A Better Way to Get Input From the Keyboard

(reference: Oualline, S. (1997). Practical C Programming, 3rd ed., O'Rielly Media, Inc.)

```c
/* Example of a robust way to get input from the keyboard */

#include <stdio.h>

int main(void){
/* Declare variables that you want to read in */
/* Your list of variables will probably be different! */
char char1;
int var1;
float var2;
char string1[20]; /* an array to hold 19 characters */

char line[100]; /* for the line of user input */

/* put a user prompt here */

/* get and process the user input */

fgets(line, sizeof(line), stdin);
sscanf(line, "%s %c %f %d", string1, &char1, &var2, &var1);
return 0;
}
```

Explanation

- Declare the variables you want to input
- Declare an array of characters that is long enough to contain what the user will enter:

```c
char line[100];
```

- Get the line of input and process it to store the variables

```c
fgets(line, sizeof(line), stdin);
sscanf(line, "%s %c %f %d", string1, &char1, &var2, &var1);
```

How it works

- fgets() gets a line of input (note: 100 characters maximum including the EOL) from stdin (the keyboard)
- sscanf() (string scanf()) scans the input line string and processes it according to the control string (i.e., the conversion specifications between the double-quotes)
- sscanf() must have *pointers* to the variables (the name of the string variable, string1 is a pointer to the first element of the string, and the ampersands in front of the other variable names makes them pointers to the variables)

  Of course the conversion specifications in the control string must match the list of variables in order for this to work properly.