Sesión 4: Estructuras (2h)

En esta sesión se trabajará el tema 3 de la asignatura (tipos de datos estructurados) y, en concreto el tipo de dato struct o estructura. Se realizarán programas sencillos que contengan definiciones de nuevos tipos de datos (estructuras) diseñados por el usuario. Se ejercitará también sobre la manipulación de variables de este tipo realizando operaciones como acceso a un campo concreto y copia de estructuras.

El objetivo es que el alumno se familiarice con la definición de tipos nuevos de datos y sea capaz de utilizar y manipular estructuras sencillas en los programas escritos en C.

Ejercicio 1:

Dados los siguientes datos que determinan las características de una piscina:
a) Indique ¿cuál sería el tipo de dato elemental que representa mejor a cada uno?:

- `profundidad en metros`
- `largo en metros`
- `ancho en metros`
- `PH del agua`
- `volumen en metros cúbicos`
- `temperatura del agua en grados Celsius`
- `número de carriles`
- `ancho del carril en centímetros`
- `uso deportivo (‘N’ natación, ‘S’ sincronizada, ‘W’ water polo, ‘A’ saltos)`

b) Defina el tipo de dato `tpiscina` mediante una estructura que contenga como campos las características listadas en el apartado a).
c) Escriba un programa en C (en el fichero *sesion4_ej1.c*) que contenga la definición del tipo de dato `tpiscina` y la declaración de la variable `p` del tipo de dicho tipo. Además, inicialice la variable `p` de manera que cada campo tenga los siguientes valores:

- `profundidad en metros: 2.5`
- `largo en metros: 50.0`
- `ancho en metros: 21.0`
- `PH del agua: 6.5`
- `volumen en metros cúbicos: 2625.0`
- `temperatura del agua en grados Celsius: 26.5`
- `número de carriles: 8`
- `ancho del carril en centímetros: 250.0`
- `uso deportivo (‘N’ natación, ‘S’ sincronizada, ‘W’ water polo, ‘A’ saltos): N`

Agregue al programa anterior las sentencias necesarias para comprobar que los datos de `p` son coherentes. De esta forma, es necesario realizar en el orden pautado a continuación las siguientes comprobaciones: 1) el ancho de la piscina es el mismo respecto al ancho y el número de carriles; y 2) el volumen de la piscina es el mismo respecto al volumen obtenido al multiplicar los valores del largo, ancho y profundidad. El programa deberá mostrar por pantalla un mensaje indicando si las comprobaciones anteriores coinciden con los valores referidos.

En cualquiera de los casos en el que las comprobaciones no hayan sido exitosas, el programa deberá modificar los valores del ancho y/o del volumen de la piscina iniciales por los valores calculados. Si se modifica algún valor, muestre un mensaje indicando que se ha cambiado. Finalmente, el programa muestra por pantalla los valores de cada campo de la variable `p`.

Ejemplo de ejecución (en negrita los datos introducidos por el usuario):

```
Ancho de la piscina no es correcto: inicial = 21.0 calculado = 20.0

Se ha modificado el ancho de la piscina

Volumen de la piscina no es correcto: inicial = 2625.0  calculado = 2500.0

Se ha modificado el volumen de la piscina


********** Caracteristicas de la piscina **********
Profundidad = 2.5       Largo = 50.0                Ancho = 20.0
Ph = 6.5                Volumen = 2500.0            Temperatura = 26.5
Carriles = 8            Ancho del carril = 250.0    Uso deportivo = N
***************************************************************************
```

Ejercicio 2:

Dado el siguiente tipo de dato `tpixel` que permite representar las características del píxel de una imagen en formato RGBA:

```
typedef struct
{
    unsigned char R;   /* Componente rojo */
    unsigned char G;   /* Componente verde */
    unsigned char B;   /* Componente azul */
    unsigned char A;   /* Transparencia */
}tpixel;
```

Escriba un programa en C (en el fichero *sesion4_ej2.c*) que lea del teclado dos píxeles en formato RGBA utilizando como formato de entrada el mostrado en el ejemplo de ejecución, los guarde en dos variables de tipo `tpixel`, y calcule el píxel que contiene el valor medio **redondeado** de los dos píxeles leídos. Finalmente, muestre por pantalla el valor medio del pixel obtenido.

