

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 características
} 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 estructura

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

struct Etiqueta* identificadorPuntero;
```

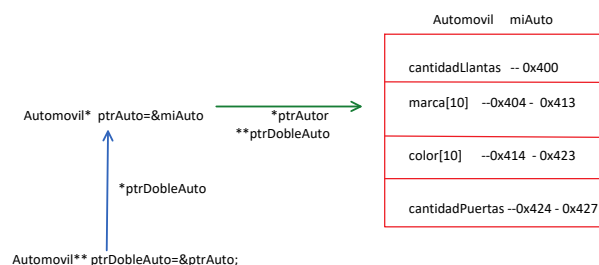
Ejemplo de acceso a estructura con apuntadores:

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

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

//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 características
};

//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 características
}
```

```

};

//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 características
};

//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 características
};

//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 %d.\n",ptrAuto->cantidadLlantas); //Equivalente a
    la instrucción anterior

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

```