

Estructuras

Estructuras

- ◉ Son equivalentes a los registros de Pascal.
- ◉ Sintaxis para declarar un tipo estructura

```
struct Nom_Tipo {  
    tipo_campo_1 nom_campo1;  
    tipo_campo_2 nom_campo1;  
    ...  
    tipo_campo_n nom_campo1;  
};
```

Definición de estructuras

Nombre del tipo



```
struct TipoCarta {  
    char *Palo;  
    int  numero;  
}
```

Campos

- ⦿ Esta declaración no reserva memoria.
- ⦿ Es sólo una declaración de tipo.

Declaración de variables

Es opcional pero si no existe, las variables sólo se declaran junto con la estructura



```
struct TipoCarta {  
    char *Palo;  
    int numero;  
} mazo[50];  
  
struct TipoCarta carta;
```

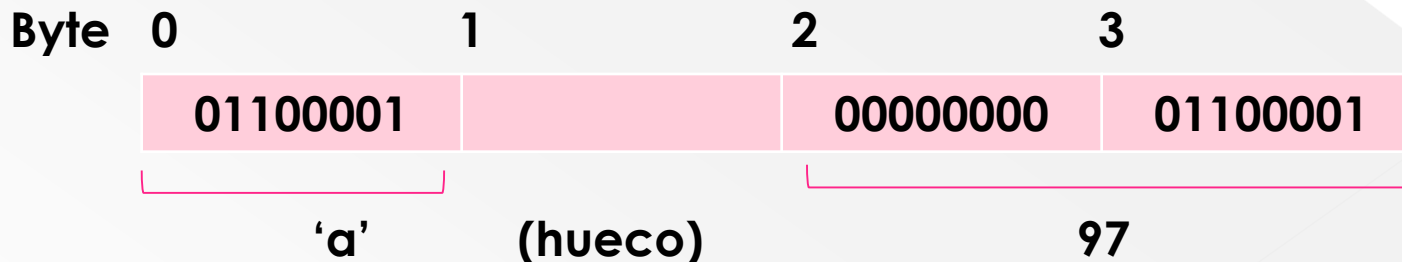
Operaciones

- ◉ Las operaciones válidas son
 - > Asignar variables de estructura a variables de estructuras del mismo tipo.
 - > Obtener la dirección de una variable estructura mediante el operador **&**.
 - > Acceder a los elementos de la estructura.

Operaciones

- Las estructuras **no** pueden compararse entre sí porque sus campos no necesariamente están almacenados en posiciones consecutivas de memoria. Puede haber "*huecos*".

```
struct ejemplo {  
    char c;  
    int i;  
} ejemplo1, ejemplo2;
```



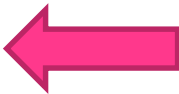
Inicialización de estructuras

- Es igual que para los arreglos, es decir que, se indica la lista de valores separados por comas y encerrados entre llaves.
- Ejemplo**

```
struct TipoCarta carta = {"Copa", 4};
```


Acceso a los campos de la estructura

**Puede usarse la notación de punto
(como Pascal)**

```
struct TipoCarta *Ptr, carta;  
scanf("%s", carta.Palo);   
Ptr = &carta;  
printf("El palo es %s\n", Ptr->Palo);
```


Acceso a los campos de la estructura

```
struct TipoCarta *Ptr, carta;  
scanf("%s", carta.Palo);  
Ptr = &carta;  
printf("El palo es %s\n", Ptr->Palo);
```



Si se accede con un puntero debe usarse ->

Acceso a los campos de la estructura

Es equivalente a (*Ptr).Palo
Los paréntesis son obligatorios

```
struct TipoCarta *Ptr, carta;
```

```
scanf("%s", carta.Palo);
```

```
Ptr = &carta;
```

```
printf("El palo es %s\n", Ptr->Palo);
```



Palabra reservada typedef

- Define un alias para un tipo existente.

Ejemplo:

```
typedef struct TipoCarta TCarta;
```

- No crea un nuevo tipo, sólo un nuevo nombre.
- Puede resultar útil tener un alias **Entero** que modifiquemos según se necesite. Por ejemplo podría ser **long int** o **short int** según el caso.

Ejemplo

- ◉ Escriba un programa que lea el nombre (20 caracteres) y la edad de 5 empleados de una empresa.
- ◉ Almacénelos en memoria utilizando un vector de 5 elementos donde c/u es una estructura.
- ◉ Al finalizar debe imprimir los nombres de quienes tengan más de 21 años.

EjStructPersona.c

```
#include <stdio.h>
int main()
{
    struct Persona{
        char nombre[30];
        int edad;
    };

    struct Persona X;

    strcpy(X.nombre, "Alguien");
    X.edad = 24;

    printf("%s tiene %d años",
           X.nombre, X.edad);
    return 0;
}
```

Utilice esta estructura para leer los datos de las 5 personas

