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. : 13

Tema - Unidad : Archivos binarios y de texto

Ensenada Baja California a 19 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.

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 y binarios.

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

4. PROCEDIMIENTO

1.- Agregar : El programa deberá ser capaz de agregar 100 registros al vector de registros (Generar automáticamente los datos).

2.- Editar Registro : El programa deberá buscar una matrícula en el vector por medio del método de búsqueda más óptimo. Mostrar los datos en forma de registro Preguntar que campo quire Editar, actualiar los datos en el vector (solo a registros activos)

3.- Eliminar Registro : El programa deberá buscar una matrícula en el vector por medio del método de búsqueda más óptimo. Utilizar banderas para escoger el método más adecuado., imprimir el registro y preguntar si se quiere eliminar el registro.

4.- Buscar : El programa deberá buscar una matrícula en el vector por medio del método de búsqueda más óptimo. Utilizar banderas para escoger el método más adecuado. Mostrar los datos en forma de registro

5.- Ordenar : El programa deberá ordenar el vector por medio del método de ordenación más óptimo. Utilizar banderas para escoger el método más adecuado se ordenará por el campo llave (matrícula)

6.- Imprimir: El programa deberá mostrar todos los registros del vector y como están en ese momento ordenado o desordenado. (mostrar en forma de tabla)

7.- Generar Archivo Texto : El programa deberá preguntar al usuario el nombre del archivo, solo nombre sin extensión, el programa generará un archivo con el nombre proporcionado por el usuario con extensión .txt los datos que pondrá en el archivo de texto serán idénticos a los contenidos en el Vector de registros. (ordenado o desordenado). El programa podrá generar múltiples archivos para comprobar las salidas.

8.- Mostrar Archivo Texto: El programa deberá preguntar al usuario el nombre del archivo, solo nombre sin extensión, el programa generará un archivo con el nombre proporcionado por el usuario con extensión .txt mostrar el archivo de texto tal y como se encuantra.

9.- Crear archivo binario : El programa deberá crear un archivo binario con los datos del vector actualizados, sustituir el archivo base, realizar respaldo del archivo anterior y guardarlo con el mismo nombre pero extencion .tmp (validar msges si el archivo no se puede crear por falta de registros en el vector)

10 .- Cargar Archivo Binario : El programa deberá cargar al vector los registros del archivo binario (solo podrá cargarse una sola vez el archivo, el archivo binario se debara llamar datos.dll y si no existe debera indicar)

11.- Mostrar Borrados: El programa deberá mostrar del archivo binario solo los registros que se eliminaron (marcados con status 0) y que fueron marcados en su momento como registros eliminados.

5. RESULTADOS Y CONCLUSIONES

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

```
typedef int Tkey;

typedef struct _alumnos{
    int status;
    Tkey key;
    char apellidoPaterno[30];
    char apellidoMaterno[30];
    char nombre[30];
    int edad;
    int sexo; } Talumns;


void menu(void);
void msg(void);

void registroAleatorio(Talumns reg[], int i, int numero);
void sexo(Talumns reg[], int i, int li, int ls);
void status(Talumns reg[], int i);
void matricula(Talumns reg[], int li, int ls, int i);
void apellidoPaterno(Talumns reg[], int i);
void apellidoMaterno(Talumns reg[], int i);
void nombre(Talumns reg[], int i);
void edad(Talumns reg[], int i, int li, int ls);
int existeValor(Talumns reg[], int n, int numero);
void eliminarRegistro(Talumns reg[], int posicion);
void buscarElemento(Talumns reg[], int posicion, int bandera);
void ordenarAscendente(Talumns reg[], int n);
int busquedaBinaria(Talumns reg[], int izquierda, int derecha);
void imprimirRegistro(Talumns reg[], int posicion);
void imprimirBuscado(Talumns reg[], int posicion);

void cambiar(Talumns reg[], int i, int j);
int particion(Talumns reg[], int menor, int mayor);
void quicksort(Talumns reg[], int menor, int mayor);
void archivoTexto(Talumns reg[], int posicion);
void numeroRegistros(void);
void generarTxt(Talumns reg[], int posicion);
int cargarArchivo(Talumns reg[], int posicion, char archivo[]);
void inactivos(Talumns reg[], int posicion);
void editarRegistro(Talumns reg[], int posicion, int bandera);

void mostrarArchivoTxt(int posicion, char archivo[]);
void imprimirRegistroTxt(Talumns reg[], int i, int posicion);
void crearArchivoBinario(Talumns reg[], int posicion);
int cargarArchivoBinario(Talumns reg[], int posicion, int maximo);
```

menu (controla todo el programa)

```c
void menu()
{
    int op;
    int posicion=0, bandera=0, cargar = 0;
    Talumns reg[N];
    char nombretxt[30];
    int numero = 900;


    do{
        system("cls");
        msg();
        op= validacionNumero("elije una opcion", 1, 12);
        switch(op)
        {
            case 1:
            if((posicion + numero) > N)
            {
                numero = N - posicion;
            }
            if(numero == 0)
            {
                printf("el registro esta lleno\n");
            }
            else
            {
                registroAleatorio(reg, posicion, numero);
                printf("llenado de 100 registros exitoso\n");
                posicion= posicion + numero;
                bandera = 0;
            }
            break;

            case 2:
            editarRegistro(reg, posicion, bandera);
            break;

            case 3:
            eliminarRegistro(reg, posicion);
            break;

            case 4:
            buscarElemento(reg, posicion, bandera);
            break;

            case 5:
            if(posicion > 500)
            {
                printf("quicksort\n");
                quicksort(reg, 0, posicion-1);
            }
            else
            {
                printf("bubblesort\n");
                ordenarAscendente(reg, posicion);
            }
            bandera = 1;
            break;

            case 6:
            printf("registro\n");
            imprimirRegistro(reg, posicion);
            break;

            case 7:
            generarTxt(reg, posicion);
            break;
```

