Universidad Autónoma de Baja California
Facultad de Ingeniería Arquitectura y Diseño

Ingeniero en Software y Tecnologías Emergentes.

Programación estructurada

FUNCIONES Y METODOS DE ORDENACION Y BUSQUEDA ESTRUCTURAS Y LIBRERÍAS

Actividad 10

Danna Guadalupe Sandez Islas

373080

REALICE EL SIGUIENTE PROGRAMA QUE CONTENGA UN MENÚ.

```
do
{
    system("CLS");
    printf("     ----MENU----\n");
    printf("1- Agregar automatico\n");
    printf("2- Agregar manualmente\n");
    printf("3- Eliminar registro\n");
    printf("4- Buscar\n");
    printf("5- Ordenar\n");
    printf("6- Imprimir\n");
    printf("0- Salir\n");
    op = validar("Ingresa una opcion: ", 0, 6);
    system("CLS");
```

Salida:

```
     ----MENU----
1- Agregar automatico
2- Agregar manualmente
3- Eliminar registro
4- Buscar
5- Ordenar
6- Imprimir
0- Salir
Ingresa una opcion:
```

1.- AGREGAR (AUTOM 10 REGISTROS)

Desarrollo:

```
case 1: // llenar registro de alumno automatico 10 veces
    if (k < 500)
    {
        for (int j = 0; j < 10; j++)
        {
            persona = datos_autom(); // guardar los datos de registro en una variable tipo tdatos para evaluar
            while (busq_seq(registros, i, persona.matricula) != -1) // evaluar si la matricula se repite
            {
                persona.matricula = numAleatorio(30000, 399999);
            }
            registros[i++] = persona; // agregar a la persona al registro
            k++;                      // contador para numero de registros creados
        }
        printf("Registro lleno exitosamente con 10 alumnos!\n");
    }
    else
    {
        printf("El registro llego a su maxima capacidad\n");
    }
    ordenado = 0; // bandera para saber si el registro esta ordenado
    system("PAUSE");
    break;
```

Funcion:

```
Tdatos datos_autom(void)
{
    Tdatos persona;
    persona.status = 1;
    persona.matricula = numAleatorio(300000, 399999);
    strcpy(persona.ap_pa, ape[rand() % 99]);
    mayus(persona.ap_pa); // convertir a mayusculas
    strcpy(persona.ap_ma, ape[rand() % 99]);
    mayus(persona.ap_ma); // convertir a mayusculas
    int n, no;
    n = numAleatorio(1, 2);
    no = numAleatorio(1, 2);
    if (n == 1) // es mujer
    {
        if (no == 1) // un nombre
        {
            strcpy(persona.nombre, nom_mujer[rand() % 19]);
        }
        else // dos nombres
        {
            strcpy(persona.nombre, nom_mujer[rand() % 19]);
            strcat(persona.nombre, " ");
            strcat(persona.nombre, nom_mujer[rand() % 19]);
        }
        persona.sexo = 1;
    }
    else // es hombre
    {
        if (no == 1) // un nombre
        {
            strcpy(persona.nombre, nom_hombre[rand() % 19]);
        }
        else // dos nombres
        {
            strcpy(persona.nombre, nom_hombre[rand() % 19]);
            strcat(persona.nombre, " ");
            strcat(persona.nombre, nom_hombre[rand() % 19]);
        }
        persona.sexo = 2;
    }
    mayus(persona.nombre); // convertir mayusculas
    persona.edad = numAleatorio(17, 30);
    return persona;
}
```

```c
/*
    Funcion de busqueda secuencial para buscar en un vector que no este ordenado
    Parametros: arreglo tipo struct, contador del arreglo, valor a buscar.
    Valor de retorno: i si se encuentra, -1 si no se encuentra
*/
int busq_seq(Tdatos registro[], int n, int num)
{
    int i;
    for (i = 0; i < n; i++)
    {
        if (registro[i].matricula == num)
        {
            return i;
        }
    }
    return -1;
}
```

```c
int numAleatorio(int ri, int rf)
{
    int rango = (rf - ri + 1);

    return rand() % rango + ri;
}
```

```c
void mayus(char cadena[]) // mayusculas
{
    int i;
    for (i = 0; cadena[i] != '\0'; i++)
    {
        if (cadena[i] >= 97)
        {
            if (cadena[i] <= 122)
            {
                cadena[i] = cadena[i] - 32;
            }
        }
    }
}
```

Salida:

```
Registro lleno exitosamente con 10 alumnos!
Presione una tecla para continuar . . .
```

