

# Estructuras y uniones

# Definición de estructura

Una estructura (también llamada registro) es un tipo de datos que agrupa varios datos de tipo simple en un solo objeto.

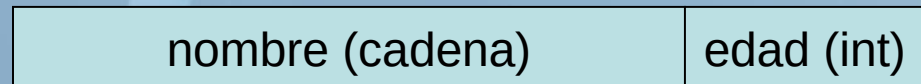
Las estructuras se declaran con la palabra reservada `struct`.

```
struct nombre{  
    campos;  
};
```

Cada campo esta formado por la declaración de una o más variables de algún otro tipo. Ejemplo:

```
struct persona{  
    char nombre[30];  
    int edad;  
};
```

persona



Campos

# Variables de tipo estructura

Una vez definida la estructura se pueden declarar variables de ese tipo:

```
struct persona{  
    char nombre[30];  
    int edad;  
};  
persona juan, maria; o  
struct persona juan, maria;
```

TAMBIEN

```
struct persona{  
    char nombre[30];  
    int edad;  
}juan, maria;
```

La asignación se hace mediante el operador “.”

```
gets(juan.nombre);  
juan.edad = 23;  
gets(maria.nombre);  
maria.edad = 17;
```

# Ejemplo

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

struct persona{
    char nombre[30];
    int edad;
};

int main(){
    struct persona per1,per2;
    strcpy(per1.nombre,"pedro");
    strcpy(per2.nombre,"maria");
    per1.edad = 23;
    per2.edad = 17;
    printf("nombre: %s\nedad: %d\n",per1.nombre,per1.edad);
    printf("nombre: %s\nedad: %d\n",per2.nombre,per2.edad);

    return 0;
}
```

# Otros ejemplos

Carta de baraja

numero	palo
--------	------

```
struct carta{  
    int numero;  
    char palo;  
}
```

Libro

titulo	autor	editorial	anyo
--------	-------	-----------	------

```
struct libro{  
    char titulo[40];  
    char autor[30];  
    char editorial[20];  
    int anyo;  
}
```

Polígono regular

nombre	lados	longitudLado
--------	-------	--------------

area
------

```
struct poligono{  
    char nombre[20];  
    int lados;  
    float longitudLado, area;  
}
```

```
struct carta c1;  
c1.numero = 3;  
c1.palo = 'D';
```

```
struct libro L1;  
strcpy(L1.titulo, "Piensa en C");  
strcpy(L1.autor, "Osvaldo Cairó");  
strcpy(L1.editorial, "Pearson");  
L1.anyo = 2006;
```

```
struct poligono p;  
strcpy(p.nombre, "hexagono");  
p.lados = 6;  
p.longitudLado = 5;  
p.area = p.lados * p.longitudLado * p.longitudLado / 2.0 /  
tan(PI / p.lados);
```

# Ejemplo de alumnos

```
struct alumno{  
    int matricula;  
    char nombre[20];  
    char carrera[20];  
    float promedio;  
    char direccion[20];  
};
```

alumno				
matricula	nombre	carrera	promedio	direccion

```
int main(){
    struct alumno a1={120,"Maria","Contabilidad",8.9,"Queretaro"},a2;
    printf("\nIngresa la matricula del alumno 2:");
    scanf("%d" , &a2.matricula);
    getc(stdin); // desecha el enter ingresado
    printf("\nIngresa el nombre del alumno 2:");
    fgets(a2.nombre, 20, stdin);
    printf("\nIngresa la carrera del alumno 2:");
    fgets(a2.carrera, 20, stdin);
    printf("\nIngresa el promedio del alumno 2:");
    scanf("%f",&a2.promedio);
    getc(stdin); // desecha el enter ingresado
    printf("\nIngresa la direccion del alumno 2:");
    fgets(a2.direccion, 20, stdin);
    printf("\nDatos del alumno 1\n");
    printf("%d\n",a1.matricula);
    printf("%s\n", a1.nombre);
    printf("%s\n",a1.carrera);
    printf("%.2f\n",a1.promedio);
    printf("%s\n",a1.direccion);
    printf("\nDatos del alumno 2\n");
    printf("%d\n",a2.matricula);
    printf("%s\n",a2.nombre);
    printf("%s\n",a2.carrera);
    printf("%.2f\n",a2.promedio);
    printf("%s\n",a2.direccion);

    return 0;

