

# PROGRAMACIÓN

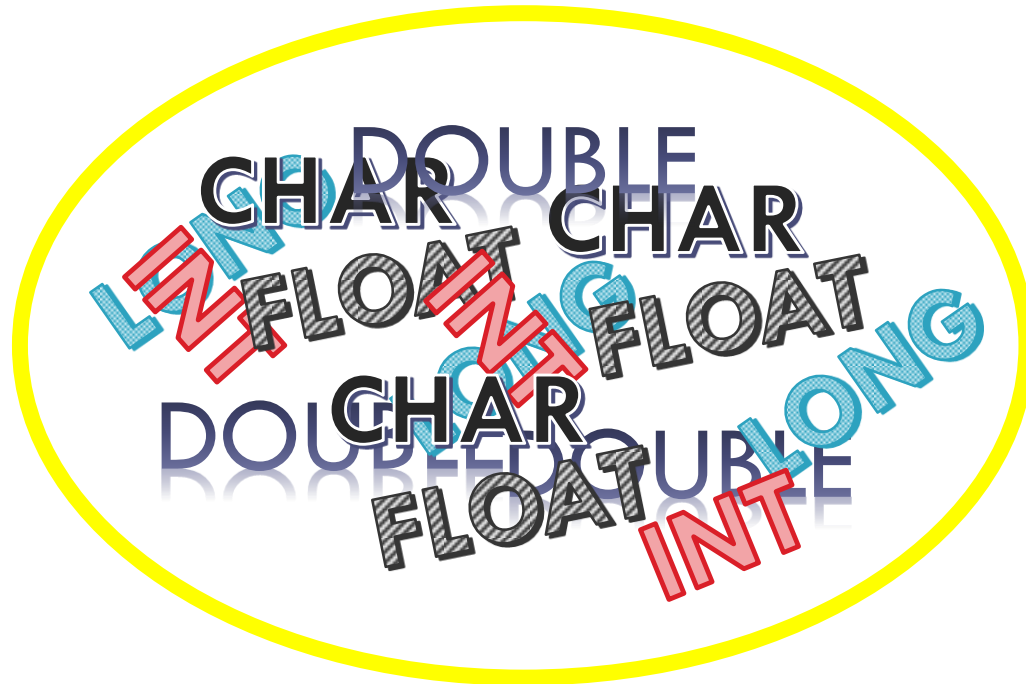
**Unidad 7 – Parte 1.** Tipos de datos derivados estructuras.  
Estructuras anidadas.

# Repasemos lo visto



# ¿Qué es una Estructura?

Una estructura es una colección de uno o mas tipos de variables, agrupadas bajo un mismo nombre para un manejo conveniente.



# Ejemplo de Estructuras

Las estructuras permiten asociar información, o lo que es lo mismo, permiten que un grupo de variables relacionadas sean tratadas como una unidad en lugar de entidades separadas.



## Ejemplo:

Una idea sencilla de una estructura son los datos de una persona:

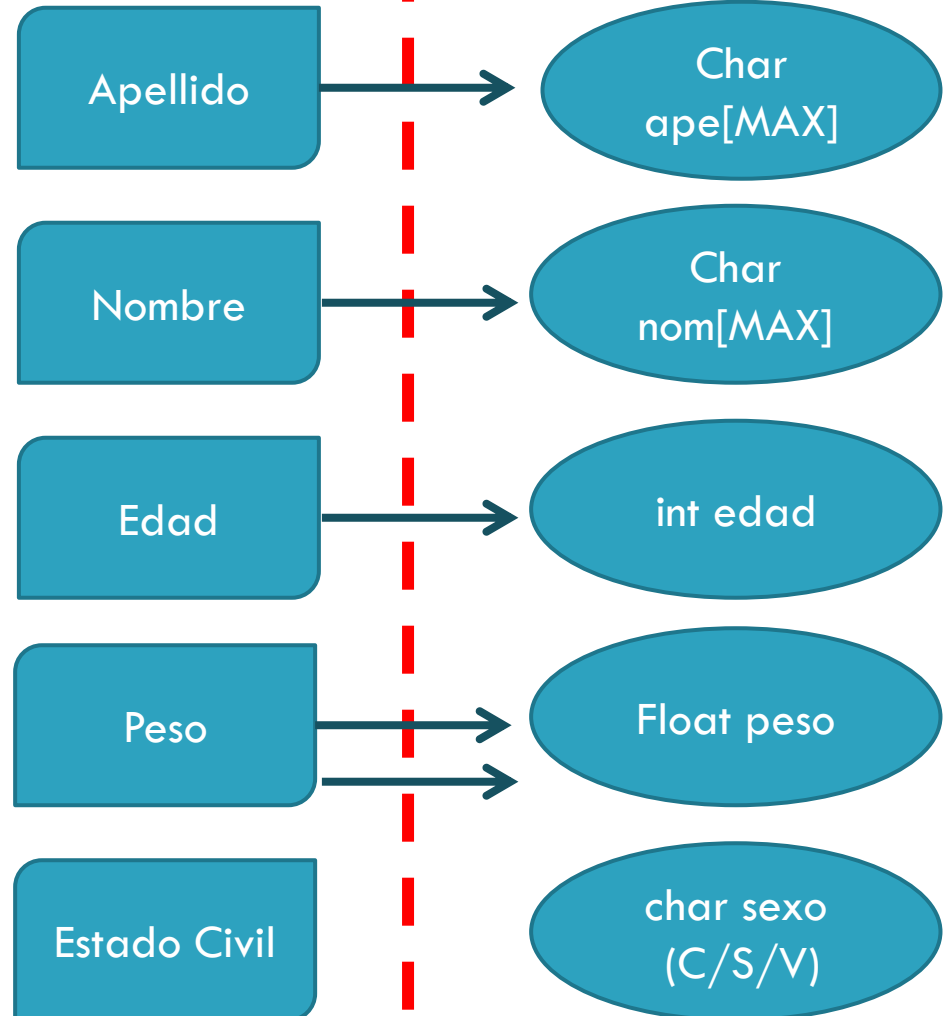
- Nombre
- Apellido
- Edad
- Fecha de nacimiento
- Peso
- Estado Civil

