	UNIVERSIDAD DON BOSCO **FACULTAD DE INGENIERIA** **ESCUELA DE COMPUTACION**
CICLO II	GUIA DE LABORATORIO #11 **Nombre de la Practica:** **Datos compuestos (struct)** **Lugar de Ejecución:** **Centro de Computo** **Tiempo Estimado:** **2 horas y 30 minutos** **MATERIA:** **Programación de Algoritmos**

I. OBJETIVOS

Que el alumno sea capaz de:

- Manejar datos de diferente tipo y valores, pero relacionados entre sí por medio de una estructura (struct).
- Crear arreglos de tipos de datos struct, para procesar grandes cantidades de información.

II. INTRODUCCION TEORICA

Tipos de datos definidos por el programador: struct

En la resolución de problemas por computadora muy a menudo surge la necesidad de agrupar datos de diferente tipo o manejar datos que serían muy difíciles de describir en los tipos de datos primitivos de los lenguajes de programación. En esta situación es preciso aprovechar una de las grandes posibilidades brindadas por C++: La posibilidad del programador de definir nuevos tipos de datos.

La capacidad de crear estos nuevos tipos de datos es una característica importante de C++ y libera al programador de restringirse a usar únicamente los tipos de datos ofrecidos por el lenguaje. Una *estructura* contiene múltiples variables, que pueden ser de diferentes tipos de datos. La estructura es importante para la creación de programas potentes, tales como bases de datos u otras aplicaciones que requieran grandes cantidades de datos.

> Una **estructura** es una colección de una o más variables simples denominadas **campos**, cada una de las cuales puede ser de un tipo de datos diferente. Son también conocidas como **Registros.**

Una estructura puede contener cualquier número de campos, cada uno de los cuales tiene un nombre único.

Declaración de una estructura

Una estructura es un tipo de dato definido por el programador, que se debe declarar antes que se pueda utilizar.

La sintaxis de la declaración es la siguiente:

```
struct nombre_estructura{
    tipo_miembro miembro1;
    tipo_miembro miembro2;
    ...
    tipo_miembro miembron;
};
```

Así, por ejemplo, considérese un programa que gestione libros y procese los siguientes datos: título del libro, nombre del autor, editorial y año de publicación. Una estructura *libro* podría ser:

```
struct libro
{
   char titulo[60];
   char autor[30];
   char editorial[30];
   int anio_publicacion;
};
```

Funciones en el interior de un struct

C++, permite incluir funciones en el interior de un struct. Estas funciones permiten manipular los datos incluidos en la estructura, y su uso está muy relacionado con la programación orientada a objetos.

Aunque esta característica se usa casi exclusivamente con las clases, como se verá más adelante en el curso, también puede usarse en las estructuras. De hecho, en C++, las diferencias entre estructuras y clases son muy tenues.

Dos funciones muy particulares son las de inicialización, o **constructor**, y el **destructor**.

El **constructor** es una función sin tipo de retorno y con el mismo nombre que la estructura. Permite inicializar los valores de las variables (campos) de la estructura.

El **destructor** tiene la misma forma, salvo que el nombre va precedido el símbolo ~ .

Definición de variables de estructuras

A una estructura se accede utilizando una variable o variables que se deben definir después de la declaración de la estructura.

En C++ existen dos conceptos similares a considerar: *declaración y definición*. La diferencia técnica es la siguiente:

Una declaración especifica simplemente el nombre y el formato de la estructura de datos, pero no reserva almacenamiento en memoria. Por consiguiente, cada definición de variable para una estructura dada crea un área de memoria en donde los datos se almacenan de acuerdo al formato de la estructura declarada.

Las variables de estructura se pueden definir de dos formas:

1- Listándolas inmediatamente después de la llave de cierre de la declaración de la estructura.
2- Listando el nombre de la estructura seguida por las variables correspondientes en cualquier lugar del programa antes de utilizarla.

A continuación se muestra la definición de variables de la estructura *libro* mediante las dos formas descritas en el párrafo anterior:

```
//    1- Listando las variables de estructura inmediatamente después
//       de la llave de cierre de la declaración de la estructura
struct libro
{
    char titulo[60];
    char autor[30];
    char editorial[30];
    int anio_publicion;

}libro1,libro2,libro3;
```

```
//2-      Listando el nombre de la estructura seguida por las variables
//        correspondientes en cualquier lugar del programa antes de utilizarla.
    libro libro1, libro2, libro3;
```

Acceso a estructuras

Para acceder a una estructura (es decir, almacenar información en la estructura o recuperar información de la misma) se debe acceder a cada miembro de la estructura utilizando el operador punto (.).