2.- AGREGAR MANUAL

Desarrollo:

```
case 2: // llenar manualmente registro de alumno
    if (k < 500)
    {
        persona = datos_manual();                              // agregar el registro a una variable tipo Tdatos
        while (busq_seq(registros, i, persona.matricula) != -1) // validar la matricula
        {
            persona.matricula = validar("Por favor ingresa otra matricula: ", 30000, 399999);
        }
        registros[i++] = persona; // agregar la variable tipo Tdatos al registro
        k++;                      // contador para numero de registros creados
        printf("Registro lleno con exito!\n");
    }
    else
    {
        printf("El registro llego a su maxima capacidad\n");
    }
    ordenado = 0; // bandera para "desactivar" la bandera de registro ordenado
    system("PAUSE");
    break;
```

Funcion:

```
Tdatos datos_manual(void)
{
    Tdatos persona;
    char ape1[20], ape2[20], nom[20];
    int op;

    persona.status = 1;
    persona.matricula = validar("Matricula: ", 300000, 399999);
    system("PAUSE");
    system("CLS");

    do
    {
        system("CLS");
        printf("Apellido paterno: ");
        fflush(stdin);
        gets(ape1);
        strcpy(persona.ap_pa, ape1);
        op = alfabetico(persona.ap_pa); // validar solo alfabaticos y espacio sencillo
        system("PAUSE");
    } while (op != 1);
    mayus(persona.ap_pa); // convertir a mayusculas
    system("CLS");
```

```c
do
{
    system("CLS");
    printf("Apellido materno: ");
    fflush(stdin);
    gets(ape2);
    strcpy(persona.ap_ma, ape2);
    op = alfabetico(persona.ap_ma); // validar solo alfabaticos y espacio sencillo
    system("PAUSE");
} while (op != 1);
mayus(persona.ap_ma); // convertir a mayusculas

system("CLS");

do
{
    system("CLS");
    printf("Nombre: ");
    fflush(stdin);
    gets(nom);
    strcpy(persona.nombre, nom);
    op = alfabetico(persona.nombre); // validar solo alfabaticos y espacio sencillo
    system("PAUSE");
} while (op != 1);
mayus(persona.nombre); // convertir a mayusculas

system("CLS");

persona.edad = validar("Edad: ", 17, 30);
```

```c
    system("PAUSE");
    system("CLS");
    persona.sexo = validar("Sexo:\n1-Mujer\n2-Hombre\nIngresa: ", 1, 2);
    return persona;
}
```

```c
int validar(char mensj[], int ri, int rf)
{
    // variables locales
    int num;
    char cadena[100];
    // desarrollo de funcion
    do
    {
        printf("%s", mensj);
        fflush(stdin);
        gets(cadena);
        num = atoi(cadena); // cambia a numeros la cadena

    } while (num < ri || num > rf);

    return num; // retorna el valor que haya tomado num, entre los rangos dados por el usuario
}
```

```c
/*
    Funcion para validar una cadena unicamente alfabetica, y con espacios sencillos.
    Parametros: cadena a validar.
    Valor de retonor: -1 si es incorrecta, 1 si es correcta
*/
int alfabetico(char cadena[])
{
    int i;
    if (cadena[0] == ' ' || cadena[0] == '\0')
    {
        printf("ERROR\n");
        return -1;
    }
    for (i = 0; cadena[i] != '\0'; i++)
    {
        if ((cadena[i] >= 'a' && cadena[i] <= 'z') || (cadena[i] >= 'A' && cadena[i] <= 'Z') || (cadena[i] == ' ' && cadena[i + 1] != ' '))
        {
        }
        else
        {
            printf("ERROR\n");
            return -1;
        }
    }
    if (cadena[i - 1] == ' ')
    {
        printf("ERROR\n");
        return -1;
    }
    return 1;
}
```

```c
void mayus(char cadena[]) // mayusculas
{
    int i;
    for (i = 0; cadena[i] != '\0'; i++)
    {
        if (cadena[i] >= 97)
        {
            if (cadena[i] <= 122)
            {
                cadena[i] = cadena[i] - 32;
            }
        }
    }
}
```

Salida:

```
Matricula: 373080
```

```
Apellido paterno: sandez
Presione una tecla para continuar . . .
```

```
Apellido materno: islas
Presione una tecla para continuar . . .
```

```
Nombre: danna guadalupe
Presione una tecla para continuar . . .
```

```
Edad: 18
Presione una tecla para continuar . . .
```

```
Sexo:
1-Mujer
2-Hombre
Ingresa: 1
Registro lleno con exito!
Presione una tecla para continuar . . .
```

3- ELIMINAR REGISTRO (lógico)

