

Memoria de la práctica 1

Estructura de computadores

Laura Álvarez Flórez
Laura Yunta García

100363965
100363785

100363965@alumnos.uc3m.es
100363785@alumnos.uc3m.es

Contenido

1. Ejercicio 1 2

 a) Contenidos pedidos..... 2

 b) Conclusiones y problemas encontrados. 5

2. Ejercicio 2 5

 a) Contenidos pedidos..... 5

 b) Conclusiones y problemas encontrados. 8

1. Ejercicio 1

a) Contenidos pedidos.

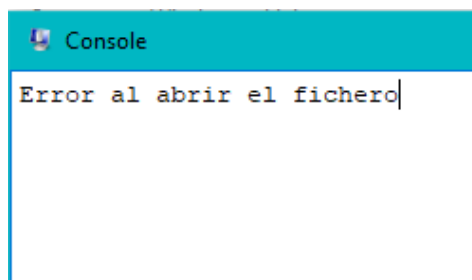
El objetivo del primer ejercicio es desarrollar un programa en ensamblador que procese un fichero de entrada formado por palabras, que podrán estar separadas por uno o más blancos o saltos de línea. El programa deberá indicar el número de palabras del texto que incluyen como subcadena otra dada.

Para ello inicialmente se han declarado una serie de datos globales pedidos por el enunciado y además, dos vectores VFichero y VCadena (que servirán de auxiliares para almacenar los datos proporcionados por el usuario), la variable nBytes (donde se almacenarán el número de bytes a leer) y por último las cadenas auxiliares de caracteres “espacio” y “salto” (que serán de gran ayuda para separar y diferenciar las palabras incluidas en el fichero) y “Error1” y “Error2” (que serán imprimidas en caso de que se produjera algún fallo).

El programa se estructura en una única función llamada funcion1 a la cual se le pasan dos parámetros (la dirección de inicio del fichero y la dirección de inicio de la cadena). Esta función además de los parámetros también utilizará unas variables locales durante su ejecución.

Para comenzar se traslada la cadena impuesta por el usuario a un vector. Este proceso se realiza a partir de la etiqueta Crear_Cadena donde se utiliza el parámetro \$a1 que almacena la dirección de inicio de la cadena y se crea el vector VCadena.

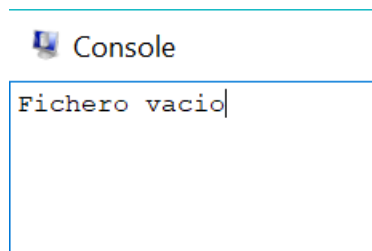
El flujo del programa continúa hacia AperturaFichero donde si existiera algún error al abrir el mismo como por ejemplo, que fuera errónea la dirección, el programa nos conduciría hacia ErrorFichero que imprimiría por pantalla el siguiente mensaje:



```
Console
Error al abrir el fichero|
```

El programa retornaría al flujo principal saliendo así de la función ejercicio1 y finalizará el programa.

En el caso contrario (no habiendo error en el fichero) se procedería a la lectura. Para conseguir una buena eficacia del programa, antes de realizar la lectura completa del fichero leemos únicamente el primer byte de este. Si se detecta que el fichero está vacío el programa nos conduciría de nuevo a la segunda excepción en la cual se muestra el siguiente mensaje por pantalla:



y a continuación retornaría del mismo modo que si la apertura fuese incorrecta.