Considérese como ejemplo la siguiente asignación al miembro *anio_publicacion* de la variable de estructura *libro1*

```
libro1.anio_publicacion=1562;
```

El operador punto proporciona el camino directo al miembro correspondiente.

Estructuras anidadas

Una estructura puede contener otras estructuras llamadas *estructuras anidadas.* Las estructuras anidadas ahorran tiempo en la escritura de programas que utilizan estructuras similares. Se deben definir los miembros comunes solo una vez en su propia estructura y a continuación utilizar esa estructura como un miembro de otra estructura.

Considérese las siguientes dos estructuras:

```
struct empleado
{
    char nombre_emp [30]; //Nombre del empleado
    char direccion [25];  //Direccion del empleado
    char departamento [20];
    char municipio [20];
    double salario; //Salario mensual del empleado
};

struct clientes
{
    char nombre_cliente [30]; //Nombre del cliente
    char direccion [25]; // Direccion del cliente
    char departamento [20];
    char municipio [20];
    double saldo; //saldo en la compañia
};
```

Nótese que estas estructuras contienen algunos datos diferentes, pero también tienen algunos datos idénticos. Así, se podría disponer de una estructura *info_dir* que tuviera los miembros comunes.

```
struct info_dir
{
     char direccion [25];
    char departamento [20];
    char municipio [20];
};
```

Esta estructura se puede utilizar como un miembro de las otras estructuras, es decir, anidarse.

```
struct empleado {
    char nombre_emp [30]; //Nombre del empleado
```

```
     struct info_dir direccion_empleado; //Direccion del empleado
    double salario; //Salario mensual del empleado
};

struct clientes
{
    char nombre_cliente [30]; //Nombre del cliente
    struct info_dir direccion_empleado; //Direccion del cliente
    double saldo; //saldo en la compañia
};
```

Arreglos de estructuras

Se puede crear un arreglo de estructuras tal como se crean arreglos de cualquier tipo de datos primitivo. Los arreglos de estructuras son idóneos para almacenar un archivo completo de empleados, un archivo de inventario, o cualquier otro conjunto de datos que se adapte a un formato de estructura.

Considérese el siguiente ejemplo en donde se crea un arreglo de 10 elementos de tipo estructura. Nótese que se hace la asignación de valores para el primer elemento del arreglo.

```
    libro listado_libros[10];
    strcpy(listado_libros[0].titulo,"Don Quijote de la Mancha");
    strcpy(listado_libros[0].autor,"Miguel de Cervantes");
    strcpy(listado_libros[0].editorial,"Clasicos Roxsil");
    listado_libros[0].anio_publicion=1562;
```

Utilización de estructuras como parámetros.

C++ permite pasar estructuras ya sea por valor o por referencia. Si la estructura es grande, el tiempo necesario para hacer una copia de la estructura puede ser prohibitivo. En tales casos, se debe considerar el método de paso por referencia.

Manejo avanzado de Cadenas de Caracteres en C++

Operando cadenas de caracteres (Librería string.h)

C++ consta de una librería exclusivamente para funciones de manejo de cadenas de caracteres, llamada **string.h**. Las funciones más comunes son descritas a continuación:

- `gets(nombreArreglo)`

Lee una cadena del teclado hasta que usuario presione tecla intro. Esto significa que permite capturar secuencias de caracteres que tengan incluidos espacios en blanco

- `int strlen(const char *s)`

Calcula la longitud de la cadena de caracteres.

- `char *strcpy(const char *dest, const char *orig)`

Copia la cadena de caracteres apuntada por orig (incluyendo el carácter terminador '\0') al vector apuntado por dest. Las cadenas no deben solaparse, y la de destino, debe ser suficientemente grande como para alojar la copia.

- `int strcmp(const char *s1, const char *s2)`

Compara las dos cadenas de caracteres s1 y s2. Devuelve un entero menor, igual o mayor que cero si se encuentra que s1 es, respectivamente, menor que, igual a, o mayor que s2.

