西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

C程序设计

Programming in C

1011014

主讲：姜学锋，计算机学院

设计函数 - 函数间的数据传递（1）

1、局部变量和全局变量

2、对象作用域

4.6 作用域和生命期

- 当函数模块越来越多的时候，就要去关注作用域（scope）和生命期（lifetimes），无论是前面介绍过的变量，还是后面的数组、指针、结构体等对象。

4.6.1 局部变量

- 在函数内部或复合语句中（简称区域）定义的变量，称为局部变量（local variable），又称为内部变量。

4.6.1 局部变量

```
1   int f1(int x,int y) // f1函数
2   {
3       int a,b,m=100;
4       if (x>y) {
5           int a,t;
                ⋮
20      }
        ⋮
35  }
36  int main() // main函数
37  {
38      int a=15,b=10,c,n=200;
39      c = f1(a,b);
            ⋮
60  }
```

a,t有效

a,b,m有效

x,y有效

a,b,c有效

4.6.1 局部变量

- 局部变量的说明。
- （1）局部变量只能在定义它的区域及其子区域中使用。例如main函数可以使用第38行的a、b、c、n，f1函数可以使用第1行的x、y（形参）和第3行的a、b、m，if分支的复合语句可以使用f1函数定义的变量和第5行的a、t；另一方面，main函数就不能使用f1函数的变量，f1函数不能使用main函数的变量。

4.6.1 局部变量

- （2）在同一个区域中不能定义相同名字的变量。例如第1行有了x、y，那么在f1函数内部就不能再使用x和y的名字了。

4.6.1 局部变量

- （3）在不同区域中允许定义相同名字的变量，但本质上它们是不同的变量，例如main函数的a、b和f1函数的a、b完全不相干。

4.6.1 局部变量

- （4）如果一个变量所处区域的子区域中有同名的变量，则该变量在子区域无效，有效的是子区域的变量，称为定义屏蔽。例如第3行的a在f1函数中，而if分支的复合语句是f1函数的子区域，并且第5行也有a定义，所以第3行的a在复合语句不可见，复合语句的a是第5行定义的。

4.6.2 全局变量

- 在源文件中，但在函数外部定义的变量，称为全局变量（global variable），全局变量的有效区域是从定义变量的位置开始到源文件结束。

4.6.2 全局变量

```
1   int m=10,n=5;
2   int f1(int x,int y) // f1函数
3   {
4       int m=100;
           ⋮
10      x = m + n;
           ⋮
20  }
21  int a=8,b=4;
22  int main() // main函数
23  {
24      int a=15,n=200,x;
25      x = a + b + m + n;
           ⋮
30  }
        ⋮
        ⋮
```

x,y,m 有效

a,n,x 有效

a,b有效

m,n有效

文件结束行

4.6.2 全局变量

- 第1行的m、n以及第21行的a、b是全局变量，其余为局部变量（虚线）。可以看到，全局变量的有效区域比局部变量大，可以跨多个函数，因而可以在多个函数中使用。
- 函数之间可以利用全局变量来交换数据，即一个函数修改了全局变量，那么另一个函数使用的是已经修改过的变量。

4.6.2 全局变量

- 程序中的全局变量都处在源文件范围内，所以不能使用相同的名字。
- 全局变量依然有定义屏蔽，例如：第10行的n是第1行定义的，尽管第1行的m有效范围一直往下，但在f1函数这个子区域内定义了同名的m，则全局变量m被屏蔽了，所以第10行的m是第4行定义的。

4.6.2 全局变量

- 函数之间数据传递尽管可以利用全局变量，但这样一来也导致两个函数彼此分不开，违背模块化的原则，所以结构化程序设计提倡少用或不用全局变量。

4.6.3 作用域

- C语言的实体通常有三类：①变量或对象，例如基本类型变量、数组对象、指针对象、结构体对象等；②函数；③类型。包含结构体类型、共用体类型。
- 作用域是程序中的一段区域。在同一个作用域上，C程序中每个名字都与唯一的实体对应；只要在不同的作用域上，那么在程序中就可以多次使用同一个名字，对应不同作用域中的不同实体。

4.6.3 作用域

- 一个C程序可以由任意多的源文件组成，每个源文件可以有任意多的函数，在函数中可以包含任意多的复合语句块，复合语句块中又可以嵌套任意多的复合语句子块；另外，一个程序还可以有任意多的函数原型、结构体类型和共用体类型声明。

4.6.3 作用域

- C语言的作用域有如下几个：
  - （1）文件作用域（file scope）
  - 文件作用域是指一个C程序中所有源文件的区域。具体到一个源文件中，文件作用域是从文件第一行开始，直到文件结束的区域。
  - （2）函数作用域（function scope）
  - 函数作用域是指一个函数从函数头开始直到函数的右大括号（}）结束之间的区域。不同的函数是不同的函数作用域。一个C程序可以有任意多的函数作用域，但所有的函数作用域都是在文件作用域中的。

4.6.3 作用域

- （3）块作用域（block scope）
- 块作用域是由复合语句的一对大括号{ }界定的区域。不同的复合语句是不同的块作用域，复合语句可以嵌套，因而块作用域也可以嵌套；块作用域只能在函数作用域内，不能直接放在文件作用域上；一个函数作用域内可以有任意多的块作用域，或者嵌套，或者平行，不同函数中的块作用域是各自不同的。

- （4）类型声明作用域（declaration scope）
- 类型声明作用域是指在结构体类型、共用体类型声明中由一对大括号{ }界定的区域。

4.6.3 作用域

- 例如：

```
struct 结构体类型名 {
    ... //成员列表   //成员列表在结构体类型声明作用域上
}
```

- 类型声明作用域可以放在文件作用域、函数作用域、块作用域中。

4.6.3 作用域

- （5）函数原型作用域（function prototype scope）
- 函数原型作用域是函数原型中括号内的区域，即形参列表所处的区域，例如：

```
int max(int a,int b,int c); // a,b,c在函数原型作用域上
```

- 函数原型作用域可以放在文件作用域、函数作用域、块作用域、类型声明作用域中。

4.6.3 作用域

- 上述作用域，文件作用域是全局作用域，其余为局部作用域。除函数原型作用域外，局部作用域都是用一对大括号{ }界定的。
- 在C语言中，全局作用域只有一个，而局部作用域可以有多个。


程序设计