

# 第15讲：深入理解指针(5)

## 目录

1. sizeof和strlen的对比
2. 数组和指针笔试题解析
3. 指针运算笔试题解析

正文开始

## 1. sizeof和strlen的对比

### 1.1 sizeof

在学习操作符的时候，我们学习了 `sizeof`，`sizeof` 计算变量所占内存空间大小的，单位是字节，如果操作数是类型的话，计算的是使用类型创建的变量所占内存空间的大小。

`sizeof` 只关注占用内存空间的大小，不在乎内存中存放什么数据。

比如：

```
1 #include <stdio.h>
2 int main()
3 {
4     int a = 10;
5     printf("%d\n", sizeof(a));
6     printf("%d\n", sizeof a);
7     printf("%d\n", sizeof(int));
8
9     return 0;
10 }
```

### 1.2 strlen

`strlen` 是C语言库函数，功能是求字符串长度。函数原型如下：

```
1 size_t strlen ( const char * str );
```

统计的是从 `strlen` 函数的参数 `str` 中这个地址开始向后，`\0` 之前字符串中字符的个数。

`strlen` 函数会一直向后找 `\0` 字符，直到找到为止，所以可能存在越界查找。

```
1 #include <stdio.h>
2 int main()
3 {
4     char arr1[3] = {'a', 'b', 'c'};
5     char arr2[] = "abc";
6     printf("%d\n", strlen(arr1));
7     printf("%d\n", strlen(arr2));
8
9     printf("%d\n", sizeof(arr1));
10    printf("%d\n", sizeof(arr2));
11    return 0;
12 }
```

### 1.3 sizeof 和 strlen的对比

| sizeof  | strlen  |
|---|---|
| <ol style="list-style-type: none"> <li>1. sizeof是操作符</li> <li>2. sizeof计算操作数所占内存的大小，单位是字节</li> <li>3. 不关注内存中存放什么数据</li> </ol> | <ol style="list-style-type: none"> <li>1. strlen是库函数，使用需要包含头文件 <code>string.h</code></li> <li>2. strlen是求字符串长度的，统计的是 <code>\0</code> 之前字符的个数</li> <li>3. 关注内存中是否有 <code>\0</code>，如果没有 <code>\0</code>，就会持续往后找，可能会越界</li> </ol> |

## 2. 数组和指针笔试题解析

### 2.1 一维数组

```
1 int a[] = {1,2,3,4};
2 printf("%d\n", sizeof(a));
3 printf("%d\n", sizeof(a+0));
4 printf("%d\n", sizeof(*a));
5 printf("%d\n", sizeof(a+1));
6 printf("%d\n", sizeof(a[1]));
7 printf("%d\n", sizeof(&a));
8 printf("%d\n", sizeof(*&a));
9 printf("%d\n", sizeof(&a+1));
```

```
10 printf("%d\n", sizeof(&a[0]));  
11 printf("%d\n", sizeof(&a[0]+1));
```

## 2.2 字符数组

代码1:

```
1 char arr[] = {'a','b','c','d','e','f'};  
2 printf("%d\n", sizeof(arr));  
3 printf("%d\n", sizeof(arr+0));  
4 printf("%d\n", sizeof(*arr));  
5 printf("%d\n", sizeof(arr[1]));  
6 printf("%d\n", sizeof(&arr));  
7 printf("%d\n", sizeof(&arr+1));  
8 printf("%d\n", sizeof(&arr[0]+1));
```

代码2:

```
1 char arr[] = {'a','b','c','d','e','f'};  
2 printf("%d\n", strlen(arr));  
3 printf("%d\n", strlen(arr+0));  
4 printf("%d\n", strlen(*arr));  
5 printf("%d\n", strlen(arr[1]));  
6 printf("%d\n", strlen(&arr));  
7 printf("%d\n", strlen(&arr+1));  
8 printf("%d\n", strlen(&arr[0]+1));
```

代码3:

```
1 char arr[] = "abcdef";  
2 printf("%d\n", sizeof(arr));  
3 printf("%d\n", sizeof(arr+0));  
4 printf("%d\n", sizeof(*arr));  
5 printf("%d\n", sizeof(arr[1]));  
6 printf("%d\n", sizeof(&arr));  
7 printf("%d\n", sizeof(&arr+1));  
8 printf("%d\n", sizeof(&arr[0]+1));
```

代码4:

```
1 char arr[] = "abcdef";
2 printf("%d\n", strlen(arr));
3 printf("%d\n", strlen(arr+0));
4 printf("%d\n", strlen(*arr));
5 printf("%d\n", strlen(arr[1]));
6 printf("%d\n", strlen(&arr));
7 printf("%d\n", strlen(&arr+1));
8 printf("%d\n", strlen(&arr[0]+1));
```

