Bugs

1. The newlines are not being registered:

   As the program runs the conditions that check the characters are correctly recongizing that it has encounter a new line character and the correct position of it but it fails to save more than the first encountered line. This is because when iterating is checking for null terminator instead of jump character, only saving the first encounter line the manually inputted.

Solution

1. Logs were set to determine which characters were and if they are passed. The only needed change was changing the evaluated character to jump line inside the while (str[i] != '\0' && isspace(str[i]))

```
while (str[i] != '\0' && isspace(str[i]))
{
        l4cPrintf(&test, L4C_INFO, "Character evaluated in iteration in %d '%c'", i, str[i]);
        if (str[i] == '\n')
        {
                result.lineStarts[result.numLines++] = i + 1;
        }
        i++;
}
```

Debugging Tecniques

1. Code Analisis and Slashing

   This was the most important techniques in this workshop since we already knew the basic functunlity of this program it was easier to determine in which parts of the code the error occurred and correspondingly set test and searches around the delimited area.

2. Log Files

   This technique allowed us to look for specific situations across the iteration of the program including errors and warnings allowing us to know the reasons why a program ends following the pre-stablished log messages that showed the behavior of the code step by step. Thanks to this we could determined that the characters were been readen correctly and that the iteration was working, leaving the only problem being the enter condition to the increasing of lines.

Reflextion

➔ The log files help me get a better understanding of functioning of the code making it easier to understand the iteration. Unlike the normal debugging tool the log file allowed me to have a detailed description of the running of the code from end to finish regardless of the analized variables. Mainly this is due to the personalized log message sthat are inserted manually to the code, this allows to obtain as much information as needed ïn the desired format.

- ➔ Based on performance and speed i prefer using the debugger tools since they allowed for a quicker run of the code without the need of altering it or adding new dependencies to which could damage the code if they are not used properly. This doesnt mean that logs file are useless in comparison to a debugger, in cases were we have a massive running app with multy threading using log files is better than the debugger since it allows to see the multiple threads and their outputs something that a debugger cannot do, other example is when we are running a program that doesnt stop when an error is encountered like a web running application that runs even if not everything is loaded this would allow to see the specific aspects. In summary, if we are looking a single run process a debugger is the bets option but if we are looking at threading non stop app loggers would be better.
- ➔ I'm confident that we've addressed all the bugs comprehensively. We've achieved this assurance through the diligent use of our logger and assertions tools. These tools provide real-time insights into the code's performance and behavior, allowing us to proactively detect and handle any unforeseen issues. Consequently, I believe we are well-equipped to analyze the code's functionality without apprehension about unexpected situations arising, as we've thoroughly tested and monitored its performance.