***EJERCICIO 38.***

#include<stdio.h>

#include<stdlib.h>

typedef struct

{

    char nombre[30];

    int sector;

} emple;

void cargar(FILE \*,emple \*);

void ordenar(emple \*,int);

void mostrar(emple \*,int);

void mostrar2(emple \*,int);

void ordenar2(emple \*,int);

int main()

{

    FILE \* arch;

    emple \*p,reg;

    int sec,cantreg;

    arch = fopen("EMPLADO.DAT","ab+");

    if(arch==NULL)

    {

        printf("No se pudo abrir el archivo\n");

        exit(1);

    }

    fclose(arch);

    arch = fopen("EMPLADO.DAT","rb+");

    if(arch==NULL)

    {

        printf("No se pudo abrir el archivo\n");

        exit(1);

    }

    do

    {

        printf("Ingrese un numero de sector: ");

        scanf("%d",&sec);

        if(sec<0||sec>5)

        {

            printf("Sector incorrecto\n");

        }

    }

    while(sec<0||sec>5);

    while(sec!=0)

    {

        reg.sector=sec;

        printf("Ingrese el nombre del empleado: ");

        fflush(stdin);

        gets(reg.nombre);

        fwrite(&reg,sizeof(emple),1,arch);

do

        {

            printf("Ingrese un numero de sector: ");

            scanf("%d",&sec);

            if(sec<0||sec>5)

            {

                printf("Sector incorrecto\n");

            }

        }

        while(sec<0||sec>5);

    }

    clearerr(arch);

    fseek(arch,0,SEEK\_END);

    cantreg=ftell(arch)/sizeof(emple);

    p=(emple \*)malloc(cantreg\*sizeof(emple));

    if(p==NULL)

    {

        printf("Insuficiente memoria\n");

        exit(-1);

    }

    cargar(arch,p);

    ordenar(p,cantreg);

    mostrar(p,cantreg);

    printf("\nEmpleados agrupados por sector\n");

    mostrar2(p,cantreg);

    ordenar2(p,cantreg);

    printf("\nEmpleados agrupados por sector con nombres ordenados alfabéticamente\n");

    mostrar2(p,cantreg);

    close(arch);

    free(p);

    return 0;

}

***void cargar(FILE \*arch,emple \*p)***

{

    int i=0;

    emple reg;

    clearerr(arch);

    fseek(arch,0,SEEK\_SET);

    fread(&reg,sizeof(emple),1,arch);

    while(!feof(arch))

    {

        \*(p+i)=reg;

        i++;

        fread(&reg,sizeof(emple),1,arch);

    }

}

void ordenar(emple \*p,int tam)

{

    int c,pas;

    emple aux;

    for(pas=1; pas<tam; pas++)

    {

        for(c=0; c<tam-pas; c++)

        {

            if((p+c)->sector<(p+c+1)->sector)

            {

                aux=\*(p+c);

                \*(p+c)=\*(p+c+1);

                \*(p+c+1)=aux;

            }

        }

    }

}

***void mostrar(emple \*p,int tam)***

{

    int i;

    for(i=0; i<tam; i++)

    {

        printf("Sector : %d Nombre: %s\n", (p+i)->sector,(p+i)->nombre);

    }

}

***void ordenar2(emple \*p,int tam)***

{

    int c,pas;

    emple aux;

    for(pas=1; pas<tam; pas++)

    {

        for(c=0; c<tam-pas; c++)

        {

            if((p+c)->sector<(p+c+1)->sector)

            {

                aux=\*(p+c);

                \*(p+c)=\*(p+c+1);

                \*(p+c+1)=aux;

            }

            else

            {

                if((p+c)->sector==(p+c+1)->sector)

                {

                    if(strcmp((p+c)->nombre,(p+c+1)->nombre)>0)

                    {

                        aux=\*(p+c);

                        \*(p+c)=\*(p+c+1);

                        \*(p+c+1)=aux;

                    }

                }

            }

        }

    }

}

***void mostrar2(emple \*p,int tam)***

{

    int i,sector;

    for(i=0; i<tam; i++)

    {

        sector=(p+i)->sector;

        printf("Sector %d\n",sector);

        while(i<tam&&(p+i)->sector==sector)

        {

            printf("Nombre: %s\n",(p+i)->nombre);

            i++;

        }

        i--;

    }

}