# Ejemplo de Estructuras




Datos

Tipo



# Características generales de las estructuras



- ❑ Tienen un nombre
- ❑ Elementos de distintos tipos llamados campos o miembros
- ❑ Acceso a las componentes vía índice o puntero

# ¿Como crear una Estructura?

Se habrá de usar la Palabra Reservada : **struct**

```
struct [nombre] {  
    tipo1 var1;  
    tipo2 var2;  
    :      :  
    tipoi  vari;  
    :      :  
    tipon  varn;  
};
```

- **Nombre:** optativo, se llama rótulo o etiqueta de la estructura.
- **tipo<sub>i</sub> var<sub>i</sub>:** son los campos o miembros de la estructura. Son variables que se declaran con su tipo asociado, que podrá ser cualquier tipo válido.

# Problema:

Suponga que usted quisiera llevar registro de sus compras en mercado libre. Querría agrupar para su referencia la siguiente información de sus comprar:



- Nombre del producto
- Categoría
- Precio
- Cantidad
- Forma de pago
- Fecha de entrega

Crear una estructura que cumpla con la información detallada



**mercado  
libre**



# Declaración de variables tipo estructurado

Una vez definido el nuevo tipo de datos, se estará en condiciones de declarar variables asociadas a el, por ejemplo:

```
#include <stdio.h>
#define MAX 100

struct persona {
    char nom[MAX];
    char ape[MAX];
    int edad;
    float peso;
    char sexo;
};
```

```
int main(){
    struct persona PACIENTE;
    struct persona CLIENTE;
```

```
int main(){
    struct persona PACIENTE, CLIENTE;
```

La declaración de las variables “PACIENTE” y “CLIENTE” asociadas al tipo persona SI IMPLICA UNA RESERVA DE MEMORIA.

# INICIALIZACION DE UNA VARIABLE ASOCIADA AL TIPO ESTRUCTURA

Se puede inicializar una variable asociada al tipo estructura en su declaración, ellos se hará siguiendo su declaración con una lista de inicializadores, cada uno con una expresión constante para sus miembros.

```
#include <stdio.h>

struct PUNTO {    int puntoX;
                  int puntoY;
};

int main()
{
    struct PUNTO punto1= { 7, 5};
}
```

# COMO ACCEDER A LOS CAMPOS O MIEMBROS DE UNA ESTRUCTURA

- El acceso a los miembros o campos: de una variables asociada al tipo estructura se hace a través del **OPERADOR PUNTO**

- La forma de uso del mismo es:

**Nombre\_de\_la\_variable.nombre\_del\_campo**

# Ejemplo

```
#include <stdio.h>
#define MAX 50

struct persona{
    char nom[MAX];
    char ape[MAX];
    int edad;
    float peso;
    char sexo;
};

int main()
{
    struct persona PACIENTE;
    puts("Ingrese la edad del paciente:");
    scanf("%d", &PACIENTE.edad);

    puts("Ingrese el peso del paciente:");
    scanf("%f", &PACIENTE.peso);

    puts("Los datos ingresados son: ");
    printf("Edad: %d \n", PACIENTE.edad);
    printf("Peso: %.3f", PACIENTE.peso);

    return 0;
}
```

# OPERACIONES



Con las variables estructuradas se pueden llevar a cabo las siguientes acciones:

- copiar como una unidad
- asignar entre ellas
- acceder y trabajar con sus campos
- anidar estructuras
- usar como argumentos de funciones: completas y por partes
- tomar su dirección (&)
- ser devueltas por funciones

# COMO CARGAR DATOS EN UNA VARIABLE ESTRUCTURADA

- Una vez declarada la variable tipo estructura, veamos de que manera podemos acceder a los miembros o campos que componen la misma.
- Si considero estructura persona, declaro una variable asociada a la misma de nombre individuo, esto es: **struct persona individuo;**

