C语言：指针从基础到进阶

数组名的理解

还记得上一篇博客出现了这么一段关于用指针访问数组的代码。

```
int arr[10] = {1,2,3,4,5,6,7,8,9,10};
int *p = &arr[0];
```

这里用&arr[0]拿到了数组首元素的地址，但其实数组名在大部分情况下本来就是首元素的地址

```
#include <stdio.h>
int main()
{
    int arr[10] = { 1,2,3,4,5,6,7,8,9,10 };
    printf("&arr[0] = %p\n", &arr[0]);
    printf("arr = %p\n", arr);        //这两个printf的输出是一样的
    return 0;
}
```

关于详细情况可以在我的另一篇博客了解：[C语言中关于数组名什么情况为首元素地址，什么情况为整个数组的地址的问题。-CSDN博客但是&arr和&arr+1相差40个字节，这就是因为&arr是数组的地址，+1操作是跳过整个数组的。这里我们发现&arr0]和&arr[0]+1相差4个字节，arr和arr+1相差4个字节，是因为&arr[0]和arr都是。• &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（整个数组的地址和数组首元素的地址是有区别的）• sizeof(数组名)，sizeof中单独放数组名，这里的数组名表示整个数组，计算的是整个数组的大小，除此之外，任何地方使用数组名，数组名都表示首元素的地址。
https://blog.csdn.net/2301_80194476/article/details/136611525?spm=1001.2014.3001.5501

使用指针访问数组

有了前面的基础，我们就可以很方便的用指针访问数组了

```
#include <stdio.h>
int main()
{
    int arr[10] = {0};
    //输入
    i = 0;
```

```
    int sz = sizeof(arr)/sizeof(arr[0]);
    //输入
    int* p = arr;
    for(i=0; i<sz; i++)
    {
        scanf("%d", p+i);
        //scanf("%d", arr+i);//也可以这样写
    }
    //输出
    for(i=0; i<sz; i++)
    {
        printf("%d ", *(p+i));
    }
    return 0;
}
```

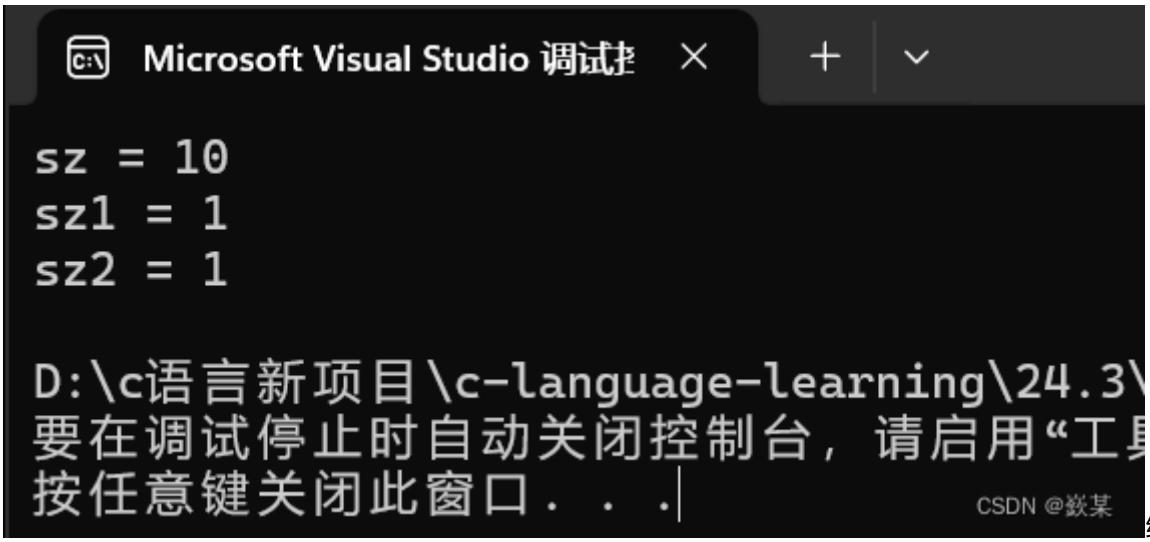
在上面的代码的中将p+i换成p[i]也能正常打印本质上p[i]和*(p+i)是一样的，同理arr[i]等价于*(arr+i)。

一维数组传参的本质

这里直接给答案：数组名是数组首元素的地址；那么在数组传参的时候，传递的是数组名，也就是说一维数组传参的本质就是传递数组首元素的地址。

```
include <stdio.h>
void test1(int arr[])//参数写成数组形式，本质也是指针
{
    int sz1 = sizeof(arr)/sizeof(arr[0]);
    printf("sz1 = %d\n", sz1);//结果为1
}
void test2(int* arr)//参数写成指针形式
{
    int sz2 = sizeof(arr)/sizeof(arr[0]);
    printf("sz2 = %d\n", sz2);//计算一个指针变量的大小
}
int main()
{
    int arr[10] = {1,2,3,4,5,6,7,8,9,10};
    int sz = sizeof(arr)/sizeof(arr[0]);
    printf("sz = %d\n", sz);//结果为10
    test1(arr);
    test2(arr);
    return 0;
}
```