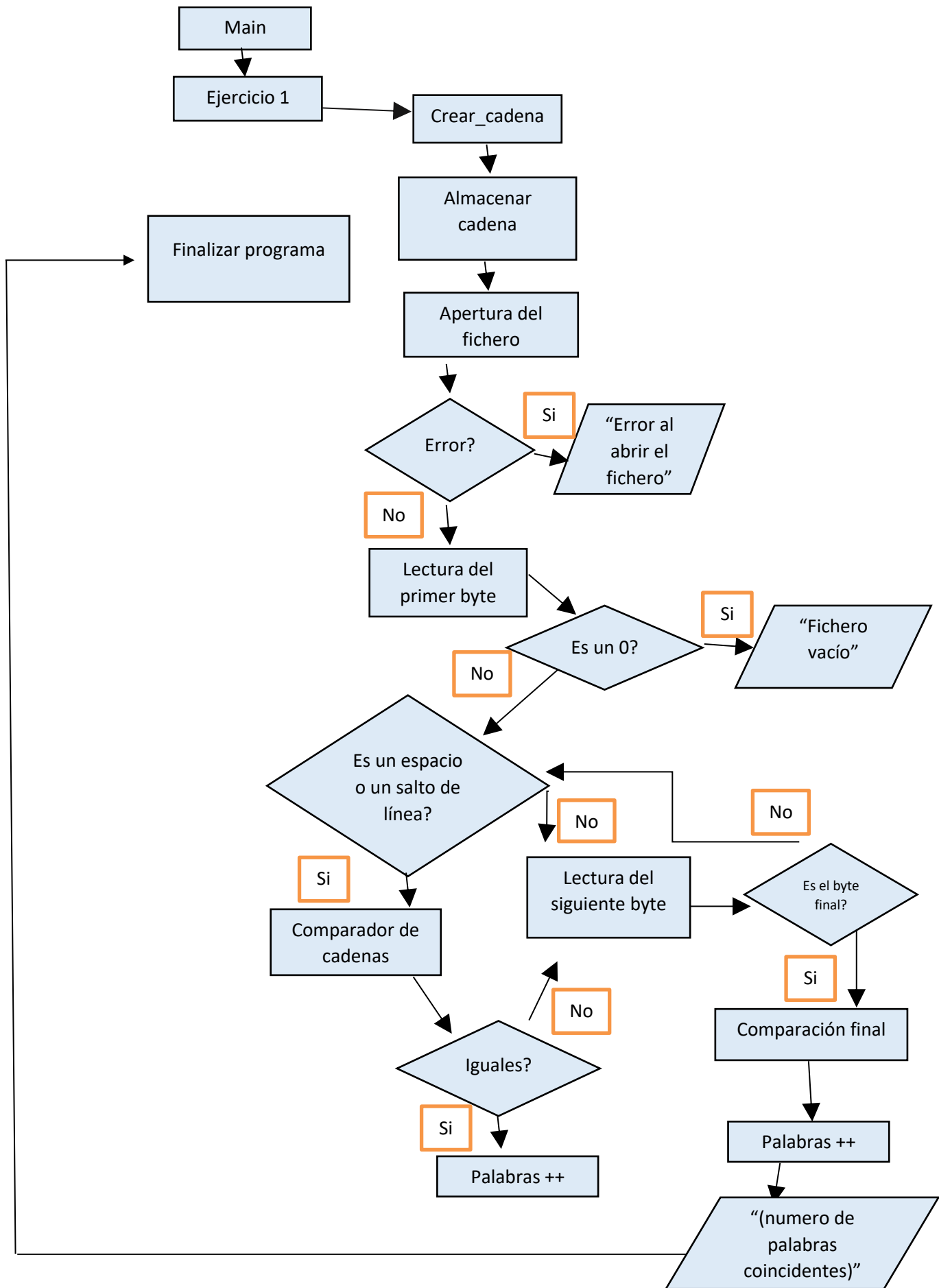
Suponiendo las condiciones ideales para nuestro programa (el fichero no está vacío) se comienza su lectura byte a byte. Esto se realiza utilizando el bucle que hemos denominado Lector el cual comprueba en cada iteración si el byte leído es un espacio o un salto de línea. En caso afirmativo se sobreentiende que la palabra ha finalizado, si no detectase ninguno de estos seguiría leyendo los caracteres de la palabra del fichero.

