第16讲：深入理解指针(6)

目录

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：

```c
#include <stdio.h>
int main()
{
    int a = 10;
    printf("%d\n", sizeof(a));
    printf("%d\n", sizeof a);
    printf("%d\n", sizeof(int));

    return 0;
}
```

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：

```c
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。

```
#include <stdio.h>
int main()
{
    char arr1[3] = {'a', 'b', 'c'};
    char arr2[] = "abc";
    printf("%d\n", strlen(arr1));
    printf("%d\n", strlen(arr2));

    printf("%d\n", sizeof(arr1));
    printf("%d\n", sizeof(arr2));
    return 0;
}
```

1.3 sizeof 和 strlen的对比

sizeof	strlen
1. sizeof是操作符 2. sizeof计算操作数所占内存的大小，单位是字节 3. 不关注内存中存放什么数据	1. strlen是库函数，使用需要包含头文件 `string.h` 2. srtlen是求字符串长度的，统计的是 `\0` 之前字符的个数 3. 关注内存中是否有 `\0` ，如果没有 `\0` ，就会持续往后找，可能会越界

2. 数组和指针笔试题解析

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。

2.1 一维数组

```
int a[] = {1,2,3,4};
```

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

2.2 字符数组

代码1：

```
#include <stdio.h>
int main()
{
    char arr[] = {'a','b','c','d','e','f'};
    printf("%d\n", sizeof(arr));
    printf("%d\n", sizeof(arr+0));
    printf("%d\n", sizeof(*arr));
    printf("%d\n", sizeof(arr[1]));
    printf("%d\n", sizeof(&arr));
    printf("%d\n", sizeof(&arr+1));
    printf("%d\n", sizeof(&arr[0]+1));
    return 0;
}
```

代码2：

```
#include <stdio.h>
#include <string.h>

int main()
{
    char arr[] = {'a','b','c','d','e','f'};
    printf("%d\n", strlen(arr));
    printf("%d\n", strlen(arr+0));
    printf("%d\n", strlen(*arr));
    printf("%d\n", strlen(arr[1]));
    printf("%d\n", strlen(&arr));
    printf("%d\n", strlen(&arr+1));
    printf("%d\n", strlen(&arr[0]+1));
```

```
    return 0;
}
```

代码3：

```
#include <stdio.h>
int main()
{
    char arr[] = "abcdef";
    printf("%d\n", sizeof(arr));
    printf("%d\n", sizeof(arr+0));
    printf("%d\n", sizeof(*arr));
    printf("%d\n", sizeof(arr[1]));
    printf("%d\n", sizeof(&arr));
    printf("%d\n", sizeof(&arr+1));
    printf("%d\n", sizeof(&arr[0]+1));
    return 0;
}
```

代码4：

```
#include <stdio.h>
#include <string.h>
int main()
{
    char arr[] = "abcdef";
    printf("%d\n", strlen(arr));
    printf("%d\n", strlen(arr+0));
    printf("%d\n", strlen(*arr));
    printf("%d\n", strlen(arr[1]));
    printf("%d\n", strlen(&arr));
    printf("%d\n", strlen(&arr+1));
    printf("%d\n", strlen(&arr[0]+1));
    return 0;
}
```

代码5：

```c
#include <stdio.h>
int main()
{
    char *p = "abcdef";
    printf("%d\n", sizeof(p));
    printf("%d\n", sizeof(p+1));
    printf("%d\n", sizeof(*p));
    printf("%d\n", sizeof(p[0]));
    printf("%d\n", sizeof(&p));
    printf("%d\n", sizeof(&p+1));
    printf("%d\n", sizeof(&p[0]+1));
    return 0;
}
```

代码6：

```c
#include <stdio.h>
#include <string.h>
int main()
{
    char *p = "abcdef";
    printf("%d\n", strlen(p));
    printf("%d\n", strlen(p+1));
    printf("%d\n", strlen(*p));
    printf("%d\n", strlen(p[0]));
    printf("%d\n", strlen(&p));
    printf("%d\n", strlen(&p+1));
    printf("%d\n", strlen(&p[0]+1));
    return 0;
}
```

2.3 二维数组

```c
#include <stdio.h>

int main()
{
    int a[3][4] = {0};
    printf("%d\n",sizeof(a));
    printf("%d\n",sizeof(a[0][0]));
```

