Universidad Autónoma de Baja California

Facultad de Ingeniería, Arquitectura y Diseño F.I.A.D.

Campus Ensenada

Ramón Mejía Martínez 372099

Grupo: 932

Actividad 13

Programación Estructurada

Archivos de acceso secuencial y aleatorio

Ensenada, Baja California a 22 de noviembre de 2023.

Universidad Autónoma de Baja California

Facultad de Ingeniería Arquitectura y Diseño

Ingeniero en Software y tecnologías emergentes

Materia: Programación Estructurada / Clave **36276**

Alumno: Ramón Mejía Martínez

Matrícula: 372099

Maestro: Pedro Núñez Yépiz

Actividad No. : 13

Tema - Unidad : Archivos de acceso secuencial y aleatorio

Ensenada Baja California a 19 de noviembre del 2023.

1. Introducción

En esta actividad se realizara un sistema de gestión de empleados usando structs, archivos de texto y binarios.

2. Competencia

Esta actividad tiene como objetivo el manejo cadenas, librerías y funciones, structs, métodos de ordenación y búsqueda, uso de archivos para guardar y cargar registros, esto con el fin de agregar herramientas que podrán ser usadas nuevamente en otros proyectos.

3. Fundamentos

Archivos

Archivo. Es un conjunto de información almacenada bajo un nombre. Cada archivo es almacenado en un dispositivo de almacenamiento, usando un único nombre. El dispositivo puede ser: el disco floppy, el disco duro, cd-rom, etc. La librería contiene las funciones para procesar archivos.

Comandos para manjar archivos.

La función remove elimina el archivo con el nombre que se le mande como argumento.

Remove(“nombre”);

La función rename cambia el nombre de un archivo ya existente.

Rename(“nombre_antiguo”,”nombre_nuevo”);

Estos son los que son usados para esta actividad.

Archivos de texto

Operaciones:

Los modos de operación son: r, w y a, a los cuales se les puede agregar un +.

- El modo de operación a se usa para leer y escribir, devuelve la dirección de memoria del archivo si existe, si no existe se crea. Coloca el cursor en la siguiente posición del último carácter del archivo para continuar escribiendo en el archivo.

- w se usa para escribir, devuelve la dirección de memoria del archivo si existe, si no existe se crea, se usa w+ para poder leer y escribir. Coloca el cursor al inicio del texto y reemplaza los datos del archivo si ya existía.

- r se usa para lectura, devuelve la dirección de memoria del archivo si se encontró o un valor nulo si no existe, se usa r+ para poder leer y escribir del archivo. El puntero al abrir un archivo de esta forma se encuentra en el inicio del archivo.

- Abrir.

Para abrir un archivo binario se define un puntero con tipo de dato FILE y se usa la instrucción fopen.

FILE *fa;

fa=fopen(“nombre.extension”,”modo_de_operacion”);

- Leer.

Para leer datos de un archivo de texto se usa fscanf.

scanf(puntero_a_archivo,”datos extraidos”,variable);

Leyendo y guardando en variables, si solo se desea mostrar el archivo se puede guardar todo un una sola variable usando fgets y mostrar carácter por carácter hasta llegar al final del archivo.

- Escribir.

Para escribir en un archivo de texto se usa fprintf.

fprintf(puntero_a_archivo,”texto_o_tipoDeDato”,variable);

Archivos binarios

Operaciones:

Las operaciones para los archivos binarios son iguales a las de los archivos de texto y se abren de la misma manera con la excepción de que se tiene que indicar al lado del modo una b de binario (fa=fopen(“nombre.dll”,”rb”);

- Leer

Para leer de un archivo binario se usa fread.

fread(&variable,sizeof(tipo_dato_variable),numero_de_registros,puntero_a_archivo);

Los datos se guardan en la variable.

Requiere saber el tamaño de un registro del archivo binario para poder dividir correctamente la información.

Normalmente de leen de 1 en 1 los registros.

- Escribir

Para escribir en un archivo binario se usa fwrite.

fwrite((&variable,sizeof(tipo_dato_variable),numero_de_registros,puntero_a_archivo);

Usa los mismos argumentos, la diferencia es que ahora se escribe lo de la variable en el archivo.

Otras funciones útiles son feof(puntero_a_archivo) que indica cuando se llega al final del archivo.

