# Tipos estructurados en C++

UNIDAD 3

PROGRAMACIÓN AVANZADA

# Tipos estructurados en C++

Arreglos y Registros. Declaración. Definición. Acceso a elementos. Operaciones básicas. Combinación arreglos y registros.

# Tipos estructurados en C++

Los ***arreglos*** son estructuras de datos que contienen un número determinado de elementos (su tamaño) y todos los elementos han de ser del ***mismo tipo de datos***. Esta característica supone una gran limitación cuando se requieren grupos de elementos con tipos diferentes de datos cada uno.

Por ejemplo,

si se dispone de una *lista de superficies*, es muy útil un ***arreglo***;

si se necesita una *lista de información de clientes* que contengan datos tales como el *nombre*, la *edad*, la *dirección*, el *número de cuenta*, etc., los arreglos no son adecuados. La solución a este problema es utilizar un tipo de dato conocido como ***registros***.

# Registros/Estructuras

Un **registro** es un **dato estructurado**, formado por elementos lógicamente relacionados, que pueden ser del mismo o de distinto tipo, a los que se denomina ***campos***.

Los campos de un registro podrán ser de un tipo estándar o de otro tipo registro previamente definido.

Los registros también se conocen como ***estructuras***, y sus elementos se conocen como ***miembros.***

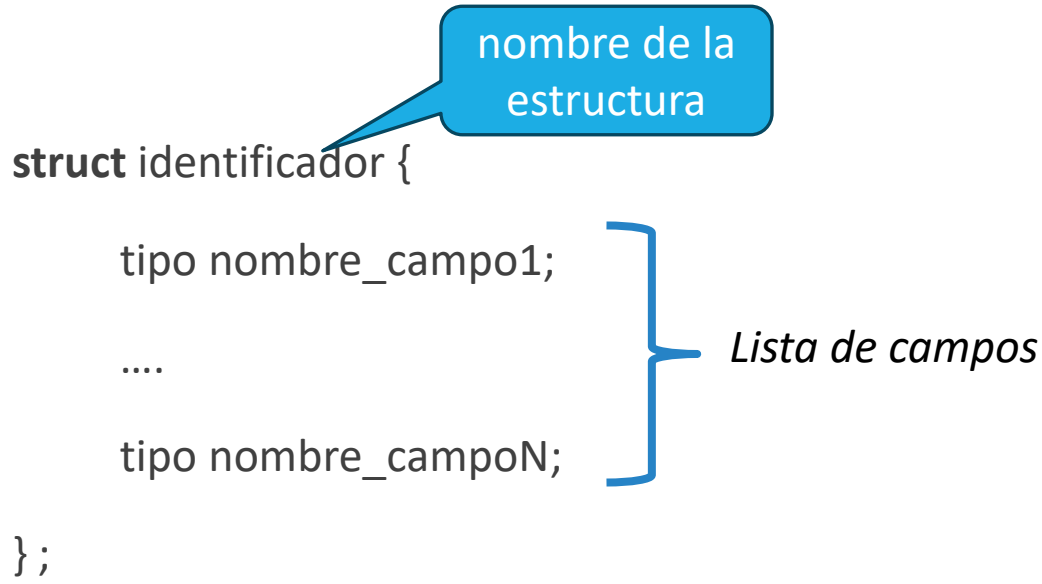
# Registros/Estructuras

En C++ las estructuras (o registros) se declaran con la palabra reservada ***struct***

Un ***struct*** es un tipo de datos compuesto conformado por un conjunto de campos de otros tipos (*básicos o compuestos*) asociados a **un identificador**.

# Registros/Estructuras. Declaración

Una declaración especifica simplemente el nombre y el formato de la estructura de datos, pero no reserva almacenamiento en memoria.



# Registros/Estructuras. Definición

Una ***definición de variable*** para una estructura dada crea un área en memoria en donde los datos se almacenan de acuerdo con el formato estructurado declarado.

Las ***variables de estructuras*** se pueden definir de dos formas:

1. listándolas inmediatamente después de la **llave de cierre** de la declaración de la estructura, o
2. listando el nombre de la estructura seguida por las variables correspondientes en cualquier lugar del programa antes de utilizarlas.

