Universidad Autónoma de Baja California
Facultad de Ingeniería, Arquitectura y Diseño

Computación

Programación Estructurara/36276

Santos Tirado Martin/00369705 Pedro

Nuñez Yepiz

Actividad N. 13

ARCHIVOS

Ensenada Baja California, 21 de noviembre del 2023

Introducción:

El alumno aprenderá a crear por su cuenta un formato archivo el cual usara en C.

Competencia:

El alumno llevará a cabo los ejercicios planteados con ayuda del profesor que dejo en un documento sobre los Archivos.

Fundamentos:

El alumno aprenderá a manejar y acostumbrarse a los códigos largos y complicados con la ayuda de funciones, vectores, matrices, librerías propias, entre otras; también, el alumno desarrollará el como usar fuentes externas via archivos.

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 más 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 óptimo. **Mostrar los datos en forma de registro** Hay que preguntar que campo quiere Editar, actualizar 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 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 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 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 extensión .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 deberá llamar datos.dll y si no existe deberá indicar)**

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

Procedimiento:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include "Nocurp3_0.h"

#define MAX 2500

int msgs();
void menu();

void agg_autom(Talum vect[], int *n, int max);
void edit_datos(Talum vect[], int n, int band);
void elim_datos(Talum vect[], int n, int band);
void busq_datos(Talum vect[], int n, int band);
void ord_datos(Talum vect[], int n, int *band);
void imp_datos(Talum vect[], int n);
void crear_arch_txt(Talum vect[], int n);
void ver_arch_txt(Talum vect[], int n);
void crear_arch_bin(Talum vect[], int n);
void cargar_arch_bin(Talum vect[], int *n, int max, int *band);

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

int msgs()
{
    int opc;
    system("cls");
    printf("------Menu------\n");
    printf("1.-  Agregar registros\n");
    printf("2.-  Editar registro\n");
    printf("3.-  Eliminar registro\n");
    printf("4.-  Buscar registro\n");
    printf("5.-  Ordenar registros\n");
    printf("6.-  Imprimir registros\n");
    printf("7.-  Generar archivo de texto\n");
    printf("8.-  Ver archivo de texto\n");
    printf("9.-  Generar archivo binario\n");
```

```
    printf("10.- Cargar archivo binario\n");
    printf("0.- Salir\n");
    opc = vali_int(0, 10, "Seleccione una opcion: ", "Opcion invalida");
    return opc;
}

void menu()
{
    Talum alumnos[MAX];
    int opc, i, band_ord, band_carg;
    FILE *arch;
    i = 0;
    band_ord = 0;
    band_carg = 0;

    do{
        opc = msgs();
        switch(opc)
        {
            case 1:
                agg_autom(alumnos, &i, MAX);
                break;
            case 2:
                edit_datos(alumnos, i, band_ord);
                break;
            case 3:
                elim_datos(alumnos, i, band_ord);
                break;
            case 4:
                busq_datos(alumnos, i, band_ord);
                break;
            case 5:
                ord_datos(alumnos, i, &band_ord);
                break;
            case 6:
                imp_datos(alumnos, i);
                break;
            case 7:
                crear_arch_txt(alumnos, i);
                break;
            case 8:
                ver_arch_txt(alumnos, i);
                break;
            case 9:
```

```
                crear_arch_bin(alumnos, i);
                break;
            case 10:
                cargar_arch_bin(alumnos, &i, MAX, &band_carg);
                break;
        }
    }while (opc != 0);
}

void agg_autom(Talum vect[], int *n, int max)
{
    int i, j;

    i = (*n);
    for(j=0; j < 2300; j++)
    {
        system("cls");
        vect[i] = datos_autom(vect, i);
        i++;
    }
    for(j=0; j < 200; j++)
    {
        system("cls");
        vect[i] = datos_autom_inac(vect, i);
        i++;

    }
    printf("Se han generado 100 datos aleatorios correctamente\n");
    system("pause");

    (*n) =  i;
}

void edit_datos(Talum vect[], int n, int band)
{
    long mat;
    int posi, temp_int, correct = 0;
    char *estatus, *sexo, temp_char[35];
    Talum reg;
```

```
    system("cls");
    if(n > 0)
    {
        mat = vali_long(300000, 399999, "Ingrese la matricula a editar: ",
"Matricula invalida");
        if(band == 0)
        {
            posi = busq_sec_talum(vect, n, mat);
        }
        else
        {
            posi = busq_bin_talum(vect, n, mat);
        }
        if(posi != -1)
        {
            system("cls");
            reg = vect[posi];

            if(reg.genero == 0)
            {
                sexo = "MASCULINO";
            }
            else
            {
                sexo = "FEMENINO";
            }
            /*
            if(reg.estatus == 0)
            {
                estatus = "INACTIVO";
            }
            else
            {
                estatus = "ACTIVO";
            }

            do
            {
                printf("Estado actual: %s", estatus);
                temp_int = vali_int(0, 1, "0.- Inactivo \n1.- Activo \nIngrese el
nuevo dato: ", "Valor invalido");
                correct = vali_int(0, 1, "0.- No \n1.- Si \nEs correcto: ",
"Valor invalido");
                system("cls");
```

