

教程目录

1 编程基础

2 C语言初探

3 变量和数据类型

4 输入输出

5 分支结构和循环结构

6 C语言数组

7 C语言函数

8 预处理命令

9 C语言指针

9.1 1分钟彻底理解指针的概念

9.2 大话C语言指针变量

9.3 C语言指针变量的运算

9.4 数组指针（指向数组的指针）

9.5 字符串指针（指向字符串的指针）

9.6 C语言数组灵活多变的访问形式

9.7 指针变量作为函数参数

9.8 用C语言指针作为函数返回值

9.9 二级指针（指向指针的指针）

9.10 空指针NULL以及void指针

9.11 注意，数组和指针绝不等价

9.12 数组在什么时候会转换为指针

9.13 指针数组（每个元素都是指针）

9.14 一道题目教你玩转指针数组

9.15 指针与二维数组

9.16 函数指针（指向函数的指针）

9.17 只需一招，彻底攻克C语言指针

9.18 用main()函数接收控制台数据

9.19 对C语言指针的总结

10 结构体、位运算以及其他

11 文件操作

12 C语言调试

用C语言指针作为函数返回值

<上一节 下一节> 关注我们： 微信公众号 新浪微博 QQ交流群：588321099

C语言中文网推出辅导班啦，包括「C语言辅导班、C++辅导班、算法/数据结构辅导班」，全部都是一对一教学：一对一辅导 + 一对一答疑 + 布置作业 + 项目实践 + 永久学习。QQ在线，随时响应！

C语言允许函数的返回值是一个指针（地址），我们将这样的函数称为**指针函数**。下面的例子定义了一个函数strlong()，用来返回两个字符串中较长的一个：

```
01. #include <stdio.h>
02. #include <string.h>
03.
04. char *strlong(char *str1, char *str2){
05.     if(strlen(str1) >= strlen(str2)){
06.         return str1;
07.     }else{
08.         return str2;
09.     }
10. }
11.
12. int main(){
13.     char str1[30], str2[30], *str;
14.     gets(str1);
15.     gets(str2);
16.     str = strlong(str1, str2);
17.     printf("Longer string: %s\n", str);
18.
19.     return 0;
20. }
```

运行结果：

```
C Language✓
c.biancheng.net✓
Longer string: c.biancheng.net
```

用指针作为函数返回值时需要注意的一点是，函数运行结束后会销毁在它内部定义的所有局部数据，包括局部变量、局部数组和形式参数，函数返回的指针请尽量不要指向这些数据，C语言没有任何机制来保证这些数据会一直有效，它们在后续使用过程中可能会引发运行时错误。请看下面的例子：

```
01. #include <stdio.h>
02.
03. int *func(){
04.     int n = 100;
05.     return &n;
06. }
07.
08. int main(){
09.     int *p = func(), n;
10.     n = *p;
11.     printf("value = %d\n", n);
12.     return 0;
13. }
```

运行结果：

http://c.biancheng.net/cpp/html/3242.html

1/2

```
value = 100
```

n 是 func() 内部的局部变量，func() 返回了指向 n 的指针，根据上面的观点，func() 运行结束后 n 将被销毁，使用 *p 应该获取不到 n 的值。但是从运行结果来看，我们的推理好像是错误的，func() 运行结束后 *p 依然可以获取局部变量 n 的值，这个上面的观点不是相悖吗？

为了进一步看清问题的本质，不妨将上面的代码稍作修改，在第9~10行之间增加一个函数调用，看看会有什么效果：

```
01. #include <stdio.h>
02.
03. int *func(){
04.     int n = 100;
05.     return &n;
06. }
07.
08. int main(){
09.     int *p = func(), n;
10.     printf("c.biancheng.net\n");
11.     n = *p;
12.     printf("value = %d\n", n);
13.     return 0;
14. }
```

运行结果：

```
c.biancheng.net
value = -2
```

可以看到，现在 p 指向的数据已经不是原来 n 的值了，它变成了一个毫无意义的甚至有些怪异的值。与前面的代码相比，该段代码仅仅是在 *p 之前增加了一个函数调用，这一细节的不同却导致运行结果有天壤之别，究竟为什么呢？

前面我们说函数运行结束后会销毁所有的局部数据，这个观点并没错，大部分C语言教材也都强调了这一点。但是，这里所谓的销毁并不是将局部数据所占用的内存全部抹掉，而是程序放弃对它的使用权限，弃之不理，后面的代码可以随意使用这块内存。对于上面的两个例子，func() 运行结束后 n 的内存依然保持原样，值还是 100，如果使用及时也能够得到正确的数据，如果有其它函数被调用就会覆盖这块内存，得到的数据就失去了意义。

关于函数调用的原理以及函数如何占用内存的更多细节，我们将在《[C语言和内存](#)》专题中深入探讨，相信你必将有所顿悟，解开心中的谜团。

第一个例子在调用其他函数之前使用 *p 抢先获得了 n 的值并将它保存起来，第二个例子显然没有抓住机会，有其他函数被调用后才使用 *p 获取数据，这个时候已经晚了，内存已经被后来的函数覆盖了，而覆盖它的究竟是一份什么样的数据我们无从推断（一般是一个没有意义甚至有些怪异的值）。

C语言中文网推出辅导班啦，包括「C语言辅导班、C++辅导班、算法/数据结构辅导班」，全部都是一对一教学：一对一辅导 + 一对一答疑 + 布置作业 + 项目实践 + 永久学习。QQ在线，随时响应！

编程帮，一个分享编程知识的公众号。跟着站长一起学习，每天都有进步。

通俗易懂，深入浅出，一篇文章只讲一个知识点。

文章不深奥，不需要钻研，在公交、在地铁、在厕所都可以阅读，随时随地涨姿势。

文章不涉及代码，不烧脑细胞，人人都可以学习。

当你决定关注「编程帮」，你已然超越了90%的程序员！



微信扫描二维码关注

<上一节

下一节>

关注我们： 微信公众号

新浪微博

QQ交流群：588321099

关于C语言中文网 | 关于站长 | 如何才能完成一部教程 | 联系我们 | 网站地图 | 手机版网站

精美而实用的网站，关注编程技术，追求极致，让您轻松愉快的学习。

Copyright ©2011-2015 biancheng.net, All Rights Reserved, 陕ICP备15000209号

biancheng.net