- `char *strncat(char *s1, const char *s2, size_t n)`

Agrega n caracteres de s2 a s1.

- `int strncmp(const char *s1, char *s2, size_t n)`

Compara los primeros n caracteres de dos cadenas.

- `char *strncpy(const char *s1, const char *s2, size_t n)`

Copia los primeros n caracteres de s2 a s1.

III. MATERIALES Y EQUIPO

Para la realización de la guía de práctica se requerirá lo siguiente:

No.	Requerimiento	Cantidad
1	Compilador de C++	1
2	Memoria USB	1

IV. PROCEDIMIENTO

1. Cree una carpeta llamada **PALguia11_CARNET** en su PC para que guarde ahí a sus archivos .cpp del procedimiento de la guía y también de los ejercicios solicitados en el análisis de resultados.

Recuerde reemplazar CARNET por su respectivo número de carnet.

2. Ahora desarrolle cada uno de los programas en C++, teniendo cuidado de aplicar las reglas siguientes en cada código fuente:

- Cuide el uso de mayúsculas y minúsculas al redactar palabras reservadas y los nombres de variables y/o funciones, **ya que lenguaje C++ es sensible a mayúsculas y minúsculas**.
- Realice la compilación respectiva, con el fin de generar el archivo .exe de aplicación.
- Desarrolle varias pruebas de ejecución, ingresando diversos datos en cada prueba, para construir una idea general del funcionamiento del programa!!
- Cuando encuentre una nota de **IMPORTANTE**, al final de un código, realice sin falta las pruebas solicitadas ahí, para deducir las respuestas apropiadas!!

1. Código cpp del Programa: PAL_ejemplo1

Un alumno de Expresión Oral y Escrita necesita calcular su nota de ciclo a partir de la nota de sus tres periodos parciales. La nota del periodo 1 vale el 30% y las restantes un 35% cada una.

Escriba un programa en C++ que ayude a este alumno a calcular su nota de ciclo y a saber si aprobó o no.

```
#include<iostream>
#include<stdlib.h>
#include<stdio.h>
#include<iomanip>

using namespace std;


struct alumno
{
   //campos
   char nombre[20];
   char apellido[20];
   char carnet[9];
   float nota_periodo1;
   float nota_periodo2;
```

```
   float nota_periodo3;
   float nota_ciclo;

   //funciones
   void calcular_nota_final(void){
      //determina nota final de la materia cursada
      //operan a los campos de la estructura
      nota_ciclo= (nota_periodo1*0.3)+ (nota_periodo2*0.35)+(nota_periodo3*0.35);

   }//fin funcion nota_final_ciclo
};//fin definicion del struct alumno

main(){
   alumno alumno1; //Creando una variable de tipo estructura alumno

   cout<<"\nIngrese los siguientes datos del estudiante:";
   cout<<"\nNombre   ? "; gets(alumno1.nombre);
   cout<<"\nApellido ? "; gets(alumno1.apellido);
   cout<<"\nCarnet   ? "; gets(alumno1.carnet);
   cout<<"\n\n-> ahora digite a cada una de las 3 notas de periodo:\n";
   cin>>alumno1.nota_periodo1>>alumno1.nota_periodo2>>alumno1.nota_periodo3;

   //llama a la funcion de la struct
   alumno1.calcular_nota_final();

   system("cls");
   cout<<"\t-------------------------------------------------------------"<<endl;
   cout<<"\t-             INFORME DE RESULTADOS                         -"<<endl;
   cout<<"\t-------------------------------------------------------------"<<endl;
   cout<<"\t Nombre del alumno: "<<alumno1.nombre<<" "<<alumno1.apellido<<endl;
   cout<<"\t Carnet: "<<alumno1.carnet<<endl;
   cout<<setprecision(3)<<"\t Nota del ciclo: "<<alumno1.nota_ciclo<<endl;
   if(alumno1.nota_ciclo>=6)
      cout<<"\t Estado: Aprobado"<<endl;
   else
      cout<<"\t Estado: Reprobado"<<endl;
   system("PAUSE");
}//fin funcion principal
```

IMPORTANTE:

En este ejemplo básico se define la estructura "alumno" y luego se declara una variable de tipo estructura llamada "alumno1". Note como se accede a los miembros de la estructura mediante el operador punto.

