ESTRUCTURAS DE DATOS

ESTRUCTURA DE DATOS

Una estructura es un conjunto de variables que se referencian con un único nombre.

ESTRUCTURA DE DATOS

```
// Para definir una estructura se utiliza la palabra clave struct.

struct nombre_estructura{
  tipo variable1;
  tipo variable2;
  ...
  ...
  tipo variableI [=expresión]
  ...
  ...
  tipo variableN;
};
```

ESTRUCTURA DE DATOS

```
// Ejemplo de definición de estructura Persona

struct persona{
  char nombre[20];
  char apellido_paterno[20];
  char apellido_materno[20];
  int edad;
};
```

ESTRUCTURA DE DATOS

```
// Ejemplo de declaración de estructura Persona

struct persona{
  char nombre[20];
  char apellido_paterno[20];
  char apellido_materno[20];
  int edad;
};

struct persona persona1;
struct persona persona2;
struct persona persona3;

int x;
int y;
int z;
```

ESTRUCTURA DE DATOS

```c
// Ejemplo de declaración de estructura Persona

struct persona {
  char nombre[20];
  char apellido_paterno[20];
  char apellido_materno[20];
  int edad;
} persona1, persona2, persona3;

int x, y, z;
```

ESTRUCTURA

```
// Ejemplo de inicialización de estructura Persona

struct persona{
  char nombre[20];
  char apellido_paterno[20];
  char apellido_materno[20];
  int edad;
};

struct persona persona1 = {"Pedro", "Marquez", "Lopez", 18};

int x = 1;
```

ESTRUCTURA

```
// Ejemplo de inicialización de estructura Persona

struct persona{
  char nombre[20];
  char apellido_paterno[20];
  char apellido_materno[20];
  int edad;
} persona1 = {"Pedro", "Marquez", "Lopez", 18};

int x = 1;
```

ESTRUCTURA

```c
// Ejemplo de inicialización de estructura Persona

struct persona{
  char nombre[20];
  char apellido_paterno[20];
  char apellido_materno[20];
  int edad;
} persona1;

persona1.nombre = "carlos";
persona1.apellido_paterno = "perez";
persona1.apellido_materno = "mata";
persona1.edad = 17;

printf("nombre = %s \n", persona1.nombre);
printf("apellido paterno = %s \n", persona1.apellido_paterno);
printf("apellido materno = %s \n", persona1.apellido_materno);
printf("edad = %d \n", persona1.edad);
```

ESTRUCTURA

```c
// Ejemplo de leer desde el teclado y almacenar en estructura Persona

struct persona{
  char nombre[20];
  char apellido_paterno[20];
  char apellido_materno[20];
  int edad;
} persona1;

printf("Proporcione su nombre \n");
scanf("%i", &persona1.edad );

printf("Proporcione su edad \n");
scanf("%i", &persona1.edad );

printf("nombre = %s \n", persona1.nombre);
printf("edad = %d \n", persona1.edad);
```

ESTRUCTURA

```
// Ejemplo de estructura Coordenada

struct coordenada{
  int latitud;
  int longitud;
  int altitud;
};

struct coordenada gps;

gps.latitud = 1;
gps.longitud = 2;
gps.altitud = 2;

printf("Coordenada en latitud= %d \n", gps.latitud);
printf("Coordenada en longitud= %d \n", gps.longitud);
printf("Coordenada en altitud= %d \n", gps.altitud);
```

EJERCICIOS

```
Definir las siguientes estructuras:

1.- Direccion: calle, numero, colonia, ciudad
2.- Fecha: dia, mes, año
3.- Hora: hora, minuto, segundo
4.- Cliente: nombre, apellido_paterno, apellido_materno
5.- Factura: direccion, fecha, hora, cliente, numero de factura
```

EJERCICIOS

```
struct Direccion{
  char calle[20];
  int numero;
  char colonia[20];
  char ciudad[20];
};

struct Fecha{
  int dia;
  int mes;
  int anio;
};

struct Hora{
  int horas;
  int minutos;
  int segundos;
};
```

