

PRUEBAS DE ESCRITORIO:

CARGA DE ARCHIVO PRINCIPAL:

```
Se cargo el archivo binario
Presione una tecla para continuar . . .
```

MENU:

```
Menu

1) Agregar
2) Eliminar
3) Buscar
4) Ordenar
5) Mostrar Todo
6) Generar Archivo de Texto
7) Empaquetar
8) Salir
Escoge una opcion: 
```

AGREGAR:

```
Se genero un trabajador y se guardo en el archivo
Presione una tecla para continuar . . .
```

ELIMINAR:

```
Eliminar empleado

Ingresa el número de empleado que deseas eliminar: 321005

Numero de Empleado: 321005
Nombre: CARMEN
Apellido Paterno: CONTRERAS
Apellido Materno MENDOZA
Sexo: MUJER
Pueso: IngRed
Estado: BC
Edad: 25
Telefono: 1016430
Deseas eliminar al empleado? (0:Si 1:No): 0

Se elimino el empleado

Presione una tecla para continuar . . .
```

BUSCAR:

```
Buscar empleado

Ingresa el número de empleado que deseas buscar: 331050

Numero de Empleado: 331050
Nombre: AURORA
Apellido Paterno: HERRERA
Apellido Materno VALENCIA
Sexo: MUJER
Puesto: EspLog
Estado: SR
Edad: 34
Telefono: 1023983

Presione una tecla para continuar . . .
```

ORDENAR:

```
El arreglo indexado ya esta ordenado

Presione una tecla para continuar . . .
```

MOSTRAR TODO:

```
Mostrar Registros

1) Normal
2) Ordenado
3) Regresar
Escoge una opcion:
```

NORMAL:

```
3353 | 311893 | Traduct | ORTIZ | MACIEL | ELENA | 6461032127 | 49 | MUJER | SL |
3354 | 321005 | IngRed | CONTRERAS | MENDOZA | CARMEN | 6461016430 | 25 | MUJER | BC |
3355 | 331050 | EspLog | HERRERA | VALENCIA | AURORA | 6461023983 | 34 | MUJER | SR |
3356 | 328851 | RepVInt | MACIEL | PARDO | ENRIQUE | 6461017951 | 47 | HOMBRE | HG |
3357 | 311976 | PsOrg | CAMACHO | ROJAS | SUSANA | 6461010268 | 25 | MUJER | SR |
3358 | 328201 | Arquitect | PARDO | FLORES | ARMANDO | 6461025737 | 45 | HOMBRE | MN |
3359 | 308728 | TrabSoc | ESTRELLA | VALENZUELA | ALEJANDRA | 6461003841 | 23 | MUJER | OC |
3360 | 310512 | AsServ | DURAN | IGLESIAS | RAQUEL | 6461010993 | 22 | MUJER | TS |
3361 | 328469 | GERENTE DE PRODUCCION | AGUIRRE | GUTIERREZ | EDGAR | 6461916979 | 56 | HOMBRE | SL |

Presione una tecla para continuar . . .
```

ORDENADO:

3355	332786	Contado	ROJAS	ARIAS	PILAR	6461025346	38	MUJER	MN
3356	332720	AsServ	ALVARADO	HIDALGO	CARMEN	6461014616	44	MUJER	GT
3357	332723	Abogado	LEON	VASQUEZ	PILAR	6461006386	35	MUJER	CC
3358	332727	ChefEje	TOVAR	TORRES	PEDRO	6461000943	48	HOMBRE	HG
3359	332732	IngElec	CASILLAS	TORRES	SILVIA	6461032505	31	MUJER	SP
3360	332757	EspLog	QUINTERO	VALDES	PEDRO	6461023786	44	HOMBRE	JC
3361	332761	EspSeg	TORRES	FLORES	MERCEDES	6461004974	49	MUJER	SP

Presione una tecla para continuar . . .

GENERAR ARCHIVO DE TEXTO:

Generar Archivo de Texto

- 1) Normal
- 2) Ordenado
- 3) Regresar

Escoge una opcion: 1

Ingresar el nombre del archivo de texto: NORMAL

MMR_ACT14_932 > NORMAL.txt									
3333	3332	317394	AnMktDg	SANDOVAL	SOSA	ISABEL	6461008		
3334	3333	306770	ProfPrim	MACIAS	CASTANEDA	NATALIA	6461006		
3335	3334	319845	TrabSoc	MOLINA	VEGA	ALEJANDRO	6461024		
3336	3335	303094	IngRed	RIVERA	RIOS	SONIA	6461009		
3337	3336	329717	ProfPrim	MERCADO	ARIAS	JOSE	6461031		
3338	3337	305034	InvCien	ALVARADO	MIRANDA	CARMEN	6461000		
3339	3338	316519	ChefEje	ROMAN	VILLANUEVA	JORGE	6461002		
3340	3339	324467	PsOrg	ORTIZ	GARZA	ANA	6461015		
3341	3340	314382	AsServ	BARRIOS	ROMERO	SOFIA	6461013		

Generar Archivo de Texto

- 1) Normal
- 2) Ordenado
- 3) Regresar

Escoge una opcion: 2

Ingresar el nombre del archivo de texto: ORDENADO








MMR_ACT14_932 > ORDENADO.txt									
1	NO	NO EMP	PUESTO	APELLIDO PATERNO	APELLIDO MATERNO	NOMBRE	TELEFON		
42	41	300466	ChefEje	SOTO	MEDINA	ADRIANA	64610144		
43	42	300469	DevSoft	OSORIO	DURAN	PILAR	64610082		
44	43	300482	EspSeg	CABRERA	MORENO	SALVADOR	64610209		
45	44	300486	EspLog	TOVAR	SERRANO	GLORIA	64610149		
46	45	300491	ConsFin	ESTRELLA	GUZMAN	ADRIANA	64610211		
47	46	300498	GteProy	GUZMAN	LOPEZ	EMILIO	64610312		

EMPAQUETAR:

Se empaqueto correctamente

Presione una tecla para continuar . . .

