Capítulo

08

Estructuras

Jorge Hochstetter Diez

Ania Cravero Leal

Departamento de Ingeniería de Sistemas
Facultad de Ingeniería, Ciencias y Administración

"Proyecto financiado por el Fondo de Desarrollo Educativo de la Facultad de Ingeniería, Ciencias y Administración de la Universidad de La Frontera"

Versión

0.9

TEMARIO

Estructuras

El objetivo principal de este capítulo es que comprendas tanto la sintaxis como el uso de estructuras en el desarrollo de programas. Para esto te presentamos los conceptos básicos que definen una estructura, el cómo definir variables tipo estructura y el acceso a los miembros de éstas, para finalmente ver cómo utilizar funciones con estructuras y el uso de estructuras con arreglos y funciones.

8.1 Conceptos Básicos

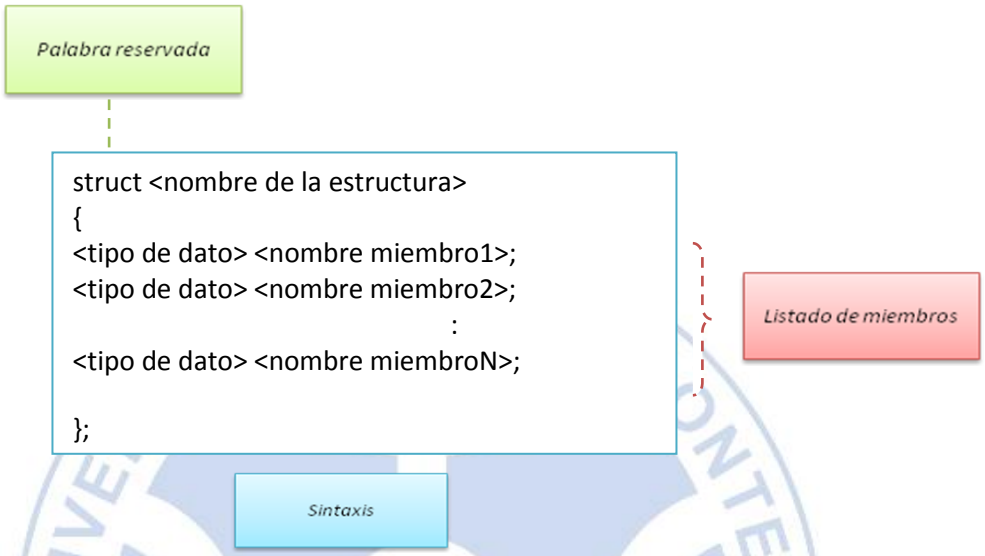
Se define estructura, como una colección de uno o más elementos, cada uno de los cuales puede ser de un tipo de dato diferente a los cuales se le asigna un solo nombre. Las estructuras de datos permiten agrupar varios datos que mantengan algún tipo de relación y permite manipularlos todos juntos con un mismo identificador o por separado.

- Cada elemento de la estructura se denomina miembro.
- Una estructura puede contener un número ilimitado de miembros.
- A las estructuras también se las llama registros.

a. Estructura

Una estructura es un tipo de dato creado por el usuario, por lo que se debe definir antes de utilizarlo. Una vez definido, se pueden crear variables de tipo estructura.

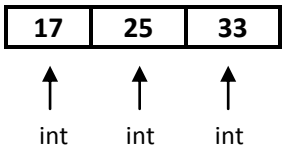
Figura 8.1: Forma general de una Estructura



Ejemplos:

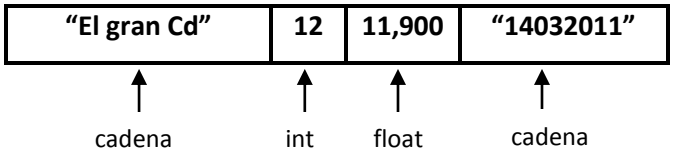
Podemos crear una estructura denominada **Tiempo** que contiene 3 miembros: ***hora, minutos y segundos*****.** A continuación su representación:

```
struct Tiempo
{
        int hora;
        int minuto;
        int segundo;
};
```