3. Observe que el ejemplo anterior, cada entrada no está validada. Por ej. se puede ingresar notas arriba de 10 e incluso notas negativas.

Por tanto, se le pide **VALIDAR DICHAS ENTRADAS** antes de continuar con el siguiente ejemplo!!!

2. Código cpp del Programa: PAL_ejemplo2

En el siguiente ejemplo se hace uso de estructuras anidadas. Se desea registrar una estructura *Empleado* que contenga como miembros a los datos de una *persona* que a su vez contenga los datos de la fecha de nacimiento.

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

#include<iomanip>
using namespace std;

struct Fecha
{
   //campos
   int dia;
   int mes;
   int anio;

   //metodos
   Fecha(){ //constructor
      dia=1; //inicializa campos con una fecha predeterminada
      mes=1;
      anio=1950;
   }

   void ingresar(char descrip[]){
      //recibe dia, mes y año en sus campos
      cout<<"\nDigite fecha de "<<descrip<<":";
      cout<<"\n\tdia ? "; cin>>dia;
      cout<<"\n\tmes ? "; cin>>mes;
      cout<<"\n\ta\xa4o ? "; cin>>anio;
   }

   void imprimir(void){
      //muestra fecha en formato dd/mm/aaaa
      cout<<dia<<"/"<<mes<<"/"<<anio<<endl;
   }
};

struct Persona
{
   //campos
   char nombre[20];
   char apellido[20];
   int edad;

   //funciones (Metodos)
   void ingresardatos(){
      cout<<"\n Apellido(s) ??"; gets(apellido);

      cout<<"\n Nombres ?? ";
      fflush(stdin); //limpiando el buffer de teclado
      gets(nombre);
```

```
      cout<<"\n Edad (a\xa4os) ??"; cin>>edad;
   }
};

struct Empleado
{
   //campos de info de un empleado
   Persona persona; //struct anidada
   Fecha nac;//struct anidada

   char cargo[20]; //cargo desempeñado
   float salario; //salario mensual

   //funciones
   void solicitardatos(){
      //Establece los valores de los miembros
      persona.ingresardatos(); //invoca metodo de struct persona
      nac.ingresar("nacimiento"); //invoca metodo de struct Fecha

      cout<<"\n Digite cargo actual: ";
      fflush(stdin); gets(cargo);

      cout<<"\n Salario: $ "; cin>>salario;
   }

   void verInfo(){
      //muestra en pantalla a la informacion de empleado

      cout<<setw(22)<<" Nombre completo: "<<this->persona.nombre
         <<" "<<this->persona.apellido<<endl;
      cout<<setw(22)<<" Edad: "<<this->persona.edad<<" a"<<char(164)<<"os"<<endl;
      cout<<setw(22)<<" Fecha de nacimiento: "; this->nac.imprimir();
      cout<<setw(22)<<" Cargo: "<<this->cargo<<endl;
      cout<<setw(22)<<" Salario: "<<"$ "<<this->salario<<endl;
      cout<<endl;
   }

}; //fin struct Empleado

main(){
   Empleado Miriam; //variable de tipo Empleado

   cout<<"\n\tIngrese los datos personales y laborales del empleado:\n";
   //invoca a metodo de struct Empleado
   Miriam.solicitardatos();

   system("cls");
   cout<<"\t-------------------------------------------------"<<endl;
   cout<<"\t-           DATOS DE EMPLEADO INGRESADO         -"<<endl;
   cout<<"\t-------------------------------------------------"<<endl;

   Miriam.verInfo();

   system("PAUSE");
}
```

3. Compile el programa anterior y haga las respectivas pruebas del mismo.

3. Nombre de los códigos fuentes: PAL_ejemplo3.cpp y PAL_publicaciones.h

La librería "El rincón del saber" requiere de una aplicación informática para le gestión de sus títulos y existencias disponibles.

En este ejemplo se aplicara:

- Creacion de una librería para definir un struct y sus métodos correspondientes.
- Invocacion de la librería anterior
- Arreglos de estructura y su transferencia entre funciones.
- Uso de funciones de comparación de cadenas.

IMPORTANTE:

- Aunque este ejemplo parezca demasiado largo y complicado, constituye una excelente base para comenzar a trabajar formalmente en el proyecto de catedra.
- Realice varias pruebas de cada una de las opciones, y recurra a su instructor en caso de no entender alguna instrucción.

4. Prepare un archivo de código vacio, para digitar ahí el código siguiente:

Archivo: **PAL_publicaciones.h**