▼	MMR_ACT14_932	●
≡	datos.bak	U
≡	datos.dat	M
≡	datosdel.bak	U

-  datos.bak
-  datos.dat
-  datos1.bak
-  datos2.bak
-  datos3.bak
-  datos4.bak
-  datosdel.bak

```

/* MejÃ a MartÃ nez, RamÃ n      372099
/* 28 de noviembre de 2023
/* Programa de gestion de empleados usando structs, distintos metodos de busqueda, ordenacion y uso de archivos de texto y binarios, usando como almacenamiento un archivo binario indexado
/* MMR_ACT13_932

#include "C:\Users\ramon\OneDrive\Escritorio\MMR_ACT1_932\MMR_ACT14_932\MMRLIB.h"
#include <stdio.h>
#include <string.h>

#define ELIMINAR 1
#define BUSCAR 0
#define ENCABEZADO " | NO      | NO EMP | PUESTO          | APELLIDO PATERNO | APELLIDO MATERNO | NOMBRE          | TELEFONO      | EDAD | SEXO      | EST |"
typedef int Tkey;

typedef struct _Wrkr{
    int status;
    Tkey key;
    char name[L];
    char apPat[LN];
    char apMat[LN];
    char sexo[15];
    char puesto[L];
    char estado[L];
    int edad;
    Tkey telefono;
}TWrkr;

typedef struct _index{
    Tkey key;
    Tkey indice;
    Tkey Telefono;
}Tindex;

int mags(void);

void menu(int REG);
void GenWrkr(Tindex vect[],int i);
int GenKey(Tindex vect[],int i,int ri,int rf);
void GenH_Name(char str[]);
void GenM_Name(char str[]);
void GenAp(char str[],char str1[]);
void GenEst(char estado[]);
void GenPuesto(char str[]);
int BuscarWrkr(Tindex vect[],int i,int orden);
int BuscarSecIndex(Tindex vect[],int i,Tkey num);
int BuscarBinIndex(Tindex vect[],int i,Tkey num);
int BuscarSecTelIndex(Tindex vect[],int i,int num);
void DelWrkr(Tindex vect[],int i,int orden);
void PrintRegReg(int indice,int razon);
void OrdenarIndex(Tindex vect[],int i,int MetodoOrden);
void BubbleSortIndex(Tindex vect[],int n);
void HeapSortIndex(Tindex vect[],int n);
void AdjustHeapIndex(Tindex vect[],int n,int raiz);
void SwapIndex(Tindex *x,Tindex *y);
void MergeSortIndex(Tindex vect[],int izq,int derch);
void FusionMergeSortIndex(Tindex vect[],int izq,int medio,int derch);
void PrintRegTab(Tindex vect[],int i,int sorden,int MetodoOrden);
void PrintArchBin(void);
void PrintBinIndex(Tindex vect[],int i);
void GenArchivoTxt(Tindex vect[],int i,int sorden,int MetodoOrden);
void GenArchTxtNormal(char name[]);
void GenArchTxtOrden(Tindex vect[],int i,char name[]);
void PrintArchTxt(char name[]);
void empaquetar(Tindex vect[],int i,int respaldo);

int main()
{
    srand(time(NULL));
    FILE *fa;
    int REG;
    system("C:\\Users\\ramon\\OneDrive\\Escritorio\\MMR_ACT1_932\\MMR_ACT14_932\\SizeArchivo.exe");
    fa=fopen("NumeroRegistros.txt","r");
    if (fa)
    {
        fscanf(fa,"%d",&REG);
        fclose(fa);
        menu(REG);
    }
    else
    {
        printf("\nEl archivo binario de registros no existe\n");
        system("pause");
    }

    return 0;
}

int mags(void)
{
    printf("  Menu\n\n");
    printf("1) Agregar\n");
    printf("2) Eliminar\n");
    printf("3) Buscar\n");
    printf("4) Ordenar\n");
    printf("5) Mostrar Todo\n");
    printf("6) Generar Archivo de Texto\n");
    printf("7) Empaquetar\n");
    printf("8) Salir\n");
    return(validNum("Escoge una opcion: ",1,9));
}

void menu(int REG)
{
    int i=0,j=0,respaldo=0;
    REG;
    int op;
    int orden=0,MetodoOrden=0;
    Tindex trabajadores[REG];
    TWrkr reg;
    FILE *fa;
    fa=fopen("datos.dat","rb");
    if (fa)
    {
        //carga de archivo binario al arreglo indexado
        while(fread(&reg,sizeof(TWrkr),1,fa)==1)
        {
            if (reg.status)
            {
                trabajadores[i].indice=j;
            }
        }
    }
}

```

```

        trabajadores[i].key=reg.key;
        trabajadores[i].Telefono=reg.telefono;
        i++;
    }
    j++;
}
fclose(fa);
if (i>3500)
{
    if (i>3750)
    {
        MetodoOrden=2;
    }
    else
    {
        MetodoOrden=1;
    }
}
printf("\nSe cargo el archivo binario\n");
system("pause");
do
{
    system("cls");
    op=msg();
    system("cls");
    switch (op)
    {
        case 1:
            if (i+1<REG)
            {
                GenWrkr(trabajadores,i++);
                orden=0;
                if (i>3500)
                {
                    if (i>3750)
                    {
                        MetodoOrden=2;
                    }
                    else
                    {
                        MetodoOrden=1;
                    }
                }
            }
            else
            {
                printf("\nLa capacidad de registros llego al limite\n");
            }
            break;
        case 2:
            DelWrkr(trabajadores,i,orden);
            break;
        case 3:
            BuscarWrkr(trabajadores,i,orden);
            break;
        case 4:
            if (!orden)
            {
                OrdenarIndex(trabajadores,i,MetodoOrden);
                orden=1;
            }
            else
            {
                printf("\nEl arreglo indexado ya esta ordenado\n");
            }
            break;
        case 5:
            PrintRegTab(trabajadores,i,orden,MetodoOrden);
            break;
        case 6:
            GenArchivoTxt(trabajadores,i,orden,MetodoOrden);
            break;
        case 7:
            empaquetar(trabajadores,i,respaldo++);
            break;
    }
    printf("\n");
    system("pause");
}while (op!=8);
}
else
{
    printf("\nEl archivo binario de registros no existe\n");
}
}

//case 1
void GenWrkr(Tindex vect[],int i)
{
    FILE *fa;
    TWrkr reg;
    int s,num;
    int j;
    reg.status=1;
    reg.key=GenKey(vect,i,300000,399999);
    vect[i].key=reg.key;
    vect[i].indice=i;
    reg.edad=(rand()%53)+18;
    GenEst(reg.estado);

    s=rand()%2;

    if (s)
    {
        strcpy(reg.sexo,"HOMBRE");
        GenH_Name(reg.name);
    }
    else
    {
        strcpy(reg.sexo,"MUJER");
        GenM_Name(reg.name);
    }

    GenAp(reg.apPat,reg.apMat);

    do
    {
        num=(rand()%100000)+(rand()%99)*100000;
    }while (BuscarSecTelIndex(vect,i,num)!=-1);
    reg.telefono=num;
}

```

```

GenPuesto(reg.puesto);

fa=fopen("datos.dat","rb");
if (fa)
{
    fclose(fa);
    fopen("datos.dat","ab");
    fwrite(&reg,sizeof(TWrkr),1,fa);
    fclose(fa);
    printf("\nSe genero un trabajador y se guardo en el archivo\n");
}
else
{
    printf("\nNo se pudo guardar el registro en el archivo\n");
    printf("\nEl archivo binario no existe\n");
}
}

void GenPuesto(char str[])
{
    char pst[12][L]={
        "OPERADOR","SUPERVISOR","MANTENIMIENTO","INGENIERO DE PROCESOS","INGENIERO DE CALIDAD",
        "OPERADOR DE MAQUINARIA", "ALMACENISTA", "JEFE DE ALMACEN", "INSPECTOR DE CALIDAD",
        "GERENTE DE PRODUCCION", "TECNICO EN SEGURIDAD", "RECURSOS HUMANOS"
    };
    strcpy(str,pst[(rand() %12)]);
}

int GenKey(Tindex vect[],int i,int ri,int rf)
{
    int num;
    int rango=rf-ri;
    do
    {
        num=(rand() %rango-1)+ri;
    }while (BuscarSecIndex(vect,i,num) !=-1);
    return num;
}

void GenH_Name(char str[])
{
    char h_name[100][LN]={
        "JOSE",
        "JUAN",
        "LUIS",
        "MANUEL",
        "ANTONIO",
        "JESUS",
        "CARLOS",
        "FRANCISCO",
        "ALBERTO",
        "JORGE",
        "MIGUEL",
        "ANGEL",
        "JAVIER",
        "ALEJANDRO",
        "ENRIQUE",
        "VICTOR",
        "ARTURO",
        "CESAR",
        "FERNANDO",
        "PEDRO",
        "MARTIN",
        "ROBERTO",
        "EDUARDO",
        "MARIO",
        "ARMANDO",
        "SERGIO",
        "RAUL",
        "ALFREDO",
        "RAFAEL",
        "RICARDO",
        "HECTOR",
        "OSCAR",
        "GERARDO",
        "DAVID",
        "DANIEL",
        "HUGO",
        "JAIME",
        "JULIO",
        "RUBEN",
        "RAMON",
        "MARCO",
        "GABRIEL",
        "EDGAR",
        "GUADALUPE",
        "ALFONSO",
        "GUILLERMO",
        "SALVADOR",
        "OMAR",
        "IVAN",
        "HUMBERTO",
        "FELIPE",
        "ERNESTO",
        "PABLO",
        "IGNACIO",
        "GUSTAVO",
        "ANDRES",
        "ADRIAN",
        "JOEL",
        "AGUSTIN",
        "RODOLFO",
        "GILBERTO",
        "ROGELIO",
        "RENE",
        "TOMAS",
        "SAUL",
        "ISRAEL",
        "OCTAVIO",
        "VICENTE",
        "NOE",
        "GREGORIO",
        "ISMAEL",
        "NICOLAS",
        "BENJAMIN",
        "MOISES",
        "SANTIAGO",
        "EFRAIN",
        "ALONSO",
    };
}