Desarrollo:

```
case 3: // eliminar registro por matricula
    mat = validar("Ingrese una matricula para eliminar: ", 300000, 399999);
    if (ordenado == 1) // registro ya ordenado, momento de utilizar busqueda binaria
    {
        /* cuado ya esta ordenado se busca con busqueda binaria  */
        encontrado = busq_binaria(registros, 300000, 399999, mat);
        if (encontrado != -1) // si se encuentra
        {
            registros[i].status = 0; // desactivar el status
            printf("La matricula ahora es inactiva\n");
        }
        else // no se encuentra
        {
            printf("La matricula no se encuentra en el registro\n");
        }
    }
    else // no esta ordenado, momento de utilizar busqueda secuencial
    {
        encontrado = busq_seq(registros, i, mat);
        if (encontrado != -1) // si se encuentra la matricula en los registros
        {
            registros[encontrado].status = 0; // desactivar status
            printf("La matricula ahora es inactiva\n");
        }
        else
        {
            printf("La matricula no se encuentra en el registro\n");
        }
    }
    system("PAUSE");
    break;
```

Funciones:

```
/*
    Funcion de busqueda binaria para buscar en un arreglo ya ordenado.
    Parametros: arreglo tipo struct, intervalo inferior, superior, y numero a buscar.
    Valor de retorno: -1 si no se encuentra, y medium si se encontro.
*/
int busq_binaria(Tdatos registro[], int ri, int rf, int matricula)
{
    while (ri <= rf)
    {
        int medium = ri + (rf - ri) / 2;

        // Checa si el numero se encuentra en medio
        if (registro[medium].matricula == matricula)
        {
            return medium;
        }

        if (registro[medium].matricula < matricula) // si el numero es mayor ignora el lado izquierdo
        {
            ri = medium + 1;
        }
        else // si el numero es menor, ignora el lado derecho
        {
            rf = medium - 1;
        }
    }
    // el numero no se encuentra
    return -1;
}
```

```c
/*
    Funcion de busqueda secuencial para buscar en un vector que no este ordenado
    Parametros: arreglo tipo struct, contador del arreglo, valor a buscar.
    Valor de retorno: i si se encuentra, -1 si no se encuentra
*/
int busq_seq(Tdatos registro[], int n, int num)
{
    int i;
    for (i = 0; i < n; i++)
    {
        if (registro[i].matricula == num)
        {
            return i;
        }
    }
    return -1;
}
```

```c
int validar(char mensj[], int ri, int rf)
{
    // variables locales
    int num;
    char cadena[100];
    // desarrollo de funcion
    do
    {
        printf("%s", mensj);
        fflush(stdin);
        gets(cadena);
        num = atoi(cadena); // cambia a numeros la cadena

    } while (num < ri || num > rf);

    return num; // retorna el valor que haya tomado num, entre los rangos dados por el usuario
}
```

SALIDA:

```
Ingrese una matricula para eliminar: 373080
La matricula ahora es inactiva
Presione una tecla para continuar . . .
```

4.- BUSCAR

Desarrollo:

```
case 4: // buscar registro de persona por matricula
    mat = validar("Ingrese una matricula para buscar: ", 300000, 399999);
    if (ordenado == 1)
    {
        encontrado = busq_binaria(registros, 300000, 399999, mat);
        if (encontrado != -1) // si se encuentra
        {
            if (registros[i].status == 0)
            {
                printf("El alumno se encuentra inactivo, matricula inactiva\n");
            }
            else
            {
                printf("Matricula en la posicion [%d]\n", encontrado);
            }
        }
        else // no se encuentra
        {
            printf("La matricula no se encuentra en el registro\n");
        }
    }
    else
    {
        encontrado = busq_seq(registros, i, mat);
        if (encontrado != -1) // si se encuentra la matricula en los registros
        {
            if (registros[encontrado].status == 0) // alumno con status inactivo. NO HAY MATRICULA
            {
                printf("El alumno se encuentra inactivo, matricula inactiva\n");
            }
            else // alumno activo, SI hay matricula
            {
                printf("Matricula en posicion [%d]\n", encontrado);
            }
        }
        else // no se encuentra la matricula
        {
            printf("La matricula no se encuentra en el registro\n");
        }
    }
    system("PAUSE");
    break;
```