```
#include<iostream>
using namespace std;
#include<iomanip>

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

struct libro{
   // Campos
   char titulo[60];//Nombre del libro
   char autor[30]; //Autor del libro
   int anio_publicacion; //Año de publicacion de libro
   float precio;//precio del libro

   // Metodos

};
```

Cuando guarde el archivo de código anterior, asegurese de:

- seleccionar la opción **Tipo:** por ***Header files (*.h)***
- asignar al archivo el nombre **PAL_publicaciones** y
- seleccionar la carpeta de trabajo usada en este procedimiento para guardar este archivo de cabecera (.h).

4. Prepare un nuevo archivo de código fuente y digite ahí al siguiente segmento de código.

Archivo **PAL_ejemplo3.cpp**

```
#include<iostream>
using namespace std;
#include<iomanip>
#include<conio.h>
#include<windows.h>

#include "PAL_publicaciones.h" //agrega a nueva libreria


main(){
   libro Baldor;
   cout << Baldor.titulo;
   getch();
}//fin main
```

Guarde este nuevo archivo con el nombre **PAL_ejemplo3.cpp**

5. Para continuar, compile el archivo cpp (PAL_ejemplo3.cpp). Si este se compila correctamente, vera una cadena de caracteres aleatorias (almacenada en el campo titulo de la variable struct Baldor).

En caso de errores de compilacion, comience su búsqueda y su solucion desde el archivo de librería (**PAL_publicaciones.h**) y luego en el archivo base (**PAL_ejemplo3.cpp**). No continue hasta que la compilación se genere correctamente.

Y recuerde…

Los archivos de librerías (.h) no pueden compilarse directamente, sino solamente al archivo fuente (.cpp) que las utiliza

6. Para continuar, dentro del archivo de librería (**PAL_publicaciones.h**), definirá a cada uno de los métodos de la struct **`libro`**.

Retorne al archivo de librería y ubique el cursor en la línea siguiente a la del comentario *//métodos* de la struct **libro**.

Agregue ahí al siguiente código, que define al método constructor del struct.

```
//metodos
libro(){ //constructor
   //inicializa valor de campos
   titulo[0]='\0';
   strcpy(titulo,"SIN TITULO");
   strcpy(autor, "DESCONOCIDO");
   anio_publicacion=1900;
   precio=0;
}
```

Guarde los cambios de este archivo. Retorne al archivo principal (**PAL_GUIA10_ejemplo3.cpp**), para compilarlo y ejecutarlo nuevamente.

Se mostrara el titulo “SIN TITULO”, comprobando la inicialización de campos hecha al ejecutarse el método constructor.

7. Puede facilitar la redacción de los métodos del struct, dividiendo su redacción en 2 etapas distintas: el prototipo y luego su definición.

8. Retorne al código del archivo de librería, ubique el cursor al final (luego del cierre del struct) y agregue una nueva línea con el comentario: **`//definición de métodos del struct`**

Seleccione las lineas que actualmente definen al método constructor, remuévalas de su posición actual y ubíquelas hasta después del nuevo comentario agregado al final del codigo.

Y en reemplazo del código removido en el interior del struct, escriba ahí a solamente el prototipo del constructor, asi:

```
libro(); //constructor
```

9. Finalmente, utilizara al **operador de ámbito (: :)** para que el compilador "enlace" a la definición del constructor (redactada fuera del struct) con su prototipo (indicada en el interior del struct).

Modifique el encabezado de la función constructor, agregando al inicio el nombre del struct y el operador (::) de esta manera:

Encabezado Original del método constructor	`//Definiciones de metodos de struct libro` `libro(){ //constructor`
Modificarlo, agregando el nombre del struct y el operador de ámbito (: :)	`//Definiciones de metodos de struct libro` `libro::libro(){ //constructor`

9. Guarde los cambios de la librería y compile de nuevo el cod. principal. Debe ejecutarse normalmente.

10. Para continuar, redacte al final del código del archivo de librería a las definiciones de los métodos restantes:

Codigo de métodos restantes del struct libro