```

```

"ABEL",
"WILBERT",
"ALVARO",
"FELIX",
"MARCOS",
"ADOLFO",
"RODRIGO",
"RAMIRO",
"SAMUEL",
"JOAQUIN",
"ABRAHAM",
"ESTEBAN",
"ULISES",
"RAYMUNDO",
"FIDEL",
"LORENZO",
"GERMAN",
"MAURICIO",
"LEONARDO",
"JOSUE",
"EMMANUEL",
"JULIAN",
"SANTOS",
};
int prob;
prob=rand() %10;
if (prob<=7)
{
    strcpy(str,h_name[rand() %100]);
}
else
{
    strcpy(str,h_name[rand() %100]);
    strConcat(str,h_name[rand() %100]);
}
}

void GenM_Name(char str[])
{
    char m_name[100][LN]={
        "MARIA",
        "GUADALUPE",
        "ROSA",
        "MARTHA",
        "ANA",
        "PATRICIA",
        "LETICIA",
        "ELENA",
        "LAURA",
        "ELIZABETH",
        "ISABEL",
        "ALICIA",
        "MARGARITA",
        "LUZ",
        "ANGELICA",
        "CLAUDIA",
        "SILVIA",
        "NORMA",
        "JUANA",
        "GABRIELA",
        "ADRIANA",
        "VERONICA",
        "TERESA",
        "ALEJANDRA",
        "ALMA",
        "BEATRIZ",
        "YANET",
        "YOLANDA",
        "BLANCA",
        "SANDRA",
        "ARACELI",
        "IRMA",
        "GLORIA",
        "CARMEN",
        "LUISA",
        "ROCIO",
        "CONCEPCION",
        "CRISTINA",
        "EDITH",
        "ESTHER",
        "LILIA",
        "LORENA",
        "MAGDALENA",
        "CECILIA",
        "JOSEFINA",
        "DOLORES",
        "KARINA",
        "LUCIA",
        "ESTELA",
        "SUSANA",
        "MARIBEL",
        "LUCINA",
        "LOURDES",
        "ERIKA",
        "MONICA",
        "AGUSTINA",
        "DIANA",
        "EUGENIA",
        "ALEJANDRINA",
        "PILAR",
        "FRANCISCA",
        "VIRGINIA",
        "LILIANA",
        "GRACIELA",
        "BERTHA",
        "OLGA",
        "MIRIAM",
        "FABIOLA",
        "DELIA",
        "CAROLINA",
        "KARLA",
        "HILDA",
        "ANTONIA",
        "MARICELA",
        "SONIA",
        "NANCY",
        "BERENICE",
        "YADIRA",
        "MAYRA",
        "JUDITH",
    }
}

```



```

"MARISOL",
"ROSARIO",
"REYNA",
"AURORA",
"OLIVIA",
"RAQUEL",
"IRENE",
"ELVIRA",
"CATALINA",
"LIDIA",
"VICTORIA",
"DULCE",
"ISELA",
"SARA",
"MARCELA",
"ELVIA",
"ESPERANZA",
"GUILLERMINA",
"DORA",
"EVA",

};

int prob;
prob=rand() %10;
if (prob<=7)
{
    strcpy(str,m_name[rand() %100]);
}
else
{
    strcpy(str,m_name[rand() %100]);
    strcat(str,m_name[rand() %100]);
}
}

void GenAp(char str[],char str1[])
{
    char ap[100][L]={
        "HERNANDEZ",
        "GARCIA",
        "MARTINEZ",
        "LOPEZ",
        "GONZALEZ",
        "RODRIGUEZ",
        "PEREZ",
        "SANCHEZ",
        "RAMIREZ",
        "FLORES",
        "CRUZ",
        "GOMEZ",
        "MORALES",
        "VAZQUEZ",
        "JIMENEZ",
        "REYES",
        "DIAZ",
        "TORRES",
        "GUTIERREZ",
        "RUIZ",
        "AGUILAR",
        "MENDOZA",
        "CASTILLO",
        "ORTIZ",
        "MORENO",
        "RIVERA",
        "RAMOS",
        "ROMERO",
        "JUAREZ",
        "ALVAREZ",
        "MENDEZ",
        "CHAVEZ",
        "HERRERA",
        "MEDINA",
        "DOMINGUEZ",
        "CASTRO",
        "GUZMAN",
        "VARGAS",
        "VELAZQUEZ",
        "SALAZAR",
        "ROJAS",
        "ORTEGA",
        "CORTES",
        "SANTIAGO",
        "GUERRERO",
        "CONTRERAS",
        "BAUTISTA",
        "ESTRADA",
        "LUNA",
        "LARA",
        "RIOS",
        "AVILA",
        "ALVARADO",
        "DE LA CRUZ",
        "SILVA",
        "DELGADO",
        "CARRILLO",
        "SOLIS",
        "SOTO",
        "LEON",
        "FERNANDEZ",
        "CERVANTES",
        "MARQUEZ",
        "ESPINOSA",
        "MEJIA",
        "VEGA",
        "SANDOVAL",
        "CAMPOS",
        "NAVA",
        "CABRERA",
        "IBARRA",
        "ESPINOZA",
        "SANTOS",
        "ACOSTA",
        "CAMACHO",
        "VALDEZ",
        "FUENTES",
        "VARELA",
        "MIRANDA",
        "MALDONADO",
        "ROBLES",
        "ROSAS",
    }
}

```

```

        "MEZA",
        "MOLINA",
        "TREJO",
        "ROSALES",
        "PACHECO",
        "NAVARRO",
        "SALGADO",
        "AGUIRRE",
        "SALAS",
        "VELASCO",
        "CARDENAS",
        "PINEDA",
        "OROZCO",
        "SERRANO",
        "RANGEL",
        "VALENCIA",
        "SOSA",
        "VASQUEZ"
    };

    int prob;
    prob=rand() %10;
    if (prob<9)
    {
        strcpy(str,ap[rand() %99]);
        strcpy(str1,ap[rand() %99]);
    }
    else
    {
        strcpy(str,ap[rand() %99]);
        str1[0]='X';
        str1[1]='\0';
    }
}

void GenEst(char estado[])
{
    char estList[33][3]={
        "AS","BC","BS","CC","CS","CH","CM","CL","MX","DG",
        "GT","GR","HG","JC","EM","MN","MS","NT","NL","OC",
        "PL","QT","QR","SP","SL","SR","TC","TS","TL","VZ",
        "YN","ZS","NE"
    };

    strcpy(estado,estList[rand() %33]);
    strM(estado);
}

int BuscarSecTelIndex(Tindex vect[],int i,int num)
{
    int j;
    for(j=0;j<i;j++)
    {
        if (vect[j].Telefono==num)
        {
            return j;
        }
    }
    return -1;
}

int BuscarSecIndex(Tindex vect[],int i,Tkey num)
{
    int j;
    for(j=0;j<i;j++)
    {
        if (vect[j].key==num)
        {
            return (vect[j].indice);
        }
    }
    return -1;
}

int BuscarBinIndex(Tindex vect[],int i,Tkey num)
{
    int izq,drcha;
    int med;
    izq=0;
    drcha=i-1;
    while (izq<=drcha)
    {
        med=(izq+drcha-izq)/2;
        if (vect[med].key==num)
        {
            return (vect[med].indice);
        }

        if (vect[med].key<num)
        {
            izq=med+1;
        }
        else
        {
            drcha=med-1;
        }
    }
    return -1;
}

//case 2
void DelWrkr(Tindex vect[],int i,int orden)
{
    int pos;
    int del;
    FILE *fa;
    printf("Eliminar empleado\n\n");
    pos=validNum("Ingresa el nA*mero de empleado que deseas eliminar: ",300000,399999);
    if (orden)
    {
        pos=BuscarBinIndex(vect,i,pos);
        if (pos!=-1)
        {
            PrintRegReg(pos,ELIMINAR);
        }
        else
        {
            printf("\nEl empleado no existe\n\n");
        }
    }
    else

