Ingeniero en Software y tecnologías emergentes

Materia: Programación Estructurada / Clave **36276**

Alumno: Solano Meza Angel Daniel

Matrícula: 372453

Maestro: Pedro Núñez Yépiz

Actividad No. : 10

Tema - Unidad : Registros - Unidad 5

Ensenada Baja California a 15 de Octubre del 2023

Incluir libreria

```c
// Solano Meza Angel Daniel Matr. 372453
// 15/10/2023
// Uso de registros para almacenar informacion de alumnos
// ADSM_ACT10_932

#include "miti_2.h"
```

```c
#include "miti_2.h"

typedef struct _alumnos
{
    int Status;
    int Matricula;
    char ApPat[30];
    char ApMat[30];
    char Nombre[30];
    int Edad;
    int Sexo;
} Talum;

//*** PROTOTIPOS DE FUNCIONES  ******
int msges();
void menu();
int BusquedaTalum(Talum vector[], int n, int num);
int OrdenarTalum(Talum vector[], int n);
void ImprimirTalum(Talum vect[], int n);
int BusquedaBinaria(Talum vect[], int izquierda, int derecha, int num);
Talum RegistroAuto();
Talum RegistroMan();
//****  main principal  *********
int main()
{
    menu();

    return 0;
}
```

```c
int msges()
{
    int op;
    system("CLS");
    printf("   M  E   N   U \n");
    printf("1.- REGISTROS AUTOMATICOS \n");
    printf("2.- REGISTROS MANUALES \n");
    printf("3.- ELIMINAR REGISTRO \n");
    printf("4.- BUSCAR MATRICULA  \n");
    printf("5.- ORDENAR MATRICULA  \n");
    printf("6.- IMPRIMIR \n");
    printf("0.- SALIR  \n");
    printf("ESCOGE UNA OPCION: ");
    scanf("%d", &op);
    return op;
}
```

Registros automaticos

```c
void menu()
{
    srand(time(NULL));
    Talum VectReg[500], temp;
    int op, j, i, apagar, buscar, encontrado, ordenado;
    i = 0;
    ordenado = 0;
    do
    {
        op = msges();
        switch (op)
        {
        case 1:
            system("CLS");
            for (j = 0; j < 10; j++)
            {
                temp = RegistroAuto();
                while (BusquedaTalum(VectReg, i, temp.Matricula) != -1)
                {
                    temp.Matricula = NumAleatorio(300000, 399999);
                }
                if (i <= 500)
                {
                    VectReg[i++] = temp;
                    ordenado = 0;
                }
            }
            if (i <= 500)
            {
                printf("REGISTROS LLENADOS AUTOMATICAMENTE\n");
            }
            else
            {
                printf("500 REGISTROS OCUPADOS\n");
            }
            system("PAUSE");
            break;
```

```
REGISTROS LLENADOS AUTOMATICAMENTE
Presione una tecla para continuar . . .
```

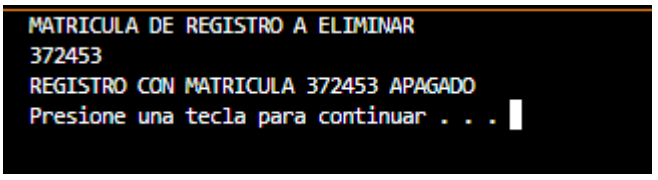
Registro manual

```c
        case 2: // REGISTROS MANUALES
            system("CLS");
            if (i <= 500)
            {
                temp = RegistroMan();
                while (BusquedaTalum(VectReg, i, temp.Matricula) != -1)
                {
                    printf("MATRICULA DUPLICADA, INGRESE UNA NUEVA\n");
                    temp.Matricula = Validar(300000, 399999);
                }
                VectReg[i++] = temp;
                ordenado = 0;
            }
            else
            {
                printf("500 REGISTROS OCUPADOS\n");
            }
            system("PAUSE");
            break;
```

