# 指针技巧之取值

在学习C语言时，指针可谓是一个较难理解的地方，为了让同学们对指针有更进一步的理解，这里传授各位同学关于通过指针取值的技巧

1. 先思考以下的指针变量，取值后会是多少

char temp[20]={11,22,33,44,55,66,77,88,99};

char \*p1 = temp;

printf("\*p1=%d\n", \*p1);// 想想看会是多少呢？

对于上例题中的结果想必很多同学知道答案，为11（你计算对了么）

咱们简单的分析一下：因为p1是一个指针变量，即可以存储指针（即地址），而程序中把temp赋值给了p1，又因为temp为char类型数组的名字，即是下标为0的元素的地址，所以p1这个指针变量存储的是temp[0]这个元素的地址，\*p1得到的结果为11，就不足为奇了

2. 接着上面的代码，再添加如下代码，

printf("\*(p1+1)=%d\n", \*(p1+1));

此时程序会输出多少呢？

答案为22

同样接下来咱们分析一下：因为p1保存了temp[0]这个元素的地址，即p1指向了temp[0]，而p1+1的结果可以理解为temp[1]的地址，所以\*(p1+1)就是取的temp[1]的值，即22

3. （预告：接下来的是重头戏）

接着上面的代码，再添加如下代码：

printf("\*(char \*)((short\*)p1+1)=%d\n", \*(char \*) ((short \*)p1+1));

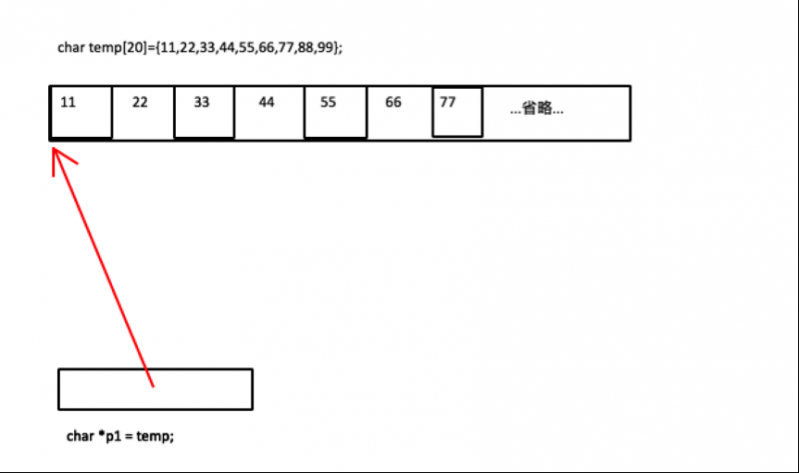
此时程序会输出多少呢？

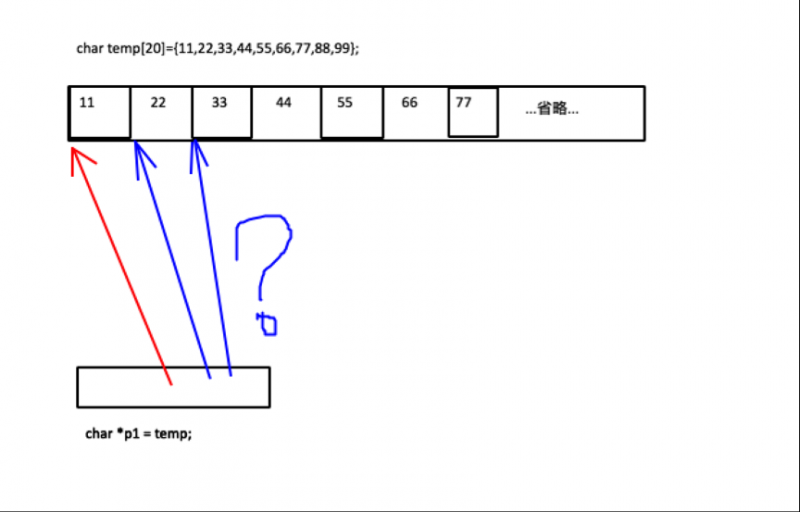
答案为33

啊？？？33？？？你是不是也有这样的感觉，怎么计算的呢？

下面咱们仔细分析一下这个值为什么会是33

<1> p1 在定义的时候已经进行了赋值，即初始化为temp[0]元素的地址



<2>根据优先级以及结合性，对应表达式\*(char\*) ((short \*)p1+1)的计算先后顺为：先对p1进行临时性的强制类型转换，即此时p1这个指针变量的类型已经变成了short\*类型；但是要注意此时p1保存的还是temp[0]元素的地址；接下来要对p1进行+1了，此时到底指向了哪里呢？ 

告诉大家一个技巧：对一个指针变量+1的时候，指针到底向后偏移多少个字节，要根据指针变量当前的类型而定，如果是char\*的那么+1会向后偏移1个字节，如果是short\*那么+1会向后偏移2个字节，如果是int\*那么就会向后偏移4个字节（具体的要根据不同的编译器而定）；

根据刚刚告诉大家的小技巧，试试看(short \*)p1+1会指向哪个元素呢？

对了，此时p1指向了temp[2]这个元素，注意，因为此时p1的类型已经转换为short\*，所以如果此时取值会根据p1指向的地址连续取2个字节的数据当做结果 此时肯定不是33，，，有的同学已经注意到了在最外边还有一个(char\*)即类型转换，此时又把p1的类型转换为char\*了，然后再取值 就会得到33