```

```

    {
        pos=BuscarSecIndex(vect,i,pos);
        if (pos!=-1)
        {
            PrintRegReg(pos,ELIMINAR);
        }
        else
        {
            printf("\nEl empleado no existe\n");
        }
    }
}

void PrintRegReg(int indice,int razon)
{
    FILE *fa;
    TWkr reg;
    int del;
    fa=fopen("datos.dat","rb+");
    if (fa)
    {
        fseek(fa,sizeof(TWkr)*indice,SEEK_SET);
        if (fread(&reg,sizeof(TWkr),1,fa)==1)
        {
            if (reg.status)
            {
                if (razon==ELIMINAR)
                {
                    {
                        printf("\nNumero de Empleado: %d\n",reg.key);
                        printf("Nombre: %s\n",reg.name);
                        printf("Apellido Paterno: %s\n",reg.apPat);
                        printf("Apellido Materno %s\n",reg.apMat);
                        printf("Sexo: %s\n",reg.sexo);
                        printf("Puesto: %s\n",reg.puesto);
                        printf("Estado: %s\n",reg.estado);
                        printf("Edad: %d\n",reg.edad);
                        printf("Telefono: %d\n",reg.telefono);
                        del=validNum("Deseas eliminar al empleado? (0:Si 1:No): ",0,1);
                        if (!del)
                        {
                            fseek(fa,-sizeof(TWkr),SEEK_CUR);
                            fseek(fa,offsetof(TWkr,status),SEEK_CUR);
                            fwrite(&del,sizeof(int),1,fa);
                            printf("\nSe elimino el empleado\n");
                        }
                    }
                }
                else
                {
                    if (razon==BUSCAR)
                    {
                        {
                            printf("\nNumero de Empleado: %d\n",reg.key);
                            printf("Nombre: %s\n",reg.name);
                            printf("Apellido Paterno: %s\n",reg.apPat);
                            printf("Apellido Materno %s\n",reg.apMat);
                            printf("Sexo: %s\n",reg.sexo);
                            printf("Puesto: %s\n",reg.puesto);
                            printf("Estado: %s\n",reg.estado);
                            printf("Edad: %d\n",reg.edad);
                            printf("Telefono: %d\n",reg.telefono);
                        }
                    }
                }
                else
                {
                    {
                        printf("\nEl empleado esta eliminando\n");
                    }
                }
            }
            else
            {
                {
                    printf("\nOcurrio un error al cargar el registro\n");
                }
            }
        }
        fclose(fa);
    }
    else
    {
        {
            printf("\nEl archivo de registros no existe\n");
        }
    }
}

//case 3
int BuscarWrkr(Tindex vect[],int i,int orden)
{
    int pos;
    FILE *fa;
    printf("Buscar empleado\n\n");
    pos=validNum("Ingresa el n°mero de empleado que deseas buscar: ",300000,399999);
    if (orden)
    {
        {
            pos=BuscarBinIndex(vect,i,pos);
            if (pos!=-1)
            {
                {
                    PrintRegReg(pos,BUSCAR);
                }
            }
            else
            {
                {
                    printf("\nEl empleado no existe\n");
                }
            }
        }
    }
    else
    {
        {
            pos=BuscarSecIndex(vect,i,pos);
            if (pos!=-1)
            {
                {
                    PrintRegReg(pos,BUSCAR);
                }
            }
            else
            {
                {
                    printf("\nEl empleado no existe\n");
                }
            }
        }
    }
}

//case 4
void OrdenarIndex(Tindex vect[],int i,int MetodoOrden)
{
    switch (MetodoOrden)
    {
        {
            case 0:
                BubbleSortIndex(vect,i);
            }
    }
}

```

```

        case 1:
            HeapSortIndex(vect,i);
        case 2:
            MergeSortIndex(vect,0,i-1);
            break;
    }
}

void BubbleSortIndex(Tindex vect[],int n)
{
    int i,j;
    int swap;
    for(i=0;i<n-1;i++)
    {
        swap=0;
        for(j=0;j<n-i-1;j++)
        {
            if (vect[j].key>vect[j+1].key)
            {
                SwapIndex(&vect[j],&vect[j+1]);
                swap=1;
            }
        }
        if (swap==0)
        {
            i=n;
        }
    }
}

void HeapSortIndex(Tindex vect[],int n)
{
    int i;
    for (i=n/2-1;i>=0;i--)
    {
        AdjustHeapIndex(vect,n,i);
    }

    for (i=n-1;i>0;i--)
    {
        SwapIndex(&vect[0],&vect[i]);
        AdjustHeapIndex(vect,i,0);
    }
}

void AdjustHeapIndex(Tindex vect[],int n,int raiz)
{
    int may=raiz;
    int izq =2 *raiz+1;
    int derch =2*raiz+2;

    if (izq<n)
    {
        if (vect[izq].key>vect[may].key)
        {
            may=izq;
        }
    }

    if (derch<n)
    {
        if (vect[derch].key>vect[may].key)
        {
            may=derch;
        }
    }

    if (may!=raiz)
    {
        SwapIndex(&vect[raiz],&vect[may]);
        AdjustHeapIndex(vect,n,may);
    }
}

void SwapIndex(Tindex *x,Tindex *y)
{
    Tindex temp = *x;
    *x = *y;
    *y = temp;
}

void MergeSortIndex(Tindex vect[],int izq,int derch)
{
    int med;

    if (izq<derch)
    {
        med=(izq+derch-izq)/2;
        MergeSortIndex(vect,izq,med);
        MergeSortIndex(vect,med+1,derch);
        FusionMergeSortIndex(vect,izq,med,derch);
    }
}

void FusionMergeSortIndex(Tindex vect[],int izq,int medio,int derch)
{
    int i,j,k;
    int n1=medio-izq+1;
    int n2=derch-medio;
    Tindex Lf[n1], R[n2];

    for (i=0;i<n1;i++)
    {
        Lf[i]=vect[izq+i];
    }
    for (j=0;j<n2;j++)
    {
        R[j]=vect[medio+1+j];
    }

    i=0;
    j=0;
    k=izq;
    while (i<n1)
    {
        if (j<n2)
        {
            if (Lf[i].key<=R[j].key)
            {

```

```

        vect[k]=Lf[i];
        i++;
    }
    else
    {
        vect[k]=R[j];
        j++;
    }
    k++;
}
else
{
    break;
}
}
while(i<n1)
{
    vect[k]=Lf[i];
    i++;
    k++;
}
while (j < n2)
{
    vect[k]=R[j];
    j++;
    k++;
}
}
}

//case 5 y 6
void PrintRegTab(Tindex vect[],int i,int &orden,int MetodoOrden)
{
    int op;
    printf("Mostrar Registros\n\n");
    printf("1) Normal\n");
    printf("2) Ordenado\n");
    printf("3) Regresar\n");
    op=validNum("Escoge una opcion: ",1,3);

    switch(op)
    {
        case 1:
            printf("\n");
            PrintArchBin();
            break;
        case 2:
            if(orden)
            {
                printf("\n");
                PrintBinIndex(vect,i);
            }
            else
            {
                printf("\n");
                OrdenarIndex(vect,i,MetodoOrden);
                orden=1;
                PrintBinIndex(vect,i);
            }
            break;
    }
}

void PrintArchBin(void)
{
    FILE *fa;
    TWrkr reg;
    int k=0;
    fa=fopen("datos.dat","rb");
    if(fa)
    {
        printf("%s\n",ENCABEZADO);
        while(fread(&reg,sizeof(TWrkr),1,fa)==1)
        {
            if(reg.status)
            {
                printf("| %-4d ",++k);
                //getch();
                printf("| %6d | %-22s | %-16s | %-17s | %-20s | 646%-7d | %-4d | %-10s | %s | \n",reg.key,reg.puesto,reg.apPat,reg.apMat,reg.name,reg.telefono,reg.edad,reg.sexo,reg.est);
            }
        }
        fclose(fa);
    }
    else
    {
        printf("\nEl archivo de registros no existe\n");
    }
}

void PrintBinIndex(Tindex vect[],int i)
{
    FILE *fa;
    TWrkr reg;
    int j,k=0;
    fa=fopen("datos.dat","rb");
    if(fa)
    {
        for(j=0;j<i;j++)
        {
            fseek(fa,0,SEEK_SET);
            fseek(fa,sizeof(TWrkr)*vect[j].indice,SEEK_SET);
            if(fread(&reg,sizeof(TWrkr),1,fa)==1)
            {
                if(reg.status)
                {
                    printf("| %-4d ",++k);
                    printf("| %6d | %-22s | %-16s | %-17s | %-20s | 646%-7d | %-4d | %-10s | %s | \n",reg.key,reg.puesto,reg.apPat,reg.apMat,reg.name,reg.telefono,reg.edad,reg.sexo,reg.est);
                }
            }
        }
        fclose(fa);
    }
    else
    {
        printf("\nEl archivo de registros no existe\n");
    }
}

//case 7
void GenArchivoTxt(Tindex vect[],int i,int &orden,int MetodoOrden)