```
            }while(correct == 0);
            reg.estatus = temp_int;
            */

            if(reg.estatus != 0)
            {
                do
                {
                    printf("Matricula actual: %d\n", reg.mat);
                    temp_int = vali_int(300000, 399999, "Ingrese el nuevo dato:
", "Valor invalido");
                    correct = vali_int(0, 1, "\n0.- No \n1.- Si \nEs correcto: ",
"Valor invalido");
                    system("cls");
                    if(existe_talum_matri(temp_int, vect, n) == 1)
                    {
                        printf("El valor ingresado ya se encuentra
registrado\n");
                        system("pause");
                        system("cls");
                    }
                }while(correct == 0 || existe_talum_matri(temp_int, vect, n) ==
1);
                reg.mat = temp_int;

                do
                {
                    printf("Apellido paterno actual: %s\n", reg.ap_pat);
                    vali_cad_obg(temp_char, "Ingrese el nuevo dato: ");
                    correct = vali_int(0, 1, "\n0.- No \n1.- Si \nEs correcto: ",
"Valor invalido");
                    system("cls");
                }while(correct == 0);
                strcpy(reg.ap_pat, temp_char);

                do
                {
                    printf("Apellido materno actual: %s\n", reg.ap_mat);
                    vali_cad_obg(temp_char, "Ingrese el nuevo dato: ");
```

```
                    correct = vali_int(0, 1, "\n0.- No \n1.- Si \nEs correcto: ",
"Valor invalido");
                    system("cls");
                }while(correct == 0);
                strcpy(reg.ap_mat, temp_char);

                do
                {
                    printf("Nombre actual: %s\n", reg.nombre);
                    vali_cad_obg(temp_char, "Ingrese el nuevo dato: ");
                    correct = vali_int(0, 1, "\n0.- No \n1.- Si \nEs correcto: ",
"Valor invalido");
                    system("cls");
                }while(correct == 0);
                strcpy(reg.nombre, temp_char);

                do
                {
                    printf("Edad actual: %d\n", reg.edad);
                    temp_int = vali_int(17, 65, "Ingrese el nuevo dato: ", "Valor
invalido");
                    correct = vali_int(0, 1, "\n0.- No \n1.- Si \nEs correcto: ",
"Valor invalido");
                    system("cls");
                }while(correct == 0);
                reg.edad = temp_int;

                do
                {
                    printf("Genero actual: %s\n", sexo);
                    temp_int = vali_int(0, 1, "0.- Masculino \n1.- Femenino
\nIngrese el nuevo dato: ", "Valor invalido");
                    correct = vali_int(0, 1, "\n0.- No \n1.- Si \nEs correcto: ",
"Valor invalido");
                    system("cls");
                }while(correct == 0);
                reg.genero = temp_int;

                vect[posi] = reg;
            }
            else
            {
                printf("La matricula que busco no se encuentra registrada\n");
                system("pause");
```

```
            }
        }
        else
        {
            printf("La matricula que busco no se encuentra registrada\n");
            system("pause");
        }
    }
    else
    {
        printf("Cargue o agregue datos antes de editar\n");
        system("pause");
    }
}

void elim_datos(Talum vect[], int i, int band)
{
    long mat;
    int posi, del;
    char *sexo, *estatus;

    system("cls");

    mat = vali_long(300000, 399999, "Ingrese la matricula a eliminar: ",
"Matricula invalida");
    if(band == 0)
    {
        posi = busq_sec_talum(vect, i, mat);
    }
    else
    {
        posi = busq_bin_talum(vect, i, mat);
    }
    if(posi != -1)
    {
        system("cls");
        if(vect[posi].estatus == 0)
        {
            printf("La matricula que busco ya fue eliminada\n");
            system("pause");
```

```
            return;
        }
        else
        {

            if(vect[posi].genero == 0)
            {
                sexo = "HOMBRE";
            }
            else
            {
                sexo = "MUJER";
            }
            if(vect[posi].estatus == 0)
            {
                estatus = "INACTIVO";
            }
            else
            {
                estatus = "ACTIVO";
            }

            if(vect[posi].estatus != 0)
            {
                printf("%-10s %-10s %-20s %-20s %-20s %-5s %-10s\n", "Estatus",
"Matricula", "Apellido paterno", "Apellido materno", "Nombre", "Edad", "Genero");
                printf("%-10s %-10ld %-20s %-20s %-20s %-5d %-10s\n", estatus,
vect[posi].mat, vect[posi].ap_pat, vect[posi].ap_mat, vect[posi].nombre,
vect[posi].edad, sexo);
                del = vali_int(0, 1, "1.- Eliminar registro \n0.- No eliminar
\nSeleccione una opcion: ", "Opcion invalida");
                if(del == 1)
                {
                    vect[posi].estatus = 0;
                    printf("Registro eliminado existosamente\n");
                    system("pause");
                }
            }
            else
            {
                printf("La matricula que busco no se encuentra registrada\n");
                system("pause");
            }
        }
```

```
    }
    else
    {
        printf("La matricula que busco no se encuentra registrada\n");
        system("pause");
    }
}

void busq_datos(Talum vect[], int n, int band)
{
    long mat;
    int posi;
    char *estatus, *sexo;

    system("cls");
    if(n > 0)
    {
        mat = vali_long(300000, 399999, "Ingrese la matricula a buscar: ",
"Matricula invalida");
        if(band == 0)
        {
            posi = busq_sec_talum(vect, n, mat);
        }
        else
        {
            posi = busq_bin_talum(vect, n, mat);
        }
        if(posi != -1)
        {
            system("cls");
            if(vect[posi].genero == 0)
            {
                sexo = "MASCULINO";
            }
            else
            {
                sexo = "FEMENINO";
            }
            if(vect[posi].estatus == 0)
            {
```

