函数

2018年10月28日　17:23

一，是什么?

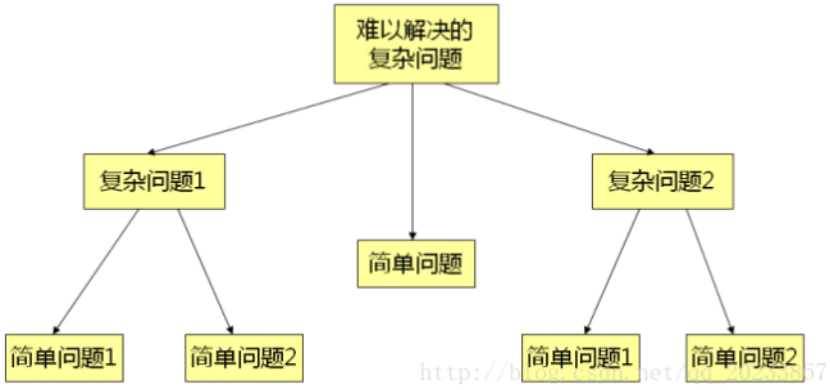
函数就是一系列C语句的集合，为了完成某个会重复使用的特定功能。需要该功能的时候，直接调用该函数即可，不用每次都堆叠一大堆的代码。需要修改该功能的时候，也只要修改和维护这一个函数即可。总之，将语句集合成函数，好处就是方便代码重用。并且，一个好的函数名，可以让人一眼就知道这个函数实现的是什么功能，方便维护。

二，有什么用?

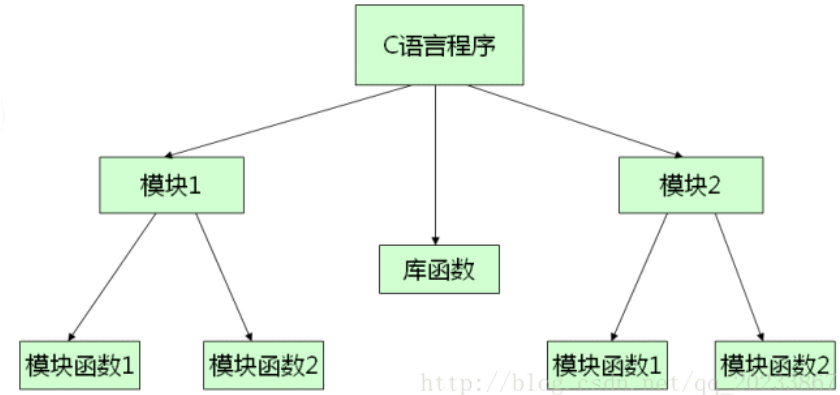
使用函数的好处：

- 使用函数可使程序清晰、精炼、简单、灵活。
- 函数就是功能。每一个函数用来实现一个特定的功能。函数名应反映其代表的功能。
- 在设计较大程序时，往往把它分为若干个程序模块，每一个模块包括一个或多个函数，每个函数实现一个特定的功能。
- 一个C程序可由一个主函数和若干个其他函数构成。由主函数调用其他函数，其他函数也可以互相调用。

函数的意义---模块化程序设计：



C语言中的模块化：



三，怎么用?

1.一些说明：

(1) 一个C程序由一个或多个程序模块组成，每一个程序模块作为一个源程序文件。较大的程序，可分别放在若干个源文件中。这样便于分别编写和编译，提高调试效率。一个源程序文件可以为多个C程序共用。

(2) 一个源程序文件由一个或多个函数以及其他有关内容（如指令、数据声明与定义等）组成。一个源程序文件是一个编译单位，在程序编译时是以源程序文件为单位进行编译的，而不是以函数为单位进行编译的。

(3) C程序的执行是从main函数开始的，如果在main函数中调用其他函数，在调用后流程返回到main函数，在main函数中结束整个程序的运行。

(4) 所有函数都是平行的，即在定义函数时是分别进行的，是互相独立的。一个函数并不从属于另一个函数，即函数不能嵌套定义。函数间可以互相调用，但不能调用main函数。main函数是被操作系统调用的。

(5) 从用户使用的角度看，函数有两种。

① 库函数，它是由系统提供的，用户不必自己定义，可直接使用它们。应该说明，不同的C语言编译系统提供的库函数的数量和功能会有一些不同，当然许多基本的函数是共同的。

② 用户自己定义的函数。它是用以解决用户专门需要的函数。

(6) 从函数的形式看，函数分两类。

① 无参函数。在调用无参函数时，主调函数不向被调用函数传递数据。

② 有参函数。在调用函数时，主调函数在调用被调用函数时，通过参数向被调用函数传递数据。

2.定义函数：

定义函数应包括以下几个内容:

