从一个案例出发

课后收到这样一个问题，有同学针对第二道作业题（逆向输出一个数字）写了如下的代码，输入 1234 时是正确的 4321，但输入 123 的时候结果会输出 320，百思不得其解。

```
#include<stdio.h>
#include<math.h>
void main()
{
    int i,x,y,z,n;//x 是输入的数字，y 输出，n 位数
    y=0,n=0;
    printf("Please input an int:");
    scanf("%d",&x);
    z=x;
    while(x>0)
    {
        x=x/10;
        n+=1;
    }
    int *p=(int *)malloc(n*sizeof(int));
    for(i=0;i<n;i++)
    {
        p[i]=z%10;
        z=z/10;
        printf("%d\n",p[i]);
    }
    for(i=0;i<n;i++)
        y+=p[i]*pow(10,n-i-1);
    printf("位数是%d\n",n);
    printf("该数倒过来为%d\n",y);
    free(p);
}
```

```
Please input an int:1234
4
3
2
1
位数是4
该数倒过来为4321
```

```
Please input an int:123
3
2
1
位数是3
该数倒过来为320
```

仔细检查每一行代码后，似乎都没有问题，那么问题到底出在哪里？当代码执行出问题，但又无法通过阅读代码找到错误时，有几种办法，一种是通过大量的 printf 语句将各个变量的每一次变化过程打印出来，但这种方法需要改变代码本身，需要大量的输入新代码，同时在找出原因后恢复代码的时候会比较麻烦。第

二种办法是 debug，使用 IDE 自带的 debug 功能，可以设置断点，添加监控列表，从而可以看到程序一步一步执行的过程中变量的变化情况。

以 codeblock 为例，需要 debug 之前先要建立 project，将待 debug 的源代码导入到 project 中，然后设置断点，建立监视窗口，输入监控变量，开始 debug。通过 debug 发现，红色代码行在第一次运行时，p[i]*pow(10,n-i-1)的结果是 300.0000，但 y 的值却为 299，导致后面加上十位 20 和个位 1 之后结果为 320 而不是 321。这就说明问题就出在这行代码上。

经过查询 pow 函数的函数格式获知，pow 函数输入参数为 double 型，返回值也是 double 型，而变量 y 的申明类型为 int 型。当我们把一个 double 型的值赋值给 int 型时，C 语言会进行类型转换，而浮点型转换为整型时，规定为截断，即抛弃小数点后面的值，而不是四舍五入。这样就可以解释程序的问题所在了。当运行 3*pow(10,2)的时候，得到的 300 其实不是精确的 300，既可能是 299.9999999999 也可能是 300.00000001 这样的浮点数，这样当进行类型转换时，如果直接抛弃小数点后面部分，则会出现变为 299 的情况，这样就导致了程序出现的问题。

解决方法：
1. 在涉及整型数值运算时，尽量不要使用 pow 这样的浮点型运算方法，而直接在每次循环的时候进行*10，这样可以保证数值的精度。
2. 在使用 pow 后，可以人为加一个 round 函数后再取整，进行四舍五入取整，避免直接抛弃小数部分。
3. 如果要直接将 pow 的值赋给一个整型变量，可以尝试加一个小的正数，如 y+=p[i]* pow(10,n-i-1)+0.00001。

本案例由朱文涛同学提供。