



Universidad Autónoma de Baja California

Facultad de Ingeniería Arquitectura y Diseño

Ingeniero en Software y tecnologías emergentes

Materia: Programación Estructurada / Clave 36276

Alumno: Antonio Ramos González

Matrícula: 372576

Maestro: Pedro Núñez Yépiz

Actividad No. : 14

Tema - Unidad : Archivos

Ensenada Baja California a 27 de Noviembre del 2023

```

/*
Antonio Ramos Gonzalez Mt: 372576
24/11/2023 || 27/11/2023
El usuario podra agregar, eliminar, buscar, ordenar, imprimir el registro original y el ordenado, crear un archivo de texto y empaquetar
RGA_Act14_932
*/
#include "Babilonia.h"
typedef int Tkay;

int msg();
void menu(int numReg);
int cargar();
void cargar2(TWrKr reg[], Tindex index[]);
TWrKr agregar(TWrKr reg[], int n);
int searchSec(Tindex index[], int n, int mt);
void printReg(TWrKr reg);
void eliminar(int n);
void buscar(int n);
void agregarbin(TWrKr reg);
int orderBu(Tindex reg[], int n);
void printOrder(Tindex indice[], int n);
void printOriginal(int n);
void crearTxtOrig();
void crearTxtOrd(Tindex indice[], int n);
void Empaqueta(int n);
void quicksort(Tindex reg[], int Low, int high);
void swap(Tindex reg[], int i, int j);
int partition(Tindex reg[], int low, int high);
void insertionSort(Tindex reg[], int n);
int searchBin(Tindex reg[], int inf, int sup, int mt);
void printRegOr(Tindex reg);

int main()
{
    int tam = (cargar());
    fflush(stdin);
    srand(time(NULL));
    menu(tam);
    return 0;
}

int msg()
{
    printf("MENU\n");
    printf("1.-AGREGAR\n");
    printf("2.-ELIMINAR\n");
    printf("3.-BUSCAR\n");
    printf("4.-ORDENAR\n");
    printf("5.-IMPRIMIR REGISTRO ORIGINAL\n");
    printf("6.-IMPRIMIR REGISTRO ORDENADO\n");
    printf("7.-CREAR TXT\n");
    printf("8.-EMPAQUETAR\n");
    printf("0.-SALIR\n");

    return valid("ELIJE UNA OPCION: ", 0, 8);
}

void menu(int numReg)
{
    int opc;
    int tam, band = 0;
    int band_order = 0;
    tam = numReg * 1.25;
    TWrKr reg[tam];
    Tindex index[tam];
    cargar2(reg, index);
    do
    {
        opc = msg();
        switch (opc)
        {
            case 1:

```

```

        if (numReg < tam)
        {
            reg[numReg] = agregar(reg, numReg);
            index[numReg].id = reg[numReg].enrollement;
            printf("%d\n", index[numReg].id);
            index[numReg].index = numReg;

            numReg++;
            band = 0;
            if (band_order == 3)
            {
                band = rand() % 2 + 1;
            }
        }
        else
        {
            printf("NO HAY ESPACIO DISPONIBLE\n");
        }

        break;
case 2:
    int mt;
    int pos;
    if (band == 0)
    {
        printf("BUSQUEDA SECUENCIAL\n");
        mt = valid("INGRESA MATRICULA A ELIMINAR: ", 300000, 399999);
        pos = searchSec(index, numReg, mt);
        if (pos != -1)
        {
            printReg(reg[pos]);
        }
    }
    else
    {
        printf("BUSQUEDA BINARIA\n");
        mt = valid("INGRESA MATRICULA A ELIMINAR: ", 300000, 399999);
        pos = searchBin(index, 0, numReg, mt);
        if (pos != -1)
        {
            printRegOr(index[pos]);
        }
    }
    break;
case 3:
    if (band == 0)
    {
        printf("BUSQUEDA SECUENCIAL\n");
        mt = valid("INGRESA MATRICULA A BUSCAR: ", 300000, 399999);
        pos = searchSec(index, numReg, mt);
        if (pos != -1)
        {
            printReg(reg[pos]);
        }
    }
    else
    {
        printf("BUSQUEDA BINARIA\n");
        mt = valid("INGRESA MATRICULA A BUSCAR: ", 300000, 399999);
        pos = searchBin(index, 0, numReg, mt);
        if (pos != -1)
        {
            printRegOr(index[pos]);
        }
    }

    break;
case 4:
    if (band == 0)
    {
        if (band_order == 0)

