

Partes clave de una estructura C.

DECLARACIÓN
DE TIPOS
PROPIOS

ESTRUCTURAS EN C

ESTRUCTURAS EN C (struct)

➤ **INTRODUCCIÓN:**

Los arrays son estructuras de datos que contienen un número determinado de elementos y todos los elementos han de ser del mismo tipo de datos; es una estructura homogénea. Esta característica supone una gran limitación cuando se requieren grupos de elementos con tipos diferentes de datos cada uno. **Por ejemplo**, si se dispone de una lista de temperaturas, es muy útil un array, sin embargo, si se necesita una lista de información de clientes que contengan elementos tales como: el nombre, edad, dirección, número de cédula, etc., los arrays no son adecuados. La solución a este problema es utilizar un tipo de dato registro, en C llamado “**estructura**”.

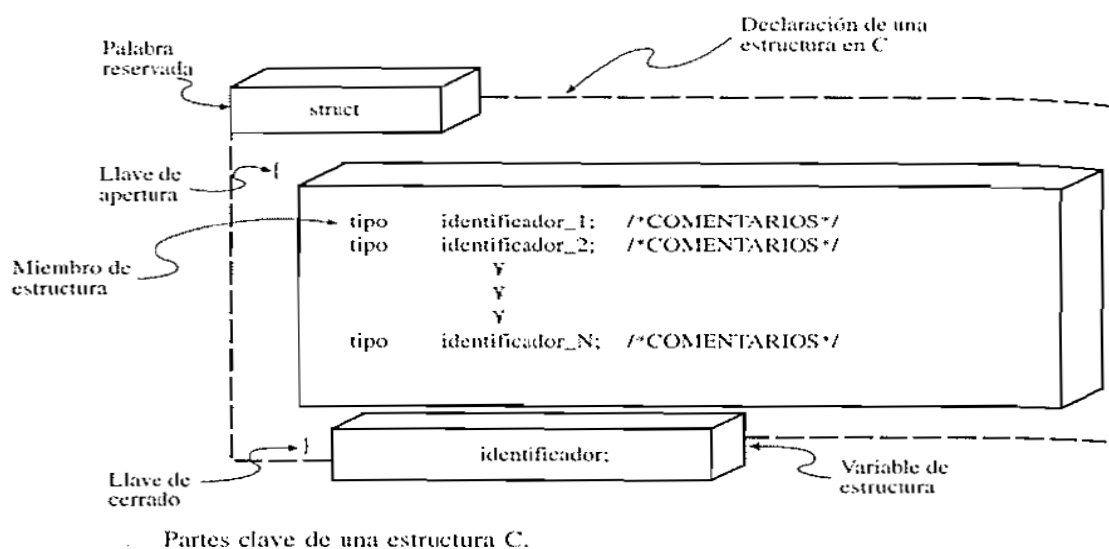
La capacidad para crear nuevos tipos es una característica importante y potente de C y libera a un programador de restringirse al uso de los tipos ofrecidos por el lenguaje. Una estructura contiene múltiples variables, que pueden ser de tipos diferentes. La estructura es importante para la creación de programas, tales como bases de datos u otras aplicaciones que requieran grandes cantidades de datos.

➤ **ESTRUCTURAS:**

Las estructuras conocidas generalmente con el nombre de **registros**, representan un tipo de datos estructurado. Se utilizan para resolver problemas que involucren tipos de datos estructurados, **heterogéneos**.

“Una estructura es una colección de elementos finita y heterogénea”.

Finita: porque se puede determinar el número de componentes y **heterogénea** porque todos los elementos pueden ser de tipos de datos diferentes. Cada componente de la estructura se denomina campo y se identifica con un nombre único.



Ejemplo: Consideremos que por cada alumno de una universidad debemos almacenar la siguiente información:

- Número de Carnet (Array de 11 caracteres).
- Nombre del alumno (Array de 60 caracteres).
- Edad del alumno (entero).
- Dirección del alumno (Array de 60 caracteres).

La estructura adecuada para almacenar esta información es la estructura. La estructura se llamará **Alumno** contiene cuatro datos miembros.

Una estructura es un tipo de datos definido por el usuario, que se debe declarar antes de que se pueda utilizar. El formato de la declaración es:

```
struct <nombre de la estructura>
{
    <tipo de dato miembro1> <nombre miembro1>
    <tipo de dato miembro2> <nombre miembro2>
    <tipo de dato miembro3> <nombre miembro3>
    .....
    <tipo de dato miembron> <nombre miembron>
};
```

Ejemplo: La declaración de la estructura **Alumno** es:

```
struct Alumno
{
    char num_carne [11];
    char nombre_alumno [60];
    int edad;
    char direccion[60];
};
```

➤ DECLARACIÓN Y DEFINICIÓN DE VARIABLES TIPO STRUCT:

A una estructura se accede utilizando una variable(s) que se deben definir después de la declaración de la estructura. En C existen dos conceptos similares a considerar, **declaración y definición**. La diferencia técnica es la siguiente, una declaración especifica simplemente el nombre y el formato de la estructura de datos, pero no reserva almacenamiento en memoria; la declaración especifica un nuevo tipo de dato: **struct <nombre-estructura>**. Por consiguiente, cada definición de variable para una estructura dada crea un área en memoria en donde los datos se almacenan de acuerdo al formato estructurado declarado.

Las variables de estructuras se pueden definir de dos formas:

1. Listándolas inmediatamente después de la llave de cierre de la declaración de la estructura.
2. Listando el tipo de la estructura creada por las variables correspondientes en cualquier lugar del programa antes de utilizarlas.

