

Sistemas Operativos

[Práctica 1 – PROGRAMACIÓN C]

NOMBRE/S AUTOR/ES

Tabla de contenido

[Autores 2](#_Toc115812632)

[Descripción del Código 3](#_Toc115812633)

[Diseño del Código 3](#_Toc115812634)

[Principales Funciones 3](#_Toc115812635)

[Casos de Prueba 3](#_Toc115812636)

[Comentarios Personales 4](#_Toc115812637)

# Autores

**Ángel Covarrubias Roldán**

**Javier Gaspariño Muñoz**

# Descripción del Código

## Diseño del Código

**Estructura del programa test:**

El programa test se puede dividir en 2 secciones:

* Comprobación de argumentos.
* Elección de la función.

Comprobación de argumentos:

Esta parte se encarga de comprobar que los distintos argumentos se pasen de forma correcta. Se tiene en consideración que:

* No se le pase ningún argumento o se le pasen más de 2. El programa en este caso daría un error diciendo el número de argumentos que se le ha pasado y terminando la ejecución de este.
* Se le pase un argumento. En este caso, si dicho argumento se corresponde con una de las distintas funciones dadas pueden ocurrir dos cosas: que no se le pase le valor de n como segundo argumento derivando que N sea 10 (como caso base) o que el programa reciba como segundo argumento el valor de N (se necesita la función atoi() para obtener el valor numérico correspondiente). En caso contrario, el programa mostrará un error pidiendo que el usuario introduzca una de las funciones disponibles.
* Por último, se retornará el valor de líneas, que es una variable local que contiene el futuro valor de N.

Elección de la función:

Tras la comprobación de argumentos, encontramos el código encargado de redireccionar a una u otra función disponible dependiendo de lo que el usuario desee. Encontramos varios if/ else if encargados de comprobar la función que el usuario a pedido que se realice redireccionando a la misma. Si por algún motivo no es ninguna de las funciones aportadas, el programa mandaría un mensaje de error (esto no debería ocurrir en ninguna circunstancia ya que el control de errores en cuanto a un paso erróneo de la función a realizar se realiza anteriormente).

* **Antes de comenzar con la explicación de cada función, se debe aclarar que cada una ella contiene una comprobación de no negativo. Con esto se consigue que en el caso que el usuario introduzca un numero negativo, el programa muestre un mensaje de error advirtiendo de que el valor de N debe ser positivo. Otra consideración a tener en cuenta es cuando el usuario pasa como argumento el valor 0, en esta ocasión el programa no mandará un error, pero sí advertirá de que no se va a imprimir ninguna línea de texto.**

Función Head:

En esta función se encuentra un pequeño bucle que se va a recorrer hasta que:

* Se haya introducido N líneas de texto.
* O el usuario ya no quiera meter por teclado más líneas.

Tras esto el programa imprimirá las N primeras líneas de texto introducidas por la salida estándar. Con esto se consigue de forma bastante simple que se impriman las N primeras líneas de un texto sin necesidad de introducir todo el texto en caso de entrada estándar o tardar el menor tiempo posible en mostrar las N primeras líneas si se le pasa un documento de texto externo al programa.

* **En las dos siguientes funciones se va a reservar espacio de memoria de forma dinámica para el almacenamiento de las N líneas de texto. Esto va a ser posible gracias a una variable local que va a guardarlas y que se va a inicializar al principio del método.**

Función Tail:

Tail se puede dividir en dos claros segmentos:

* Primer segmento: se va a encargar de ir guardando las líneas de texto. Cuando el array este completo se va a ir substituyendo la línea más antigua por la línea nueva. De esta forma se consigue que no se tenga que almacenar toda la información introducida por el usuario haciendo que no se tenga que reservar cantidades insanas de memoria.
* Segundo segmento: se enfoca en mostrar la información contenida en el array. Primero se va a mostrar la línea más antigua por ello necesitamos saber dónde se quedó el índice para poder imprimir a partir de este (índice va a tener más una posición del valor en el que realmente se quedó).