```
void libro::ver(void){
   //despliega los valores registrados en los campos
   cout<<endl;
   cout<<setw(10)<<"Titulo: "<<titulo<<endl;
   cout<<setw(10)<<"Autor: "<<autor<<endl;
   cout<<"A\xa4o publicacion: "<<anio_publicacion<<endl;
   cout<<setw(10)<<"Precio: $ "<<precio<<endl;
}

void libro::registrar(void){
   //solicita valores de campos del libro
   system("cls");
   cout<<"\t-------------------------------------------------"<<endl;
      cout<<"\t-          REGISTRANDO NUEVO LIBRO              -"<<endl;
      cout<<"\t-------------------------------------------------"<<endl;
      cout<<"\nTitulo    ? "; fflush(stdin); gets(this->titulo);
      cout<<"\nAutor(es) ? "; fflush(stdin); gets(this->autor);
      cout<<"\nA"<<char(164)<<"o de publicacion ? ";
      cin>>this->anio_publicacion;
      cout<<"\nPrecio del libro ? $ "; cin>>this->precio;
      cout<<"\n LIBRO REGISTRADO EXITOSAMENTE"<<endl;
}//fin metodo registrar


void libro::modificar(void){
   //permite a usuario elegir campo a modificar
   bool salir=false;
   char opc, menu;
```

```
   do{
      system("cls");
   cout<<"----------MODIFICACION DE LIBRO-----------------------"<<endl;
      cout<<"-- Datos del libro elegido:";

   this->ver(); //muestra datos del libro que se modificara

      cout<<"\n * Seleccione letra de la accion que desea ejecutar:"<<endl;
      cout<<"  Modificar ... \n";
      cout<<" (t) ..Titulo    (a) ..Autor(es)    (p) ..A\xa4o publicacion";
   cout<<"    (v) ..Precio\n\n (s o ESC) Finalizar modificacion\n";

   opc=Capturar_MenuModificar();//captura opcion elegida por usuario

   switch(opc){
      case 't': case 'T':
         cout<<"\nIngresa nuevo titulo: ";
         fflush(stdin); gets(this->titulo);
         break;
      case 'a': case 'A':
         cout<<"\nIngresa nuevo autor: ";
         fflush(stdin); gets(this->autor);
         break;
      case 'p': case 'P':
         cout<<"\nDigita a\xa4o de publicacion: ";
         cin>>this->anio_publicacion;
         break;
      case 'v': case 'V':
         cout<<"\nDigita nuevo precio: $";
         cin>>this->precio;
         break;
      default:
          salir=true;
         //finaliza modificacion actual

   }//fin switch opc

   }while(!salir);

}//fin metodo modificar


char libro::Capturar_MenuModificar(){
//genera ciclo infinito hasta que usuario presione opcion menu correcta
   char menu;
   do{
        menu=getch(); //captura tecla presionada
        switch(menu){ //valida opcion de menu
           case 't': case 'T': case 'a': case 'A':
           case 'p': case 'P': case 'v': case 'V':
           case 's': case 'S': case 27:
              return(menu); //finaliza ciclo infinito
        }
   }while(true);
}//fin Capturar_MenuModificar
```

11. Seleccione el encabezado de cada una de las funciones anteriores, para copiarlo como prototipo en el interior del struct libro. Finalmente, en cada prototipo, elimine la combinación de acceso de ámbito (`libro::`)

Por ejemplo, para el encabezado copiado del método Capturar_MenuModificar:

```
char libro::Capturar_MenuModificar(){
```

Su respectivo prototipo dentro del struct libro quedara solamente asi:

```
char Capturar_MenuModificar();
```

12. Guarde los cambios en la librería y compile el código principal. Debera ejecutarse normalmente.

13. Para completar el ejercicio, retorne al código del programa principal, seleccione y elimine a main y su contenido. Luego reemplácelo por el siguiente código (no debe modificar a la inclusión actual de las librerias).

Codigo restante del archivo **PAL_ejemplo3.cpp**

```
//prototipos funciones
void listar_libros(int cantidad, libro libros[]);
void buscar_por_autor(char cadena_busqueda[],int cantidad, libro listado_libros[]);