```

```

{
    FILE *fa;
    int bname;
    int op;
    char name[LN];
    int len;
    printf("Generar Archivo de Texto\n\n");
    printf("1) Normal\n");
    printf("2) Ordenado\n");
    printf("3) Regresar\n");
    op=validNum("Escoge una opcion: ",1,3);

    if(op<3)
    {
        do
        {
            bname=0;
            printf("\n\nIngresa el nombre del archivo de texto: ");
            validStr(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("\nYa existe un archivo con ese nombre\n");
                fclose(fa);
                bname=1;
            }
        }while(bname);
        fclose(fa);

    }

    switch(op)
    {
        case 1:
            printf("\n\n");
            GenArchTxtNormal(name);
            printf("\nArchivo %s\n\n",name);
            PrintArchTxt(name);
            break;
        case 2:
            if(ordena)
            {
                printf("\n\n");
                GenArchTxtOrden(vect,i,name);
                PrintArchTxt(name);
            }
            else
            {
                printf("\n\n");
                OrdenarIndex(vect,i,MetodoOrden);
                orden=1;
                GenArchTxtOrden(vect,i,name);
                PrintArchTxt(name);
            }
            break;
    }
}

void GenArchTxtNormal(char name[])
{
    FILE *bin,*txt;
    TWkr reg;
    int k=0;
    bin=fopen("datos.dat","rb");
    if(bin)
    {
        txt=fopen(name,"w");
        fprintf(txt,"%s\n",ENCABEZADO);
        while(fread(&reg,sizeof(TWkr),1,bin)==1)
        {
            if(reg.status)
            {
                fprintf(txt,"%i %4d | %6d | %-22s | %-16s | %-17s | %-20s | 646%-7d | %-4d | %-10s | %s |%n",++k,reg.key,reg.puesto,reg.apPat,reg.apMat,reg.name,reg.telefono,reg.edad,reg.sex);
            }
        }
        fclose(txt);
        fclose(bin);
    }
    else
    {
        printf("\nEl archivo binario de registros no existe\n");
    }
}

void GenArchTxtOrden(Tindex vect[],int i,char name[])
{
    FILE *bin,*txt;
    TWkr reg;
    int j,k=0;
    bin=fopen("datos.dat","rb");
    if(bin)
    {
        txt=fopen(name,"w");
        fprintf(txt,"%s\n",ENCABEZADO);
        for(j=0;j<i;j++)
        {
            fseek(bin,0,SEEK_SET);
            fseek(bin,sizeof(TWkr)*vect[j].indice,SEEK_SET);
            if(fread(&reg,sizeof(TWkr),1,bin)==1)
            {
                if(reg.status)
                {
                    fprintf(txt,"%i %4d ",++k);
                    fprintf(txt,"%6d | %-22s | %-16s | %-17s | %-20s | 646%-7d | %-4d | %-10s | %s |%n",reg.key,reg.puesto,reg.apPat,reg.apMat,reg.name,reg.telefono,reg.edad,reg.sex);
                }
            }
        }
        fclose(txt);
        fclose(bin);
    }
    else
    {
        printf("\nEl archivo binario de registros no existe\n");
    }
}

```

```

}

void PrintArchTxt(char name[])
{
    FILE *fa;
    char c;
    fa=fopen(name,"r");
    if(fa)
    {
        do
        {
            c=fgetc(fa);
            printf("%c",c);
        }while(c!=EOF);
        fclose(fa);
    }
    else
    {
        printf("\nEl archivo de texto %s no existe\n",name);
    }
}

//case 8
void empaquetar(Tindex vect[],int i,int respaldo)
{
    FILE *bin,*binbak,*bindel,*tmp;
    TWkrkr reg;
    char binBakName[30];
    char c[2];
    bin=fopen("datos.dat","rb");
    if(bin)
    {
        fclose(bin);
        rename("datos.dat","datos.tmp");
        tmp=fopen("datos.tmp","rb");
        if(tmp)
        {
            if(respaldo==0)
            {
                bin=fopen("datos.dat","wb");
                binbak=fopen("datos.bak","wb");
                bindel=fopen("datosdel.bak","ab");
                //respaldo
                while(fread(&reg,sizeof(TWkrkr),1,tmp))
                {
                    if(reg.status)
                    {
                        fwrite(&reg,sizeof(TWkrkr),1,bin);
                        fwrite(&reg,sizeof(TWkrkr),1,binbak);
                    }
                    else
                    {
                        fwrite(&reg,sizeof(TWkrkr),1,bindel);
                        fwrite(&reg,sizeof(TWkrkr),1,binbak);
                    }
                }
                printf("\nSe empaqueto correctamente\n");
                fclose(tmp);
                fclose(bindel);
                fclose(binbak);
                fclose(bin);
            }
            else
            {
                if(respaldo>0)
                {
                    strcpy(binBakName,"datos");
                    strcat(binBakName,itoa(respaldo,c,10));
                    strcat(binBakName,".bak");
                    bin=fopen("datos.dat","wb");
                    binbak=fopen(binBakName,"wb");
                    bindel=fopen("datosdel.bak","a");
                    //respaldo
                    while(fread(&reg,sizeof(TWkrkr),1,tmp))
                    {
                        if(reg.status)
                        {
                            fwrite(&reg,sizeof(TWkrkr),1,bin);
                            fwrite(&reg,sizeof(TWkrkr),1,binbak);
                        }
                        else
                        {
                            //332686
                            fwrite(&reg,sizeof(TWkrkr),1,bindel);
                            fwrite(&reg,sizeof(TWkrkr),1,binbak);
                        }
                    }
                    printf("\nSe empaqueto correctamente\n");
                    fclose(tmp);
                    fclose(bindel);
                    fclose(binbak);
                    fclose(bin);
                }
            }
            remove("datos.tmp");
        }
        else
        {
            printf("\nError al empaquetar\n");
        }
    }
    else
    {
        printf("\nEl archivo binario principal no existe\n");
    }
}
}

