Grado Ingeniería Informática (grupo A)
Programación de Sistemas y Concurrencia
Primer Control - Abril de 2017

Apellidos y Nombre: _____

Grupo: _____ **DNI:** _____ **Ordenador:** _____

Ejercicio lenguaje C

Se desea realizar una aplicación que lea un texto de entrada y vaya introduciendo cada palabra que tenga entre 3 y 13 letras en una lista de palabras. Para ello se debe utilizar un array de listas, con todas las palabras que tengan la misma longitud en una lista. Una palabra es una secuencia arbitraria de caracteres que termina en un carácter blanco o fin de línea.

Una vez creado el array de listas, se escribirá por consola o en un nuevo archivo de salida con el siguiente formato:

- Una línea de texto donde se indique el tamaño de las palabras que viene a continuación. Ejemplo: "Palabras de XX letras:"
- En las siguientes líneas todas las palabras de XX letras separadas por comas.
- Una línea en blanco.
- Repetimos esta estructura para todas las listas de 3 a 13 letras.
- En el caso de que la lista para una determina cantidad de letras esté vacía, se pondrá el siguiente mensaje indicándolo: "No se han encontrado palabras de XX letras".

NOTA: Ver ejemplo en la siguiente página

La estructura de datos de las listas debe ser una lista enlazada simple que debe seguir la interfaz de programación propuesta en el archivo lista.h.

- Apartado A. Leer desde el archivo por defecto (Lorem_Ipsum.txt) y escribir la salida por la consola, utilizando los métodos crear(), lista_vacia(), insertar(), destruir() y escribir() de la lista enlazada (75%).

  Para ello, se debe implementar un método main con el siguiente comportamiento:

  - creación de un array de listas de palabras,
  - lectura del fichero e inserción de cada palabra en la lista correspondiente
  - escritura en la consola de cada una de las listas con el formato indicado arriba
  - destrucción de todas las listas creadas

- Apartado B. Añadir que además de por consola, la salida se escriba en el archivo Palabras3_13.txt. Implementar escribir_fichero() de la lista enlazada (10%)
- Apartado C. Refactorizar el código de modo que la función escribir() haga uso de la función escribir_fichero() para la consola (salida estándar o stdout). Asimismo simplifique el código para que la salida por consola y por fichero se traten de la misma forma en main(), idealmente en una función adicional en el archivo principal. (5%)
- Apartado D. Añadir la opción de que no se guarden palabras repetidas. Implemente la función buscar_palabra() de la lista enlazada (10%)

Recordar que para la manipulación de string deben utilizarse las funciones strlen, strcmp, strcpy de la librería String.h

```
/*
 * Crea una lista vacía
 */
void crear(Lista *l);

/*
 * Comprueba si un a lista está vacía
 * Devuelve 0 si NO lo está
 */
int lista_vacia(Lista l);

/*
 * Escribe en consola el contenido de una lista de palabras separadas por coma
 * l: lista enlazada de palabras
 */
void escribir(Lista l);

/*
 * Escribe en un fichero de salida el contenido de una lista de palabras
 * separadas por coma.
 * fp: Puntero a un objeto FILE que identifica el stream de salida
 * l: lista enlazada de palabras
 */
void escribir_fichero(FILE * fp, Lista l);

/*
 * Inserta una palabra al final de una lista enlazada.
 * No comprueba si la palabra existe, si se desea no repetir palabras
 * se debe utilizar buscar_palabra() y comprobar antes de invocar esta función
 * palabra: la palabra que se desea insertar
 * l: lista enlazada de palabras
 */
void insertar(char* palabra, Lista* l);

/*
 * Elimina todos los ítems de la lista enlazada
 * Debe devolver la memoria dinámica utilizada para cada uno de ellos
 * Para comprobar que se eliminan los ítems, escriba un mensaje por consola
 * indicando la palabra de ítem que se va a eliminar
 * l: La lista enlazada que se desea eliminar
 */
void destruir(Lista* l);

/*
 * Comprueba si una palabra está en la lista enlazada
 * palabra: la palabra que se desea buscar
 * l: lista enlazada de palabras
 * Devuelve 0 si la palabra NO está en la lista
 */
int buscar_palabra(char* palabra, Lista l);
```

Nota 1: El archivo test_lista.c le puede servir como pruebas de la lista. Recuerde ir renombarando los main() que solo se puede tener uno. Si todo funciona bien la salida debería ser algo como esto:

```
Lista vacia? TRUE

UnaPalabra,
UnaPalabra, DosPalabras,
UnaPalabra, DosPalabras, TresPalabras,
Lista vacia? FALSE
UnaPalabra, DosPalabras, TresPalabras, CuatroPalabras,
Esta UnaPalabra ? TRUE
Esta OtraPalabra ? FALSE
Eliminando item UnaPalabra
Eliminando item DosPalabras
Eliminando item TresPalabras
Eliminando item CuatroPalabras
Lista vacia? TRUE
```

Nota2: La salida para el archivo de entrada Lorem_Ipsum.txt **sin repeticiones** es la siguiente

```
Palabras de 3 letras:
sit, sem, non, Nam, sed, eu,, vel, Sed,

Palabras de 4 letras:
nisl, quis, quam, eros, Cras, amet, dui., eget, Duis, odio, urna, elit, orci,

Palabras de 5 letras:
Lorem, ipsum, dolor, amet,, elit., Donec, augue, orci,, diam., Fusce, velit, vitae, quis.,
ante., felis, Nulla, metus, quam,, porta, orci.,

Palabras de 6 letras:
lorem., libero, mauris, turpis, metus,, purus,, ornare, felis,, purus., tempor, mollis,
tellus, vitae,,

Palabras de 7 letras:
pretium, rhoncus, dictum., maximus, laoreet, lectus., Quisque, tortor,, euismod, sodales,
posuere, tellus., viverra,

Palabras de 8 letras:
volutpat, vehicula, pulvinar, ultrices, finibus., suscipit,

Palabras de 9 letras:
venenatis, faucibus,, porttitor, vulputate, facilisis, fermentum, elementum, imperdiet,
suscipit.,

Palabras de 10 letras:
adipiscing, efficitur., efficitur,, vestibulum, Vestibulum, porttitor., dignissim.,

Palabras de 11 letras:
consectetur,

Palabras de 12 letras:
pellentesque,

Palabras de 13 letras:
No se han encontrado palabras de 13 letras
```