Unidad Nº 7: Estructuras

Ejercicios

1) Suponga que se definen las siguientes estructuras de datos para guardar la información sobre las celdas con las que tiene posibilidad de conexión un teléfono móvil:

```
1  #define SIZE 100
2  /* Información sobre la celda */
3  struct informacion_celda
4  {
5     char nombre[SIZE];              /* Nombre de la celda */
6     unsigned int identificador;     /* Número identificador */
7     float calidad_senal;            /* Calidad de la señal (entre 0 y 100) */
8     struct informacion_operador *ptr_operador; /* Puntero a una segunda estructura */
9  };
10
11 /* Información sobre el operador */
12 struct informacion_operador
13 {
14    char nombre[SIZE];          /* Cadena de texto con el nombre */
15    unsigned int prioridad;     /* Prioridad de conexión */
16    unsigned int ultima_comprob; /* Fecha de la última comprobación */
17 };
```

Responde a las siguientes preguntas:

a) ¿Qué tamaño en bytes ocupa una variable de tipo struct informacion_celda en memoria?
b) Las siguientes dos líneas declaran dos variables. ¿Cuál de ellas ocupa más espacio en memoria?

struct informacion_celda a;

struct informacion_celda *b;

c) ¿Qué tamaño tienen las siguientes variables?

struct informacion_celda *ptr1, *ptr2;

struct informacion_operador *i1, *i2;

d) Si una variable de tipo struct informacion_celda está almacenada en la posición de memoria 100, ¿qué dirección tienen cada uno de sus campos?

e) Si una variable de tipo struct informacion_celda * está almacenada en la posición de memoria 100, ¿qué dirección tiene cada uno de sus campos?

f) ¿Qué cambios debes hacer en las definiciones de la parte izquierda para que sean equivalentes a las descripciones de la parte derecha?

struct informacion_celda c; struct informacion_celda **c_ptr;	// variable de tipo estructura informacion_celda // puntero a estructura informacion_celda;

g) ¿Se pueden hacer las siguientes asignaciones? ¿Qué declara exactamente la linea 3?

1	struct informacion_celda c;
2	struct informacion_celda *c_ptr = &c;
3	struct informacion_celda d;
4	struct informacion_celda *d_ptr = c_ptr;

h) Considera la siguiente declaración y asignación:

1	struct informacion_celda c;
2	struct informacion_celda *c_ptr;
3	
4	c_ptr = *c;

¿Es correcta? Y si lo es, ¿Qué contiene la variable c_ptr (no se pregunta por lo que apunta, sino su contenido)?

i) Si se declara una variable como "struct informacion_celda c;", ¿qué tipo de datos es el que devuelve la expresión"&c.ptr_operador"?

2) Dado el siguiente código:

```
1  struct pack3
2  {
3     int a;
4  };
5  struct pack2
6  {
7     int b;
8     struct pack3 *next;
9  };
10 struct pack1
11 {
12    int c;
13    struct pack2 *next;
14 };
15
16 struct pack1 data1, *data_ptr;
17 struct pack2 data2;
18 struct pack3 data3;
19
20 data1.c = 30;
21 data2.b = 20;
22 data3.a = 10;
23 dataPtr = &data1;
```

```
24  data1.next = &data2;
25  data2.next = &data3;
```

Decide si las siguientes expresiones son correctas y en caso de que lo sean escribe a que datos se acceden.

Expresión	Correcta	Valor
data1.c		
data_ptr->c		
data_ptr.c		
data1.next->b		
data_ptr->next->b		
data_ptr.next.b		
data_ptr->next.b		
(*(data_ptr->next)).b		
data1.next->next->a		
data_ptr->next->next.a		
data_ptr->next->next->a		
data_ptr->next->a		
data_ptr->next->next->b		

3) Suponga que se escriben las siguientes declaraciones y asignaciones en un programa:

```
1  info_celda c;
```

2	info_celda_ptr c_ptr = &c;
3	info_operador op;
4	info_operador_ptr op_ptr = &op;

a) La estructura "c" contiene el campo "ptr_operador" precisamente para almacenar la información relativa al operador. ¿Qué expresión hay que usar en el código para guardar la información del operador "op" como parte de la estructura "c"? Teniendo en cuenta los valores que se asignan en las declaraciones, escribe cuatro versiones equivalentes de esta expresión (utiliza "c","c_ptr", "op" y "op_ptr").

b) Suponga ahora que la aplicación en la que se usan estas estructuras necesita almacenar la información para un máximo de 10 celdas. ¿Qué estructura de datos definirías?

4) Escriba un bucle con la variable declarada en el ejercicio anterior que asigne al campo ptr_operador el valor vacío.

5) La información sobre las celdas que se almacena en la estructura del ejercicio anterior la debe utilizar la aplicación para recordar cuál de ellas es la más próxima. Esta información puede cambiar a lo largo del tiempo. ¿Qué tipo de datos sugieres para almacenar esta información? Ofrece dos alternativas.

6) Se dispone de una estructura de tipo "info_celda c" que a su vez, en el campo "ptr_operador" tiene un puntero a una estructura "info_operador cr". ¿Qué tamaño tiene la estructura "c"? ¿Qué tamaño total ocupa la información incluyendo la información sobre el operador?

7) Escribe el cuerpo de la siguiente función:

   void fill_in(info_celda_ptr dato, unsigned int id, float sq, info_operador_ptr op_ptr)

   que asigna el valor de los parámetros "id", "sq" y "op_ptr" a los campos "identificador", "calidad_senal" y"ptr_operador" respectivamente de la estructura apuntada por el parámetro "dato".

   ¿Cómo explica que esta función asigne valores a unos campos y no devuelva resultado?

8) Considere las dos versiones del siguiente programa:

Versión 1	Versión 2

#include <stdio.h>
 	
struct package
 	
{
 	
int q;
 	
};
 	
void set_value(struct package data, int value)
 	
{
 	
data.q = value;
 	
}
 	
int main()
 	
{
 	
struct package p;
 	
p.q = 10;
 	
set_value(p, 20);
 	
printf("Value = %d\n", p.q);
 	
return 0;
 	
}
	#include <stdio.h>
	
	struct package
	
	{
	
	int q;
	
	};
	
	void set_value(struct package *d_ptr, int value)
	
	{
	
	d_ptr->q = value;
	
	}
	
	int main()
	
	{
	
	struct package p;
	
	p.q = 10;
	
	set_value(&p, 20);
	
	printf("Value = %d\n", p.q);
	
	return 0;
	
	}

La versión 1 del programa imprime el valor 10 por pantalla, y la versión 2 imprime el valor 20. Explique por qué.

9) ¿Qué cantidad de memoria ocupan estas dos estructuras? ¿Cuál es su diferencia?

info_celda t[SIZE];

cell_info *t[SIZE];

Una aplicación de gestión de fotografías en tu móvil tiene definido el catálogo de fotos de la siguiente forma:

```
#define SIZE_NAME


struct picture_info

{

   char name[SIZE_NAME];

   int date_time;

} pictures[SIZE];
```

¿Qué tamaño tiene esta estructura de datos? La aplicación necesita crear una segunda tabla del mismo número de elementos, pero en lugar de tener los datos de las fotos quiere tener los punteros a los datos de las fotos. En otras palabras, es una tabla con idéntico número de elementos que la anterior, pero sus elementos no son estructuras sino punteros a las correspondientes estructuras de la tabla pictures. Escribe la declaración y el código para rellenar esa tabla.