Ingeniero en computación

Ingeniero en Software y tecnologías emergentes

Materia: Programación Estructurada / Clave **36276**

Alumno: Hernandez Ceseña Ivan Fernando

Matrícula: 373077

Maestro: Pedro Núñez Yépiz

Actividad No. : 14

Tema - Unidad : Archivos Binarios (archivos indexados)

Ensenada Baja California a 28 de noviembre del 2023

1. INTRODUCCIÓN

Se van a usar funciones para tener un programa con un operador múltiple (switch) y poder acceder por medio de este a la acción que se quiere ejecutar. acompañados de ciclos para automatizar selecciones y entradas de datos. Utilizando vectores y matrices.

Se implementa el uso de una librería (.h) hecha por el alumno. En ella se encontrarán las funciones que se vayan a usar durante el programa.

También se implementa el método de búsqueda para poder llenar vectores y matrices sin repetir valores.

El otro método que se implementa es el de ordenamiento en base al método de la burbuja, pero implementando más funciones dentro del bloque de código de la función.

Se usaron funciones como el fprintf y el sscanf para poder modificar y leer archivos de texto.

Se usaron funciones como el fwrite para poder editar nuestro archivo binario.

se utilizaron archivos indexados para manipular los datos del archivo binario original.

2. COMPETENCIA

usar funciones para hacer un código más claro fácil de entender, además de poner en práctica la optimización de código. utilizando vectores, matrices, cadenas y archivos de texto, binarios e indexados.

3. FUNDAMENTOS

- sintaxis de ciclos.
- funcionamiento de ciclos.
- proceso de ciclos.
- movilización de una función a otra.
- funcionamiento de vectores y matrices y cómo utilizarlas correctamente.
- declaraciones de vectores y matrices.
- metodos de ordenacion y busqueda
- uso de librerías propias
- archivos de texto.
- archivos binarios
- punteros
- archivos indexados

4. PROCEDIMIENTO

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

en este apartado se declaran prototipos de funciones que vamos a usar para el programa

```c
typedef int Tkey;

typedef struct _work{
    int status;
    Tkey key;
    char nombre[30];
    char apPat[50];
    char apMat[50];
    char sexo[15];
    char puesto[30];
    char estado[30];
    int edad;
    Tkey celuar;
} Twrkr;

typedef struct index{
    Tkey llave;
    int  consecutivo;
} Tindex;

void menu(int n);
int cargar(Twrkr reg[], Tindex reg2[], int i);
int getIndexFile(Tindex reg[], int n);
int fillIndexRegister(/*Tindex index[]*/Twrkr reg[]);
void agregar(Twrkr reg[], Tindex reg2[], int posicion, int n);
void eliminar(Twrkr registro[], Tindex reg2[], int posicion, int ordenar);
void buscar(Twrkr registro[], Tindex reg2[], int posicion, int ordenar);
void ordenar(Tindex reg[], int posicion , int ordenado, int preordenado);
void imprimirOriginal(Tindex reg[], int posicion);
void imprimirOrdenado(Tindex reg[], int posicion, int ordenado, int preordenado);
void archivoTxt(Tindex reg[], int posicion, int ordenado, int preordenado);
void empaquetado();


void imprimir(Twrkr reg[], int posicion);
void imprimir1(Twrkr reg[], int posicion);
void imprimirReg2(Tindex reg[], int posicion);
void imprimirReg2_1(Tindex reg[], int posicion);

void sexo(Twrkr reg[], int i, int li, int ls);
void status(Twrkr reg[], int i);
void matricula(Twrkr reg[], Tindex reg2[], int li, int ls, int posicion);
void apellidoPaterno(Twrkr reg[], int i);
void apellidoMaterno(Twrkr reg[], int i);
void nombre(Twrkr reg[], int i);
void edad(Twrkr reg[], int i, int li, int ls);
void state(Twrkr reg[], int posicion, int li, int ls);
void jobPossition(Twrkr reg[], int posicion, int li, int ls);
int existeValor(Twrkr reg[], int n, int numero);
int existeValor2(Tindex reg2[], int n, int numero);
int busquedaBinaria(Tindex reg2[], int izquierda, int derecha);
void ordenarVectorAscendente(Tindex reg[], int n);
void cambiar(Tindex reg[], int i, int j);
int particion(Tindex reg[], int menor, int mayor);
void quicksort(Tindex reg[], int menor, int mayor);
void merge(Tindex arr[], int l, int m, int r);
void mergeSort(Tindex arr[], int l, int r);
```