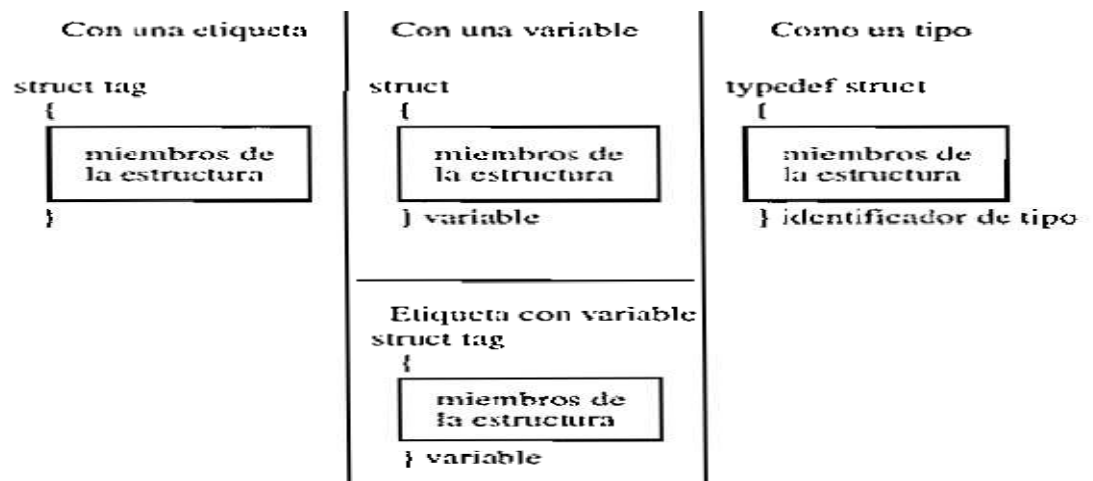


Figura 8.3. Formas de declarar estructuras.

La definición y declaración de la estructura “**Alumno**” se puede hacer por cualquiera de los dos métodos:

1. **struct Alumno**

```

{
    char num_carne [11];
    char nombre_alumno [60];
    int edad;
    char direccion[60];
}alum1,alum2,alum3;
```

2. **struct Alumno alum1,alum2,alum3;**

Otros ejemplos de definición/declaración:

Una estructura fecha podría ser:

```
struct fecha
{
    int dia; /* Elementos de la estructura */
    int mes;
    int anyo;
};
```

La definición de la estructura se puede hacer así:

1.

```
struct fecha
{
    int dia;          /* Elementos de la estructura */
    int mes;
    int anyo;
}; fech1,fech2,fech3;
```
2.

```
struct fecha fech1,fech2,fech3;
```

➤ **INICIALIZACIÓN DE UN VARIABLE DE TIPO STRUCT:**

Una estructura se puede inicializar de dos formas:

- ◆ Dentro de la sección de código de su programa.
- ◆ Como parte de la definición de la estructura, especificando los valores iniciales, entre llaves, después de la definición de variables estructura.

El formato general en este caso es:

```
struct <tipo> <nombre variable estructura >
{
    valor miembro1,
    valor miembro2,
    .....
    valor miembron
}= variable_estructura {datos de inicialización};
```

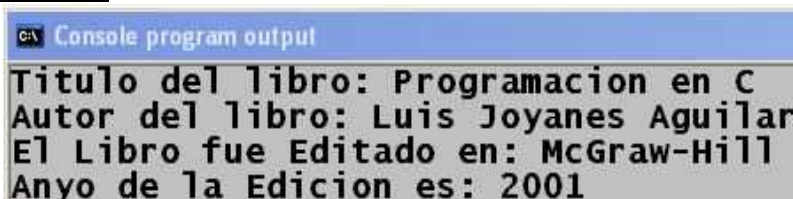
Ejemplo #1: Este programa imprime el valor de una variable de tipo struct `info_libro` que fue inicializada durante la declaración.

`//inicia_struct.c`

```
#include<stdio.h>
struct info_libro
{
    char titulo_lib [50];
    char autor [30];
    char editorial [30];
    int anyo;
} libro1 = {"Programacion en C","Luis Joyanes Aguilar", "McGraw-Hill",2001};

void main()
{
    printf("Titulo del libro: %s\n",libro1.titulo_lib);
    printf("Autor del libro: %s\n",libro1.autor);
    printf("El Libro fue Editado en: %s\n",libro1.editorial);
    printf("Anyo de la Edicion es: %d\n",libro1.anyo);
}
```

Salida:



```
C:\ Console program output
Titulo del libro: Programacion en C
Autor del libro: Luis Joyanes Aguilar
El Libro fue Editado en: McGraw-Hill
Anyo de la Edicion es: 2001
```

➤ TAMAÑO DE UNA ESTRUCTURA:

El operador **sizeof** se aplica sobre un tipo de datos, o bien sobre una variable. Se puede aplicar para determinar el tamaño que ocupa en memoria una estructura. El siguiente programa ilustra el uso del operador `sizeof` para determinar el tamaño de una estructura:

Ejemplo #2: Este programa permite visualizar el tamaño en bytes de una estructura.

```
#include <stdio.h>
struct persona
{
    char nombre [30];
    int edad;
    float altura;
    float peso;
};
```

```
void main( )
{
    struct persona per;
    printf ("El Tamanyo en Bytes de struct(persona) es: %d\n",sizeof(per)) ;
}
```

Salida:

➤ **ACCESO A ESTRUCTURAS:**

Se puede acceder a los miembros de una estructura de dos formas:

- ❖ Utilizando el operador punto (.).
- ❖ Utilizando el operador puntero ->.

➤ **ACCESO A UNA ESTRUCTURA DE DATOS MEDIANTE EL OPERADOR PUNTO:**

La asignación de datos a los miembros de una variable estructura se puede hacer mediante el operador punto.

La sintaxis C es la siguiente:

<Nombre variable estructura>.<Nombre miembro de la estructura> = datos;

Ejemplos:

strcpy(libro1.titulo_lib,"C-C++");

strcpy(libro1.autor,"Francisco Javier Ceballos");

strcpy(libro1.editorial,"RA-MA");

libro1.anyo= 2002;

El operador punto proporciona el camino directo al miembro correspondiente. Los datos que se almacenan en un miembro dado deben ser del mismo tipo que el tipo declarado para ese miembro.

