

LENGUAJE

C

Tema 5 – Estructuras de datos



TIPOS DE ESTRUCTURAS

- ❖ Una estructura es una agrupación de datos (posiblemente) heterogéneos (de distintos tipos), que se denomina bajo un único nombre, proporcionando un medio eficaz de mantener junta la información relacionada.
- ❖ Cada dato de una estructura es llamado campo o miembro.
- ❖ Cada campo de una estructura tiene un nombre que lo identifica y un tipo de datos asociado.
- ❖ Una estructura puede tener campos que sean a su vez estructuras.
- ❖ El lenguaje C tiene tres tipos de estructuras de datos:
 - ❖ Registro o estructura (struct).
 - ❖ Unión de campos (union).
 - ❖ Tipo enumerado (enum).

STRUCT

- ❖ Al declarar una estructura se indica el nombre y tipo de todos sus campos.
- ❖ De manera opcional se puede dar nombre a la estructura y/o declarar variables de ese tipo de estructura (mínimo: alguna de las dos cosas).
- ❖ Es diferente declarar una estructura que una variable de tipo estructura. (Se puede hacer ambas cosas a la vez)

```
struct nombre_estr { campos }; /*  
    Sólo estruc. */  
struct { campos } nombre_var; /*  
    Sólo variable */  
struct nombre_estr nombre_var2; /*  
    * Sólo variable */  
struct nombre_estr2 { campos }  
    nombre_var3; /* estruc. y  
    variable */
```

Ejemplos:

```
struct ficha {  
    char nombre[20], apellido1[20];  
    int edad; float nota;  
};  
struct {int a, b;} dos_enteros;  
struct ficha alumno1, alumno2;  
struct complejo {float real,  
    imaginaria;} nro_complejo;
```

STRUCT 2

- ❖ Se puede inicializar una variable de tipo estructura en el momento de declararla (y sólo ahí de esta manera) mediante una lista de valores separados por comas y entre llaves.

- ❖ Ejemplos:

```
struct complejo c2 = { -3.15, 18.25 };  
struct ficha a = {"Alberto", "Ruiz",  
"Puerta", 20, 10.0f};
```

- ❖ Para acceder a los campos de una variable de tipo estructura, se utiliza su nombre seguido de un punto y el nombre del campo al que se desea acceder.

- ❖ Ejemplos:

```
dos_enteros.a = 7;  
dos_enteros.b = dos_enteros.a - 4;  
scanf("%f", &c.real);  
c.imaginaria = c.real;  
printf("Nombre: %s\n", a.nombre);  
printf("1er ape.: %s\n", a.apellido1);  
printf("2do ape.: %s\n", a.apellido2);
```

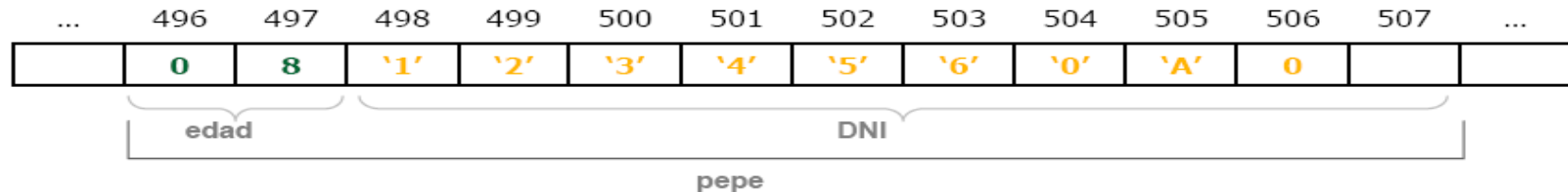
OPERACIONES CON STRUCT

- ❖ Al igual que con los vectores, no se pueden hacer operaciones básicas con estructuras completas, pero sí con sus miembros.
- ❖ Excepción: Con estructuras sí se puede hacer asignación y sí se puede devolver una estructura en una función con return.
- ❖ Los argumentos de tipo estructura se pasan por valor.

