C语言中的函数应用时易出现的错误

蒋春蕾[1]，杨 双[2]，岳富强[1]，刘 辉[3]

（1.西昌学院，四川 西昌 615022；2.桂林航天工业高等专科学校，广西 桂林 541004；
3.成都电子高专，四川 成都）

【摘 要】C语言程序完全由函数组成。除程序必须包含的main函数（主函数）和语言本身提供的库函数外，用户还可根据需要定义任意多个自己的函数。通过主函数调用其他函数（库函数，用户自定义函数），其他函数之间的相互调用，完成相应的程序功能。C 语言程序一般是由大量的小函数而不是由少量大函数构成的，这样可以让各部分相互充分独立，并且任务单一。文章对函数在C语言中的应用时易出现的错误进行了分析。

【关键词】 变量；函数；函数调用；参数；参数传递；返回值

【中图分类号】TP312 【文献标识码】A 【文章编号】1673-1891(2005)03-0090-04

一个C语言程序可由一个主函数和若干个函数构成，程序从主函数开始执行。 函数是一个自我包含的完成一定相关功能的执行代码段。我们可以把函数看成一个"黑盒子"，你只要将数据送进去就能得到结果，而函数内部究竟是如何工作的的，外部程序是不知道的。外部程序所知道的仅限于输入给函数什么以及函数输出什么。函数提供了编制程序的手段，使之容易读、写、理解、排除错误、修改和维护。

C语言的一个主要特点是可以建立库函数。Turbo C2.0提供的运行程序库有400多个函数， 每个函数都完成一定的功能，可由用户随意调用。这些函数总的分为输入输出函数、数学函数、字符串和内存函数、与BIOS和DOS有关的函数、 字符屏幕和图形功能函数、过程控制函数、目录函数等。除了系统提供的库函数外，为解决用户的专门需要，用户可以自己定义函数。函数是C语言的构造模块，它是C程序的关键部件，函数编写的准确性、可行性直接决定了程序的正确性、可行性。我们在应用函数的时候，常常会出现如下的一些问题。

1 常见错误一

```
main( )
 {int x;
scanf ("%d", x); /* 该语句有错 */
 printf("%d",x);}
```

语句scanf ("%d",x)中，scanf()函数需要的是参数的地址表列，因此务必在x的前面加上地址运算符"&"，将其改为scanf("%d",&x)；否则，printf()函数无法显示正确的x的值。这是初学者常犯的错误。Scanf、printf函数应用时应注意如下的问题：

（1） scanf()函数

I. scanf函数中的格式控制后面应当是变量地址，而不应是变量名。例如：如果a,b为整型变量，则：scanf("%d,%d",a,b)；是不对的，应该为 scanf("%d,%d",&a, &b)。

II. 如果在格式控制后面除了格式说明以外还有其他字符，则在输入时应输入与这些字符相同的字符。例如：scanf("%d，%d",&a, &b)，输入时应用如下形式3,4 如果输入时不用逗号而用空格或其他字符都是不对的。

III. 当用scanf函数给一个字符数组赋值时，由于数组名代表数组的起始地址，因此地址参数应用数组名。

如：char str[10];scanf("%s",str)；而用scanf("%s",&str)不对。

（2） printf()函数

I. 注意函数参数的求值顺序。例：

int i=2； printf("%d,%d,%d",i++,i++,i--)；

收稿日期：2005-07-01

作者简介：蒋春蕾(1979-)，女，助教，主要从事计算机科学技术与教育教学研究工作。

该程序的结果很容易判断为2,3,3。但turbo c2.0在执行printf()函数时,参数自右向左依次压入栈中,即先压入i--参数的值,再压入第二个i++参数的值,最后将第一个i++参数的值压入;弹出时则依次为i++,i++,i--,因此结果为2,1,2。即turbo c2.0对参数的求值顺序是自右而左。若没有注意函数的求值顺序和++,--运算的前、后缀运算,甚至会感到结果莫名其妙。

II. 注意(i++)+(i++)作为printf()输出项参数和作为赋值表达式一部分的区别。例如:

i=2;printf("%d",(i++)+(i++));在此语句中先处理右边的(i++),即先用i的原值2,然后i自加,这时左边的(i++)的值就是3,进行2+3运算,故结果为5。而在下列例子中:

i=2;j=((i++)+(i++));printf("%d",j);结果就不同了,先将原值i在取出来在整个表达式中应用,进行2+2运算,赋值给j,然后再执行两次自加操作,结果为4。