➤ LECTURA Y RECUPERACIÓN DE INFORMACIÓN DE UNA ESTRUCTURA:

Se puede almacenar información en una estructura mediante inicialización, asignación directa o lectura del teclado.

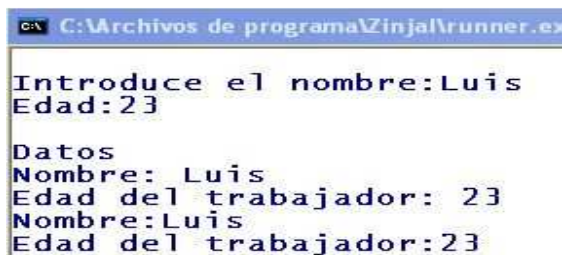
Ejemplo #3: Este programa lee del teclado los datos de una variable de tipo estructura llamada “trabajadores” y los visualiza utilizando el operando punto para acceder a los datos miembros de la estructura.

```
#include <stdio.h>
#define LONGITUD 30

struct trabajadores
{
    char nombre[LONGITUD];
    int edad;
};

int main()
{
    struct trabajadores tra_1, tra_2;
    printf ("\nIntroduce el nombre:");
    gets (tra_1.nombre);
    printf ("\nEdad:");
    scanf ("%d",&tra_1.edad);
    tra_2 = tra_1;
    printf ("LISTA DE TRABAJADORES\n");
    printf ("\nNombre: %s\n",tra_1.nombre);
    printf ("\tEdad del trabajador: %d\n",tra_1.edad);
    printf ("\tNombre:%s\n", tra_2.nombre);
    printf ("\tEdad del trabajador:%d\n",tra_2.edad);
    return 0;
}
```

Ejemplo de Salida:



```
C:\Archivos de programa\Zinjal\runner.exe
Introduce el nombre:Luis
Edad:23

Datos
Nombre: Luis
Edad del trabajador: 23
Nombre:Luis
Edad del trabajador:23
```


➤ **CREACIÓN DE SINÓNIMOS O ALIAS:**

La instrucción `typedef` permite al usuario definir alias o sinónimos, es decir, nuevos tipos de datos equivalentes a los ya existentes. El objetivo de esta instrucción consiste en utilizar nombres más apropiados y más cortos para los tipos de datos, puesto que evitamos escribir la palabra `struct` en la declaración de las variables.

En los tipos estructurados, específicamente las estructuras:

...

```
typedef struct Alumno
```

```
{
```

```
    char num_carne [11];
```

```
    char nombre_alumno [60];
```

```
    int edad;
```

```
    char direccion[60];
```

```
}Alum;
```

```
void main()
```

```
{
```

```
Alum a1,a2; /*En este caso se evita escribir la palabra struct en la declaración  
de las variables tipo Alumno*/
```

```
...
```

```
}
```

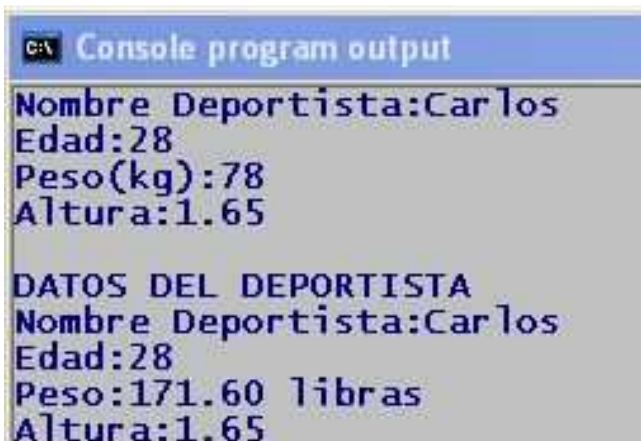
Ejemplo #4: El dueño del gimnasio “LYON GYM” desea automatizar la información de los deportistas que asisten a su gimnasio para realizar ejercicios. Los datos con que se cuenta de cada deportista es: nombre, edad, peso (kg), altura. Programa en C que permite registrar un nuevo cliente del gimnasio e imprimir dichos datos. Nota: El peso se leerá en kg y se imprimirá su valor correspondiente en libras.

```
//gimnasio.c
```

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

```
typedef struct deportista
{
    char nombre[30];
    int edad;
    float peso, altura;
}depor;

void main()
{
    depor d1;
    printf("Nombre Deportista:");
    gets(d1.nombre);
    printf("Edad:");
    scanf("%d",&d1.edad);
    printf("Peso(kg):");
    scanf("%f",&d1.peso);
    printf("Altura:");
    scanf("%f",&d1.altura);
    printf("\nDATOS DEL DEPORTISTA\n");
    printf("Nombre Deportista:%s\n",d1.nombre);
    printf("Edad:%d\n",d1.edad);
    printf("Peso:%.2f libras\n",d1.peso*2.2);
    printf("Altura:%.2f\n",d1.altura);
}
```

Ejemplo de Salida:

```
C:\ Console program output
Nombre Deportista:Carlos
Edad:28
Peso(kg):78
Altura:1.65

DATOS DEL DEPORTISTA
Nombre Deportista:Carlos
Edad:28
Peso:171.60 libras
Altura:1.65
```

Ejemplo #5: Este programa lee del teclado los datos de una variable tipo struct llamada “estudiante”, calcula e imprime el promedio de las 5 calificaciones de un estudiante.

```
//struct_estudiante.c
#include<stdio.h>
#include<string.h>

typedef struct estudiante
{
    char nombre[30];
    char num_carne[11];
    int nota[5];
}estud;

void main()
{
    estud est;
    int n, sum=0;
    float prom;
    printf("Nombre Estudiante: ");
    gets(est.nombre);
    printf("Numero de Carnet: ");
    fflush(stdin);
    gets(est.num_carne);
    printf("***Las 5 calificaciones son***\n");

    for(n=0;n<5;n++)
    {
        printf("Nota[%d]: ",n+1);
        scanf("%d",&est.nota[n]);
        sum += est.nota[n];
    }

    prom=sum/5;
    printf("\n****DATOS DEL ESTUDIANTE****\n");
    printf("Nombre:%s\n",est.nombre);
    printf("Numero de Carnet:%s\n",est.num_carne);
    printf("Promedio: %.f\n",prom);
}
```

