LICENCIATURA EN INFORMATICA

PROGRAMADOR UNIVERSITARIO

UNIVERSIDAD NACIONAL DE TUCUMÁN
facet
FACULTAD DE CIENCIAS EXACTAS Y TECNOLOGÍA

PROGRAMACIÓN

Unidad 7 : Tipos de datos derivados- Estructuras

Lic. Mariela A. Velázquez

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

Usar campos de una estructura como argumentos de funciones

Funciones que devuelven estructuras

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.

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

```c
float promedio(float nota1, float nota2, float nota3);

int main()
{
    struct Alumno estudiante;
    float prom;

    puts("Ingrese el apellido del alumno: ");
    gets(estudiante.ape);

    puts("Ingrese el nombre del alumno:");
    gets(estudiante.nom);

    puts("Ingrese la edad del alum:");
    scanf("%d", &estudiante.edad);

    puts("Ingrese 1ra nota:");
    scanf("%f", &estudiante.nota1);

    puts("Ingrese 2da nota:");
    scanf("%f", &estudiante.nota2);

    puts("Ingrese 3ra nota:");
    scanf("%f", &estudiante.nota3);

    prom = promedio(estudiante.nota1, estudiante.nota2, estudiante.nota3);

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

    return 0;
}

float promedio(float nota1, float nota2, float nota3)
{
    return((nota1+nota2+nota3)/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

Retomamos el ejemplo anterior: Mostrar los datos cargados del alumno.

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

```c
void mostrarDatos(struct Alumno alum)
{
    puts("Los datos cargado del alumno son:");

    printf("Apellido del alumno: ");
    puts(alum.ape);

    printf("Apellido del alumno: ");
    puts(alum.nom);

    printf("Edad del alumno:  %d\n", alum.edad);

    printf("Nota 1: %d\n", alum.nota1);
    printf("Nota 2: %d\n", alum.nota2);
    printf("Nota 3: %d\n", alum.nota3);
}
```

```c
void mostrarDatos(struct Alumno alum);

int main()
{
    struct Alumno estudiante;

    puts("Ingrese el apellido del alumno: ");
    gets(estudiante.ape);

    puts("Ingrese el nombre del alumno:");
    gets(estudiante.nom);

    puts("Ingrese la edad del alum:");
    scanf("%d", &estudiante.edad);

    puts("Ingrese 1ra nota:");
    scanf("%f", &estudiante.nota1);

    puts("Ingrese 2da nota:");
    scanf("%f", &estudiante.nota2);

    puts("Ingrese 3ra nota:");
    scanf("%f", &estudiante.nota3);


    mostrarDatos(estudiante);

    return 0;
}
```

Funciones que devuelven Estructuras

Retomemos el ejemplo visto anteriormente:

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

```
struct Alumno cargaDatos();

int main()
{
    struct Alumno estudiante;

    estudiante = cargaDatos();

    //Utilizamos la funcion realizada en el ejemplo anterior
    mostrarDatos(estudiante);

    return 0;
}

struct Alumno cargaDatos()
{
   struct Alumno alum;

    puts("Ingrese el apellido del alumno: ");
    gets(alum.ape);

    puts("Ingrese el nombre del alumno:");
    gets(alum.nom);

    puts("Ingrese la edad del alum:");
    scanf("%d", &alum.edad);

    puts("Ingrese 1ra nota:");
    scanf("%f", &alum.nota1);

    puts("Ingrese 2da nota:");
    scanf("%f", &alum.nota2);

    puts("Ingrese 3ra nota:");
    scanf("%f", &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 *Ppunto;
```

Ejemplo

```c
#include <stdio.h>

struct Punto { int x, y;};

int main()
{
    struct Punto P1, *Ppunto;

    Ppunto = &P1;

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

    printf("El punto es: %d, %d", (*Ppunto).x, (*Ppunto).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 seria 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→ miembroDeLaEstructura;

```c
#include <stdio.h>

struct Punto { int x, y;};

int main()
{
    struct Punto P1, *Ppunto;

    Ppunto = &P1;

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

    printf("El punto es: %d, %d", Ppunto->x, Ppunto->y);

    return 0;
}
```

Arreglo de estructura

- Ahora necesitamos sacar el promedio de notas de un grupo de 20 alumnos.
- ¿Qué deberíamos hacer para responder a ese requisito?
- Tengo que declarar 20 variables de tipo struct?

Arreglo de estructura

- Respondemos esas preguntas desarrollando el siguiente ejemplo:

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

En este ejemplo podemos ver que trabajamos con arreglos de estructuras, haciendo uso del índice, como lo veníamos haciendo con los arreglos en general.

```c
float promedio(float nota1, float nota2, float nota3);

int main()
{
    struct Alumno estudiante[20];
    float prom;

    for (int i = 0; i < 20; i++)
    {
        printf("Datos del alumno N°: %d", i);

        puts("Ingrese el apellido del alumno : ");
        gets(estudiante[i].ape);

        puts("Ingrese el nombre del alumno:");
        gets(estudiante[i].nom);

        puts("Ingrese la edad del alum:");
        scanf("%d", &estudiante[i].edad);

        puts("Ingrese 1ra nota:");
        scanf("%f", &estudiante[i].nota1);

        puts("Ingrese 2da nota:");
        scanf("%f", &estudiante[i].nota2);

        puts("Ingrese 3ra nota:");
        scanf("%f", &estudiante[i].nota3);

        prom = promedio(estudiante[i].nota1, estudiante[i].nota2, estudiante[i].nota3);

        printf("El promedio del alumno es: %f", prom);
    }

    return 0;
}

float promedio(float nota1, float nota2, float nota3)
{
    return((nota1+nota2+nota3)/3);
}
```

Arreglo de estructuras y punteros

Cuando tenemos un puntero a un arreglo de estructura, para acceder a cada campo de la estructura lo hacemos mediante el operador flecha.

```c
int main()
{
    struct Alumno estudiante[20], *palum;

    float prom;

    palum = estudiante;

    for (int i = 0; i < 20; i++)
    {
        printf("Datos del alumno N°: %d\n", i);

        puts("Ingrese el apellido del alumno : ");
        gets(palum->ape);

        puts("Ingrese el nombre del alumno:");
        gets(palum->nom);

        puts("Ingrese la edad del alum:");
        scanf("%d", &palum->edad);

        puts("Ingrese 1ra nota:");
        scanf("%f", &palum->nota1);

        puts("Ingrese 2da nota:");
        scanf("%f", &palum->nota2);

        puts("Ingrese 3ra nota:");
        scanf("%f", &palum->nota3);

        prom = promedio(palum->nota1, palum->nota2, palum->nota3);

        printf("El promedio del alumno es: %f\n", prom);

        palum++;
    }
```

Bonus track: typedef

- Typedef permite crear identificadores para algún tipo de dato que ya existente.
- Es una facilidad proporcionada por el lenguaje C para trabajar con identificadores
- Typedef no introduce nuevos tipos, sólo formas sinónimos para tipos que se podrían mencionar de otra forma.

```c
#include <stdio.h>

int main()
{
    typedef int entero;

    entero numero=5;

    printf("El valor del número es: %d", numero);

    return 0;
}
```

Hace del nombre **entero** sea sinónimo de **int**.

```c
#include <stdio.h>

struct cd{
            char nombre[30];
            char interprete[25];
            int cant_temas;
            int duracion;
        };

typedef struct cd Discos;


int main()
{
    Discos mios;
```

Hace del nombre **Discos** sea sinónimo de **struct cd**.

```c
#include <stdio.h>

int main()
{
    printf("Hasta la próxima clase!!\n");

    return 0;
}
```