Eliminar registro

```c
case 3: // ELIMINAR REGISTRO
    system("CLS");
    if (i == 0)
    {
        printf("INCAPAZ DE ELIMINAR REGISTROS VACIOS\n");
    }
    else
    {
        printf("MATRICULA DE REGISTRO A ELIMINAR\n");
        apagar = Validar(300000, 399999);
        if (ordenado == 0)
        {
            encontrado = BusquedaTalum(VectReg, i, apagar);
        }
        else
        {
            encontrado = BusquedaBinaria(VectReg, 0, i, apagar);
        }

        if (encontrado == -1)
        {
            printf("MATRICULA NO EXISTENTE\n");
        }
        else // Encontrada
        {
            if (VectReg[encontrado].Status == 0)
            {
                printf("REGISTRO YA ELIMINADO\n");
            }
            else
            {
                VectReg[encontrado].Status = 0;
                printf("REGISTRO CON MATRICULA %d APAGADO\n", apagar);
            }
        }
    }
    system("PAUSE");
    break;
```



```
MATRICULA DE REGISTRO A ELIMINAR
372453
REGISTRO YA ELIMINADO
Presione una tecla para continuar . . .
```

Buscar en registros

```
case 4:
    system("CLS");
    if (i == 0)
    {
        printf("INCAPAZ DE BUSCAR EN REGISTROS VACIOS\n");
    }
    else
    {
        printf("MATRICULA A BUSCAR\n");
        buscar = Validar(300000, 399999);
        if (ordenado == 0)
        {
            encontrado = BusquedaTalum(VectReg, i, buscar);
        }
        else
        {
            encontrado = BusquedaBinaria(VectReg, 0, i, buscar);
        }

        if (encontrado == -1)
        {
            printf("MATRICULA NO ENCONTRADA\n");
        }
        else
        {
            if (VectReg[encontrado].Status == 0)
            {
                printf("MATRICULA DE ALUMNO DESACTIVADO EN: %d\n", encontrado);
            }
            else
            {
                printf("MATRICULA EN REGISTRO: %d\n", encontrado);
            }
        }
    }
    system("PAUSE");
    break;
```

```
MATRICULA A BUSCAR
372453
MATRICULA DE ALUMNO DESACTIVADO EN: 10
Presione una tecla para continuar . . .
```

Ordenar registros

```
        case 5: // ORDENAR REGISTROS
            system("CLS");
            if (i <= 1)
            {
                if (i == 0)
                {
                    printf("INCAPAZ DE ORDENAR UN REGISTRO VACIO\n");
                }
                else
                {
                    printf("UN SOLO REGISTRO ORDENADO\n");
                }
            }
            else
            {
                if (ordenado == 0)
                {
                    ordenado = OrdenarTalum(VectReg, i);
                    printf("REGISTROS ORDENADOS\n");
                }
                else
                {
                    printf("REGISTROS YA ORDENADOS\n");
                }
            }
            system("PAUSE");
            break;
```

```
REGISTROS ORDENADOS
Presione una tecla para continuar . . .
```

```
REGISTROS YA ORDENADOS
Presione una tecla para continuar . . .
```

Imprimir registros

```
        case 6: // IMPRIMIR REGISTROS
            system("CLS");
            if (i == 0)
            {
                printf("REGISTROS VACIOS\n");
            }
            else
            {
                ImprimirTalum(VectReg, i);
            }
            system("PAUSE");
            break;
        }

    } while (op != 0);
}
```

ESTATUS	MATRICULA	NOMBRES	APELLIDO PATERNO	APELLIDO MATERNO	EDAD	SEXO
1	306614	CRISTINA SILVIA	RIOS	ROMERO	27	MUJER
1	311182	PAULA	VIDAL	LUGO	21	MUJER
1	312245	TERESA VALERIA	MONTOYA	PANTOJA	20	MUJER
1	312518	ANA RAQUEL	NAVARRO	ESPINOZA	26	MUJER
1	314271	DANIEL OSCAR	ROBLES	MEDINA	25	HOMBRE
1	315260	PATRICIA ADRIANA	HERRERA	MEDINA	26	MUJER
1	327820	ANA RAQUEL	REYES	MEZA	24	MUJER
1	328300	ANDRES	MONTOYA	DELGADO	19	HOMBRE
1	330267	LAURA	GUERRA	REYES	20	MUJER
1	332696	ANTONIO	BLANCO	GALLEGOS	27	HOMBRE
0	372453	ANGEL DANIEL	SOLANO	MEZA	18	HOMBRE