}
```



# Copia de estructuras

El operador = se puede utilizar para asignar todos los campos de una estructura a otra.

Ejemplo:

```
#include<stdio.h>

struct libro{
    char titulo[30], autor[30], editorial[15];
    int anyo,edicion;
};

int main(){
    struct libro a = {"El Quijote","Cervantes","Limusa",1987,2}, b;
    b = a;//copia todos los datos de a en b
    printf("%s - %s - %s - %d - %d\n", b.titulo, b.autor, b.editorial,
b.edicion, b.anyo);
    return 0;
}
```

# Estructuras y apuntadores

Para acceder a una estructura mediante un apuntador se utiliza la siguiente notación:

`(*estructura).campo` o `estructura->campo`

Es necesario crear mediante malloc la estructura antes de usarla.

Ejemplo:

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

struct alumno{
    int matricula;
    char nombre[20];
    char carrera[20];
    float promedio;
    char direccion[20];
};

int main(){
    struct alumno *a;
    a = (struct alumno*)malloc(sizeof(struct alumno));
    (*a).matricula = 234; //a->matricula = 234;
    strcpy((*a).nombre, "Juan Perez");
        //strcpy(a->nombre, "Juan Perez");
    strcpy((*a).carrera, "Mate");
        //strcpy(a->carrera, "Mate");
    (*a).promedio = 6.7;
        //a->promedio = 6.7;
    strcpy((*a).direccion, "Lomas 34");
        //strcpy(a->direccion, "Lomas 34");


    printf("%d - %s - %s - %f - %s\n", a->matricula, a->
        nombre, a->carrera, a->promedio, a->direccion);

    return 0;
}
```

# Funciones de tipo estructura

```
#include <stdio.h>
#include <string.h>
struct dir{
    char calle[20];
    int numero;
    char colonia[20];
    int cp;
};
struct dir dameDir(){
    struct dir a={"alamo",43,"lomas",78000};
    return a;
}
```

Una función puede regresar una estructura



```
int main(){
    struct dir b;
    b = dameDir();
    printf("calle: %s %d\n",b.calle,b.numero);
    printf("colonia: %s\n",b.colonia);
    printf("CP: %d\n",b.cp);
    return 0;
}
```

# Funciones con parámetros de estructuras

Las estructuras se pasan como parámetros a las funciones de la misma forma que las variables de tipo simple.

Función para escribir datos de un alumno:

```
void escribeAlumno(struct alumno a){  
    printf("\nDatos del alumno\n");  
    printf("%d\n", a.matricula);  
    printf("%s\n", a.nombre);  
    printf("%s\n", a.carrera);  
    printf("%.2f\n", a.promedio);  
    printf("%s\n", a.direccion);  
}
```

Función para leer un alumno:

```
void lectura(struct alumno *a){
    printf("\nIngrese la matricula del alumno:");
    scanf("%d",&(a->matricula));
    getc(stdin);
    printf("\nIngrese el nombre del alumno:");
    fgets(a->nombre,20,stdin);
    printf("\nIngrese la carrera del alumno:");
    fgets((*a).carrera, 20, stdin);
    printf("\nIngrese el promedio del alumno:");
    scanf("%f",&a->promedio);
    getc(stdin);
    printf("\nIngrese la direccion del alumno:");
    fgets(a->direccion, 20, stdin);
}
```

`(*estructura).campo`      ⇔      `estructura->campo`

# Estructuras anidadas

Las estructuras pueden contener a otras estructuras como componentes.