Al terminar de trabajar con los archivos se deben cerrar para mantener la integridad de los datos, esto se hace usando fclose(puntero_a_archivo).

4. Procedimiento

ACTIVIDAD 13

REALICE EL SIGUIENTE PROGRAMA QUE CONTENGA UN MENÚ.

MENÚ

1.- AGREGAR (AUTOM 100 REGISTROS)
2.- EDITAR REGISTRO
3.- ELIMINAR REGISTRO (lógico)
4.- BUSCAR
5.- ORDENAR
6- IMPRIMIR
7.- GENERAR ARCHIVO TEXTO
8.- VER ARCHIVO TEXTO
9.- CREAR ARCH BINARIO
10.- CARGAR ARCH BINARIO
11.- MOSTRAR ELIMINADOS
0.- SALIR

UTILIZAR UN ARREGLO DE 5000 REGISTROS
SE DEBERÁ UTILIZAR ESTRUCTURAS CON LOS DATOS BÁSICOS DE UN EMPLEADO
preguntar nombre de archivo binario o de archivo texto
Busqueda y Ordenacion por CAMPO LLAVE

nota: usar librería propia con funciones
nota2: 100 % validado, Cuidar desbordamiento de vector
nota3: Campo llave matricula no repetido, archivos solo cargar 1 sola vez.
nota4: Usar el tipo Tkey para hacer mas practico el programa

INSTRUCCIONES DEL MENU

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:

Structs y constantes:

```
#define NM 4
#define STR 100

#define L 30
#define LN 50

//** Structs    ***************************


typedef int Tkey;

typedef struct _empleado{
    Tkey key;
    int NoEmpleado;
    char nombre[LN];
    char apPat[L];
    char apMat[L];
    char sexo;
    int status;
    char puesto[L];
    double telefono;
}Tdato;
```

Main:

La función principal se usó para llamar al menú del programa y poner la semilla para los valores aleatorios.

```
int main()
{
    srand(time(NULL));
    menu();
    return 0;
}
```

Función menu:

- Se elimina los registros binarios de ejecuciones pasadas, se pregunta al usuario la opción que desea y se ejecutan las llamadas a las funciones correspondientes.

```
void menu(void)
{
    FILE *fa;
    Tdato empleados[REG];
    int op;
    int orden=0,ordenMetodo=0,cargBin=1,del=0;
    int i=0;
    fa=fopen("registrosbin.dll","rb");
    if(fa)
    {
        fclose(fa);
        remove("registrosbin.dll");
    }

    do
    {
        system("cls");
        op=msgs();
        system("cls");
        switch(op)
        {
            case 1:
```

Caso 1:

- Se llama a la función genReg y se cambian las banderas si es necesario.

```
case 1:
    i=genReg(empleados,i);
    if(i>2500)
    {
        ordenMetodo=1;
    }
    orden=0;
    break;
```

Función genReg:

- Genera registros aleatorios.
- Se genera la cantidad según el numero de registros actuales comparando con el limte para no superarlo.
- Recibe todos los registros y el número de registros existentes.
- Devuelve la cantidad de empleados que se agregaron.

```
//case 1
int genReg(Tdato vect[],int i)
{
    int j;
    if(i+100<=REG)
    {
        for(j=0;j<100;j++)
        {
            genRegRand(vect,i++);
        }
        printf("\nSe generaron 100 registros\n");
    }
    else
    {
        if(i==REG)…
        else…
    }
    return i;
}
```

Función genRegRand:

- Genera asignaciones aleatorias según el campo.
- Recibe todos los registros y el número de registros existentes.

```
void genRegRand(Tdato vect[],int i)
{
    int s;
    int j;
    double num;
    vect[i].status=1;
    vect[i].NoEmpleado=genKey(vect,i,300000,399999);
    vect[i].key=vect[i].NoEmpleado;

    for(j=0;j<10;j++)
    {
        s=rand()%2;
    }

    if(s)
    {
        vect[i].sexo='H';
    }
    else…

    if(s)
    {
        genH_Name(vect,i);
    }
    else…

    genAp(vect,i);

    do
    {
        num=6460000000+(rand()%100000)+(rand()%99)*100000;
    }while(buscarTelefono(vect,i,num)!=-1);
    vect[i].telefono=num;
```

Función genH_Name:

Función genM_Name:

Función genAp:

Función genPuesto:

Estas tres funciones trabajan de la misma manera.