```

```

case 4:
    if (band == 0)
    {
        if (band_order == 0)
        {
            quicksort(index, 0, numReg - 1);
            band_order = 3;
        }
        if (band_order == 1)
        {
            insertionSort(index, numReg);
        }

        if (band_order == 2)
        {
            orderBu(index, numReg);
        }
    }

    band = 1;
    break;
case 5:
    printOriginal(numReg);
    break;
case 6:
    printOrder(index, numReg);
    break;
case 7:
    int elec;
    printf("1.-ORDENADO\n");
    printf("2.-Desordenado\n");
    printf("0.-SALIR\n");
    elec = valid("COMO DESEA GENERAR EL ARCHIVO DE TEXTO: ", 0, 2);
    if (elec == 1)
    {
        crearTxtOrd(index, numReg);
    }
    else
    {
        crearTxtOrig();
    }
    break;
case 8:
    Empaqueta(numReg);
    break;
}
} while (opc != 0);
}

```

```

int cargar()
{
    TWrKr temp;
    int i = 0;
    FILE *fa;
    fa = fopen("datos.dat", "rb");
    if (fa == NULL)
    {
        printf("ERROR AL ABRIR ARCHIVO");
    }
}

```

```

else
{
    while (fread(&temp, sizeof(TWrKr), 1, fa))
    {
        i++;
    }
}
fclose(fa);
return i;
}
}

```

```

void cargar2(TWrKr reg[], Tindex index[])
{
    TWrKr temp;
    int i = 0;
    FILE *fa;
    fa = fopen("datos.dat", "rb");
    if (fa == NULL)
    {
        printf("ERROR AL ABRIR ARCHIVO");
    }
    else
    {
        while (fread(&temp, sizeof(TWrKr), 1, fa))
        {
            reg[i] = temp;
            index[i].id = temp.enrollement;
            index[i].index = i;
            i++;
        }
        fclose(fa);
    }
}
}

```

```

TWrKr agregar(TWrKr reg[], int n)
{
    TWrKr temp;
    int sex;
    temp.status = 1;
    do
    {
        temp.enrollement = rand() % 1000000 + 300000;
    } while (verMt(reg, n, temp.enrollement));
    sex = rand() % 2 + 1;
    nombreAl(temp.name, sex);
    apAl(temp.LastName1);
    apAl(temp.LastName2);
    if (sex == 1)
    {
        strcat(temp.sex, "HOMBRE");
    }
    else
    {
        strcat(temp.sex, "MUJER");
    }
    puestoAl(temp.JobPstion);

    estados(temp.state);
}

```

```

    estados(temp.state);

    temp.age = rand() % 60 + 18;
    do
    {
        temp.CellPhone = rand() % 10000000 + 1999999;
    } while (verMt(reg, n, temp.CellPhone));
    agregarbin(temp);
    return temp;
}

int searchSec(Tindex index[], int n, int mt)
{
    int i;
    i = 0;
    for (i = 0; i <= n; i++)
    {
        if (index[i].id == mt)
        {
            return i;
        }
    }
    return -1;
}

void printReg(TWrKr reg)
{
    printf("NOMBRE: %s\n", reg.name);
    printf("APELLIDO PATERNO: %s\n", reg.LastName1);
    printf("APELLIDO MATERNO: %s\n", reg.LastName2);
    printf("MATRICULA: %d\n", reg.enrollement);
    printf("EDAD: %d\n", reg.age);
    printf("SEXO: %s\n", reg.sex);
    printf("PUESTO: %s\n", reg.JobPstion);
    printf("ESTADO: %s\n", reg.state);
}

void printRegOr(Tindex reg)
{
    FILE *fa;
    TWrKr temp;
    fa = fopen("datos.dat", "rb");
    fseek(fa, reg.index * sizeof(TWrKr), SEEK_SET);
    fread(&temp, sizeof(TWrKr), 1, fa);
    fclose(fa);
    printf("NOMBRE: %s\n", temp.name);
    printf("APELLIDO PATERNO: %s\n", temp.LastName1);
    printf("APELLIDO MATERNO: %s\n", temp.LastName2);
    printf("MATRICULA: %d\n", temp.enrollement);
    printf("EDAD: %d\n", temp.age);
    printf("SEXO: %s\n", temp.sex);
    printf("PUESTO: %s\n", temp.JobPstion);
    printf("ESTADO: %s\n", temp.state);
}

void eliminar(int n)
{
    int opc;
    FILE *fa;