输出结果：



编辑

冒泡排序

后面我会出一期专门的排序算法，手撕常见排序算法！这里主要说关于指针的知识。

```c
void bubble_sort(int arr[], int sz)//参数接收数组元素个数
{
    int i = 0;
    for(i=0; i<sz-1; i++)
    {
        int j = 0;
        for(j=0; j<sz-i-1; j++)
        {
            if(arr[j] > arr[j+1])
            {
                int tmp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = tmp;
            }
        }
    }
}
int main()
{
    int arr[] = {3,1,7,5,8,9,0,2,4,6};
    int sz = sizeof(arr)/sizeof(arr[0]);
    bubble_sort(arr, sz);
    for(i=0; i<sz; i++)
    {
        printf("%d ", arr[i]);
    }
    return 0;
}
//方法2 - 优化
void bubble_sort(int arr[], int sz)//参数接收数组元素个数
{
    int i = 0;
```

```
    for(i=0; i<sz-1; i++)
    {
        int flag = 1;//假设这一趟已经有序了
        int j = 0;
        for(j=0; j<sz-i-1; j++)
        {
            if(arr[j] > arr[j+1])
            {
                flag = 0;//发生交换就说明，无序
                int tmp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = tmp;
            }
        }
        if(flag == 1)//这一趟没交换就说明已经有序，后续无序排序了
        break;
    }
}
int main()
{
    int arr[] = {3,1,7,5,8,9,0,2,4,6};
    int sz = sizeof(arr)/sizeof(arr[0]);
    bubble_sort(arr, sz);
    for(i=0; i<sz; i++)
    {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

二级指针

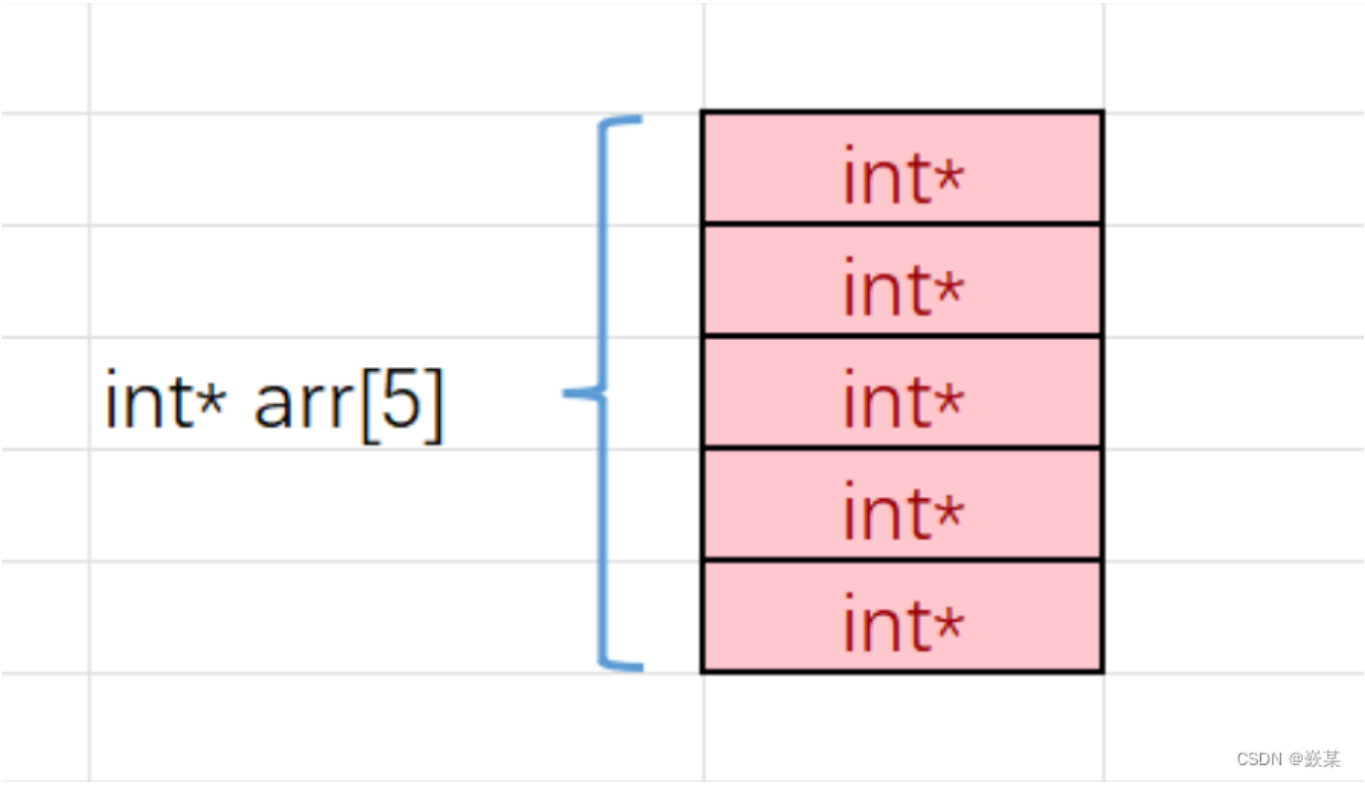
指针变量也是变量啊，那么指针变量也会有地址，指针变量的地址就储存在二级指针变量里面，二级指针变量也有地址，就存在三级指针变量里面.....

