Tarea Evaluación Continua 2024-2025 **Estructura de Datos y Algoritmos II**

Ejercicio sobre Memoria Secundaria. Generalización del Método de Dispersión

---

El objetivo de esta práctica es la generalización del método de dispersión, de forma que nos permita crear archivos de diferentes tipos de registro (alumnos, asignaturas, titulaciones, empleados, departamentos, etc.) y diferentes parámetros de configuración (cubos, cubos de desborde, densidades de ocupación, etc.).

Para establecer algunos de los parámetros de configuración, y para guardar algunas de las características del fichero, se utilizará un **registro de cabecera**, de tipo *configReg*, como primer registro de cualquier fichero, previo a los cubos que contienen los registros de datos del fichero. La definición de esta estructura se proporciona en *dispersion.h* y se muestra en listado 1.

Listado 1: Tipos Método Dispersión

```
#ifdef ALUM
  #include "alumno.h"
  typedef tAlumno tipoReg;
#endif
#ifdef ASIG
  #include "asignatura.h"
  typedef  tAsignatura tipoReg;
#endif
typedef struct {
  int nCubos;        // Número de cubos en el área prima (>8)
  int nCubosDes;     // Número de cubos en el área de desborde (>4)
  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 {
  tipoReg reg[C];
  int numRegAsignados;
  int desbordado;    // Este campo indica si el cubo se ha desbordado(1) o no(0)
  } tipoCubo;

typedef struct {
  int cubo;      // Número de cubo asignado a un registro
  int cuboDes;   // Número de cubo en que se encuentra si se ha desbordado, -1 en otro caso
  int posReg;    // Posición del registro en el cubo
  } tPosicion;
```

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**, sus valores deben ser al menos 8 y 4 respectivamente.
- 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. El campo *desbordado* solo se pondrá a verdadero si el cubo

tiene al menos un registro en el área de desborde, es decir, si tiene C registros y *desbordado* es falso, el cubo estará completo pero no hay ningún registro desbordado.

- Todos los cubos tienen capacidad **C**(definida en el fichero cabecera correspondiente a *tipoReg*), su valor debe ser al menos 4.
- Los registros desbordados se irán almacenando secuencialmente en el área de desborde. Los C primeros registros desbordados irán al primer cubo de desborde, los C 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, actualizando los valores adecuados en el registro de configuración.
- 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
- El campo utilizado como clave y la función hash para asignar un registro a un cubo, dependerá en cada caso del fichero que estemos generando y, por tanto, de la definición de *tipoReg*.

Parte 1. Funciones genéricas para la creación de ficheros

1. Implementar la función

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

   que toma como entrada el nombre del fichero que se desea organizar, el nombre del fichero que va a crear 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 parámetros *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, como por ejemplo, que alguno de los parámetros de configuración no sean correctos

   Para implementar correctamente esta función debe utilizar, obligatoriamente, las funciones auxiliares (*creaHvacio, inserta*) que se especifican en los apartados 2 y 3.

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

   int 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*

   La función devuelve -2 si hay problemas con el fichero y 0 en caso contrario.

3. Una función para añadir nuevos registros en el fichero

   int insertar(FILE *f, tipoReg *reg, regConfig *regC)

   que inserta un nuevo registro *reg* en el fichero *fHash* y actualiza los campos adecuados en el registro de configuración *regC*. Esta función debe utilizar obligatoriamente la función *desborde* que se especifica en el apartado 3.
   La función devuelve -2 si hay problemas con el fichero y 0 en caso contrario.

4. Una función para el tratamiento de los resgistros desbordados

   int desborde(FILE *fHash,tipoReg *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.
   La función devuelve -2 si hay problemas con el fichero y 0 en caso contrario.