```

```

FILE *fa;
TWrKr temp;

fa = fopen("datos.dat", "rb+");
fseek(fa, n * sizeof(TWrKr), SEEK_SET);
fread(&temp, sizeof(TWrKr), 1, fa);
if (temp.status == 1)
{
    printReg(temp);
    opc = valid("\nDesea eliminarlo?\n1.-Si\n2.-No  ", 1, 2);
}
else
{
    printf("El registro ya esta eliminado\n");
}

if (opc == 1)
{
    temp.status = 0;
    fseek(fa, n * sizeof(TWrKr), SEEK_SET);
    fwrite(&temp, sizeof(TWrKr), 1, fa);
    printf("Se ha eliminado con exito\n");
}
else
{
    printf("No se ha eliminado\n");
}
fclose(fa);
}

```

```

void agregarbin(TWrKr reg)
{
    FILE *fa;
    fa = fopen("datos.dat", "ab");
    if (fa == NULL)
    {
        printf("ERROR AL ABRIR ARCHIVO");
    }
    else
    {
        fwrite(&reg, sizeof(TWrKr), 1, fa);
    }
    fclose(fa);
}

```

```

int orderBu(Tindex reg[], int n)
{
    int i, j;
    Tindex temp;

    for (i = 0; i < n - 1; i++)
    {
        for (j = i + 1; j < n; j++)
        {
            if (reg[j].id < reg[i].id)
            {
                temp = reg[i];
                reg[i] = reg[j];
                reg[j] = temp;
            }
        }
    }
}

```

```
reg[j] = temp;
```

```
}
```

```
}
```

```
}  
return 0;
```

```
}
```

```
void printOrder(Tindex indice[], int n)
```

```
{
```

```
FILE *fa;
```

```
TWrKr temp;
```

```
fa = fopen("datos.dat", "rb");
```

```
for (int i = 0; i < n; i++)
```

```
{
```

```
fseek(fa, indice[i].index * sizeof(TWrKr), SEEK_SET);
```

```
fread(&temp, sizeof(TWrKr), 1, fa);
```

```
printf("%s %s %s %s %s %s %d %d\n", temp.name, temp.LastName1, temp.LastName2,  
temp.sex, temp.JobPstion, temp.state, temp.age, temp.CellPhone);
```

```
}
```

```
fclose(fa);
```

```
}
```

```
void printOriginal(int n)
```

```
{
```

```
FILE *fa;
```

```
TWrKr temp;
```

```
fa = fopen("datos.dat", "rb");
```

```
for (int i = 0; i < n; i++)
```

```
{
```

```
fseek(fa, i * sizeof(TWrKr), SEEK_SET);
```

```
fread(&temp, sizeof(TWrKr), 1, fa);
```

```
printf("%s %s %s %s %s %s %d %d\n", temp.name, temp.LastName1, temp.LastName2,  
temp.sex, temp.JobPstion, temp.state, temp.age, temp.CellPhone);
```

```
}
```

```
fclose(fa);
```

```
}
```

```
void crearTxtOrig()
```

```
{
```

```
FILE *fabin;
```

```
FILE *fatxt;
```

```
TWrKr temp;
```

```
fabin = fopen("datos.dat", "rb");
```

```
fatxt = fopen("datos.txt", "w");
```

```
if (fabin == NULL || fatxt == NULL)
```

```
{
```

```
printf("ERROR AL ABRIR ARCHIVO");
```

```
}
```

```
else
```

```
{
```



```

    while (fread(&temp, sizeof(TWrKr), 1, fabin))
    {
        fprintf(fatxt, "%-10s %-10s %-10s %-10s %-10s %-10s %-10d %d\n", temp.name, temp.LastName1,
            temp.LastName2, temp.sex, temp.JobPstion, temp.state, temp.age, temp.CellPhone);
    }
}
printf("SE HA CREADO EL ARCHIVO DE TEXTO\n");

fclose(fabin);
fclose(fatxt);
}

