1.指针是什么?

指针是一个对象，利用地址，它的值直接指向存在的电脑内存中的另一个地方的值，由于通过地址能找到所需要的变量单元，可以说，地址指向该变量单元。因此。将地址形象化的成为“指针”。通过它能找到以它为地址的内存单元。

- *int* p = &a; //p是一个**存放地址**的指针**变量** int *类型*
- *一个内存单元多大? 一个字节*
- *32bit 一个指针变量大小4字节*

sizeof(char)=sizeof(short*)=sizeof(int*)=sizeof(double*)=4*

- *64bit 一个指针变量大小8字节*

2.指针和指针类型

- *int* pa =&a;*

char pc = &a;*

pa 和 pc输出的结果一样

但是:

- *int a = ox11223344;*

int pa =&a;*

**pa = 0; //之后pa变成0x00000000 也就是改变四个字节*

char pc = &a;*

**pc = 0; //之后pc变成0x11223300 也就是只能改变一个字节*

指针类型决定了指针进行解引用操作的时候，能够访问空间的大小:

int* p：能够访问四个字节

char*p：能够访问一个字节

double*p：能够访问八个字节

.....

1. 指针+-整数 （指针类型决定了指针的步长）-- 解引用

```c
int main()
{
    int a = 0x11223344;
    int* pa = &a;
    char* pc = &a;
    printf("%p\n", pa);
    printf("%p\n", pa+1);

    printf("%p\n", pc);
    printf("%p\n", pc+1);

    return 0;
}
```

```
C:\WINDOWS\system32\cmd.exe
0095FB58
0095FB5C
0095FB58
0095FB59
请按任意键继续. . .
```

int指针 + 4 字节

char指针 + 1 字节

3.野指针

指针指向的位置是未知的 （随机的、不正确的）

原因：

a. 指针变量未初始化

b.指针越界访问

c.指针指向的空间释放

```c
int* test()
{
    int a = 10;
    return &a;
}
int main()
{
    int *p = test();
    *p = 20;

    return 0;
}
```

如何避免野指针?

a. 指针初始化

b.小心指针越界

c.指针指向空间释放即时赋值NULL

d.指针使用前检查有效性

指针相减：指针指向的是同一个数组空间

```
int main()
{
    char ch[5] = {0};
    int arr[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
    printf("%d\n", &arr[0] - &arr[9]);
    printf("%d\n", &arr[9] - &ch[0]);

    return 0;
}
```

4.指针和数组

1.求字符串长度：

```
int my_strlen(char* str)
{
 char* start = str;
 char* end = str;
 while(*end != '\0')
 {
 end++;
 }
 return end - start;
}
```

2 .指针的关系运算

```
#define N_VALUES 5
float values[N_VALUES];
float *vp;
for(vp = &values[N_VALUES]; vp > &values[0])
{
 *--vp = 0;
}

```

标准规定：

> 允许指向数组元素的指针与指向数组最后一个元素的那个内存位置的指针比较，但是不允许与指向第一个元素之间的那个内存位置的指针进行比较
>
> 数组名----首元素的地址
> &数组名 -- 数组名表示整个数组 取出的是整个数组的地址
> sizeof（数组名） -- 数组名表示整个数组 计算的是整个数组的大小

5.二级指针

```
int main()
{
```

```
 int a = 10;
 int* pa = &a;
 int** ppa = &pa; //指向指针的地址 指向int*类型
 return 0;
}
```

6.指针数组---（本质是数组--存放指针的数组）

```
int main()
{
 int a = 1;
 int b = 1;
 int c = 1;
 int* pa = &a;
 int* pb = &b;
 int* pc = &c;

 //指针数组
 int* arr2[3] = {&a, &b, &c};
 for (int i=0; i<3; i++)
 {
 printf("%d", *(arr2[i]));
 }
 return 0;
}
```