**指向指针的指针**



[masterli1688](https://blog.csdn.net/masterli1688) 2019-03-06 18:27:30  7733  收藏 73

分类专栏： [c++](https://blog.csdn.net/masterli1688/category_8727026.html) [复习](https://blog.csdn.net/masterli1688/category_8727117.html) 文章标签： [c++](https://www.csdn.net/tags/Ntjacg0sNDk3Ny1ibG9n.html)

**指向指针的指针！！（能让初学者绕晕的东西）**

如果一个指针变量存放的又是另一个指针变量的地址，则称这个变量为指向指针的指针变量或指向指针的指针。

定义方式：   数据类型  \*\*变量名;

1. int a=10; //地址为&a
2. int \*p=&a; //指针地址为&p 如果是p就是a的地址
3. int \*\*p1=&p; //指针地址为&p1 如果是\*p1就是&p</span>

要注意这几个地址的区别，不能混淆了。

**指针运算符**

指针运算符有两种：

* 取地址运算符&：& 是单目运算符，其结合性为自右至左，功能是取变量的地址。
* 取内容运算符\*：\* 是单目运算符，其结合性为自右至左，用来表示指针变量所指的变量。在 \* 运算符之后跟的变量必须是指针变量。

需要注意的是指针运算符(\*)和指针变量说明中的指针说明符(\*)不是一回事。在指针变量说明中，“\*”是类型说明符，表示其后的变量是指针类型，而表达式中出现的“\*”则是一个运算符用以表示指针变量所指的变量。例如：

#include <stdio.h>

void main(){

int a=5;

int \*p=&a;

printf("%d\n", \*p);

}

运行结果：  
5  
  
int \*p=&a;表示指针变量p取得了整型变量a的地址,printf("%d\n",\*p);语句表示输出变量a的值。

**赋值运算**

指针变量的赋值运算有以下几种形式。  
  
1) 指针变量初始化赋值。  
  
2) 把一个变量的地址赋予指向相同数据类型的指针变量。例如：

1. int a, \*pa;
2. pa=&a; //把整型变量a的地址赋予整型指针变量pa

3) 把一个指针变量的值赋予指向相同类型变量的另一个指针变量。如：

1. int a, \*pa=&a, \*pb;
2. pb=pa; //把a的地址赋予指针变量pb

由于pa、pb均为指向整型变量的指针变量，因此可以相互赋值。  
  
4) 把数组的首地址赋予指向数组的指针变量。例如：

1. int a[5], \*pa;
2. pa=a; //数组名表示数组的首地址，可以赋予指向数组的指针变量pa

也可写为：

pa=&a[0]; //数组第一个元素的地址也是整个数组的首地址，也可赋予pa

当然也可采取初始化赋值的方法：

int a[5], \*pa=a;

**加减算术运算（针对数组）**

对于指向数组的指针变量，可以加上或减去一个整数n。设pa是指向数组a的指针变量，则pa+n、pa-n、pa++、++pa、pa-、--pa运算都是合法的。指针变量加或减一个整数n的意义是把指针指向的当前位置（指向某数组元素）向前或向后移动n个位置。应该注意，数组指针变量向前或向后移动一个位置和地址加1或减1在概念上是不同的。因为数组可以有不同的类型，各种类型的数组元素所占的字节长度是不同的。如指针变量加1，即向后移动1 个位置表示指针变量指向下一个数据元素的首地址。而不是在原地址基础上加1。例如：

1. int a[5],\*pa;
2. pa=a; //pa指向数组a，也是指向a[0]
3. pa=pa+2; //pa指向a[2]，即pa的值为&pa[2]