```

```

void crearTxtOrd(Tindex indice[], int n)
{
    FILE *fabin;
    FILE *fatxt;

    TWrKr temp;

    fabin = fopen("datos.dat", "rb");
    fatxt = fopen("datosOrdenados.txt", "w");

    if (fabin == NULL || fatxt == NULL)
    {
        printf("ERROR AL ABRIR ARCHIVO");
    }
    else
    {
        for (int i = 0; i < n; i++)
        {
            fseek(fabin, indice[i].index * sizeof(TWrKr), SEEK_SET);
            fread(&temp, sizeof(TWrKr), 1, fabin);

            fprintf(fatxt, "%-10s %-10s %-10s %-10s %-10s %-10s %-10d %d\n", temp.name, temp.LastName1,
                temp.LastName2, temp.sex, temp.JobPstion, temp.state, temp.age, temp.CellPhone);
        }
        printf("SE HA CREADO EL ARCHIVO DE TEXTO\n");

        fclose(fabin);
        fclose(fatxt);
    }
}

```

```

void Empaqueta(int n)
{
    FILE *fad, *fab;
    TWrKr temp;
    char nombre[11] = "datos.dat";
    rename("datos.dat", "datos.bak");
    fad = fopen(nombre, "wb");
    fab = fopen("datos.bak", "rb");
    if (fad == NULL)
    {
        printf("ERROR AL ABRIR ARCHIVO");
    }
    else

```

```

    else
    {
        for (int i = 0; i < n; i++)
        {
            fseek(fab, i * sizeof(TWrKr), SEEK_SET);
            fread(&temp, sizeof(TWrKr), 1, fab);
            if (temp.status == 1)
            {
                fwrite(&temp, sizeof(TWrKr), 1, fad);
            }
        }
        fclose(fad);
        fclose(fab);
    }
}

```

```

void quicksort(Tindex reg[], int low, int high)
{
    if (low < high)
    {
        int pi = partition(reg, low, high);

        quicksort(reg, low, pi - 1);
        quicksort(reg, pi + 1, high);
    }
}

```

```

int partition(Tindex reg[], int low, int high)
{
    Tindex pivot;
    pivot.id = reg[high].id;
    int i = low - 1;

    for (int j = low; j <= high - 1; j++)
    {
        if (reg[j].id <= pivot.id)
        {
            i++;
            swap(reg, i, j);
        }
    }
    swap(reg, i + 1, high);
    return i + 1;
}

```

```

void swap(Tindex reg[], int i, int j)
{
    Tindex temp = reg[i];
    reg[i] = reg[j];
    reg[j] = temp;
}

```

```

void insertionSort(Tindex reg[], int n)
{
    Tindex temp;
    int j;
    for (int i = 1; i < n; i++)
    {
        temp = reg[i];

```

```

    temp = reg[i];
    j = i - 1;
    while (j >= 0 && reg[j].id > temp.id)
    {
        reg[j + 1] = reg[j];
        j--;
    }
    reg[j + 1] = temp;
}

int searchBin(Tindex reg[], int inf, int sup, int mt)
{
    int med;
    while (inf <= sup)
    {
        med = (inf + sup) / 2;
        if (reg[med].id == mt)
        {
            return med;
        }
        else
        {
            if (mt < reg[med].id)
            {
                sup = med--;
            }
            else
            {
                inf = med++;
            }
        }
    }
    return -1;
}