(1) 指定函数的名字，以便以后按名调用。

(2) 指定函数的类型，即函数返回值的类型。

(3) 指定函数的参数的名字和类型，以便在调用函数时向它们传递数据。对无参函数不需要这项。

(4) 指定函数应当完成什么操作，也就是函数是做什么的，即函数的功能。这是最重要的，是在函数体中解决的。

定义无参函数：

```
类型名  函数名()
{
    函数体
}
```

或

```
类型名  函数名(void)
{
    函数体
}
```

函数名后面括号内的void表示“空”，即函数没有参数。

函数体包括**声明部分**和**语句部分**。

在定义函数时要用“类型标识符”(即类型名)指定函数值的类型，即指定函数带回来的值的类型。

定义有参函数：

```
类型名  函数名(形式参数列表)
{
    函数体
```

}

3.调用函数：

1. 函数调用语句

把函数调用单独作为一个语句。如printf_star();

这时不要求函数带值，只要求函数完成一定的操作。

2. 函数表达式

函数调用出现在另一个表达式中，如c=max(a,b);

这时要求函数带一个确定的值以参加表达式的运算。

3. 函数参数

函数调用作为另一个函数调用时的实参。如m=max(a,max(b,c));，又如:printf ("%d", max (a,b));

在调用有参函数时，主调函数和被调用函数之间有数据传递关系。

在定义函数时函数名后面括号中的变量名称为“**形式参数**”（简称“形参”）或“虚拟参数”。

在主调函数中调用一个函数时，函数名后面括号中的参数称为“**实际参数**”（简称“实参”）。 实际参数可以是常量、变量或表达式，但要求它们有确定的值。

实参与形参的类型应相同或赋值兼容。赋值兼容是指实参与形参类型不同时能按不同类型数值的赋值规则进行转换。

4.函数调用的过程：

(1) 在定义函数中指定的形参，在未出现函数调用时，它们并不占内存中的存储单元。在发生函数调用时，函数的形参才被临时分配内存单元。

(2) 将实参的值传递给对应形参。

(3) 在执行函数期间，由于形参已经有值，就可以利用形参进行有关的运算。

(4) 通过return语句将函数值带回到主调函数。应当注意返回值的类型与函数类型一致。如果函数不需要返回值，则不需要return语句。这时函数的类型应定义为void类型。

(5) 调用结束，形参单元被释放。注意: 实参单元仍保留并维持原值，没有改变。如果在执行一个被调用函数时，形参的值发生改变，不会改变主调函数的实参的值。因为实参与形参是两个不同的存储单元。

5.在一个函数中调用另一个函数（即被调用函数）需要具备如下条件:

(1) 首先被调用的函数必须是已经定义的函数（是库函数或用户自己定义的函数）。

(2) 如果使用库函数，应该在本文件开头用#include指令将调用有关库函数时所需用到的信息“包含”到本文件中来。

(3) 如果使用用户自己定义的函数，而该函数的位置在调用它的函数（即主调函数）的后面（在同一个文件中），应该在主调函数中对被调用的函数作声明(declaration)。声明的作用是把函数名、函数参数的个数和参数类型等信息通知编译系统，以便在遇到函数调用时，编译系统能正确识别函数并检查调用是否合法。

4.函数的返回值：

通常，希望通过函数调用使主调函数能得到一个确定的值，这就是**函数值**(函数的返回值)。

(1) **函数的返回值是通过函数中的return语句获得的。**一个函数中可以有一个以上的return语句，执行到哪一个return语句，哪一个return语句就起作用。return语句后面的括号可以不要，如“return z;”与“return(z);”等价。return后面的值可以是一个表达式。

(2) **函数值的类型。**函数值的类型在定义函数时指定。

(3) **在定义函数时指定的函数类型一般应该和return语句中的表达式类型一致。**

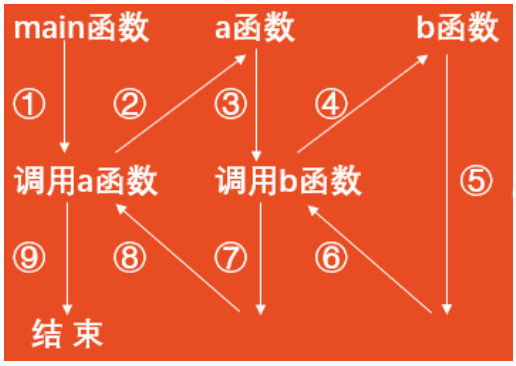
如果函数值的类型和return语句中表达式的值不一致，则以函数类型为准。对数值型数据，可以自动进行类型转换。即**函数类型决定返回值的类型。**

