Documentación del algoritmo

Estamos presentando el algoritmo/código de una problemática planteada de una empresa que necesita llevar registro de sus empleados, exempleados y todo lo radica en torno a ellos, nos manejamos a través de una información que fue prevista de la misma, la cual especificaba que necesitaban un programa que pudiese a través de módulos (véase ingreso, consulta, modificación y eliminado) de sus empleados.

Nos están pidiendo que registremos sus cambios en múltiples archivos convergentes el cuál nosotros decidimos manejar a través de archivos en formato .txt para su practicidad el cuál decidimos nombrar:

1.trabajadoresprincipal.txt para nuestro archivo principal

2.extrabajadores.txt para almacenar eliminados de la empresa

3.cargos.txt para almacenar trabajadores según su cargo

4.departamentos.txt para almacenar trabajadores según su departamento

Todos a excepción de (extrabajadores.txt) se interpretan a través de formato “w”, “w+” “r” “r+”, y su excepción se firma en formato “a+” debido a qué no se guarda registros en ninguna otra parte de la memoria del código para ellos.

El código ofrece:

1.Modularización de todas las áreas requeridas sumado a la intra-modularización de las mismas

2.Manejo de registros/estructuras de llamadas Empleado el cuál contienen la información requerida en la petición (ci, nombre, departamento, cargo, fecha de ingreso, salario); Empleado.ci, Empleado.nombre, etc.

3.Utilización de arreglos unidimensional limitados a 257 espacios el cuál cada uno guarda el registro a modo de una variable llamada igual al registro pero en plural; *Empleados[i].ci, Empleados[i].nombre*, etc.

4.Uso de múltiples librerías que ofrece el lenguaje de programación C para convenir al programa a ejecutar.

Funciona de manera qué:

1. El código llena el arreglos de registros unidimensionales desde un procedimiento llamado Cargar\_Empleados, el cuál no hace más que leer el archivo y almacenar cada línea limitada por 100 caracteres según el código en una casilla del arreglo a referir.

2.El código trabaja mediante su uso de arreglos, tanto para listar a sus empleados por cualquiera de sus variantes a requerir como para modificar o eliminar a los mismos.

3.Este mismo, utiliza también otros arreglos de registros auxiliares para poder modificar de manera más sencilla X casilla a tratar, y así evitando que represente un problema el hecho de sobreescribir directamente sí ocurriese un problema durante la ejecución del archivo.

4.Sus funciones de retorno entero o booleano son aprovechadas según su retorno para hacer la experiencia del usuario más satisfactoria según su necesidad.

5.El código válida sus entradas hasta dónde el conocimiento adquirido en clase lo permite, es decir este código es capaz de verificar si un número de un menú de opciones fue mal escrito, si una fecha no existe o sí X término llega a ser inválido reportandolo a su main

6.El código tiene como objetivo hacer una experiencia buena al usuario y no tediosa, evitando a toda costa cualquiera de sus bugs y limpiando constantemente la pantalla para evitar el cúmulo de texto innecesario, utilizando a su vez términos como el \n para evitar un texto sucio.

7.Usa variables y nombres de sus procedimientos claros para evitar confusiones sí alguien más desea leer el código.

8.A su vez, el código está intradocumentado para evitar confusión a otros programadores sí desea revisar o modificar alguna cuestión

9.Usa constantes para definir términos utilizados de manera consecuente, para hacer más fácil sus modificaciones.

10.El código, a pesar de ser un poco largo (poco menos de 800 líneas) llega a ser claro y conciso con lo que hace.

Problemas que pueden ocurrir y limitaciones:

1.La mayor limitación de este código es qué utiliza un array estático para su memoria a utilizar, esto debido a la prohibición de uso de ejecuciones no antes vistas en clase como malloc y calloc, o cualquier modificación a la memoria dinámica. Por lo tanto tiene un límite a almacenar, el cuál volvemos a mencionar que es de 257 casillas.

2.El código no tiene manera de validar si un int o un long int es realmente uno, es decir sí se escribiese un carácter un término no asignado ocasionaría problemas

3.El código en su primera apertura suele tardar algo de tiempo en ejecutarse, no alertarse pues es algo normal.

4.El código solo ha sido probado una vez en Ubuntu Linux, el cuál en esa ocasión funciono como se esperaba, sin embargo este código fue hecho a través de Windows en VSCode y Code::Blocks, en dónde a su vez se hicieron sus últimos bugs testing antes de ser enviado.