Tarea 6 - Programación y algoritmos
Giovanni Gamaliel López Padilla

Índice

1. Archivo de entrada

El archivo que contiene los datos iniciales del programa se dará al momento de iniciarse el programa. Este archivo debe contener como cabecera los siguientes titulos: *Calificacion, Edad, Grupo, Turno* y *Nombre*.

Los datos que contiene el archivo inicialmente se muestran en la tabla 1.

Calificacion	Edad	Grupo	Turno	Nombre
A+	9	C	V	Valeria Quirarte
A-	12	A	V	Valentina Quirarte
B	10	D	M	Luis Aldama
B-	14	A	V	Marco Lopez
A	15	A	V	Mayra Medellin
C	13	E	M	Pedro Arturo

Tabla 1: Datos iniciales del archivo data.txt.

1.1. Lectura de datos

Al inicio del programa se obtiene el número de personas (filas) que contiene el archivo. Esta cantidad es obtenida en la función obtain_size. Con esta cantidad es iniciado el arreglo de estructuras students. Al tener ya resevada la memoria para los datos de la tabla 1, se realiza la lectura de datos. Este mecanismo es realizado en la función obtain_information.

2. Menú

Al tener los datos almacenados dentro del arreglo, se inicia el menú. El menú contiene las siguientes opciones:

- Imprimir un archivo
- Ordenar por nombbre
- Ordenar por edad
- Ordenar por promedio
- Numero de estudiantes por grupo
- Numero de estudiantes por turno
- Baja de un estudiante
- Alta de un estudiante
- Invertir estudiante
- Salir

Estas opciones se ejecutan con la función menu. La función lee un número entero entre 0-9 para elegir las opciones posibles, ejecuta la función de la opción y al termino regresa a la lectura de la nueva opción. La única manera de salirse del menú es seleccionando la opción *Salir*.

3. Imprimir en un archivo

Esta opción es habilitada una vez se hayan leido los datos base. Al seleccionar esta opción por medio del menú ejecuta la función print_file. La cual realizará la impresión en el archivo dado siguiendo el formato descrito en la sección 1.

Al inicio de la función el archivo es cerrado, para ser abierto en modo de escritura. Al termino de la función, el archivo será cerrado y abierto nuevamente en modo de lectura. Esto para asegurar los cambios dentro del archivo si es que el programa llega a cerrarse inesperadamente.

4. Ordenar información

EL menú contiene descritos diferentes modos de ordenar la información de los estudiantes. Los criterios creados para el ordenamiento son los siguientes:

- Ordenar por nombre
- Ordenar por edad
- Ordenar por promedio

Los criterios antes mencionados siguien el mismo algortimo de ordenamiento. EL algoritmo de ordenamiento planteado es el quick_sort, el cual habia sido desarrollado en tareas anteriores. Esta función fue reimplementada para recibir como argumento una función el cual le indica que objeto (conjunto de carácteres o números) es mayor con respecto a otro. Los algoritmos de cada modo de ordenamiento estan contenidos en el archivo sort.h. Los datos ordenados son almacenados en el programa y no sobreescriben el archivo dado. Para guardar estos datos se tiene que ejecutar la opción descrita en la sección 3.

4.1. Ordenar por nombre

La comparación de los nombres es realizada por orden alfabetico. Si un nombre es prefijo del otro, este se pondrá ocupara un lugar menor. Por ejemplo Luisa > Luis. Esta función esta contenida en comparison_names. El programa supone que los nombres estan bien escritos, esto es que el inicio de los nombres son mayusculas.

4.2. Ordenar por edad

La comparación de las edades es realizada en orden ascendente. Al ser datos de tipo entero su comparación es más sencilla. Esta función esta contenida en comparison_ages.

4.3. Ordenar por promedio

La comparación de las calificaciones es realizada de modo descendente. Esto siguiendo el orden de USA, esto es, A+ > A > A- > B+ > B > B- > C. Esta función es implementada en comparison_grades. El algoritmo es semejante al ordenamiento de nombres, la diferencia radica en que máximo tendremos dos carácteres en cada elemento. Entonces se realizará el siguiente algoritmo:

```
1     int compare = grade1[0] - grade2[0];
2     if (compare == 0)
3     {
4         compare = compare_sign(grade1[1], grade2[1]);
5         if (compare != 0)
6             return compare;
7         compare = compare_sign(grade2[1], grade1[1]);
8         if (compare != 0)
9             return compare;
10        compare = grade1[1] - grade2[1];
11        return compare;
12    }
13    return compare;
```

En la linea 2, se comprueba que no sea la misma letra inicial, si no lo es, entonces devolvera la diferencia al algoritmo de quick_sort. Si es igual, entonces recibará si se trata de un signo +, - o en blanco. Esta función de auxilia de compare_sign para realizar llevar el orden de preferencia en los signos. Si el programa llega a la linea 11, esto quiere decir que las calificaciónes que esta comparando tienen signo.

