数组名

# 1数组名

|  |
| --- |
| //二维数组也是线性排列的 |
| void printArray(int \*a, int size)  {  int i = 0;  printf("printArray: %d\n", sizeof(a));  for(i=0; i<size; i++)  {  printf("%d\n", a[i]);  }  }  int main()  {  int a[2][3] = {{1, 2, 3}, {4, 5, 6}};  char cc[10][30];  int\* p = &a[0][0];  printf("sizeof(&a):%d \n", sizeof(&a));  printf("sizeof(a):%d \n", sizeof(a));  printf("sizeof(\*a):%d \n", sizeof(\*a));  printf("sizeof(&cc):%d \n", sizeof(&cc));  printf("sizeof(cc):%d \n", sizeof(cc));  printf("sizeof(\*cc):%d \n", sizeof(\*cc));  //printArray(p, 6);  getchar();  return 0;  } |
| 2、本质分析  //int a[5] 一维数组名代表数组首元素的地址  //int a[5] ===> a的类型为int\*  //二维数组名同样代表数组首元素的地址  //int b[2][5]===>b的类型为int(\*)[5]  //如何测试：指针也是一种数据类型，它的数据类型是指它所执行的内存空间的数据类型  //如何测试b的步长?  //推导。。。。。。。 |
| //结论:二维数组名 char cc[10][30] 是一个数组指针，char (\*)[30] |
|  |

# 2多维数组做函数参数退化

|  |
| --- |
| 1. C语言中只会以机械式的值拷贝的方式传递参数（实参把值传给形参） |
| int fun(char a[20], size\_t b) {    printf("%d\t%d",b,sizeof(a)); } |
| 原因1：高效 |
| 原因2： C语言处理a[n]的时候，它没有办法知道n是几，它只知道&n[0]是多少，它的值作为参数传递进去了 虽然c语言可以做到直接int fun(char a[20])，然后函数能得到20这个数字，但是，C没有这么做。 |
|  |
| 2、二维数组参数同样存在退化的问题 |
| 二维数组可以看做是一维数组  二维数组中的每个元素是一维数组  二维数组参数中第一维的参数可以省略  void f(int a[5]) ====》void f(int a[]); ===》 void f(int\* a);  void g(int a[3][3])====》 void g(int a[][3]); ====》 void g(int (\*a)[3]); |
| 3、等价关系 |
| 数组参数 等效的指针参数    一维数组 char a[30] 指针 char\*  指针数组 char \*a[30] 指针的指针 char \*\*a  二维数组 char a[10][30] 数组的指针 char(\*a)[30] |