```
                estatus = "INACTIVO";
            }
            else
            {
                estatus = "ACTIVO";
            }

            if(vect[posi].estatus != 0)
            {
                printf("%-10s %-10s %-20s %-20s %-20s %-5s %-10s\n", "Estatus",
"Matricula", "Apellido paterno", "Apellido materno", "Nombre", "Edad", "Genero");
                printf("%-10s %-10ld %-20s %-20s %-20s %-5d %-10s\n", estatus,
vect[posi].mat, vect[posi].ap_pat, vect[posi].ap_mat, vect[posi].nombre,
vect[posi].edad, sexo);
                system("pause");
            }
            else
            {
                printf("La matricula que busco no se encuentra registrada\n");
                system("pause");
            }
        }
        else
        {
            printf("La matricula que busco no se encuentra registrada\n");
            system("pause");
        }
    }
    else
    {
        printf("Cargue o agregue datos antes de buscar\n");
        system("pause");
    }
}

void ord_datos(Talum vect[], int n, int *band)
{
    int bande;
    bande = (*band);

    system("cls");
    if(n > 0)
    {
        if(bande == 0)
```

```
        {
            ord_quicksort(vect, 0, n-1);
            bande = 1;
        }
        else
        {
            ord_burb(vect, n);
        }
        printf("Datos ordenados correctamente\n");
        system("pause");
    }
    else
    {
        printf("Cargue o agregue datos antes de ordenarlos\n");
        system("pause");
    }

    (*band) = bande;
}

void imp_datos(Talum vect[], int n)
{
    int j, k;
    char *sexo;

    system("cls");

    if(n > 0)
    {
        j = 0;
        while(j < n)
        {
            printf("%-7s %-10s %-20s %-20s %-20s %-5s %-10s\n","Indice",
"Matricula", "Apellido paterno", "Apellido materno", "Nombre", "Edad", "Genero");
            for(k = 0; k < 40; k++)
            {
                if(vect[j].genero == 0)
                {
                    sexo = "MASCULINO";
                }
```

```
                else
                {
                    sexo = "FEMENINO";
                }
                printf("%-7d %-10ld %-20s %-20s %-20s %-5d %-10s\n", j,
vect[j].mat, vect[j].ap_pat, vect[j].ap_mat, vect[j].nombre, vect[j].edad, sexo);
                j++;
            }
            system("pause");
            system("cls");
        }
    }
    else
    {
        printf("Ingrese datos antes de imprimir\n");
        system("pause");
    }
}

void crear_arch_txt(Talum vect[], int n)
{
    int j;
    char *sexo, *estatus, nombre[25], txt[5];
    Talum reg;
    FILE *arch;

    system("cls");

    if(n > 0)
    {
        vali_cad_obg(nombre, "Ingrese el nombre para el archivo: ");
        strlwr(nombre);
        txt[0] = '.'; txt[1] = 't'; txt[2] = 'x'; txt[3] = 't'; txt[4] = '\0';

        strcat(nombre, txt);

        arch = fopen(nombre, "w");
        if (arch)
        {
            system("cls");

            if (arch == NULL)
            {
                printf ("Este archivo no existe!");
```

```
                return;
            }

            fprintf(arch, "%-7s %-10s %-10s %-20s %-20s %-20s %-5s %-
10s\n","Indice", "Estatus", "Matricula", "Apellido paterno", "Apellido materno",
"Nombre", "Edad", "Genero");
            for(j=0; j < 2500; j++)
            {
                reg = vect[j];
                if(reg.genero == 0)
                {
                    sexo = "HOMBRE";
                }
                else
                {
                    sexo = "MUJER";
                }
                if(vect[j].estatus == 0)
                {
                    estatus = "INACTIVO";
                }
                else
                {
                    estatus = "ACTIVO";
                }
                if(vect[j].estatus != 2)
                {
                    fprintf(arch, "%-7d %-10s %-10ld %-20s %-20s %-20s %-5d %-
10s\n", j, estatus, reg.mat, reg.ap_pat, reg.ap_mat, reg.nombre, reg.edad, sexo);
                }
            }
            fclose(arch);
            printf("Archivo generado exitosamente!\n");
            system("pause");
        }
    }
    else
    {
        printf("Ingrese datos antes de generar un archivo\n");
        system("pause");
```

```
    }
}

void ver_arch_txt(Talum vect[], int n)
{
    Talum reg;
    FILE *arch;
    char caractr, nombre[25], txt[5];
    long i;

    i = 1;
    system("cls");
    vali_cad_obg(nombre, "Ingrese el nombre del archivo a ver: ");
    strlwr(nombre);
    txt[0] = '.'; txt[1] = 't'; txt[2] = 'x'; txt[3] = 't'; txt[4] = '\0';
    strcat(nombre, txt);

    arch = fopen(nombre, "r");
    if(arch)
    {
        while(!feof(arch))
        {
                caractr = fgetc(arch);
                fprintf(stdout,"%c", caractr);
                i++;
                if(i >= 4400)
                {
                    i = 1;
                    printf("\n");
                    system("pause");
                    system("cls");
                    printf("%-7s %-10s %-10s %-20s %-20s %-20s %-5s %-
10s\n","Indice", "Estatus", "Matricula", "Apellido paterno", "Apellido materno",
"Nombre", "Edad", "Genero");
                }
        }
        fclose(arch);
        system("pause");

    }
    else
    {
        printf("El archivo no existe\n");
```

```
    }

}

void crear_arch_bin(Talum vect[], int n)
{
    int j;
    char nombre[25], bin[5];
    Talum reg;
    FILE *arch;

    system("cls");

    if(n > 0)
    {
        vali_cad_obg(nombre, "Ingrese el nombre para el archivo: ");
        strlwr(nombre);
        bin[0] = '.'; bin[1] = 'd'; bin[2] = 'a'; bin[3] = 't'; bin[4] = '\0';
        strcat(nombre, bin);

        arch = fopen(nombre, "wb");
        if (arch)
        {
            system("cls");

            if (arch == NULL)
            {
                printf ("Este archivo no existe!");
                return;
            }

            for(j=0; j < 2500; j++)
            {
                reg = vect[j];
                fwrite(&reg, sizeof(Talum), 1, arch);
            }
            fclose(arch);
            printf("Archivo generado exitosamente!\n");
            system("pause");
        }
```

