**Enunciado Entrega Evaluación Continua 2022-2023** **Estructura de Datos y Algoritmos II**

**Ejercicio sobre Memoria Secundaria. Organización Directa. Dispersión**

---

Partiendo del archivo *alumnos.dat* que contiene varios registros de *tipoAlumno*, tal como se define en dispersion.h, se debe construir un nuevo archivo *alumnosC.hash*, que organice estos registros según el Método de Dispersión.
Para establecer algunos de los parámetros de configuración y para guardar algunas de las características del fichero se utiliza un **registro de cabecera**, de tipo *configReg*, como primer registro del fichero, previo a los cubos que contienen los registros de tipo alumno. La definición de esta estructura se proporciona en dispersion.h y se muestra en listado 1.

Listado 1: Tipos Método Dispersión

```
#define C 5           // Capacidad del cubo
typedef struct {
  int nCubos;         // Número de cubos en el área prima
  int nCubosDes;      // Número de cubos en el área de desborde
  int nCuboDesAct;    // Número del primer cubo desborde con espacio para más registros
  int densidadMax;    // Máxima densidad de ocupación permitida
  int densidadMin;    // Mínima densidad de ocupación permitida
  int numReg;         // Número total de registros en el archivo
  int numRegDes;      // Número de registros desbordados
} regConfig;

typedef struct {
  tipoAlumno reg[C];
  int numRegAsignados;
  int desbordado;    // Este campo indica si el cubo se ha desbordado(1) o no(0)
  }tipoCubo;
```

Para conseguir esta organización, se deben seguir los siguientes criterios:

- El número de cubos iniciales y de desborde, debe incluirse en el registro de configuración, en los campos **nCubos** y **nCubosDes**, respectivamente.
- Todos los cubos tienen capacidad **C**(definida en dispersion.h)
- Cada cubo debe almacenar, en el campo *numRegAsignados*, **solamente** el número de registros que contiene, sin contar los desbordados, y un valor lógico, en el campo *desbordado*, que inidica si el cubo se ha desbordado(VERDADERO) o no(FALSO). La definición de la estructura *tipoCubo* se proporciona en dispersion.h y se muestra en listado 1.
- Los registros desbordados se irán almacenando secuencialmente en el área de desborde. Los 5 primeros registros desbordados irán al primer cubo de desborde, los 5 siguientes al siguiente cubo de desborde, etc. El registro de configuración contiene el valor del primer cubo de desborde que tiene espacio de almacenamiento disponible, campo ***nCuboDesAct***. Este valor debe actualizarse cada vez que se llene el cubo y cuando se llene el último debe crearse un nuevo cubo de desborde.
- El registro de configuración debe contener además información sobre las densidades de ocupación máxima y mínima permitidas, el número de registros que almacena el fichero y el número de ellos que se han desbordado, campos ***densidadMax, densidadMin, numReg y numRegDes***, respectivamente
- Utilizar como clave el campo DNI de los registros de tipoAlumno y aplicar la función Módulo como función hash para asignar un registro a un cubo.

## Ejercicio 1. Creación del fichero.

Teniendo en cuenta estas especificaciones se debe obtener el fichero *alumnosC.hash* y para ello deben implementarse, obligatoriamente, las siguientes funciones:

1. Implementar la función

   **int creaHash(char *fichEntrada,char *fichHash, regConfig *regC)**

   que toma como entrada el nombre del fichero que se desea organizar (*alumnos.dat*), el nombre del fichero que va a crear (*alumnosC.hash*) y el registro con los parámetros inciales de configuración (nCubos, nCubosDes, densidadMax y densidadMin)necesarios para implementar el método de dispersión.
   La función irá modificando el resto de campos del registro de configuración (numReg, numRegDesb y nCuboDesAct) a medida que se van añadiendo registros al fichero y registrará estos cambios con la información final en el fichero.
   La función devuelve un entero que indica:

   - 0 Si el proceso acaba correctamente y la densidad de ocupación está dentro de los límites indicados por los parámtros densidadMax y densidadMin.
   - -1 Si hay problemas con el fichero de entrada
   - -2 Si hay problemas con el fichero de salida
   - -3 Si se supera la densidad Máxima de ocupación
   - -4 Si se reduce la densidad Mínima de ocupación
   - -5 Si ocurre algún otro error en el proceso

   **Para implementar correctamente esta función debe utilizar, obligatoriamente, las dos funciones auxiliares (creaHvacio y desborde) que se especifican a continuación.**

