Licenciatura en Sistemas
Programación de computadoras

Equipo docente: Jorge Golfieri, Natalia Romero, Romina Masilla y Nicolás Perez
Mails: jgolfieri@hotmail.com , nataliab_romero@yahoo.com.ar , romina.e.mansilla@gmail.com, nperez_dcao_smn@outlook.com
Facebook: https://www.facebook.com/groups/171510736842353
Git: ***http://***github.com/UNLASistemasProgramacion/Programacion-de-Computadoras

---

Unidad 5:

Situaciones problemáticas de aplicación: estructuras de datos; declaración; tipos de variables; typedef; array de estructuras.

Bibliografía citada:

C algoritmos, programación y estructuras de datos. Luis Joyanes Aguilar, Andrés Castillo Sanz, y Lucas Sánchez García. Capítulo 11.

Unidad 5: Estructuras

Hasta ahora hemos trabajado con los tipos de datos que nos brinda C, como por ejemplo los int, los float, los char y los vectores o arreglos de estos datos. Pero muchas veces estos datos no alcanzan para describir lo que en verdad necesitamos trabajar, por ejemplo si quereos guardar un alumno, no nos sirve solo guardar su nombre, char[], o guardar solo su dni, int; nos vendría bien guardar todos sus datos en un único tipo de datos; para esto es que se definen las estructuras, para guardar varios datos de forma agrupada, donde este agrupación es creada por el programador.

Una estructura es un tipo de dato definido por el usuario, que se debe declarar antes de que se pueda utilizar.
Es decir, hay que declararlo antes de poder crear una variable de ese tipo.

El formato de la declaracion es:

```
struct <nombre_de_la_estructura>
{
<tipo_variable> <nombre_variable> ;
<tipo_variable> <nombre_variable> ;
...
<tipo_variable> <nombre_variable> ;
};
```

¿Cómo se ve reflejado esto en un ejemplo?:

```
struct InfoLibro
{
char titulo[60];
char autor[30];
```

```
char editorial[30];
int anio;
};
```

Vemos como esta estructura agrupa los datos necesarios para describir un libro, como por ejemplo si título su autor, su editorial y el año.

Para crear variables del tipo que acabamos de definir hay dos formas:

1) Listarlas inmediatamente después de la llave que cierra la declaración de la estructura:

```
struct <nombre_de_la_estructura>
{
<tipo_variable> <nombre_variable> ;
<tipo_variable> <nombre_variable> ;
...
<tipo_variable> <nombre_variable> ;
} estructura1, estructura2;
```

Ejemplo:

```
struct InfoLibro
{
char titulo[60];
char autor[30];
char editorial[30];
int anio;
} libro1, libro 2, libro3;
```

2) Crearlas de la misma forma que creamos los otros tipos de variable.

struct <nombre_de_la_estructura> est1, est2;

Ejemplo:
struct InfoLibro libro1, libro2, libro3;

Si queremos asignar valores a una estructura, podemos hacerlo cargando todos los campos al mismo tiempo o de a uno.

1) Para cargar todos los datos juntos, se puede hacer:

a) En la declaración:

```
struct InfoLibro
{
char titulo[60];
char autor[30];
char editorial[30];
int anio;
} libro1 =
{"Fundamentos de Programación","Luis Joyanes Aguilar","Mc Grauw Hill",2003};
```

b) En la creación de la variable:

```
struct InfoLibro libro1 = {"Fundamentos de Programacion","Luis Joyanes Aguilar","Mc Grauw Hill",2003};
```

2) Para cargar los datos campo por campo, se accede a los mismos usando <nombre_variable>.<nombre_campo>

Ej:

var InfoLibro libro1;

strcpy(libro1.titulo, "Fundamentos de Programacion");
strcpy(libro1.autor, "Luis Joyanes Aguilar");
strcpy(libro1.editorial, "Mc Grauw Hill");
libro1.anio = 2003;

Entonces, si queremos mostrar el año:

printf("El anio es: %d", libro1.anio);

Y si queremos cargar el dato ingresándolo:

printf("Ingrese el anio del libro: ");
scanf("$d",&libro1.anio);

Como una estructura es un tipo de dato similar a un int o un char:

1) Se puede asignar una estructura a otra.

Ej: libro1 = libro3;

2) Se pueden armar vectores de estructuras:

struct InfoLibro libros[100];

-----> libros[0] seria entonces el primer elemento del vector. Para cargarle datos seria:

strcpy(libros[0].titulo, "Fundamentos de Programacion");

```
strcpy(libros[0].autor, "Luis Joyanes Aguilar");
strcpy(libros[0].editorial, "Mc Grauw Hill");
libros[0].anio = 2003;
```

Nota:

Los campos de una estructura pueden ser:
1) Otra estructura ---> es decir, estructuras anidadas.
2) Vectores

Ahora bien, ¿para qué nos pudiera servir este nuevo tema aprendido? Porque no vemos como se ve reflejado el uso de estructuras en nuestro programa con menú para las **ABM de Alumnos.**

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


struct Alumno
        {
                int nroalumno;
                int n1;
                int n2;
                int n3;
                int n4;
                char nombres[15];
        };
```

```
int opcion = 99;
int ultimo = 0;
struct Alumno alumnos[10];

// Esta es una forma de resolverlo,
//llamando funciones dentro de funciones. No es necesario

void Modif(int nro){
   opcion = 99;

   system("CLS");
   printf("Que desea modificar?\n");
   printf("-------------------\n");
   printf("1 - Nro. de Alumno (%d)\n", alumnos[nro].nroalumno);
   printf("2 - Nombre (%s)\n", alumnos[nro].nombres);
   printf("3 - Nota 1 (%d)\n", alumnos[nro].n1);
   printf("4 - Nota 2 (%d)\n", alumnos[nro].n2);
   printf("5 - Nota 3 (%d)\n", alumnos[nro].n3);
   printf("6 - Nota 4 (%d)\n", alumnos[nro].n4);
   printf("0 - Volver a la pantalla anterior\n");

   while(opcion > 6 || opcion < 0){
      printf("Ingrese el numero de orden a modificar: ");
      scanf("%d", &opcion);

      switch (opcion){
            case 0:
               return;
               break;
            case 1:
               printf("Ingrese el numero de alumno nuevo: ");
               scanf("%d", &alumnos[nro].nroalumno);
               break;
            case 2:
```