```
    }
    else
    {
        printf("Ingrese datos antes de generar un archivo\n");
        system("pause");
    }
}

void cargar_arch_bin(Talum vect[], int *n, int max, int *band)
{
    int i , bande;
    FILE *arch;
    Talum reg;
    char basura[30], nombre[25], bin[5], *estatus;

    system("cls");

    i = (*n);
    bande = (*band);

    vali_cad_obg(nombre, "Ingrese el nombre del archivo a abrir: ");
    strlwr(nombre);
    bin[0] = '.'; bin[1] = 'd'; bin[2] = 'a'; bin[3] = 't'; bin[4] = '\0';
    strcat(nombre, bin);

    if (bande == 0)
    {

        arch = fopen(nombre, "rb");

        if (arch)
        {
            while(fread(&reg, sizeof(Talum), 1, arch))
            {

                vect[i] = reg;
                i++;
                if(i == max)
                {
                    printf("No fue posible agregar mas datos");
                    system("pause");
                    return;
                }
            }
```

```
            fclose(arch);
            bande = 1;

            printf("Los datos fueron cargados correctamente\n");
            system("pause");
        }
        else
        {
            printf("No se encontro el archivo a abrir\n");
            system("pause");
        }
    }
    else
    {
        if (bande == 1)
        {
            printf("Los datos ya fueron cargaron\n");
            system("pause");
        }
        else
        {
            printf("No fue posible agregar mas datos\n");
            system("pause");
        }
    }