Parte 2. Implementación de funciones genéricas para el acceso al fichero

Para manipular el fichero 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:

   int busquedaHash(FILE *fHash, tipoReg *reg, tPosicion *posicion)

   que busca en el archivo creado un registro a partir de su clave, puesto que la clave depende del *tipoReg* utilizado en la creación fichero, esta debe proporcionarse en el campo adecuado del segundo parámetro. Este parámetro, por tanto, es de entrada/salida, pues en él debe devolverse el registro buscado si lo encuentra. En el tercer parámetro, de tipo *tPosicion* definido en *dispersion*.h (listado 1), la función debe devolver información sobre la situación del registro en el fichero en cada uno de sus campos (información que puede utilizarse en procesos posteriores de modificación y/o eliminación):

   - ***cubo:*** número de cubo en el que se encuentra el registro si no está desbordado o en el que debería estar si está desbordado.
   - ***cuboDes:*** 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)

   Por otra parte la función devuelve un entero que permite detectar si ha habido algún error en el proceso:

   - 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
   - -5 si ocurre algún otro error en el proceso

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

   int modificarReg(FILE *fHash, tipoReg *reg, tPosicion *posicion);

   donde el primer parámetro indica el nombre del fichero *hash*, el segundo parámetro, que depende del tipo de registro del fichero, debe incluir la clave del registro a modificar y los campos pertinentes modificados. Es imprescindible utilizar en esta función la función de búsqueda previa, para evitar modificar registros diferentes al de la clave deseada, la cual devolverá los valores apropiados que necesita el tercer parámetro sobre la posición del registro en el fichero. La función devuelve un entero con el siguiente valor:

   - 0 si el proceso acaba correctamente
   - -1 si el registro no existe
   - -2 si hay problemas con el fichero de datos
   - -5 si ocurre algún otro error en el proceso

Parte 3. Tareas a realizar para probar el correcto funcionamiento de las partes previas

1. Partiendo del archivo *alumnos.dat* que contiene varios registros de *tAlumno*, tal como se define en *alumnos.h* (listado 2), se debe construir un nuevo archivo *alumnosC.hash*, que organice estos registros según el Método de Dispersión. Utilizar como clave el campo *dni* y aplicar la función *módulo* como función hash para asignar un registro a un cubo.

Listado 2: Definición tipoAlumno

```
#define C 8 // Capacidad del cubo fichero de alumnos
typedef struct {
  char dni[9];     // campo clave
  char nombre[19];
  char ape1[19];
  char ape2[19];
  char provincia[11];
  } tAlumno;
int funcionHash(tAlumno *reg,int nCubos);
void mostrarReg(tAlumno *reg);
int cmpClave(tAlumno *reg1, tAlumno *reg2);
int buscar(char *fichero, char *dni);
int modificar(char *fichero, char *dni,char *provincia);
```

2. Partiendo del archivo *asignaturas.dat* que contiene varios registros de *tAsignatura*, tal como se define en *asignatura.h* (listado 3), se debe construir un nuevo archivo *asignaturasC.hash*, que organice estos registros según el Método de Dispersión. Utilizar como clave el campo *código* y aplicar la función *módulo* como función hash para asignar un registro a un cubo.

Listado 3: Definición tipoAsinatura

```
#define C 6              // Capacidad del cubo fichero asignaturas
typedef struct {
  int codigo; // campo clave
  char nombre[60];
  char curso;
  float creditosT;
  float creditosP;
  char tipo;
  char cuatrimestre;
  int numGrT;
  int numGrP;
  } tAsignatura;
int funcionHash(tAsignatura *reg,int nCubos);
void mostrarReg(tAsignatura *reg);
int cmpClave(tAsignatura *reg1, tAsignatura *reg2);
int buscar(char *fichero, int codigo);
int modificar(char *fichero, int codigo,float creditosT, float creditosP);
```

3. Utilizando las funciones del apartado 2 implementar una función que permita modificar la provincia de un alumno conocida su clave (*dni*) y otra que permita seleccionar y mostrar un alumno conocida su clave (prototipos en listado 2).