Ejemplo de Salida:


```

C:\ Console program output
Nombre Estudiante: Camila
Numero de Carnet: 01-00492-0
***Las 5 calificaciones son***
Nota[1]: 67
Nota[2]: 89
Nota[3]: 34
Nota[4]: 56
Nota[5]: 76

****DATOS DEL ESTUDIANTE****
Nombre:Camila
Numero de Carnet:01-00492-0
Promedio: 64

```

➤ **ACCESO A UNA ESTRUCTURA DE DATOS MEDIANTE EL OPERADOR PUNTERO “->”**

El operador **puntero** -> sirve para acceder a los elementos de la estructura a partir de un puntero. Para utilizar este operador se debe definir primero una variable puntero para apuntar a la estructura. A continuación, utilice el operador puntero para apuntar a un miembro dado.

La asignación de datos a estructuras utilizando el operador -> tiene el siguiente formato:

<Puntero Estructura> -> <Nombre miembro> = datos;

Ejemplo #6: Este programa lee del teclado los datos de una variable de tipo estructura llamada “corredor” y los visualiza. Utilizando el operando puntero -> para acceder a los datos miembros de la estructura.

```

//leer_struct_2.c
#include<stdio.h>
#include<string.h>
struct corredor
{
    char nombre[30];
    char clubnom[25];
    char categoria[12];
    char sexo;
    int edad;
};

```

```

void main()
{
    struct corredor c,*ptr_c;
    ptr_c=&c;
    printf("Nombre Corredor: ");
    gets(ptr_c->nombre);
    printf("Edad: ");
    fflush(stdin);
    scanf("%d",&ptr_c->edad);
    printf("Sexo Corredor (f - m): ");
    fflush(stdin);
    scanf("%c",&ptr_c->sexo);
    printf("Nombre del Club: ");
    fflush(stdin);
    gets(ptr_c->clubnom);
    if(ptr_c->edad <= 18)
        strcpy(ptr_c->categoria,"Juvenil");
    else if(ptr_c->edad >=19 && ptr_c->edad<= 40)
        strcpy(ptr_c->categoria,"Adulto");
    else
        strcpy(ptr_c->categoria,"Veterano");
    printf("\n****DATOS DEL CORREDOR****\n");
    printf("Nombre Corredor:%s\n",ptr_c->nombre);
    printf("Edad Corredor:%d\n",ptr_c->edad);
    printf("Sexo Corredor:%c\n",ptr_c->sexo);
    printf("Categoria Corredor:%s\n",ptr_c->categoria);
    printf("Nombre Club:%s\n",ptr_c->clubnom);
}

```

Ejemplo de salida:



```

C:\ Console program output
Nombre Corredor: Luis
Edad: 34
Sexo Corredor: masculino
Nombre del Club: Los Poderosos

****DATOS DEL CORREDOR****
Nombre Corredor:Luis
Edad Corredor:34
Sexo Corredor:masculino
Nombre Club:Los
Categoria Corredor:Adulto

```

➤ **ESTRUCTURAS ANIDADAS:**

Una estructura puede contener otras estructuras llamadas **estructuras anidadas**, las cuales ahorran tiempo en la escritura de programas que utilizan estructuras similares. Se han de definir los miembros comunes sólo una vez en su propia estructura y a continuación utilizar esa estructura como miembro de otra estructura.

struct empleado

```
{
    char nombre_emp[30];
    char direccion[25];
    char ciudad[20];
    char provincia[20];
    long int cod_postal;
    double salario;
};
```

struct clientes

```
{
    char nombre_cli[30];
    char direccion[25];
    char ciudad[20];
    char provincia[20];
    long int cod_postal;
    double saldo;
};
```

Estas estructuras contienen datos diferentes aunque hay datos que están solapados. Así se podría crear una estructura `info_dir`, que contenga los miembros comunes.

struct info_dir

```
{
    char direccion[25];
    char ciudad[20];
    char provincia[20];
    long int cod_postal;
};
```

Esta estructura se puede utilizar como miembro de las otras estructuras, es decir, anidarse.

```

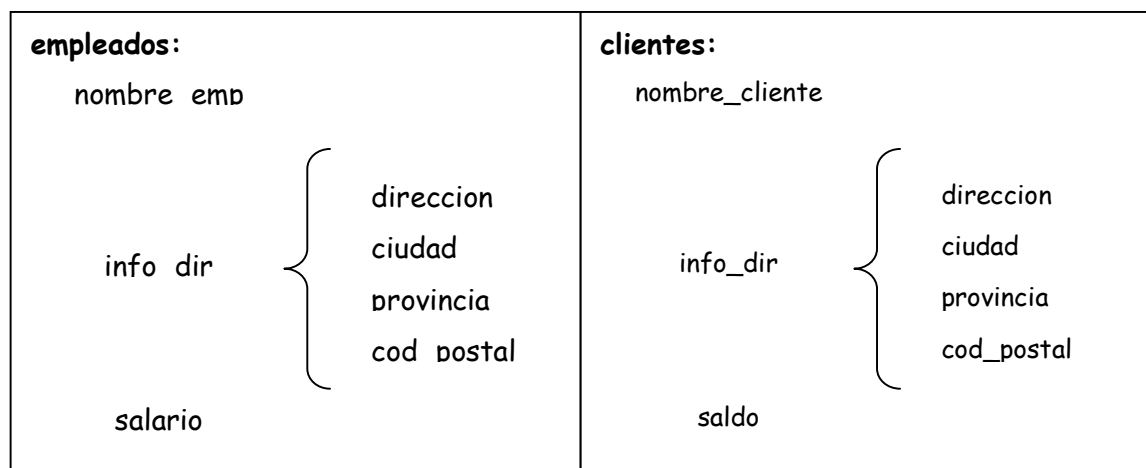
struct empleados
{
    char nombre_emp[30];
    struct info_dir domicilio_emp;
    double salario;
};