    (*n) = i;
    (*band) = bande;
}
```

Libreria:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define TRUE 1
#define FALSE 0
#define NUM_COL 4

typedef long Tkey;

typedef struct _talum{
    int estatus;
    Tkey mat;
    char ap_pat[20];
    char ap_mat[20];
    char nombre[40];
    int edad;
    int genero;
}Talum;

int vali_int(int ran_inf, int ran_sup, char msg[], char msg_error[])
{
    int num;
    char char_num[25];

    do{
        printf("%s", msg);
        fflush(stdin);
        gets(char_num);
        num = atoi(char_num);

        if (num < ran_inf || num > ran_sup)
        {
            printf("%s\n", msg_error);
        }
    }while(num < ran_inf || num > ran_sup);

    return num;
}

long vali_long(long ran_inf, long ran_sup, char msg[], char msg_error[])
{
    long num;
```

```
    char char_num[25];

    do{
        printf("%s", msg);
        fflush(stdin);
        gets(char_num);
        num = atoi(char_num);

        if (num < ran_inf || num > ran_sup)
        {
            printf("%s\n", msg_error);
        }
    }while(num < ran_inf || num > ran_sup);

    return num;
}

void vali_num_cad(char cadena[], int ran_inf, int ran_sup, int max, char *msg,
char *msg_error)
{
    char xnum[max];
    int num;

    do{
        printf("%s", msg);
        fflush(stdin);
        gets(xnum);
        num = atoi(xnum);
        if((num < ran_inf || num > ran_sup) || strlen(xnum) > max)
        {
            printf("%s\n", msg_error);
        }
    }while((num < ran_inf || num > ran_sup) || strlen(xnum) > max);

    strcpy(cadena, xnum);

}

int busq_sec(int vect[], int len, int val)
{
```

```
    int i;
    for(i=0; i<len; i++)
    {
        if(val == vect[i])
        {
            return i;
        }
    }
    return -1;
}

char busq_vocal(char cadena[])
{
    int tam, i;
    tam = strlen(cadena);

    for(i=1; i<tam; i++)
    {
        if(cadena[i]=='A' || cadena[i]=='E' || cadena[i]=='I' || cadena[i]=='O'
|| cadena[i]=='U')
        {
            return cadena[i];
        }
    }

    return 'X';
}

char busq_vocal_dela(char cadena[])
{
    int tam, i;
    tam = strlen(cadena);

    for(i=7; i<tam; i++)
    {
        if(cadena[i]=='A' || cadena[i]=='E' || cadena[i]=='I' || cadena[i]=='O'
|| cadena[i]=='U')
        {
            return cadena[i];
        }
    }

    return 'X';
}
```

```
char busq_vocal_del(char cadena[])
{
    int tam, i;
    tam = strlen(cadena);

    for(i=5; i<tam; i++)
    {
        if(cadena[i]=='A' || cadena[i]=='E' || cadena[i]=='I' || cadena[i]=='O'
|| cadena[i]=='U')
        {
            return cadena[i];
        }
    }

    return 'X';
}

char busq_cons(char cadena[])
{
    int tam, i;
    tam = strlen(cadena);

    for(i=1; i<tam; i++)
    {
        if(cadena[i]!='A' && cadena[i]!='E' && cadena[i]!='I' && cadena[i]!='O'
&& cadena[i]!='U')
        {
            return cadena[i];
        }
    }

    return 'X';
}

char busq_cons_dela(char cadena[])
{
    int tam, i;
    tam = strlen(cadena);
```

```
    for(i=7; i<tam; i++)
    {
        if(cadena[i]!='A' && cadena[i]!='E' && cadena[i]!='I' && cadena[i]!='O'
&& cadena[i]!='U')
        {
            return cadena[i];
        }
    }

    return 'X';
}

char busq_cons_del(char cadena[])
{
    int tam, i;
    tam = strlen(cadena);

    for(i=5; i<tam; i++)
    {
        if(cadena[i]!='A' && cadena[i]!='E' && cadena[i]!='I' && cadena[i]!='O'
&& cadena[i]!='U')
        {
            return cadena[i];
        }
    }

    return 'X';
}

int busq_sec_talum(Talum *alum, int i, long matri)
{
    int j;
    for (j = 0; j < i; j++)
    {
        if(alum[j].mat == matri)
        {
            return j;
        }
    }

    return -1;
}

int busq_bin_talum(Talum *alum, int i, long matri)
```

```
{
    int min, max, mid;
    min = 0;
    max = i - 1;

    while(min <= max)
    {
        mid = (min + max)/2;

        if(matri == alum[mid].mat)
        {
            return mid;
        }
        else
        {
            if(matri < alum[mid].mat)
            {
                max = mid - 1;
            }
            else
            {
                min = mid + 1;
            }
        }
    }

    return -1;
}

void ord_burb(Talum alum[], int i)
{
    int j, k;
    long temp = 0;
    Talum tem;

    for (j = 1; j < i; j++)
        {
            for (k = 0; k < i - j; k++)
            {
                if (alum[k].mat > alum[k + 1].mat)
```

```
                {
                    tem = alum[k];
                    alum[k] = alum[k + 1];
                    alum[k + 1] = tem;
                }
            }
        }
}

void swap(long *a, long *b)
{
    long t;

    t = *a;
    *a = *b;
    *b = t;
}

int partition(Talum alum[], int min, int max)
{
    long pivote;
    pivote = alum[max].mat;

    int i;

    i = (min - 1);

    for (int j = min; j < max; j++)
    {
        if (alum[j].mat <= pivote)
        {
            i++;
            swap(&alum[i].mat, &alum[j].mat);
        }
    }

    swap(&alum[i + 1].mat, &alum[max].mat);
    return (i + 1);
}

void ord_quicksort(Talum alum[], int min, int max)
{
    if (min < max)
    {
```

```
    int pi = partition(alum, min, max);

    ord_quicksort(alum, min, pi - 1);

    ord_quicksort(alum, pi + 1, max);
  }
}

int existe_vect(int num, int *vect, int len)
{
    int i;
    for(i=0; i <= len; i++)
    {
        if (num == vect[i])
        {
            return 1;
        }
    }
    return 0;
}

int existe_talum_matri(long matri, Talum *alum, int i)
{
    int j;
    for(j=0; j <= i; j++)
    {
        if (matri == alum[j].mat)
        {
            return 1;
        }
    }
    return 0;
}


int existe_mat(int num, int mat[][NUM_COL], int len)
{
    int i, j;
    for(i=0; i <= len; i++)
```

```
    {
        for(j=0; j <= len; j++)
        {
            if (num == mat[i][j])
            {
                return 1;
            }
        }
    }
    return 0;
}

void uno_a_dos_digitos(char cadena[2])
{
    int xnum_temp;
    xnum_temp = atoi(cadena);

    if(xnum_temp < 10)
    {
        cadena[0] = '0';
        cadena[1] = xnum_temp + '0';
        cadena[2] = '\0';
    }
}

int comp_esp(char cadena[])
{
    int tam;
    int i;

    tam = strlen(cadena);
    if(cadena[0] == '\0')
            return 0;
    for(i = 1; i <= tam; i++)
    {
        if(cadena[i] == ' ' && cadena[i+1] == ' ')
            return -1;
    }

    return 2;
}

int solo_letras(char cadena[])
{
```

```
    int tam;
    int i;

    tam = strlen(cadena);
    if (comp_esp(cadena) == 0)
    {
        return 0;
    }
    for(i =  0; i <= tam; i++)
    {
        if(cadena[i] < 'A' || cadena[i] > 'z')
        {
            return -1;
        }
        if(cadena[i] > 'Z' && cadena[i] < 'a')
        {
            return -1;
        }

        return 0;
    }

    return 2;
}

void vali_cad(char cadena[], char msg[])
{
    int tam, band;

    band = 1;
    do{

        printf("%s", msg);
        fflush(stdin);
        gets(cadena);

        if (solo_letras(cadena) == 0)
        {
            if (comp_esp(cadena) == -1)
            {
```

```
                printf("No se permite doble espacio\n");
                continue;
            }
        }
        else
        {
            if(cadena[0] == ' ')
            {
                printf("No se permiten espacios\n");
                continue;
            }
            else
            {
                printf("Solo puedes ingresar letras\n");
                continue;
            }
        }

        strupr(cadena);
        band = 0;

    }while(band == 1);
}

int comp_esp_obg(char cadena[])
{
    int tam;
    int i;

    tam = strlen(cadena);
    for(i = 1; i <= tam; i++)
    {
        if(cadena[i] == ' ' && cadena[i+1] == ' ')
            return -1;
    }

    return 2;
}

int solo_letras_obg(char cadena[])
{
    int tam;
    int i;
```

```
    tam = strlen(cadena);
    for(i =  0; i <= tam; i++)
    {
        if(cadena[i] < 'A' || cadena[i] > 'z')
        {
            return -1;
        }
        if(cadena[i] > 'Z' && cadena[i] < 'a')
        {
            return -1;
        }

        return 0;
    }

    return 2;
}

void vali_cad_obg(char cadena[], char msg[])
{
    int tam, band;

    band = 1;
    do{

        printf("%s", msg);
        fflush(stdin);
        gets(cadena);

        if (solo_letras_obg(cadena) == 0)
        {
            if (comp_esp_obg(cadena) == -1)
            {
                printf("No se permite doble espacio\n");
                continue;
            }
        }
        else
        {
            if(cadena[0] == ' ')
```

```
            {
                printf("No se permiten espacios\n");
                continue;
            }
            else
            {
                printf("Solo puedes ingresar letras\n");
                continue;
            }
        }

        strupr(cadena);
        band = 0;

    }while(band == 1);
}

void enie(char cadena[])
{
    int i, len;

    len = strlen(cadena);

    for(i=0; i<len; i++)
    {
        if (cadena[i] == -92 || cadena[i] == -91)
        {
            cadena[i] = 'X';
        }
    }
}

void enie_mayus(char cadena[])
{
    int i, len;

    len = strlen(cadena);

    for(i=0; i<len; i++)
    {
        if (cadena[i] == -92)
        {
            cadena[i] = -91;
        }
```

```
    }
}

int dela_del(char ap[])
{
    if(ap[0] == 'D' && ap[1] == 'E' && ap[2] == ' ' && ap[3] == 'L' && ap[4] ==
'A' && ap[5] == ' ')
    {
        return 2;
    }
    else
    {
        if(ap[0] == 'D' && ap[1] == 'E' && ap[2] == 'L' && ap[3] == ' ')
        {
            return 1;
        }
    }

    return 0;
}

Talum datos_autom(Talum alum[], int i)
{
    Talum reg;
    int j, edad, ap_pat, ap_mat, nombre, genero;
    Tkey mat;
    char nombre_h[30][10] = {"IVAN", "ALEJANDRO", "JOHAN", "DANIEL", "FRANCISCO",
"ISAAC", "ALBERTO", "PABLO", "JAVIER", "HUGO", "ALDO", "ADONIS", "JESUS",
"LEONARDO", "MANUEL", "ALVARO", "ADRIAN", "ADAM", "AXEL", "ERICK", "GAEL",
"BRUNO", "DYLAN", "LUCAS", "OLIVER", "MARCO", "HUGO", "YAEL", "FELIX", "GERMAN"};
    char nombre_m[30][10] = {"MARIA", "FERNANDA", "MIA", "PAOLA", "CAMILA",
"MONSERRAT", "VALERIA", "ALICIA", "SOFIA", "GINA", "AURA", "BRUNA", "FRIDA",
"GINEBRA", "JULIETA", "MAGDALENA", "ABRIL", "ALEXA", "ALEJANDRA", "VALENTINA",
"MARTINA", "ALEXIA", "AMANDA", "BIANCA", "ELENA", "ESMERALDA", "TERESA",
"REBECA", "OLIVIA", "DAFNE"};
    char apellidos[30][15] = {"MARTINEZ", "SANCHEZ", "HERNANDEZ", "PEREZ",
"RODRIGUEZ", "LOPEZ", "GARCIA", "RUIZ", "COTA", "JUAREZ", "ALVAREZ", "ANDRADE",
"BENITEZ", "CASTILLO", "CASTRO", "CONTRERAS", "DEL TORO", "DE LA FUENTE", "DIAZ",
"DUARTE", "ESPINOZA", "FERNANDEZ", "FLORES", "MEJIA", "MORA", "PINEDA", "QUISPE",
"ROJAS", "SILVA", "VERA"};
```

```
    do{
        mat = (rand() % 99999) + 300000;
    }while(existe_talum_matri(mat, alum, i) == 1);

    edad = (rand() % 54) + 17;
    genero = rand() % 2;

    ap_pat = rand() % 30;
    ap_mat = rand() % 30;
    nombre = rand() % 30;

    reg.estatus = 1;
    reg.mat = mat;
    strcpy(reg.ap_pat, apellidos[ap_pat]);
    strcpy(reg.ap_mat, apellidos[ap_mat]);
    if(genero == 0)
    {
        strcpy(reg.nombre, nombre_h[nombre]);
    }
    else
    {
        strcpy(reg.nombre, nombre_m[nombre]);
    }
    reg.edad = edad;
    reg.genero = genero;

    return reg;
}

Talum datos_autom_inac(Talum alum[], int i)
{
    Talum reg;
    int j, edad, ap_pat, ap_mat, nombre, genero;
    Tkey mat;
    char nombre_h[30][10] = {"IVAN", "ALEJANDRO", "JOHAN", "DANIEL", "FRANCISCO",
"ISAAC", "ALBERTO", "PABLO", "JAVIER", "HUGO", "ALDO", "ADONIS", "JESUS",
"LEONARDO", "MANUEL", "ALVARO", "ADRIAN", "ADAM", "AXEL", "ERICK", "GAEL",
"BRUNO", "DYLAN", "LUCAS", "OLIVER", "MARCO", "HUGO", "YAEL", "FELIX", "GERMAN"};
    char nombre_m[30][10] = {"MARIA", "FERNANDA", "MIA", "PAOLA", "CAMILA",
"MONSERRAT", "VALERIA", "ALICIA", "SOFIA", "GINA", "AURA", "BRUNA", "FRIDA",
"GINEBRA", "JULIETA", "MAGDALENA", "ABRIL", "ALEXA", "ALEJANDRA", "VALENTINA",
"MARTINA", "ALEXIA", "AMANDA", "BIANCA", "ELENA", "ESMERALDA", "TERESA",
"REBECA", "OLIVIA", "DAFNE"};
```

```
    char apellidos[30][15] = {"MARTINEZ", "SANCHEZ", "HERNANDEZ", "PEREZ",
"RODRIGUEZ", "LOPEZ", "GARCIA", "RUIZ", "COTA", "JUAREZ", "ALVAREZ", "ANDRADE",
"BENITEZ", "CASTILLO", "CASTRO", "CONTRERAS", "DEL TORO", "DE LA FUENTE", "DIAZ",
"DUARTE", "ESPINOZA", "FERNANDEZ", "FLORES", "MEJIA", "MORA", "PINEDA", "QUISPE",
"ROJAS", "SILVA", "VERA"};


    do{
        mat = (rand() % 99999) + 300000;
    }while(existe_talum_matri(mat, alum, i) == 1);

    edad = (rand() % 54) + 17;
    genero = rand() % 2;

    ap_pat = rand() % 30;
    ap_mat = rand() % 30;
    nombre = rand() % 30;

    reg.estatus = 0;
    reg.mat = mat;
    strcpy(reg.ap_pat, apellidos[ap_pat]);
    strcpy(reg.ap_mat, apellidos[ap_mat]);
    if(genero == 0)
    {
        strcpy(reg.nombre, nombre_h[nombre]);
    }
    else
    {
        strcpy(reg.nombre, nombre_m[nombre]);
    }
    reg.edad = edad;
    reg.genero = genero;

    return reg;
}
```

