



Programación de Sistemas

CCPG1008

Federico Domínguez, PhD.

Unidad 1 – Sesión 5: Tipos de datos compuestos en C

Agenda

- Tipo de datos compuesto
- *struct*
- Por valor y por referencia
- Punteros
- Práctica

Un tipo de datos compuesto es creado a partir de la combinación de tipos de datos nativos...

En C, un tipo de datos compuesto puede ser creado usando ***struct***.

struct : lista de variables agregadas en un bloque de memoria contiguo bajo un mismo nombre.

Ejemplo:

```
struct Punto{  
    int x;  
    int y;  
    int z;  
};
```

Tipos de datos nativos:

- *char*
- *int*
- *float*
- *double*

Modificadores de datos nativos:

- *long*
- *short*
- *signed*
- *unsigned*

Ejemplos:

- *unsigned char*
- *long int*

Declaración estructura

```
struct Paciente {  
    char nombre[50];  
    char apellido[50];  
    char telefono[50];  
    int edad;  
    int estatura;  
    float peso;  
};
```

Declaración variable

```
int main()  
{  
    struct Paciente pac1;  
    struct Paciente pac2;
```

Uso de campos de la estructura

```
    printf("Ingrese nombre: ");  
    scanf("%s", pac1.nombre);
```

```
    printf("Ingrese apellido: ");  
    scanf("%s", pac1.apellido);
```

```
    pac1.edad = 50;  
    pac1.peso = 71.3;
```

```
}
```

Podemos usar *typedef* para definir nuestros propios tipos de datos.

```
#define TRUE 1  
#define FALSE 0
```

```
typedef unsigned char bool;
```

Alias nuevo tipo de datos



```
int main()  
{  
    bool isActive = TRUE;  
  
    if(isActive) {  
        ...  
    }  
}
```

TIP: Usar *#include <stdbool.h>* si se desea usar booleanos.

Podemos definir nuestros propios tipos de datos compuestos usando *typedef* y *struct*.

```
typedef struct Paciente {
```

```
    char nombre[50];
```

```
    char apellido[50];
```

```
    char telefono[50];
```

```
    int edad;
```

```
    int estatura;
```

```
    float peso;
```

```
} Paciente_t;
```

Nombre de la estructura



Alias nuevo tipo de datos



```
int main()
```

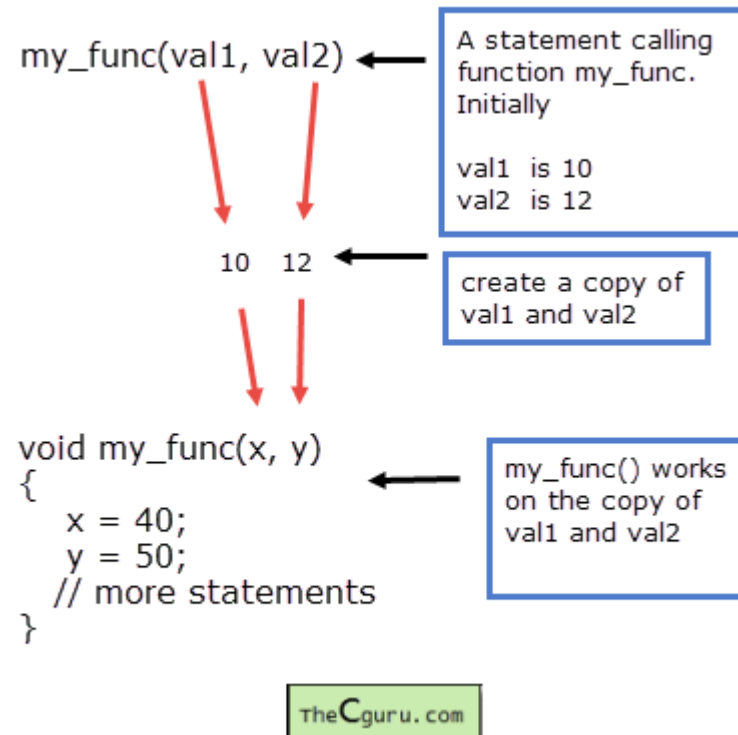
```
{
```

```
    Paciente_t pac1;
```

```
    Paciente_t pac2;
```

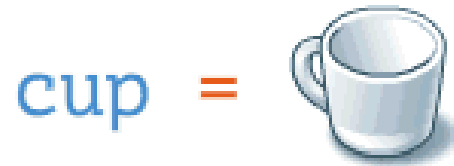
Al igual que tipos de datos nativos, los tipos de datos *struct* se pasan por valor...

Parámetros son pasados por valor...



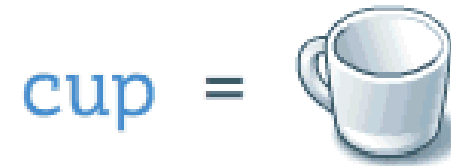
Pasar por referencia vs. pasar por valor

pass by reference



fillCup()

pass by value



fillCup()

www.mathwarehouse.com

En C, para pasar valores por referencia usamos **punteros**.

Un **puntero** es un tipo de datos que referencia una locación en memoria.

En C, se usa el asterisco '*' para indicar que un tipo de datos es puntero o también para obtener el valor de un puntero.

Además, se usa el '&' para obtener la referencia (es decir el puntero) de una variable.

```
int *a;  
int *b;  
int c = 5;
```

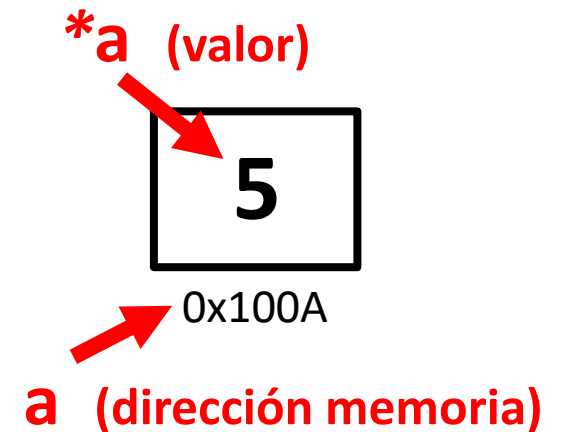
a y b son punteros

```
a = &c;  
b = a;  
c = 10;
```

a representa la dirección

***a representa el valor**

```
/* En este punto a, b apuntan al mismo sitio de memoria */  
printf("a = %d \t b = %d \t c = %d\n", *a, *b, c);
```




Si queremos modificar los datos tipo *struct*, podemos pasarlos por referencia.

```
void cambiarPesoLibras(Paciente_t *p)
{
    p->peso = p->peso * 2.2;
}
```

```
int main()
{
    Paciente_t pac1;
    printf("Ingrese peso: ");
    scanf("%f", &pac1.peso);

    cambiarPesoLibras(&pac1);

    printf("Peso: %f\n", pac1.peso);
}
```



Accedemos a los campos de un puntero a *struct* con '*->*' en lugar de punto '*'*'

Práctica: Almacenamiento de datos estructurados en memoria

Usar *struct* para almacenar información de un usuario en memoria y validarla.

TIPS: Online C compiler:

- https://www.onlinegdb.com/online_c_compiler