menu (controla todo el programa)

```c
void menu(int n)
{
    // printf("%d\n", n);
    Twrkr reg[n];
    Tindex reg2[n];
    int posicion = 0;
    int op;
    int inc = 1;
    int ordenado = 0;
    int preordenado = 0;

    posicion = getIndexFile(reg2, n);
    //cargar(reg, reg2, posicion);//
    // imprimir(reg, posicion);
    // imprimirReg2(reg2, posicion);

    do
    {
        msg();
        op = validacionNumero("elije una opcion", 1, 9);

        switch(op)
        {
            case 1:
            // fillIndexRegister(/*reg2*/reg);

            if(posicion + inc > n)
            {
                inc = n - posicion;
            }
            if(inc == 0)
            {
                printf("registro lleno\n");
            }
            else
            {
                agregar(reg, reg2, posicion, inc);
                // fillIndexRegister(/*reg2*/reg);
                // imprimir1(reg, posicion);
                // // imprimirReg2_1(reg2, posicion);
                posicion = posicion + inc;
                ordenado = 0;
                preordenado++;
                if(preordenado > 5)
                {
                    ordenado = 0;
                }
            }
            break;

            case 2:
            eliminar(reg, reg2, posicion, ordenado);
            break;

            case 3:
            buscar(reg, reg2, posicion, ordenado);
            break;

            case 4:
            ordenar(reg2, posicion, ordenado, preordenado);
            ordenado = 1;
            break;

            case 5:
            imprimirOriginal(reg2, posicion);
            break;

            case 6:
            imprimirOrdenado(reg2, posicion, ordenado, preordenado);
            break;

            case 7:
            archivoTxt(reg2, posicion, ordenado, preordenado);
            break;

            case 8:
            empaquetado();
            break;
        }
    } while (op != 9);
}
```

apartado en el que se cargan los datos del archivo binario previamente proporcionado

```c
int main()
{
    int contador =0; ///////
    char archivo[30]= "datos";
    char parametros[30];

    system("gcc.exe archivoExterno.c -o archivoExterno");
    sprintf(parametros, "archivoExterno.exe %s", archivo);
    contador = system(parametros);
    contador = contador * 1.25;
    menu(contador);
    return 0;
}

void msg()
{
    printf("MENU\n");
    printf("1) agregar\n");
    printf("2) eliminar\n");
    printf("3) buscar\n");
    printf("4) ordenar\n");
    printf("5) imprimir registro archivo original\n");
    printf("6) imprimir registro archivo ordenado\n");
    printf("7) generar archivo de texto\n");
    printf("8) empaquetar\n");
    printf("9) salir\n");
}

void menu(int n)
{
    // printf("%d\n", n);
    Twrkr reg[n];
    Tindex reg2[n];
    int posicion = 0;
    int op;
    int inc = 1;
    int ordenado = 0;
    int preordenado = 0;

    posicion = getIndexFile(reg2, n);
    //cargar(reg, reg2, posicion);//
    // imprimir(reg, posicion);
    // imprimirReg2(reg2, posicion);
```

Todas las funciones interactúan con archivos binarios indexados.

El uso de una librería facilita mucho el entendimiento claro del código, también ayuda a que no se vea muy abultado. Esto nos ayuda a corregir errores de manera más rápida.

Los métodos de ordenamiento y búsqueda también son una herramienta bastante útil a la hora de usar funciones en las que se necesitan valores en órdenes específicos o que no se requieren valores repetidos.

El separar acciones específicas también ayuda a poder usarlas de mejor manera dentro de otros bloques de código y tener códigos más limpios y fáciles de entender.

6. ANEXOS

anexos en el otro archivo:

nombre del archivo: anexo_HCIF_RP14_PE

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