Tipos de Datos Abstractos

TEMAS

Julio, 2013
Guadalajara, Jalisco. México
Autor
Prof. Sabrina Lizbeth Vega Maldonado

TDA (Tipos de Datos Abstractos)

Introducción

Un TDA es un tipo de dato definido por el programador, de manera que se puede usar de forma similar a los tipos de datos primitivos, es decir no sólo es un conjunto de datos sino también las funciones que pueden aplicarse sobre él. Por lo tanto un TDA se puede definir como un conjunto de datos sobre los que se puede aplicar un conjunto de funciones.

Entonces un TDA se compone de 2 cosas : DEFINICIÓN y FUNCIONES

Definición de TDA

La definición de un tipo de dato abstracto son los elementos que corresponde a ese tipo de dato

Funciones de TDA

Las funciones de un TDA son las operaciones que se pueden realizar sobre los elementos

Definición de una estructura simple

Un TDA se puede definir con una estructura la cual se va a implementar en el lenguaje de C con la palabra reservada “struct” y en ella podemos almacenar varios datos de diferentes formatos con relación a un conjunto de información.

Su sintaxis es:

```
struct <Identificador> {
    <tipoDato> <identificadorDato-1>;
    <tipoDato> <identificadorDato-n>;
};
```

Una vez que está definida la estructura podemos acceder a esta estructura a través de una llamada como se muestra en la figura 1-a, o bien acceder a ella a través de un tipo definido como se muestra en la figura 1-b.

```
struct Revista {
    char titlulo [25];
    float precio;
    int pag;
};

struct Revista  a, b ;
```

Figura 1-a. Estructura simple

```
struct Revista {
    char titlulo [25];
    float precio;
    int pag;
} ;

typedef struct Revista RCientifica ;
RCientifica a, b ;
```

Figura 1-b. Estructura simple con su tipo definido

Definición de Variables

Se pueden definir variables en la definición de la estructura para através de estas variables acceder a los datos de la estructura. En la figura 2 podemos ver un ejemplo de esto.

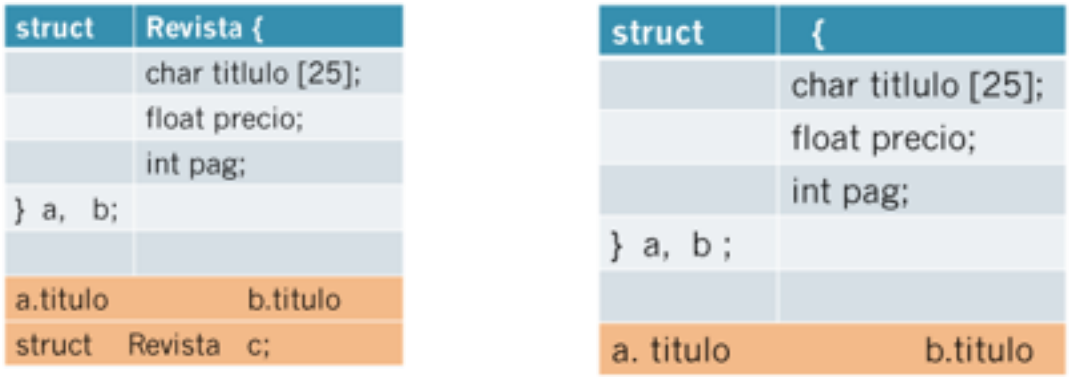

Figura 2. Definición de variables en una estructura

Definición de un Alias

Un alias se puede definir en la estructura para usar ese aleas en lugar del identificador de la misma estructura, esto puede ser muy útil al momento de acceder a la estructura. Un ejemplo de esto lo podemos ver en la figura 3.

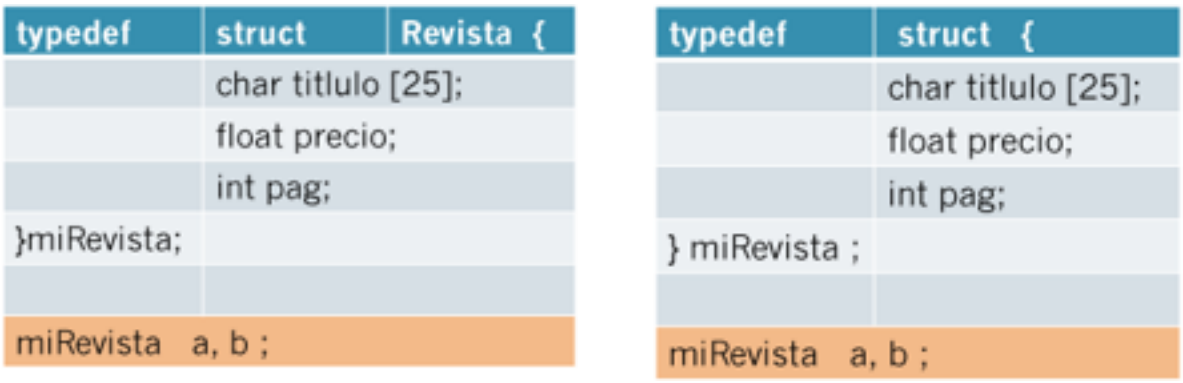

Figura 3. Definición de un alias en una estructura

Operaciones

A las operaciones que se pueden realizar en una estructura se les conoce como funciones. Estas operaciones pueden ser inicializar datos en la estructura, obtener la dirección de la estructura mediante apuntadores y asignar una estructura a otra estructura. A continuación se presentan algunos ejemplos de estas operaciones.

Inicializar

```
struct Revista a =  {"IEEE Computer", 23.54} ;
Revista b = {"C++", 35.46} ;
```

Obtener la dirección

```
*p y *q  apuntan a una Revista
struct Revista  *p  = &a;
Revista *q  = &b;
```

Asignarla a otra estructura

```
a  = b ;
```

Estructuras anidadas

Son estructuras cuyos campos son a su vez estructuras. En la figura 4-a se puede ver un ejemplo de tres estructuras simples que están relacionadas entre ellas, es decir, una persona puede tener domicilio y una fecha de nacimiento. En lugar de dejar estas tres estructuras por separado en la figura 4-b se puede observar el ejemplo de cómo anidar estas tres estrcutras de la figura 4-a.

```
typedef   struct  {
          char calle[20];
          Int numero;
          char colonia[20];
}  Domicilio;
typedef   struct  {
          char mes[20];
          int dia;
          Int anio;
}  Fecha;
typedef   Struct  {
          char nombre [20];
          Domicilio dom;
          Fecha nacimiento;
}  Persona;
```

Figura 4-a. Tres estructuras

```
typedef   struct   {
          char nombre[20];
          struct  {
                    char calle[20];
                    int numero;
                    char colonia[20];
          } dom;
          struct {
                    char mes[20];
                    int dia;
                    Int anio;
          } nacimiento;
}  Persona;
```

Figura 4-b. Una estructura anidada