2. Una función que cree el fichero con todos los cubos inicialmente vacios

   **void creaHvacio(char *fichHash, regConfig *regC)**

   dónde el parámetro regC incluirá la información de configuración necesaria para la creación del fichero (nCubos, nCubosDesborde, densidadMax y densidadMin). El resto de campos deben tener los valores iniciales apropiados:

   - numReg y numRegDesb valor 0
   - nCuboDesAct coincide incialmente con el valor del campo nCubos

3. Una función para el tratamiento de los resgistros desbordados

   **void desborde(FILE *fHash,tipoAlumno *reg, regConfig *regC)**

   Cuando un registro sea asignado a un cubo lleno debe guardarse en el primer cubo de desborde que tenga espacio (valor del campo *nCuboDesAct* del registro de configuración). Este valor debe actualizarse cuando el cubo de desborde actual se llene. Además si se llena el último cubo de desborde, la función debe añadir un nuevo cubo de desborde vacio en el fichero, modificando los campos del registro de configuración adecuados.

## Ejercicio 2. Acceso al fichero.

Para manipular el fichero (*alumnosC.hash*) que se obtiene como resultado del método de dispersión del ejercicio 1, se deben implementar las siguientes funciones:

1. Una función de búsqueda que aporte información sobre la ubicación del registro buscado en el fichero. El prototipo de esta nueva función será:

   **tipoAlumno *busquedaHash(FILE *f, char *dni, int *nCubo, int *ncuboDes, int *posReg, int *error)**

   que busca en el archivo creado un registro a partir de su clave, parámetro *dni* de entrada a la función. Esta función devuelve el registro si lo encuentra o el valor NULL si no existe. Además devuelve información sobre la situación del registro en el fichero en los últimos tres parámetros, información que puede utilizarse en procesos posteriores de modificación y/o eliminación:

   - **nCubo:** número de cubo en el que se encuentra el registro si no está desbordado o en el que debería estar si está desbordado.
   - **nCuboDes:** si el registro está desbordado el número de cubo de desborde en el que se encuentra, considerando los cubos desbordados numerados secuencialmente a continuación de CUBOS. Si el registro no está en el área de desborde se le asigna a este parámetro el valor -1.
   - **posReg:** posición del registro en el cubo en el que se encuentra (inicial o desbordado)
   - **error:** este parámetro sirve para detectar si ha habido algún error en el proceso, la función debe asignarle alguno de los siguientes valores:
     - 0 si el proceso acaba correctamente y el registro exite
     - -1 si el proceso acaba correctamente pero el registro no existe
     - -2 si hay problemas con el fichero de datos
     - -4 si ocurre algún otro error en el proceso

2. Una función que, utilizando la información que aporta la función previa, permita modificar el campo provincia a los registros del fichero que se obtiene como resultado en el método de dispersión (*alumnosC.hash*). La función a implementar debe seguir el prototipo:

   **int modificarReg(char *fichero, char *dni, char *provincia)**

   donde el primer parámetro indica el nombre del fichero *hash*, el segundo el *dni* del alumno a modificar y el último, el nuevo valor del campo provincia que se desea asignar al alumno. La función devuelve un entero con el siguiente valor:

   - el número de cubo en el que se encuentra el registro modificado si existe
   - -1 si el registro no existe
   - -2 si hay problemas con el fichero de datos
   - -4 si ocurre algún otro error en el proceso

Para la realización de la práctica se proporcionan tres ficheros:

- **dispersion.h** con los tipos y prototipos necesarios para la implementacion del ejercicio
- **dispersion.c** con la implementación de la función *leeHash*, que permite mostrar la información que almacena el fichero aulmnosC.hash
- **resultado15_4.txt** que muestra el resultado de la organización por dispersión utilizando 15 cubos iniciales y 4 de desbode. Es decir, muestra el resultado de aplicar la función leeHash al fichero alumnosC.hash, organizado en 15 cubos iniciales y 4 de desborde

**Condiciones de la ENTREGA:**

- Se debe subir un **único fichero comprimido** que incluya:
  - un fichero con el código C de las funciones implementadas (dispersion.c)
  - un **fichero de prueba** que permita verificar el correcto funcionamiento de las funciones implementadas
  - un fichero de resultados (resultado10_4.txt) que muestre la organización final del fichero utilizando 10 cubos iniciales y 4 de desborde, como valores iniciales en los campos del registro de configuración
  - un fichero **makefile** que permita la perfecta compilación de estos ficheros
- Todos los ficheros deben incluir una línea inicial con el **nombre, dni y grupo de prácticas** del alumno. El fichero comprimido debe subirse a la tarea antes de las **23:55** horas del **5 de Mayo de 2023**. No obstante, la tarea permitirá entregas retrasadas hasta el día **8 de Junio a las 14:00**.