Una vez acabada la lectura de la palabra, el programa continúa hacia la función Comparador. Esta comparará la palabra leída con la cadena impuesta por el usuario. Para ello utiliza una serie de registros que funcionan como auxiliares almacenando los diferentes desplazamientos realizados por ambos vectores o los caracteres de los vectores.

Para comparar las cadenas se utiliza un bucle mediante el cual se van comparando todos los bytes de una palabra almacenados en el vector correspondiente al fichero con cada uno de los caracteres del vector de la cadena en orden. Si se detecta que son equivalentes entre sí, es decir, que el carácter es igual; entonces, el flujo del programa continúa desviándose a la etiqueta "Iguales" la cual va comprobando si a su vez los caracteres contiguos de la cadena se corresponden con los siguientes caracteres del vector. En caso de que ocurra se incrementaría en uno el valor anterior que contenía la variable que almacena el número de palabras que contiene la cadena a comparar. En caso contrario se continúa analizando la palabra.

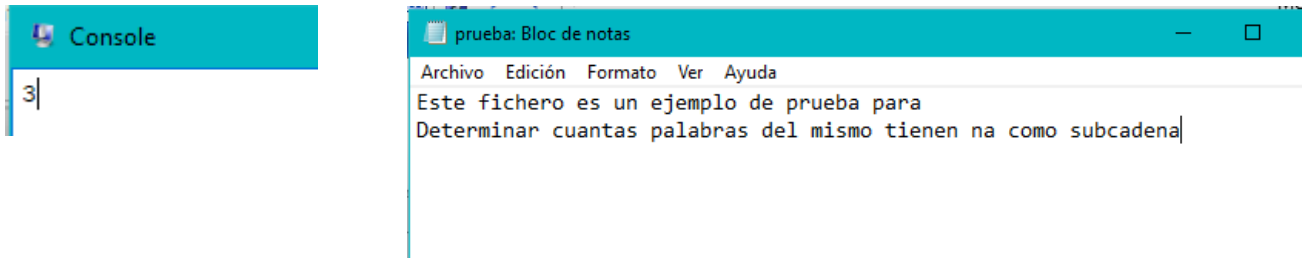
Una vez acabado el análisis el comparador vuelve al bucle Lector donde se realizaría esta función descrita anteriormente de forma recursiva hasta que el programa detecte que se ha finalizado el fichero. En el instante en el que este finalice nos dirige a la etiqueta ComparadorFinal que se ejecuta de forma similar a Comparador aunque en esta, una vez acabado el análisis no vuelve al bucle Lector, sino que continua hacia la etiqueta FinFichero. A partir de aquí se imprime el número de palabras detectadas con la cadena y se retorna al flujo principal donde se finalizaría la ejecución del programa.

Para que sean más claras las explicaciones, se ha realizado un diagrama de flujo explicativo:

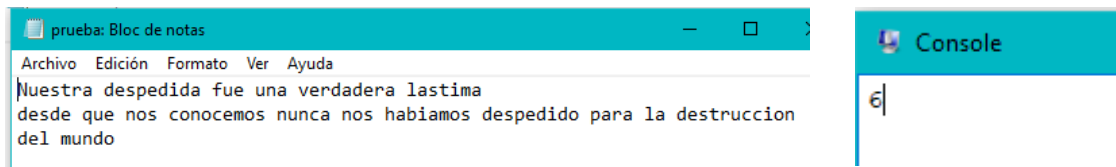


b) Conclusiones y problemas encontrados.

Ya habiendo comprobado anteriormente que detecta los dos errores fundamentales de la práctica, es decir que haya un error al abrir el fichero y que el fichero este vacío. Pasamos a comprobar el funcionamiento del programa, de tal forma que poniendo en un fichero la prueba pedida en el enunciado este es el resultado:



Además también hemos comprobado que nuestro programa funciona para cualquier cadena aleatoria que le metamos con otro fichero distinto. En este caso utilizamos la cadena “de” en un nuevo fichero y obtenemos el resultado esperado: 6



2. Ejercicio 2

a) Contenidos pedidos.

