UNIVERSIDAD DE GUADALAJARA

Red Universitaria e Institución Benemérita de Jalisco

Centro Universitario de Ciencias Exactas e Ingenierias

Departamento de Ciencias Computacionales

Programación

Registros

Profesora: Patricia Sanchez Rosario

Alumno: Jonathan Silva Morales

Codigo: 216852287

Carrera: Ingenieria en Computación (INCO)

Materia: I5882

NRC: 42555

Sección: D07

Ciclo: 2020B

Estructuras de Registros.

Los Arrays, son un ejemplo de estructuras (de tipo homogéneo), sin embargo, éste tipo de estructuras posee una gran limitante, puesto que, sólo admite datos del mismo tipo (entero, flotante, carácter); las estructuras que vamos a estudiar, están compuestas por un grupo de variables, no necesariamente del mismo tipo, en las cuales, podemos almacenar diferente información (en cuanto a tipo), pero referente a un mismo tópico.

Por ejemplo, en nuestra INE, de la universidad, aparecen muchos datos acerca de nosotros. Nombres, apellidos, edad, dirección, fecha de nacimiento, grado, sección, ciclo...

Estos datos, son de diferente tipo, pero, en C, podemos almacenarlos utilizando un tipo de dato registro al que llamamos Estructura.

Por tanto, una estructura es una colección de una o más tipos de elementos denominados miembros, cada uno de los cuales puede ser de un tipo de dato diferente.

Es ello que radica la importancia de las estructuras, porque nos ahora tiempo además que las estructuras son una herramienta importante para la creación de programas potentes y bases de datos.

Declaración de una Estructura

Como toda variable, en C, debemos declarar una estructura para poder hacer uso de ella, y la forma de hacerlo es la siguiente:

```
struct <nombre de la estructura>
{
        <tipo de dato del miembro1> <nombre del miembro 1>;
        <tipo de dato del miembro 2> <nombre de miembro 2>:
        ...
        <tipo de dato del miembro n> <nombre del miembro n>;
}
```

Por ejemplo, para los datos anteriores, la forma de declararla sería la siguiente:

```
struct datos
{
        char nombre[30];
```

```
        char apellido[20];
        int edad;
        char dirección[100];
        char fecha_nac[8];
};
```

Definición de variables del tipo estructura

Al igual que las funciones, las estructuras son declaradas y definidas; en la declaración es que, le estamos indicando al compilador que las variables de tipo datos estarán compuestas por los elementos, tales como nombre, apellido etc; pero además, debemos definir variables que, serán de ese tipo. Así como en alguna parte de C, están declarados los tipos de datos char, int, float; así también nosotros debemos definir las estructuras. Y cuando en el main, definimos las variables que serán del tipo int, del tipo float o del tipo char, de igual manera, debemos definir que variables serán del tipo de la estructura que hemos creado. Para ello existen dos procedimientos:

1. listar las variables inmediantamente después de cerrar la llave de la estructura
2. Listar las variables que serpán del tipo estructura creadas, inmeditamente después del identificador de la estructura; en la zona de declaraciones del programa.

Ejemplo:

```
1. struct datos
{
        char nombre[30];
        char apellido[20];
        int edad;
        char dirección[100];
        char fecha_nac[8];
} alumno1, alumno2, alumno3;

2. struct datos alumno1, alumno2, alumno3;
```

Si por algún caso, los datos de las tres variables, fuesen los mismos, podemos asignar los valores de ésta forma:

```
Alumno1=alumno2;
Alumno3=alumno2;
```

O también:

```
Alumno1=alumno3=alumno2;
```

Ya que, al contener los mismos miembros, es como si se tratacen de datos tipo int, float o char, por consiguiente podemos hacer las asignaciones anteriores.
Una estructura la podemos inicializar así:

```
struct datos
{
        char nombre[30];
        char apellido[20];
        int edad;
        char dirección[100];
        char fecha_nac[8];
}alumno1 = {“Manuel”,
                “Ortez”,
                20,
                “San Salvador, El Salvador”,
                “27/04/86”,
               };
```

o también así:

```
struct datos alumno2={“Carolina”,
                      “Pelayo”,
                        20,
                        “San Salvador, El Salvador”,
                        “14/05/86”,
                     };
```

El tamaño de una estructura es determinado de forma muy simple, consiste en sumar, el tamaño de todos los miembros, para nuestro caso particular, el tamaño (en bytes) de la estructura datos sería:

Ciertamente que, este proceso, lo podemos simplificar utilizando la sentencia:

Sizeof(datos);
Cuyo resultado será: 160.

Acceso a una estructura.

Para acceder a una estructura, o bién a la información guardada en ella, podemos hacer uso de dos nuevos amigos:

1. El operador punto (.)
2. El operador puntero –flecha- (->)

Por ejemplo:

```
Alumno1.nombre=”Manuel”;
```

Strcpy(alumno1.apellido, “Ortez”);

Para utilizar el operador flecha, debemos hacer uso de punteros (y creíste que ya nos habíamos olvidado de los punteros verdad?).
Por ejemplo, si tenemos un puntero a una estructura:

```
.struct datos *ptr;
ptr=&alumno1;
ptr->edad=20;
```

Estructuras Anidadas

Al igual que los ciclos, las decisiones, las expresiones, etc, las estructuras también pueden estar dentro de otras, a esto es que se le llama estructuras anidadas.
Supongamos que tenemos dos estructuras siguientes:

```
stuct empleado
{
        char nom[30];
        char puesto[10];
        int edad;
        float sueldo;
        char municipio[20];
        char ciudad[10];
        char dirección[50];
};

struct cliente
{
        char nom[30];
        char fecha_deuda[8];
        float saldo;
        char municipio[20];
        char ciudad[10];
        char dirección[50];
};
```

Observamos que, en ambas estructuras, hay fatos que se repiten, los cuales los podríamos ubicar en otra structura, así:

```
struct direc
{
char municipio[20];
        char ciudad[10];
        char dirección[50];
};
```

por tanto, las estructuras de empleado y cliente, sería de la siguiente forma:

```
stuct empleado
{
        char nom[30];
```

```
        char puesto[10];
        int edad;
        float sueldo;
        struct direc direc_empleado;
};

struct cliente
{
        char nom[30];
        char fecha_deuda[8];
        float saldo;
        struct direc direc_cliente;
};
```

En C, podemos definir una estructura dentro de otra estructura, claro siempre y cuando la declaración de ésta, haya sido previo.