```

MENU

1.-AGREGAR

2.-ELIMINAR

3.-BUSCAR

4.-ORDENAR

5.-IMPRIMIR REGISTRO ORIGINAL

6.-IMPRIMIR REGISTRO ORDENADO

7.-CREAR TXT

8.-EMPAQUETAR

0.-SALIR

ELIJE UNA OPCION: 1

BUSQUEDA SECUENCIAL

INGRESA MATRICULA A BUSCAR: 316053

NOMBRE: ANTONIA

APELLIDO PATERNO: PEREZ

APELLIDO MATERNO: ROJAS

MATRICULA: 316053

EDAD: 62

SEXO: aMUJER

PUESTO: COMERCIAL

ESTADO: HG

MENU

LAURA GOMEZ MEDINA MUJER EspRRHH MS 28 1032366
HECTOR MOLINA AGUILAR HOMBRE AnMktDg SP 29 1014722
ELENA MORA JIMENEZ MUJER ConsFin DF 19 1026083
GABRIELA ESTRELLA CRUZ MUJER PsOrg ZS 46 1023441
MARTA DURAN VARGAS MUJER IngElec HG 39 1023203
JOSE CONTRERAS CASTILLO HOMBRE AnlSist YN 46 1025268
PILAR LARA ARIAS MUJER Traduct NE 38 1009239
ELENA ORTIZ MACIEL MUJER Traduct SL 49 1032127
CARMEN CONTRERAS MENDOZA MUJER IngRed BC 25 1016430
AURORA HERRERA VALENCIA MUJER EspLog SR 34 1023983
ENRIQUE MACIEL PARDO HOMBRE RepVInt HG 47 1017951
SUSANA CAMACHO ROJAS MUJER PsOrg SR 25 1010268
ARMANDO PARDO FLORES HOMBRE Arquitect MN 45 1025737
ALEJANDRA ESTRELLA VALENZUELA MUJER TrabSoc OC 23 1003841
RAQUEL DURAN IGLESIAS MUJER AsServ TS 22 1010993
CLAUDIA AGUIRRE HERRERA MUJER CONTADOR Tabasco 62 2023021
RAFAELA RODRIGUEZ OCHOA MUJER REPARTIDOR QT 26 2001868
ANA GOMEZ DELGADO aMUJER ADMINISTRADOR JC 42 2012257
PILAR ORTEGA RIVAS aMUJERMUJER INTENDENTE SR 40 2022092
LORENA ARAYA MENDOZA aMUJERMUJERMUJESECRETARIA SECRETARIA
SARA CABRERA AGUIRRE aMUJER SECRETARIA CS 32 2012269
ARMANDO SILVA GONZALEZ aHOMBRE ADMINISTRADOR GT 24 2002308
ANTONIA PEREZ ROJAS aMUJER COMERCIAL HG 62 2002775
IGNACIO MEDINA NAVARRO aHOMBRE TECNICO TS 38 2008740
EVA NAVARRO MORALES aMUJER INGENIERO SL 29 2006736
ENRIQUE MENDEZ VARGAS aHOMBRE REPARTIDOR GT 45 2006920

SALVADOR MOLINA VARGAS HOMBRE ChefEje CM 48 1025107
JUAN CHACON RIOS HOMBRE EspRRHH TS 38 1013784
CONSUELO ESCOBAR CABRERA MUJER DisUX ZS 32 1018491
GLORIA VASQUEZ MUNGUIA MUJER DevSoft OC 48 1000328
ROBERTO TOVAR REYES HOMBRE PsOrg BS 26 1001674
MARIO ORTIZ MEDINA HOMBRE ChefEje AG 21 1011038
MIGUEL ALVAREZ URIBE HOMBRE DisGraf CC 30 1024358
SALVADOR LARA SOLIS HOMBRE EspSeg CS 23 1002437
MERCEDES PAREDES BELTRAN MUJER GteProj JC 46 1011613
MERCEDES SUAREZ CAMACHO MUJER TecSup TL 22 1010523
JOSE CASTANEDA VASQUEZ HOMBRE InvCien ZS 22 1008462
ARTURO GONZALES TORRES HOMBRE Traduct OC 31 1007788
PAULA ESPINOSA ZAMORA MUJER TecSup NT 30 1016786
PILAR ROJAS ARIAS MUJER Contado MN 38 1025346
CARMEN ALVARADO HIDALGO MUJER AsServ GT 44 1014616
PILAR LEON VASQUEZ MUJER Abogado CC 35 1006386
PEDRO TOVAR TORRES HOMBRE ChefEje HG 48 1000943
SILVIA CASILLAS TORRES MUJER IngElec SP 31 1032505
PEDRO QUINTERO VALDES HOMBRE EspLog JC 44 1023786
MERCEDES TORRES FLORES MUJER EspSeg SP 49 1004974

MENU

- 1.-AGREGAR
- 2.-ELIMINAR
- 3.-BUSCAR
- 4.-ORDENAR
- 5.-IMPRIMIR REGISTRO ORIGINAL
- 6.-IMPRIMIR REGISTRO ORDENADO
- 7.-CREAR TXT
- 8.-EMPAQUETAR
- 0.-SALIR

ELIJE UNA OPCION:

MENU

1.-AGREGAR

2.-ELIMINAR

3.-BUSCAR

4.-ORDENAR

5.-IMPRIMIR REGISTRO ORIGINAL

6.-IMPRIMIR REGISTRO ORDENADO

7.-CREAR TXT

8.-EMPAQUETAR

0.-SALIR

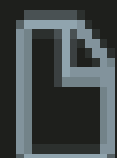







ELIJE UNA OPCION: 8

1.-ORDENADO

2.-Desordenado

0.-SALIR

COMO DESEA GENERAR EL ARCHIVO DE TEXTO:

	datos.bak	M
	datos.dat	M
	datos.txt	M
	datosOrdenados.txt	M
	RGA_Act14_932.exe	M
	Babilonia.h	
	datos.dat	U
	RGA_Act14_932.c...	2, M