El objetivo del segundo ejercicio es crear un programa que procese un fichero de entrada formado por palabras, que podrán estar separadas por uno o más blancos o saltos de línea. El programa deberá contabilizar la longitud de todas las palabras incluidas en el fichero y generará un fichero de salida en el que indicará el número de ocurrencias que aparecen en el texto de una determinada longitud y una línea final con la longitud media de las palabras leídas.

En el segmento data de este fichero se almacena la dirección donde se encuentra el fichero de entrada y la del fichero de salida, otros datos necesarios para la lectura y escritura del fichero como el tipo de fichero y el número de bytes a leer auxiliares para la impresión del resultado (cadenas de String). Por otro lado también reservamos memoria para dos vectores uno para almacenar los datos que se encuentran en el fichero de entrada y otro para los del fichero de salida.

La estructura del programa es similar a la del ejercicio1, se pasan el fichero de entrada y el de salida por parámetros y el programa comienza con la apertura del fichero. De la misma forma se comprueba si existe un error en el fichero, en caso afirmativo el programa saltaría hacia la

instrucción `ErrorFichero`, imprimiendo la excepción y volviendo al flujo principal donde a continuación se cerraría el programa.

Por otro lado en caso de que el elemento funcione correctamente, se comienza la lectura byte a byte almacenándose cada uno de ellos en el vector `VFicheroE` comprobándose si el fichero está o no vacío, si lo estuviera la ejecución sería similar a la ejecución explicada anteriormente cuando hay un error. A continuación, proseguimos la lectura y cuando un espacio o un salto de línea sean detectados el programa supone que una palabra ha sido leída por lo que continúa hacia la etiqueta `PalabraLeida` donde en el vector `VNPalabras` (el cual está formado por 256 posiciones; es decir, el máximo de bytes que serán permitidos para cada una de las palabras permitidas del fichero). Previamente en el registro `%s2` se habría almacenado el número de bytes de la palabra que acaba de ser leída imaginando que la palabra leída tuviera `X` bytes, se accedería a la posición `X` del vector `VNPalabras`. El programa a su vez recuperará el valor anterior en esa posición siendo este el número de palabras de esa longitud e incrementará en uno ese valor quedando así constancia de que existe una nueva palabra con esa longitud.

Esta operación se realizara sucesivamente de forma recursiva hasta que se detecte el último byte del fichero. En ese caso procederemos a un salto a la etiqueta `PalabraLeidaFinal`. Al tratarse de la última palabra, tras almacenar en el vector mencionado anteriormente su información procederemos a recopilar cual es el número de palabras de cada longitud. Para ello, se realiza un bucle el cual recorre cada una de las posiciones del vector y a su vez llama a la función `ImpresionOcurrencia` que detecta si el valor que corresponde al número de palabras de dicha longitud es o no 0, si es 0 retornará y el bucle continuará su ciclo. En otro caso, el programa imprimirá la información obtenida por pantalla y almacenará los datos en el vector `VFicheroS` para su posterior escritura en el fichero de salida. En el vector se almacenan los números en su equivalente `ascii`. Así serán reconocidos como caracteres por el compilador cuando proceda a su escritura.

Una vez finalizado se procede a determinar la media de las palabras en la etiqueta `DeterminarMedia` en la cual se deben convertir los dos números enteros (el número de palabras y el número de caracteres total del fichero) a dos números en coma flotante de simple precisión, se realizara la división entre ellos y se imprimirá por pantalla el número obtenido que nos determinará la media.

