指针的进阶

1.字符指针

> 指针的大小是固定的4/8个字节（32位/64位）

```c
int main()
{
 char ch = 'w';
 char* pc = &ch;
 return 0;
}

int main()
{
 char arr[] = "asdfgh";
 char* pc = arr;
 printf("%s", arr);
 printf("%s", pc);
 return 0;
}

int main()
{
 //此时可以给char* p前加const
 char* p = "aaasdsa";
 //此时不能修改*p 比如*p = 'w'
 //segmentation fault - 段错误 内存访问错误
 printf("%s",*p); //p中存放的是首字符的地址 *p表示a
 printf("%s", p); //p表示整个常量字符串
 return 0;
}
```

面试题：

```c
int main()
{
 char arr1[] = "abcdef";
 char arr2[] = "abcdef";
 char* p1 = "abcdef";
 char* p2 = "abcdef";
 if(arr1 == arr2)
 {
```

```
 printf("hehe");
 }else{
 printf("haha");
 }//此时输出haha

 if(p1 == p2)
 {
 printf("hehe");
 }else{
 printf("haha");
 }//此时输出hehe
 return 0;
}
```

2.数组指针

是一种指向数组的指针 存放数组的地址
int (* 变量名)[10] = &arr;

```
int arr[10]={0};
//arr 首元素地址
//&arr[0] 首元素地址
//&arr 数组地址
int (*p)[10] = &arr;
//p就是数组指针
```

面试题：

```
int main()
{
 char* arr[5];
 char* (*pa)[5] = &arr; //注意此时pa类型的设置为关键
 //[5]：表示pa指向的数组是5个元素
 //*说明pa是指针
 //char* 表示pa指向的数组的元素类型
 return 0;
}

int arr[5] //arr是一个整型数组
int* par1[10] //是一个数组 有是个元素 每个元素的类型是int*
int (*parr2)[10] //par2是一个指针指向一个数组 每个元素的类型是int par2是数组指针
int (*parr3[10])[5] //parr3是一个数组 该数组有10个元素 每个元素是一个数组指针 该数组指针指向的数组有5个元素 每个元素int

```

3.指针数组

指针数组是数组 用来存放指针

```
int main()
{
int a[5] = {1,1,1,1,1};
int b[5 ] = {2,2,2,2,2};
int* par[2] = {a, b}; //存放整型指针的数组
int i=0;
for(i=0;i<2;i++)
{
int j = 0;
 for(j=0;j<5;j++)
 {
 printf("%d\n", *(par[i]+j));
 }
 }
 char* pch[4];
 return 0;
 }
```

4.数组传参和指针传参

1.一维数组传参

```
void test(int arr[]);

void test(int arr[]);

void test(int arr[]);

void test2(int *arr);

void test2(int *arr[20]);

 void test2(int **arr);

int main()
{
 int arr[10] = {0};
 int *arr2[20] = {0};
 test(arr);
 test2(arr2);
}
```

2.二维数组传参

```
void test(int arr[3][5])
{}
void test(int arr[][5])
{}
void test(int arr[3][]) //错误
{}
void test(int arr[][]) //错误
{}
void test(int* arr);//error
 {}
 void test(int (*arr)[5]);
 {}

 int main()
 {
  int arr[3][5] = {0};
  test(arr);
  return 0;
 }


 void test(int **p)
 {}

 int main()
 {
  int *ptr;
  test(&ptr);
  return 0;
 }
```

5.函数指针

指向函数的指针 存放函数的地址

```
int add(int x, int y)
{
 return x+y;
}

int main()
{
```

```c
int a = 10;
int b = 10;
 add(a, b);
 printf("%p",&add);
 printf("%p",add); //&add和add都是函数的地址

 int (*pa)(int, int) = add;
 printf("%d", (*pa)(2, 3));


 return 0;
}

(*(void (*)())0)()
//void(*)() 函数指针类型
//把0强制类型转换成函数指针类型 也就是把0当成一个函数的地址


void (*signal(int, void(*)(int)))(int);
//signal是一个函数声明 参数有两个 第一个是int 第二个是函数指针
//该函数指针指向的函数参数是int 返回类型是void
//signal的返回类型也是函数指针 该函数指针指向的函数的参数是int 返回类型是void
//返回类型 函数指针void (* )(int)
//用一下代码可以代替
tyoedef void(* pfun_t)(int); //此时pfun_t表示函数指针类型 void(*) (int)
pfun_t signal(int, pfun_t);


```