Acceso a estructuras

jueves, 21 de octubre de 2021 07:21 a. m.

```
struct Etiqueta{
        <Tipo de dato> atributo1;
        …
};

typedef struct{
        <Tipo de dato> atributo1;
        …
} Etiqueta;
```

Para acceder a una estructura ya sea para escritura o lectura se utiliza el operador ".".

Sintaxis Lectura:

identificadorEstructura.atributoAlQueSeDeseaAcceder;

Sintaxis Escritura:
identificadorEstructura.atributoAlQueSeDeseaAcceder=<<valor>>;

Ejemplo:

```
#include <stdio.h>
#include <string.h>
#define TAM 40

typedef struct{
        int cantidadLlantas;
        char color[TAM];
        char marca[TAM];
        int cantidadPuertas;
        //...  más caracteristicas
} Automovil;

void main(){
        //Definición e instancia de estructura
        Automovil miCarro={4,"Rojo","Ford",3};

        puts("Datos de mi vehiculo:");
        //Acceso lectura:
        printf("Mi automovil tiene %d llantas.\n",miCarro.cantidadLlantas);
        printf("Mi automovil tiene color %s.\n",miCarro.color);
        printf("Mi automovil es de la marca %s.\n",miCarro.marca);
        printf("Mi automovil tiene %d puertas.\n",miCarro.cantidadPuertas);
        printf("La marca de mi automovil comienza con la letra '%c'.\n",*(miCarro.marca));
        //printf("La marca de mi automovil comienza con la letra '%c'.\n",miCarro.marca[0]);
        //Equivalente a la instrucción anterior.

        //Escritura en atributos de estructura:
```

```
	miCarro.cantidadLlantas=3;
	printf("Ahora mi automovil tiene %d llantas.\n",miCarro.cantidadLlantas);

	//miCarro.marca="Chevrolet"; // Semantica incorrecta
	strcpy(miCarro.marca,"Chevrolet");
	  miCarro.marca[0]='H';
	miCarro.marca[1]='o';
	miCarro.marca[2]='n';
	miCarro.marca[3]='d';
	miCarro.marca[4]='a';
	miCarro.marca[5]=0; // No hay que olvidar poner el fin de cadena si es que la palabra es de
	menor longitud que la anterior

	printf("Mi automovil es de la marca %s.\n",miCarro.marca);

	strcpy(miCarro.marca,"Ford");

	printf("Mi automovil es de la marca %s.\n",miCarro.marca);
}
```

Apuntadores a estructuras

martes, 26 de octubre de 2021 07:17 a. m.

Formas de declarar un apuntador a una estructura

Al definir una estructura

```
struct Etiqueta{
        atributo1;
        atributo2;
        atributoN;
}*  identificadorPuntero;
```

Después de haber declarado la estr uctura

```
struct Etiqueta{
        atributo1;
        atributo2;
        atributoN;
};

struct Etiqueta* identificadorPuntero;
```

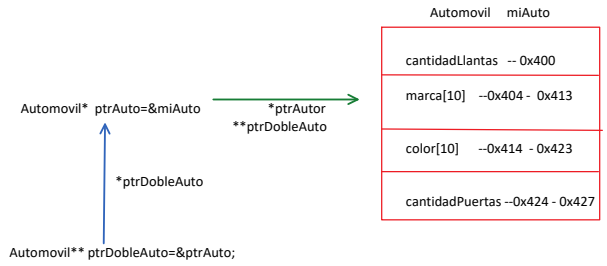
Ejemplo de acceso a estructura con apuntadores:

```
#include<stdio.h>
#define TAM 10

struct Automovil{
        int cantidadLlantas;
        char color[TAM];
        char marca[TAM];
        int cantidadPuertas;
        //...  más caracteristicas
} ;

//Ejemplo utilizando el apuntador para acceder a los datos de una estructura
void main(){
        struct Automovil miAuto={4,"Negro","Ford",5};
        struct Automovil* ptrAuto=NULL;
        ptrAuto=&miAuto;

        //Lectura de datos:
        printf("La marca de mi carro es: %s y es de color %s con %d puertas.
        \n",miAuto.marca,miAuto.color, miAuto.cantidadPuertas);
        printf(" (Utilizando apuntadores)La marca de mi carro es: %s y es de color %s con %d puertas.
        \n",(*ptrAuto).marca,(*ptrAuto).color, (*ptrAuto).cantidadPuertas);
}
```