```
    printf("%d\n",sizeof(a[0]));
    printf("%d\n",sizeof(a[0]+1));
    printf("%d\n",sizeof(*(a[0]+1)));
    printf("%d\n",sizeof(a+1));
    printf("%d\n",sizeof(*(a+1)));
    printf("%d\n",sizeof(&a[0]+1));
    printf("%d\n",sizeof(*(&a[0]+1)));
    printf("%d\n",sizeof(*a));
    printf("%d\n",sizeof(a[3]));
    return 0;
}
```

3. 指针运算笔试题解析

3.1 题目1:

```
#include <stdio.h>

int main()
{
    int a[5] = { 1, 2, 3, 4, 5 };
    int *ptr = (int *)(&a + 1);
    printf( "%d,%d", *(a + 1), *(ptr - 1));
    return 0;
}
//程序的结果是什么?
```

a数组

1	2	3	4	5

3.2 题目2

```
//在X86环境下
```

```c
//假设结构体的大小是20个字节
//程序输出的结果是啥?
struct Test
{
    int Num;
    char *pcName;
    short sDate;
    char cha[2];
    short sBa[4];
}*p = (struct Test*)0x100000;

int main()
{
    printf("%p\n", p + 0x1);
    printf("%p\n", (unsigned long)p + 0x1);
    printf("%p\n", (unsigned int*)p + 0x1);
    return 0;
}
```

3.3 题目3

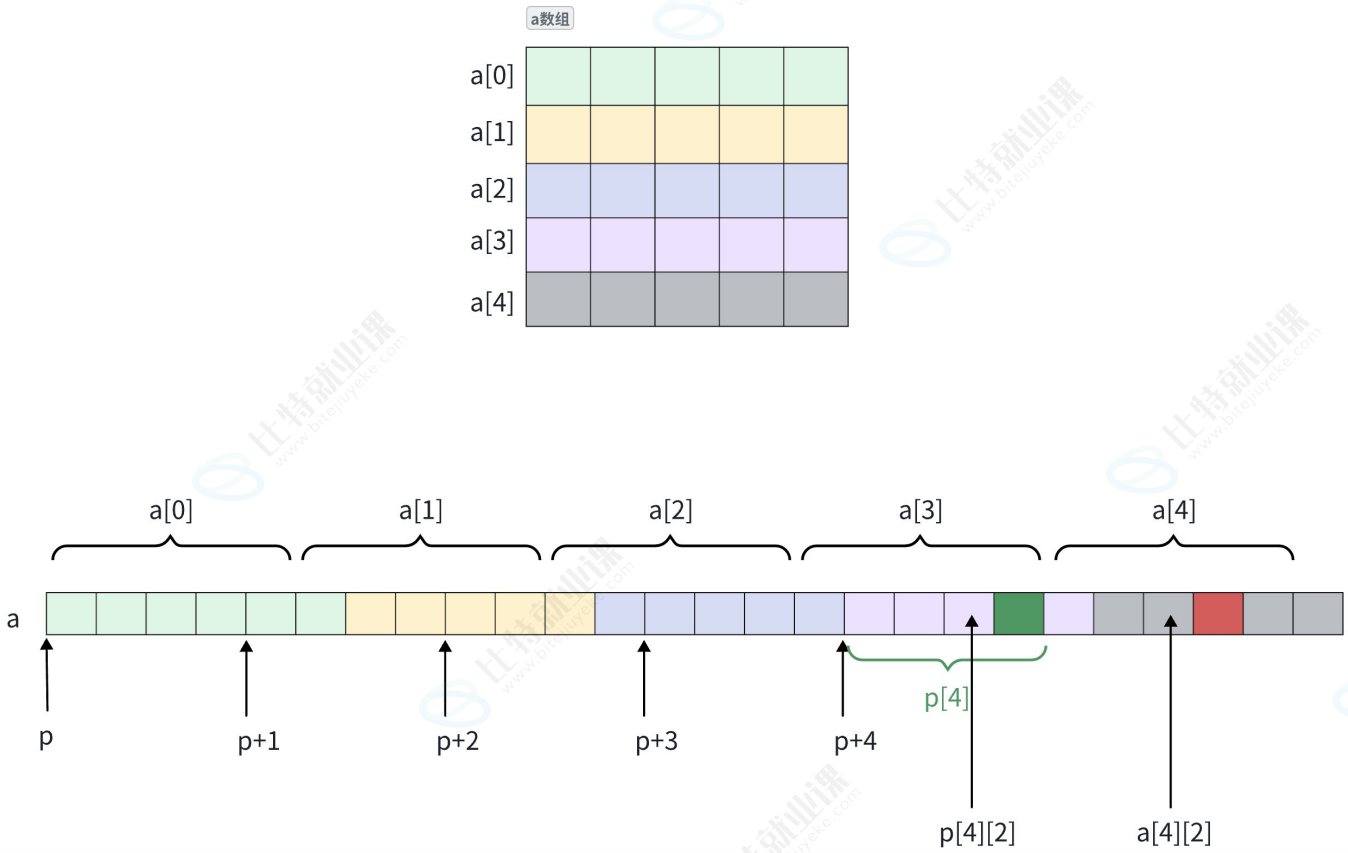
```c
#include <stdio.h>
int main()
{
    int a[3][2] = { (0, 1), (2, 3), (4, 5) };
    int *p;
    p = a[0];
    printf( "%d", p[0]);
    return 0;
}
```

3.4 题目4