```

```
#include <stdio.h>
#include <sys/stat.h>

#define L 30
#define LN 50

typedef int Tkey;

typedef struct _Wrkr{
    int status;
    Tkey NoEmpleado;
    char name[L];
    char apPat[LN];
    char apMat[LN];
    char sexo[15];
    char puesto[L];
    char estado[L];
    int age;
    Tkey telefono;
}TWrkr;

int main()
{
    FILE *fa;
    int cantidad;
    struct stat archivo;
    stat("datos.dat",&archivo);
    cantidad=archivo.st_size/sizeof(TWrkr);
    cantidad*=1.25;
    fa=fopen("NumeroRegistros.txt","w");
    fprintf(fa,"%d",cantidad);
    fclose(fa);
    return 0;
}
```



```

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <conio.h>

#define NM 4
#define STR 100

#define L 30
#define LN 50

/** Structs *****

typedef int Tkey;

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

/** Validacion *****

//Valida un numero entero
//Parametros:
//-txt Cadena a mostrar antes de leer la varia
//-limi Limite inferior aceptado
//-lims Limite superior aceptado
int validNum(char txt[],int limi,int lims);

//Valida una cadena
//Parametros:
//-str Cadena a validar
//Devuelve:
//-invalido Numero de errores
//Solo permite letras
//No permite empezar ni terminar con espacios
//No permite dobles pesacios
//Muestra los errores
void validStr(char str[]);
int simbolo(char c1);
int dieresis(char c);

/** Cadenas *****

//Concatena una cadena con otra
//Parametros:
//-str2 Cadena donde se juntaran las cadenas
//-str1 Cadena que se juntara a la anterior
void strConcat(char str2[],char str1[]);

//Convierte una cadena a mayusculas
//Parametros:
//-str Cadena a convertir a mayusculas
void strM(char str[]);

//Convierte una cadena a minusculas
//Parametros:
//-str Cadena a convertir a minusculas
void strm(char str[]);

//Regresa la longitud de la cadena
//Parametros:
//-str Cadena de la que se quiere saber su longitud
int strlen(char str[]);

//Imprime una cadena
//Parametros:
//-str Cadena que se quiere imprimir
void strprint(char str[]);

```

```

//Copia la cadena en la cadena 2
//Parametros:
//-str Cadena original
//-str2 Cadena que en la que se copiara str
void strcpy(char str2[],char str[]);

//Valida un numero entero
//Parametros:
//-txt Texto a mostrar al leer el numero
//Devuelve:
//Numero validado
//No requiere un limite para funcionar
int validNumUnLimit(char txt[]);

/** Arreglos ****

/** Asignacion **
//Regresan 1, indica que se lleno

//Llena un vector
//Parametros:
//-vect Vector a llenar
//-n Tamaño
//-limi Limite inferior aceptable
//-lims Limite superior aceptable
int scanVector(int vect[],int n,int limi,int lims);

//Llena un vector con numeros aleatorios sin repetir
//Parametros:
//-vect Vector a llenar
//-n Tamaño
//-ri Rango inicial
//-rf Rango final
int randVector(int vect[],int n,int ri,int rf);

//Llena una matriz
//Parametros:
//-matriz Matriz a llenar
//-r Numero de renglones
//-c Numero de columnas
//-limi Limite inferior aceptable
//-lims Limite superior aceptable
int scanMatriz(int matriz[][NM],int r,int c,int limi,int lims);

//Llena una matriz con numeros aleatorios sin repetir
//Parametros:
//-matriz Matriz a llenar
//-r Numero de renglones
//-c Numero de columnas
//-ri Rango inicial
//-rf Rango final
int randMatriz(int matriz[][NM],int r,int c,int ri,int rf);

//Imprimir vector
//Parametros:
//-vect Vector a imprimir
//-n Tamaño
void printVector(int vect[],int n);

//Imprimir matriz
//Parametros:
//-matriz Matriz a imprimir
//-r Numero de renglones
//-c Numero de columnas
void printMatriz(int matriz[][NM],int r,int c);

//Busca un valor en el vector
//Parametros:
//-vect Vector en el que se buscara
//-n Tamaño
//-num Numero a buscar
//Devuelve:
//Posicion si encontro el numero
//-1 Si no encontro el numero
int searchVector(int vect[],int n,int num);

//Busca un valor en una matriz

```

```

//Parametros:
//-matriz   Matriz en la que se buscara
//-pos      Guarda la posicion en i y j
//-r        Numero de renglones
//-c        Numero de columnas
//-num      Numero a buscar
//Devuelve:
//Renglon en que se encontro el numero
//-1 Si no encontro el numero
int searchMatriz(int matriz[][NM],int pos[],int r,int c, int num);

/** Metodos de ordenacion ****

void improvedBubbleSort(int vect[],int n);

/** Manejo de structs ****

/** Ordenar structs

//Ordena un vector de structs
//Parametros:
//-vect     El vector a ordenar
//-izq      Debe recibir un 0
//-derch    Debe recibir n-1, n=tamaño
//No devuelve nada
void mergeSortT(Tdato vect[],int izq,int derch);
void fusionMergeSortT(Tdato vect[],int izq,int medio,int derch);

//Ordena un vector de structs
//Parametros:
//-n Tamaño del vector
//No devuelve nada
void heapSortT(Tdato vect[], int n);
void swapT(Tdato *x,Tdato *y);
void adjustHeapT(Tdato vect[],int n,int raiz);

/** Buscar structs

//Busca por campo key un registro del vector
//El vector debe estar ordenado
//Parametros:
//-vect     Vector de registros
//-i        Numero de registros ocupados
//-num      Numero de key que se desea buscar
//Retorno:
//-i        Posicion si se encontro
//-1        Si no se encontro
int buscarTBin(Tdato vect[],int i,int num);

//Busca por campo key un registro del vector
//Parametros:
//-vect     Vector de registros
//-i        Numero de registros ocupados
//-num      Numero de key que se desea buscar
//Retorno:
//-i        Posicion si se encontro
//-1        Si no se encontro
int buscarTSec(Tdato vect[],int i,int num);

/** Generación automática

//Genera un campo key sin repetir aleatoriamente
//Parametros:
//-vect     Vector de registros
//-i        Numero de registros ocupados
//-ri       Rango inicial
//-rf       Rango final
//Retorno:
//Matricula generada
int genKey(Tdato vect[],int i,int ri,int rf);

//Genera un nombre de hombre aleatoriamente
//Parametros:
//-vect     Vector de registros
//-i        Numero de registros ocupados
//No retorna nignun valor
void genH_Name(Tdato vect[],int i);

```

```

//Genera un nombre de mujer aleatoriamente
//Parametros:
//-vect      Vector de registros
//-i          Numero de registros ocupados
//No retorna nignun valor
void genM_Name(Tdato vect[],int i);

//Genera un apellido aleatoriamente
//Parametros:
//-vect      Vector de registros
//-i          Numero de registros ocupados
//No retorna nignun valor
void genAp(Tdato vect[],int i);

int validNum(char txt[],int limi,int lims)
{
    char r[STR];
    int rint;
    do
    {
        printf("%s",txt);
        fflush(stdin);
        gets(r);
        rint=atoi(r);
        if(rint<limi || rint>lims)
        {
            printf("Valor invalido, ingresalo de nuevo\n");
        }
    }while(rint<limi || rint>lims);
    return rint;
}

void validStr(char str[])
{
    int i,valido,band,espacio;
    do
    {
        fflush(stdin);
        gets(str);
        band=1;
        espacio=1;
        i=0;
        do
        {
            valido=0;
            //fflush(stdin);
            //c=getch();
            if(str[i]>='A')
            {
                if(str[i]<='Z')
                {
                    valido++;
                }
            }

            if(str[i]>='a')
            {
                if(str[i]<='z')
                {
                    valido++;
                }
            }

            if(str[i]==-91) //Ã±
            {
                str[i]='X';
                valido++;
            }

            if(str[i]==-92) //Ã
            {
                str[i]='X';
                valido++;
            }

            if(simbolo(str[i]))

```

```

    {
        valido++;
    }

    if(dieresis(str[i]))
    {
        valido++;
    }

    if(str[i]==-127)
    {
        str[i]='U';
        valido++;
    }
    if(str[i]==-102)
    {
        str[i]='U';
        valido++;
    }
    if(str[i]==' ')
    {
        if(i==0)
        {
            printf("\nla cadena no puede iniciar con espacio\n");
            printf("\nVuela a ingresar la cadena: ");
            espacio=0;
        }
        else
        {
            if(str[i-1]==' ')
            {
                printf("\nla cadena no tener dos espacios consecutivos\n");
                printf("\nVuela a ingresar la cadena: ");
                espacio=0;
            }
            else
            {
                valido++;
            }
        }
    }

    if(str[i]=='\0')
    {
        if(str[i-1]!=' ')
        {
            valido++;
            band=0;
        }
        else
        {
            printf("\nla cadena no puede acabar con espacio\n");
            printf("\nVuela a ingresar la cadena: ");
            espacio=0;
        }
    }
    if(valido!=0)
    {
        i++;
    }
    else
    {
        if(espacio)
        {
            printf("\nla cadena tiene caracteres invalidos\n");
            printf("\nVuela a ingresar la cadena: ");
        }
        band=0;
    }
}while(band);

}while(valido==0);
}

int validNumUnLimit(char txt[])

