5 bool swap( int *  , int *  );
6 int max( int a , int b , int c );
```

作用域：

只是当前文件可见。虽然写在函数体外但是并不是全局的。

函数头中的作用域：

```
1 int max( int a , int b , int c ) // 函数头
2 {
3  // 函数体
4 }
```

作用域：

max 默认是全局可见的

例外情况 被static 修饰则本文件可见

变量 a, b, c 是属于该函数的局部变量， 因此作用域在函数体内部

局部作用域：

概念： 在代码块内u不定义的变量 ， 可见范围从定义语句开始直到该代码块的右括号右边结束 }

示例：

```
1 {
```

```
    int b = 250 ;

        {
            int c = 399 ;// c 作用域开始
            int d = 249;

        }// c 作用域的结束
        printf("c:%d\n" , c) ; // 已经离开了c的作用域因此不能使用
}

例如实际中：
 for (size_t i = 0; i < argc ; i++)
        {
            printf("argv[%ld]:%s\n" , i , argv[i]);

        }

        printf("%d" , i ); // i未定义， 已经离开的I的可见范围

```

注意：

- 代码块指的是一对 { } 所括起来的区域
- 代码块可互相嵌套，外层的标识符，可以被内层识别， 反之则不行
- 在内码块内部定义的标识符， 在外面其它的代码块中是不可见，因此称为局部作用域

全局作用域：

概念： 在代码块的外部定义，他的可见范围是可以跨文件可见的。

```
int global ; // 变量 global 就是跨文件可见的全局变量

int main(int argc, char const *argv[])
{
    int a = 1 ;
    int b = 250 ;
}
```

注意：

在函数体外部定义

不可以被static修饰， 修饰之后就变成本文件可见（可见范围被缩小）。

作用域的临时掩盖：

如果有多个不同的作用域互相嵌套，小的作用域的作用范围会临时掩盖大的作用域（标识符名字相同）。临时失去大作用域的值。

```
int a = 1 ;

{
    printf("1a:%d\n" , a);// 输出 为 1  ，  使用的是外面大的作用域

    int a = 250 ; // 从这里开始 a 的值被临时覆盖为 250 外面的1临时失效
    printf("2a:%d\n" , a);// 输出 为 250  ，  使用的是内部的小的作用域
}

printf("3a:%d\n" , a); // 离开小的作用域后 ，a 恢复为  1
```

static 关键字：

在C语言中非常重要的一个角色， 它在不同的场合表现的意义不一样。

1. 把可见范围进行缩小为本文将可见：
    - 修饰全局变量
    - 修饰普通的函数
2. 把变量的存储区修改为静态数据（数据段）：
    - 修饰局部变量, 使得局部变量的存储区从栈改为数据段