Función Longlines:

Por último, esta función la podemos segmentar en 3 partes:

* Primera segmentación: encargado de introducir la información de la entrada estándar. En un primer momento se meten en el array las N primeras líneas sin comprobar nada sobre su longitud. Tras esto, se va a buscar la línea cuya longitud es menor y se sustituirá en el caso que la nueva línea introducida tenga mayor número de caracteres. Esto ocurrirá hasta que el usuario quiera terminar.
* Segunda segmentación: tras la introducción de información, se ordenará el array mediante el uso del algoritmo de la burbuja.
* Tercera segmentación: para terminar, el programa recorrerá el array mostrando su contenido.

Texto

Descripción generada automáticamente con confianza media

**Pseudocódigo: Algoritmo de la burbuja.**

## Principales Funciones

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **comprobarArgumentos** | **Nombre** | **Tipo** | **Descripción** |
| **Argumentos** | Argumento 1 | **argc** | **int** | **Número de argumentos que se pasan.** |
|  | Argumento 2 | **argv[]** | **char \*** | **Array que guarda los argumentos pasados.** |
| **Variables Locales** | Variable 1 | **lineas** | **int** | **Variable que contiene el número de líneas.** |
| **Valor Devuelto** | **lineas** |  |  | **Futuro valor de N** |
| **Descripción de la Función** | **Se comprueba que se han introducido los argumentos correctamente.** |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **main** | **Nombre** | **Tipo** | **Descripción** |
| **Argumentos** | Argumento 1 | **argc** | **int** | **Número de argumentos que se pasan.** |
|  | Argumento 2 | **argv[]** | **char \*** | **Array que guarda los argumentos pasados.** |
| **Variables Locales** | Variable 1 | **n** | **int** | **Variable que contiene el número de líneas.** |
| **Valor Devuelto** | **0** |  |  | **Si todo va bien devuelve 0 si no devuelve 1.** |
| **Descripción de la Función** | **Función principal, en la cual se decide que función ejecutar dependiendo del argumento que haya recibido.** |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **head** | **Nombre** | **Tipo** | **Descripción** |
| **Argumentos** | Argumento 1 | **N** | **int** | **Número de líneas.** |
| **Variables Locales** | Variable 1 | **count** | **int** | **Variable que representa un contador que guardara el número de líneas introducidas.** |
|  | Variable 2 | **buffer[]** | **char** | **Array que guardara los caracteres del texto introducido.** |
| **Valor Devuelto** | **0** |  |  | **Si todo va bien devuelve 0 si no devuelve 1.** |
| **Descripción de la Función** | **Función que se encarga de mostrar las N primeras líneas.** |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **tail** | **Nombre** | **Tipo** | **Descripción** |
| **Argumentos** | Argumento 1 | **N** | **int** | **Número de líneas.** |
| **Variables Locales** | Variable 1 | **índice** | **int** | **Variable que guardara el índice por el que empezar a imprimir el contenido que nos interesa.** |
|  | Variable 2 | **lineasTail[]** | **char \*** | **Array que guardara las N ultimas líneas introducidas** |
|  | Variable 3 | **buffer[]** | **char** | **Array que guardara los caracteres del texto introducido.** |
| **Valor Devuelto** | **0** |  |  | **Si todo va bien devuelve 0 si no devuelve 1.** |
| **Descripción de la Función** | **Función que se encarga de mostrar las N ultimas líneas.** |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **longlines** | **Nombre** | **Tipo** | **Descripción** |
| **Argumentos** | Argumento 1 | **N** | **int** | **Número de líneas.** |
| **Variables Locales** | Variable 1 | **lineasLongLines[]** | **Char \*** | **Array que guardara las N líneas con mayor longitud introducidas.** |
|  | Variable 2 | **count** | **int** | **Variable que llevará la cuenta de las líneas introducidas hasta llegar al valor de N.** |
|  | Variable 3 | **buffer[]** | **char** | **Array que guardara los caracteres del texto introducido.** |
| **Valor Devuelto** | **0** |  |  | **Si todo va bien devuelve 0 si no devuelve 1.** |
| **Descripción de la Función** | **Función que se encarga de mostrar las N líneas con mayor longitud.** |  |  |  |