```

```

struct clientes
{
    char nombre_cli[30];
    struct info_dir domicilio_emp;
    double saldo;
};

```



Estructuras Anidadas

Ejercicio #7: Programa en C que permita almacenar la información de una agenda telefónica implementando estructuras anidadas.

```

#include <stdio.h>
struct fecha
{
    int dia, mes, anyo;
};

typedef struct agenda_personal
{
    char nombre[30];
    char telefono[10];
    struct fecha f_cumple;
}agenda;

```

```

int main(void)
{
    agenda p;
    printf("Nombre: ");
    scanf("%s",p.nombre);
    printf("Telefono: ");
    scanf("%s",p.telefono);
    fflush(stdin);
    printf("Fecha de Nacimiento(dd-mm-aaaa): ");
    scanf("%d %d %d",&p.f_cumple.dia,&p.f_cumple.mes,&p.f_cumple.anyo);
    printf("\n***Datos de Contacto***\n");
    printf("Nombre:%s\n",p.nombre);
    printf("Telefono:%s\n",p.telefono);
    printf("Fecha de Cumpleanos: %d-%d-%d\n", p.f_cumple.dia, p.f_cumple.mes,
    p.f_cumple.anyo);
    return 0;
}

```

Ejemplo de Salida:

```

Nombre: Luis
Telefono: 23113456
Fecha de Nacimiento(dd-mm-aaaa): 12 12 1976

***Datos de Contacto***
Nombre:Luis
Telefono:23113456
Fecha de Cumpleanos: 12-12-1976

```

➤ ARRAYS DE ESTRUCTURAS:

Se puede crear un array de estructuras tal como se crea un array de otros tipos. Los arrays de estructuras son idóneos para almacenar un archivo completo de empleados, un archivo de inventario o cualquier otro conjunto de datos que se adapte a un formato de estructura. Mientras que los arrays proporcionan un medio práctico de almacenar diversos valores del mismo tipo, los arrays de estructuras le permiten almacenar juntos diversos valores de diferentes tipos, agrupados como estructuras.

“Para acceder a los miembros de cada uno de los elementos estructura se utiliza una notación de array”.

Ejemplo:

```

struct info_libro libros[100];    /*Asigna un array de 100 elementos
denominados libros*/

```


Para inicializar el primer elemento de “libros”, por ejemplo, su código debe hacer referencia a los miembros de libros[0] de la forma siguiente:

```
strcpy(libros[0].titulo,"C-C++");  
strcpy(libros[0].autor,"Francisco Ceballos");  
strcpy(libros[0].titulo,"McGraw-Hill");  
libros[0].anyo=1999;
```

También puede inicializarse un array de estructuras en el punto de la declaración encerrando la lista de valores entre llaves {}.

Por ejemplo:

```
struct info_libro libros[2] = {"C++ a su alcance","Luis Joyanes", "McGraw-Hill",1999, "C","Francisco Ceballos", "RA-MA",2001, "Programación en C","Angel Hermoso", "McGraw-Hill",2000};
```

Ejercicio #8: El profesor de la asignatura de Programación Estructurada desea conocer el porcentaje de:

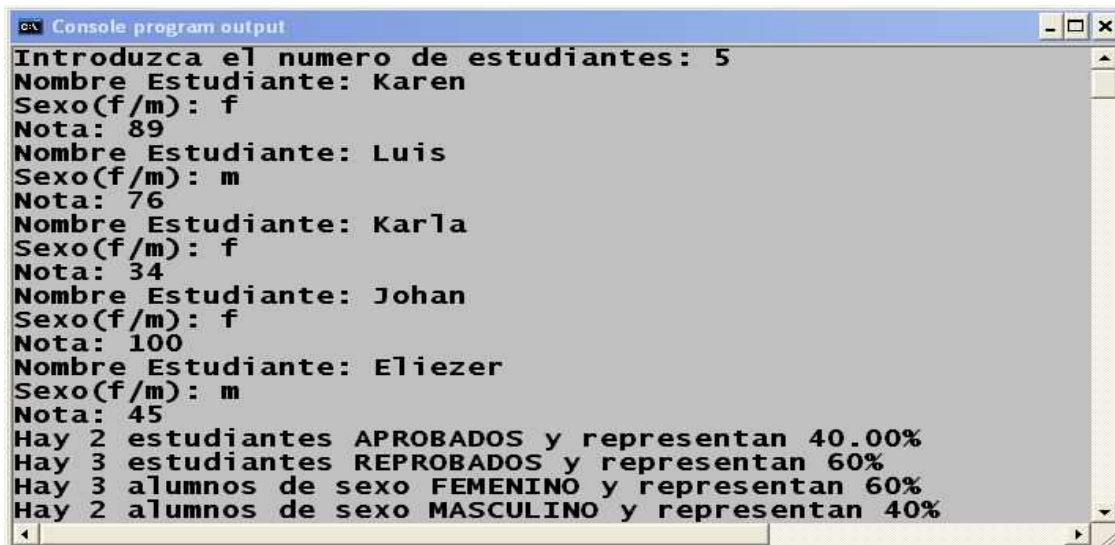
- Alumnos de Sexo Femenino y Masculino.
- Aprobados y reprobados, tomando en cuenta que la nota mínima para aprobar es 60 pts.