Ejemplo de ejecución (en negrita los datos introducidos por el usuario):

```
Introduzca primer pixel (R,G,B,A): (128,23,204,255)
Introduzca segundo pixel (R,G,B,A): (75,89,124,128)
El valor del pixel medio es: (102,56,164,192)
```

Observe que si calcula el valor medio entero de la componente de rojo para el ejemplo de ejecución, el resultado es `(128+75)/2 = 101`. Sin embargo, el valor de la componente de rojo del pixel promedio es `102`, porque se debe considerar el resultado real de la operación que es `101.5`, y a continuación redondearlo obteniendo `102`.

Nota: En la instrucción `scanf` el especificador de formato que permite leer desde el teclado un valor de tipo `unsigned char` es `%hhu`.

Ejercicio 3:

Dado el tipo de dato `texpr_horaria` definido como sigue:

```
typedef struct
{
  unsigned int hh;  /* Horas */
  unsigned int mm;  /* Minutos */
  unsigned int ss;  /* Segundos */
}texpr_horaria;
```

a) Escriba un programa en C (en el fichero *sesion4_ej3a.c)* que lea del teclado una expresión horaria con formato: (hh mm ss), la guarde en una variable de tipo `texpr_horaria`, convierta dicha expresión a segundos y muestre el resultado por pantalla.

Ejemplo de ejecución (en negrita los datos introducidos por el usuario):

```
Expresion horaria (hh mm ss): (13 21 4)

Segundos: 48064
```

Nota: En la instrucción `scanf` el especificador de formato que permite leer desde el teclado un valor de tipo `unsigned int` es `%u`.

b) Escriba un programa en C (en el fichero *sesion4_ej3b.c*) que lea del teclado dos expresiones horarias, las guarde en dos variables de tipo `texpr_horaria`, e indique por pantalla ¿cuál de las dos es la más cercana a la medianoche? Tenga en cuenta que puede estar cerca de la medianoche del día anterior o del mismo día.

Ejemplo de ejecución (en negrita los datos introducidos por el usuario):

```
Introduzca primera expresion horaria (hh mm ss): (8 31 4)
Introduzca segunda expresion horaria (hh mm ss): (13 15 9)

La expresion horaria mas cercana a la medianoche es: (08 31 04)
```

Nota: Para que se muestren por pantalla los valores a 2 dígitos (completando con ceros cuando corresponda), es necesario indicarlo en el formato de la instrucción `printf`. En este caso, para mostrar por pantalla el valor a dos dígitos de una variable de tipo entera sin signo, el especificador de formato es: `%02u`.

Ejercicio 4:

Dado el siguiente tipo de dato `tfraccion` que almacena la información de una fracción:

```
typedef struct
{
  int num;            /* Numerador */
  int den;            /* Denominador */
}tfraccion;
```

Escriba un programa en C (en el fichero *sesion4_ej4.c)*, que sea capaz de leer desde teclado dos fracciones y muestre por pantalla el resultado de sumarlas, restarlas, multiplicarlas y dividirlas.

Nota: No es necesario simplificar las fracciones.

Ejemplos de ejecución (en negrita los datos introducidos por el usuario):

```
Fraccion 1 (formato x/y): 1/2
Fraccion 2 (formato x/y): 7/-2

RESULTADO DE LAS OPERACIONES:

Suma: 1/2 + 7/-2 = -6/2

Resta: 1/2 - 7/-2 = 8/2

Multiplicacion: 1/2 * 7/-2 = 7/-4

Division: 1/2 * 7/-2 = -2/14
```

```
Fraccion 1 (formato x/y): 3/4
Fraccion 2 (formato x/y): -7/2

RESULTADO DE LAS OPERACIONES:

Suma: 3/4 + -7/2 = -11/4

Resta: 3/4 - -7/2 = 17/4

Multiplicacion: 3/4 * -7/2 = -21/8

Division: 3/4 * -7/2 = 6/-28
```

Ejercicio 5:

Dadas las definiciones de una serie de constantes y dado el siguiente tipo de dato `tcarta` que almacena la información de una carta de la baraja española:

```
#define BASTOS 0
#define ESPADAS 1
#define COPAS 2
#define OROS 3
typedef struct
{
  int fig;  /* Figura: 12 (Rey), 11 (Caballo), 10 (Sota), 9, ..., 1 */
  int pal;  /* Palo: OROS, COPAS, ESPADAS, BASTOS */
}tcarta;
```

a) Escriba un programa en C (en el fichero *sesion4_ej5a.c)* que lea desde teclado (como caracteres) la figura y el palo de una carta, almacene la información en una variable de tipo tcarta y finalmente, la muestre por pantalla. El formato de entrada y de salida del programa se detalla en los ejemplos de ejecución que se muestran a continuación (en negrita se muestran los datos introducidos por el usuario):

```
Figura carta: r o R, c o C, s o S, 9, ... , 1
Palo carta: o o O(OROS), c o C(COPAS), e o E(ESPADAS) o b o B(BASTOS)
Carta (figura,palo): (r,c)

(R,C)
```

```
Figura carta: r o R, c o C, s o S, 9, ... , 1
Palo carta: o o O(OROS), c o C(COPAS), e o E(ESPADAS) o b o B(BASTOS)
Carta (figura,palo): (6,O)

(6,O)
```

```
Figura carta: r o R, c o C, s o S, 9, ... , 1
Palo carta: o o O(OROS), c o C(COPAS), e o E(ESPADAS) o b o B(BASTOS)
Carta (figura,palo): (C,B)

(C,B)
```

```
Figura carta: r o R, c o C, s o S, 9, ... , 1
Palo carta: o o O(OROS), c o C(COPAS), e o E(ESPADAS) o b o
B(BASTOS)Carta (figura,palo): (1,e)

(1,E)
```

Nota: Observe que al mostrar la carta por pantalla el palo se imprime en colores, mostrando diferente color según el palo (color rojo para copas, amarillo para oros, verde para bastos y azul para espadas). Utilice la librería `colours.h` para mostrar el palo de la carta en el color que corresponde.

b) Copie el código del programa anterior en el fichero *sesion4_ej5b.c*, e incluya todas las sentencias que considere necesarias para que el programa lea del teclado una secuencia de cartas que finalizan con el carácter punto y determine cuál de las cartas introducidas es la mayor suponiendo que: OROS > COPAS > ESPADAS > BASTOS y, en caso de tener el mismo palo, 12 (Rey) > 11 (Caballo) > 10 (Sota) > 9 > ... > 1. Finalmente el programa debe mostrar por pantalla la carta.

Nota: Suponga que la secuencia de cartas no está vacía y que como mínimo tiene 1 carta.

Ejemplos de ejecución (en negrita los datos introducidos por el usuario):

```
Figura carta: r o R, c o C, s o S, 9, ... , 1
Palo carta: o o O(OROS), c o C(COPAS), e o E(ESPADAS) o b o B(BASTOS)
Introduzca cartas (fig,palo),..,(fig,palo). : (3,o),(8,o),(8,B),(R,o).

La carta mayor es: (R,O)
```

```
Figura carta: r o R, c o C, s o S, 9, ... , 1
Palo carta: o o O(OROS), c o C(COPAS), e o E(ESPADAS) o b o B(BASTOS)
Introduzca cartas (fig,palo),..,(fig,palo). : (1,c),(5,b),(c,e),(1,c),(c,b).

La carta mayor es: (1,C)
```

Ejercicio 6:

Dado el siguiente tipo de dato `testudiante`:

```
typedef struct
{
    int id;
    float nota_promedio;
    int fecha_nacimiento;          /* Formato: aaaammdd */
    int num_asignaturas_aprobadas;
}testudiante;
```

Escriba un programa en C (en el fichero *sesion4_ej6.c)* que lea desde teclado los datos de una secuencia de estudiantes que finaliza cuando el identificador de un estudiante es -1. Además, el programa debe mostrar por pantalla:

1. Los datos del estudiante con mayor número de asignaturas aprobadas. Considere que puede haber en la secuencia varios estudiantes que tengan el máximo de asignaturas aprobadas. En este caso, el programa debe mostrar los datos del último estudiante en la secuencia que tiene el mayor número de asignaturas aprobadas.
2. El número total de estudiantes que tienen nota promedio excelente (mayor a 8.5) y que tienen más de 10 asignaturas aprobadas.

Nota: Si la secuencia de estudiantes estuviese vacía, el programa deberá mostrar por pantalla un mensaje indicándolo.