//definicion de funciones

main(){

   libro listado_libros[50];//definiendo arreglo de tipo "libro"
   int cant=0;//cantidad de libros registrados
   bool salir=false;//controla la salida del menu principal

   char opcion;//opcion seleccionada en el menu principal
   int codigo;//codigo del libro seleccionado


   bool salir_submenu=false;
   char opcion_busqueda;
   char cadena_busqueda[50];//Almacenará la cadena a buscar


   do{
      system("cls");
      cout<<"\t------------------------------------------------------"<<endl;
      cout<<"\t-       LIBRERIA EL RINCON DEL SABER                 -"<<endl;
      cout<<"\t------------------------------------------------------"<<endl;
      cout<<"\t- OPCIONES DISPONIBLES:                               -"<<endl;
      cout<<"\t- 1- Registrar nuevo libro                            -"<<endl;
      cout<<"\t- 2- Modificar libro existente                        -"<<endl;
      cout<<"\t- 3- Listar libros existentes                         -"<<endl;
      cout<<"\t- 4- Buscar libros                                    -"<<endl;
      cout<<"\t- 9- Salir                                           -"<<endl;
      cout<<"\t------------------------------------------------------"<<endl;
      cout<<"\n\tSelecciona una opcion: ";
```

```
      opcion=getche();//Leyendo la opcion seleccionada
      switch(opcion){
         case '1'://Registrar nuevo libro
            system("cls");
            //invoca a motodo registrar () del libro actual
            listado_libros[cant].registrar();

            cant++;//aumentamos la cantidad de libros en 1
            system("PAUSE");
            break;

         case '2'://Modificar libro existente
            system("cls");
            cout<<"Modificando a un libro ...";
      /*llamada a la funcion listar libros, se pasa como parametro la  cantidad
      de libros registrados y el vector de variables de tipo "libro" a listar*/
            listar_libros(cant,listado_libros);

            cout<<"\n\nIngresa el codigo del libro que deseas modificar: ";
            cin>>codigo;
            //Verificamos si el codigo del libro existe o no
            if(codigo>cant){
               cout<<"\nNo existe un libro registrado con este codigo!!!!"<<endl;
               system("PAUSE");
               continue;
            }else
            //llamada a metodo modificar de posicion de vector de libros
               listado_libros[codigo-1].modificar();

            break;
         case '3'://Listar libros existentes
            system("cls");
   /*llamada a la funcion listar libros, se pasa como parametro la  cantidad
   de libros registrados y el vector de variables de tipo "libro" a listar*/
            listar_libros(cant,listado_libros);

            system("PAUSE");

            break;
         case '4': //buscar libros

            salir_submenu=false;
            do{
            system("cls");
      cout<<"\t-------------------------------------------------------"<<endl;
      cout<<"\t-              BUSQUEDA DE LIBROS                     -"<<endl;
      cout<<"\t-------------------------------------------------------"<<endl;
      cout<<"\t- OPCIONES DISPONIBLES:                               -"<<endl;
      cout<<"\t- A- Buscar por autor                                 -"<<endl;
      cout<<"\t- B- Buscar por titulo                                -"<<endl;
      cout<<"\t- S- Regresar al menu principal                       -"<<endl;
      cout<<"\t-------------------------------------------------------"<<endl;
            cout<<"Selecciona una opcion de busqueda: ";
            opcion_busqueda=getche();
            switch(opcion_busqueda){
```

```
               case 'A': case 'a':
                  //busqueda por autor
                  cout<<"\n Ingresa el nombre del autor a buscar: ";
                  fflush(stdin); gets(cadena_busqueda);
                  system("cls");
                  /*llamada a la funcion buscar_por_autor pasando como parametros
       la cadena a buscar, la cantidad de libros registrados
              y el arreglo de variables de tipo libro*/
            buscar_por_autor(cadena_busqueda,cant,listado_libros);
                  cout<<endl;
                  system("PAUSE");

                  break;
               case 'B': case 'b':
                  break;
               case 'S': case 's':
                  salir_submenu=true;
                  break;
            }//fin switch opcion_busqueda
            }while(salir_submenu==false);


            break;
         case '9':
            cout<<"\n\t GRACIAS POR VISITARNOS, HASTA LUEGO"<<endl;
            Sleep(1000);
            salir=true;
      }
       system("cls");
   }
   while(salir==false);