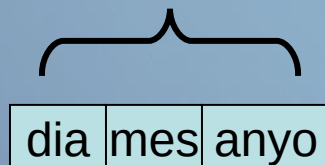
Una estructura que tiene componentes de tipo estructura se llama *estructura anidada*.

# Ejemplo

Ejemplo de estructuras anidadas

```
struct fecha{  
    int dia,mes,anyo;  
};  
  
struct persona{  
    char nombre[20];  
    char apellido[20];  
    struct fecha nacimiento;  
    int edad;  
    int sexo;  
    char CURP[19];  
    char telefono[20];  
};
```

nombre	apellido	nacimiento	edad	sexo	CURP	telefono
--------	----------	------------	------	------	------	----------



# Mostrar una fecha

```
void despliegaFecha(struct fecha f){  
    printf("%d de ",f.dia);  
    switch(f.mes){  
        case 1:printf("ene");break;  
        case 2:printf("feb");break;  
        case 3:printf("mar");break;  
        case 4:printf("abr");break;  
        case 5:printf("may");break;  
        case 6:printf("jun");break;  
        case 7:printf("jul");break;  
        case 8:printf("ago");break;  
        case 9:printf("sep");break;  
        case 10:printf("oct");break;  
        case 11:printf("nov");break;  
        case 12:printf("dic");break;  
    }  
    printf(" de %d\n",f.anyo);  
}
```



# Mostrar una persona

```
void despliegaPersona(struct persona p){  
    printf("Nombre: %s\n",p.nombre);  
    printf("Apellidos: %s\n",p.apellido);  
    printf("Fecha de nacimiento: ");  
    despliegaFecha(p.nacimiento);  
    printf("Edad: %d\n",p.edad);  
    if(p.sexo) // si es 1  
        printf("Sexo: masculino\n");  
    else  
        printf("Sexo: femenino\n");  
    printf("CURP: %s\n",p.CURP);  
    printf("Telefono: %s\n",p.telefono);  
}
```

# Leer una fecha

```
void leerFecha(struct fecha *f)
{
    printf("Dia? ");
    scanf("%d",&(f->dia));
    printf("Mes? ");
    scanf("%d",&(f->mes));
    printf("Anyo? ");
    scanf("%d",&(f->anyo));
}
```

# Leer una persona

```
void leerPersona(struct persona *p){  
    printf("Nombre? ");  
    scanf ("%[^, '\ n']", p->nombre);  
    printf("Apellidos? ");  
    scanf ("%[^, '\ n']", p->apellido);  
    printf("Fecha de nacimiento:\n");  
    leerFecha(&p->nacimiento);  
    printf("Edad? ");  
    scanf ("%d", &p->edad);  
    printf("Sexo (1-Hombre, 0-Mujer)? ");  
    scanf ("%d", &p->sexo);  
    printf("CURP? ");  
    scanf ("%[^, '\ n']", p->CURP);  
    printf("Telefono? ");  
    scanf ("%[^, '\ n']", p->telefono);  
}
```

# Acceso a estructuras anidadas

Se puede acceder a los campos de una estructura anidada mediante el operador “.”. Por ejemplo:

```
persona per,*per2;
```

```
per.nacimiento.dia = 5;  
per.nacimiento.mes = 7;  
per.nacimiento.anyo = 1998;
```

```
per2->nacimiento.dia = 1;  
per2->nacimiento.mes = 8;  
per2->nacimiento.anyo = 2005;
```

Note que el campo anidado se accede mediante el operador “.” y el no anidado mediante “->”.

