

PROGRAMACIÓN

Unidad 7: Tipos de datos derivados: Estructuras y Funciones – Parte 2

Repaso de lo visto



Operaciones

- Entre variables estructuradas y las funciones se pueden realizar distintas operaciones, tales como:
 - Usar como argumentos de funciones: completas o por parte.
 - Ser devueltas por funciones.
 - Tomar su dirección.

Casos Posibles



Funciones que devuelven
estructuras

Usar campos de una
estructura como
argumentos de funciones

Usar variables
estructuradas completas
como argumentos de
funciones – Pasaje por
valor

Usar campos de una estructura como argumentos de funciones

- Se trabajara con el campo de la estructura usando el operador punto.
- Cuando se pasa un campo o miembro de una estructura a una función como argumento, en realidad se pasa el valor de ese campo.

Ejemplo

Supongamos que tenemos una estructura de nombre **Alumno** que almacena los datos de un alumno como ser sus datos personales (apellido, nombre, edad) y las notas trimestrales del mismo. Nuestra tarea será realizar una función que calcule el promedio de 3 notas.

```
struct Alumno{  
    char  
    ape[100];  
    char  
    nom[100];  
    int edad;  
    float nota1;  
    float nota2;  
    float nota3;  
};
```

```
float promedio(float n1, float n2, float n3);
```

```
int main(void)  
{  
    struct Alumno alum;  
    float prom;  
    ...  
    prom = promedio(alum.nota1, alum.nota2,  
                    alum.nota3);  
    printf("El promedio es: %f", prom);  
    return 0;  
}
```

```
float promedio(float n1, float n2, float n3){  
    return((n1+n2+n3)/3); }
```

Usar variables estructuradas completas como argumentos de funciones

- Al pasar una estructura completa a una función, se usa el mecanismo de pasaje por valor.
- Sabemos que de esta manera cualquier cambio en los contenidos de los campos de la estructura dentro de la función, no se reflejan en el bloque que la invoco.

Ejemplo

```
struct Alumno{  
    char  
    ape[100];  
    char  
    nom[100];  
    int edad;  
    float nota1;  
    float nota2;  
    float nota3;  
};
```

```
float promedio(struct Alumno alum);
```

```
int main(void)
```

```
{
```

```
    struct Alumno alum;
```

```
    float prom;
```

```
    ...
```

```
    prom = promedio(alum);
```

```
    printf("El promedio es: %f", prom);
```

```
    return 0;
```

```
}
```

```
float promedio(struct Alumno alum)
```

```
{
```

```
    float promedio;
```

```
    promedio = (alum.nota1 + alum.nota2+alum.nota3)/3;
```

```
    return(promedio);
```

```
}
```


Funciones que devuelven Estructuras

Retomemos el ejemplo visto anteriormente:

```
struct Alumno{  
    char ape[100];  
    char nom[100];  
    int edad;  
    float nota1;  
    float nota2;  
    float nota3;  
};
```

```
int main(void) {  
    setvbuf(stdout,NULL,_IONBF,o);  
  
    struct Alumno persona;  
  
    persona = cargaDatos();  
  
    puts(persona.ape);  
    return 0;  
}
```

```
Struct Alumno cargaDatos(void)  
{  
    struct Alumno alum;  
  
    puts("Ingrese el apellido del alumno: ");  
    gets(alum.ape);  
  
    puts("Ingrese el nombre del alumno: ");  
    gets(alum.nom);  
  
    puts("Edad:");  
    scanf("%d", &alum.edad);  
  
    puts("Ingrese las 3 notas: ");  
    scanf("%f %f %f", &alum.nota1,  
        &alum.nota2, &alum.nota3 );  
  
    return(alum);  
}
```

Punteros a Estructuras

- Los punteros son como los punteros a variables extraordinarias.

Si partimos de la siguiente estructura:

```
Struct Punto {  
    int x;  
    int y;  
};
```

Entonces la declaración del puntero a la estructura será:

```
struct Punto *p;
```

Ejemplo

```
#include <stdio.h>
#include <stdlib.h>

struct Punto { int x, y;};

int main(void) {
    setvbuf(stdout,NULL,_IONBF,o);

    struct Punto punt, *Ppunt;

    Ppunt = &punt;

    (*Ppunt).x = 3;
    (*Ppunt).y = 5;

    printf("El punto es: %d, %d", (*Ppunt).x, (*Ppunt).y);
    return 0;
}
```

Importante!

En la operación : $(*Ppunt).x = 3;$

Los () son necesarios porque la precedencia del operador punto es mayor que la precedencia del operador de indirección *

Por la jerarquía descripta: $*p.x$, se interpreta como $(*p.x)$, se leería como lo apuntado por $p.x$, y esto no sería correcto ya que $p.x$ no es un puntero.

Operador Flecha

- El operador flecha es usado cuando trabajamos con estructuras y punteros. Nos permitirá mas claridad en la notación.
- Si p es un puntero a la estructura, entonces la notación es:
 $p \rightarrow \text{miembroDeLaEstructura};$

```
#include <stdio.h>
#include <stdlib.h>

struct Punto { int x, y;};

int main(void) {
    setvbuf(stdout, NULL, _IONBF, 0);

    struct Punto punt, *Ppunt;

    Ppunt = &punt;

    Ppunt->x = 3;
    Ppunt->y = 5;

    printf("El punto es: %d, %d", Ppunt->x, Ppunt->y);
    return 0;
}
```

Ejercicio

Considere la siguiente información de las páginas amarillas de Tucumán: nombre de comercio, teléfono, dirección, localidad, rubro y fecha en el que comenzaron a anunciar su comercio en Páginas Amarillas.



- Escriba un módulo que le permita leer los datos del comercio. Use una función que devuelva una estructura.
- Escriba un módulo que escriba los datos del comercio. Use punteros a estructura.
- Escriba un módulo que controle si el comercio se encuentra en la localidad de Yerba Buena. El módulo test deberá retornar un valor de verdad.