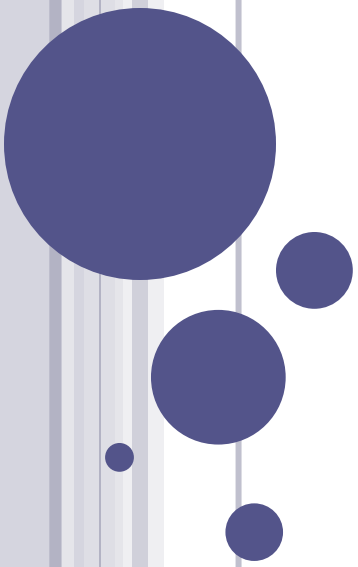
Estructuras (Registros)



Estructuras

- Al momento de resolver un problema y codificar un programa a menudo se requiere agrupar y manipular datos de distinto tipo (estructurarlos para operar sobre ellos) de una manera sencilla y eficiente.

- Un ejemplo muy citado es en el caso de una factura, donde es necesario tener los datos del cliente que realizó la compra, de los productos o artículos que éste compró y probablemente quizás del empleado que realizó la venta. Hasta lo que se sabe para resolver este tipo de problemas se tendrían que declarar demasiadas variables independientes lo cual se vuelve muy poco recomendable.

Definición de una Estructura

- Un registro o estructura es un conjunto de n *elementos heterogéneos* que están agrupados bajo un único nombre (en una sola variable).

  - Heterogéneo: los elementos son por lo general de distinto tipo de dato.
  - Es un nuevo tipo de dato definido por el programador.
  - A cada uno de los elementos de una estructura se le conoce con el nombre de campo o miembro.

Definición de una Estructura

- Así se puede decir que una estructura o registro es una agrupación de datos relacionados a una misma entidad, entendiéndose por entidad cualquier sujeto o cosa.

- Las estructuras son equiparadas por lo tanto con los registros que hay en un archivo o en una base de datos. Incluso se les suele usar a estas para extraer y almacenar información en ellos.

- Una estructura al igual que una variable, puede ser global o local:

  - es global cuando es declarada fuera de toda función, y,
  - es local cuando esta definida dentro de alguna.

Declaración de Registros o Estructuras

- En lenguaje C, existen tres formas de declararlos:
  - Primera forma: se declara la estructura y al mismo tiempo se define(n) la(s) variable(s) (alias) de tipo la estructura.
  - Segunda forma: se declara primero la estructura y después se define(n) la variable(s) (alias) de tipo la estructura.
  - Tercera forma (recomendada): se hace uso del operador *typedef*, así se declara un *nuevo tipo de dato* definido por el programador, el cual se puede usar como cualquier otro tipo de dato predefinido en C al declarar la(s) variable(s) (alias) de tipo la estructura.

Declaración de Registros o Estructuras

- Sintaxis para la primera forma:

```
struct <nombre_de_la_estructura>
{
    <tipo> <nombre_del_campo1>;
    <tipo> <nombre_del_campo2>;
    ....
    <tipo> <nombre_del_campoN>;
} <variable_de_tipo_la_estructura>;
```

Declaración de Registros o Estructuras

- Ejemplos para la primera forma:

```
struct Ave
{
    char especie[20 + 1];
    char colorPlumaje[10 + 1];
    float alturaVuelo;
    char habitat[80 + 1];
} aguila;
```

```
struct Persona
{
    char nombre[50 + 1];
    int edad;
    char sexo[9 + 1];
    int peso;
    int altura;
} juan, rosa, pedro;
```

Declaración de Registros o Estructuras

- Sintaxis para la segunda forma:

```
struct <nombre_de_la_estructura>
{
   <tipo> <nombre_del_campo1>;
   <tipo> <nombre_del_campo2>;
   ….
   <tipo> <nombre_del_campoN>;
};
```

Variable(s) de tipo la estructura se declara(n) como:

```
struct <nombre_de_la_estructura> <variable1>, <variable2>, …, <variableN>
```

- **Ventaja:** pueden definirse otras variables del mismo tipo sin tener que repetir la estructura.