代码5:

```
1 char *p = "abcdef";
2 printf("%d\n", sizeof(p));
3 printf("%d\n", sizeof(p+1));
4 printf("%d\n", sizeof(*p));
5 printf("%d\n", sizeof(p[0]));
6 printf("%d\n", sizeof(&p));
7 printf("%d\n", sizeof(&p+1));
8 printf("%d\n", sizeof(&p[0]+1));
```

代码6:

```
1 char *p = "abcdef";
2 printf("%d\n", strlen(p));
3 printf("%d\n", strlen(p+1));
4 printf("%d\n", strlen(*p));
5 printf("%d\n", strlen(p[0]));
6 printf("%d\n", strlen(&p));
7 printf("%d\n", strlen(&p+1));
8 printf("%d\n", strlen(&p[0]+1));
```

## 2.3 二维数组

```

1 int a[3][4] = {0};
2 printf("%d\n", sizeof(a));
3 printf("%d\n", sizeof(a[0][0]));
4 printf("%d\n", sizeof(a[0]));
5 printf("%d\n", sizeof(a[0]+1));
6 printf("%d\n", sizeof(*(a[0]+1)));
7 printf("%d\n", sizeof(a+1));
8 printf("%d\n", sizeof(*(a+1)));
9 printf("%d\n", sizeof(&a[0]+1));
10 printf("%d\n", sizeof(*(&a[0]+1)));
11 printf("%d\n", sizeof(*a));
12 printf("%d\n", sizeof(a[3]));

```

数组名的意义:

1. sizeof(数组名), 这里的数组名表示整个数组, 计算的是整个数组的大小。
2. &数组名, 这里的数组名表示整个数组, 取出的是整个数组的地址。
3. 除此之外所有的数组名都表示首元素的地址。

### 3. 指针运算笔试题解析

#### 3.1 题目1:

```

1 #include <stdio.h>
2
3 int main()
4 {
5     int a[5] = { 1, 2, 3, 4, 5 };
6     int *ptr = (int *)(&a + 1);
7     printf( "%d,%d", *(a + 1), *(ptr - 1));
8     return 0;
9 }
10 //程序的结果是什么?

```

#### 3.2 题目2

```

1 //在X86环境下
2 //假设结构体的大小是20个字节
3 //程序输出的结果是啥?
4 struct Test
5 {

```

```
6     int Num;
7     char *pcName;
8     short sDate;
9     char cha[2];
10    short sBa[4];
11 }*p = (struct Test*)0x100000;
12
13 int main()
14 {
15     printf("%p\n", p + 0x1);
16     printf("%p\n", (unsigned long)p + 0x1);
17     printf("%p\n", (unsigned int*)p + 0x1);
18     return 0;
19 }
```

### 3.3 题目3

```
1 #include <stdio.h>
2 int main()
3 {
4     int a[3][2] = { (0, 1), (2, 3), (4, 5) };
5     int *p;
6     p = a[0];
7     printf( "%d", p[0]);
8     return 0;
9 }
```

### 3.4 题目4

```
1 //假设环境是x86环境，程序输出的结果是啥？
2 #include <stdio.h>
3
4 int main()
5 {
6     int a[5][5];
7     int(*p)[4];
8     p = a;
9     printf( "%p,%d\n", &p[4][2] - &a[4][2], &p[4][2] - &a[4][2]);
10    return 0;
11 }
```

### 3.5 题目5

```
1 #include <stdio.h>
2 int main()
3 {
4     int aa[2][5] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
5     int *ptr1 = (int *)(&aa + 1);
6     int *ptr2 = (int *)(* (aa + 1));
7     printf( "%d,%d", *(ptr1 - 1), *(ptr2 - 1));
8     return 0;
9 }
```

### 3.6 题目6

```
1 #include <stdio.h>
2 int main()
3 {
4     char *a[] = {"work","at","alibaba"};
5     char**pa = a;
6     pa++;
7     printf("%s\n", *pa);
8     return 0;
9 }
```

### 3.7 题目7

```
1 #include <stdio.h>
2 int main()
3 {
4     char *c[] = {"ENTER","NEW","POINT","FIRST"};
5     char**cp[] = {c+3,c+2,c+1,c};
6     char***cpp = cp;
7     printf("%s\n", *++cpp);
8     printf("%s\n", *--++cpp+3);
9     printf("%s\n", *cpp[-2]+3);
10    printf("%s\n", cpp[-1][-1]+1);
11    return 0;
12 }
```

完

比特就业课