La información de los estudiantes (nombre, sexo, nota) está almacenada en una variable de tipo struct llamada “estudiante”. El siguiente programa crea un array de estructuras con los datos de los N estudiantes.

```
#include<stdio.h>  
#include<string.h>  
#include<ctype.h>  
struct estudiante  
{  
    char nombre[30];  
    char sexo;  
    int nota;  
};  
  
int main()  
{  
    struct estudiante est[100];  
    int n,nest,ap=0,re=0,nef=0,nem=0;  
    float pap,prep,pf,pm;  
    printf("Introduzca el numero de estudiantes: ");
```

```
scanf("%d",&nest);
for(n=1;n<=nest;n++)
{
    printf("Nombre Estudiante: ");
    fflush(stdin);
    gets(est[n].nombre);
    do
    {
        printf("Sexo(f/m): ");
        fflush(stdin);
        est[n].sexo=(toupper(getchar()));
    }while(est[n].sexo !='F' && est[n].sexo!='M');
    if(est[n].sexo=='F')
        nef++;
    if(est[n].sexo=='M')
        nem++;
    printf("Nota: ");
    scanf("%d",&est[n].nota);

    if(est[n].nota >=0 && est[n].nota<=60)
        re++;
    else
        ap++;
}
pap = (ap*100)/nest;
prep = (re*100)/nest;
pf=(nef*100)/nest;
pm=(nem*100)/nest;
printf("Hay %d estudiantes APROBADOS y representan %.2f%%\n",ap,pap);
printf("Hay %d estudiantes REPROBADOS y representan %.2f%%\n",
re,prep);
printf("Hay %d alumnos de sexo FEMENINO y representan %.2f%%\n",
nef,pf);
printf("Hay %d alumnos de sexo MASCULINO y representan %.2f%%\n",
nem,pm);
return(0);
}
```

Ejemplo de Salida:


```

C:\ Console program output
Introduzca el numero de estudiantes: 5
Nombre Estudiante: Karen
Sexo(f/m): f
Nota: 89
Nombre Estudiante: Luis
Sexo(f/m): m
Nota: 76
Nombre Estudiante: Karla
Sexo(f/m): f
Nota: 34
Nombre Estudiante: Johan
Sexo(f/m): f
Nota: 100
Nombre Estudiante: Eliezer
Sexo(f/m): m
Nota: 45
Hay 2 estudiantes APROBADOS y representan 40.00%
Hay 3 estudiantes REPROBADOS y representan 60%
Hay 3 alumnos de sexo FEMENINO y representan 60%
Hay 2 alumnos de sexo MASCULINO y representan 40%

```

Ejemplo #9: La información de todos los empleados de la empresa DATASYSTEM está almacenada en una variable de tipo struct llamada “empleado”. La información con que se cuenta para cada empleado es: nombre, sexo y sueldo.

Este programa crea un array de estructuras con los datos de los N trabajadores de la empresa e imprime cuál es el empleado con mayor y menor salario.

```

#include<stdio.h>
struct empleado
{
    char nombre[30];
    char sexo[12];
    float salario;
};

void main()
{
    struct empleado emp[100];
    int e,nemp,pmay,pmen;
    float mayor=0.0, menor=9999.0;
    printf("Introduzca el numero de empleados: ");
    scanf("%d",&nemp);
    for(e=0;e<nemp;e++)
    {
        printf("\n");
        printf("Nombre[%d]: ",e+1);
    }
}

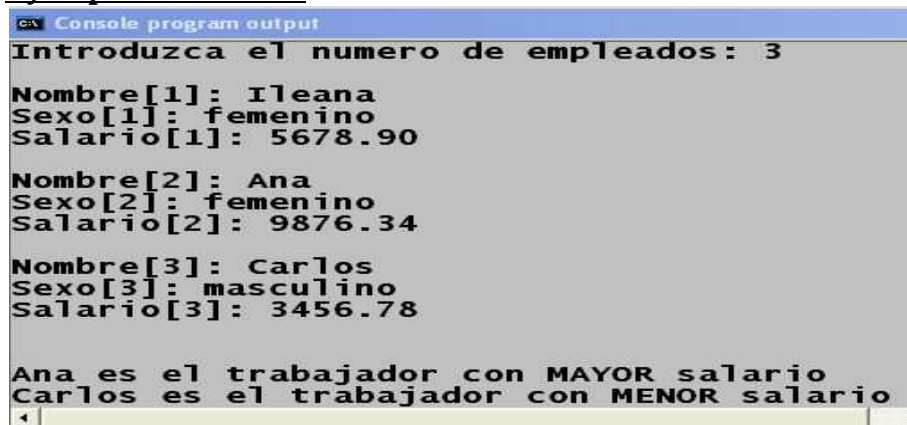
```

```

        scanf("%s",emp[e].nombre);
        printf("Sexo[%d]: ",e+1);
        scanf("%s",emp[e].sexo);
        printf("Salario[%d]: ",e+1);
        scanf("%f",&emp[e].salario);
        if(emp[e].salario>mayor)
        {
            mayor=emp[e].salario;
            pmay=e;
        }
        if(emp[e].salario<menor)
        {
            menor=emp[e].salario;
            pmen=e;
        }
    }
    printf("\n\n%s es el trabajador con MAYOR salario\n",emp[pmay].nombre);
    printf("%s es el trabajador con MENOR salario\n",emp[pmen].nombre);
}

```

Ejemplo de Salida:



```

C:\ Console program output
Introduzca el numero de empleados: 3

Nombre[1]: Ileana
Sexo[1]: femenino
Salario[1]: 5678.90

Nombre[2]: Ana
Sexo[2]: femenino
Salario[2]: 9876.34

Nombre[3]: Carlos
Sexo[3]: masculino
Salario[3]: 3456.78

Ana es el trabajador con MAYOR salario
Carlos es el trabajador con MENOR salario

```

Ejemplo #10: El Director del Área de Deporte de la UNAN-León desea automatizar la información de los deportistas de las diferentes Carreras. A partir de la declaración de las siguientes estructuras:

```

struct datos
{
    char nombre[40];
    int edad;
    char carrera[25];
};

```

```
struct deportista
```

```
{  
    struct datos pers;  
    char deporte[30];  
    int nmedallas;  
};
```

Programa en C crea un array de estructuras los datos de los N deportistas inscritos y que calcule e imprima:

- **Número promedio de medallas obtenidas.**
- **Número de estudiantes que tienen un número de medallas mayor que el promedio.**
- **Nombre y carrera del estudiante que ha obtenido mayor número de medallas.**
- **Nombre y carrera del estudiante que ha obtenido menor número de medallas.**