Podemos crear una estructura llamada denominada ***Compac Disk*** que contiene 4 miembros: ***título del Cd, número de pistas, precio y fecha de compra***. A continuación su representación:

```
struct Compac_Disk
{
        char titulo[35];
        int num_pistas;
        float precio;
        char fecha_compra[8];
};
```



8.2 Declaración de variables de tipo estructura:

Una vez definido el tipo de dato estructura, se necesita declarar variables de ese tipo.
Existen dos formas diferentes para declarar las variables:

1. En la definición del tipo de datos estructura.

```
struct Compac_Disk
{
        char titulo[35];
        int num_pistas;
        float precio;
        char fecha_compra[8];
} cd1,cd2,cd3;
```

El compilador reserva la memoria necesaria para almacenar las 7 variables.

2. Como el resto de las variables.

```
...
Compac_Disk cd1,cd2,cd3;
...
```

Los miembros de cada variable se almacenan en posiciones consecutivas en memoria.

Inicialización de variables de tipo estructura:
Para ello se especifican los valores de cada uno de los miembros entre llaves y separados por comas.

```
...
struct Compac_Disk
{
        char titulo[35];;
        int num_pistas;
        float precio;
        char fecha_compra[8];
};
...
Compac_Disk cd = { "El gran Cd", 15, 18, "14032011" };
...
```

Definición de la estructura

"El gran Cd"	12	11,900	"14032011"

8.3 Acceso a los miembros de una variable de tipo estructura:

Una vez declarada una variable de tipo estructura, se puede acceder a los miembros de dicha variable. Se puede modificar la información de alguno de los miembros, como recuperar información para imprimirla en pantalla.

Para acceder a los miembros de una estructura se usa el punto u operador miembro **(.)** La sintaxis es: **cd.titulo, cd.num_pistas, cd.precio, cd.fecha_compra**

```
...
struct Compac_Disk
{
        char titulo[35];;
        int num_pistas;
        float precio;
        char fecha_compra[8];
};
...
Compac_Disk cd;
...

cd.titulo = "El gran Cd"
cd.num_pistas= 12
cd.precio= 11,900
cd. fecha_compra= "14032011"
...
```

Definición de la estructura

Inicialización de la variable cd

"El gran Cd"	12	11,900	"14032011"

También se pude acceder a los miembro utilizando el operador puntero **(–>)**, pero por ahora no lo utilizaremos.

Acceso a los miembros de una variable de tipo estructura en C++:

En el siguiente ejemplo se ocupa la función **getline** que con unión al **cin,** permite la lectura correcta de cadenas, sirve para leer una línea completa incluyendo espacios en blanco

En el siguiente ejemplo utilizamos como ejemplo la función getline para almacenar la fecha.

```
…
struct Compac_Disk
{
        char titulo[35];;
        int num_pistas;
        float precio;
        char fecha_compra[8];
};
…
Compac_Disk cd;
…

cout <<"Introduzca título del Compac Disk"<<endl;
cin.getline (cd.titulo, 35);
cout << "Introduzca el número de pistas"<<endl;
cin>> cd.num_pistas;
cout << "Introduzca el precio"<<endl;
cin>> cd.precio;
cout << "Introduzca fecha de compra"<<endl;
cin.getline (cd.fecha_compra, 8);

precio_final = cd.precio * 0,90;
cd.precio = precio_final;
```

Definición de la estructura

Almacenar información en la variable cd mediante el teclado

Recuperar y modificar información de la variable cd

Ejemplo:

1. Leer Tiempo

```
//Lee los valores que representan un instante
        cin >> objetoTiempo.hora;
        cin >> objetoTiempo.minuto;
        cin >> objetoTiempo.segundo;
```

Imprimir Tiempo

```
//Muestra los valores que representan un instante
        cout << setfill('0')<<setw(2)<<objetoTiempo.hora<<':';
        cout << setw(2)<< objetoTiempo.minuto<<':';
        cout << setw(2)<< objetoTiempo.segundo<<endl<<endl;
```

En éste código hemos utilizado el manipulador ***setfill()*** para especificar que se desea rellenar los espacios en blanco, definidos por el manipulador ***setw()***, con el carácter de punto. Se puede

observar que al utilizar el manipulador setfill() su efecto permanece para todos los flujos de E/S hasta que se selecciona otro carácter diferente.