EJERCICIOS

```
struct Cliente{
  char nombre[20];
  char apellido_materno[20];
  char apellido_paterno[20];
};

struct Factura{
  struct Direccion direccion;
  struct Cliente cliente;
  struct Fecha fecha;
  struct Hora hora;
  int numero_factura;
};
```

EJERCICIOS

Dadas las estructuras definidas en la clase anterio realizar el siguiente programa:

Desarrollar una aplicación que permita las siguientes operaciones sobre las estructuras de datos definidas anteriormente:

- Capturar nueva factura
- Editar una factura existente
- Eliminar una factura existente
- Mostrar lista de facturas (solo mostrar numero de factura)
- Mostrar una factura en particular

Cada operacion debe ser realizada utilizando funciones

EJERCICIOS

```c
#include <stdio.h>

struct Cliente{
        char nombre[50] = "Este sera el nombre del cliente";
        char apellido_materno[20];
        char apellido_paterno[20];
};

main(){
  struct Cliente cliente;
        cliente.nombre = "Adrian Alarcon";
}
```

EJERCICIOS

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

struct Cliente{
  char nombre[50] = "Este sera el nombre del cliente";
  char apellido_materno[20];
  char apellido_paterno[20];
};

main(){
  struct Cliente cliente;
  strncpy(cliente.nombre, "Adrian Alarcon", 50);

  gets (cliente.nombre);
  fflush(stdin);
}
```

UNIÓN

Las uniones son similares a las estructuras en cuanto que agrupan a una serie de miembros, pero la forma de almacenamiento es diferente y por consiguiente, efectos diferentes.

Una estructura (struct) permite almacenar elementos relacionados juntos y almacenados en posiciones contiguaes de memoria; las uniones (union) almacenan tambien miembros multiples en un paquete, sin embargo en lugar de situar sus miembros unos detras de otros, en una union, todos los miembros se solapan entre si en la misma posicion.

El tamaño ocupado por una unión es el tamaño del mayor elemento de la misma.

UNIÓN

```
union NombreUnion{
  TipoVar1 NombreVar1 ;
        TipoVar2 NombreVar2 ;
  TipoVar3 NombreVar3 ;
};

 union NombreUnion Varia1Union;
```

EJERCICIOS

```
union PruebaUnion{
  float flotante;
  int entero;
};

union PruebaUnion pu;

pu.flotante = 2.1;
...
pu.entero = 5;
...
```

ESTRUCTURAS Y APUNTADORES

Se puede crear un apuntador a una estructura, y para acceder a sus elementos se debe utilizar el operador ->

```
struct Cliente{
  char nombre[20];
  char apellido_materno[20];
  char apellido_paterno[20];
};

struct Cliente cliente;
strncpy(cliente.nombre, "Carlos", 20);
printf(" %s ", clientep.nombre);

struct Cliente *clientep;
strncpy(clientep->nombre, "Pedro", 20);
printf(" %s ", clientep->nombre);
```

EJERCICIOS

Escribir un programa que realice los siguientes puntos:

1. Declare un tipo de dato para representar estaciones del año.
2. Escribe una función que devuelva la estacion del año que ha leido del teclado. La función debe de ser del tipo declarado en el ejercicio 1.
3. Declare un tipo de dato estructura para representar un alumno; los campos que tiene que tener son:
   nombre, materia, edad, domicilio y notas de 10 asignaturas.
   Escribe el codigo necesario para leer todos los campos de una variable tipo estructura alumno.
4. Declara un tipo de estructura fecha,
   escribe una funcion que reciba dos fechas y devuelva
   la cantidad de dias entre ellos.
5. Defina una estructura para representar clientes,
   un cliente tiene nombre, saldo y estado (0-puntal,1- moroso).
   Escriba un programa que pemita tener un listado de al menos
   10 clientes, permita capturarlos, mostrarlos todos,
   mostrar los morosos, mostrar los puntuales.
   Cree funciones para cada opcion del menu.