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

```
struct datos  
{  
    char nombre[40];  
    char carrera[25];  
};
```

```
struct deportista  
{  
    struct datos pers;  
    char deporte[30];  
    int nmedallas;  
};
```

```
void main()
```

```
{  
    struct deportista dep[100]; //Array de Estructuras  
    int d,posmay,maynmed=0,posmen,menmed=9999,nest,sum=0;  
    int mayprom=0,prom;  
  
    printf("Cuantos estudiantes hay afiliados?:");  
    scanf("%d",&nest);  
  
    for(d=0;d<nest;d++)  
    {  
        printf("\n");  
        printf("Nombre[%d]: ",d+1);  
        fflush(stdin);  
        scanf("%s",dep[d].pers.nombre);  
        printf("Carrera[%d]: ",d+1);
```

```
        fflush(stdin);
        scanf("%s",dep[d].pers.carrera);
        printf("Que deporte practica[%d]: ",d+1);
        fflush(stdin);
        scanf("%s",dep[d].deporte);
        printf("Numero de Medallas Ganadas[%d]: ",d+1);
        fflush(stdin);
        scanf("%d",&dep[d].nmedallas);
        sum+= dep[d].nmedallas;
    }

    prom= sum/nest;

    for(d=0;d<nest;d++)
    {
        //Permite encontrar el estudiante con mayor número de medallas
        if(dep[d].nmedallas>maynmed)
        {
            posmay=d;
            maynmed=dep[d].nmedallas;
        }
        if(dep[d].nmedallas<menmed)
        {
            posmen=d;
            menmed=dep[d].nmedallas;
        }
        if(dep[d].nmedallas>prom)
        {
            mayprom++;
        }
    }

    printf("\n****NUMERO PROMEDIO DE MEDALLAS ES:%d****\n",prom);
    printf("*****HAY %d ESTUDIANTES CON NUMERO DE MEDALLAS MAYORES QUE EL PROMEDIO(%d)****\n",mayprom,prom);

    printf("\n\t****ESTUDIANTE CON MAYOR NUMERO DE MEDALLAS****\n");
    printf("NOMBRE: %s\n",dep[posmay].pers.nombre);
    printf("CARRERA: %s\n",dep[posmay].pers.carrera);
    printf("NUMERO DE MEDALLAS GANADAS: %d\n",dep[posmay].nmedallas);

    printf("\n\t****ESTUDIANTE CON MENOR NUMERO DE MEDALLAS****\n");
    printf("NOMBRE: %s\n",dep[posmen].pers.nombre);
    printf("CARRERA: %s\n",dep[posmen].pers.carrera);
    printf("NUMERO DE MEDALLAS GANADAS: %d\n",dep[posmen].nmedallas);
}
```

➤ UTILIZACIÓN DE ESTRUCTURAS COMO PARÁMETROS DE UNA FUNCIÓN:

C permite pasar estructuras a funciones, bien por valor o bien por referencia utilizando el operador &. Si la estructura es grande, el tiempo necesario para copiar un parámetro struct a la pila puede ser prohibitivo. En tales casos, se debe considerar el método de pasar la dirección de la estructura.

Ejemplo #11: Programa en C que permite crear una agenda telefónica con N contactos.

```
#include <stdio.h>
struct fecha
{
    unsigned int dia, mes, anyo;
};
```

```
struct agenda_personal
{
    char nombre[30];
    char telefono[10];
    struct fecha f_cumple;
};
```

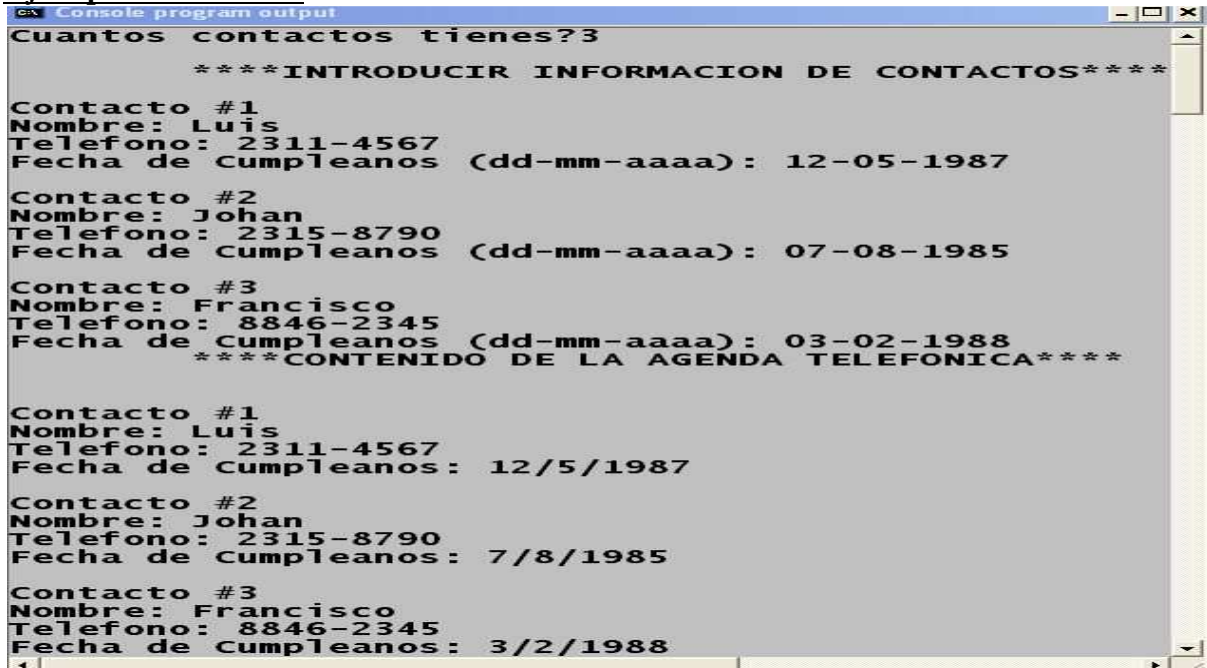
//Prototipos de Funciones

```
void LeerAgenda(struct agenda_personal pag[],int ncont);
void ImprimirAgenda(struct agenda_personal pag[],int ncont);
```

```
void main(void)
{
    struct agenda_personal ag[100];
    int npers;
    printf("Cuantos contactos tienes?");
    scanf("%d",&npers);
    printf("\n\t****INTRODUCIR INFORMACION DE CONTACTOS****\n");
    LeerAgenda(ag,npers);
    printf("\t****CONTENIDO DE LA AGENDA TELEFONICA****\n");
    ImprimirAgenda(ag,npers);
    printf("\n\n\n");
}
```

```
void LeerAgenda(struct agenda_personal agp[],int ncontactos)
{
    int c;
    for(c=0;c<ncontactos;c++)
    {
        printf("\nContacto #%d\n",c+1);
        printf("Nombre: ");
        scanf("%s",agp[c].nombre);
        printf("Telefono: ");
        scanf("%s",agp[c].telefono);
        printf("Fecha de Cumpleanos (dd-mm-aaaa): ");
        fflush(stdin);
        scanf("%d-%d-%d",&agp[c].f_cumple.dia,&agp[c].f_cumple.mes,
            &agp[c].f_cumple.anyo);
    }
}

void ImprimirAgenda(struct agenda_personal agp[], int ncontactos)
{
    int c;
    for(c=0;c<ncontactos;c++)
    {
        printf("\n\nContacto #%d\n",c+1);
        printf("Nombre: ");
        printf("%s\n",agp[c].nombre);
        printf("Telefono: ");
        printf("%s\n",agp[c].telefono);
        printf("Fecha de Cumpleanos: ");
        printf("%d/%d/%d",agp[c].f_cumple.dia,agp[c].f_cumple.mes,
            agp[c].f_cumple.anyo);
    }
}
```