```

```

{
    char r[STR];
    int rint;
    printf("%s",txt);
    fflush(stdin);
    gets(r);
    rint=atoi(r);
    return rint;
}

void strM(char str[])
{
    int i;
    for(i=0;str[i]!='\0';i++)
    {
        if(str[i]>=97)
        {
            if(str[i]<=122)
            {
                str[i]-=32;
            }
        }
    }
}

void strm(char str[])
{
    int i;
    for(i=0;str[i]!='\0';i++)
    {
        if(str[i]>=65)
        {
            if(str[i]<=90)
            {
                str[i]+=32;
            }
        }
    }
}

int strlen(char str[])
{
    int i=0;
    while(str[i]!='\0')
    {
        i++;
    }
    return i;
}

void strprint(char str[])
{
    int i;
    for(i=0;str[i]!='\0';i++)
    {
        printf("%c",str[i]);
    }
}

void strcpy(char str2[],char str[])
{
    int i=0;
    for(i=0;str[i]!='\0';i++)
    {
        str2[i]=str[i];
    }
    str2[i]='\0';
}

int scanVector(int vect[],int n,int limi,int lims)
{
    int i;
    for(i=0;i<n;i++)
    {
        printf("Ingresa el valor de la posicion: %d\n",i+1);
        vect[i]=validNum("",limi,lims);
    }
}

```

```

        return 1;
    }

int randVector(int vect[],int n,int ri,int rf)
{
    int i,num,range;
    srand(time(NULL));
    range=(rf-ri)+1;
    for(i=0;i<n;i++)
    {
        do
        {
            num=(rand()%range)+ri;
        }while(searchVector(vect,n,num)!=-1);
        vect[i]=num;
    }
    return 1;
}

int scanMatriz(int matriz[][NM],int r,int c,int limi,int lims)
{
    int i,j;
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
        {
            printf("Ingresa el valor de la posicion: [%d][%d]\n",i+1,j+1);
            matriz[i][j]=validNum("",limi,lims);
        }
    }
    return 1;
}

int randMatriz(int matriz[][NM],int r,int c,int ri,int rf)
{
    int i,j,k;
    int n;
    n=r*c;
    int vect[n];
    randVector(vect,n,ri,rf);
    k=0;
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
        {
            matriz[i][j]=vect[k];
            k++;
        }
    }
    return 1;
}

void printVector(int vect[],int n)
{
    int i;
    for(i=0;i<n;i++)
    {
        printf("%d ",vect[i]);
    }
}

void printMatriz(int matriz[][NM],int r,int c)
{
    int i,j;
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
        {
            printf("%4d ",matriz[i][j]);
        }
        printf("\n");
    }
}

int searchVector(int vect[],int n,int num)
{
    int i;

```

```

for(i=0;i<n;i++)
{
    if(vect[i]==num)
    {
        return i;
    }
}
return -1;
}

int searchMatriz(int matriz[][NM],int pos[],int r,int c,int num)
{
    int i,j;
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
        {
            if(matriz[i][j]==num)
            {
                pos[0]=i;
                pos[1]=j;
                return i;
            }
        }
    }
    return -1;
}

void improvedBubbleSort(int vect[],int n)
{
    int i,j;
    int temp,swap;
    for(i=0;i<n-1;i++)
    {
        swap=0;
        for(j=0;j<n-i-1;j++)
        {
            if(vect[j]>vect[j+1])
            {
                temp=vect[j];
                vect[j]=vect[j+1];
                vect[j+1]=temp;
                swap=1;
            }
        }
        if (swap==0)
        {
            i=n;
        }
    }
}

void strConcat(char str2[],char str1[])
{
    int j,k=0;
    j=strlen(str2);
    str2[j++]=' ';
    while(str1[k]!='\0')
    {
        str2[j]=str1[k];
        k++;
        j++;
    }
    str2[j]='\0';
}

void mergeSortT(Tdato vect[],int izq,int derch)
{
    int med;

    if(izq<derch)
    {
        med=izq+(derch-izq)/2;
        mergeSortT(vect,izq,med);
        mergeSortT(vect,med+1,derch);
        fusionMergeSortT(vect,izq,med,derch);
    }
}

```



```

}

void fusionMergeSortT(Tdato vect[],int izq,int medio,int derch)
{
    int i,j,k;
    int n1=medio-izq+1;
    int n2=derch-medio;
    Tdato Lf[n1], R[n2];

    for (i=0;i<n1;i++)
    {
        Lf[i]=vect[izq+i];
    }
    for (j=0;j<n2;j++)
    {
        R[j]=vect[medio+1+j];
    }

    i=0;
    j=0;
    k=izq;
    while (i<n1)
    {
        if (j<n2)
        {
            if (Lf[i].key<=R[j].key)
            {
                vect[k]=Lf[i];
                i++;
            }
            else
            {
                vect[k]=R[j];
                j++;
            }
            k++;
        }
        else
        {
            break;
        }
    }
    while(i<n1)
    {
        vect[k]=Lf[i];
        i++;
        k++;
    }
    while (j < n2)
    {
        vect[k]=R[j];
        j++;
        k++;
    }
}

void swapT(Tdato *x,Tdato *y)
{
    Tdato temp = *x;
    *x = *y;
    *y = temp;
}

void adjustHeapT(Tdato vect[],int n,int raiz)
{
    int may=raiz;
    int izq =2 *raiz+1;
    int derch =2*raiz+2;

    if(izq<n)
    {
        if (vect[izq].key>vect[may].key)
        {
            may=izq;
        }
    }
}

```

```

if (derch < n)
{
    if (vect[derch].key > vect[may].key)
    {
        may = derch;
    }

}

if (may != raiz)
{
    swapT(&vect[raiz], &vect[may]);
    adjustHeapT(vect, n, may);
}
}

void heapSortT(Tdato vect[], int n)
{
    int i;
    for (i = n/2 - 1; i >= 0; i--)
    {
        adjustHeapT(vect, n, i);
    }

    for (i = n - 1; i > 0; i--)
    {
        swapT(&vect[0], &vect[i]);
        adjustHeapT(vect, i, 0);
    }
}

int buscarTBin(Tdato vect[], int i, int num)
{
    int izq, drcha;
    int med;
    izq = 0;
    drcha = i - 1;
    while (izq <= drcha)
    {
        med = izq + (drcha - izq) / 2;
        if (vect[med].key == num)
        {
            return med;
        }

        if (vect[med].key < num)
        {
            izq = med + 1;
        }
        else
        {
            drcha = med - 1;
        }
    }
    return -1;
}

int buscarTSec(Tdato vect[], int i, int num)
{
    int j;
    for (j = 0; j < i; j++)
    {
        if (num == vect[j].key)
        {
            return j;
        }
    }
    return -1;
}

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;
}

void genH_Name(Tdato vect[],int i)
{
    char h_name[100][LN]={
        "JOSE",
        "JUAN",
        "LUIS",
        "MANUEL",
        "ANTONIO",
        "JESUS",
        "CARLOS",
        "FRANCISCO",
        "ALBERTO",
        "JORGE",
        "MIGUEL",
        "ANGEL",
        "JAVIER",
        "ALEJANDRO",
        "ENRIQUE",
        "VICTOR",
        "ARTURO",
        "CESAR",
        "FERNANDO",
        "PEDRO",
        "MARTIN",
        "ROBERTO",
        "EDUARDO",
        "MARIO",
        "ARMANDO",
        "SERGIO",
        "RAUL",
        "ALFREDO",
        "RAFAEL",
        "RICARDO",
        "HECTOR",
        "OSCAR",
        "GERARDO",
        "DAVID",
        "DANIEL",
        "HUGO",
        "JAIME",
        "JULIO",
        "RUBEN",
        "RAMON",
        "MARCO",
        "GABRIEL",
        "EDGAR",
        "GUADALUPE",
        "ALFONSO",
        "GUILLERMO",
        "SALVADOR",
        "OMAR",
        "IVAN",
        "HUMBERTO",
        "FELIPE",
        "ERNESTO",
        "PABLO",
        "IGNACIO",
        "GUSTAVO",
        "ANDRES",
        "ADRIAN",
        "JOEL",
        "AGUSTIN",
        "RODOLFO",
        "GILBERTO",
        "ROGELIO",
        "RENE",
        "TOMAS",
        "SAUL",
        "ISRAEL",
        "OCTAVIO",
        "VICENTE",
        "NOE",
        "GREGORIO",
    }
}