```
                printf("Ingrese el nombre nuevo: ");
                scanf("%s", alumnos[nro].nombres);
                break;
            case 3:
                printf("Ingrese la Nota 1 nueva: ");
                scanf("%d",&alumnos[nro].n1);
                break;
            case 4:
                printf("Ingrese la Nota 2 nueva: ");
                scanf("%d",&alumnos[nro].n2);
                break;
            case 5:
                printf("Ingrese la Nota 3 nueva: ");
                scanf("%d",&alumnos[nro].n3);
                break;
            case 6:
                printf("Ingrese la Nota 4 nueva: ");
                scanf("%d",&alumnos[nro].n4);
            default:
                printf("La opcion elegida no es valida. Debe volver a elegir! \n");
        }
        system("PAUSE");
    }
}

void Modificaciones(){
    while(1){
        int nro = 99;
        system("CLS");

        while(nro > ultimo || nro < 0){
            printf("Ingrese el numero a modificar (99- Volver al menu): ");
            scanf("%d", &nro);
```

```
            if(nro != 99){
                Modif(nro);
            }else{
                return;
            }
        }

    }
}


void Bajas(){
    while(1){
        int i=0;

        system("CLS");
        printf("Bajas\n");
        printf("--------------------\n");

        int nro = -1;
        while(nro > ultimo || nro < 0){
            printf("Ingrese el numero a dar de baja (99- Volver al menu): ");
            scanf("%d", &nro);

            if(nro != 99){

                for(i=nro;i<ultimo;i++){

                    strcpy(alumnos[i].nombres, alumnos[i+1].nombres);
                    alumnos[i].nroalumno = alumnos[i+1].nroalumno;
                    alumnos[i].n1 = alumnos[i+1].n1;
                    alumnos[i].n2 = alumnos[i+1].n2;
                    alumnos[i].n3 = alumnos[i+1].n3;
                    alumnos[i].n4 = alumnos[i+1].n4;
```

```
            }

            strcpy(alumnos[ultimo].nombres, "");
            alumnos[ultimo].nroalumno = 0;
            alumnos[ultimo].n1 = 0;
            alumnos[ultimo].n2 = 0;
            alumnos[ultimo].n3 = 0;
            alumnos[ultimo].n4 = 0;

            ultimo--;

        }else{
            return;
        }
    }
}


void Altas(){
   PedirDatos();
}

void PedirDatos(){
   system("CLS");
   printf("Altas\n");
   printf("--------------------\n");

   printf("Numero: ");
   scanf("%d", &alumnos[ultimo].nroalumno);
   printf("Nombre: ");
   scanf("%s", alumnos[ultimo].nombres);
   printf("Nota 1: ");
   scanf("%d",&alumnos[ultimo].n1);
```

```
    printf("Nota 2: ");
    scanf("%d",&alumnos[ultimo].n2);
    printf("Nota 3: ");
    scanf("%d",&alumnos[ultimo].n3);
    printf("Nota 4: ");
    scanf("%d",&alumnos[ultimo].n4);

    ultimo++;
};

void Consultas(){
    int i;
    for(i=0;i<10;i++){
        printf("%s", alumnos[i].nombres);
        printf("  %d  %d  %d  %d  %d\n",  alumnos[i].nroalumno,  alumnos[i].n1,
alumnos[i].n2, alumnos[i].n3, alumnos[i].n4);
    }
}
int main()
{
    while(opcion!=0){
        opcion = 99;
        while(opcion > 4 || opcion < 0){
            system("CLS");
            printf("Menu de alumnos\n");
            printf("--------------------\n");
            printf("1 - Altas\n");
            printf("2 - Bajas\n");
            printf("3 - Modificaciones\n");
            printf("4 - Consultas\n");
            printf("0 - Salir del programa\n");
            printf("Opcion seleccionada: ");

            scanf("%d", &opcion);
```

```
        switch (opcion){
            case 0:
                printf("Hasta Pronto!\n");
                break;
            case 1:
                Altas();
                break;
            case 2:
                Bajas();
                break;
            case 3:
                Modificaciones();
                break;
            case 4:
                Consultas();
                break;
            default:
                printf("La opcion elegida no es valida. Debe volver a elegir! \n");
        }
        system("PAUSE");
    }

  }


  return 0;
}
```

Ejercicio 1: Hacer un programa que realice lo siguiente:

a) Definir una estructura Temperaturas definida por tres valores que indican, la temperatura máxima, la mínima y el día que se registraron.

b) Declarar un vector de 20 elementos de la estructura.

c) Llenarlos con valores aleatorios que van de 0º a 40ºC.

d) Llamar a un procedimiento que muestre las temperaturas del vector en grados Farenheit. (ºF = ºC*9/5+32).

e) Mostrar el día con la mínima más baja y el de la máxima más alta.

Ejercicio 2: Crear una estructura que represente un punto en la plano x e y.

Además, crear una estructura que represente un vector (matemático) compuesta de dos estructuras punto.

Se pide, crear una función que reciba dos puntos y calcule la distancia entre los mismos. Además, generar la ecuación de la recta que pasa por estos dos puntos y mostrarla por pantalla por medio de un procedimiento.

Seguimos ejercitando con el uso de estructuras y estructuras anidadas, es decir una estructura dentro de otra, y además le agregamos la complejidad y/o beneficio de definir vectores de estructuras, es decir tener varios datos que comparten el "esqueleto" de los datos pertinentes a la estructura.

Además, nos planteamos que generar nuevos datos, borrarlos o modificarlos tienen que ser acciones gobernadas por los menús es decir acciones seleccionadas por el usuario. Entonces, para finalizar el preámbulo, se puede decir que seguimos reforzando el uso de menús, estructuras, funciones y procedimientos.

Ejercitación 1: Crear hasta 30 clientes, donde cada cliente tiene asignadas 12 cuotas para pagar un producto. El ejercicio debe ser resuelto con el uso de estructuras y se

deben calcular los intereses de las cuotas, usando el interés simple, interés compuesto e interés directo. Deben estar las siguientes funciones:

void Altas();

void Bajas();

void Modificaciones();

void Consultas();

void Modificar(int nro);

void ImprimirFecha(struct Fecha fecha);

void ImprimirCliente(int nroOrden);

void CalcularCuotas(int nro);

void ListaClientes();

void ListaCuotas();

void CuotasInteresSimple(int nroOrden);

void CuotasInteresCompuesto(int nroOrden);

void CuotasInteresDirecto(int nroOrden);

Solución propuesta:

```
#include <stdio.h>

#include <stdlib.h>
```

```
#include <string.h>

/// Definicion de las estructuras que se van a utilizar

struct Fecha{
                    int dia;
                    int mes;
                    int anio;
};

struct Cuota{
      struct Fecha fechaCta;
      float capitalCta;
      float interesCta;
};

struct Cliente{
   int idCliente;
   char nombre[30];
   char producto[30];
   float capital;
   float interes;
   int cantCuotas;
```

