**指针**

前言：优先级比较 ()>[]>\*

（一）指针和数组

<1>指针数组

1. 字面分析：实质是一个数组，数组存储的元素类型为指针。
2. 定义与初始化

格式：类型 \*数组名【】

初始化：char \*ptr[3] = {"sundi","liushasha","hulei"};

说明：指针的类型是char \* [3]；指针所指向的数据类型是char

比起二维数组：

char str[3][10] = {

"jsdhjs",

"bhhzhz",

"jhj"

};

有两大好处：一、不用像二维数组一样必须指定元素的个数；二、指针操作很方便

printf("%s %c\n",\*(ptr+1),\*(\*(ptr+1)+2)); //ptr[1]：liushasha ptr[1][2]：u

<2>数组指针

a, 字面分析：实质是一个指针，这个指针指向一个数组。

b, 定义与初始化

格式：类型 （\*指针名）【】

初始化：int array[2][3] = {1,2,3,4,5,6};

int (\*p)[2];

p = array;

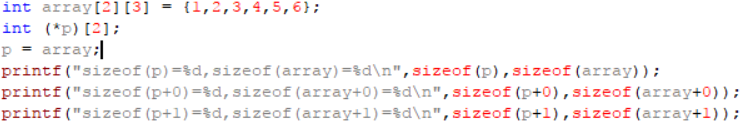
或者：int array[6] = {11,12,13,14,15,16};

p = &array;

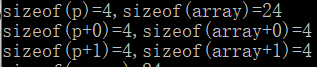
//p = array也不会报错，因为&array（整个数组的首地址）和array（数组首元素的首地址）的地址是一样的。

1. 数组名作为指针变量传参

代码如下：



输出结果:



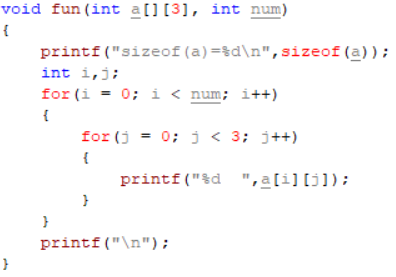
sizeof(array)=24, 即是6\*sizeof(int)，此时array表示的是数组名

sizeof(array+0), 即是sizeof(int \*), 此时array相当于指针变量，（array+i）此时array相当于指针变量

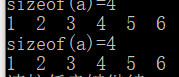
1. 作为函数参数传递

作为形参时，数组名array此时相当于指针变量来传递，sizeof(array) = 4





输出结果：



（二）指针和函数

<1>指针函数

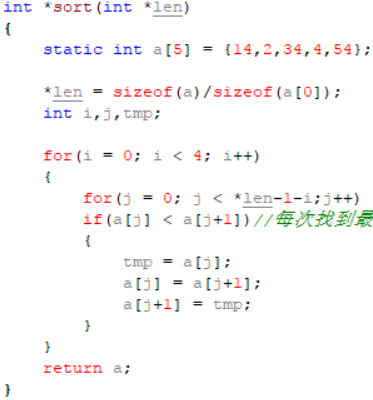
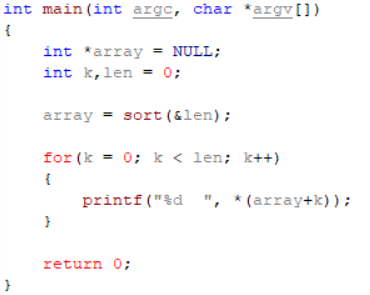
a, 字面分析：实质是一个函数，函数返回值类型为指针类型，它必须用同类型的指针变量来接收。

1. 定义与初始化

格式：类型 \*函数名(参数1，参数2，，，)

1. 代码分析

pointer\_func.c



输出结果：

注：必须要在代码中使用static类型，我们在sort函数中定义的数组a的，是在栈中申请的内存空间，当函数调用完成之后，函数中的局部变量默认会被消除掉。也就是说，我们在sort函数内部的一些操作是可行的。但是，一旦函数sort被调用完成，a中存储的数据就会被消掉。我们会获得未知的数据。因此，设置static，保证函数结束后，数组a的内存不会被释放掉。

