监听键盘可以使用C语言的字符输入函数，例如 getchar、getch、getche 等， 下面是使用getche函数监听键盘的例子：

1. #include <stdio.h>
2. #include <conio.h>
3. int main(){
4. char ch;
5. int i = 0;
6. //循环监听，直到按Esc键退出
7. **while**(ch = getch()){
8. **if**(ch == 27){
9. **break**;
10. }**else**{
11. printf("Number: %d\n", ++i);
12. }
13. }
14. **return** 0;
15. }

运行结果：  
Number: 1  //按下任意键  
Number: 2  //按下任意键  
Number: 3  
Number: 4  
Number: 5  //按下Esc键  
  
这段代码虽然达到了监听键盘的目的，但是每次都必须按下一个键才能执行getch后面的代码，也就是说，getch后面的代码被阻塞了。  
  
阻塞式键盘监听非常不方便，尤其是在游戏中，往往意味着用户要不停按键游戏才能进行，所以一般采用非阻塞式键盘监听。  
  
使用conio.h头文件中的kbhit函数可以实现非阻塞式键盘监听。  
  
我们每按下一个键，都会将对应的字符放到键盘缓冲区，kbhit 函数会检测缓冲区中是否有字符，如果有字符返回非0值，没有返回0。但是kbhit不会读取字符，字符仍然留在缓冲区。请看下面的例子：

1. #include <stdio.h>
2. #include <windows.h>
3. #include <conio.h>
4. int main(){
5. char ch;
6. int i = 0;
7. //循环监听，直到按Esc键退出
8. **while**(1){
9. **if**(kbhit()){
10. ch = getch();
11. **if**(ch == 27){
12. **break**;
13. }
14. }
15. printf("Number: %d\n", ++i);
16. Sleep(1000); //暂停1秒
17. }
18. **return** 0;
19. }

运行结果：  
Number: 1  
Number: 2  
Number: 3  
Number: 4  
Number: 5  //按下Esc键  
  
每次循环，kbhit 会检测用户是否按下某个键（也就是检测缓冲区中是否有字符），没有的话继续执行后面的语句，有的话就通过 getch 读取，并判断是否是 Esc，是的话就退出循环，否则继续循环。  
  
kbhit 之所以能够实现非阻塞式监听是因为它只检测字符，而不要求输入字符。  
  
Sleep 是“睡眠”的意思，用来让程序暂停执行一段时间，以毫秒记。