Para manejar los apuntadores a estructuras de manera más simple existe un operador que permite sustituir la combinación de operadores de indirección (*) y el operador de acceso a estructura (.), el cual se llama operador flecha(->).

El operador flecha solo puede ser utilizado siempre y cuando la variable a la izquierda del operador sea un apuntador a estructura.

//Ejemplo con operador flecha

```
#include<stdio.h>
#define TAM 10

struct Automovil{
        int cantidadLlantas;
        char color[TAM];
        char marca[TAM];
        int cantidadPuertas;
        //...  más caracteristicas
} ;

//Ejemplo utilizando el apuntador y el operador flecha para acceder a los datos de una estructura
void main(){
        struct Automovil miAuto={4,"Negro","Ford",5};
        struct Automovil* ptrAuto=NULL;
        ptrAuto=&miAuto;

        //Incrementando la cantidad de puertas (Escritura):
        (*ptrAuto).cantidadPuertas=5;
        ptrAuto->cantidadPuertas++;  //Equivalente a la instrucción anterior.
        //Lectura de datos:
        printf("La marca de mi carro es: %s y es de color %s con %d puertas.
        \n",miAuto.marca,miAuto.color, miAuto.cantidadPuertas);
        printf(" (Utilizando apuntadores)La marca de mi carro es: %s y es de color %s con %d puertas.
        \n",ptrAuto->marca,ptrAuto->color, ptrAuto->cantidadPuertas);
}
```

//Ejemplo de manejo de arreglos dentro de una estructura

```
#include<stdio.h>
#define TAM 10

struct Automovil{
        int cantidadLlantas;
        int piezas[4];
        char color[TAM];
        char marca[TAM];
        int cantidadPuertas;
        //...  más caracteristicas
```

```
} ;

//Ejemplo utilizando el apuntador y el operador flecha para acceder a un arreglo dentro de una
estructura
void main(){
        struct Automovil miAuto={4,{0,34,54,70},"Negro","Ford",5};  //Instancia de los arreglos dentro de
        la estructura
        struct Automovil* ptrAuto=NULL;
        ptrAuto=&miAuto;

        //Modificando la posición 2 del arreglo "piezas" dentro de una estructura
        (*ptrAuto).piezas[2]=98;
        ptrAuto->piezas[2]=99;  //Equivalente a la instrucción anterior.


        //Lectura de datos:
        printf("(miAuto)El contenido del arreglo en la pos 2 es %d.\n",miAuto.piezas[2]);
        printf("(ptrAuto)El contenido del arreglo en la pos 2 es %d.\n",(*ptrAuto).piezas[2]);
        printf("(ptrAuto ->)El contenido del arreglo en la pos 2 es %d.\n",ptrAuto->piezas[2]);
}
```