Funciones

```
//********************
Talum RegistroAuto()
{
    Talum alum;
    int i, apellido, sexo, nombres;

    i = NumAleatorio(0, 14);
    nombres = NumAleatorio(1, 2);
    sexo = NumAleatorio(1, 2);
    if (sexo == 1)
    {
        if (nombres == 1)
        {
            strcpy(alum.Nombre, NombresHombre[i]);
        }
        else
        {
            strcpy(alum.Nombre, NombresHombre[i]);
            strcat(alum.Nombre, SegundoHombre[i]);
        }
    }
    else
    {
        if (nombres == 1)
        {
            strcpy(alum.Nombre, NombresMujer[i]);
        }
        else
        {
            strcpy(alum.Nombre, NombresMujer[i]);
            strcat(alum.Nombre, SegundoMujer[i]);
        }
    }

    alum.Status = 1;
    alum.Matricula = NumAleatorio(300000, 399999);
```

```
    apellido = NumAleatorio(0, 88);
    strcpy(alum.ApPat, Apellidos[apellido]);
    apellido = NumAleatorio(0, 88);
    strcpy(alum.ApMat, Apellidos[apellido]);

    alum.Edad = NumAleatorio(18, 28);
    alum.Sexo = sexo;
    return alum;
}
```

```
Talum RegistroMan()
{
    Talum alum;
    char nombre[30], ap1[30], ap2[30];

    system("CLS");
    printf("ESTATUS\n");
    alum.Status = Validar(0, 1);

    system("CLS");
    printf("MATRICULA\n");
    alum.Matricula = Validar(300000, 399999);

    system("CLS");
    printf("NOMBRE\n");
    fflush(stdin);
    gets(nombre);
    ValiCadena(nombre, "NOMBRE");
    strcpy(alum.Nombre, nombre);

    system("CLS");
    printf("APELLIDO PATERNO\n");
    fflush(stdin);
    gets(ap1);
    ValiCadena(ap1, "APELLIDO PATERNO");
    strcpy(alum.ApPat, ap1);

    system("CLS");
    printf("APELLIDO MATERNO\n");
    fflush(stdin);
    gets(ap2);
    ValiCadena(ap2, "APELLIDO MATERNO");
    strcpy(alum.ApMat, ap2);
```

```
    system("CLS");
    printf("EDAD\n");
    alum.Edad = Validar(18, 50);

    system("CLS");
    printf("SEXO\n");
    alum.Sexo = Validar(1, 2);

    return alum;
}
/**********************
```

```c
int BusquedaTalum(Talum vector[], int n, int num)
{
    int i;
    for (i = 0; i < n; i++)
    {
        if (vector[i].Matricula == num)
        {
            return i;
        }
    }
    return -1;
}
```

```c
int OrdenarTalum(Talum vector[], int n)
{
    Talum temp;
    int i, j;
    for (i = 0; i < n; i++)
    {
        for (j = i + 1; j < n; j++)
        {
            if (vector[j].Matricula < vector[i].Matricula)
            {
                temp = vector[i];
                vector[i] = vector[j];
                vector[j] = temp;
            }
        }
    }
    return 1;
}
```

```c
int BusquedaOrdenadaTalum(Talum vect[], int n, int num)
{
    int i;
    i = 0;
    while (i < n)
    {

        if (num >= vect[i].Matricula)
        {
            if (num == vect[i].Matricula)
            {
                return i;
            }
        }
        else
        {
            return -1;
        }
        i++;
    }
    return -1;
}
```

```c
void ImprimirTalum(Talum vect[], int n)
{
    int i;
    printf("ESTATUS  MATRICULA     NOMBRES                   APELLIDO PATERNO              APELLIDO MATERNO                  EDAD       SEXO\n");
    for (i = 0; i < n; i++)
    {
        printf("  %-2d      %-9d   %-30s   %-30s   %-30s   %-4d      %-6s \n", vect[i].Status, vect[i].Matricula, vect[i].Nombre, vect[i].ApPat, vect[i].ApMat, vect[i].
    }
}
```

```c
int BusquedaBinaria(Talum vect[], int izquierda, int derecha, int num)
{
    while (izquierda <= derecha)
    {
        int medio = izquierda + (derecha - izquierda) / 2;

        if (vect[medio].Matricula == num)
        {
            return medio;
        }
        if (vect[medio].Matricula < num)
        {
            izquierda = medio + 1;
        }
        else
        {
            derecha = medio - 1;
        }
    }
    return -1;
}
```