# Ejemplo de empleado y estudiante

```
struct fecha{
    int dia,mes,anyo;
};
struct direccionStruct{
    char calle[30],colonia[20],
        ciudad[30],estado[15],pais[20];
    int numero,cp;
};
struct nombreStruct{
    char nombre[20],apellidos[20];
};
struct nomdir{
    struct nombreStruct nom;
    struct direccionStruct dir;
};
```

# Cont.

```
struct posicion{  
    char depto[5];  
    char trabajo[20];  
};
```

```
struct empleado{  
    struct nomdir nombreDireccion;  
    struct posicion trabajo;  
    float salario;  
    int numDepto;  
    struct fecha fechaIngreso;  
};
```

```
struct estudiante{  
    struct nomdir nombreDireccion;  
    char carrera[20];  
    float promedio;  
    int creditos;  
};
```

# Gráfico de las estructuras

fecha		
dia	mes	anyo

direccionStruct						
calle	colonia	ciudad	estado	pais	num	cp

nombreStruct	
nombre	apellidos

nomdir								
nombreStruct		direccionStruct						
nombre	apellidos	calle	colonia	ciudad	estado	pais	num	cp

posicion	
depto	trabajo

empleado																	
nombredireccion									posicion		salario	numDepto	fechaIngreso				
nom		dir							depto	trabajo				dia	mes	año	
nombre	apellidos	calle	colonia	ciudad	estado	pais	num	cp									

estudiante													
nombredireccion								carrera		promedio		creditos	
nom		dir											
nombre	apellidos	calle	colonia	ciudad	estado	pais	num						



```
int main(){
    struct nomdir per = {"juan", "perez lopez",
        {"olmo", "lomas", "SLP", "SLP", "Mexico", 32, 78000}};
    struct estudiante est = {}, {"fisica", 7.5, 210};
    struct empleado emp = {}, {"dep1", "afanador", 30000, 2, {5, 5, 2003}};
    est.nombreDireccion = per;
    emp.nombreDireccion = per;
    printf("nombre: %s\n", per.nom.nombre);
    printf("apellidos: %s\n", per.nom.apellidos);
    printf("nombre: %s\n", est.nombreDireccion.nom.nombre);
    printf("apellidos: %s\n", est.nombreDireccion.nom.apellidos);
    printf("promerio: %f\n", est.promedio);
    printf("nombre: %s\n", emp.nombreDireccion.nom.nombre);
    printf("apellidos: %s\n", emp.nombreDireccion.nom.apellidos);
    printf("Fecha ingreso: %d/%d/%d\n", emp.fechaIngreso.dia, emp.fechaIngreso.mes,
        emp.fechaIngreso.anyo);
    printf("Salario: %f\n", emp.salario);
    return 0;
}
```

# Arreglos de estructuras

Un arreglo de estructuras contiene elementos que son estructuras.

El siguiente ejemplo es un arreglo unidimensional de 50 elementos de tipo alumno.

```
struct estudiante{  
    struct nomdir nombreDireccion;  
    char carrera[20];  
    float promedio;  
    int creditos;  
};
```

```
main(){  
    struct estudiante est[50];  
    ...
```

# Pacientes en un hospital

La información de los pacientes de un hospital consiste de:

Nombre y apellidos (cadenas de caracteres)

Edad (entero)

Sexo (carácter)

Condición (entero )

Domicilio(estructura)

calle(cadena de caracteres)

número (entero)

Colonia (cadena de caracteres)

Código postal (cadena de caracteres)

Ciudad (cadena de caracteres)

Teléfono (cadena de caracteres)

Donde condición es un entero entre 1 y 5, 1 mínimo de gravedad, 5 máximo de gravedad.

# Estructura para pacientes

```
struct direccion{  
    char calle[30], colonia[20], ciudad[30];  
    int numero, cp;  
};
```

```
struct paciente{  
    char nombre[30];  
    int edad;  
    char sexo;  
    struct direccion dir;  
    int concicion;  
    char telefono[20];  
};
```

```
main(){  
    struct paciente pas[50]; //arreglo para 50 pacientes
```

# Ejemplos

Despliega nombre y teléfono de los pacientes con la máxima gravedad.

```
for(i = 0;i<50;i++)
    if(pas[i].condicion==5)
        printf("Nombre: %s, telefono: %s\n",
                pas[i].nombre,pas[i].telefono);
```