Ejemplo de Inicialización abreviada de Estructuras:
Si pensamos en la siguiente estructura

```
struct estudiante{
        char nombre[35];
        char nMatricula[11];
        char codCarrera[4];
        bool titulado;
        };
```

Podemos iniciar un objeto rápidamente de la forma:

```
        estudiante John;
        John = {"John B. “, "11345678104“, "3005“, true}
```

Ejemplo en C++: A continuación un ejemplo de estructuras en C++:

```
#include <iostream.h>

struct alumno {
      char nombre[40];
      float p1, p2, pp, ep, ef, prom;
} a;

int main()
{
      cout<<"Ingresar nombre: ";
      cin>>a.nombre;
      cout<<"Ingresar nota Prueba 1: ";
      cin>>a.p1;
      cout<<"Ingresar nota Prueba 2: ";
      cin>>a.p2;
      cout<<"Ingresar nota examen parcial: ";
      cin>>a.ep;
      cout<<"Ingresar nota examen final: ";
      cin>>a.ef;
      a.pp=(a.p1+a.p2)/2;
      a.prom=(a.pp+a.ep+a.ef)/3;
      cout<<"El promedio final de "<<a.nombre<<" es "<<a.prom<<endl;
      system("PAUSE");
      return 0;
}
```

Definición de la estructura

Almacenar información en la variable a mediante el teclado

Recuperar y modificar información de la variable a

8.4 Estructuras anidadas

Uso de estructuras anidadas:
Un miembro de una estructura puede ser a su vez otra estructura.

Ejemplo 1: Agregar a la estructura ***estudiante*** el miembro fecha de nacimiento del siguiente tipo

A diferencia de cómo se manejo la fecha en el ejemplo inicial en donde la fecha sólo como ejemplo se ocupo un char de 8, ahora aprovechando la estructura se pude trabajar en forma separada. Para aclarar este cambio si le pregunto ¿cómo sabe cuál es el cd es más antiguo? Seguramente si tiene el campo de la fecha como un char no podrá, en cambio al crear una estructura para fecha si se podrá acceder en éste caso al miembro año de la estructura fecha.

```
struct tipoFecha{
        int dia;
        int mes
        int agnio;
};
```

Ejemplo de Estructuras Anidadas

```
struct estudiante{
        tipoFecha fechaNac;
        char nombre[35];
        char nMatricula[11];
        char codCarrera[4];
        int titulado;
        };
```

```
estudiante John;
John.fechaNac.dia
John.fechaNac.mes
John.fechaNac.agnio
```

} Acceso a los miembros internos del sub-tipo

Ejemplo en C++: Estructuras Anidadas

A continuación veremos un programa escrito en C++ de estructuras anidadas, en donde se definen 2 estructuras en donde el miembro de la estructura estudiante es de tipo fecha (estructura) , se pide al usuario ingresar datos y luego se imprime en pantalla lo ingresado:

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

struct fecha{
    int dia;
    int mes;
    int agno;
};

struct estudiante{
    float nota;
    char nombre[20];
    char matricula[11];
    int carrera;
    fecha fecha_ing;
    fecha fecha_tit;
};

int main()
{
    estudiante estudiante1;

    cout << "Ingrese Nombre Estudiante: " << endl;
    cin >> estudiante1.nombre;
    cout << "Ingrese Nota Estudiante: " << endl;
    cin >> estudiante1.nota;
    cout << "Ingrese matricula Estudiante: " << endl;
    cin >> estudiante1.matricula;
    cout << "Ingrese carrera Estudiante: " << endl;
    cin >> estudiante1.carrera;
    cout << "Ingrese dia de Ingreso Estudiante: " << endl;
    cin >> estudiante1.fecha_ing.dia;
    cout << "Ingrese mes de Ingreso Estudiante: " << endl;
    cin >> estudiante1.fecha_ing.mes;
    cout << "Ingrese año de Ingreso Estudiante: " << endl;
    cin >> estudiante1.fecha_ing.agno;
    cout << "Ingrese dia de Titulación Estudiante: " << endl;
    cin >> estudiante1.fecha_tit.dia;
    cout << "Ingrese mes de Titulación Estudiante: " << endl;
    cin >> estudiante1.fecha_tit.mes;
    cout << "Ingrese año de Titulación Estudiante: " << endl;
    cin >> estudiante1.fecha_tit.agno;
```

Definición de la estructura

Definición de la estructura

Fecha_ing es de tipo fecha

Almacenar información en las variables

