Universidad Autónoma de Baja California

Facultad de Ingeniería Arquitectura y Diseño

Ingeniero en computación

Ingeniero en Software y tecnologías emergentes

Materia: Programación Estructurada / Clave **36276**

Alumno: BRAYAN IVAN PEREZ VENTURA

Matrícula: 372781

Maestro: Pedro Núñez Yépiz

Actividad No. : 14

Tema - Unidad : ARCHIVOS INDEXADOS

Ensenada Baja California a 28 de Noviembre del 2023

1. INTRODUCCIÓN

Para la presente práctica, se crea un menú interactivo donde existirán varias opciones el cual, el usuario podrá navegar e interactuar generando archivos y cargando archivos binarios, así mismo, considerando el porqué son necesarios backups.

2. COMPETENCIA

Se evaluará la capacidad del estudiante para realizar los problemas presentados en el procedimiento, siguiendo las reglas de la programación estructurada y su correcta implementación en cada una de estas, así mismo, también la capacidad de las funciones creadas dentro de los archivos.

3. FUNDAMENTOS

TEORÍA TOMADA DEL MANUAL OFICIAL DE PRÁCTICAS

4. PROCEDIMIENTO

Universidad Autónoma de Baja California

Facultad de Ingeniería Arquitectura y Diseño

INSTRUCCIONES: Programa que contenga el menú anterior, el programa utiliza un vector de índices de la siguiente estructura: **[llave, índice]** donde *el campo llave es* **noempleado.**
registros.dat es el archivo con los registros a cargar en el vector de índices **archivo binario sera proporcionado**,

CARGAR ARCHIVO : El programa deberá cargar al arrancar el programa, el archivo Binario generará el vector de índices (**llave, indice**) sólo con registros válidos **(el tamaño del vector debera ser 25% mas grande que el la cantidad de registros que contenga el archivo binario) utiliza un archivo externo para averiguar tamaño y retorne cantidad de registros.**

1.- Agregar :
El programa deberá ser capaz de agregar un registro al arreglo de índices y al final del archivo Binario. (agregar forma automatica no repetido el campo llave)

2.- Eliminar :

- El programa deberá buscar una **noempleado** en el vector de índices por medio del método de búsqueda más óptimo.
- La función deberá *retornar*, el índice donde se encuentra la matrícula en el archivo Binario, utilizar banderas para escoger el método más adecuado.
- Una vez obtenido el índice moverse dentro del archivo binario (usar fseek) usando el índice del vector de índices.
- Leer el registro en la posición correcta, preguntar si se quiere eliminar registro.
- Cambiar el status del registro si la respuesta es afirmativa, volver a posición anterior y sobreescribir el registro.

3.- Buscar :

- El programa deberá buscar un **noempleado** en el vector de índices por medio del método de búsqueda más óptimo.
- La función deberá *retornar*, el índice donde se encuentra la matrícula en el archivo Binario, utilizar banderas para escoger el método más adecuado.
- Una vez obtenido el índice moverse dentro del archivo binario (usar fseek) usando el índice del vector de índices.
- Leer el registro en la posición correcta, y desplegar el registro.

4.- Ordenar :

El programa deberá ordenar el vector de índices por medio del método de ordenación más óptimo. Utilizar banderas para escoger el método más adecuado por el que se ordenará por el campo llave (**noempleado**) o no ordenarse si ya está ordenado. (utilizar 3 metodos de ordenacion diferentes segun sea el caso que se necesite Justificar los metodos en el reporte)

5 Y 6.- Mostrar Todo:

El programa deberá mostrar todos los registros del Archivo Binario, preguntar: **ordenado o normal**. Usar el vector de índices para imprimirlo ordenado, y directamente desde el archivo si es normal.

7.- GENERAR ARCHIVO TEXTO:

El programa deberá generar un archivo de texto, el usuario debe proporcionar el nombre del archivo.
El programa deberá mostrar todos los registros del Archivo Binario, preguntar: **ordenado o normal**. Usar el vector de índices para imprimirlo ordenado, y directamente desde el archivo si es normal.
el programa podrá generar múltiples archivos para comprobar las salidas.

8.- EMPAQUETAR :

El programa deberá actualizar el Archivo Binario, a partir de solo registros válidos, y eliminarlos del archivo binario. Crear copia y archivo de respaldo .bak del archivo de antes de eliminarlos.

5. RESULTADOS Y CONCLUSIONES

Durante la realización del presente código, se comprendió la flexibilidad, optimización y seguridad el usar archivos binarios y el cómo implementarlos en diferentes códigos.

Así mismo, se comprendió complementa la razón por la cual mantener la información con un backup es necesario y el tomar entre decisiones de si menos procesos y más seguridad, o menos seguridad y más procesos.

```
293  int OrderIndex(int position, int flag)
294  {
295      TIndexStrct Index[position];
296      FILE *fa;
297
298      if (flag)
299      {
300          printf("El vector ya ha sido ordenado con anterioridad");
301      }
302      else
303      {
304          fillIndexRegister(Index);
305          bubbleSort(Index, position);
306          fa = fopen("datos_index.dat", "wb");
307          fwrite(Index, sizeof(TWrkr), 1, fa);
308          fclose(fa);
309          printf("El vector ha sido ordenado\n");
310          return 1;
311      }
312      return flag;
313  }
```

6. ANEXOS

ANEXADO EN UN ARCHIVO PDF.

7. REFERENCIAS

Diseño de algoritmos y su codificación en lenguaje C

Corona, M.A. y Ancona, M.A. (2011)..

España: McGraw-Hill.

ISBN: 9786071505712

Programación estructurada a fondo:implementación de algoritmos en C

:Pearson Educación.Sznajdleder, P. A. (2017)..

Buenos Aires,Argentina: Alfaomega

Como programar en C/C++

H.M. Deitel/ P.J. Deitel

Segunda edición

Editorial: Prentice Hall.

ISBN:9688804711

Programación en C.Metodología, estructura de datos y objetos

Joyanes, L. y Zahonero, I. (2001)..

España:McGraw-Hill.

ISBN: 8448130138