Memoria Práctica 1

Programación en C

Autor: Iván Martín Gómez

Índice de contenidos

1. Librerías

Hemos creado una librería llamada “libreria.h” que contiene la cabecera de tres funciones:

int head (int N)
int tail (int N)
int longlines (int N)

El cuerpo de estas funciones está alojado en el fichero “libreria.c” en el que el la primera línea hacemos #include “libreria.h”, notar que utilizamos comillas y no <> debido a que ambos ficheros se encuentran en la misma carpeta.

Por último, tenemos un fichero “test.c” donde hacemos uso de la librería poniendo al principio del fichero: #include “libreria.h” y #include “libreria.c”.

2. Funciones implementadas

Primero mencionaremos las características comunes que hemos tenido que llevar a cabo para un correcto funcionamiento de las funciones.

- **Reserva de memoria dinámica con malloc**: creamos un puntero doble de tipo char para almacenar líneas de caracteres. El tamaño de la memoria reservada vendrá dado en función de N. Para cada línea reservaremos 1024 bits, mientras que para la estructura donde almacenamos las líneas reservaremos 1024*N bits.
- **Captura de las líneas introducidas por stdin para su almacenamiento en la estructura de datos creada**: para llevar a cabo esta tarea utilizamos la función fgets(). Adicionalmente hemos utilizado la función strcpy() para copiar datos.
- **Imprimir el resultado por pantalla mediante printf()**: una vez realizada la tarea que debe hacer la función se imprime por pantalla el resultado.
- **Liberar memoria dinámica con free()**: antes de salir de cada una de las funciones y volver a la función, cada uno de las funciones libera la memoria dinámica reservada al inicio de la función.
- **Limpiar stdin mediante rewind()**: dado que vamos a utilizar varias veces la entrada estándar la limpiaremos de datos no deseados antes de salir de cada una de las funciones.

a. int head (int N)

Esta función muestra las N primeras líneas que se introduzcan por la entrada estándar. En caso de que el número de líneas introducidas por stdin sea menor que N, entonces se mostraran todas las líneas introducidas por stdin.

Esta función empieza almacenar en la estructura de datos las líneas introducidas por stdin empezando desde la primera posición y dejara de hacerlo cuando la estructura esté completa. Notar que no se corta el flujo de datos entrante a través de stdin aunque ya hayamos completado la estructura, simplemente no se almacenarán.

b. int tail (int N)

Esta función muestra las N últimas líneas que se introduzcan por la entrada estándar. En caso de que el número de líneas introducidas por stdin sea menor que N, entonces se mostraran todas las líneas introducidas por stdin. El sistema que hemos utilizado para almacenar las últimas N líneas.

Esta función empieza almacenar en la estructura de datos las líneas introducidas por stdin empezando desde la primera posición. En este caso almacenamos los datos entrantes a través de stdin hasta completar la estructura de datos, una vez completa, si hay más elementos entrantes procedentes de stdin, realizamos un desplazamiento de todos los datos de la estructura. Sacamos el dato almacenado en la primera posición de la estructura y desplazamos todos los datos restantes a una posición inferior con respecto a donde se encontraban, por último introducimos en la última posición de la estructura al nuevo dato entrante.

c. int longlines (int N)

Esta función muestra las N líneas largas que se introduzcan por la entrada estándar. En caso de que el número de líneas introducidas por stdin sea menor que N, entonces se mostraran todas las líneas introducidas por stdin ordenadas de mayor a menor tamaño.

Esta función empieza almacenar en la estructura de datos las líneas introducidas por stdin empezando desde la primera posición. Una vez completa cada nuevo dato entrante por stdin se compara con el dato de menor tamaño previamente almacenado en la estructura, de esta forma nos aseguramos que en nuestra estructura solo tendremos los datos más grandes.

Una vez que se termina la entrada de datos por stdin, sabemos que en nuestra estructura sólo tenemos los N datos más largos que han entrado por stdin.

El último paso será ordenar los datos de la estructura en orden decreciente. Para ello hemos hecho uso de un algoritmo de ordenación bien conocido denominado método de la “burbuja” que tiene una complejidad de N^2. Para la comparación entre los datos hemos utilizado la función strlen() que devuelve un entero.

d. test.c

Este fichero contiene el método int main (int argc, char *argv[]){} que es el punto de entrada del programa. Será desde esta función desde donde llamaremos a cada una de las tres funciones implementadas. El valor del número de líneas N que queremos utilizar en las diferentes funciones se le pasará como parámetro de entrada de esta función, en el caso de que no reciba ningún parámetro de entrada se le asignará por defecto el valor N=10.

Cada una de las 3 funciones devuelve un entero para saber cómo ha ido la ejecución de la función, en caso de que todo haya ido bien ese número entero tendrá un valor de 0, en caso contrario algo habrá ido mal. A su vez la función main() devuelve un entero con el mismo criterio. Por lo que si alguna de las funciones devuelve un entero distinto de cero supondremos que algo ha ido mal y se abortará le ejecución de la función main() devolviendo el número entero devuelto a su vez por la función que ha provocado el fallo del programa. En caso de que las tres funciones hagan bien su trabajo main() devolverá 0.

3. Comentarios personales

El grado de dificultad de la práctica es alto. Aún teniendo nociones de programación se hace muy difícil programar en C utilizando nano.