```

```

        "ISMAEL",
        "NICOLAS",
        "BENJAMIN",
        "MOISES",
        "SANTIAGO",
        "EFRAIN",
        "ALONSO",
        "ABEL",
        "WILBERT",
        "ALVARO",
        "FELIX",
        "MARCOS",
        "ADOLFO",
        "RODRIGO",
        "RAMIRO",
        "SAMUEL",
        "JOAQUIN",
        "ABRAHAM",
        "ESTEBAN",
        "ULISES",
        "RAYMUNDO",
        "FIDEL",
        "LORENZO",
        "GERMAN",
        "MAURICIO",
        "LEONARDO",
        "JOSUE",
        "EMMANUEL",
        "JULIAN",
        "SANTOS",
};

int prob;
prob=rand()%10;
if (prob<=7)
{
    strcpy(vect[i].nombre,h_name[rand()%100]);
}
else
{
    strcpy(vect[i].nombre,h_name[rand()%100]);
    strConcat(vect[i].nombre,h_name[rand()%100]);
}
}

void genM_Name(Tdato vect[],int i)
{
    char m_name[100][LN]={
        "MARIA",
        "GUADALUPE",
        "ROSA",
        "MARTHA",
        "ANA",
        "PATRICIA",
        "LETICIA",
        "ELENA",
        "LAURA",
        "ELIZABETH",
        "ISABEL",
        "ALICIA",
        "MARGARITA",
        "LUZ",
        "ANGELICA",
        "CLAUDIA",
        "SILVIA",
        "NORMA",
        "JUANA",
        "GABRIELA",
        "ADRIANA",
        "VERONICA",
        "TERESA",
        "ALEJANDRA",
        "ALMA",
        "BEATRIZ",
        "YANET",
        "YOLANDA",
        "BLANCA",
        "SANDRA",
    };

```

```

"ARACELI",
"IRMA",
"GLORIA",
"CARMEN",
"LUISA",
"ROCIO",
"CONCEPCION",
"CRISTINA",
"EDITH",
"ESTHER",
"LILIA",
"LORENA",
"MAGDALENA",
"CECILIA",
"JOSEFINA",
"DOLORES",
"KARINA",
"LUCIA",
"ESTELA",
"SUSANA",
"MARIABEL",
"LUCINA",
"LOURDES",
"ERIKA",
"MONICA",
"AGUSTINA",
"DIANA",
"EUGENIA",
"ALEJANDRINA",
"PILAR",
"FRANCISCA",
"VIRGINIA",
"LILIANA",
"GRACIELA",
"BERTHA",
"OLGA",
"MIRIAM",
"FABIOLA",
"DELIA",
"CAROLINA",
"KARLA",
"HILDA",
"ANTONIA",
"MARICELA",
"SONIA",
"NANCY",
"BERENICE",
"YADIRA",
"MAYRA",
"JUDITH",
"MARISOL",
"ROSARIO",
"REYNA",
"AURORA",
"OLIVIA",
"RAQUEL",
"IRENE",
"ELVIRA",
"CATALINA",
"LIDIA",
"VICTORIA",
"DULCE",
"ISELA",
"SARA",
"MARCELA",
"ELVIA",
"ESPERANZA",
"GUILLERMINA",
"DORA",
"EVA",

};

int prob;
prob=rand()%10;
if (prob<=7)
{
    strcpy(vect[i].nombre,m_name[rand()%100]);

```

```

    }
    else
    {
        strcpy(vect[i].nombre,m_name[rand()%100]);
        strConcat(vect[i].nombre,m_name[rand()%100]);
    }
}

void genAp(Tdato vect[],int i)
{
    char ap[100][L]={
        "HERNANDEZ",
        "GARCIA",
        "MARTINEZ",
        "LOPEZ",
        "GONZALEZ",
        "RODRIGUEZ",
        "PEREZ",
        "SANCHEZ",
        "RAMIREZ",
        "FLORES",
        "CRUZ",
        "GOMEZ",
        "MORALES",
        "VAZQUEZ",
        "JIMENEZ",
        "REYES",
        "DIAZ",
        "TORRES",
        "GUTIERREZ",
        "RUIZ",
        "AGUILAR",
        "MENDOZA",
        "CASTILLO",
        "ORTIZ",
        "MORENO",
        "RIVERA",
        "RAMOS",
        "ROMERO",
        "JUAREZ",
        "ALVAREZ",
        "MENDEZ",
        "CHAVEZ",
        "HERRERA",
        "MEDINA",
        "DOMINGUEZ",
        "CASTRO",
        "GUZMAN",
        "VARGAS",
        "VELAZQUEZ",
        "SALAZAR",
        "ROJAS",
        "ORTEGA",
        "CORTES",
        "SANTIAGO",
        "GUERRERO",
        "CONTRERAS",
        "BAUTISTA",
        "ESTRADA",
        "LUNA",
        "LARA",
        "RIOS",
        "AVILA",
        "ALVARADO",
        "DE LA CRUZ",
        "SILVA",
        "DELGADO",
        "CARRILLO",
        "SOLIS",
        "SOTO",
        "LEON",
        "FERNANDEZ",
        "CERVANTES",
        "MARQUEZ",
        "ESPINOSA",
        "MEJIA",
        "VEGA",
    }
}

```

```

        "SANDOVAL",
        "CAMPOS",
        "NAVA",
        "CABRERA",
        "IBARRA",
        "ESPINOZA",
        "SANTOS",
        "ACOSTA",
        "CAMACHO",
        "VALDEZ",
        "FUENTES",
        "VARELA",
        "MIRANDA",
        "MALDONADO",
        "ROBLES",
        "ROSAS",
        "MEZA",
        "MOLINA",
        "TREJO",
        "ROSALES",
        "PACHECO",
        "NAVARRO",
        "SALGADO",
        "AGUIRRE",
        "SALAS",
        "VELASCO",
        "CARDENAS",
        "PINEDA",
        "OROZCO",
        "SERRANO",
        "RANGEL",
        "VALENCIA",
        "SOSA",
        "VASQUEZ"
    };
    int prob;
    prob=rand()%10;
    if (prob<9)
    {
        strcpy(vect[i].apPat,ap[rand()%99]);
        strcpy(vect[i].apMat,ap[rand()%99]);
    }
    else
    {
        strcpy(vect[i].apPat,ap[rand()%99]);
        vect[i].apMat[0]='X';
        vect[i].apMat[1]='\0';
    }
}

int simbolo(char c1)
{
    switch(c1)
    {
        case 39:
        case 45:
        case 47:
        case 46:
            return 1;
            break;
        default:
            return 0;
    }
}

int dieresis(char c)
{
    switch(c)
    {
        case -114: //a
        case -45: //e
        case -40: //i
        case -103: //o
        case -124: //a
        case -119: //e
        case -117: //i
        case -108: //o
    }
}

```

```
        return 1;
        break;
    default:
        return 0;
    }
}
```