Ejemplos de ejecución (en negrita los datos introducidos por el usuario):

```
Id del estudiante: 3
Nota promedio: 8.76
Fecha nacimiento (aaaammdd): 19850312
Total de asignaturas aprobadas: 6

Id del estudiante: 5
Nota promedio: 8.12
Fecha nacimiento (aaaammdd): 19841111
Total de asignaturas aprobadas: 6

Id del estudiante: -1

Estudiante con mas asignaturas aprobadas:
Id: 5
Nota promedio: 8.12
Fecha nacimiento: 19841111
Total de asignaturas aprobadas: 6

Total de estudiantes con nota > 8.5 y mas de 10 asignaturas
aprobadas: 0
```

```
Id del estudiante: -1
No se han ingresado estudiantes
```

```
Id del estudiante: 3
Nota promedio: 8.76
Fecha nacimiento (aaaammdd): 19850312
Total de asignaturas aprobadas: 11

Id del estudiante: 5
Nota promedio: 8.12
Fecha nacimiento (aaaammdd): 19841111
Total de asignaturas aprobadas: 3

Id del estudiante: -1

Estudiante con mas asignaturas aprobadas:
Id: 3
Nota promedio: 8.76
Fecha nacimiento: 19850312
Total de asignaturas aprobadas: 11

Total de estudiantes con nota > 8.5 y mas de 10 asignaturas
aprobadas: 1
```

Ejercicio 7:

a) Escriba un programa en C (en el fichero *sesion4_ej7.c*) con la definición del tipo de dato `tciudad` como una estructura con los siguientes campos:

- `Identificador numérico de ciudad`
- `Superficie en kilómetros cuadrados`
- `Número de habitantes`
- `Edad media de los habitantes`
- `Mes y año de fundación /* formato: mmaaaa */`
- `Costera 'C' o interior 'I'`

b) Añada al programa anterior el código necesario para introducir desde teclado los datos de 10 ciudades. Es necesario controlar que el mes de la fecha de fundación sea válido, es decir, no se aceptará la fecha hasta que el mes no esté comprendido entre los valores 1 y 12.
Al final del programa, se debe mostrar por pantalla la superficie media (con 2 decimales) y el número total de habitantes de todas las ciudades ingresadas. También se debe mostrar por pantalla todos los datos de las ciudades que tienen la mayor y la menor superficie.

Ejemplo de ejecución (en negrita los datos introducidos por el usuario):

```
Identificador: 1
Superficie: 3456
Numero de habitantes: 45
Edad media: 56
Fecha fundacion (mmaaaa): 152014
Mes invalido. Ingrese nuevamente la fecha de fundacion: -012014
Mes invalido. Ingrese nuevamente la fecha de fundacion: 122014
Costera o Interior: C

Identificador: 2
Superficie: 456.78
Numero de habitantes: 14
Edad media: 13
Fecha fundacion (mmaaaa): 032011
Costera o Interior: I
```

Identificador: **3**
Superficie: **1567.21**
Numero de habitantes: **34**
Edad media: **23**
Fecha fundacion (mmaaaa): **112009**
Costera o Interior: **C**

Identificador: **4**
Superficie: **7896.11**
Numero de habitantes: **22**
Edad media: **89**
Fecha fundacion (mmaaaa): **102010**
Costera o Interior: **I**

Identificador: **5**
Superficie: **112.21**
Numero de habitantes: **9**
Edad media: **31**
Fecha fundacion (mmaaaa): **092010**
Costera o Interior: **I**

Identificador: **6**
Superficie: **7778.9**
Numero de habitantes: **43**
Edad media: **102**
Fecha fundacion (mmaaaa): **011987**
Costera o Interior: **I**

Identificador: **7**
Superficie: **456.78**
Numero de habitantes: **31**
Edad media: **58**
Fecha fundacion (mmaaaa): **051996**
Costera o Interior: **C**

Identificador: **8**
Superficie: **987.23**
Numero de habitantes: **32**
Edad media: **86**
Fecha fundacion (mmaaaa): **081937**
Costera o Interior: **I**

Identificador: **9**
Superficie: **112.45**
Numero de habitantes: **17**
Edad media: **19**
Fecha fundacion (mmaaaa): **042009**
Costera o Interior: **I**

Identificador: **10**
Superficie: **456**
Numero de habitantes: **78**
Edad media: **76**
Fecha fundacion (mmaaaa): **042007**
Costera o Interior: **C**

Superficie media: 2327.97
Total habitantes: 325

```
Ciudad de mayor superficie:
Id = 4 superficie = 7896.11 poblacion = 22 edad promedio = 89.00
fundacion = 102010 tipo = I


Ciudad de menor superficie:
Id = 5 superficie = 112.21 poblacion = 9 edad promedio = 31.00
fundacion = 092010 tipo = I
```