(4) **对于不带回值的函数，应当用定义函数为“void类型”**（或称“空类型”）。这样，系统就保证不使函数带回任何值，即禁止在调用函数中使用被调用函数的返回值。此时在函数体中不得出现return语句。

5.函数的嵌套使用

C语言的函数定义是互相平行、独立的，也就是说，在定义函数时，一个函数内不能再定义另一个函数，即不能嵌套定义，但可以嵌套调用函数，即在调用一个函数的过程中，又调用另一个函数。

① 执行main函数的开头部分；
② 遇函数调用语句，调用函数a，流程转去a函数；
③ 执行a函数的开头部分；
④ 遇函数调用语句，调用函数b，流程转去函数b；
⑤ 执行b函数，如果再无其他嵌套的函数，则完成b函数的全部操作；
⑥ 返回到a函数中调用b函数的位置；
⑦ 继续执行a函数中尚未执行的部分，直到a函数结束；
⑧ 返回main函数中调用a函数的位置；
⑨ 继续执行main函数的剩余部分直到结束。



6.局部变量与全局变量

定义变量可能有3种情况:

(1) 在函数的开头定义；

(2) 在函数内的复合语句内定义；

(3) 在函数的外部定义。

1.局部变量：

在一个函数内部定义的变量只在本函数范围内有效，也就是说只有在本函数内才能引用它们，在此函数以外是不能使用这些变量的。在复合语句内定义的变量只在本复合语句范围内有效，只有在本复合语句内才能引用它们。在该复合语句以外是不能使用这些变量的，以上这些称为“**局部变量**”。

(1) 主函数中定义的变量也只在主函数中有效。主函数也不能使用其他函数中定义的变量。
(2) 不同函数中可以使用同名的变量，它们代表不同的对象，互不干扰。
(3) 形式参数也是局部变量。只在定义它的函数中有效。其他函数中不能直接引用形参。

(4) 在一个函数内部，可以在复合语句中定义变量，这些变量只在本复合语句中有效，这种复合语句也称为“**分程序**”或“**程序块**”。

2.全局变量：

程序的编译单位是源程序文件,一个源文件可以包含一个或若干个函数。在函数内定义的变量是局部变量,而在函数之外定义的变量称为**外部变量**,外部变量是**全局变量**(也称全程变量)。全局变量可以为本文件中其他函数所共用。它的有效范围为从定义变量的位置开始到本源文件结束。

设置全局变量的作用是增加了函数间数据联系的渠道。由于同一文件中的所有函数都能引用全局变量的值，因此如果在一个函数中改变了全局变量的值，就能影响到其他函数中全局变量的值。相当于各个函数间有直接的传递通道。由于函数的调用只能带回一个函数返回值，因此有时可以利用全局变量来增加函数间的联系渠道，通过函数调用能得到一个以上的值。

但是，建议不在必要时不要使用全局变量，原因如下:

① 全局变量在程序的全部执行过程中都占用存储单元，而不是仅在需要时才开辟单元。

② 它使函数的通用性降低了，因为如果在函数中引用了全局变量，那么执行情况会受到有关的外部变量的影响，如果将一个函数移到另一个文件中，还要考虑把有关的外部变量及其值一起移过去。但是若该外部变量与其他文件的变量同名时，就会出现问题。这就降低了程序的可靠性和通用性。在程序设计中，在划分模块时要求模块的“内聚性”强、与其他模块的“耦合性”弱。即模块的功能要单一（不要把许多互不相干的功能放到一个模块中），与其他模块的相互影响要尽量少，而用全局变量是不符合这个原则的。一般要求把C程序中的函数做成一个相对的封闭体，除了可以通过“实参—形参”的渠道与外界发生联系外，没有其他渠道。这样的程序移植性好，可读性强。

③ 使用全局变量过多，会降低程序的清晰性，人们往往难以清楚地判断出每个瞬时各个外部变量的值。由于在各个函数执行时都可能改变外部变量的值，程序容易出错。因此，要限制使用全局变量。