Ejemplo de Salida:


```

Console program output
Cuantos contactos tienes?3

****INTRODUCIR INFORMACION DE CONTACTOS****

Contacto #1
Nombre: Luis
Telefono: 2311-4567
Fecha de Cumpleanos (dd-mm-aaaa): 12-05-1987

Contacto #2
Nombre: Johan
Telefono: 2315-8790
Fecha de Cumpleanos (dd-mm-aaaa): 07-08-1985

Contacto #3
Nombre: Francisco
Telefono: 8846-2345
Fecha de Cumpleanos (dd-mm-aaaa): 03-02-1988
****CONTENIDO DE LA AGENDA TELEFONICA****

Contacto #1
Nombre: Luis
Telefono: 2311-4567
Fecha de Cumpleanos: 12/5/1987

Contacto #2
Nombre: Johan
Telefono: 2315-8790
Fecha de Cumpleanos: 7/8/1985

Contacto #3
Nombre: Francisco
Telefono: 8846-2345
Fecha de Cumpleanos: 3/2/1988

```

➤ EJERCICIOS PROPUESTOS:

Ejercicio 1: Escribe la declaración de un tipo de estructura que se llamará `t_empleado`, con la siguiente información:

- nombre: Cadena de caracteres.
- antigüedad: Dato entero.
- ventas: Un array de 3 elementos decimales que contiene el monto de las 3 ventas realizadas por cada vendedor.

Realice un programa en C que lea en un array de estructuras la información de N vendedores e imprima la siguiente información.

- Promedio de ventas de cada vendedor.
- Nombre del vendedor con mayor monto de ventas.
- Nombre del vendedor con mayor antigüedad en la empresa.

Ejercicio 2: En el Hospital “La Fraternidad” almacenan la siguiente información de sus pacientes:

- Nombre y apellidos: Cadena de caracteres.
- Edad: Entero.
- Sexo: Carácter.
- Condición: Entero (Se refiere al estado de salud en que ingresa el paciente. Los valores que toma van de 1...5, y representan el máximo grado de gravedad).
- Teléfono.

Escribir un programa en C que genere lo siguiente:

El porcentaje de hombres y mujeres registrados en el hospital.

El número de paciente de cada una de las categorías de condición.

El nombre y teléfono de todos los pacientes que tuvieron una condición de ingreso de máxima gravedad (5).

Ejercicio 3: Una empresa de bienes raíces de León, lleva información sobre las propiedades que tiene disponibles tanto para venta como para renta.

- **Clave de la propiedad (cadena de caracteres).**
- **Superficie cubierta (decimal).**
- **Superficie del terreno (decimal).**
- **Características (cadena de caracteres)**
- **Ubicación geográfica (estructura)**
 - **Barrio (cadena de caracteres)**
 - **Dirección (cadena de caracteres)**
- **Precio (decimal).**
- **Disponibilidad (carácter).**

Diseñe un programa en C que realice lo siguiente:

- **Listado de las propiedades disponibles para venta en el Barrio de Subtiava cuyo valor oscile entre C\$ 50000 y C\$ 100000.**
- **Implemente una función que al recibir el nombre de un determinado barrio y un cierto rango respecto al monto, muestre un listado de todas las propiedades disponibles para renta.**

Ejemplo de Prototipo de Funciones:

- **void lectura_bienes(struct bienes_raices[], int npropiedades);**
- **void propiedades_disponibles(struct bienes_raices[],int npropiedades);**
- **void propiedades_renta(struct bienes_raices[], int npropiedades);**

➤ **BIBLIOGRAFÍA BÁSICA:**

- ◆ Ceballos, Francisco Javier: **C/C++ Curso de Programación**, 2da Edición. RA-MA, 2002.
- ◆ Cairó, Osvaldo. Fundamentos de Programación. Piensa en C. 1era Edición. PEARSON EDUCACIÓN, 2006.
- ◆ Gottfried, Byron S: **Programación en C**. McGraw Hill, 1991.
- ◆ Joyanes Aguilar, Luis; Zahonero Martínez Ignacio: **Programación en C**. McGraw Hill, 2001.