```
    struct Fecha fechaCompra;
    struct Cuota cuotas[12];
};

/// Definición de variables globales

struct Cliente clientes[30];
int ultimo = 0;

/// Definicion de procedimientos

void Altas();
void Bajas();
void Modificaciones();
void Consultas();
void Modificar(int nro);

void ImprimirFecha(struct Fecha fecha);
void ImprimirCliente(int nroOrden);

void CalcularCuotas(int nro);
void ListaClientes();
void ListaCuotas();
```

```
void CuotasInteresSimple(int nroOrden);

void CuotasInteresCompuesto(int nroOrden);

void CuotasInteresDirecto(int nroOrden);


/// Programa Principal

int main()

{

   int opcion;

   /// Si no se indica, no se sale del programa

   while(opcion!=0){

      opcion = 99;

      /// Si se un numero que no sea valido, se sigue pidiendo el ingreso

      while(opcion > 4 || opcion < 0){


         /// Listamos las opciones a elegir

         system("CLS");

         printf("Menu de alumnos\n");

         printf("-------------------\n");

         printf("1 - Altas\n");

         printf("2 - Bajas\n");

         printf("3 - Modificaciones\n");

         printf("4 - Consultas\n");

         printf("0 - Salir del programa\n");
```

```
        printf("Opcion seleccionada: ");

        scanf("%d", &opcion);

        switch (opcion){
            case 0:
                printf("\nHasta Pronto!\n\n");
                break;
            case 1:
                Altas();
                break;
            case 2:
                Bajas();
                break;
            case 3:
                Modificaciones();
                break;
            case 4:
                Consultas();
                break;
            default:
                printf("La opcion elegida no es valida. Debe volver a elegir! \n");
        }
```

```
            system("PAUSE");
        }


    }



    return 0;
}

void Altas(){
    system("CLS");
    printf("Altas\n--------------------\n");

    /// Pedimos que se ingresen por teclado los datos de nuevo cliente
    printf("Numero Cliente: ");
    scanf("%d", &clientes[ultimo].idCliente);
    printf("Nombre: ");
    scanf("%s", clientes[ultimo].nombre);
    printf("Producto: ");
    scanf("%s", clientes[ultimo].producto);
    printf("Capital: ");
    scanf("%f",&clientes[ultimo].capital);
    printf("Interes: ");
```

```
    scanf("%f",&clientes[ultimo].interes);
    printf("Cantidad de Cuotas: ");
    scanf("%d",&clientes[ultimo].cantCuotas);
    printf("Fecha compra: \n");
    scanf("%d",&clientes[ultimo].fechaCompra.dia);
    printf("/");
    scanf("%d",&clientes[ultimo].fechaCompra.mes);
    printf("/");
    scanf("%d",&clientes[ultimo].fechaCompra.anio);
    printf("\n");

    /// Llamamos un procedimiento que calcule las cuotas segun el tipo de interes
    CalcularCuotas(ultimo);
    ultimo++;
};

void Bajas(){
    /// Usamos el while(1) para que siempre no salga del procedimiento hasta que lo indiquemos
    while(1){
        system("CLS");
        printf("Bajas\n--------------------\n");
```

```
        int nro = 99;
        /// Si se un numero que no sea valido, se sigue pidiendo el ingreso
        while(nro > ultimo || nro < 0){
            printf("Ingrese el numero de orden a dar de baja (99- Volver al menu): ");
            scanf("%d", &nro);

            /// Si se ingresa un numero valido
            if(nro != 99){
                int i,j;

                /// Pisamos todos los datos del cliente que se quiere dar de baja con los datos del siguiente
                for(i=nro;i<ultimo;i++){

                    clientes[i].idCliente = clientes[i+1].idCliente;
                    strcpy(clientes[i].producto, clientes[i+1].nombre);
                    strcpy(clientes[i].producto, clientes[i+1].producto);
                    clientes[i].capital = clientes[i+1].capital;
                    clientes[i].interes = clientes[i+1].interes;
                    clientes[i].cantCuotas = clientes[i+1].cantCuotas;
                    clientes[i].fechaCompra.dia = clientes[i+1].fechaCompra.dia;
                    clientes[i].fechaCompra.mes = clientes[i+1].fechaCompra.mes;
                    clientes[i].fechaCompra.anio = clientes[i+1].fechaCompra.anio;
```

```
                /// En el caso de las cuotas, como se trata de un vector, tenemos que recorrer todas las posiciones para copiarlo
                for(j=0 ; j <12 ; j++){
                    clientes[i].cuotas[j].fechaCta.dia = clientes[i+1].cuotas[j].fechaCta.dia;
                    clientes[i].cuotas[j].fechaCta.mes = clientes[i+1].cuotas[j].fechaCta.mes;
                    clientes[i].cuotas[j].fechaCta.anio = clientes[i+1].cuotas[j].fechaCta.anio;
                    clientes[i].cuotas[j].capitalCta = clientes[i+1].cuotas[j].capitalCta;
                    clientes[i].cuotas[j].interesCta = clientes[i+1].cuotas[j].interesCta;
                }
            }

            /// El ultimo cliente hay que "vaciarlo"
            clientes[i].idCliente = 0;
            strcpy(clientes[i].producto, "");
            clientes[i].capital = 0;
            clientes[i].cantCuotas = 0;
            clientes[i].interes = 0;
            clientes[i].fechaCompra.dia = 0;
            clientes[i].fechaCompra.mes = 0;
            clientes[i].fechaCompra.anio = 0;

            /// Para vaciar el vector de cuotas tambien lo recorremos
```

```
            for(j=0 ; j <12 ; j++){
                clientes[i].cuotas[j].fechaCta.dia = 0;
                clientes[i].cuotas[j].fechaCta.mes = 0;
                clientes[i].cuotas[j].fechaCta.anio = 0;
                clientes[i].cuotas[j].capitalCta = 0;
                clientes[i].cuotas[j].interesCta = 0;
            }

            ultimo--;

        }else{
            /// Si la opcion elegida es 99, salimos del procedimiento
            return;
        }
    }
}

void Modificaciones(){
   /// Usamos el while(1) para que siempre no salga del procedimiento hasta que lo indiquemos
   while(1){
      int nro = 99;
```

```
        system("CLS");


        /// Si se un numero que no sea valido, se sigue pidiendo el ingreso
        while(nro > ultimo || nro < 0){
            printf("Ingrese el numero de orden a modificar (99- Volver al menu): ");
            scanf("%d", &nro);

            if(nro != 99){
                /// Si se ingresa un numero valido, llamamos un procedimiento para realizar la modificacion deseada
                Modificar(nro);
            }else{
                /// Si la opcion elegida es 99, salimos del procedimiento
                return;
            }
        }

