Práctico 4:

Registros en C: Structs

La teoría general para este práctico puede consultarse en los capítulos 6 y 7 de las Notas de Clase de la cátedra.

1. Escriba código C para:

a. Definir un registro llamado **parte**, que sirve para guardar datos de piezas (partes) que contenga un campo entero numero para el número de parte y un campo string nombre para el nombre de parte cuya longitud puede ser a lo sumo de 25 caracteres (incluido el carácter nulo).

b. Definir **Parte** como un tipo struct parte.

c. Use **Parte** para declarar una variable **a** de tipo struct parte y un arreglo **b[10]** de tipo struct parte.

d. Lea un número y un nombre de parte desde el teclado y los almacene en los campos de la variable **a**.

e. Asigne los valores de los campos de la variable **a** al elemento 3 del arreglo b.

f. Imprima los campos del elemento 3 del arreglo **b**.

2. Dados los siguientes datos, declarar los registros (structs) que los representen:

a. Datos para calcular la medida de superficie de: un rectángulo, un triángulo y un trapecio.

b. Un par de coordenadas cartesianas (x, y) donde x e y son dos números reales.

c. Una dirección: calle y número, localidad, código postal, número de teléfono.

d. Un alumno: nombre (30 caracteres); número de registro (entero de seis dígitos); tipo y número de documento de identidad; dirección; materias (para cada una: nombre, fecha de regularización y de aprobación).

e. Un empleado: nombre (30 caracteres); tipo y número de documento de identidad; dirección; fecha de nacimiento; estado civil; cantidad de hijos a cargo; sexo.

f. Un aula: tipo de aula deberá incluir: Nº de identificación, ubicación (bloque) y tipo (Laboratorio, Conferencia, Teoría-Práctica); cantidad de mesas, cantidad de sillas, cantidad de pizarras, proyector (deberá incluir: marca, color y control remoto (si o no)).

3. Hacer una función que ingrese los datos de un aula y otra que los muestre por pantalla. Probar el uso combinado de las dos funciones en un programa.

4. Diseñar un registro para mantener el registro de salud de una persona. Los campos del registro deberán incluir el primer nombre de la persona, su apellido, sexo, la fecha de nacimiento (consistiendo de día, mes y año), el peso (en Kg) y la altura (en metros). El programa deberá contar con una función que pida los datos y los almacene en cada uno de los campos del registro de salud. Además, deberá incluir funciones que calculen y retornen la edad en años y el índice de masa corporal (IMC). El programa deberá solicitar la información de la persona, crear el registro para la persona y mostrar la información almacenada en dicho registro, incluyendo el nombre y apellido, el género, la fecha de nacimiento, la altura y el peso; luego deberá calcular y mostrar la edad en años de la persona y si índice de masa corporal, así como en qué categoría se encuentra de acuerdo a su IMC. El IBM se calcula por la fórmula

IMC = peso en kg / altura en metros

Se considera bajo de peso cuando el IMC es menor que 18.5; normal cuando es mayor o igual a 18.5 y menor que 25; con sobrepeso cuando es mayor o igual que 25 y menor a 30, y obeso cuando el IMC es 30 o más.

5. Hacer un programa que permita opcionalmente: (a) ingresar los datos de aulas en un arreglo mientras el usuario lo desee; (b) mostrar por pantalla los datos de cada uno de las aulas ingresadas. Realizar esto utilizando funciones.

6. Considerando la estructura de empleado definida en el ejercicio 2 e) se pide:

(a) declarar el tipo empleado y la constante MAX con valor 100.

(b) declarar en el main un arreglo reg_empleados de manera tal que pueda almacenar la cantidad MAX de empleados.

(c) realizar una función que permita cargar los datos de un empleado.

(d) realice una función que permita cargar n empleados en el arreglo, donde n es ingresado por el usuario.

(e) realizar una función que dado el tipo y numero de documento muestre por pantalla los datos del mismo, en caso de no encontrarlo en el arreglo debe informarlo por pantalla.

(f) realice un menú de opciones que permita realizar las funcionalidades de los puntos (d) y (e) las veces que desee el usuario, siempre y cuando sea posible.

Ejercicios Complementarios

1. Completar los espacios en blanco con lo que corresponda:

a. Los elementos que forman parte de un registro son llamados _____.

b. La palabra clave _____ introduce una declaración de registro en C.

c. En C, la palabra clave _____ es usada para definir un sinónimo de un tipo de dato previamente definido.

2. Indicar si son verdaderas o falsas las siguientes afirmaciones:

a. Un registro es una colección de datos relacionados lógicamente, bajo un mismo nombre.

b. Los registros sólo pueden contener campos que sean todos de un mismo tipo.

c. Dos structs no pueden compararse usando los operadores == y !=.

d. Una variable de tipo struct no se puede asignar a otra variable del mismo tipo.

e. En una declaración de una variable el nombre del struct es opcional.

f. Los campos o miembros de distintos registros deben tener nombres únicos.

g. La palabra clave typedef es usada para definir nuevos tipos de datos.

h. Los structs en C son siempre pasados por referencia a las funciones.

3. Dadas las siguientes definiciones y declaraciones en C:

```
typedef struct fecha {
      int dia, mes, anio;
      } Fecha;


struct emp{
      char nombre[30];
      unsigned int nroLegajo;
      unsigned short int edad;
      char estadoCivil;
      Fecha fechaIngreso;
} empleado;
struct espo{
      char nombre[30];
      Fecha fechaNac;
      unsigned short int edad;
```

```
} esposa;
```

¿Cuáles de los siguientes grupos de sentencias son válidas?

a. empleado.nombre[10]= esposa.nombre[10];

empleado.edad = esposa.edad;

b. emp.nombre[10] = espo.nombre[10];

emp.fechaIngreso =espo.fechaNac;

c. empleado.fechaIngreso = {1, 1, 2010};

d. edad = 26; fechaIngreso.dia = 1; estadoCivil = ‘c’;

nombre ="Pedro Rodriguez";

e. Fecha.dia = 10; Fecha.mes = 8; Fecha.anio = 1910;

f. struct espo esposo = {"Juan", {10,4,1978}, 35};

esposa = {"Maria", {1,1,1980}, 33};