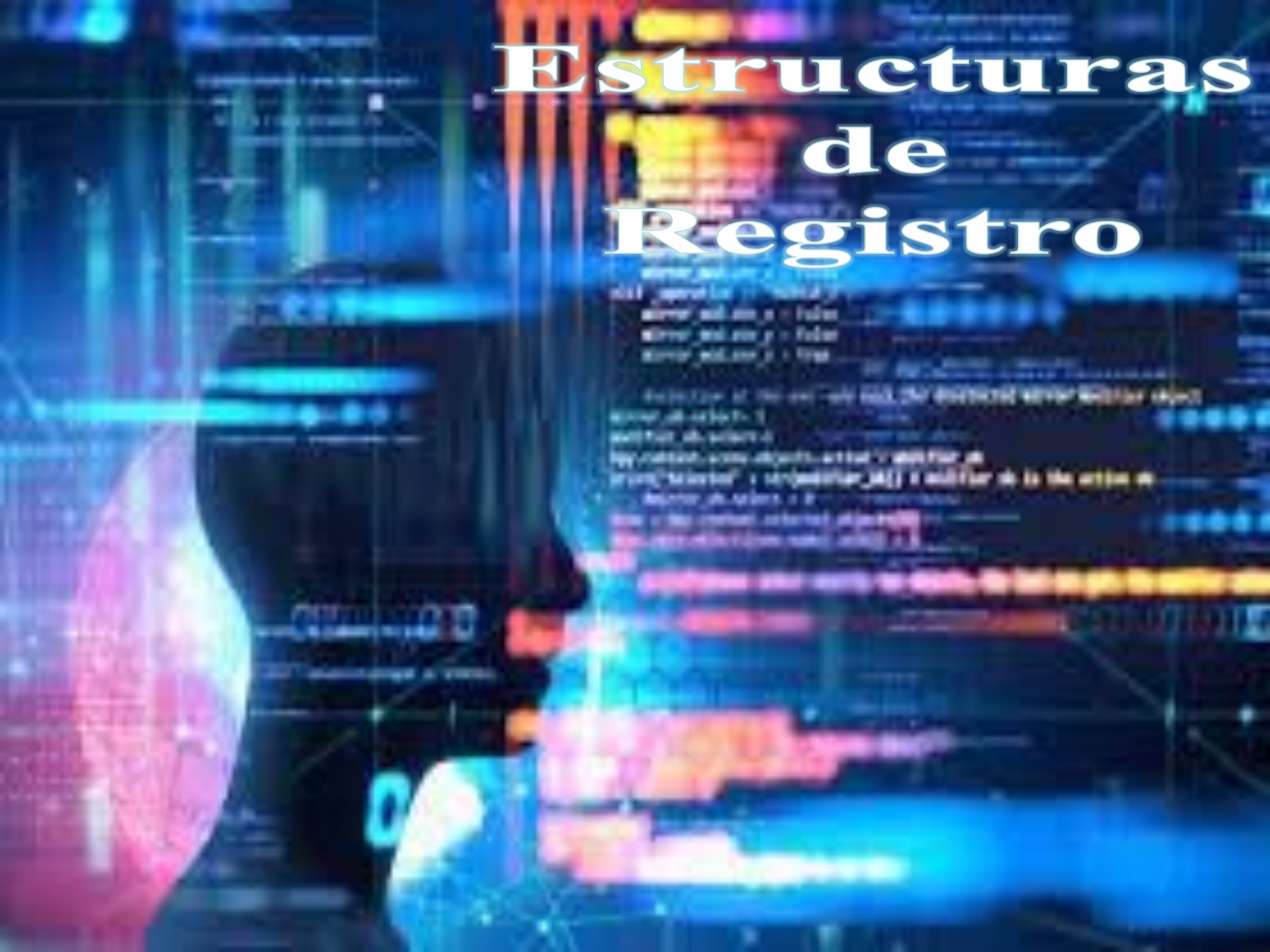
Estructuras
de
Registro

Supongamos que queremos hacer una agenda con los números de teléfono de nuestros amigos.

Necesitaríamos un array de cadenas para almacenar sus nombres, otro para sus apellidos y otro para sus números de teléfono.

Esto puede hacer que el programa quede desordenado y difícil de seguir. Y aquí es donde presenta ventajas el uso de las estructuras.