    }
}

void Consultas(){
    system("CLS");
    int opcion;
```

```
    /// Listamos los tipos de informes a consultar
    printf("Que informe desea ver?\n");
    printf("-------------------------\n");
    printf("1 - Listado de Clientes\n");
    printf("2 - Cuotas por Cliente\n");
    printf("0 - Volver a la pantalla anterior\n");

    opcion = 99;
    /// Si se un numero que no sea valido, se sigue pidiendo el ingreso
    while(opcion > 6 || opcion < 0){
        printf("Ingrese el numero de opcion: ");
        scanf("%d", &opcion);

        switch (opcion){
                case 0:
                    return;
                    break;
                case 1:
                    ListaClientes();
                    break;
                case 2:
                    ListaCuotas();
                    break;
```

```
            default:
                printf("La opcion elegida no es valida. Debe volver a elegir! \n");
        }
        system("PAUSE");
    }
}

void Modificar(int nro){
    int opcion;

    system("CLS");

    /// Se listan los datos que se pueden modificar del cliente
    printf("Que desea modificar?\n");
    printf("--------------------\n");
    printf("1 - Nro. de Cliente (%d)\n", clientes[nro].idCliente);
    printf("2 - Nombre (%s)\n", clientes[nro].nombre);
    printf("3 - Producto (%s)\n", clientes[nro].producto);
    printf("4 - Capital (%f)\n", clientes[nro].capital);
    printf("5 - Interes (%f)\n", clientes[nro].interes);
    printf("6 - Cant. Cuotas (%d)\n", clientes[nro].cantCuotas);
    printf("7 - Fecha Compra (%d/%d/%d)\n", clientes[nro].fechaCompra.dia, clientes[nro].fechaCompra.mes, clientes[nro].fechaCompra.anio);
```

```
    printf("8 - Cuotas \n");
    printf("0 - Volver a la pantalla anterior\n");

    opcion = 99;
    /// Si se un numero que no sea valido, se sigue pidiendo el ingreso
    while(opcion > 6 || opcion < 0){
        printf("Ingrese la opcion elegida: ");
        scanf("%d", &opcion);

        /// Segun la opcion elegida, se pide el ingreso del nuevo dato y se modifica el que corresponda.
        switch (opcion){
            case 0:
                return;
                break;
            case 1:
                printf("Ingrese el Nro. de Cliente nuevo: ");
                scanf("%d", &clientes[nro].idCliente);
                break;
            case 2:
                printf("Ingrese el Nombre nuevo: ");
                scanf("%s", clientes[nro].nombre);
                break;
```

```
            case 3:
                printf("Ingrese el Producto nuevo: ");
                scanf("%s", clientes[nro].producto);
                break;
            case 4:
                printf("Ingrese el Capital nuevo: ");
                scanf("%f",&clientes[nro].capital);
                break;
            case 5:
                printf("Ingrese el Interes nuevo: ");
                scanf("%f",&clientes[nro].interes);
                break;
            case 6:
                printf("Ingrese la nueva Cantidad de Cuotas: ");
                scanf("%d",&clientes[nro].cantCuotas);
                CalcularCuotas(nro);
                break;
            case 7:
                printf("Ingrese la Fecha Compra nueva:"); // la primer cuota es a 30 dias
                scanf("%d",&clientes[nro].fechaCompra.dia);
                printf("/");
                scanf("%d",&clientes[nro].fechaCompra.mes);
                printf("/");
```

```
                scanf("%d",&clientes[nro].fechaCompra.anio);
                break;
            case 8:
                CalcularCuotas(nro);
                break;
            default:
                printf("La opcion elegida no es valida. Debe volver a elegir! \n");
        }
        system("PAUSE");
    }
}

/// Armamos un procedimiento auxiliar para imprimir mas facilmente los STRUCT de tipo FECHA
void ImprimirFecha(struct Fecha fecha){
    printf("%d / %d / %d", fecha.dia, fecha.mes, fecha.anio);
}

void CalcularCuotas(int nroOrden){
    int opcion;

    opcion = 99;
```

```
    system("CLS");


    /// Se listan los tipos de interes con los que se pueden
    printf("Que tipo de interes se debe utilizar?\n");
    printf("-------------------------------------\n");
    printf("1 - Interes Simple\n");
    printf("2 - Interes Compuesto\n");
    printf("3 - Interes Directo\n");
    printf("0 - Volver a la pantalla anterior\n");


    /// Si se un numero que no sea valido, se sigue pidiendo el ingreso
    while(opcion > 6 || opcion < 0){
        printf("Ingrese la opcion elegida: ");
        scanf("%d", &opcion);


        /// Segun la opcion elegida se llama un procedimiento que genera las cuotas con el tipo de interes correcto
        switch (opcion){
                case 0:
                    return;
                    break;
                case 1:
                    CuotasInteresSimple(nroOrden);
```

```
                break;
            case 2:
                CuotasInteresCompuesto(nroOrden);
                break;
            case 3:
                CuotasInteresDirecto(nroOrden);
                break;
            default:
                printf("La opcion elegida no es valida. Debe volver a elegir! \n");
        }
        //system("PAUSE");
    }
}

/// Procedimiento auxiliar para imprimir mas facilmente un Cliente
void ImprimirCliente(int nroOrden){
    printf("Numero Cliente: %d ", clientes[nroOrden].idCliente);
    printf("Producto: %s ", clientes[nroOrden].producto);
    printf("Capital: %f ", clientes[nroOrden].capital);
    printf("Intereses: %f ", clientes[nroOrden].interes);
    printf("Cuotas: %d ", clientes[nroOrden].cantCuotas);
    printf("Fecha inicial: ");
    ImprimirFecha(clientes[nroOrden].fechaCompra);
```

```
}

/// Procedimiento para imprimir todos los clientes
void ListaClientes(){
    system("CLS");

    /// Si no se dio de alta ningun cliente, se avisa
    (ultimo == 0)? printf("\nAun no se cargaron datos.\n\n"):(printf("Consultas\n----------
----------\n"));

    int i ;
    printf("\n");

    /// Si hay clientes, se imprimen todos
    for(i=0;i<ultimo;i++){
        ImprimirCliente(i);
    }
    printf("\n");
}

/// Procedimiento para imprimir los datos de un cliente, incluyendo las cuotas
void ListaCuotas(){
    system("CLS");
```