```
    cout << endl << endl;
    cout << "Los datos del estudiante son: " << endl;
    cout << "Nombre: " << estudiante1.nombre << endl;
    cout << "Nota: " << estudiante1.nota << endl;
    cout << "Matricula: " << estudiante1.matricula << endl;
    cout << "Carrera: " << estudiante1.carrera << endl;
    cout << "Fecha de Ingreso: " << estudiante1.fecha_ing.dia << "-" <<
    estudiante1.fecha_ing.mes << "-" << estudiante1.fecha_ing.agno << endl;
    cout << "Fecha de Titulación: " << estudiante1.fecha_tit.dia << "-" <<
    estudiante1.fecha_tit.mes << "-" << estudiante1.fecha_tit.agno << endl;
system("pause");
}
```

Imprime en pantalla la información almacenada

8.5 Ejemplos de estructuras y funciones

A continuación veremos un ejemplo en donde se define, declara, inicializa e imprime una estructura, utilizando una función.

Ejemplo 1: Una estructura y una función

```
#include <iostream>

using namespace std;


struct estudiante {
                    char nombre[40];
                    char nMatricula[12];
                    char codCarrera[5];
                    bool titulado;
                  };

void imprimirEstudiante(estudiante);

int main()
{
    estudiante luis={"Jorge Stuard B.",
                     "12345678104",
                     "3002",
                     false
                    };

    imprimirEstudiante(luis);
    system("pause");
    return 0;
}
```

Definición de la estructura

Prototipo de la función

Luis es un objeto o variable de tipo estudiante y se le asignan los valores según los miembros de la estructura

Se pasa el objeto Luis a la función imprimirEstudiante

```
void imprimirEstudiante(estudiante al)
{
     cout<<"Nombre     : "<<al.nombre<<endl;
     cout<<"Matricula  : "<<al.nMatricula<<endl;
     cout<<"Nombre     : "<<al.codCarrera<<endl;
     cout<<"Titulado   : "<<(al.titulado?"Si":"No")<<endl<<endl;
     return;
}
```

Al es una variable de tipo estudiante la cual toma el valor de la variable Luis dentro de la función imprimirEstudiante

Ejemplo 2: Estructuras y funciones, en este ejemplo utilizamos una función (Parámetros por Referencia)

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

struct fecha{
     int dia;
     int mes;
     int agno;
};

struct estudiante{
     float nota;
     char nombre[20];
     char matricula[11];
     int carrera;
     fecha fecha_ing;
     fecha fecha_tit;
};

void Ingresar(estudiante &);
void Mostrar(estudiante);

void main()
{
     estudiante est;

     Ingresar(est);
     Mostrar(est);

}
```

Definición de la estructura

Definición de la estructura

Fecha_ing es de tipo fecha

Fecha_tit es de tipo fecha

Se referencia el objeto a modificar

Prototipo de la función

est es de tipo estructura

Pasamos el objeto est a la función ingresar

Pasamos el objeto est a la función mostrar, con las modificaciones realizadas en la función Ingresar

```
void Ingresar(estudiante &estudiante1)
{
        cout << "Ingrese Nombre Estudiante: " << endl;
        cin >> estudiante1.nombre;
        cout << "Ingrese Nota Estudiante: " << endl;
        cin >> estudiante1.nota;
        cout << "Ingrese matricula Estudiante: " << endl;
        cin >> estudiante1.matricula;
        cout << "Ingrese carrera Estudiante: " << endl;
        cin >> estudiante1.carrera;
        cout << "Ingrese dia de Ingreso Estudiante: " << endl;
        cin >> estudiante1.fecha_ing.dia;
        cout << "Ingrese mes de Ingreso Estudiante: " << endl;
        cin >> estudiante1.fecha_ing.mes;
        cout << "Ingrese agno de Ingreso Estudiante: " << endl;
        cin >> estudiante1.fecha_ing.agno;
        cout << "Ingrese dia de Titulacion Estudiante: " << endl;
        cin >> estudiante1.fecha_tit.dia;
        cout << "Ingrese mes de Titulacion Estudiante: " << endl;
        cin >> estudiante1.fecha_tit.mes;
        cout << "Ingrese agno de Titulacion Estudiante: " << endl;
        cin >> estudiante1.fecha_tit.agno;
        cout << endl << endl;
}
```