Funciones:

```c
/*
    Funcion de busqueda binaria para buscar en un arreglo ya ordenado.
    Parametros: arreglo tipo struct, intervalo inferior, superior, y numero a buscar.
    Valor de retorno: -1 si no se encuentra, y medium si se encontro.
*/
int busq_binaria(Tdatos registro[], int ri, int rf, int matricula)
{
    while (ri <= rf)
    {
        int medium = ri + (rf - ri) / 2;

        // Checa si el numero se encuentra en medio
        if (registro[medium].matricula == matricula)
        {
            return medium;
        }

        if (registro[medium].matricula < matricula) // si el numero es mayor ignora el lado izquierdo
        {
            ri = medium + 1;
        }
        else // si el numero es menor, ignora el lado derecho
        {
            rf = medium - 1;
        }
    }
    // el numero no se encuentra
    return -1;
}
```

```c
/*
    Funcion de busqueda secuencial para buscar en un vector que no este ordenado
    Parametros: arreglo tipo struct, contador del arreglo, valor a buscar.
    Valor de retorno: i si se encuentra, -1 si no se encuentra
*/
int busq_seq(Tdatos registro[], int n, int num)
{
    int i;
    for (i = 0; i < n; i++)
    {
        if (registro[i].matricula == num)
        {
            return i;
        }
    }
    return -1;
}
```

```c
int validar(char mensj[], int ri, int rf)
{
    // variables locales
    int num;
    char cadena[100];
    // desarrollo de funcion
    do
    {
        printf("%s", mensj);
        fflush(stdin);
        gets(cadena);
        num = atoi(cadena); // cambia a numeros la cadena

    } while (num < ri || num > rf);

    return num; // retorna el valor que haya tomado num, entre los rangos dados por el usuario
}
```

Salida:

```
Ingrese una matricula para buscar: 373080
El alumno se encuentra inactivo, matricula inactiva
Presione una tecla para continuar . . .
```

5- ORDENAR

Desarrollo:

```c
case 5: // ordenar registro por matricula
    if (ordenado == 1)
    {
        printf("Registro ya se encuentra ordenado\n");
    }
    else
    {
        ordenado = ordenar(registros, i); // ordenas registro
        printf("Registro ordenado\n");
    }
    system("PAUSE");
    break;
```

Funcion:

```c
/*
    Funcion para ordenar un vector.
    Parametros: arreglo tipo struct, contador del arreglo y numero a buscar.
    Valor de retorno: 1, significa que ya esta ordenado
*/
int ordenar(Tdatos registro[], int n)
{
    int i, j;
    Tdatos temporal;
    for (i = 0; i < n; i++)
    {
        for (j = i + 1; j < n; j++)
        {
            if (registro[j].matricula < registro[i].matricula)
            {
                temporal = registro[i];
                registro[i] = registro[j];
                registro[j] = temporal;
            }
        }
    }
    return 1;
}
```

Salida:

```
Registro ordenado
Presione una tecla para continuar . . .
```

6.- IMPRIMIR

Desarrollo:

```
    case 6: // imprimir registro
        print_reg(registros, i);
        system("PAUSE");
        break;
    }
} while (op != 0);
```

Funcion:

```
/*
    Funcion para imprimir arreglo tipo struct.
    parametros: arreglo tipo struct, contador del arreglo.
*/
void print_reg(Tdatos registro[], int j)
{
    int i;
    printf("No.      STATUS              MATRICULA              NOMBRES             APELLIDO PATERNO             APELLIDO MATERNO                EDAD     SEXO\n\n");
    for (i = 0; i < j; i++)
    {
        if (registro[i].status == 1)
        {
            printf("%-8d   %-18d   %-22d   %-23s   %-29s   %-29s   %-5d   %s\n", i + 1, registro[i].status, registro[i].matricula, registro[i].nombre, registro[i].ap_pa, registro[i].ap_ma,
        }
    }
}
```

```
registro[i].edad, (registro[i].sexo == 1 ? "MUJER" : "HOMBRE"));
```

Salida:

```
No.      STATUS              MATRICULA              NOMBRES             APELLIDO PATERNO             APELLIDO MATERNO                EDAD     SEXO

1        1                   300468                 MIGUEL              GUZMAN                       BLANCO                          25       HOMBRE
2        1                   301128                 FERNANDO            PEREZ                        AVILA                           25       HOMBRE
3        1                   309727                 ANGEL               FERNANDEZ                    VARGAS                          21       HOMBRE
4        1                   309912                 CARMEN ELENA        CASTRO                       SANDEZ                          18       MUJER
5        1                   313091                 GABRIELA CARMEN     GUZMAN                       SANDEZ                          20       MUJER
6        1                   313307                 LUIS                ROJAS                        SOTO                            21       HOMBRE
7        1                   314347                 ELENA               RIOS                         CASTRO                          17       MUJER
8        1                   315970                 LUIS ALEJANDRO      ORTEGA                       CORDERO                         24       HOMBRE
9        1                   317418                 MIGUEL              DIAZ                         FLORES                          24       HOMBRE
10       1                   329206                 SOFIA               CASTRO                       GONZALEZ                        23       MUJER
Presione una tecla para continuar . . .
```

0.- SALIR

Desarrollo:

```
} while (op != 0);
printf("Gracias por usar el programa. Hasta luego!");
return 0;
```

Salida:

```
Gracias por usar el programa. Hasta luego!
PS C:\Users\User\Desktop\o_o\3er SEM\PE\Dapp
```