2　常见错误二

```
# include <stdio.h>
main( )
 {char s1[10],s2[10];
scanf ("%s",s1);
gets (s2);
printf("%s",s1);          printf("%s",s2);}
```

该程序中的语句gets (s2);并不接收键盘输入,原因是scanf()语句从流stdin中读完字符后,流stdin对于gets()而言已经不再为空,(gets()也从stdin中读字符),故在语句gets (s2);前应该释放stdin,即增加一条语句fflush(stdin),之后语句gets (s2);便可接收键盘输入。

3　常见错误三

```
main( )
 {float x,y;          scanf ("%f%f",&x,&y);
printf("%d", add (x,y));}      /* 调用add函数 */
float add(float x,float y)      /* 用户自定义函数add的定义 */
 {return  x+y;}
```

此程序初看没有什么错误,但是细心的读者会发现在主函数中printf("%d", add(x,y));语句表示main函数期望调用add()函数后能够返回的是一个整型的值,而add函数返回的则是一个浮点型的数据。如果主函数main()和被调用的函数add()是分开编译的话,该错误是不能被发现的。此类错误的修改方法就是要使用函数原型(function prototype),即要对用户定义的函数在主函数中对其进行声明,这样做可以在程序的编译阶段对调用函数的合法性进行全面的检查。使用函数原型不仅是良好的程序设计风格,而且能够尽快地查出错误,缩短调试时间。

再看下面这个程序:

```
main( )
 {float a,b; int c; scanf ("%f,%f",&a,&b);
c=max (a,b);          /* 调用max函数 */
printf ("%d",c);}
max  (float x,float y)      /* 用户自定义函数max的定义 */
 {float z;      z=x>y?x:y;      return (z);}
```

此程序中返回的z值的类型为浮点型,而max()函数的类型为整型,假如我们从键盘上输入的a,b的值分别为1.5,2.5,程序运行后并不能得到用户所期望的值2,原因仍然是因为用户没有使用函数原型,修改的方法是为该程序添加函数原型:

```
main( )
 {int max (float x,float y);     /* 用户自定义函数的声明 */
float a,b;          int c;
scanf ("%f,%f",&a,&b);
c=max (a,b);          /* 调用max函数 */
printf ("%d",c);}
max (float x,float y)          /* 用户自定义函数max的定义 */
 {float z;   z=x>y?x:y;      return (z);}
```

4　常见错误四

```
char *strcut  (char *s,  int m,  int n);
/* 用户自定义函数的声明 */
main ( )
 { static char s[ ]="Good  Morning!";
char *strcut ( ), *p;
p=strcut ( s,3,4);     /* 调用strcut函数 */
printf ("%s",p);
```

```
}
char *strcut(char *s, int m, int n)      /*用户自定义函数的定义*/
{char substr[20];
register int i
for(i=0;i<n;i++)          substr[i]=s[m+i-1];
substr[i]='max=%d,min=%d",max,min);}
```

该程序中若不给用户定义的指针变量赋值,令其有确定的指向，或不把max,min**变量定义为全局变量,程序都会出错。**

5 常见错误五

```
char p(char *s2);     /*用户自定义函数的声明*/
void main()
{char (* s1)(), ch;   /*定义指向函数的指针变量*/
s1=p();               /*此语句有错*/
ch=(*s1)("abcd");
printf("%c",ch);}
char p(char *s2)      /*用户自定义函数的定义*/
{return s2[1];}
```

语句char (* s1)()**表示定义了一个指向函数的指针变量,指针**s1**是一函数型指针,它需要的是某函数的地址,是专门用来存放函数的入口地址的。在给这种函数指针变量赋值的时候，只需要给出函数名即可,而不涉及到参数问题。因此**s1=p()**语句出错,该语句表示把**p()**返回的值赋给了一个指针,这显然不对,造成**ch=(*s1)("abcd")**语句得不到正确的结果。应该将**s1=p()**语句改为**s1=p**即可。再看下面的正确程序：**

```
main()
{int max(int,int);    /*用户自定义函数的声明*/
int (*p)();           /*定义指向函数的指针变量*/
int a,b,c;
p=max;                /*给函数指针变量赋值*/
scanf("%d,%d",&a,&b);
c=(*p)(a,b);          /*用函数指针变量调用函数*/
printf("max=%d",c);}
max(int x, int y)     /*用户自定义函数的定义*/
{int z;
if(x>y)     z=x;
else        z=y;      return(z);}
```

