关于快速读入

一般来说快速读入就是指用getchar()函数和字符的处理，
来代替cin和scanf()函数对整型数字的读入。

基于这一原理，再来解释快速读入的过程。

由于一个整型的数是用一个正负号加一串数字字符组成，
因此在读入的时候可以通过判断来看这个字符是否是一个数字里的。
而多个数字之间由除负号以外的其它字符隔开。
单个数字可以用 已经读入的数 $*$ 10 $+$ 该数字 来表示

快速读入函数read()模板

```
int read(){
    int x = 0,f = 1;
    char ch = getchar();

    while(ch <'0' || ch > '9'){
        if(ch == '-') f = -1;
        ch = getchar();
    }
    while(ch >= '0' && ch <= '9'){
        x = (x << 3) + (x << 1) + (ch ^ 48);
        ch = getchar();
    }
    return x*f;
}
```

结合着模板，我们来一步一步看，首先是初始化一些变量
我们将 $x * f$ 作为我们最终要读取的数字，

- f 在这里代表符号，如果 $f = 1$ 则为正，$f = -1$ 则为负；
- 那么 x 在这里就表示要读入的数的绝对值，因此 $x >= 0$ 恒成立
- ch 在这里就是循环中要处理的主要变量啦。

先看第一个 `while(ch <'0' || ch > '9'){}`

由于ASCII表中0-9的编码是连续的，
所以 `ch < '0' || ch > '9'` 在这里就表示不是数字的所有字符
这些字符是需要跳过的，也就是说让ch重新读一个字符就好
也就是 `ch = getchar()` 。

但是有个特殊情况——**负号**
所以需要特殊判断一下当读到负号的时候将 f 改成 -1

于是有代码如下

```
    while(ch <'0' || ch > '9'){  //读到的字符不是数字
        if(ch == '-') f = -1;    //如果读到了负号就把f改成-1
        ch = getchar();          //然后继续读入
    }
```

再看第二个 `while(ch >= '0' && ch <= '9'){}`

看完第一个 `while()` 的条件，再观察这个 `while()` 的条件，
不难发现这个条件就是指【当字符为数字时】
所以这个时候我们需要把 已经读入的数 $*10$ 然后加上这个字符对应的数字
在模板中我用 `x = (x << 3) + (x << 1) + (ch ^ 48)` 来实现以上效果

在这里 `<<` 代表向左移位
对于正整数用这一操作相当于给原本的数乘2的n次方
比如 `a << 3` 就表示 $a*2^3$

所以这个位置 `(x << 3) + (x << 1)` 就表示 $x*2^3+x*2^1$ 也就是 $x*10$（因为位运算更快）

同样的 `^` 这个符号在C++中表示的是异或运算
而48在这对应的是ASCII表中 `'0'` 对应的十进制的值
用字符 `'0'` 到 `'9'` 去减48就得到了数字0-9
但是，异或运算在这里怎么会充当减法的作用呢?
我们先算出48对应的二进制码,也就是

十进制	二进制
48	00110000

这个二进制就比较有意思了，它的后四位都是0，只有前两位是1，
这个时候如果读入一个比48稍大的数，
这两个数异或运算会怎么样?

十进制	二进制
48	0011 0000
49	0011 0001
异或48	0000 0001

十进制	二进制
48	0011 0000
50	0011 0010
异或48	0000 0010

十进制	二进制
48	0011 0000
51	0011 0011
异或48	0000 0011

不难看出，从 0011 0000 (48) 到 0011 1111 (63) 进行异或运算起到的都是 $x - 48$ 的效果
而ASCII表上`'0'`到`'9'`的编码都在这个范围内
，所以这里用`ch ^ 48`来实现减48的效果（因为位运算更快）
这样就实现了把这一个数加入到 x 中的操作

处理完这一个字符，接下来就需要去处理其它是数字的字符，所以再读一个字符，继续这个操作

当读到第一个不是数字的字符时，`while`结束。

然后返回 $x * f$ 的值，快读结束