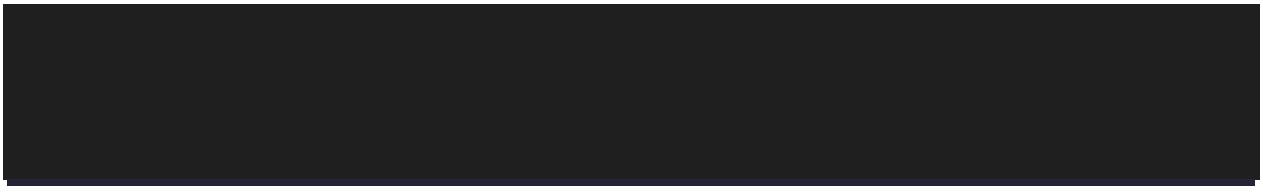
Se uso un cpp aparte:

```
printf("%-10s %-10s %-20s %-20s %-20s %-5s %-10s\n", "Estatus", "Matricula",
"Apellido paterno", "Apellido materno", "Nombre", "Edad", "Genero")
```

```
------Menu------
  1.-  Agregar registros
  2.-  Editar registro
  3.-  Eliminar registro
  4.-  Buscar registro
  5.-  Ordenar registros
  6.-  Imprimir registros
  7.-  Generar archivo de texto
  8.-  Ver archivo de texto
  9.-  Generar archivo binario
  10.- Cargar archivo binario
  0.- Salir
  Seleccione una opcion:
```

```
Se han generado 100 datos aleatorios correctamente
 Press any key to continue . . .
```