}//fin main


/*Este procedimiento recorre lista de libros ingresados hasta ese momento.
 Se pasan como parametros el arreglo de tipo libro y la cantidad de
 libros que se han ingresado*/
void listar_libros(int cantidad, libro libros[]){
      cout<<"\n-------------------------------------------------------"<<endl;
      cout<<"             LISTADO DE LIBROS DISPONIBLES             -"<<endl;
      cout<<"-------------------------------------------------------"<<endl;

   if(cantidad>0){ //si ya hay libros registrados
      cout.setf(ios::left);
      cout<<"\n  # "<<setw(20)<<"Titulo"<<setw(20)<<"Autor"<<endl;
      for(int i=0;i<cantidad;i++){
         //muestra datos de un libro especifico
         cout<<setw(4)<<i+1<<" "<<setw(20)<<libros[i].titulo<<
         setw(20)<<libros[i].autor<<endl;

      }
      cout.unsetf(ios::left);
      }
   else{ //Si aun no se ha registrado ningun libro ....
```

```
      cout<<"\n Aun no se ha registrado ningun libro"<<endl;
      }

}//fin funcion listar_libros

void buscar_por_autor(char cadena_busqueda[],int cantidad, libro listado_libros[]){
     int resultados=0;//Cantidad de resultados de la busqueda
    cout<<"\t-------------------------------------------------------------"<<endl;
      cout<<"\t-               BUSQUEDA POR AUTOR                         -"<<endl;
      cout<<"\t-------------------------------------------------------------"<<endl;
      cout<<"Autor a buscar: "<<cadena_busqueda<<endl;
    cout<<"\nResultados de la busqueda: ";
     for(int i=0;i<cantidad;i++){
        //compara si dos cadenas son identicas...
        if(strcmp(cadena_busqueda, listado_libros[i].autor)==0){
           //imprime datos de los libros encontrados en la busqueda
           listado_libros[i].ver();
   resultados++;//Se aumentan en 1 la cantidad de resultados obtenidos
        }
     }
     if(resultados==0)//Si no hay ningun resultado entonces...
     cout<<"\n Lo siento, no se encontraron coincidencias"<<endl;
}//fin funcion buscar_por_autor
```

V. DISCUSION DE RESULTADOS

PROBLEMA 1 (40%)

Modifique el cod. Fuente del último ejemplo (PAL_ejemplo3.cpp), para que se permita buscar a los libros por el título.

Y además, al struct `libro`, sobrecargue a su método `registrar( )` con otro método que reciba en parámetros al título, autor y precio del libro, para ser asignados internamente a sus campos correspondientes.

PROBLEMA 2 (60%)

Cree una librería llamada **`agendatelefonica.h`**, la cual defina en su interior a una estructura llamada **`contacto`**, que permita registrar la siguiente información:

Nombres Completo Edad Email

La estructura anterior debe contar con un método diferente para cada una de estas acciones hechas a un contacto:

a) Inicializar los valores de los campos.
b) Solicite el nombre del contacto
c) Reciba en un parámetro a la edad de la persona del contacto. La edad valida de una persona puede estar entre 1 a 100 años, de lo contrario, asigna cero.
d) Recibe al email del contacto, divido en 2 partes: nombre de la cuenta y el nombre del servidor de correo. Por ej.
   Un email a registrar (cristian@udb.edu.sv) debe ser recibido en 2 partes: nombre de cuenta (cristian) y servidor de correo (udb.edu.sv)
e) Mostrar los datos del contacto registrados en ese momento

Finalmente, haga lo necesario para demostrar que el struct anterior y sus métodos funcionan correctamente.

VII. BIBLIOGRAFIA

- Programación en C++: Algoritmos, estructuras de datos y objetos. Joyanes Aguilar, Luis. No. De Clasificación 005.362 J88 2000. Editorial: MCGRAW HILL
- Cómo Programar en C/C++. Deitel, Harvey M... No. De Clasificación 005.362 D325 1995 Editorial: PRENTICE HALL
- Programación y diseño en C++: Introducción a la programación y al diseño orientado a objetos. Cohoon, James P; Davidson, Jack W. No. De Clasificación 005.362 C678 2000. Editorial: MCGRAW HILL.