Ejemplo:

```
struct ficha lee_ficha();  
void escribe_ficha(struct  
    ficha f);  
alumno1 = lee_ficha();
```

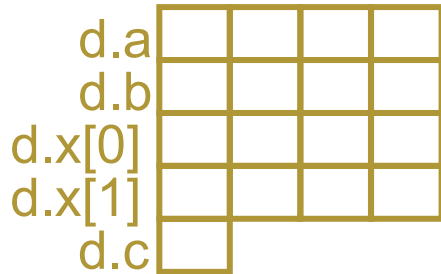
```
struct Persona {  
    int edad;  
    char DNI[10];  
};  
struct Persona pepe;  
int main(void) {
```



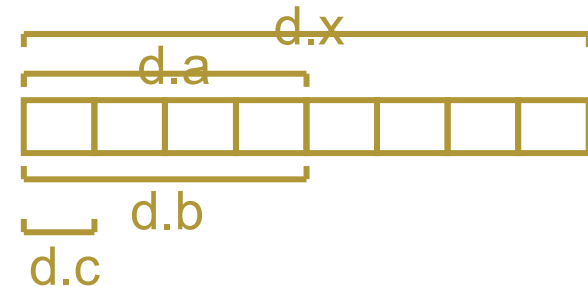
UNION

- ❖ Un union es similar a un struct, pero todos los campos comparten la misma memoria.

```
struct datos
{
    int a,b;
    int x[2];
    char c;
} d;
```



```
union datos
{
    int a,b;
    int x[2];
    char c;
} d;
```



USO DE UNION

Los union se usan para diferentes representaciones de los datos o para información condicionada:

```
union
{
    int  integer;
    char oct[4];
} data;
```

```
struct empleado
{
    char nombre[40];
    int  tipo_contrato;
    union
    {
        int  nomina;
        int  valor_hora;
    } sueldo;
} p;
```

ENUM

Las enumeraciones son conjuntos de constantes numéricas definidas por el usuario.

```
enum color {rojo, verde, azul} fondo;
enum color letras, borde=verde;

enum tipo_empleado {contratado=1,
                    temporal=2,
                    becario=3};

enum dias {lunes, martes, miercoles, jueves, viernes, sabado,
           domingo};
enum dias i, j, k;

i = martes;
for (j=lunes; j<=viernes; j++)
...;
```


DEFINICIÓN DE NUEVOS TIPOS

Las sentencias **typedef** se usan para definir nuevos tipos en base a tipos ya definidos:

```
typedef int boolean;
boolean a;
typedef struct persona Tpersona;
typedef struct punto
{
    int coord[3];
    enum color col;
} Tpunto;
TPersona p[4];
```

VECTORES DE ESTRUCTURAS

- ❖ Pueden crearse vectores de estructuras, donde cada componente del vector será una estructura.
- ❖ Para declarar un vector de estructuras, se debe definir primero la estructura y luego declarar un variable vector de dicho tipo.
- ❖ Acceso: con los corchetes se indica el elemento del vector al que acceder, en el que luego puede indicarse un nombre de campo para acceder a un miembro de la estructura.
- ❖ Ejemplos:

```
#include <stdio.h>
struct datos{
    char nombre[30];
    char direccion[30];
    int n_empleado, edad;
    float sueldo;
}
int main(){
    struct datos persona[10];
    for (int i=0;i<10;i++){
```

```
        printf("\n Ingrese número de empleado:");
        scanf("%d",&persona[i].n_empleado);
        printf("\n Ingrese nombre:");
        scanf("%s",persona[i].nombre);
        printf("\n Ingrese edad:");
        scanf("%d",&persona[i].edad);
        printf("\n Ingrese dirección:");
        scanf("%s",persona[i].direccion);
        printf("\n Ingrese sueldo:");
        scanf("%f",&persona[i].sueldo);
    }
    return 0;
}
```

BIBLIOGRAFÍA & LICENCIA

- ❖ Textos tomados, corregidos y modificados de diferentes páginas de Internet, tutoriales y documentos, entre los que destaco el libro: *C/C++ Curso de programación*, 2da Ed, Javier Ceballos, Alfaomega Ra-Ma.
- ❖ Este documento se encuentra bajo Licencia Creative Commons Attribution – NonCommercial - ShareAlike 4.0 International (CC BY-NC-SA 4.0), por la cual se permite su exhibición, distribución, copia y posibilita hacer obras derivadas a partir de la misma, siempre y cuando se cite la autoría del Prof. Matías E. García y sólo podrá distribuir la obra derivada resultante bajo una licencia idéntica a ésta.
- ❖ Autor:

Matías E. García

Prof. & Tec. en Informática Aplicada

www.profmatiasgarcia.com.ar

info@profmatiasgarcia.com.ar



www.profmatiasgarcia.com.ar