# **指针的进阶应用**

## **指向函数的指针（函数指针）**

**之前有学到过函数参数是指针类型的函数以及指针函数，熟悉了一些基本的指针与函数的应用，了解了变量的地址与指针变量。**

**思考一下：那既然变量有地址，数组也有地址，那么函数会有对应的地址么？**

### **什么是函数指针？**

**首先，函数是会占内存空间的。在程序中定义了一个函数，在编译时，编译系统会为函数代码分配一段存储空间，这段存储空间就是函数的地址，这段地址的起始地址（又称入口地址）就称为这个函数的指针（或函数的首地址）。**

**既然函数也有地址，那么我们能不能用一个指针指向函数的地址呢？**

**既然都是地址，那么就可以用指针指向它。**

**指向整型变量地址的指针是整型指针，指向字符型变量地址的指针是字符型指针，指向单精度变量地址的指针是float型指针，那指向函数的指针是什么指针呢？**

**这就是接下来要接触到的函数指针了。**

### **函数指针的定义**

**简单来说，函数指针就是指向函数的指针。**

**定义函数指针的一般格式：数据类型 (\*函数名)(函数参数列表);**

**如：int (\*funp)(int,char);**

**//定义了一个指针函数，用于指向返回值类型为int型、函数参数为(int,char)型的函数**

### **函数指针的初始化及使用**

**如：**

**int funsum(int x,int y)**

**{ //求和函数**

**return x+y;**

**}**

**int funmax(int x,int y)**

**{ //求最大值函数**

**return x>y?x:y;**

**}**

**int funmin(int x,int y)**

**{ //求最小值函数**

**return x<y?x:y;**

**}**

**int main()**

**{**

**int (\*funp)(int,int)=funsum; //定义一个函数指针funp，初始化赋值指向函数funsum**

**int a=10,b=20,c; //函数指针有以下两个赋值方式和两种调用方式**

**c=(\*funp)(a,b); //通过函数指针funp调用函数funsum，(\*)区分funp是个函数指针**

**funp=&funmax; //改变函数指针funp的指向，&取函数地址，使其指向函数funmax**

**c=funp(a,b); //此时是通过函数指针funp调用函数funmax**

**funp=funmin; //改变函数指针funp的指向，可以省略&取址符，使其指向函数**

**funmin**

**c=funp(a,b); //通过函数指针funp调用函数funmin**

**return 0;**

**}**

### **使用函数指针作为函数参数（回调函数）**

**函数指针的一个重要的用途是把函数的地址作为参数传递到其他函数。**

**回调函数：通过函数指针调用的函数。如果你把函数的指针（地址）作为参数传递给另一个函数，当这个指针被用来调用其所指向的函数时，我们就称这是回调函数。回调函数不是由该函数的实现方直接调用，而是在特定的事件或条件发生时由另外的一方调用的，用于对该事件或条件进行响应。**

**如：**

**int function(int x,int y,int (\*fun)(int,int))**

**{//这里的函数形式参数fun为函数指针类型**

**return (\*fun)(x,y); //可以通过函数fun调用其所指向的函数**

**}**

**typedef int (\*a)(int,int);**

**function(1,2,sum); //调用时直接以函数名作为函数参数**

**function(10,20,max);**

### **使用typedef给指针函数取别名**

**typedef (\*funp)(int,int); //给 (\*)(int,int)类型的函数指针取一个别名为funp**

**funp \*p1; //用别名funp定义(\*)(int,int)类型的函数指针p1**

**p1 = sum; //函数指针p1指向函数sum**

**p1(1,2); //通过函数指针p1调用函数**

### **指针函数和函数指针的区别**

**所谓的指针函数，其本质上是个函数，是返回值为指针类型的函数**

**所谓的函数指针，其本质上是个指针，是指向函数的指针**

## **指针数组和数组指针**

### **指针数组**

1. **什么指针数组**

**所谓指针数组，其本质上是一个数组，数组中的每一个元素都是指针类型的，都可以指向对应数据类型的地址。**

1. **指针数组的定义**

**定义的一般格式：数据类型 \*指针数组名[数组元素个数];**

**如：int \*p[6]; //定义一个指针数组，有6个元素，分别可以指向六个地址**

1. **指针数组的使用**

**使用指针数组指向二维字符数组**

**char arr[10][10],\*p[10];**

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

**{**

**p[i]=arr[i]; //指针数组p中的每个元素指向二维数组arr的每一行**

**}**

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

**{**

**scanf(“%s”,p[i]); //使用指针数组p给二维数组arr赋值**

**}**

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

**{**

**printf(“%s”,p[i]); //使用指针数组p输出二维数组arr**

**}**

**使用指针数组存储字符串的形式代替二维字符数组，达到省空间的目的**

**char \*p[6]={“abc”,”123456”,”abcdefg”,”hello world!”,”xiaowei”,”x”}**

### **数组指针**

1. **什么是数组指针**

**所谓的数组指针，其本质上是一个指针，是一个用于指向数组地址的指针。**

1. **数组指针的定义**

**定义的一般格式：数据类型 (\*指针变量名)[所指向数组的大小];**

**如：int a[3][4];**

**int (\*p)[4]; //定义一个整型数组指针p，用于指向大小为4的整型数组**

**//行指针**

**p=a; //将整型数组指针p指向二维数组a的第一行**

1. **数组指针的使用**

**使用数组指针完成二维数组的输入和输出**

**int a[3][6];**

**int (\*p)[6];**

**p = a;**

**for (int i = 0; i < 3; i++)**

**for (int j = 0; j < 6;j++)**

**{**

**scanf("%d",&p[i][j]);**

**}**

**for (int i = 0; i < 3; i++)**

**for (int j = 0; j < 6;j++)**

**{**

**printf("%d\t",p[i][j]);**

**}**

1. **数组指针的移动**

**p++; //移动一整行**

**\*(\*(p+i)+j) //可以把数组指针理解成为一个二级指针，通过两次解引用得到元素值**

**p[i][j]; //指针带数组下标的形式访问数组元素**

## **指针常量与常量指针**

1. **指针常量**

**指针常量就是指向常量的指针**

**const int \*p;**

**它的指向能够被改变，但是不能够改变值**

1. **常量指针**

**常量指针就是指针的指向不能够被改变的指针**

**int \* const p;**

**它的能够改变值，但是不能够改变指向**

**示例：**

**int a = 10,b=20;**

**const int \*p;**

**p = &a;**

**printf("%d\n",\*p);**

**//\*p=30; //值不能被改变**

**p = &b; //可以改变指向**

**printf("%d\n",\*p);**

**int \* const q = &a; //必须初始化指针指向**

**\*q = 30; //可以改变值**

**//q = &b; //不能改变指向**

**printf("%d\n",\*q);**

**指针常量和常量指针一般用于函数参数的传递，为了使在函数使用中不改变值以及指针的指向。**