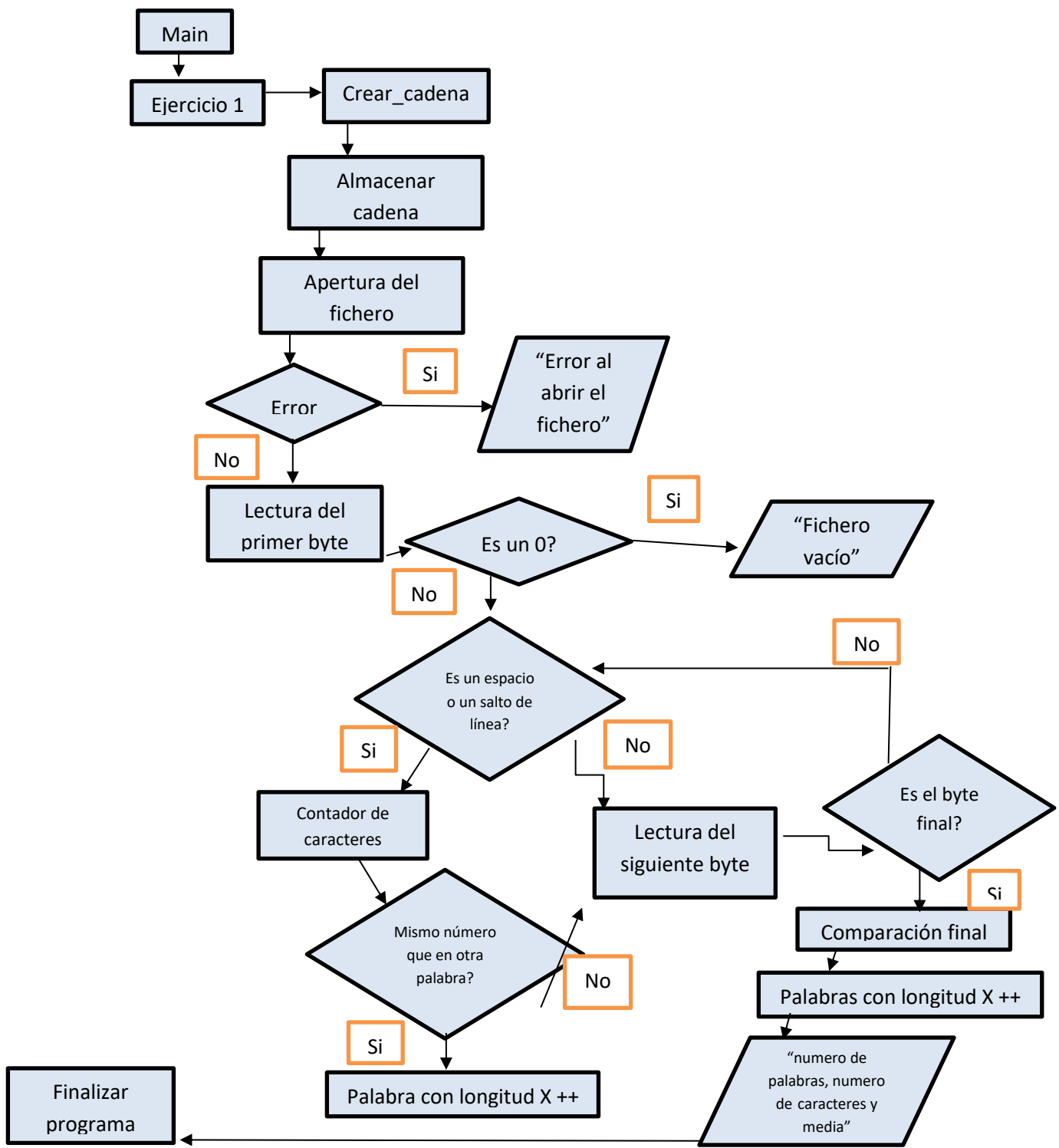
El próximo paso del programa sería crear el fichero de salida. Se establece la dirección recibida por parámetro en la cual se truncará el fichero ya existente. A continuación, se indica al programa el vector del cual se van a leer los bytes y el número de bytes a leer a su vez.

Hemos elegido una estructura de bucle para llevar a cabo la escritura en el fichero. Ayudándonos de registros temporales que funcionan como auxiliares de desplazamiento recorreremos el vector donde se almacena la información y en una iteración se imprime la cadena "Longitud", la posición del vector que indica una longitud, la cadena "Dos puntos" y finalmente el número de palabras de esa longitud (dato almacenado en la posición contigua a la de la longitud en el vector), así sucesivamente hasta finalizar el mismo.