6 常见错误六

```
void main(int argc,char *argv[])
{char c;
FILE *fp1, *fp2;
if(argc!=3)
{printf("be lack of parameters");
exit(1);}
if(((fp1=fopen(argv[1],"rb"))==NULL)||((fp2=fopen(argv[2],"w+b"))==NULL))
{printf("can not open the file); exit(1);}
while(!feof(fp1))
{c=getc(fp1);          putc(c,fp2);}
fclose(fp1);           fcolse(fp2);}
```

程序的循环中,while(!feof(fp1))**是先判断**feof()**后读取文件,这导致当文件中的最后一个字符读出后,**feof()**仍然为**0**,程序继续循环,于是将结束符也作为一个字节写入新文件中，使得新文件比旧文件多一个字符。正确的方法应该是:先读取文件,后判断**feof()**。程序如下：**

```
void main(int argc,char *argv[])
{char c;
FILE *fp1, *fp2;
if(argc!=3)
{printf("be lack of parameters");
exit(1);}
if(((fp1=fopen(argv[1],"rb"))==NULL)||((fp2=fopen(argv[2],"w+b"))==NULL))
{printf("can not open the file); exit(1);}
while(1)
{c=getc(fp1);
if(!feof(fp1))        putc(c,fp2);
else break;}
fclose(fp1);          fcolse(fp2);}
```

其实,在C语言的程序设计中,对于函数的应用过程中还有许多可能出现的错误，比如对于函数的

变量的作用域的问题,若不仔细分析程序,也极有可能出错。C语言中变量可以在各个层次的子程序中加以说明。当然，内层中的变量即使与外层中的变量名字相同，它们之间也是没有关系的。只是,用户在设计、分析程序的时候,应该考虑清楚同名变量的作用域。如：

```
int  a=5；      /* 全局变量a的定义 */
fun(int  b)      /* 用户自定义函数的定义 */
{static  int  a=10；    /* 局部变量与全局变量同名,局部变量起作用 */
a+=b++；    /* 此处的变量a为局部变量a  */
printf("%d",a)；}
main(  )
{nt  c=20；      fun(  c  )；
a+=c++；    /* 此处的变量a为全局变量a */
printf("%d",a)；}
```

程序的运行结果为30　25，从程序运行的结果不难看出程序中各变量之间的关系，以及各个变量的作用域。

致谢:感谢罗明英副教授的指导！

参考文献：

〔1〕谭浩强.C程序设计〔M〕.清华大学出版社.
〔2〕谭浩强.C程序设计等级考试辅导〔M〕.清华大学出版社.
〔3〕谭浩强.程序设计〔M〕.清华大学出版社.
〔4〕黄迪明.C++程序设计基础〔M〕.电子工业出版社.

Mistake Apting to Appear when the Function is used in Language C

JIANG Chun-lei[1], YANG Shuang[2], YUE Fu-qiang[1], LIU Hui[3]

(*1.Information and technology department of Xichang College, Xichang 615022, Sichuan; 2.Aerospale Industry College of Guilin Guilin 541004, Guang Xi; 3. Chengdu Electromechanical College, Chengdu 61000, Sichuan*)

Abstract: C language procedure is totally made up of function. Except main function (main function) that procedure must include and language storehouse function that itself offers, users can also define a lot of one´s own function according to their needs. Transfer other function through main function (the function of the storehouse , user´s self-defining function), transferring each other between other functions, finish the corresponding procedure function. C language procedure is generally by a large amount of small function instead of forming by a small amount of great function, can let every part sufficient independence each other so, and the task is single . The mistake apt to appear has been analyzed at the time of the application of function in C language in this article.

Key words: Variable; Function; Function is transferred; Parameter; The parameter is transmitted; Returning value