Calcula porcentaje de pacientes hombres y mujeres

```
suma = 0; sumaF = 0;
for(i = 0;i<50;i++)
    if(pas[i].sexo=='H')
        sumaH++;
    else
        sumaF++;
printf("% de Hombres= %.2f\n", sumaH/50*100);
printf("% de Mujeres= %.2f\n", sumaF/50*100);
```

Número de pacientes en cada condición

```
int c[5] = {0};
for(i = 0;i<50;i++)
    c[pas[i].condicion-1]++;
for(i = 0;i<5;i++)
    printf("Pacientes en condicion %d: %d\n",i+1,c[i]);
```

Nombre calle y número de pacientes masculinos en condición de máxima gravedad.

```
for(i = 0;i<50;i++)
    if(pas[i].condicion==5&&pas[i].sexo='H')
        printf("Nombre: %s, direccion: %s #%d\n",
            pas[i].nombre, pas[i].dir.calle, pas[i].dir.numero);
```

# Ejemplo de cartas

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

```
typedef struct {
    int numero;
    char palo;
}carta;
```

```
void inicia(carta *b){
    int i;
    for(i=0;i<52;i++){
        b[i].numero = i%13+1;
        b[i].palo = i/13+3;
    }
}
```

```
void revuelve(carta *b){
    int i,j;
    carta temp;
    for(i=51;i>=0;i--){
        j = rand()%(i+1);
        temp = b[i];
        b[i] = b[j];
        b[j] = temp;
    }
}

void despliegaCarta(carta c){
    printf("%d%c",c.numero,c.palo);
}

void despliegaBaraja(carta *c){
    int i;
    for(i=0;i<52;i++){
        despliegaCarta(c[i]);
        printf(" ");
    }
    printf("\n");
}
```

```

int main(){
    carta baraja[52];
    inicia(baraja);
    despliegaBaraja(baraja);

    revuelve(baraja);
    despliegaBaraja(baraja);

    return 0;
}

```

1♥ 2♥ 3♥ 4♥ 5♥ 6♥ 7♥ 8♥ 9♥ 10♥ 11♥ 12♥ 13♥ 1♦ 2♦ 3♦ 4♦ 5♦ 6♦ 7♦ 8♦ 9♦ 10♦ 11♦ 12♦ 13♦ 1♣ 2♣ 3♣ 4♣ 5♣ 6♣ 7♣ 8♣ 9♣ 10♣ 11♣ 12♣ 13♣ 1♠ 2♠ 3♠ 4♠ 5♠ 6♠ 7♠ 8♠ 9♠ 10♠ 11♠ 12♠ 13♠  
 10♦ 7♥ 7♣ 11♦ 11♣ 3♣ 10♥ 8♦ 9♠ 4♣ 6♣ 9♦ 13♦ 4♠ 5♥ 5♣ 13♣ 3♦ 12♣ 12♥ 2♣ 1♦ 1♥ 2♦ 8♠ 1♠ 4♦ 13♥ 9♥ 3♥ 4♥ 5♠ 7♠ 8♣ 13♠ 8♥ 11♥ 12♠ 6♠ 2♥ 7♦ 10♣ 10♠ 11♠ 6♦ 12♦ 1♣ 5♦ 2♠ 9♣ 6♥ 3♠



# Uniones

Una unión es una estructura en la que se comparte una región de memoria para almacenar datos de tipos distintos.

El tamaño de la unión es igual al del tipo de datos más grande.