<2>函数指针

a, 字面分析：实质是一个指针，保存函数的入口地址，是一个指向函数的指针变量 。

1. 定义与初始化

格式：类型 （\*指针名）(参数1，参数2，，，)

例如：

int (\*fun)(int ,int );

fun = max;

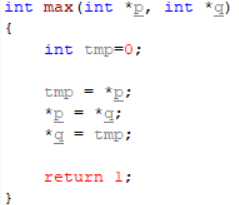
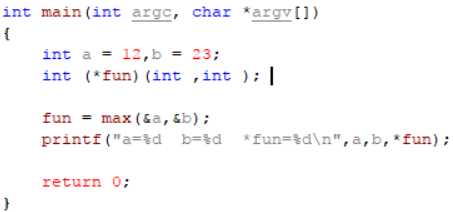
注：fun是一个指针变量，它指向max函数的地址，max函数的类型肯定是max(int,int）类型。

函数指针变量fun，只能用来保存返回值是int，并且有两个int参数的函数入口地址

Int max(int a, int b);

1. 代码说明

Func\_pointer.c



输出结果：

（三）二维数组与指针的应用

定义：二维数组 a[2][3];

行指针：a就是首地址，是一个指针也是一个常量，指向第一个元素的地址，也是指向第一行的首地址，是指首行一整行，并不是指某个具体元素。称为“行指针”。

行指针如下表：

|  |  |  |
| --- | --- | --- |
| **表示形式** | **含义** | **指针类型** |
| **a或者a+0** | **指向第1行** | **行指针** |
| **a+1** | **指向第2行** | **行指针** |
| **a+2** | **指向第3行** | **行指针** |

列指针：a[0]是这个数组的首地址，也就是第一个元素的地址，只不过指针方向是指向列的方向。a[0]+0、a[0]+1、a[0]+2都是指具体的元素，那么我们称之为“列指针”.

将第一行a[0]视为一维数组，得到下表：

|  |  |  |
| --- | --- | --- |
| **表示形式** | **含义** | **指针类型** |
| **a[0]** | **是一维数组的名称，也是它的首地址，而且是第1个元素的地址（a[0]+0）** | **列指针** |
| **a[0]+1** | **第1行，第2个元素的地址** | **列指针** |
| **a[0]+2** | **第1行，第3个元素的地址** | **列指针** |

行指针和列指针之间可以互相转化：

\*行指针----列指针

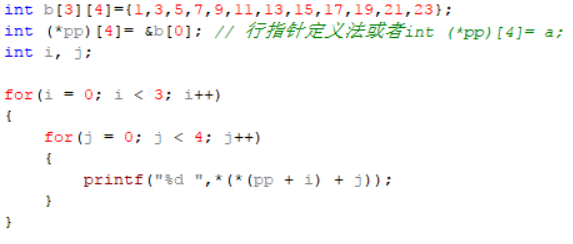
&列指针----行指针

如下表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **行指针** | **转换成：列指针** | **列指针等价表示** | **内容** | **内容等价表示** | **含义** |
| **a或a+0** | **\*a** | **a[0]** | **\*a[0]** | **\*(\*a)** | **a[0][0]** |
| **a+1** | **\*(a+1)** | **a[1]** | **\*a[1]** | **\*(\*(a+1))** | **a[1][0]** |
| **a+2** | **\*(a+2)** | **a[2]** | **\*a[2]** | **\*(\*(a+2))** | **a[2][0]** |

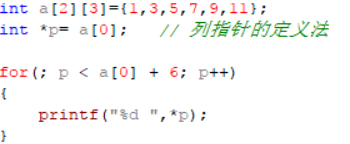
|  |  |  |  |
| --- | --- | --- | --- |
| **列指针** | **行指针** | **等价表示** | **含义** |
| **a[0]** | **&a[0]** | **&a或&(a+0)** | **第1行** |
| **a[1]** | **&a[1]** | **&(a+1)** | **第2行** |
| **a[2]** | **&a[2]** | **&(a+2)** | **第3行** |

行指针打印二维数组



输出结果：

列指针打印二维数组：



输出结果：