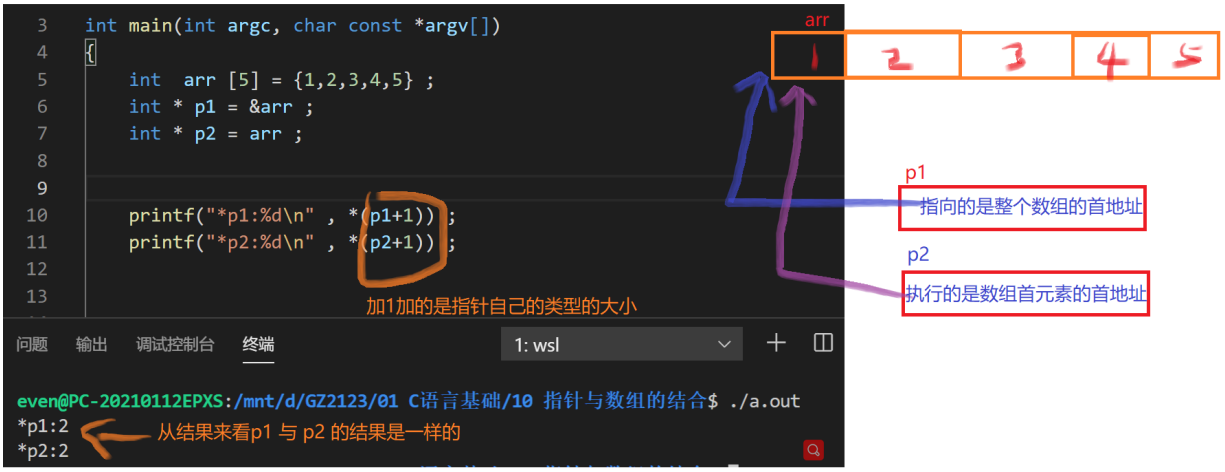
示例1：

```c
int  arr [5] = {1,2,3,4,5} ;
int * p1 = &arr ;
int * p2 = arr ;


printf("*p1:%d\n" , *(p1+1)) ;
printf("*p2:%d\n" , *(p2+1)) ;
```



实例2：

```c
int  arr [5] = {1,2,3,4,5} ;

int * p2 = &arr[2] ;

printf("*p2-1:%d\n" , *(p2-1)) ;
printf("*p2:%d\n" , *p2) ;
printf("*p2+1:%d\n" , *(p2+1)) ;
```

以上代码是通过指针p2 来访问数组中的元素。

一开始定义p2并初始化让指针p2 指向数组中第3个元素的地址。

当我们使用p2 进行指针加减运算的时候， 由于指针是整型的，可以访问到数组中的下一个元素以及上一个元素。

实例3：

数组指针： 专门用来指向一个数组的指针。

```c
int * p ;
int (* p) [5] ; //定义一个 名为p 的指针，
 //并且确定他指向的类型为int [5] ,一个拥有5个元素的整型数组
```

```
4
5  int  arr [5] = {1,2,3,4,5} ;
6
7  int (*p) [5] = arr ;
8
9  printf("arr:%p\n" , &arr );
10
11  printf("%p---%d\n" ,p , (*p)[2] ); // 3 * p ==> arr
12  printf("%p---%d\n" , p+1 , (*(p+1))[2] ); // 已经越界访问
```

注意：

以上代码中 p指向的是 一个整型数组并有5个元素。 因此在对p 进行加减运算时， 是加减一个数组 。

实例4 ：

指针数组： 专门用来存放指针的数组，称为指针数组。

```
1  int a = 100 ;
2  int b = 250 ;
3  int c = 550 ;
4  int d = 256 ;
5  int e = 998 ;
6
7  int  * p [5] = {&a, &b , &c , &d , &e} ; // 定义一个名字为 p的数组， 并且确定该数组中用来存放int * 整型地址
8
9
10  for (size_t i = 0; i < 5; i++)
11  {
12      printf("*p[%d]:%d \n" , i , *(p[i])  ); // p[0]  --》 &a
13  }
14
15
```

操作：

把以上几个实例自己动手写一下理解一下