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

```
union prueba{
```

```
    int a;
```

```
    float b;
```

```
    char c;
```

```
};
```

```
int main(){
```

```
    union prueba x;
```

```
    x.a = 5;
```

```
    printf("a= %d, b= %f, c= %c\n", x.a, x.b, x.c);
```

```
    x.b = 5.0;
```

```
    printf("a= %d, b= %f, c= %c\n", x.a, x.b, x.c);
```

```
    x.c = '5';
```

```
    printf("a= %d, b= %f, c= %c\n", x.a, x.b, x.c);
```

```
    return 0;
```

```
}
```

prueba
a
b
c

a= 5, b= 0.000000, c= ♣

a= 1084227584, b= 5.000000, c=

a= 1084227637, b= 5.000025, c= 5

# Ejemplo

```
struct fecha{
    int dia,mes,anyo;
};

struct persona{
    char nombre[20],apellido[20];
    struct fecha nacimiento;
    char sexo;
    union{
        struct {
            float peso,estatura;
        }varon;
        struct {
            int medidas[3];
        }mujer;
    };
};
```

```
void escribePersona(struct persona p){
    printf("nombre: %s %s\n",p.nombre,p.apellido);
    printf("fecha de nacimiento: %d/%d/%d\n",
        p.nacimiento.dia,p.nacimiento.mes,p.nacimiento.anyo);
    if(p.sexo=='H'){
        printf("sexo: masculino\n");
        printf("peso: %.1f, estatura: %.1f\n",
            p.varon.peso,p.varon.estatura);
    }
    else{
        printf("sexo: femenino\n");
        printf("medidas: %d, %d, %d\n",p.mujer.medidas[0],
            p.mujer.medidas[1],p.mujer.medidas[2]);
    }
}
```

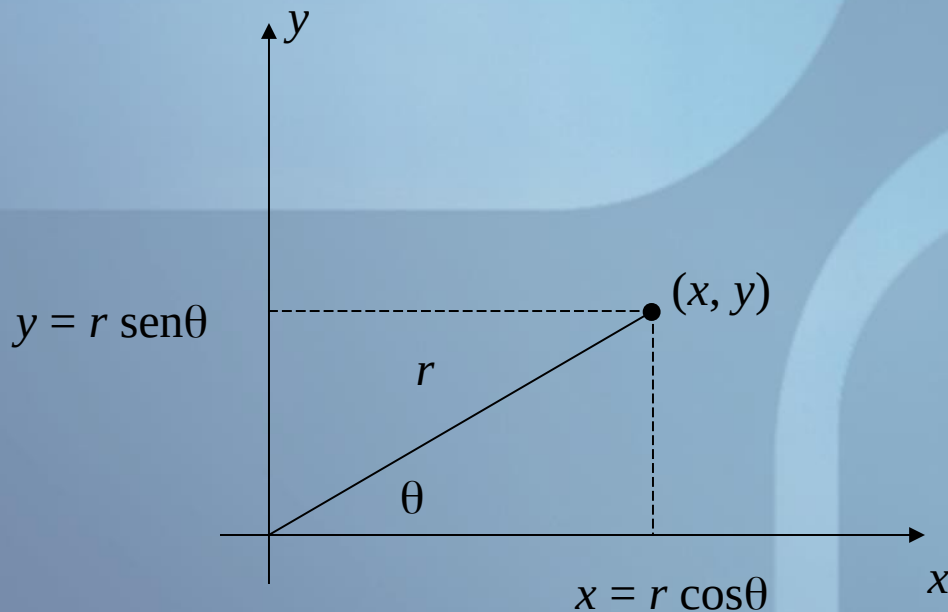
```
int main(){
    struct persona a = {"Juan", "Perez", {3,4,1980},
        'H', 80, 1.83},
        b = {"Luisa", "Lane", {16,7,1990},
        'M', 90, 60};
    escribePersona(a);
    escribePersona(b);
    b.mujer.medidas[0]=90;
    b.mujer.medidas[1]=60;
    b.mujer.medidas[2]=90;
    escribePersona(b);
    return 0;
}
```

```
nombre: Juan Perez
fecha de nacimiento: 3/4/1980
sexo: masculino
peso: 80.0, estatura: 1.8
nombre: Luisa Lane
fecha de nacimiento: 16/7/1990
sexo: femenino
medidas: 1119092736, 1114636288, 0
nombre: Luisa Lane
fecha de nacimiento: 16/7/1990
sexo: femenino
medidas: 90, 60, 90
```

# Coordenadas rectangulares y polares

Un punto en el plano se puede representar en coordenadas rectangulares o polares.