//Ejemplo de manejo de arreglos de la estructura con aritmética de apuntadores:

```
#include<stdio.h>
#define TAM 10

struct Automovil{
        int cantidadLlantas;
        int piezas[4];
        char color[TAM];
        char marca[TAM];
        int cantidadPuertas;
        //...  más caracteristicas
} ;

//Ejemplo utilizando el apuntador y el operador flecha para acceder a un arreglo dentro de una
estructura
void main(){
        struct Automovil miAuto={4,{0,34,54,70},"Negro","Ford",5};  //Instancia de los arreglos dentro de
        la estructura
        struct Automovil* ptrAuto=NULL;
        ptrAuto=&miAuto;

        //Modificando la posición 2 del arreglo "piezas" dentro de una estructura
        *((*ptrAuto).piezas)=8;   // Equivalente a (*ptrAuto).piezas[0]=8
        *(ptrAuto->piezas+1)=99;  //Equivalente a ptrAuto->piezas[1]=99
        *(miAuto.piezas+2)=23;   //Equivalente a  miAuto.piezas[2]=23

        //Lectura de datos:
        printf("(miAuto)El contenido del arreglo en la pos 0 es %d.\n",miAuto.piezas[0]);
        printf("(ptrAuto)El contenido del arreglo en la pos 1 es %d.\n",(*ptrAuto).piezas[1]);
        printf("(ptrAuto ->)El contenido del arreglo en la pos 2 es %d.\n",ptrAuto->piezas[2]);
}
```

//Ejemplo utilizando apuntadores dobles a estructuras:

```
#include<stdio.h>
#define TAM 10

struct Automovil{
        int cantidadLlantas;
        int piezas[4];
        char color[TAM];
        char marca[TAM];
        int cantidadPuertas;
        //...  más caracteristicas
} ;

//Ejemplo utilizando apuntadores dobles
void main(){
        struct Automovil miAuto={4,{0,34,54,70},"Negro","Ford",5};  //Instancia de los arreglos dentro de
        la estructura
        struct Automovil* ptrAuto=NULL;
        struct Automovil** ptrDobleAuto=NULL;
        ptrAuto=&miAuto;                    //Los apuntadores a estructuras tienen que tomar la dirección de
        una variable de tipo estructura
        ptrDobleAuto=&ptrAuto; //Los apuntadores dobles tienen que tomar la dirección de otro
        apuntador

        //Modificando la cantidad de llantas dentro de la estructura con un apuntador doble
         (*(*ptrDobleAuto)).cantidadLlantas=10;

        //Lectura de datos:
        printf("(miAuto)La cantidad de llantas es %d.\n",miAuto.cantidadLlantas);
        printf("(ptrAuto)La cantidad de llantas es %d.\n",(*ptrAuto).cantidadLlantas);
        printf("(ptrAuto ->)La cantidad de llantas es es %d.\n",ptrAuto->cantidadLlantas); //Equivalente a
        la instrucción anterior

        printf("(ptrDobleAuto)La cantidad de llantas es es %d.\n",(*ptrAuto).cantidadLlantas);
        printf("(ptrDobleAuto ->)La cantidad de llantas es es %d.\n",ptrAuto->cantidadLlantas);
}
```

Uniones

miércoles, 27 de octubre de 2021 07:18 a. m.

Las uniones son estructuras pero que solo pueden albergar la información de un solo atributo.

Sintaxis:

```
union Etiqueta{
        <tipo de dato> atributo1;
        <tipo de dato> atributo2;
        <tipo de dato> atributo3;
        <tipo de dato> atributoN;
};

typedef union {
        <tipo de dato> atributo1;
        <tipo de dato> atributo2;
        <tipo de dato> atributo3;
        <tipo de dato> atributoN;
} Etiqueta;
```

Estructura (cada atributo tiene su propio espacio):

miembro1---0x300
miembro2 ---0x500
…
…
miembroN

Union

Miembro 1
Miembro 1
Miembro 1
Miembro 1
Miembro 1

```
union Data{
        int valor1;
        float valor2;
        char caracter;

} info2;
```

info2.valor1=90;

0x400	0x401	0x402	0x403
0000 0000	0000 0000	0000 0000	0010 1110

//info2.valor2=40.5;

0x400	0x401	0x402	0x403
0000 0000	0001 1000	0000 0000	1010 1111

//info2.caracter='a';