## Casos de Prueba

* N es un número negativo: al pasarle como argumento un numero negativo el programa va a realizar la comprobación de argumentos y va a llegar a la inicialización de la función escogida. Sin embargo, en esta se va a comprobar que N sea no negativo. En caso de que lo sea saltara un mensaje de error informando que N no puede ser negativo y acabando con el programa.
* El primer argumento pasado deber ser una de las funciones especificadas. (head, tail, longlines): tras la comprobación de argumentos se comprobará si el programa ejecuta la función deseada. Por ello se ha comprobado que, por ejemplo, si el argumento pasado es “Head” el programa ejecute la función Head y así con las demás.
* Caso en el que N sea igual a 0: si se le pasa como valor 0 al segundo argumento el programa no debería mostrar ninguna línea de texto introducido por la entrada estándar. Por ello y para saber al usuario que no había ningún tipo de error se muestra un pequeño mensaje para advertirle que no se va a imprimir ninguna línea de texto ya que N vale 0.
* Que el primer argumento no sea una de las funciones especificadas: si se le pasa una función que no está contemplada en el programa, este mandará un mensaje de error explicando que el usuario debe introducir una de las funciones implementadas.
* No se le pasa ningún argumento o más de dos argumentos: en el caso en el que no se le pase ningún argumento o se le pase más de 2 argumentos, el programa imprimirá un mensaje de error especificando cuantos argumentos se han pasado al programa y explicando que al menos se debe introducir la función deseada.
* Se le pasa solo la función a realizar: también se ha comprobado que ocurriría si solo se introduce el método a realizar. El programa pondrá 10 como valor de N siendo este el caso base.

# Comentarios Personales

Problemas encontrados:

Durante la realización de esta práctica nos hemos topado con varios problemas siendo los que más destacan los relacionados con cómo funciona C.

* Muchos de los errores que nos daba el programa se debían a el mal entendimiento de cómo funcionan los punteros. En ocasiones operábamos con las direcciones de memoria y no con el contenido del puntero. Pero, sobre todo, lo que más nos costó fue entender la creación de array y el almacenamiento de las líneas de texto o strings en estas.
* También tuvimos diversos errores con los valores que guarda argc y argv[] ya que no sabíamos que aunque pasásemos 2 argumentos, argc valía 3 ya que en realidad se le pasan 3 argumentos (los 2 introducidos por el usuario y la dirección del programa a ejecutar, es decir, del archivo c). Añadir que esta dirección también se guarda en el array de argumentos por lo que muchas veces queríamos acceder al primer argumento introducido por el usuario mediante argv[0] ocasionando diversos problemas.

Puntos a mejorar:

* Mayor limpieza en el código: aunque hemos intentado programar desde un principio de forma limpia sí que es verdad que en ocasiones podía ser un poco caótico. Sin embargo, la homogeneidad del código se ha mantenido en todo momento.
* Mayor creación de funciones (mayor separación de código): se podría haber realizado distintas funciones para dividir el código en partes más claras.
* Menor complejidad en el código: sobre todo en la función longlines, ya que se tiene una complejidad de 2 O(n2). Esta se podría haber reducido se hubiese planteado de distinta forma.

Evaluación del tiempo dedicado:

En cuanto al tiempo dedicado, ambos integrantes hemos dedicado un tiempo suficiente como para realizar la tarea de forma adecuada, sin embargo, se ha de admitir que este tiempo no ha sido el adecuado si se quería perfeccionar esta. El comienzo de la practica fue un poco a ciegas debido a que no se habían repasado los conocimientos de C. Esto ocasionó más tarde al compilar el código que apareciesen varios errores que se podían haber evitado si se hubiese tenido más interiorizado la semántica y cómo funcionan los arrays y punteros en C. A su vez, esto hizo que se tuviese que revisar los apuntes proporcionados y que se investigara por internet haciendo que se perdiese bastante tiempo.