```
    /// Si no se dio de alta ningun cliente, se avisa
    if (ultimo == 0){
        printf("\nAun no se cargaron datos.\n\n");
    } else {
        int i;
        int nro = 99;

        /// Si se un numero que no sea valido, se sigue pidiendo el ingreso
        while(nro > ultimo || nro < 0){
            printf("Ingrese el numero de orden del cliente (99- Volver al menu): ");
            scanf("%d", &nro);

            /// Si es un numero valido, se hace la impresi{on
            if(nro!= 99){

                system("CLS");
                /// Imprimo datos simple del cliente seleccionado.
                ImprimirCliente(nro);
                printf("\n");
                /// Imprimo las cuotas del cliente seleccionado.
                for(i=0;i<clientes[nro].cantCuotas;i++){
                    ImprimirFecha(clientes[nro].cuotas[i].fechaCta);
```

```
                printf("\t %.2f  \t", clientes[nro].cuotas[i].capitalCta);
                printf("%.2f \n",clientes[nro].cuotas[i].interesCta);
            }
        } else{
            return;
        }
    }
}

void CuotasInteresSimple(int nroOrden){
    int i;
    struct Fecha fechaAux = clientes[nroOrden].fechaCompra;

    float capitalCta, interesCta;

    capitalCta = clientes[nroOrden].capital / clientes[nroOrden].cantCuotas;
    interesCta = capitalCta * (clientes[nroOrden].interes/100);

    for(i = 0; i < clientes[nroOrden].cantCuotas; i++){
        fechaAux.mes++;
        if (fechaAux.mes==13) {
            fechaAux.mes=1;
```

```
            fechaAux.anio++;
        }

        clientes[nroOrden].cuotas[i].fechaCta = fechaAux;
        clientes[nroOrden].cuotas[i].capitalCta = capitalCta;
        clientes[nroOrden].cuotas[i].interesCta = interesCta;
    }

}

void CuotasInteresCompuesto( int nroOrden){
    int i;

    struct Fecha fechaAux = clientes[nroOrden].fechaCompra;

    clientes[nroOrden].cuotas[0].capitalCta = clientes[nroOrden].capital / clientes[nroOrden].cantCuotas;
    clientes[nroOrden].cuotas[0].interesCta = clientes[nroOrden].cuotas[0].capitalCta * (clientes[nroOrden].interes/100);

    for(i = 1; i < clientes[nroOrden].cantCuotas; i++){
        fechaAux.mes++;
        if (fechaAux.mes==13) {
```

```
            fechaAux.mes=1;
            fechaAux.anio++;
        }

        clientes[nroOrden].cuotas[i].fechaCta = fechaAux;
        clientes[nroOrden].cuotas[i].capitalCta = clientes[nroOrden].cuotas[i-1].capitalCta
+ clientes[nroOrden].cuotas[i-1].interesCta;
        clientes[nroOrden].cuotas[i].interesCta = clientes[nroOrden].cuotas[i].capitalCta *
(clientes[nroOrden].interes/100);

    }
}

void CuotasInteresDirecto( int nroOrden){

    int i;

    struct Fecha fechaAux = clientes[nroOrden].fechaCompra;

    float capitalCta, interesCta;

    capitalCta = clientes[nroOrden].capital / clientes[nroOrden].cantCuotas;
    interesCta = clientes[nroOrden].capital * (clientes[nroOrden].interes/100);
```

```
    for(i = 0; i < clientes[nroOrden].cantCuotas; i++){
        fechaAux.mes++;
        if (fechaAux.mes==13) {
            fechaAux.mes=1;
            fechaAux.anio++;
        }

        clientes[nroOrden].cuotas[i].fechaCta = fechaAux;
        clientes[nroOrden].cuotas[i].capitalCta = capitalCta;
        clientes[nroOrden].cuotas[i].interesCta = interesCta;
    }
}
```

<u>Problema, orden de los clientes</u>

Ahora, bien, ¿qué ocurre si les pido que me muestren a los clientes del ejercicio anterior ordenados por el total acumulado en cuotas, o que me los muestren en orden alfabético?

Para resolver este problema dimos volvemos al tema ***"Ordenamiento".*** Al final de la guía anterior habíamos dicho a modo de ejemplo, que existían funciones muy útiles que se las solían usar muy a menudo por lo que vimos que la creación de funciones y procedimientos eran esenciales para reutilizar código ya existente.

Esta semana lo que trabajaremos son métodos, o algoritmos para ordenar, inicialmente vectores numérico, pero luego ordenar estructuras.

El primer ordenamiento que veremos es el ordenamiento del tipo **Burbuja**, o ordenamiento por **burbujeo**. La idea básica del ordenamiento de la burbuja es recorrer el conjunto de elementos en forma secuencial varias veces. Cada paso compara un elemento del conjunto con su sucesor (x[i] con x[i+i]), e intercambia los dos elementos si no están en el orden adecuado.

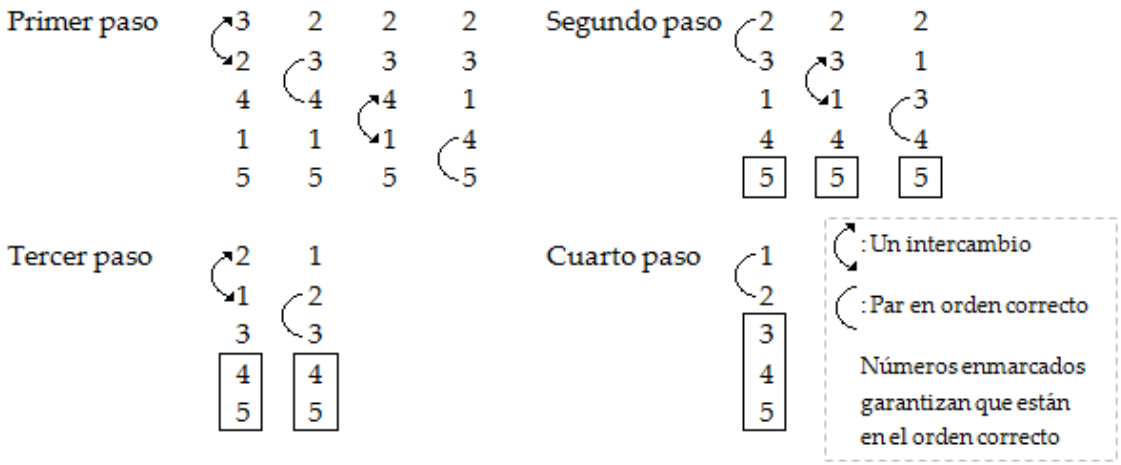
El algoritmo utiliza una bandera que cambia cuando se realiza algún intercambio de valores, y permanece intacta cuando no se intercambia ningún valor, pudiendo así detener el ciclo y terminar el proceso de ordenamiento cuando no se realicen intercambios, lo que indica que este ya está ordenado.

Este algoritmo es de **fácil comprensión y programación**, pero **es poco eficiente** puesto que existen n-1 pasos y n-i comprobaciones en cada paso, aunque es mejor que el algoritmo de ordenamiento por intercambio.

En el peor de los casos cuando los elementos están en el orden inverso, el número máximo de recorridos es n-1 y el número de intercambios o comparaciones está dado por $(n-1) * (n-1) = n^2 - 2n + 1$.

En el mejor de los casos cuando los elementos están en su orden, el número de recorridos es mínimo 1 y el ciclo de comparaciones es n-1.

A modo de ejemplo con un simple vector los pasos serian los siguientes:



Ejercicio 2: Método Burbuja – Crear un vector de 20 componentes enteras, crearlo de forma automática, con números aleatorios o simplemente dados por el índice del for. Una vez cargado el vector ordenarlo y mostrarlo por medio del ordenamiento de burbuja.

/////// ORDENAR VECTOR DE NUMEROS

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

#define TAMANIO 20 //Es una constante

int a[TAMANIO];

int main() {

   int i;
```

```
    /// LLeno el vector de forma inversa y lo imprimo
    printf("\nVector original\n");
    for(i=TAMANIO; i>0;i--){
        a[TAMANIO-i] = i;
        printf(" %d ", a[TAMANIO-i]);
    }


    /// Ordeno el vector usando BURBUJA con FOR
    int x,y,tmp;
    int n = TAMANIO;

    printf("\nVector ordenado usando FOR\n");
    for(x = 1; x < n; x++) {
        for(y = 0; y < n - x; y++) {
            if(a[y] > a[y + 1]) {
                tmp = a[y];
                a[y] = a[y + 1];
                a[y + 1] = tmp;
            }
        }
    }

    /// Imprimo el vector ordenado
    for(i=0; i<TAMANIO;i++){
        printf(" %d ", a[i]);
    }

    /// LLeno el vector de forma inversa para tener de nuevo el valor original y lo imprimo
    printf("\nVector original\n");
    for(i=TAMANIO; i>0;i--){
        a[TAMANIO-i] = i;
        printf(" %d ", a[TAMANIO-i]);
```

```
    }

    /// Ordeno el vector usando BURBUJA con WHILE
    printf("\nVector ordenado usando WHILE\n");
    char senial = ' ';
    while(senial !='n'){
        senial = 'n';
        for(x=0;x<TAMANIO-1;x++){

            if(a[x] > a[x + 1]) {
                tmp = a[x+1];
                a[x+1] = a[x];
                a[x] = tmp;
                senial = 's';
            }
        }
    }



    /// Imprimo el vector ordenado
    for(i=0; i<TAMANIO;i++){
        printf(" %d ", a[i]);
    }
    return 0;
}
```