- Generan un nombre, apellido o puesto de manera aleatoria, según corresponda.
- Recibe los registros y el número de registros.
- Copia el nombre del índice aleatoria de la lista y lo asigna al campo correspondiente.

```
void genH_Name(Tdato vect[],int i)
{
    char h_name[100][LN]={···
    int prob;
    prob=rand()%10;
    if(prob<=7)
    {
        strcopy(vect[i].nombre,h_name[rand()%100]);
    }
    else
    {
        strcopy(vect[i].nombre,h_name[rand()%100]);
        strConcat(vect[i].nombre,h_name[rand()%100]);
    }
}
```

```
void genPuesto(Tdato vect[],int i)
{
    char pst[12][L]={
        "OPERADOR","SUPERVISOR","MANTENIMIENTO","INGEN
        "OPERADOR DE MAQUINARIA", "ALMACENISTA", "JEFE
        "GERENTE DE PRODUCCION", "TECNICO EN SEGURIDAD
    };
    strcopy(vect[i].puesto,pst[(rand()%12)]);
}
```

Función genKey:

- Genera un campo key aleatoria que aún no exista en el registro.
- Recibe los registros y el número de registros.
- Devuelve el campo key generado.

```
int genKey(Tdato vect[],int i,int ri,int rf)
{
    int num;
    int rango=rf-ri;
    do
    {
        num=(rand()%rango+1)+ri;
    }while(buscarTSec(vect,i,num)!=-1);
    return num;
}
```

Caso 2:

Función editarReg:

- Busca un registro con el método más apropiado para editar alguno de sus campos.
- Recibe el vector de registros, el número de registros y la bandera que indica si esta ordenado.

```
void editarReg(Tdato vect[],int i,int orden)
{
    int pos;
    int edit;
    if(i>0)
    {
        printf("Editar empleado\n\n");
        pos=validNum("Ingrese el numero de emplado que desea editar: ",300
        if(orden)
        {
            pos=buscarTBin(vect,i,pos);
            if(pos!=-1)
            {
                if(vect[i].status)
                {
                    printEmpleadoReg(vect[pos]);
                    editMenu(vect,pos);
                }
                else…
            }
            else…
        }
        else…
    }
    else
    {
        printf("\nAun no hay empleados registrados\n");
    }
}
```

Función editMenu:

- Muestra un menú del campo que se desea editar y permite editar el campo.
- Recibe el vector de registros y la posición del registro que se desea editar.

```
void editMenu(Tdato vect[],int pos)
{
    int op;
    int s;
    int j;
    char num[11];
    printf("\nEditar\n\n");
    printf("1) Puesto\n");
    printf("2) Nombre\n");
    printf("3) Apellido Paterno\n");
    printf("4) Apellido Materno\n");
    printf("5) Telefono\n");
    printf("6) Sexo\n");
    printf("7) Salir\n");
    op=validNum("Esoge una opcion: ",1,7);
    system("cls");
    switch(op)
    {
        case 1:
            printf("Editar puesto\n\nIngresa el puesto: ");
            editPuesto(vect,pos);
            printf("\nSe edito el campo deseado\n");
            break;
        case 2:
            printf("Editar nombre\n\nIngresa el nuevo nombre
            validStr(vect[pos].nombre);
            printf("\nSe edito el campo deseado\n");
            break;
```

Caso 3:

Función eliminarEmpleado:

- Cambia el status del empleado deseado a 0.
- Recibe el vector de registros, el número de registros y la bandera que indica si esta ordenado.
- Realiza la búsqueda por el método más optimo.
- Devuelve si se elimino un empelado para acumularlo a una variable que registra el número de empleados eliminados.

```
int eliminarEmpleado(Tdato vect[],int i,int orden)
{
    int pos;
    int del;
    if(i>0)
    {
        printf("Eliminar empleado\n\n");
        pos=validNum("Ingrese el numero de emplado que desea eliminar: ",300
        if(orden)
        {
            pos=buscarTBin(vect,i,pos);
            if(pos!=-1)
            {
                if(vect[pos].status)
                {
                    printEmpleadoReg(vect[pos]);
                    printf("\nDesea eliminar al empleado? (1:Si 2:No)\n");
                    del=validNum("Escoge una opcion: ",1,2);
                    if(del==1)
                    {
                        vect[pos].status=0;
                        return 1;
                    }
                }
                else···
            }
            else···
        }
        else···
    }
    else···
    return 0;
}
```

