

**Programación de estructuras de datos y  
algoritmos fundamentales (Gpo 14)**

**Act 2.3 - Actividad Integral estructura de datos lineales  
(Evidencia Competencia)**

Alumno:

Alejandro Daniel González Carrillo	A01570396
Agustín Blanco Oliver	A00828415

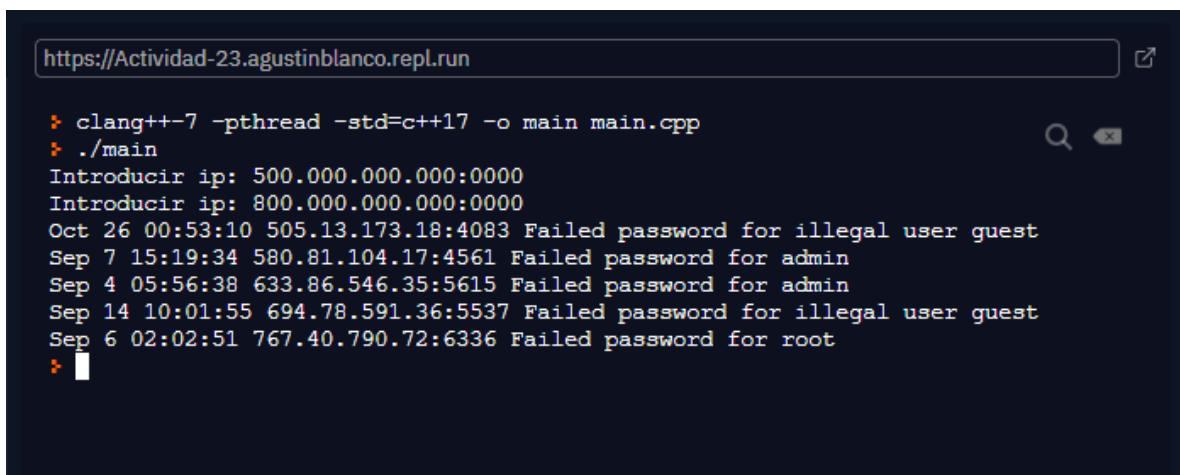
Profesores:

Luis Humberto González Guerra

© 2020 Derechos reservados: Ninguna parte de esta obra puede ser reproducida o transmitida, mediante ningún sistema o método, electrónico o mecánico, sin conocimiento por escrito de los autores.

Monterrey, Nuevo León. 10 de octubre de 2020.

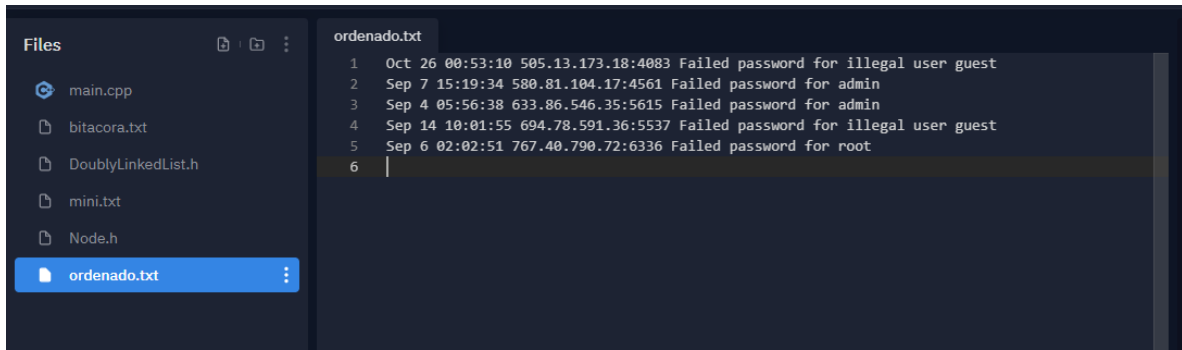
**Reflexión:** Durante el desarrollo de este proyecto, el cual era similar al pasado sin embargo, en este teníamos que utilizar Doubly Linked Lists para el almacenamiento y manejo de datos. Una de las cosas o desventajas que vi al momento de que utilizáramos Double linked list es que es mas lenta al momento de correr. Sin embargo, al utilizar un DLL a comparación de una SLL es que ahora tenemos un pointer prev el cual nos permite regresar a valores anteriores. Esto nos brinda muchas ventajas con la eficiencia del código. Primero que nada podemos atravesar en ambas direcciones la DLL permitiéndonos movernos libremente dentro de la linked list. Segundo nos podemos evitar muchos problemas al momento de querer ubicar un valor dado por el usuario. Y finalmente tercero se pueden agregar mas nodos antes de la linked list. Durante el desarrollo nomas utilizamos addFirst, addLast, sort, search, destructor y constructor. El addFirst nos permitía agregar un nodo siempre al inicio de la linkedList. El addLast nos permitía agregar al final de la linkedlist un nodo. Para el sort utilizamos bubble sort ya que fue recomendado por el profesor y es una buena opción para lo que tratamos de hacer que es comparar valores. Y para el método search utilizamos una búsqueda secuencial la cual iba en cada nodo y comparaba el valor de la key con los dos parámetros que eran los IPs que el usuario introducía. Para concluir puedo decir que fue difícil ya que nunca habíamos realizado una doubly linked list sin embargo pude desarrollar mas mis conocimientos respecto a los pointers. Pudimos lograr que funcionara para un txt de 30 líneas, la idea era probarlo con el de 16800 sin embargo, el programa tarda mucho.



```
https://Actividad-23.agustinblanco.repl.run

> clang++-7 -pthread -std=c++17 -o main main.cpp
> ./main
Introducir ip: 500.000.000.000:0000
Introducir ip: 800.000.000.000:0000
Oct 26 00:53:10 505.13.173.18:4083 Failed password for illegal user guest
Sep 7 15:19:34 580.81.104.17:4561 Failed password for admin
Sep 4 05:56:38 633.86.546.35:5615 Failed password for admin
Sep 14 10:01:55 694.78.591.36:5537 Failed password for illegal user guest
Sep 6 02:02:51 767.40.790.72:6336 Failed password for root
> 
```

(TEST CASE CON UN TXT DE 30 LINEAS, SI FUNCIONO)



The image shows a file editor interface with a dark theme. On the left, a sidebar titled 'Files' contains a list of files: main.cpp, bitacora.txt, DoublyLinkedList.h, mini.txt, Node.h, and ordenado.txt. The file 'ordenado.txt' is selected and highlighted in blue. The main editor area displays the contents of 'ordenado.txt', which is a list of five lines of log data, each preceded by a line number (1 to 5). The data represents failed password attempts for different users and IP addresses. Line 6 is currently empty, with a cursor at the start.

```
ordenado.txt
1 Oct 26 00:53:10 505.13.173.18:4083 Failed password for illegal user guest
2 Sep 7 15:19:34 580.81.104.17:4561 Failed password for admin
3 Sep 4 05:56:38 633.86.546.35:5615 Failed password for admin
4 Sep 14 10:01:55 694.78.591.36:5537 Failed password for illegal user guest
5 Sep 6 02:02:51 767.40.790.72:6336 Failed password for root
6
```

(REPL.IT SE CREA EL ARCHIVO Ordenado.txt CON LOS DATOS CORRECTOS)