Ejercicio 3 – Método Burbuja: Ordenar una estructura de Clientes, primero por DNI y luego por orden alfabético.

Cliente debe ser una estructura, ustedes pueden elegir la complejidad de dicha estructura, primordialmente ordenenlos por DNI (se lo dejamos a ustedes puesto que es

un ordenamiento numérico como ya hemos realizado). En este documento me enfocare en mostrarles como es el ordenamiento por orden alfabético.

Decir si un numero esta antes o después de otro, es muy simple, con las relaciones < o >, pero ¿cómo hacemos para saber si una palabra esta antes de la otra? , para esto nos apoyaremos una vez más en la librería string.h. esta librería nos dará la función strcmp. Esta función compara dos cadenas de caracteres, donde compara posición a posición cada palabra, y devuelve un valor positivo o negativo, o cero (si las palabras son iguales), positivo si la primera palabra esta después de la segunda en orden alfabético, y negativo en el caso inverso. Entonces, por ejemplo:

char a[10] = “abc”;

char b[10] = “abc”;

char c[10] = “defgabc”;

CERO es el retorno de strcmp(a,b);

MENOS UNO es el retorno de strcmp(b,c);

UNO es el retorno de strcmp(c,b);

Solución propuesta, usando strcmp:

/////// ORDENAR ESTRUCTURAS

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

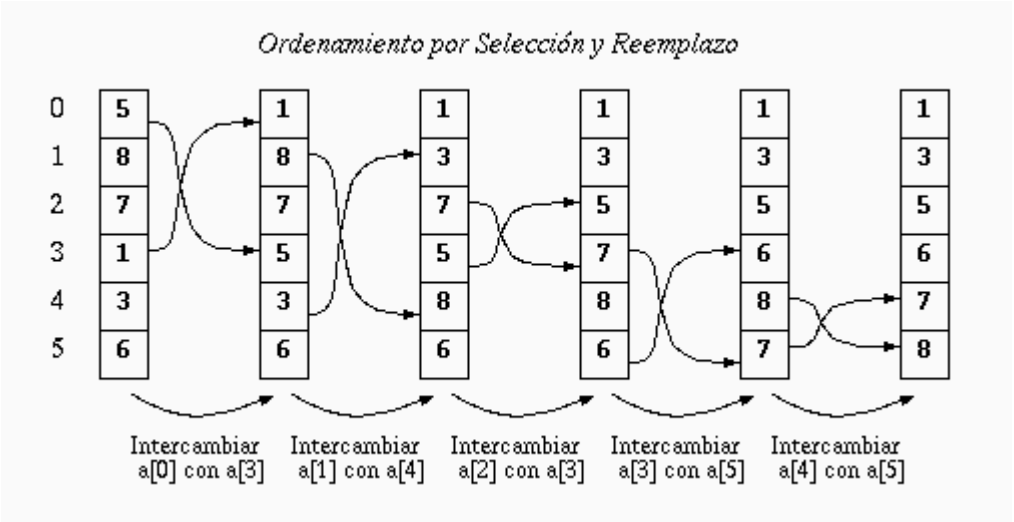
```
#include <stdlib.h>

#define TAMANIO 5 //Es una constante

int a[TAMANIO];
struct clientes{
    char nombre[10];
    char direccion[10];
    int dni;
    struct fecha{
        int dia;
        int mes;
        int anio;
    }fec;
}cli[TAMANIO],tmp;

int main() {

    int i;

    /// Pido los datos de los clientes
    for(i=0;i<TAMANIO; i++){
        printf("Nombre: ");
        scanf("%s", cli[i].nombre);
        printf("Direccion: ");
        scanf("%s", cli[i].direccion);
        printf("DNI: ");
        scanf("%d",&cli[i].dni);
        printf("Dia: ");
        scanf("%d",&cli[i].fec.dia);
        printf("Mes: ");
        scanf("%d",&cli[i].fec.mes);
        printf("Anio: ");
        scanf("%d",&cli[i].fec.anio);
```

```
        system("cls");
    }

     printf("\nVector Original\n");
    /// Imprimo el vector Original
    for(i=0; i<TAMANIO;i++){
        printf("\nNombre: %s", cli[i].nombre);
        printf("\nDireccion: %s", cli[i].direccion);
        printf("\nDNI: %d",cli[i].dni);
        printf("\nDia: %d",cli[i].fec.dia);
        printf("\nMes: %d",cli[i].fec.mes);
        printf("\nAnio: %d",cli[i].fec.anio);
        printf("\n ");
    }


    /// Ordeno el vector usando BURBUJA con WHILE
    printf("\nVector ordenado usando WHILE\n");
    char senial = ' ';
    while(senial !='n'){
        senial = 'n';
        for(i=0;i<TAMANIO-1;i++){

            if(strcmp(cli[i].nombre,cli[i + 1].nombre)>0) {
                tmp = cli[i+1];
                cli[i+1] = cli[i];
                cli[i] = tmp;
                senial = 's';
            }
        }
    }


    /// Imprimo el vector Original
```

```
    for(i=0; i<TAMANIO;i++){
        printf("\nNombre: %s", cli[i].nombre);
        printf("\nDireccion: %s", cli[i].direccion);
        printf("\nDNI: %d",cli[i].dni);
        printf("\nDia: %d",cli[i].fec.dia);
        printf("\nMes: %d",cli[i].fec.mes);
        printf("\nAnio: %d",cli[i].fec.anio);
        printf("\n ");
    }

    return 0;
}
```