Caso 4:

Función buscarEmpleado:

- Muestra en forma de registro el empleado buscado si está activo.
- Recibe el vector de registros, el número de registros y la bandera que indica si esta ordenado.
- Realiza la búsqueda por el método más optimo.

```
void buscarEmpleado(Tdato vect[],int i,int orden)
{
    int pos;
    if(i>0)
    {
        printf("Buscar empleado\n\n");
        pos=validNum("Ingrese el numero de empleado que desea buscar: ",30
        if(orden)
        {
            pos=buscarTBin(vect,i,pos);
            if(pos!=-1)
            {
                if(vect[pos].status)
                {
                    printEmpleadoReg(vect[pos]);
                }
                else
```

Caso 5:

```
case 5:
    if(!orden)
    {
        orden=ordenEmpleado(empleados,i,ordenMetodo);
    }
    else…
    break;
```

Función ordenEmpleado:

- Ordena los registros por el campo key con el método mas apropiado.
- Recibe el vector de registros, el número de registros y la bandera que indica que método se debe usar.
- Devuelve 1 para cambiar la bandera que indica que esta ordenado.

```
int ordenEmpleado(Tdato vect[],int i,int ordenM)
{
    if(i>0)
    {
        if(ordenM)
        {
            mergeSortT(vect,0,i-1);
        }
        else
        {
            heapSortT(vect,i);
        }
        printf("\nSe ordenaron los empleados\n");
        return 1;
    }
    else
    {
        printf("\nAun no hay empleado registrados\n");
        return 0;
    }
}
```

Caso 6:

Función printEmpleados:

- Imprime el registro de empleados en forma de tabla si está activo.
- Recibe el vector de registros y la posición del registro que se desea editar.

```
void printEmpleados(Tdato vect[],int i)
{
    int j,k=0;
    if(i>0)
    {
        printf("| No   | No.Empleado |         Puesto         | Apellido Paterno | Ape
        for(j=0;j<i;j++)
        {
            if(vect[j].status)
            {
                printf("| %4d ",k);
                printEmpleadoTab(vect[j]);
                k++;
            }
        }
    }
    else
    {
        printf("\nAun no hay empleado registrados\n");
    }
}
```

Caso 7:

Función genArchivoTxt:

- Genera un archivo de texto con el nombre que desee el usuario.
- Recibe el vector de registros y el número de registros.

```
void genArchivoTxt(Tdato vect[],int i)
{
    FILE *fa;
    int j,k;
    int len;
    char name[L];
    printf("Generar archivo\n\n");
    if(i>0)
    {
        printf("Ingresa el nombre del archivo de texto: ");
        gets(name);
        len=strlen(name);
        name[len++]='.';
        name[len++]='t';
        name[len++]='x';
        name[len++]='t';
        name[len]='\0';

        fa=fopen(name,"r");
        if(!fa)
        {
            fa=fopen(name,"a");
```

```
if(!fa)
{
    fa=fopen(name,"a");
    fprintf(fa,"| No   | No.Empleado |         Puesto
    k=0;
    for(j=0;j<i;j++)
    {
        if(vect[j].status)
        {
            fprintf(fa,"| %4d ",k);
            fprintf(fa,"| %11d | %-22s | %-16s | %-17
            if(vect[j].sexo=='H')
            {
                fprintf(fa," MASCULINO |\n");
            }
            else···
            k++;
```

Caso 8:

Función printArchTxt:

- Muestra un archivo de texto si coincide con el nombre leído.

```
void printArchTxt(void)
{
    FILE *fa;
    char name[L];
    int len;
    char c;
    printf("Mostrar archivo\n\n");
    printf("Ingresa el nombre del archivo de texto: ");
    gets(name);
    len=strlen(name);
    name[len++]='.';
    name[len++]='t';
    name[len++]='x';
    name[len++]='t';
    name[len]='\0';

    fa=fopen(name,"r");
    if(fa)
    {
        printf("\n %s\n\n",name);
        do
        {
            c=fgetc(fa);
            printf("%c",c);
        }while(!feof(fa));
        fclose(fa);
    }
    else…
```

