C 语言函数的调用

函数调用的一般形式为：

```
函数名(实参列表);
```

实参可以是常数、变量、表达式等，多个实参用逗号,分隔。

在 C 语言中，函数调用的方式有多种，例如：

```
1. // 函数作为表达式中的一项出现在表达式中
2. z = max(x, y);
3. m = n + max(x, y);
4. // 函数作为一个单独的语句
5. printf("%d", a);
6. scanf("%d", &b);
7. // 函数作为调用另一个函数时的实参
8. printf( "%d", max(x, y) );
9. total( max(x, y), min(m, n) );
```

在函数调用中还应该注意的一个问题是求值顺序。所谓求值顺序是指对实参列表中各个参数是自左向右使用呢，还是自右向左使用。对此，各系统的规定不一定相同。

【示例】在 VC6.0 和 C-Free 5.0 下运行以下代码。

```
1. #include <stdio.h>
2. int main(){
3.     int i=8;
4.     printf("%d %d %d %d\n",++i,++i,--i,--i);
5.     return 0;
6. }
```

运行结果：

```
8 7 6 7
```

可见 VC 6.0 是按照从右至左的顺序求值。如果按照从左至右求值，结果应为：

```
9 10 9 8
```

函数的嵌套调用

函数不能嵌套定义，但可以嵌套调用，也就是在一个函数的定义中出现对另一个函数的调用。这样就出现了函数的嵌套调用，即在被调函数中又调用其它函数。

【示例】计算 sum = 1! + 2! + 3! + … + (n-1)! + n!

分析：可以编写两个函数，一个用来计算阶乘，一个用来计算累加的和。

```
#include <stdio.h>

//求阶乘
long factorial(int n){
    int i;
    long result=1;
    for(i=1; i<=n; i++){
        result *= i;
    }
    return result;
}

// 求累加的和
long sum(long n){
    int i;
    long result = 0;
    for(i=1; i<=n; i++){
        //嵌套调用
        result += factorial(i);
    }
    return result;
}

int main(){
    printf("1!+2!+...+9!+10! = %ld\n", sum(10));
    return 0;
}
```

运行结果：

```
1!+2!+…+9!+10! = 4037913
```

函数声明和函数原型

C 语言代码由上到下依次执行，函数定义要出现在函数调用之前。

但是，如果在函数调用前进行了函数声明，那么函数定义就可以出现在任何地方了，甚至是其他文件。

函数声明的一般形式为：

```
返回值类型  函数名( 类型 形参, 类型 形参... );
```

或为：

```
返回值类型  函数名( 类型, 类型...);
```

函数声明给出了函数名、返回值类型、参数列表（参数类型）等与该函数有关的信息，称为函数原型（Function Prototype）。

函数原型的作用是告诉编译器与该函数有关的信息，让编译器知道函数的存在，以及存在的形式，即使函数暂时没有定义，也不会出错。

更改上面的代码，将 factorial 和 sum 函数的定义放到 main 函数后面：

```
1. #include <stdio.h>
2. 
3. // 函数声明
4. long factorial(int n);  //也可以写作 long factorial(int);
5. long sum(long n);  //也可以写作 long sum(long);
6. 
7. int main(){
8.     printf("1!+2!+...+9!+10! = %ld\n", sum(10));
9.     return 0;
10.}
11. 
12.//求阶乘
13.long factorial(int n){
14.    int i;
```

```
    long result=1;
    for(i=1; i<=n; i++){
        result *= i;
    }
    return result;
}

// 求累加的和
long sum(long n){
    int i;
    long result = 0;
    for(i=1; i<=n; i++){
        //嵌套调用
        result += factorial(i);
    }
    return result;
}
```

运行结果：

```
1!+2!+…+9!+10! = 4037913
```