Tecnólogo en Informática - Principios de Programación - curso 2007

Práctico 13: Estructuras

Ejercicio 1
Decir cuáles sentencias del siguiente programa son válidas:

```
struct fecha {
       int dia;
       char mes[12];
       int anio;
       char bisiesto;
};

main()
{
       struct fecha f1,f2;

       f1.dia = 30;
       f1.mes[0] = 'enero';
       f1.dia = 48;
       f1.anio = 2000;
       f1.bisiesto = si;
       f2.anio = f1.dia;
       f2.bisiesto = 'n';
       printf("%d",f1.mes);
       if(f2.anio < f1.anio)
              printf("%c",f2.mes);

       return 0;
}
```

Ejercicio 2
Decir si las siguientes afirmaciones son verdaderas o falsas:

i. Las estructuras pueden contener únicamente un tipo de datos.
ii. Las estructuras no pueden ser comparadas.
iii. Una estructura puede contener un campo que sea un arreglo.
iv. Un arreglo no puede contener datos de tipo estructura.
v. Una estructura puede ser asignada a otra estructura del mismo tipo.

Ejercicio 3
Para las siguientes definiciones de tipos:

```
struct estr1 {
       int campo1;
       char campo2,campo3;
       int campo4[10];
};
struct estr2 {
       struct estr1 camp1;
       int camp2;
};
```

y las siguientes declaraciones de variables:

```
struct estr2 e, arrE[20];
struct estr1 e1;
```

decir el tipo de las expresiones que se dan a continuación:

```
i.    arrE[5]
ii.   e[0].camp1
iii.  arrE[1].camp1.campo1
```

```
iv.    e[5].camp2
v.     arrE[0].camp1.campo4
vi.    arrE[0].camp1.campo4[8]
vii.   e1.campo4
viii.  e1.campo3
```

Ejercicio 3
Definir un tipo de datos que permita guardar información sobre pacientes de un hospital. De los pacientes, interesa conocer: su nombre (que tendrá hasta 15 letras), su cédula (de 7 dígitos), su edad (un número entero) y su fecha de nacimiento. Para la fecha de nacimiento, habrá que definir un tipo apropiado que permita manejar día, mes, año (tres valores enteros).

Ejercicio 4
Escribir un programa que lea de la entrada una palabra de largo <= 15, la guarde en la estructura **palabra** que se da a continuación y luego la despliegue.

```
#define N 15
struct palabra {
       char pal[N];
       int largo;
};
```

Los campos de la estructura se utilizan del siguiente modo:
- **pal** es un arreglo que contiene las letras de la palabra.
- **largo** es un entero que indica el largo de la palabra leída.

Ejercicio 5
Los números complejos tienen dos componentes: una parte real y una parte imaginaria, cada una de las cuales se representa mediante un número real. La siguiente estructura es útil para representar números complejos:

```
struct complejo {
       float re, im;
};
```

Escriba dos funciones con tres parámetros del tipo complejo (dos por valor y uno por referencia) que lleven a cabo la suma y la multiplicación de números complejos. Si re1 e im1 son los componentes del primer número complejo y re2 e im2 son los componentes del segundo número complejo, la suma y la multiplicación de ambos números complejos se calculan del siguiente modo:

Suma:

re3 = re1 + re2
im3 = im1 + im2

Multiplicación:

re3 = re1 * re2 - im1 * im2
im3 = im1 * re2 + im2 * re1

Ejercicio 6
Dadas las siguientes constantes y estructuras (leer utilización de **typedef** en sección 10.6 del libro):

```
#define MAX_PAL 15
#define CANT_CUR 4
typedef struct {
       char codigo;
       int calif;
} Curso;
```

```
typedef struct {
       char letras[MAX_PAL];
       int largo;
} Palabra;
typedef struct {
       Palabra nombre;
       int edad;
       Curso cursados[CANT_CUR];
} Estudiante;
```

Escribir un programa que lea de la entrada una línea con el nombre de un estudiante, su edad, y los pares código-calificación correspondientes a 4 cursos como se muestra en el ejemplo. Los datos leídos deben guardarse en las estructuras dadas, la estructura Palabra se utiliza del mismo modo que la del ejercicio 4. Luego, los datos deben desplegarse como se muestra en el ejemplo.

Ejemplo de entrada:

```
pedro/23/p-10,m-12,l-8,a-0/
```

Ejemplo de salida:

```
pedro
23
p      10
m      12
l      8
a      0
```