info:

Here are three nice string methods that work directly on a character array (C string).
Start by adding to the includes below iostream:
#include <string.h>

- **strlen()**
  - Takes a character array and counts the items until '\0'.
  - ***strlen(my_string);***
- **strcpy()**
  - Takes two character arrays and copies each character from the second one into the first one until '\0' is reached. The '\0' is also copied.
  - ***strcpy(my_string, other_string);***
  - ***strcpy(my_char_array, "This is a string that will be copied");***
- **strcmp()**
  - Takes two character arrays and compares each item until it encounters '\0' in either of them.
  - Returns 0 if they are equal, -1 if the first one is "lower", 1 if the second one is.
  - ***if(strcmp(answer, "read") == 0){***

a)
Write a program that writes each word that the user types to the terminal into a text file. Don't worry too much about the format. Add to the program functionality to read the text back from the file, word for word, and print it to the terminal.
Here you should use the << and >> operators on the file stream exactly as when reading and writing on the terminal.
Experiment with writing values of other types; int, char, double, float, bool.

b)
Write a program that writes into a binary file and reads from it. Use the write() and read() operations when working with binary files rather than the << and >> stream operators.
Allow the user to choose whether the program will write new values to the file or read from it.
The file should include information on how many strings are in the file. Each string should have information on how long it is, followed by the string characters themselves.

info:

Programs can take arguments off the terminal when they are run.
An example is the g++ program:

- **g++ program.cpp**
    - The string “program.cpp” is an argument
- **g++ -o executable program.cpp**
    - There are three arguments:
    - “-o”
    - “executable”
    - “program.cpp”

The arguments are stored in an argument list, where the first argument (index 0) is the name of the program itself and the other arguments come after it in order.
The classic name for the argument list is ***argv*** and the classic name for the argument count is ***argc*** (but really you can choose the names of the variables). You can change the definition of your main function:

- **int main(int argc, char *argv[])**
    - ***char *argv[]*** simply means a double array of characters, or an array of strings
        - a single array of characters is a string, remember?
        - You can imagine it is char ***argv[][]***, but that syntax will not work in this particular context.
    - **g++ -o executable program.cpp**
        - argc == 4
        - argv[0] is “g++”
        - argv[1] is “-o”
        - argv[2] is “executable”
        - argv[3] is “program.cpp”

You can use argc to make sure you don’t try to read outside the array of strings. Remember to use ***strcmp()*** to check what the values of the arguments are.
You can also ***#include <sdtlib.h>*** and use the functions ***atoi()*** and ***atof()*** to change strings into integers or floating points (double) respectively.

c)
Write a program that prints a name (string), a number (int) and a height (double) to the terminal. It uses default values if no argument is given for each value. The first argument is the name, the second argument the number and the third argument is the height. If only one or two arguments are given the default values are used for the remaining variables.

- ./arguments.exe John 23 1.79
    - output:
      Name: John
      Number: 23
      Height: 1.79 m