In the code you provided, there is only one getchar() function call. However, after the scanf() function calls, you need to add an extra getchar() function call to consume the newline character left in the input buffer. This is because when you enter a number followed by the Enter key, the scanf() function reads the number but leaves the newline character in the input buffer. The next time you call getchar(), it reads this newline character instead of the operator character you intended to read.

By calling getchar() twice, you consume the newline character left in the input buffer and ensure that the next getchar() call reads the correct character.

You are absolutely correct! When using `scanf()` to read input, it often leaves a newline character (`\n`) in the input buffer, which can be inadvertently read by a subsequent `getchar()` call, leading to unexpected behavior. To avoid this, an extra `getchar()` can be used to consume the newline character left in the buffer by `scanf()`.

Here’s a small example to illustrate this:

```c

#include <stdio.h>

int main() {

int num;

char op;

printf("Enter a number: ");

scanf("%d", &num); // After entering a number and pressing Enter, a newline character is left in the buffer.

getchar(); // This will consume the newline character left in the buffer by the previous scanf.

printf("Enter an operator: ");

op = getchar(); // Now, this getchar will read the operator character as intended.

printf("You entered number: %d and operator: %c\n", num, op);

return 0;

}

```

In this example, after entering a number, the user presses the Enter key, leaving a newline character in the input buffer. The first `getchar()` consumes this newline character, and the second `getchar()` reads the operator character as intended.