

# Curso Básico de C

## Clase #7

*Registros, Arreglos de Registros y  
Definición de Tipos*

**Prof. Patricia Guerrero**

# Registros

**Un registro** es un conjunto de una o más variables, cuyo tipo de dato es igual o diferente, agrupadas bajo un mismo nombre.

La sintaxis de su declaración es la siguiente:

```
struct nom_registro  
{  
    tipo variable1;  
    tipo variable2;  
    tipo variable3;  
};
```

Nombre del  
registro

Variables dentro  
del registro

# Registros

- **Definición de variables tipo registro:**

```
struct trabajador  
{  
    char nombre[20];  
    char apellidos[40];  
    int edad;  
    char puesto[10];  
};
```


```
struct trabajador fijo, temporal;
```

Definición de las  
variables fijo y  
temporal

# Registros

- **Definición de variables tipo registro:**

```
struct trabajador  
{  
    char nombre[20];  
    char apellidos[40];  
    int edad;  
    char puesto[10];  
}fijo, temporal;
```



Definición de las  
variables fijo y  
temporal

# Registros

- **Declaración e inicialización de registros:**

```
struct estudiante  
{  
    char nombre[20];  
    int edad;  
};
```

Los registros deben declararse antes de la función principal

```
struct estudiante est1;  
est1.nombre = "Juan";  
est1.edad = 39;
```

Inicialización de registros

# Registros

- **Inicialización de registros:**

```
struct trabajador fijo = {"Pedro", "Hernández Suárez", 32,  
"gerente"};
```

```
struct notas  
{  
    char nombre[30];  
    int notas[3];  
};
```

```
struct notas alumno = {"Carlos Pérez", {9, 6, 10} };
```

Si uno de los campos es un arreglo de números, los valores de la inicialización deben ir entre llaves

# Arreglos de Registros

Es posible agrupar un conjunto de elementos de tipo registro en un arreglo:

```
struct trabajador  
{  
    char nombre[20];  
    char apellidos[40];  
    int edad;  
};  
  
struct trabajador fijo[20];
```

Aquí podemos  
almacenar los datos de  
hasta 20 trabajadores

# Arreglos de Registros

- **Acceso a los campos de un registro:**

```
struct trabajador fijo[20];
```

```
fijo[3].nombre = "Maria";  
printf("%s", fijo[3].nombre);
```

Declaración e  
inicialización

```
struct trabajador fijo[20] = { {"José", "Herrero  
Martínez", 29}, {"Luis", "García Sánchez", 46} };
```



# Definición de Tipos

C dispone de una declaración llamada **typedef** que permite la creación de nuevos tipos de datos.

```
typedef int entero;  
entero a, b = 3;
```

Creación del tipo  
de dato **entero**

```
struct trabajador  
{  
    char nombre[20];  
    int edad;  
};
```

```
typedef struct trabajador empleado;  
empleado fijo, temporal;
```

Creación del tipo  
de dato **empleado**

# Registros

- **Ejercicio 1:**

Diseñe un programa que lea los datos asociados a N estudiantes (nombre y cédula), así como la información correspondiente a las materias inscritas (código, nombre y nota final). Tome en cuenta que un estudiante sólo puede inscribir un máximo de cinco (5) materias. El programa debe generar las siguientes salidas:

- Promedio de notas por estudiante.
- Promedio de notas por todos los estudiantes.
- Número de estudiantes con cédulas mayores a 19.000.000.
- Nombre del estudiante con el promedio más alto.

# Registros

- **Ejercicio 1 - Continuación:**

Asimismo, considerando que los estudiantes inscritos en la materia X presentaron un examen, registre los datos asociados a ellos (nombre y calificación), con el fin de proporcionar la siguiente información:

- Número de estudiantes inscritos en la materia X.
- Número de estudiantes aprobados y número de estudiantes reprobados.
- Calcular la media del curso.
- Porcentaje de aprobados y reprobados.

# Registros

- **Ejercicio 2:**

Una compañía de ventas está distribuida en 10 oficinas. En cada oficina existe una línea telefónica, la cual permite un máximo de 30 llamadas salientes. Ahora bien, la compañía desea llevar un control del número total de llamadas (entrantes/salientes) y del importe total asociada a las mismas.

Por cada llamada se debe registrar la siguiente información: número y lugar de destino, importe, duración, fecha y hora. Por otra parte, la compañía desea conocer cuál es la oficina con el mayor número de llamadas salientes y aquella con el mayor importe acumulado, así como determinar el promedio de llamadas entrantes y salientes entre todas las oficinas.