Indice	Matricula	Apellido paterno	Apellido materno	Nombre	Edad	Genero
0	311146	ROJAS	PEREZ	VALERIA	42	FEMENINO
1	324795	MARTINEZ	MEJIA	VALERIA	54	FEMENINO
2	326698	RUIZ	RUIZ	ALEJANDRO	54	MASCULINO
3	326945	ESPINOZA	GARCIA	VALERIA	68	FEMENINO
4	323808	RODRIGUEZ	RODRIGUEZ	JOHAN	17	MASCULINO
5	305760	JUAREZ	RODRIGUEZ	ADONIS	54	MASCULINO
6	328735	FLORES	GARCIA	ALVARO	43	MASCULINO
7	326250	SILVA	MEJIA	YAEL	19	MASCULINO
8	325892	FLORES	FERNANDEZ	AURA	64	FEMENINO
9	317749	SANCHEZ	FERNANDEZ	ALDO	22	MASCULINO
10	330160	VERA	RUIZ	ALEXIA	21	FEMENINO
11	306125	COTA	RODRIGUEZ	JAVIER	64	MASCULINO
12	305668	DEL TORO	LOPEZ	AMANDA	66	FEMENINO
13	326070	LOPEZ	CASTRO	MIA	26	FEMENINO
14	323230	BENITEZ	SANCHEZ	DANIEL	68	MASCULINO
15	315527	DIAZ	LOPEZ	MAGDALENA	51	FEMENINO
16	316436	SANCHEZ	COTA	ADAM	18	MASCULINO
17	310723	DEL TORO	SILVA	ADONIS	26	MASCULINO
18	320112	CASTILLO	ALVAREZ	ELENA	45	FEMENINO
19	326394	FLORES	QUISPE	GAEL	55	MASCULINO
20	321771	MORA	MEJIA	YAEL	68	MASCULINO
21	305184	DEL TORO	FERNANDEZ	LUCAS	63	MASCULINO
22	314860	SANCHEZ	JUAREZ	CAMILA	18	FEMENINO
23	307245	ESPINOZA	CASTRO	HUGO	27	MASCULINO
24	307011	ROJAS	LOPEZ	ALEXIA	35	FEMENINO
25	308213	RUIZ	ESPINOZA	MARTINA	39	FEMENINO
26	315041	GARCIA	DE LA FUENTE	DAFNE	55	FEMENINO
27	300873	CONTRERAS	FERNANDEZ	JESUS	24	MASCULINO
28	329307	CASTRO	DEL TORO	ESMERALDA	32	FEMENINO
29	330829	FLORES	SILVA	VALENTINA	38	FEMENINO
30	311903	FLORES	HERNANDEZ	ADAM	66	MASCULINO



