# **C语言的灵魂——指针**

## **什么是指针？**

**在了解指针之前先要弄清楚地址的概念。**

**如果在程序中定义了一个变量，在对程序进行编译时，系统就会给这个变量分配内存单元。编译系统根据城西中定义的变量类型，分配一定长度的空间。例如：整型变量分配4个字节，字符型分配1个字节，单精度分配4个字节等。内存区的每一个字节有一个编号，这就是“地址编号”，它就相当于旅馆中的房间号，每一个房间都可以看作一块内存区域，都可以用来存放东西，我们给每个房间都编一个房间门牌号，用于更好的区分每一个房间，内存中也是一样的，整个内存由很多个字节组成，每个字节都有其对应的“房间号”，这就是“地址”了。通过这个“房间号”就可以找到其对应的“房间”，然后就可以从房间里取东西，或者把东西放进房间里了。**

**理解了地址的概念之后，那所谓的指针，就是内存地址，也就是地址的编号，可以把“指针指向地址”理解成“用小本本把房间号记下来”，那这个小本本就相当于一个用于记房间号的指针了，一个变量的地址称为此变量的“指针”。**

## **指针常量与指针变量**

### **1、指针常量**

**之前有了解过不同数据类型的变量所占内存字节数的这个概念，那么系统在编译时给一个变量分配的内存地址就称为此变量的“指针”，这个指针的指向是无法改变的，所以又称为指针常量，数组的地址也是指针常量（也称为地址常量）。**

### **指针变量**

#### **（1）指针变量的概念**

**如果有一个变量专门用来存放另一个变量的地址，则称这个变量为“指针变量”，也就是说C语言中有一类变量是专门用来存储（指向）地址的，我们将它称为“指针变量”，指针变量的中存储的地址可以被改变，也就是可以改变指针变量的指向，就好比一张纸或一个小本本，写着一个房间的房间号，那把这个房间的房间号擦掉，写上另一个房间的房间号也是可以的，这就是指针变量和指针常量最大的区别所在了，可以改变指针变量的指向。**

#### **指针变量的定义**

**定义指针变量的一般格式：**

**类型名 \* 指针变量名;**

**例如：int \*p,\*q; char \*p1,\*q1; double \*p2,\*q2;**

**# 注意：左端的int、char等是在定义指针变量时必须指定的“基类型”。指针变量的基类型用来规定此指针变量可以指向的变量的类型。如：上面定义的\*p和\*q只能用于指向int整型变量的地址，p2和q2只能用于指向double双精度类型变量的地址。**

#### **指针变量的引用**

**与指针和地址相关运算符：’\*’（指针运算符）和’&’（取地址运算符）//区别对待位运算符&**

**例如：int a,\*p; p=&a; \*p=10;**

**在引用指针变量时，有以下几种情况：**

**①给指针变量赋值**

**如：int a = 10,b=20;**

**int \*p=&a;**

**//定义一个整型指针变量p，初始化p的值为a的地址，也就是p指向a地址**

**解引用：**

**\*p=30; //通过指针变量p引用a变量，改变a的值为30**

**//这里的’\*’为解引用符号，\*p引用指针变量p所指向地址中对应的值**

**scanf(“%d”,p); //scanf通过指针变量p给变量a赋值**

**printf(“%d\n”,\*p); //通过指针变量p输出变量a的值**

**\*p=b; //将b的值放入指针变量p所指向的内存地址中（a的地址单元中）**

**p=&b; //改变指针p的指向，指针p不再指向a的地址了，而是指向b的地址**

**printf(“%d\n”,\*p); //输出变量b的值**

**②输出内存地址编号**

**printf(“%p\n”,p); //以十六进制的格式输出指针变量p所指向地址的内存地址编号**

**printf(“%d\n”,&a); //以十进制的格式输出变量a所在的内存地址编号**

**printf(“%o\n”,&b); //以八进制的格式输出变量b所在的内存地址编号**

**printf(“%p\n”,&p); //以十六进制的格式输出指针变量p所在的内存地址编号**

## **指针变量作为函数参数**

**函数的参数不仅可以是整型、浮点型、字符型的数据，还可以是指针类型。它的作用是将一个变量的地址传递到另一个函数中。**

**1、函数参数为指针类型的函数**

**如：**

**void fun1(int x,int y)**

**{//这里定义了一个普通函数fun1**

**printf(“x=%d\ty=%d\n”,x++,y++);**

**}**

**void fun2(int \*x,int \*y)**

**{//这里定义了一个形参为整型指针类型函数fun2,其形参为指针类型的变量**

**printf(“x=%d\ty=%d\n”,\*x++,\*y++);**

**//注意：和printf(“x=%d\ty=%d\n”,x++,y++);的区别，也就是没有\*和有\*的区别**

**}**

**int main()**

**{**

**int a=10,b=20,\*p,\*q;**

**fun(&a,&b); //这里调用函数fun时，所传递的实参必须是地址**

**printf(“a=%d\tb=%d\n”,a,b);**

**p=&a; //使用整型指针变量p指向整型变量a的地址**

**q=&b; //使用整型指针变量q指向整型变量b的地址**

**fun(p,q); //这里使用指针变量p和q作为实参传递**

**printf(“\*p=%d\t\*q=%d\n”,\*p,\*q);**

**return 0;**

**}**

**2、指针函数**

**返回值为指针类型的函数称为“指针函数”。**

**如：**

**int \*fun3(int \*x,int \*y) //这是一个指针函数，返回值类型为整型int指针类型**

**{**

**\*x+=\*y;**

**printf(“%d”,++\*x);**

**return x; //返回指针变量x所指向的内存地址**

**}**

## **四、通过指针引用数组**

### **1、数组元素的指针**

**数组元素的地址表示：**

**如：int a[10] = {0,1,2,3,4,5,6,7,8,9},\*p;**

**&a[0]; //引用数组元素a[0]地址的表示方法**

### **2、指针指向数组元素**

**p=&a[1]; //指针变量p指向数组元素a[1]的地址**

### **3、指针指向的移动（指针的偏移）**

**p=&a[0];**

**\*p++; //指针指向地址中的数值加1**

**printf(“%#p\n”,p); //打印指针变量p所指向的地址编号**

**p++; //指针移动到数组元素a[1]的位置**

**printf(“%#X\n”,p); //打印移动后指针变量p所指向的地址编号**

**//指针变量++（或--）移动一次是移动其基类型大小的内存区域**

**指针的偏移：指针每次会以其基类型所占字节数为单位进行偏移。**

**for(int i=0;i<10;i++)**

**{**

**printf(“%d”,\*(p+i)); //通过指针移动引用数组元素，输出数组**

**}**

**for(int i=0;i<10;i++)**

**{**

**printf(“%d”,p[i])); //通过指针带下标的形式引用数组元素**

**}**

### **4、指针指向字符串**

**如：char \*p = “abcdefg”;**

**while (\*p)**

**{**

**printf("%c\t", \*(p++));**

**}**