UNIVERSIDAD DE MÁLAGA | uma.es

Dpto. de Lenguajes y Ciencias de la Computación

Programación de Sistemas y Concurrencia

1ª Convocatoria Ordinaria. Curso 2020-2021
15 de Junio de 2021.

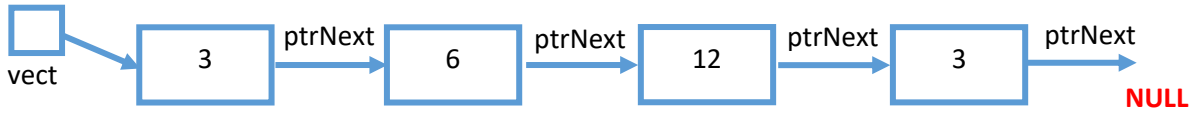
Enunciado

En el presente ejercicio se pide implementar el módulo de C necesario para trabajar con vectores de enteros. Un vector tiene la estructura de una lista enlazada a través de los siguientes tipos proporcionados en el fichero de cabecera Vector.h:

```
typedef struct VectorNode * TVector;
struct VectorNode {
	int value;
	struct VectorNode * ptrNext;

};
```

Siguiendo estas estructuras, un vector, como por ejemplo [3, 6, 12, 3], se representará mediante los siguientes nodoz enlazados:



También se proporciona al alumno un fichero `Driver.c` con una función `main` que testea el funcionamiento de las distintas funciones que se han de desarrollar. Las funciones, que hay que implementar en el fichero **Vector.c**, son las siguientes:

```
/* Crea un vector vacío (0,5 pts) */
TVector create();

/* Añade un entero 'n' al final del vector (1 pt) */
/* El vector se pasa por referencia */
void putTail(TVector * ptrVector, int n);

/* Devuelve el número de enteros que contiene el vector (0,5 pts) */
/* El vector se pasa por valor */
int length(TVector vector);

/* Libera la memoria usada por el vector (1 pt) */
/* El vector se resetea al vector vacío */
/* El vector se pasa por referencia */
void destroy(TVector * ptrVector);

/* Muestra en consola el contenido del vector, desde el principio hasta el final (1 pt) */
/* El vector se pasa por valor */
void display(TVector vector);

/* Suma dos vectores (1,5 pts) */
/* Los enteros del segundo vector 'otherVector' se suman a los
 * valores del primer vector 'targetVector'.
 * Así, el contenido del primer vector queda modificado.
 * Si las longitudes de los vectores son distintas, se muestra un mensaje
```

```
 * de error y se retorna.
 */
/* Ambos vectores se pasan por valor */
void add(TVector targetVector, TVector otherVector);

/* Guarda el contenido de un vector en un fichero binario (1,5 pts) */
/* El primer item del fichero debe ser la longitud del vector y, posteriormente
 * se guardan los valores del vector, desde el primero hasta el último.
 */
/* El vector se pasa por valor */
void saveToFile(TVector vector, char* filename);

/* Carga un fichero con la estructura de la función anterior 'saveToFile' (1,5 pts) */
/* Se retorna un nuevo vector con los datos cargados */
TVector loadFromFile(char* filename);
```

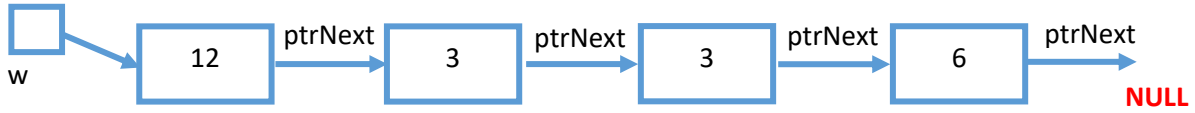
Para poder probar la función loadFromFile sin necesidad de tener que implementar la función saveToFile, se proporciona un fichero pre-generado example.bin que, una vez cargado con la implementación del alumno de loadFromFile debe devolver el vector [13, 56, 12, 67, 89, 3, 4, 56, 3, 4].

```
/* Devuelve un nuevo vector con los mismos datos que 'vector' pero en el orden dado
por 'reorder' (1,5 pts) */
/* 'reorder' es un array de tantos enteros como sea la longitud de 'vector' */
/* 'reorder' no puede tener valores repetidos. La primera posición es la 0 */
/* Se retorna un nuevo vector con los datos reordenados */
/* Si el parámetro 'reorder' no tiene el formato adecuado, se emite un mensaje de
error y se retorna NULL */
TVector shuffle(TVector vector, int* reorder);
```

Como ejemplo de funcionamiento de la función shuffle, si ésta es invocada con el vector vect del dibujo anterior de la siguiente forma:

```
int reorder[] = {2,3,0,1};
w = shuffle(vect, reorder);
display(w);
```

entonces el resultado será w = [12, 3, 3, 6]:



Anexo. Los prototipos de las funciones de lectura y escritura en ficheros de la biblioteca <stdio.h> son los siguientes (se dan por conocidos los prototipos de las funciones de <stdlib.h> que necesites, como free o malloc):

FILE *fopen(const char *path, const char *mode): Abre el fichero especificado en el modo indicado ("rb"/ y "wb" para lectura/escritura binaria y "rt"/"wt" para lectura/escritura de texto). Devuelve un puntero al manejador del fichero en caso de éxito y NULL en caso de error.

int fclose(FILE *fp): Guarda el contenido del buffer y cierra el fichero especificado. Devuelve 0 en caso de éxito y -1 en caso de error.

LECTURA/ESCRITURA BINARIA

unsigned fread(void *ptr, unsigned size, unsigned nmemb, FILE *stream): Lee nmemb elementos de datos, cada uno de tamaño size bytes, desde el fichero stream, y los almacena en la dirección apuntada por ptr. Devuelve el número de elementos leídos.

unsigned fwrite(const void *ptr, unsigned size, unsigned nmemb, FILE *stream): Escribe nmemb elementos de datos, cada uno de tamaño size, al fichero stream, obteniéndolos desde la dirección apuntada por ptr. Devuelve el número de elementos escritos.

LECTURA/ESCRITURA TEXTO

int fscanf(FILE *stream, const char *format, ...): Lee del fichero stream los datos con el formato especificado en el parámetro format, el resto de parámetros son las variables en las que se almacenan los datos leídos en el formato correspondiente. La función devuelve el número de variables que se han leído con éxito.

int fprintf(FILE *stream, const char *format, …): Escribe en el fichero stream los datos con el formato especificado en el parámetro format. El resto de parámetros son las variables en las que se almacenan los datos que hay que escribir. La función devuelve el número de variables que se han escrito con éxito.