La función recibe la referencia a est en estudiante1 para modificar el objeto

Se modifica el objeto estudiante1, referencia de est, y por ende se modifica est

```
void Mostrar(estudiante estudiante1)
{
    cout << endl << endl;
    cout << "Los datos del estudiante son: " << endl;
    cout << "Nombre: " << estudiante1.nombre << endl;
    cout << "Nota: " << estudiante1.nota << endl;
    cout << "Matricula: " << estudiante1.matricula << endl;
    cout << "Carrera: " << estudiante1.carrera << endl;
    cout << "Fecha de Ingreso: " << estudiante1.fecha_ing.dia << "-"
    << estudiante1.fecha_ing.mes << "-" << estudiante1.fecha_ing.agno << endl;
    cout << "Fecha de Titulacion: " << estudiante1.fecha_tit.dia << "-"
     << estudiante1.fecha_tit.mes << "-" << estudiante1.fecha_tit.agno << endl;

}
```

La función recibe el valor de la variable est y lo imprime en pantalla

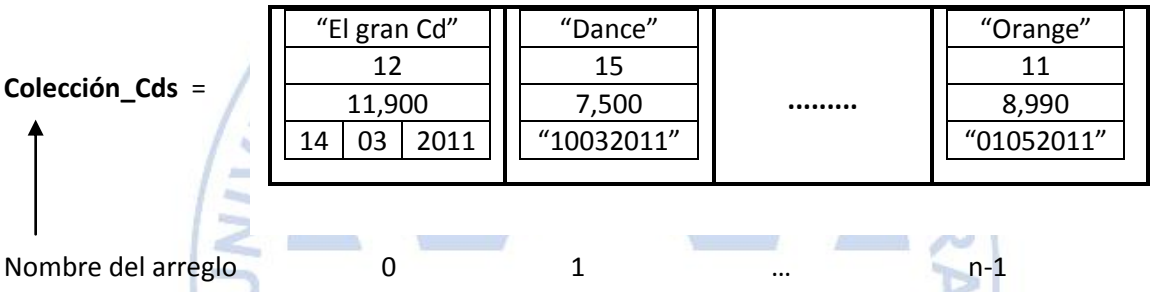
8.6 Arreglos de estructuras:

Como sabemos, un arreglo es una zona de memoria (variable) que puede almacenar un conjunto de N datos del mismo tipo; ejemplificando si se quiere almacenar una colección de los Cds

deberíamos utilizar arreglos, lo mismo si pensamos en registrar datos de varios alumnos y luego imprimir en pantalla sus datos.

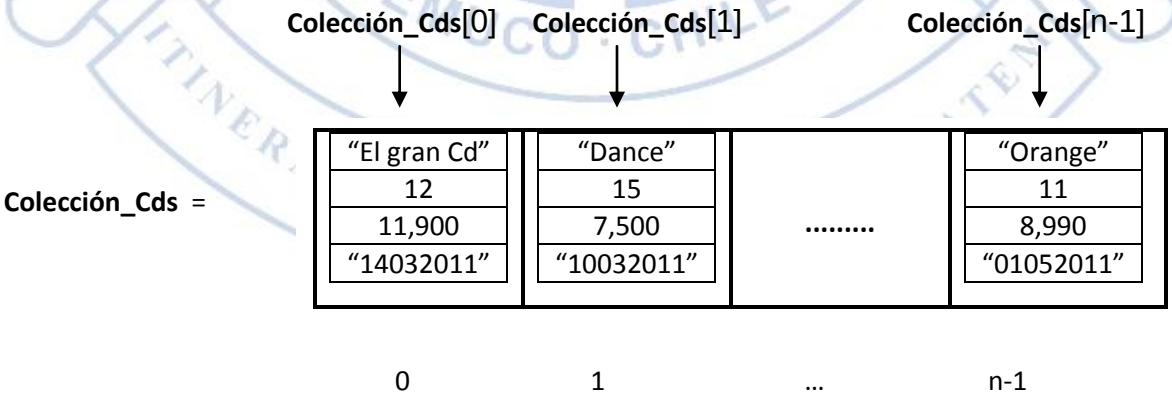
Ejemplo: Si queremos almacenar información de **todos los Compac Disk** que hemos adquirido. Hasta ahora con una variable de tipo Compac Disk que sería la variable **cd**, solo podíamos guardar los datos de un Compac Disk., por lo que ahora necesitaremos un arreglo de Compac Disk como se presenta a continuación:

Nota: A partir de aquí utilizaremos fecha como estructura, ya que sien algún momento necesitamos estadísticas respecto de la fecha podemos consular por día, mes y año en forma separada.



8.7 Arreglos de estructuras: Acceso a sus elementos

El acceso a los elementos se realiza como en un arreglo normal, se escribe el nombre del arreglo seguido del índice del elemento al que queremos acceder.



Ahora, veremos cómo acceder a los miembros de los elementos del arreglo:

```
Colección_Cds[0].titulo = "El gran Cd";        cin>>Colección_Cds[0].num_pistas ;
Colección_Cds[1].num_pistas = 12;              cout << Colección_Cds[1].precio ;
```

Ejemplo:
A continuación se presenta un ejemplo de estructura y arreglo escrito en lenguaje C++

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

struct fecha{
    int dia;
    int mes;
    int agno;
};

struct estudiante{
    float nota;
    char nombre[20];
    char matricula[11];
    int carrera;
    fecha fecha_ing;
    fecha fecha_tit;
};


void Ingresar(estudiante []);
void Mostrar(estudiante []);

int main()
{
    estudiante est[5];

    Ingresar(est);
    Mostrar(est);

}
```

Definición de la estructura

Definición de la estructura

Prototipo de la función

est es de tipo estructura

est es un arreglo o vector de dimensión 5

Pasamos el objeto est a la función ingresar

Pasamos el objeto est a la función mostrar, con las modificaciones realizadas en la función Ingresar

```
void Ingresar(estudiante estudiante1[])
{
    int i;

    for (i=0;i<5;i++){
        cout << "Ingrese Nombre Estudiante: " << endl;
        cin >> estudiante1[i].nombre;
        cout << "Ingrese Nota Estudiante: " << endl;
        cin >> estudiante1[i].nota;
        cout << "Ingrese matricula Estudiante: " << endl;
        cin >> estudiante1[i].matricula;
        cout << "Ingrese carrera Estudiante: " << endl;
        cin >> estudiante1[i].carrera;
        cout << "Ingrese dia de Ingreso Estudiante: " << endl;
        cin >> estudiante1[i].fecha_ing.dia;
        cout << "Ingrese mes de Ingreso Estudiante: " << endl;
        cin >> estudiante1[i].fecha_ing.mes;
        cout << "Ingrese agno de Ingreso Estudiante: " << endl;
        cin >> estudiante1[i].fecha_ing.agno;
        cout << "Ingrese dia de Titulacion Estudiante: " << endl;
        cin >> estudiante1[i].fecha_tit.dia;
        cout << "Ingrese mes de Titulacion Estudiante: " << endl;
        cin >> estudiante1[i].fecha_tit.mes;
        cout << "Ingrese agno de Titulacion Estudiante: " << endl;
        cin >> estudiante1[i].fecha_tit.agno;
        cout << endl << endl;
    }
}
```

Estudiante1[] es un arreglo de tipo estudiante el cual toma el valor del objeto est dentro de la función Ingresar