Luego vimos otro método de ordenamiento, ***ordenamiento por selección***, probablemente éste es el algoritmo más intuitivo y el más usado para ordenar arreglos pequeños.

El algoritmo de selección y reemplazo ordena los elementos de un arreglo nativo sin utilizar un segundo arreglo. El algoritmo se basa en la siguiente idea. En el arreglo ordenado, el mínimo debe quedar en a[0], de modo que se ubica su posición real en el arreglo. Por ejemplo, se determina que está en a[3] como lo indica el primer arreglo de izquierda a derecha en la siguiente figura:




```
void seleccion(int[ ] a, int n) {
  for (int i= 0; i<n; i++) {
    // Buscamos la posicion del minimo en a[i], a[i+1], ..., a[n-1]
    int k= i;
    for (int j= i+1; j<n; j++) {
      if (a[j]<a[k])
        k= j;
    }
    // intercambiamos a[i] con a[k]
    int aux= a[i];
    a[i]= a[k];
    a[k]= aux;
  }
}
```

Ejercicio 4: Se propone adaptar el ejercicio de cliente para ordenarlo por dni y apellido, utilizando el ordenamiento por selección.

Ejercicio 5: Realizar la prueba de escritorio de este algoritmo para un vector de 5 elementos. Luego compararlo con la prueba de escritorio del método burbuja.

Enum

También vimos un tema quizás no tan importante como los ordenamientos, pero que le será muy útil a la hora de trabajar con opciones, vimos los ENUM, enumeraciones que nos sirven para ir creando datos a medida del usuario, donde transformamos determinadas opciones en números, que podemos manejar como enteros.

Veamos un breve ejemplo para ver cómo funcionan:

```
#include <stdio.h>

enum semana { lunes, martes,miercoles, jueves, viernes, sabado, domingo};
enum mes {enero=-33, febrero, marzo, abril, mayo=20, junio, julio, agosto, septiembre, octubre, noviembre, diciembre};

int main()
{

    enum semana hoy;
    enum mes mesDeHoy;

    hoy = domingo;
    mesDeHoy = julio;
```

```
    printf("Hoy es el dia de la semana numero:  %d \n",hoy);
    printf("Hoy estamos en el mes numero:  %d \n",mesDeHoy);

    if (mesDeHoy == 22){
        printf("Correcto");}

        if (mesDeHoy == julio){
        printf("Correcto");}

           printf("%d ", numero?5 5:numero);

    return 0;
}
```

Ejercicio 5: Agregue algún ENUM a la estructura que dejo a continuación, una vez agregado el enum, generar las funciones cargarFactura, cargarDetalle, mostrarFacturas.

```
struct Factura {

int nroFactura;
```

```
char tipo;
char cabecera[20];
char fecha[8];
float total;

struct Detalle{
int nroDetalle;
char detalle[20];
float precioUnitario;
int cantidad;
}detalles[10];
}facturas[15];
```

Búsqueda Secuencial, Búsqueda Binaria y recursividad

Búsqueda* secuencia*: Si les diéramos un vector numérico de, supongamos, de 100 componentes, es decir int vector[100], y les pedimos que verifiquen si el numero 20193 está en el vector, ¿cómo harían?, sin saber que es una búsqueda secuencial uno lo que haría por intuición seria recorrer todo el vector desde i = 0 hasta i<100 e ir verificando componente a componente si esta el número 20193, esto es una **Búsqueda secuencial**. Es decir es lo que ya veníamos haciendo hace varias semanas. Pero ¿qué ocurre si el número 20193 es el último del vector? Esto podría demorar mucho y se harían demasiadas comparaciones. Para esto aparece la **Búsqueda Binaria.**

Búsqueda binaria: El ejemplo clásico para explicar el funcionamiento de este algoritmo es la búsqueda en una guía telefónica o un diccionario en dónde se aprovecha el orden alfabético.

Si quisiéramos buscar una palabra que empiece con la letra M lo que haríamos normalmente es abrir por la mitad nuestro diccionario e ir avanzando o retrocediendo páginas hasta llegar a la M.

Esto reduce sucesivamente la operación de búsqueda eliminando la mitad de la lista restante. Sin embargo para poder usar este algoritmo necesitamos como precondición que la estructura se encuentre ordenada y conocer la cantidad de elementos totales sobre los cuales vamos a buscar.

Este algoritmo utiliza una técnica de resolución de problemas que se conoce como "divide y vencerás". Esta estrategia parte de la idea de dividir al problema en problemas de menor tamaño para ir resolviéndolos individualmente y así alcanzar la solución global del problema principal.

El funcionamiento del algoritmo sería el siguiente:

- Comparar el elemento buscado con el que ocupa la posición central.
- Si es igual parar la búsqueda
- Si es menor o mayor repetir el proceso tomando la primera o segunda mitad de los elementos según corresponda en cada caso.