La relación entre ambos sistemas se muestra en la figura.



$$x = r \cos \theta$$

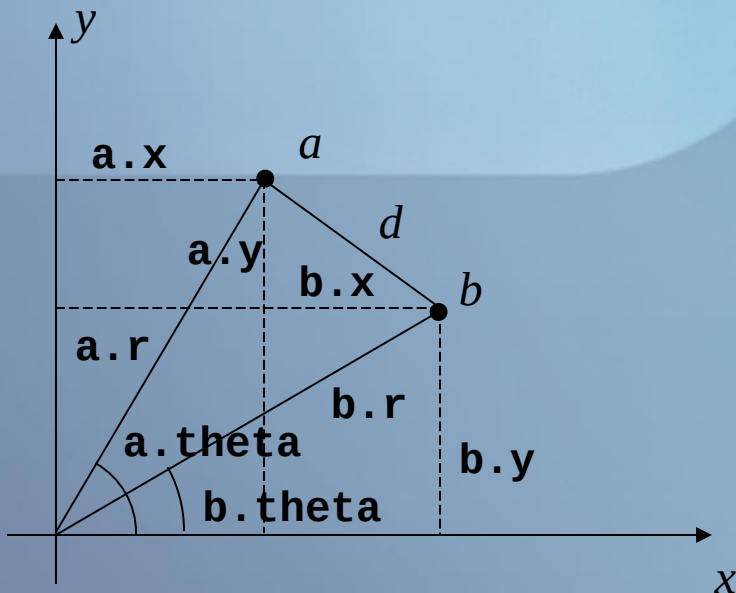
$$y = r \text{ sen} \theta$$

# Distancia entre dos puntos

La distancia entre dos puntos puede calcularse mediante el teorema de Pitágoras o haciendo uso de la ley de los cosenos.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$d = \sqrt{r_1^2 + r_2^2 - 2r_1r_2\cos(\theta_2 - \theta_1)}$$



```
struct coordenada{  
    int tipo;  
    union {  
        struct{double x,y;}rect;  
        struct{double r,theta;}pol;  
    };  
};
```

```
struct coordenada a,b;
```

a.rect.x - componente x de a  
a.rect.y - componente y de a  
a.pol.r - distancia al origen de a  
a.pol.theta - ángulo con eje x de a

# Cálculo de la distancia

```
double distancia(coordenada a, coordenada b){
    double d;
    switch(a.tipo){
        case CARTESIANA: switch(b.tipo){
            case CARTESIANA:
                d = sqrt((a.rect.x-b.rect.x)*(a.rect.x-b.rect.x)+
                        (a.rect.y-b.rect.y)*(a.rect.y-b.rect.y));
                break;

            case POLAR:
                d = sqrt((a.rect.x-b.pol.r*cos(b.pol.theta))*
                        (a.rect.x-b.pol.r*cos(b.pol.theta))+
                        (a.rect.y-b.pol.r*sin(b.pol.theta))*
                        (a.rect.y-b.pol.r*sin(b.pol.theta)));
                break;
        } break;
        case POLAR: switch(b.tipo){
            case CARTESIANA:
                d = sqrt((a.pol.r*sin(a.pol.theta)-b.rect.x)*
                        (a.pol.r*sin(a.pol.theta)-b.rect.x)+
                        (a.pol.r*cos(a.pol.theta)-b.rect.y)*
                        (a.pol.r*cos(a.pol.theta)-b.rect.y));
                break;
            case POLAR: d = sqrt(a.pol.r*a.pol.r+b.pol.r*b.pol.r-
                        2*a.pol.r*b.pol.r*cos(a.pol.theta-b.pol.theta));
                break;
        }
    }
    return d;
}
```