Estructura de Registro

Una estructura o registro es una estructura de datos que agrupa variables que pueden tener tipos diferentes.

Es decir, en una estructura en la que se puede definir datos o valores de diferentes tipos.

Cada componente de un registro se conoce como campo o miembro.

Definición de la Estructura

```
Struct  tiponuevo
    {
        tipo1  campo1;
        tipo2  campo2;
                :
         tipon  campon;
     }

struct tiponuevo nombre;
```

Los componentes de la estructura se declaran dentro de ella, cada uno con su tipo y nombre de variable, los tipos pueden ser diferentes.
El nombre (tiponuevo) de la estructura se considera como un tipo, no una variable.

Operaciones con Arreglos de Estructuras

- Cargar un arreglo de estructuras.
- Recorrer un arreglo de estructuras.
- Buscar un elemento en particular de estructuras.
- Acceder a un elemento en una posición determinada.
- Acceder a una posición determinada y mostrar su contenido.
- Insertar un nuevo elemento.
- Eliminar un elemento.
- Ordenar un arreglo de estructuras.

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

struct amigo
{
        int id;
        char nombre[20];
        char dir[20];
};

int main()
{
        struct amigo record = {0}; //declarar la estructura

        // asignar datos  a cada elemento de la estructura
        record.id=1;
        strcpy(record.nombre, "Raju");
        strcpy(record.dir, "catamarca 30");


        // mostrar por pantalla los datos de la estructura

        printf(" Id es: %d \n", record.id);
        printf(" Name es: %s \n", record.nombre);
        printf(" Dirección es: %s \n", record.dir);
        return 0;
}
```

```
#include <stdio.h>

struct estructura_amigo {           /* Definimos la estructura estructura_amigo */
            char nombre[30];
            char apellido[40];
            char telefono[10];
            int edad;
};

struct estructura_amigo amigo;   /* Declaramos la variable amigo con esa estructura */

int main()
{
  struct estructura_amigo amigo;/* Declaramos la variable amigo con esa estructura */
  printf( "Escribe el nombre del amigo: " );
  scanf( "%s", &amigo.nombre );
  printf( "Escribe el apellido del amigo: " );
  scanf( "%s", &amigo.apellido );
  printf( "Escribe el número de teléfono del amigo: " );
  scanf( "%s", &amigo.telefono );
  printf( "Mi amigo %s %s tiene el número: %s.\n", amigo.nombre, amigo.apellido, amigo.telefono );

return 0;
}
```

Supongamos ahora que queremos guardar la información de varios amigos. Con una variable de estructura sólo podemos guardar los datos de uno.

Para manejar los datos de más gente necesitamos declarar *arrays o arreglos* de estructuras.

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

struct agenda
{
        int id;
        char nombre[20];
        char dir[20];
};

int main()
{
        int i;

        struct agenda record[2]; //declarar la estructura

        // asignar datos  a cada elemento de la estructura
        record[0].id=1;
        strcpy(record[0].nombre, "Raju");
        strcpy(record[0].dir, "catamarca 30");

        record[1].id=1;
        strcpy(record[1].nombre, "Pepito");
        strcpy(record[1].dir, "Paz 350");

        record[2].id=1;
        strcpy(record[2].nombre, "Silvia");
        strcpy(record[2].dir, "cordoba 130");

        for(i=0; i<3; i++)
        {
                printf("    Registro de Estudiantes : %d \n", i+1);
                printf(" Id es: %d \n", record[i].id);
                printf(" Nombre es: %s \n", record[i].nombre);
                printf(" Dirección  es: %s \n", record[i].dir);
        }
        return 0;
}
```