```
void Mostrar(estudiante estudiante1[])
{
    int i;

    for (i=0;i<5;i++){
    cout << endl << endl;
    cout << "Los datos del estudiante son: " << endl;
    cout << "Nombre: " << estudiante1[i].nombre << endl;
    cout << "Nota: " << estudiante1[i].nota << endl;
    cout << "Matricula: " << estudiante1[i].matricula << endl;
    cout << "Carrera: " << estudiante1[i].carrera << endl;
    cout << "Fecha de Ingreso: " << estudiante1[i].fecha_ing.dia << "-" <<
    estudiante1[i].fecha_ing.mes << "-" << estudiante1[i].fecha_ing.agno << endl;
    cout << "Fecha de Titulacion: " << estudiante1[i].fecha_tit.dia << "-" <<
    estudiante1[i].fecha_tit.mes << "-" << estudiante1[i].fecha_tit.agno << endl;

    }
}
```

La función recibe el valor de la variable est y lo imprime en pantalla

8.8 Ejercicios Resueltos

A continuación se presenta un programa escrito en C++, en donde se registran las mascotas atendidas en una veterinaria, para ello se definen las estructuras y los prototipos de las 4 funciones utilizadas.

```
#include<iostream>

using namespace std;

struct fecha{
    int dia;
    int mes;
    int agno;
};

struct mascota{
    char nombre[10];
    char tipo[10];
    char raza[10];
    char sexo;
    int edad;
    float peso;
    char enf;
    char trat;
    fecha fechaConsulta;
};

void menu();
void recepcion(mascota []);
void atencion(mascota []);
void mostrarFicha(mascota []);

int main(){

    menu();

}
```

```
void menu(){
    int opc;
    mascota animal[100]={0};

    do{
        system("CLS");
        cout << "**********************" << endl;
        cout << "* Menu Veterinaria   *" << endl;
        cout << "**********************" << endl;
        cout << "* 1.- Recepcion      *" << endl;
        cout << "* 2.- Atencion       *" << endl;
        cout << "* 3.- Mostrar Ficha  *" << endl;
        cout << "* 4.- Salir          *" << endl;
        cout << "**********************" << endl;
        cin >> opc;
        switch (opc){
        case 1:recepcion(animal);break;
        case 2:atencion(animal);break;
        case 3:mostrarFicha(animal);break;
        default: break;
        }
        system("PAUSE");
    }while(opc!=4);
}

void recepcion(mascota animal[100]){
    int i=0;

    cout << "Bienvenido a Recepcion" << endl;
    cout << "Por favor Ingrese Datos de la Mascota" << endl;
    cout << "Nombre: ";
    cin >> animal[i].nombre;
    cout << "Tipo: ";
    cin >> animal[i].tipo;
    cout << "Raza: ";
    cin >> animal[i].raza;
    cout << "Sexo: ";
    cin >> animal[i].sexo;
}
```

```
void atencion(mascota animal[100]){
    int i=0,j=0;
//  char enfermedad[10],tratamiento[10];

    cout << "Bienvenido a Atencion" << endl;
    cout << "Por favor Ingrese Datos de la Mascota" << endl;
    cout << "Edad: ";
    cin >> animal[i].edad;
    cout << "Peso: ";
    cin >> animal[i].peso;
    cout << "Enfermedad: ";
    cin >> animal[i].enf;
    cout << "Tratamiento: ";
    cin >> animal[i].trat;
    cout << "Fecha de Consulta (dd/mm/aaaa): ";
    cin >> animal[i].fechaConsulta.dia;
    cin >> animal[i].fechaConsulta.mes;
    cin >> animal[i].fechaConsulta.agno;
}

void mostrarFicha(mascota animal[100]){
    int num,j=0;

    cout << "Bienvenido a Mostrar Ficha" << endl;

    cout << "Ingrese Numero de Mascota: ";
    cin >> num;
    cout << "Nombre: " << animal[num].nombre << endl;
    cout << "Tipo: " << animal[num].tipo << endl;
    cout << "Raza: " << animal[num].raza << endl;
    cout << "Sexo: " << animal[num].sexo << endl;
    cout << endl;
    cout << "Fecha de Consulta: " << animal[num].fechaConsulta.dia << " / "
    << animal[num].fechaConsulta.mes << " / " << animal[num].fechaConsulta.agno << endl;
    cout << "Edad: " << animal[num].edad << endl;
    cout << "Peso: " << animal[num].peso << endl;
    cout << "Enfermedad: " << animal[num].enf << endl;
    cout << "Tratamiento: " << animal[num].trat << endl;
}
```