Caso 9:

Función genArchivoBin:

- Genera un archivo binario con los datos actuales del registro actual.
- Recibe el vector de registros y el número de registros.
- Renombra y elimina archivos si es necesario, mantiene un archivo.dll y un archivo.tmp, va reemplazando el dll por datos actuales y el tmp por el dll.

```
void genArchivoBin(Tdato vect[],int i)
{
    FILE *fa;
    int j;
    if(i>0)
    {
        fa=fopen("registrosbin.dll","rb");
        if(fa)
        {
            fclose(fa);
            fa=fopen("reigstrosbin.tmp","rb");
            if(fa)
            {
                remove("registros.tmp");
                rename("registrosbin.dll","registros.tmp");
                fa=fopen("registrosbin.dll","wb");
                for(j=0;j<i;j++)
                {
                    fwrite(&vect[j],sizeof(Tdato),1,fa);
                }
                fclose(fa);
            }
            else…
        }
        else…
        printf("\nSe genero el archivo binario\n");
    }
    else…
}
```

Caso 10:

Cambia el estado de las banderas después de cargar los datos.

```
case 10:
    if(cargBin)
    {
        i=cargarArchivoBin(empleados,i);
        if(i>2500)
        {
            ordenMetodo=1;
        }
        orden=0;
        cargBin=0;
```

Función cargarArchivoBin:

- Carga el archivo datos.dll en el vector de registros.
- Recibe el vector de registros y el número de registros.
- Devuelve la cantidad de registros que se cargaron.

```
int cargarArchivoBin(Tdato vect[],int i)
{
    FILE *fa;
    int j;
    fa=fopen("datos.dll","rb");
    if(fa)
    {
        j=0;
        while(!feof(fa))
        {
            fread(&vect[j],sizeof(vect[j]),1,fa);
            vect[j].key=vect[j].NoEmpleado;
            j++;
        }
        printf("\nSe cargo el archivo binario datos.dll a los registros\n");
        fclose(fa);
        return j;
    }
    else
    {
        printf("\nEl archivo binario a cargar no existe\n");
    }
    return 0;
}
```

Función calcularEdad:

- Calcula la edad actual del alumno según la fecha del sistema y su fecha de nacimiento.
- Recibe el año, mes y día de nacimiento del alumno.
- Devuelve la edad del alumno.

```
int calcularEdad(int anio,int mes,int dia)
{
    int edad;
    time_t t;
    t=time(NULL);
    struct tm *fecha;
    fecha=localtime(&t);

    edad=(fecha->tm_year+1900)-anio;
    if(mes>fecha->tm_mon+1)
    {
        edad--;
    }
    else
    {
        if(mes==fecha->tm_mon+1)
        {
            if(dia>fecha->tm_mday)
            {
                edad--;
            }
        }
    }
    return edad;
}
```

Caso 11:

Función printDel:

- Imprime los registros eliminados del archivo binario.
- Guarda los registros con status 0 en un registro e imprime el registro en forma de tabla.

```
void printDel(void)
{
    FILE *fa;
    Tdato reg;
    int k;
    fa=fopen("registrosbin.dll","rb");
    if(fa)
    {
        printf("\nEmpleados eliminados\n\n");
        printf("| No   | No.Empleado |         Puesto         | Ape
        k=0;
        while(!feof(fa))
        {
            fread(&reg,sizeof(Tdato),1,fa);
            if(!reg.status)
            {
                printf("| %4d ",k);
                printEmpleadoTab(reg);
                k++;
            }
        }
        fclose(fa);
```

Conclusión:

Los archivos binarios me resultaron ser más útiles para la carga de datos desde archivos, ya que se trabajan con ellos más fácilmente sin indicar el tipo de dato manualmente. El uso de funciones usando el campo key facilita la reutilización de código, en general el programa es similar a los anteriores por lo que fue sencillo de implementar las nuevas características requeridas.

6. Anexos

PDF:

https://drive.google.com/file/d/12bFkGfdPiNmczQji16_K8JjkSN59_HIL/view?usp=sharing

Repositorio:

https://github.com/Muners24/Programacion-Estructurada

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

El Lenguaje de programación C (S/f). Universidad de Coruña. Recuperado el 20 de noviembre de 2023, de https://www.dc.fi.udc.es/~so-grado/current/Varios/CursoC.pdf