```c
//假设环境是x86环境，程序输出的结果是啥?
#include <stdio.h>

int main()
{
    int a[5][5];
    int(*p)[4];
    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

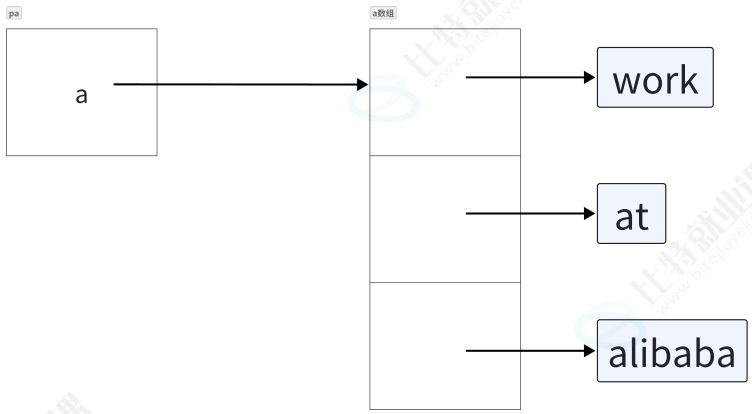
```
1   #include <stdio.h>
2   int main()
3   {
4       int aa[2][5] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
5       int *ptr1 = (int *)(&aa + 1);
6       int *ptr2 = (int *)(*(aa + 1));
7       printf( "%d,%d", *(ptr1 - 1), *(ptr2 - 1));
8       return 0;
9   }
```

aa数组

1	2	3	4	5
6	7	8	9	10

3.6 题目6

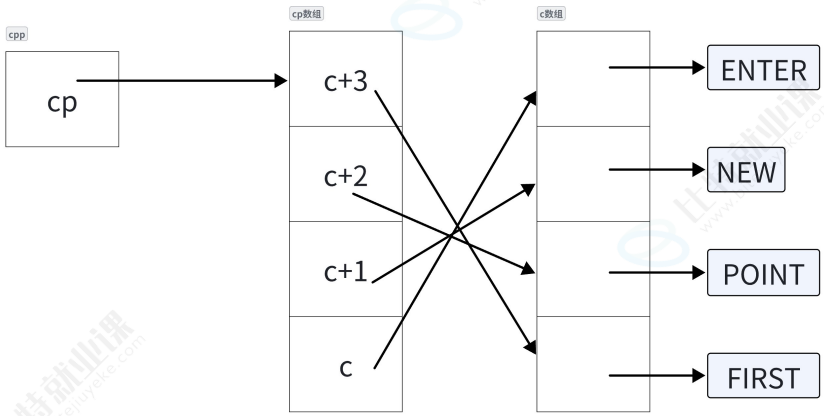
```c
#include <stdio.h>
int main()
{
    char *a[] = {"work","at","alibaba"};
    char**pa = a;
    pa++;
    printf("%s\n", *pa);
    return 0;
}
```



3.7 题目7

```c
#include <stdio.h>
int main()
{
    char *c[] = {"ENTER","NEW","POINT","FIRST"};
    char**cp[] = {c+3,c+2,c+1,c};
    char***cpp = cp;
    printf("%s\n", ****cpp);
    printf("%s\n", *--***cpp+3);
    printf("%s\n", *cpp[-2]+3);
```

```
10      printf("%s\n", cpp[-1][-1]+1);
11      return 0;
12  }
```



完