¿Qué problema presenta el programa ?

```
#include <stdio.h>
#define ELEMENTOS 3
struct estructura_amigo
{
 char nombre[30];
char apellido[40];
char telefono[10];
int edad; };

int main()
{
struct estructura_amigo amigo[ELEMENTOS];
 int num_amigo;
for( num_amigo=0; num_amigo<ELEMENTOS; num_amigo++ )
{
printf( "\nDatos del amigo número %i:\n", num_amigo+1 );
printf( "Nombre: " );
gets(amigo[num_amigo].nombre);
printf( "Apellido: " );
gets(amigo[num_amigo].apellido);
printf( "Teléfono: " );
gets(amigo[num_amigo].telefono);
printf( "Edad: " );
scanf( "%i", &amigo[num_amigo].edad );

while(getchar()!= '\n'); /* Vacía el buffer de entrada */ }
/* Impresión de los datos */

for( num_amigo=0; num_amigo<ELEMENTOS; num_amigo++ )
{
printf( "Mi amigo %s ", amigo[num_amigo].nombre );
printf( "%s tiene ", amigo[num_amigo].apellido );
 printf( "%i años ", amigo[num_amigo].edad );
 printf( "y su teléfono es el %s.\n" , amigo[num_amigo].telefono );
}
return 0;
}
```

```c
#include <stdio.h>
#define ELEMENTOS 3
struct estructura_amigo
{
		char nombre[30];
		char apellido[40];
		char telefono[10];
		int edad; };
int main()
{
		struct estructura_amigo amigo[ELEMENTOS];
		int num_amigo;
		int sum_edad=0;
		float prom_edad=0;;

		for( num_amigo=0; num_amigo<ELEMENTOS; num_amigo++ )
		{
				printf( "\nDatos del amigo número %i:\n", num_amigo+1 );
				printf( "Nombre: " );
				gets(amigo[num_amigo].nombre);
				printf( "Apellido: " );
				gets(amigo[num_amigo].apellido);
				printf( "Teléfono: " );
				gets(amigo[num_amigo].telefono);
				printf( "Edad: " );
				scanf( "%i", &amigo[num_amigo].edad );

				//while(getchar()!= '\n'); /* Vacía el buffer de entrada */
		  sum_edad=sum_edad+amigo[num_amigo].edad;

		}
		/* Impresión de los datos */
		for( num_amigo=0; num_amigo<ELEMENTOS; num_amigo++ )
		{
				printf( "Mi amigo %s ", amigo[num_amigo].nombre );
				printf( "%s tiene ", amigo[num_amigo].apellido );
				printf( "%i años ", amigo[num_amigo].edad );
				printf( "y su teléfono es el %s.\n" , amigo[num_amigo].telefono );
		}
		prom_edad=sum_edad/ELEMENTOS;
		printf("\n");
		printf ("Promedio de edad: %f\n",prom_edad);
		return 0;
}
```

¿Es posible hacer el programa anterior con funciones ?

Otro ejemplo

```
#include <stdio.h>
#include <conio.h>
# define nro 5

struct cuentas
        {
                int nro_cuenta;
                char nombre[30];
                int saldo;
        };

int main()

        {
                struct cuentas cuen[nro];
                int i,deudor=0,acreedor=0;
```

```
//Carga del Arreglo Cuentas

for(i=0;i<nro;i++)
{
        printf("Ingrese datos Estructura: %d\n",i+1);

        printf("\nIngrese Nro de Cuenta\n");
        scanf("%i",&cuen[i].nro_cuenta);

        printf("Ingrese nombre del Cliente\n");
        scanf("%s",&cuen[i].nombre);

        printf("Ingrese Saldo del Cliente\n");
        scanf("%d",&cuen[i].saldo);
        printf("\n");
}
```

```
//Mostramos por pantalla los registros cargados

for(i=0;i<nro;i++)
 {
     printf("Nro Cuenta= %d\n",cuen[i].nro_cuenta);
     printf("Cliente= %s\n",cuen[i].nombre);
     printf("Saldo= %d\n",cuen[i].saldo);
 }

//Mostramos por pantalla los registros con formato de informe

printf("Ingrese datos arreglo: %d\n",i+1);
printf("Nro. Cuenta      Cliente          Saldo   \n");

for(i=0;i<nro;i++)
 {
     printf("%d" , cuen[i].nro_cuenta);
     printf("                %s",       cuen[i].nombre);
     printf("           %d\n",cuen[i].saldo);
 }
```

```
//Informamos por Cliente Estados de las Cuentas.
printf("\n");

for(i=0;i<nro;i++)
 {
   if (cuen[i].saldo<0)
     {
        printf("El estado de la cuenta para el cliente: %s es: EUDOR\n",cuen[i].nombre);
         deudor++;
     }
       else
        {
            printf("El estado de la cuenta para el cliente: %s es:ACREEDOR\n",cuen[i].nombre);
            acreedor++;
        }
 }

printf("Hay %d Ciientes Deudores\n",deudor);
printf("Hay %d Clientes Acreedores\n",acreedor);

return 0;

}
```