Declaración de Registros o Estructuras

- Ejemplos para la segunda forma:

```
struct Ave
{
    char especie[20 + 1];
    char colorPlumaje[10 + 1];
    float alturaVuelo;
    char habitat[80 + 1];
};

struct Ave aguila, halcon;
```

```
struct Persona
{
    char nombre[50 + 1];
    int edad;
    char sexo[9 + 1];
    int peso;
    int altura;
};

struct Persona rosa, pedro;
```

Declaración de Registros o Estructuras

- Sintaxis para la tercera forma:

```
typedef struct [<nombre_de_la_estructura>]
{
   <tipo> <nombre_del_campo1>;
   <tipo> <nombre_del_campo2>;
   ….
   <tipo> <nombre_del_campoN>;
} <nombre_de_tipo_la_estructura>;
```

Variable(s) de tipo la estructura se declara(n) como:

<nombre_de_tipo_la_estructura> <variable1>, <variable2>, …, <variableN>

- **Ventaja:** independencia de la declaración y definición del tipo de dato y la declaración de variables de ese tipo.

Declaración de Registros o Estructuras

- Ejemplos para la tercera forma:

```
typedef struct
{
    char especie[20 + 1];
    char colorPlumaje[10 + 1];
    float alturaVuelo;
    char habitat[80 + 1];
} Ave;

Ave aguila, halcon;
```

```
typedef struct Persona
{
    char nombre[50 + 1];
    int edad;
    char sexo[9 + 1];
    int peso;
    int altura;
} Individuo;

Individuo juan, rosa, menores[10];
```

Declaración de Registros o Estructuras

- De esta manera, los componentes de toda estructura son sus datos y el nombre que la identifica, con lo cual, la estructura es manipulada por un alias (variable) dentro del programa, así, a través del alias es que se permite el acceso a los miembros (campos) de la estructura.

Acceso a los elementos de un Registro o Estructura

- Hay dos formas de acceso a los miembros de una estructura:
  - Mediante la variable (alias) de tipo la estructura y el operador de acceso punto (.):

    <variable_de_tipo_la_estructura>.<nombre_del_campo>

  - Mediante un apuntador a la variable (alias) de tipo la estructura y el operador de acceso apuntador a la variable (->):

    <apuntador_a_la_variable>-><nombre_del_campo>

  - Los elementos (campos o miembros) de una estructura son variables de un tipo determinado y se pueden tratar como tal.

Acceso a los elementos de un Registro o Estructura

```
typedef struct Persona
{
    char nombre[50 + 1];
    int edad;
    char sexo[9 + 1];
    int peso;
    int altura;
} Individuo;

Individuo juan, *rosa;
//leer los datos de juan y rosa
….
//mostrar sus datos
printf(“%s\n”,juan.nombre);
fprintf(stdout,“%d\n”,juan.edad);
….
fprintf(stdout,“%s\n”,rosa->nombre);
printf(“%d\n”,rosa->edad);
….
```

Asignación de valores a los miembros de un Registro o Estructura

- El procesamiento de los elementos o miembros (campos) de una estructura sólo se puede hacer de manera individual (campo a campo).

- A una estructura se le pueden asignar valores de varias maneras:

  - Asignando inicialmente los valores (inicialización):

Asignación de valores a los miembros de un Registro o Estructura

```
typedef struct Persona
{
    char nombre[50 + 1];
    int edad;
    char sexo[9 + 1];
    int peso;
    int altura;
} Individuo;

Individuo juan = {“Juan Pérez López”, 35, “Masculino”, 80, 183}
```

- Asignando valores con sentencias de asignación a sus elementos:

Asignación de valores a los miembros de un Registro o Estructura

```
Individuo juan, juanito;

strcpy(juan.nombre,“Juan Pérez López”);
juan.edad = 35;
strcpy(juan.sexo,“Masculino”);
juan.peso = 80;
juan.altura = 183;
....
juanito.sexo = juan.sexo;
….
```