Como auxiliar para la comprobación del buen funcionamiento del programa, se imprime por pantalla la información bien recogida del fichero; es decir, el número de palabras y el número de caracteres.

Finalmente, se cierra el fichero y se recoge del registro \$ra la dirección contigua al lugar donde la función ejercicio1 ha sido llamada.

A continuación se adjuntará un diagrama de flujo que servirá como documento aclarativo del algoritmo anterior:



b) Conclusiones y problemas encontrados.

Durante la realización de este ejercicio, hemos encontrado dificultades a la hora de trasladar los resultados obtenidos del análisis del fichero a un nuevo fichero.

El método de escritura en el fichero aportado en el guion de la práctica, sigue la siguiente estructura: se indica la dirección de un espacio en memoria y se indica el número de bytes a leer del mismo; hemos conseguido realizar comprobaciones de que el método de escritura en el fichero funcionaba correctamente.

Para ello se han realizado diferentes pruebas escribiendo cadenas de caracteres como datos del ejercicio e integrando las mismas en el fichero de salida.

Por otro lado, nuestro conjunto de datos obtenidos los cuales han sido almacenados en un vector no eran leídos correctamente, sino que en el fichero de salida se observaban una serie de caracteres y palabras confusas que no se asemejaban a los resultados esperados.

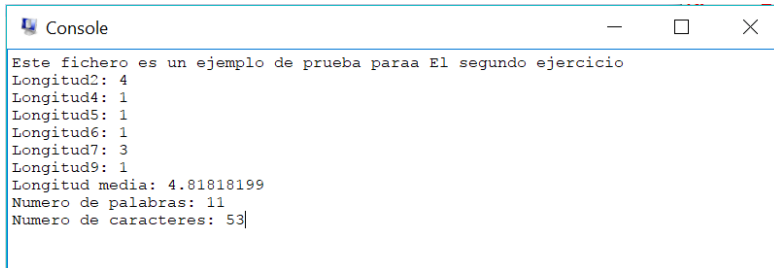
El problema reside en que nuestros datos son de tipo entero, por lo tanto ocupan una palabra en memoria. El método de lectura propuesto lee byte a byte, no pudiendo ser así capaces de escribir los resultados obtenidos en el fichero.

Finalmente, ante esta adversidad, hemos sido capaces de convertir nuestros datos enteros a su equivalente ascii. De esta manera el sistema reconoce los bytes como los números deseados e imprime satisfactoriamente la información obtenida durante el transcurso del programa

En definitiva, no hemos conseguido llevar a cabo la escritura de un elemento en coma flotante sobre el fichero, debido a que no podíamos convertirlo a un carácter para su lectura byte a byte. Adjuntamos la imagen que se esperaría ver producto de abrir el fichero tras su correspondiente escritura:

```
Longitud2:4
Longitud4:1
Longitud5:1
Longitud6:1
Longitud7:3
Longitud9:1
Longitud media:4
```

Finalmente, al observar la impresión por pantalla se puede determinar que el algoritmo que produce la media de las palabras es correcto pese al fallo de conversión a la hora de realizar la escritura en el fichero.

A screenshot of a Windows-style console window titled "Console". The window has standard minimize, maximize, and close buttons. The text inside the console is as follows:

```
Este fichero es un ejemplo de prueba paraa El segundo ejercicio
Longitud2: 4
Longitud4: 1
Longitud5: 1
Longitud6: 1
Longitud7: 3
Longitud9: 1
Longitud media: 4.81818199
Numero de palabras: 11
Numero de caracteres: 53|
```

Aclaración:

Durante el proceso de creación del programa hemos intentado detallar en los comentarios del código las pequeñas instrucciones u operaciones, para poder centrarnos en una vista general del programa en esta memoria.