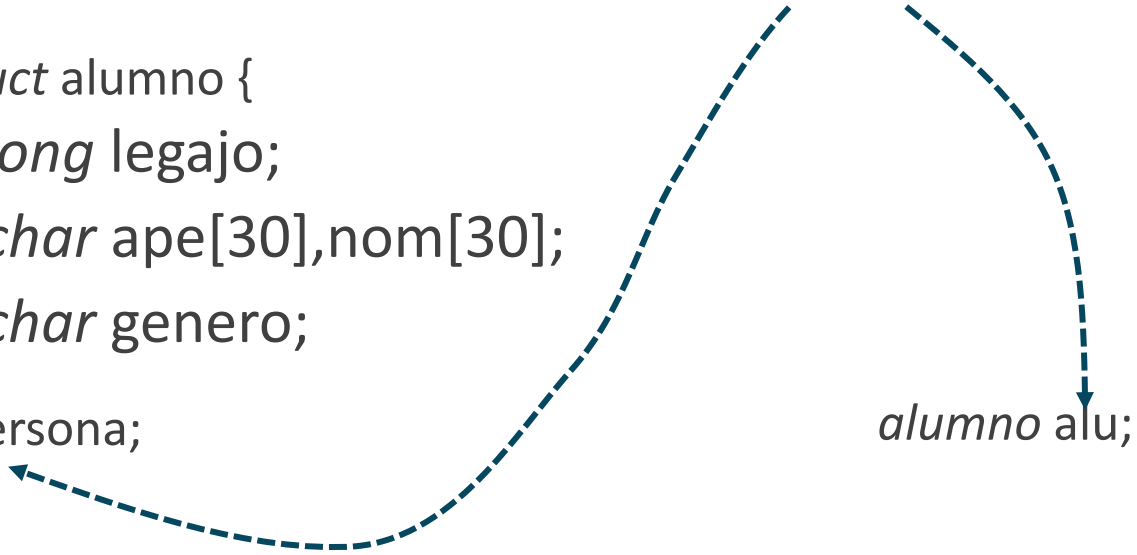
# Registros/Estructuras

**DECLARACION**

```
struct alumno {
    long legajo;
    char ape[30],nom[30];
    char genero;
} persona;
```

**DEFINICION**

```
alumno alu;
```



# Tamaño de una estructura

sizeof.cpp

```cpp
// sizeof.cpp
#include <iostream>
using namespace std;

//declarar una estructura Persona
struct Persona {
char nombre[30];
int edad;
float altura;
float peso;
};

void main(){
    Persona persona;
    cout << "sizeof(Persona): " << sizeof(Persona);
}

```

Al ejecutar el programa se produce la salida:

**sizeof(Persona): 40**

El resultado se obtiene determinando el número de bytes que ocupa la estructura

| Persona | Miembros dato | Tamaño (bytes) |
|---|---|---|
| nombre[30] | char(1) | 30 |
| edad | int(2) | 2 |
| altura | float(4) | 4 |
| peso | float(4) | 4 |
| *Total* | | 40 |

# Acceso a Estructuras

Cuando se accede a una estructura, se **almacena** información en la estructura o se **recupera** la información de la estructura.

Se puede acceder a los ***miembros*** de una estructura de dos formas:

1) utilizando el operador punto (**.**), o bien
2) utilizando el operador puntero **->**.

# Acceso a una estructura

**Acceso a una estructura de datos mediante el operador punto**

El operador (.) se utiliza para especificar un miembro de una variable de tipo estructura. La sintaxis en C++ es:

```
<nombre variable estructura> . <nombre miembro>
```

# Estructuras. Operaciones

**Lectura/Escritura**

- Campo por campo

**Lectura/Escritura**

```
cin>>alu.legajo;
cout<<alu.legajo;
```

# Arreglos y Registros

- Arreglos de registros
- Registros de arreglos
- Registros de registros

# Arreglo de estructuras

**DECLARACIÓN**

```
struct alumno {
    long legajo;
    char ape[30],nom[30];
    char genero;
};
```

**DEFINICIÓN**

```
alumno alu[30];
```

**USO**

```
cout<< alu[0].legajo;
```

# Práctica #1

Un empleador desea llevar el registro de sus empleados. Los datos a considerar por cada empleado son: ***Legajo, Nro de documento, Apellido y Nombre***. Desarrollar la carga de empleados y la muestra de datos mediante funciones.

Mediante un menú el usuario podrá elegir entre

- Cargar datos
- Mostrar datos de un empleado X
- Mostrar datos de todos los empleados
- Salir del programa

# Arreglos y Estructuras

En las combinaciones anidadas

- Estructuras de arreglos
- Estructuras de estructuras

Existirán algunos campos que serán de tipo *struct* y/o de tipo arreglo.

# Estructura de arreglos

**DECLARACIÓN**

```
struct alumno {
    long legajo;
    char ape[30],nom[30];
    char genero;
    float promedioAnual[5];
};
```

**DEFINICIÓN**

```
alumno alu[30];
```

**USO**

```
alu[0].promedioAnual[1]=0.0;
```

# Estructura de estructura

**DECLARACIÓN**

```
struct fecha{
    short día,mes,anio;
};

struct alumno {
    long legajo;
    char ape[30],nom[30];
    char genero;
    fecha fechaIngreso;
};
```

**DEFINICIÓN**

```
alumno alu[30],dato;
```

**USO**

```
cout<< alu[0].fechaIngreso.dia;
cout<<dato.fechaingreso.día;
```

# Práctica #2

Teniendo en cuenta la Práctica #1, modificar la estructura tal que, además, se pueda registrar por cada empleado: *CUIL, fecha de Ingreso, inasistencias por mes.*

# Práctica #3

Realizar un programa que gestione los datos de stock de un Kiosco, la información a recoger por cada producto será: código, *nombre, precio unitario, cantidad en stock*. El kiosco dispone de 10 productos distintos. El programa debe ser capaz de:

a) Agregar un nuevo producto.

b) Buscar un producto por su nombre.

c) Modificar el stock y precio de un producto dado.

# Actividad

REALIZAR LA ACTIVIDAD INDICADA EN EL AULA VIRTUAL.