5. Contabilización

El menú contiene diferentes modos para contabilizar la información de los estudiantes. Los criterios creados son los siguientes:

- Cobtabilización por grupo
- Contabilización por turno

Los dos criterios reciben un arreglo el cual llevara el conteo de cada tipo. Al inicio de cada criterio se inicializa en ceros todos los elementos del arreglo, ya que no se conoce si se realizó algún cambio en los datos. Los programas de esta sección se encuentran contenidos en el archivo count.h.

5.1. Contabilización por grupo

Los grupos existentes son los siguientes: A, B, C, D , E y F. Entonces el arreglo que recibe debe ser de dimensión 6. Al finalizar el conteo de este arreglo se ejecuta una función para imprimir en pantalla el número de alumnos en cada grupo.

5.2. Contabilización por turno

Los turnos existentes son matutino (M) y vespertino (V). Entonces el arreglo que recibe debe ser de dimensión 2. Al finalizar el conteo de este arreglo se ejecuta una función para imprimir en pantalla el número de alumnos en cada turno.

6. Actualización de estudiantes

La actualización de estudiantes abarca dos sistemas, la baja y alta de un estudiante. Estas funciones hacen uso de la variable que guarda la cantidad total de estudiantes que hay en la base de datos, ya que las dos funciones modificaran indirectamente esta variable. Estos sistemas

desconocen si los datos se encuentran ordenados, entonces estos realizarán movimientos en las posiciones de los estudiantes. Las funciones se encuentran en el archivo update.h. Los datos de los estudiantes eliminados o añadidos se almacenan dentro del programa. Para ver reflejados estos datos en el archivo dado se tiene que ejecutar la función descrita en la sección 3.

6.1. Baja de estudiantes

La baja de un estudiante se inicia preguntando el ID del mismo. El ID es la posición en el arreglo de los estudiantes. Una manera de obtener el ID del estudiante al que se dara de baja es ubicar al estudiante en el archivo de datos y su número de fila es el ID del estudiante.

Como pueden suceder errores y esta acción no es revertible, entonces se pregunta si se selecciono bien al estudiante, si introduce la variable n, entonces se volverá a preguntar el ID, si se marca cualquier otro carácter, entonces se dara de baja al estudiante.

El mecanismo para dar de baja un estudiante es el siguiente: El estudiante que se encuentre en la última posición del arreglo sobre escribira los datos del estudiante que se quiere dar de baja y el número de estudiantes se reduce en uno. Si el estudiante que se quiere dar se baja es el último en el arreglo se omite la sobreescritura. El tamaño de memoria se queda intacto, esto es porque pueden suceder errores al modificar el arreglo a un espacio menor al asignado anteriormente. La función que contiene a este mecanismo es delete_student.

6.2. Alta de estudiantes

La alta de estudiantes se inicia aumentando la memoria para poder almancenar a un estudiante más, esto es usando la función realloc. En seguida se inicializa su estructura y se leen los datos del estudiante por medio de la terminal. Al termino de la lectura se aumenta en uno la variable del número de los estudiantes. El mecanismo para dar de alta a un estudiante se encuentra en la función add_studen.

7. Inversión de los datos

Para la inversión de los datos se conto con el siguiente algoritmo:

```
1    for (int i = 0; i < size / 2; i++)
2    {
3        aux = students[i];
4        students[i] = students[size - i - 1];
5        students[size - i - 1] = aux;
6    }
```

En la linea 1, se ejecuta un ciclo que corre hasta la mitad de los elementos, esto es porque los elementos de los extremos intercambiaran su posición. Esta acción es realizada por la ayuda de una estructura auxiliar que guardará los valores de un estudiante y no sean perdidos en el proceso. Si el número de estudiantes es impar, entonces el estudiante que se encuentre en la posición central no cambiara de posición. Este mecanismo se encuentra en el archivo inverse.h en la función inverse_file. Los cambios realizados no se veran reflejados en un archivo, para obtener este resultado es necesario ejecutar la función expuesta en la sección 3.