0x400	0x401	0x402	0x403
0000 0000	0001 1000	0000 0000	0011 0001

Ejemplo:

```
#include<stdio.h>
#include<string.h>

struct Planeta{                 //34Bytes
        char nombre[15];
        int diametro;
};

struct Datos{                   //9 Bytes
        int valor1;
        float valor2;
        char caracter;
};

union Data{                             //4 Bytes
        int valor1;
        float valor2;
        char caracter;
};

union PLANETA{                          //400 Bytes
        char nombre[30];
        int diametro;
};

void main(){
        struct Planeta tierra={"tierra", 400000};
        struct Datos info1={10,20.1,'a'};
        union PLANETA tierra2;
        union Data info2;

        tierra2.diametro=400000;
        strcpy(tierra2.nombre,"tierra");


        info2.valor1=78;
        //info2.caracter='a';

        printf("El tamaño de una estructura Planeta es %d.\n", sizeof(struct Planeta));
        printf("El tamaño de una estructura Datos es %d.\n", sizeof(info1));
        printf("El tamaño de una union PLANETA es %d.\n", sizeof(union PLANETA));
        printf("El tamaño de una union Data es %d.\n", sizeof(union Data));
```

```
/*      puts("Imprimiendo struct Planeta vs union PLANETA");
        printf("(struct Planeta) Nombre del planeta: %s.\n",tierra.nombre);
        printf("(struct Planeta) Diametro del planeta: %d.\n",tierra.diametro);
        printf("(union PLANETA) Nombre del planeta2: %s.\n",tierra2.nombre);
        printf("(union PLANETA) Diametro del planeta2: %d.\n",tierra2.diametro); */


        puts("Imprimiendo struct Datos vs union Data");

        printf("(struct Datos) valor1=%d.\n",info1.valor1);
        printf("(struct Datos) valor2=%f.\n",info1.valor2);
        printf("(struct Datos) caracter=%c.\n",info1.caracter);


        printf("(union Data) valor1=%d.\n",info2.valor1);
        //printf("(union Data) valor2=%f.\n",info2.valor2);
        printf("(union Data) caracter=%c.\n",info2.caracter);
}
```

Tipos de Dato Abstracto(1era parte)

miércoles, 27 de octubre de 2021 08:11 a. m.

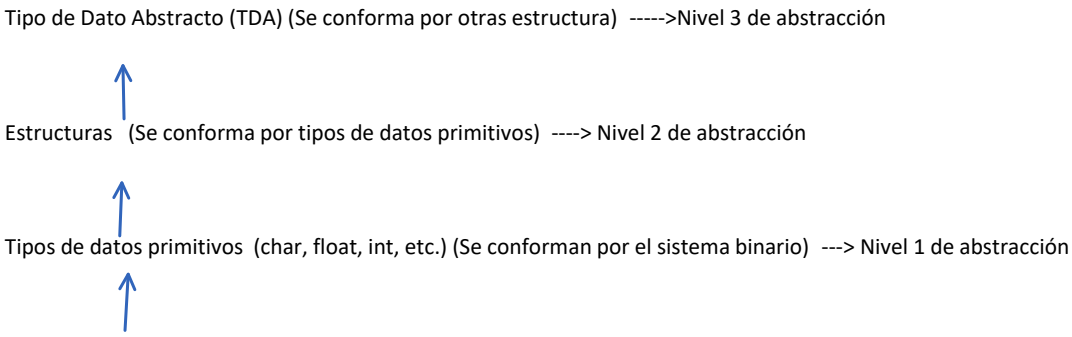
Abstracción: Es un proceso en el cual vamos a obtener las características más generales de una colección de objetos que proviene del mundo real y especificarlas en una Entidad Abstracta.

Colección de objetos: balón de soccer, pelota de ping pong, balón de futbol americano, pelota de front-ton.

Entidad: Juego, Pelota=Balon

Proceso abstracción: Pelota-->{ valorCircunferencia, material, diseño, marca, color}

Instanciar una estructura de la entidad abstracta: Pelota -> pelota de beisbol.



Ejemplo de TDA: CASA -> {Puertas, Ventanas, etc.. }

Puertas---> (int altura, flot ancho, char color[20])

Ventanas -->(int ancho, int alto, char material[40])