指针变量的加减运算只能对数组指针变量进行，对指向其它类型变量的指针变量作加减运算是毫无意义的。

重点（特别容易晕的地方）【示例】二维数组与指针举例。

#include <stdio.h>

void main(){

//a[0]表示第一行数组，它是变量名，不是单纯的变量，a[1],a[2]同理。

int a[3][4]={0,1,2,3,4,5,6,7,8,9,10,11}; //二维数组可以拆分成多个一维数组，C语言中允许这样拆分。

printf(" a=%d,",a); //第一行数组的第一个数据的地址（a[0][0]）

printf(" \*a=%d,",\*a); //第一行数组的第一个数据的地址（a[0][0]）

printf(" a[0]=%d,",a[0]); //第一行数组的第一个数据的地址（a[0][0]）

printf(" &a[0]=%d,",&a[0]); //第一行数组的第一个数据的地址（a[0][0]）

printf(" &a[0][0]=%d\n",&a[0][0]); //第一行数组的第一个数据的地址（a[0][0]）

printf("a+1=%d,",a+1); //第二行数组的第一个数据的地址（a[1][0]）

printf(" \*(a+1)=%d,",\*(a+1)); //第二行数组的第一个数据的地址（a[1][0]）

printf(" a[1]=%d,",a[1]); //第二行数组的第一个数据的地址（a[1][0]）

printf(" &a[1]=%d,",&a[1]); //第二行数组的第一个数据的地址（a[1][0]）

printf(" &a[1][0]=%d\n",&a[1][0]); //第二行数组的第一个数据的地址（a[1][0]）

printf("a+2=%d,",a+2); //第三行数组的第一个数据的地址（a[2][0]）

printf(" \*(a+2)=%d,",\*(a+2)); //第三行数组的第一个数据的地址（a[2][0]）

printf(" a[2]=%d,",a[2]); //第三行数组的第一个数据的地址（a[2][0]）

printf(" &a[2]=%d,",&a[2]); //第三行数组的第一个数据的地址（a[2][0]）

printf(" &a[2][0]=%d\n\n",&a[2][0]); //第三行数组的第一个数据的地址（a[2][0]）

printf(" a[1]+1=%-8d,",a[1]+1); //第二行数组的第二个数据的地址（a[1][1]）

printf(" \*(a+1)+1=%-8d\n",\*(a+1)+1); //第二行数组的第二个数据的地址（a[1][1]）

printf("\*(a[1]+1)=%-8d,",\*(a[1]+1)); //a[1][1]的数据

printf(" \*(\*(a+1)+1)=%-8d\n",\*(\*(a+1)+1)); //a[1][1]的数据

getchar();

}

运行结果：

1. a=2686736, \*a=2686736, a[0]=2686736, &a[0]=2686736, &a[0][0]=2686736
2. a+1=2686752, \*(a+1)=2686752, a[1]=2686752, &a[1]=2686752, &a[1][0]=2686752
3. a+2=2686768, \*(a+2)=2686768, a[2]=2686768, &a[2]=2686768, &a[2][0]=2686768
5. a[1]+1=2686756 , \*(a+1)+1=2686756
6. \*(a[1]+1)=5 , \*(\*(a+1)+1)=5

总结：①a[0]是第一个一维数组的数组名和首地址，a、a[0]、\*(a+0)、\*a、&a[0][0]是相等的；②a+1、a[1]、\*(a+1)、&a[1][0]是等同的。

所以推理出：a+i，a[i]，\*(a+i)，&a[i][0]是等同的。此外，&a[i]和a[i]也是等同的。因为在二维数组中不能把&a[i]理解为元素a[i]的地址，不存在元素a[i]。C语言规定，它是一种地址计算方法，表示数组a第i行首地址。由此，我们得出：a[i]，&a[i]，\*(a+i)和a+i也都是等同的。另外，a[0]也可以看成是a[0]+0，是一维数组a[0]的0号元素的首地址，而a[0]+1则是a[0]的第1个元素首地址，由此可得出a[i]+j则是一维数组a[i]的j号元素首地址，它等于&a[i][j]。由a[i]=\*(a+i)得a[i]+j=\*(a+i)+j。由于\*(a+i)+j是二维数组a的i行j列元素的首地址，所以，该元素的值等于\*(\*(a+i)+j)。