2008	316415	RODRIGUEZ	DUARTE	REBECA	41	FEMENINO
2009	323949	ANDRADE	SILVA	BRUNO	25	MASCULINO
2010	331788	CONTRERAS	QUISPE	MANUEL	39	MASCULINO
2011	317730	DUARTE	SILVA	GINEBRA	68	FEMENINO
2012	303744	ESPINOZA	CONTRERAS	MIA	32	FEMENINO
2013	328577	DUARTE	CASTRO	MAGDALENA	21	FEMENINO
2014	330041	DEL TORO	COTA	OLIVER	22	MASCULINO
2015	313234	VERA	FLORES	ADAM	24	MASCULINO
2016	303996	FERNANDEZ	PINEDA	ALEJANDRO	40	MASCULINO
2017	317068	RODRIGUEZ	ALVAREZ	FERNANDA	52	FEMENINO
2018	321169	CONTRERAS	ESPINOZA	OLIVER	59	MASCULINO
2019	306190	FLORES	BENITEZ	ELENA	41	FEMENINO
2020	301212	SILVA	GARCIA	ALEXIA	63	FEMENINO
2021	314011	MORA	GARCIA	MARTINA	70	FEMENINO
2022	310994	ESPINOZA	CASTRO	ADAM	26	MASCULINO
2023	328109	DIAZ	JUAREZ	MARTINA	37	FEMENINO
2024	300788	PEREZ	MEJIA	IVAN	19	MASCULINO
2025	316092	CONTRERAS	HERNANDEZ	ESMERALDA	24	FEMENINO
2026	301747	JUAREZ	HERNANDEZ	OLIVIA	49	FEMENINO
2027	325325	QUISPE	DEL TORO	ADONIS	21	MASCULINO
2028	301275	MORA	JUAREZ	ABRIL	63	FEMENINO
2029	320777	DE LA FUENTE	ESPINOZA	TERESA	24	FEMENINO
2030	302713	PINEDA	RUIZ	ALDO	46	MASCULINO
2031	311918	PEREZ	PINEDA	GAEL	63	MASCULINO
2032	303563	GARCIA	JUAREZ	ADRIAN	37	MASCULINO
2033	310226	HERNANDEZ	JUAREZ	SOFIA	66	FEMENINO
2034	331364	MORA	VERA	MAGDALENA	66	FEMENINO
2035	315186	CASTRO	COTA	AMANDA	24	FEMENINO
2036	302750	JUAREZ	GARCIA	PAOLA	54	FEMENINO
2037	331839	PEREZ	SANCHEZ	MARTINA	60	FEMENINO
2038	306756	ALVAREZ	ROJAS	DAFNE	29	FEMENINO
2039	308266	RUIZ	RUIZ	MIA	29	FEMENINO

```
Archivo generado exitosamente!
Press any key to continue . . .
```

```
Ingrese el nombre del archivo a ver: abc
```

Indice	Estatus	Matricula	Apellido paterno	Apellido materno	Nombre	Edad	Genero
0	ACTIVO	311917	JUAREZ	COTA	ALEJANDRA	41	MUJER
1	ACTIVO	318013	FERNANDEZ	FLORES	AMANDA	61	MUJER
2	ACTIVO	313706	PEREZ	DUARTE	BRUNA	67	MUJER
3	ACTIVO	327978	FLORES	MEJIA	ALVARO	18	HOMBRE
4	ACTIVO	314350	ALVAREZ	JUAREZ	BIANCA	55	MUJER
5	ACTIVO	310213	QUISPE	QUISPE	GINEBRA	28	MUJER
6	ACTIVO	318772	CASTRO	ROJAS	JAVIER	21	HOMBRE
7	ACTIVO	313872	CASTILLO	ANDRADE	PABLO	26	HOMBRE
8	ACTIVO	300281	DE LA FUENTE	MARTINEZ	AURA	61	MUJER
9	ACTIVO	332161	COTA	PEREZ	ADRIAN	37	HOMBRE
10	ACTIVO	329260	FERNANDEZ	DEL TORO	FRANCISCO	63	HOMBRE
11	ACTIVO	306379	ESPINOZA	CONTRERAS	ALEXA	21	MUJER
12	ACTIVO	318170	QUISPE	RUIZ	MAGDALENA	23	MUJER
13	ACTIVO	331502	MARTINEZ	GARCIA	ALBERTO	59	HOMBRE
14	ACTIVO	311640	RODRIGUEZ	GARCIA	AMANDA	69	MUJER
15	ACTIVO	329089	MORA	JUAREZ	GAEL	45	HOMBRE
16	ACTIVO	300721	ESPINOZA	CONTRERAS	GERMAN	24	HOMBRE
17	ACTIVO	304702	DEL TORO	JUAREZ	MAGDALENA	66	MUJER
18	ACTIVO	326226	ESPINOZA	DIAZ	JOHAN	52	HOMBRE
19	ACTIVO	324557	CASTILLO	ROJAS	AMANDA	25	MUJER
20	ACTIVO	308979	CASTILLO	FERNANDEZ	MIA	51	MUJER
21	ACTIVO	312789	FERNANDEZ	MARTINEZ	ALDO	57	HOMBRE
22	ACTIVO	306375	PEREZ	PINEDA	ALEXA	41	MUJER
23	ACTIVO	311360	VERA	MARTINEZ	ALEXIA	50	MUJER
24	ACTIVO	305383	FLORES	VERA	OLIVIA	68	MUJER
25	ACTIVO	320974	DIAZ	MORA	ADRIAN	32	HOMBRE
26	ACTIVO	316924	ANDRADE	DUARTE	ERICK	62	HOMBRE
27	ACTIVO	332252	PEREZ	HERNANDEZ	ESMERALDA	69	MUJER
28	ACTIVO	307756	FLORES	COTA	DYLAN	68	HOMBRE
29	ACTIVO	320885	GARCIA	GARCIA	MARTINA	52	MUJER