# COMO CARGAR DATOS EN UNA VARIABLE ESTRUCTURADA

Podemos usar esta variable para conocer de que manera puedo acceder a los campos de la variable. Si lo que me propongo es por ejemplo, “cargar los datos del individuo”, deberé para ello usar el operador que me permite acceder a los campos de la variable estructurada, o sea el operador punto, de la siguiente manera:

```
gets(individuo.ape);
```

```
scanf("%d", &individuo.edad);
```

## A cartoon illustration of a female nurse with brown hair tied in a bun, wearing a light blue nurse's uniform and a matching headband. She is pushing a male patient in a wheelchair. The patient has brown hair and is wearing a green long-sleeved shirt, dark pants, and orange shoes. He is sitting in a purple wheelchair with large black wheels. The nurse is standing behind the wheelchair, holding the handles. The background is plain white.

```
#include <stdio.h>

struct persona {
    char ape[100];
    char nom[100];
};

int main()
{
    struct persona PACIENTE1;
    struct persona PACIENTE3;

    puts("Paciente Nº 1");
    puts("Ingrese el apellido del paciente");
    gets(PACIENTE1.ape);
    puts("Ingrese el nombre del paciente");
    gets(PACIENTE1.nom);

    PACIENTE3 = PACIENTE1;

    puts("El nombre del Paciente 1 es:");
    puts(PACIENTE1.nom);

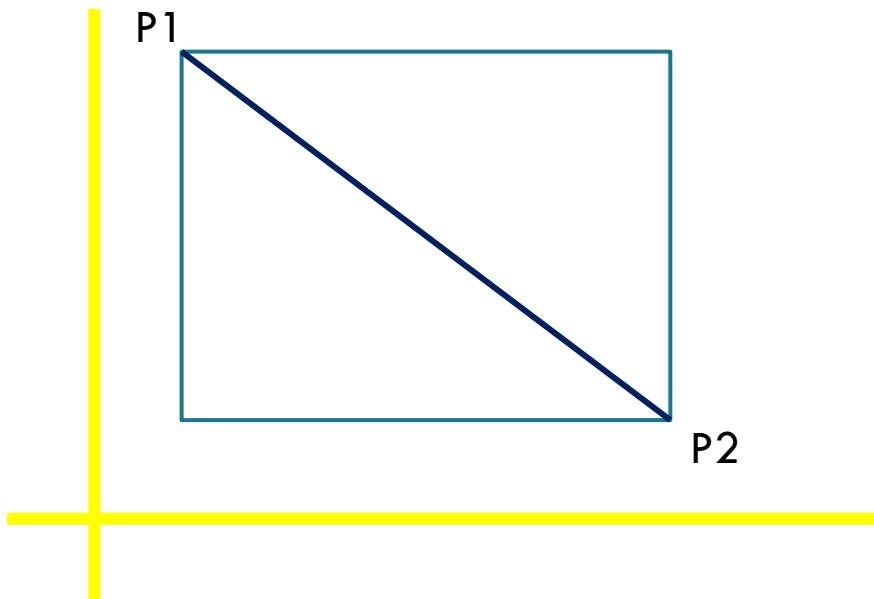
    puts("El nombre del Paciente 3 es:");
    puts(PACIENTE3.nom);

    return 0;
}
```



# ESTRUCTURAS ANIDADAS

Las estructuras pueden anidarse. Una representación de un rectángulo se puede hacer como un par de puntos que denotan las esquinas diagonalmente opuestas



```
#include <stdio.h>

struct PUNTO {
    int puntoX;
    int puntoY;
};

struct RECTA {
    struct PUNTO P1;
    struct PUNTO P2;
};
```

# Typedef

- Typedef permite crear identificadores para algún tipo de dato que ya existente.
- Es una facilidad proporcionada por el lenguaje C para trabajar con identificadores
- Typedef no introduce nuevos tipos, sólo formas sinónimos para tipos que se podrían mencionar de otra forma.

```
#include <stdio.h>

int main()
{
    typedef int entero;

    entero numero=5;

    printf("El valor del número es: %d", numero);

    return 0;
}
```

Hace del nombre **entero** sea sinónimo de **int**.

```
#include <stdio.h>

struct cd{
    char nombre[30];
    char interprete[25];
    int cant_temas;
    int duracion;
};

typedef struct cd Discos;

int main()
{
    Discos mios;
}
```

Hace que el nombre **Discos** sea sinónimo de **struct cd**.

```
#include <stdio.h>

struct
{
    char nombre[30];
    char interprete[25];
    int cant_temas;
    int duracion;
}typedef Discos;

int main()
{
    Discos mios;
}
```

Define el nombre de la estructura en la misma declaración, evitando el uso de la palabra **struct**.

# Retomemos nuestro Ejercicio:



1. ¿En que caso podríamos tener estructuras anidadas en nuestro ejercicio?
2. Cargue los datos de dos productos comprados.
3. Muestre por pantalla cada uno de los campos de la estructura.
4. Compare los precios de ambos productos, e indique cual es el mas costoso.



mercado  
libre

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

```
int main()
```

```
{
```

```
    printf("Hasta la próxima clase!!\n");
```

```
    return 0;
```

```
}
```