```
int a = 10;
int* pa = &a;
printf("%d\n",*pa);
int** ppa = &pa;//将一级指针pa的地址存到二级指针ppa里面
printf("%d\n",**ppa);//二级指针经过两次解引用找到a
```

指针数组

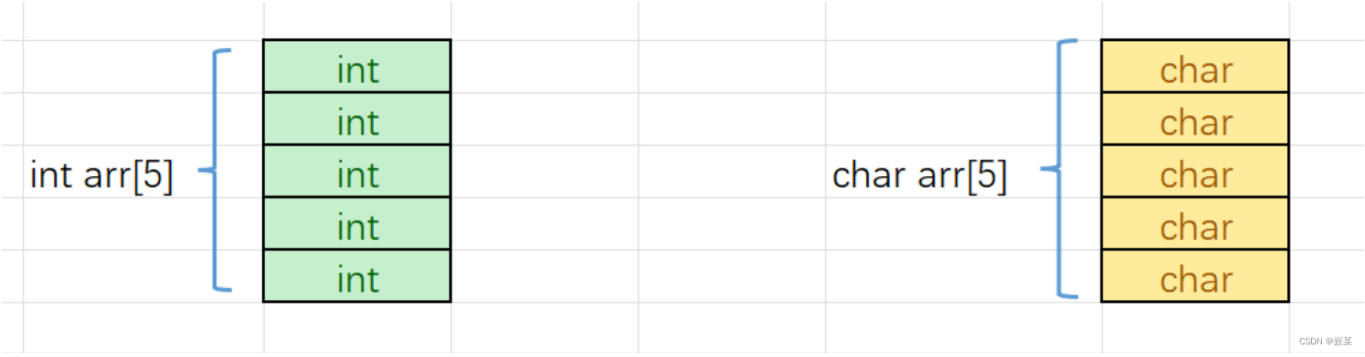
指针数组是指针还是数组？？？

其实指针数组还是数组，只不过这个数组里面放的都是指针。里面的每个元素都是int*类型的。



编辑

我们可以类比一下，整型数组，是存放整型的数组，字符数组就是存放字符的数组。

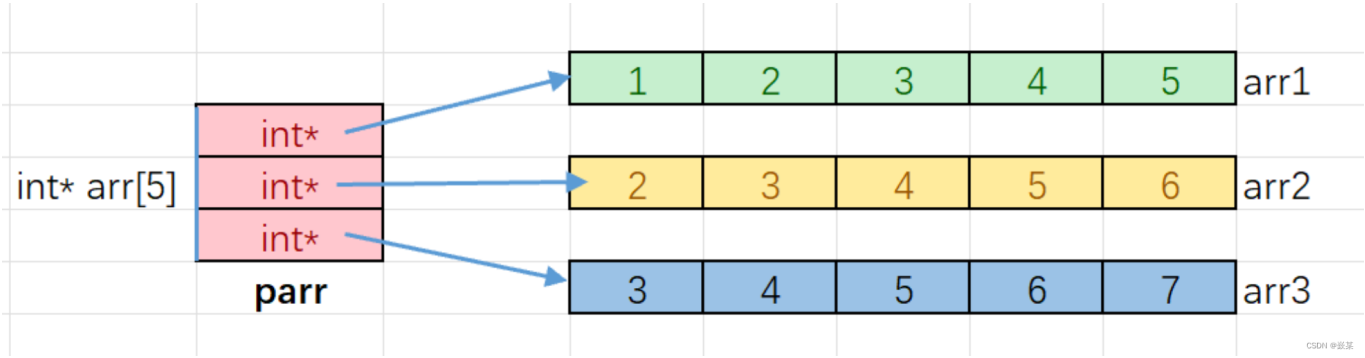


编辑

指针数组模拟二维数组

```
int main()
{
    int arr1[] = {1,2,3,4,5};
    int arr2[] = {2,3,4,5,6};
    int arr3[] = {3,4,5,6,7};
    //数组名是数组首元素的地址，类型是int*的，就可以存放在parr数组中
    int* parr[3] = {arr1, arr2, arr3};
    int i = 0;
    int j = 0;
```

```
    for(i=0; i<3; i++)
    {
        for(j=0; j<5; j++)
        {
            printf("%d ", parr[i][j]);
        }
    printf("\n");
    }
    return 0;
}
```



编辑

parr[i]是访问parr数组的元素，parr[i]找到的数组元素指向了整型一维数组，parr[i][j]就是整型一维数组中的元素。 上述的代码模拟出二维数组的效果，实际上并非完全是二维数组，因为每一行并非是连续的。 **本期博客到这里就结束了，如果有什么错误，欢迎指出，如果对你有帮助，请点个赞，谢谢！**