menu que controla todo el programa

```
            {
                printf("el registro esta lleno\n");
            }
            else
            {
                registroAleatorio(reg, posicion, numero);
                printf("llenado de 10 registros exitoso\n");
                posicion= posicion + numero;
                bandera = 0;
            }
            break;

            case 3:
            eliminarRegistro(reg, posicion);
            break;

            case 4:
            buscarElemento(reg, posicion, bandera);
            break;

            case 5:
            if(posicion > 500)
            {
                printf("quicksort\n");
                quicksort(reg, 0, posicion-1);
            }
            else
            {
                printf("bubblesort\n");
                ordenarAscendente(reg, posicion);
            }
            bandera = 1;
            break;

            case 6:
            printf("registro\n");
            imprimirRegistro(reg, posicion);
            break;

            case 7:
            generarTxt(reg, posicion);
            break;

            case 8:
            numeroRegistros();
            break;

            case 9:
            inactivos(reg, posicion);
            break;
        }
        system("pause");
    }while(op != 10);
}
```

```
            case 7:
            generarTxt(reg, posicion);
            break;

            case 8:
            validacionArchivo("dame el nombre del archivo que quieres leer", nombretxt, 30);
            mostrarArchivoTxt(posicion, nombretxt);
            break;

            case 9:
            crearArchivoBinario(reg, posicion);
            bandera = 0;
            break;

            case 10:
            if(cargar == 1)
            {
                printf("el archivo binario solo se puede cargar una vez");
            }
            else
            {
                posicion = cargarArchivoBinario(reg, posicion, N);
                cargar =1;
            }
            break;

            case 11:
            inactivos(reg, posicion);
            break;
        }
        system("pause");
    }while(op != 12);
}
```

funciones que interactúan con archivos txt

```c
void generarTxt(Talumns reg[], int posicion)
{
    int i;
    FILE *fa;
    char archivo[30];

    validacionArchivo("escribe como quieres nombrar tu archivo", archivo, 30);

    strcat(archivo, ".txt");
    fa = fopen(archivo, "w");

    for(i =0; i<posicion; i++)
    {
        if(reg[i].status == 1)
        {
            fprintf(fa, "%6d.- %10d %10s %15s %15s %5d", i+1, reg[i].key, reg[i].nombre, reg[i].apellidoPa
            if(reg[i].sexo == 1)
            {
                fprintf(fa, " %8s\n", "HOMBRE");
            }
            else
            {
                fprintf(fa, " %8s\n", "MUJER");
            }
        }
    }

    fclose(fa);
    printf("archivo generado\n");
}
```

```c
void mostrarArchivoTxt(int posicion, char archivo[])
{
    FILE *fa;
    Talumns registro;
    int num;
    char sex[10];
    strcat(archivo, ".txt");
    fa = fopen(archivo, "r");
    if(fa)
    {
        while(!feof(fa))
        {
            fscanf(fa, "%d.- %d %s %s %s %d %s", &num, &registro.key, registro.nombre, registro.apellidoPa
            registro.status = 1;
            if(strcmp(sex, "HOMBRE") == 0)
            {
                registro.sexo = 1;
            }
            else
            {
                if(strcmp(sex, "MASCULINO") == 0)
                {
                    registro.sexo = 1;
                }
                else
                {
                    if(strcmp(sex, "MUJER") == 0)
                    {
                        registro.sexo = 2;
                    }
                    else
                    {
                        if(strcmp(sex, "FEMENINO") == 0)
                        {
                            registro.sexo = 2;
                        }
                    }
                }
            }
            printf("%3d %10d %20s %20s %20s %4d", num, registro.key, registro.nombre, registro.apellidoPate
            if(registro.sexo == 1)
            {
                printf(" %8s\n", "HOMBRE");
            }
            else
            {
                printf(" %8s\n", "MUJER");
            }
            posicion++;
        }
    }
    else
    {
        printf("el archivo no existe\n");
    }
}
```

funciones que interactuan con archivos binarios

```c
void crearArchivoBinario(Talumns reg[], int posicion)
{
    Talumns registro;
    int i;
    char nombre[30];
    FILE *fa;
    validacionArchivo("dame el nombre para tu archivo binario", nombre, 30);
    strcat(nombre, ".dll");
    fa = fopen(nombre, "ab");

    for(i=0; i<posicion; i++)
    {
        registro = reg[i];
        fwrite(&registro, sizeof(Talumns), 1, fa);
    }
    fclose(fa);
}

int cargarArchivoBinario(Talumns reg[], int posicion, int maximo)
{
    int i=0;
    Talumns registro;
    FILE *fa;
    char nombre[30];
    validacionArchivo("dame el nombre del archivo binario", nombre, 30);
    strcat(nombre, ".dll");
    fa = fopen(nombre, "rb");

    if(fa)
    {
        while(fread(&registro, sizeof(Talumns), 1, fa))
        {
            if(posicion + 1 > maximo)
            {
                printf("no se pudo cargar todo el archivo.\nsolo se cargaron %d registros\n", i);
                return posicion;
            }
            else
            {
                reg[posicion++] = registro;
                i++;
            }
        }
        fclose(fa);
    }
    return posicion;
}
```

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_RP13_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