El proceso termina cuando nos quedemos sin elementos indicando que el dato buscado no se encontraba.

Por ejemplo, se tiene una estructura con los siguientes elementos:

12 14 15 18 19 21 24 27 33 37 41

Y el dato que queremos buscar es el 27.

Nuestro elemento central en una primera instancia para este caso es el número 21

12 14 15 18 19 **21** 24 27 33 37 41

Como 27 es mayor que 21 se desprecia la primera parte y se revisa la segunda:

24 27 33 37 41

Para este nuevo grupo de elementos volvemos a tomar el elemento central que sería el 33:

24 27 **33** 37 41

Como 27 es menor que 33 se desprecia la segunda parte de los elementos:

24 27

En este caso como no tenemos término central tomamos el término seguido inmediatamente a nuestro central anterior, que en este caso es 27, el dato que buscábamos.

24 **27**

Un ejemplo de una función que realiza este procedimiento podría ser:

```
int busquedaBinaria ( int vector[], int tam, int dato)
{
   int sup    = 0;
   int inf    = 0;
   int centro  = 0;

   sup = tam - 1;

   while(inf <= sup)
   {
      centro = (sup + inf)/2;

      if(vector[centro]==dato)
      {
         return centro;
      }
      else
```

```
        {
            if(dato>vector[centro])

            {
                inf = centro + 1;
            }
            else
            {
                sup = centro - 1;
            }
        }
    }

    return -1;

}
```

Ejercicio 1: Dado el vector: (5,4,3,5,4,2,-3,-9,14,56,82,12,-36,6,5). Realizar la búsqueda secuencial de un valor ingresado por teclado, en caso de encontrarlo mostrar por pantalla la posición, en caso contrario mostrar un cartel de NO SE ENCONTRO. Luego buscar este mismo valor pero con búsqueda binaria. ¿Es posible esto?, realizarlo en caso de ser posible.

Ejercicio 2: Realizar una estructura para guardar productos con sus precios, con el código del producto, buscar dentro del vector de estructuras por medio de búsqueda binaria.

Recursividad

Principalmente es llamar a una función dentro de ella misma. Hace unas semanas atrás en algún momento lo hemos realizado sin saberlo. Por ejemplo cuando llamamos al main() dentro del mismo main(). Resolver un problema mediante recursión significa que la solución depende de las soluciones de pequeñas instancias del mismo problema.

La mayoría de los lenguajes de programación dan soporte a la recursión permitiendo a una función llamarse a sí misma desde el texto del programa.

Si una función se llama a sí misma, puede ser que esto no termine jamás, por esto lo más importante en la recursividad es definir la condición de corte del llamado.

Ejercicio 3: Un ejemplo clásico de recursividad es calcular la factorial. Factorial es multiplicar a un número por todos los anteriores, hasta el 1. Por ejemplo, la factorial de 6, es 6.5.4.3.2.1.

```
#include<stdio.h>


long factorial(int);


int main()
{
  int n;
  long f;
```

```
  printf("de que numero quiere el factorial\n");
  scanf("%d", &n);


  if (n < 0)
    printf("Ignoramos los numeros negativos para el factorial\n");
  else
  {
    f = factorial(n);
    printf("%d! = %ld\n", n, f);
  }

  return 0;
}

long factorial(int n)
{
  if (n == 0)
    return 1;
  else
    return(n * factorial(n-1));
```

}

Ejercicio 4: Realizar una función recursiva que cambie un número en base 10 a cualquier otra base ingresada por teclado.

Ejercicio integrador: A modo de cierre de la primera parte de la materia se propuso en la última clase el siguiente ejercicio. Se pretende regular un listado de alumnos, donde cada alumno tiene nombre, apellido, dni, numero de alumno y fecha de nacimiento. Se pide:

a- Cargar los alumnos y mostrarlos
b- Ordenar los alumnos por DNI y mostrarlos
c- Ordenarlos por apellido y mostrarlos
d- Buscar alumno por DNI con búsqueda binaria.
e- Mostrar los alumnos con su DNI en número binario, para esto crear una función recursiva que transforme el DNI en binario.

A modo de ayuda y guía les adjunto el esqueleto.

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


///DEFINICION DE LA ESTRUCTURA ALUMNOS

typedef struct

{
```

```
    int numAlumno;
    char nombre[20];
    char apellido[20];
    int dni;
    char nacimiento[10];
}alumnos;

///DEFINICION DE FUNCIONES Y PROCEDIMIENTOS

void CargarDatos(); ///CARGA LOS DATOS DE TODOS LOS ALUMNOS
void Mostrar(); ///MUESTRA LOS DATOS DE TODOS LOS ALUMNOS
void OrdenarPorDni(); ///ORDENA POR DNI A LOS ALUMNOS
void OrdenarPorApellido(); ///ORDENA POR APELLIDO A LOS ALUMNOS
int BuscarDni(int tam, int dato); ///BUSCA UN DNI Y MANDO COMO ARGUMENTO EL TAMAÑO DEL VECTOR ALUMNO Y EL DNI QUE QUIERO BUSCAR
void PasarABinario(int dni); ///PASO A BINARIO EL DNI QUE ENCONTRE

///DEFINICION DE VARIABLES GLOBALES

alumnos alumno[5]; ///VECTOR QUE GUARDA TODOS LOS ALUMNOS
alumnos valor; ///AUXILIAR PARA HACER UNA COPIA DE LA ESTRUCTURA EN LOS METODOS DE ORDENAMIENTO
int x = 0; ///AUXILIAR PARA GUARDAR EL DNI
```

```
//-----------------------------------------------------------------------------------------

///EJECUCION DE LA FUNCION MAIN

int main()

{

   int opcion = 0;

   while(opcion >= 0 && opcion < 7) ///MIENTRAS LA OPCION SEA ALGUNA DE LAS MOSTRADAS EL MENU SE EJECUTA

   {

      printf("\nMENU:\n\n");

      printf("Opciones:\n");

      printf("1 - Cargar datos de los alumnos\n");

      printf("2 - Mostrar\n");

      printf("3 - Ordenar por apellido y mostrar a los alumnos\n");

      printf("4 - Ordenar por DNI y mostrar a los alumnos\n");

      printf("5 - Ingresar un DNI y mostrar al alumno\n");

      printf("6 - Mostrar en binario el DNI encontrado\n");

      printf("7 - Salir del programa\n");

      scanf("%d", &opcion); ///GUARDA LA OPCION DESEADA

      fflush(stdin);
```

```
    return 0;
}
//-------------------------------------------------------------------------------------
```