4. Utilizando las funciones del apartado 2 implementar una función que permita modificar los créditos de una asignatura conocida su clave (*código*) y otra que permita seleccionar y mostrar una asignatura conocida su clave (prototipos en listado 3).

5. Implementar el código necesario para insertar nuevos alumnos y nuevas asignaturas en los ficheros generados en los apartados 1 y 2 respectivamente.

Para la realización de la práctica se proporcionan los siguientes ficheros, situados en las rutas adecuadas según el tipo de fichero:

- Los ficheros secuenciales cuyos registros deben organizarse mediante el método de dispersión en *datos*:
  - *alumnos.dat*
  - *asignaturas.dat*
  - en esta ruta deben almacenarse los fichero *alumnosC.hash* y *asignaturasC.hash*
- Los ficheros cabecera en *include*:
  - ***dispersion.h*** con los tipos y prototipos necesarios para la implementación de las funciones genércias (partes 1 y 2)
  - ***alumno.h*** con los tipos y prototipos necesarios para la implementación de las funciones especificas correspondientes al tipo tAlumno
  - ***asignatura.h*** con los tipos y prototipos necesarios para la implementación de las funciones especificas correspondientes al tipo tAsignatura
- Los ficheros con codigo fuente en *src*:
  - ***dispersion.c*** con la implementación de la función *leeHash*, que permite mostrar la información que almacena cualquier fichero organizado con el método de dispersión planteado en esta tarea.
  - ***genFicheroHash.c*** con el código necesario para poder probar adecuadamente la funciones que se deben implementar para la generación de los ficheros hash. Dependiendo de las directivas de compilación este fichero de prueba permite generar el fichero *alumnosC.hash* o *asignaturasC.hash* y mostrar su contenido. La ejecución de este código puede observarse en los siguientes ficheros de *resultados*:
    - ***alumnos15_4.txt*** muestra el resultado de aplicar la función leeHash al fichero alumnosC.hash, organizado en 15 cubos iniciales y 4 de desborde, siendo la capacidad del cubo de 8 registros
    - ***asignaturas10_4.txt*** muestra el resultado de aplicar la función leeHash al fichero asignaturasC.hash, organizado en 10 cubos iniciales y 4 de desborde, siendo la capacidad del cubo de 6 registros
  - ***modFicheroHash.c*** con el código necesario para poder probar adecuadamente la funciones que se deben implementar para el acceso a los ficheros hash. Dependiendo de las directivas de compilación este fichero de prueba permite consultar y modificar registros en el fichero *alumnosC.hash* o *asignaturasC.hash*. La ejecución de este código puede observarse en los siguientes ficheros de resultados:
    - ***modAlumnos.txt***
    - ***modAsignaturas.txt***
  - en esta ruta deben estar todos los ficheros fuente necesarios para la realización de la tarea
- ***makefile***, en *bin*, para la correcta compilación de todos los ficheros que forman parte del proyecto. El código compilado que genera este fichero se guarda en:
  - código ejecutable en *bin*
  - ćodigo objeto en *lib*

Condiciones de la ENTREGA:

- Se debe subir un **único fichero comprimido** que incluya:
  - los tres ficheros fuente con el código C de las funciones implementadas (dispersion.c, alumno.c y asignatura.c)
  - un fichero de resultados que muestre la organización final del fichero alumnosC.hash utilizando 10 cubos iniciales y 4 de desborde y que incluya tanto la modificación de algún registro existente como la inserción de algún nuevo registro
  - un fichero de resultados que muestre la organización final del fichero asignaturasC.hash utilizando 8 cubos iniciales y 4 de desborde y que incluya tanto la modificación de algún registro existente como la inserción de algún nuevo registro
- 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 **21 de Mayo de 2025**. No obstante, la tarea permitirá entregas retrasadas hasta el día **28 de Mayo a las 18:00**.