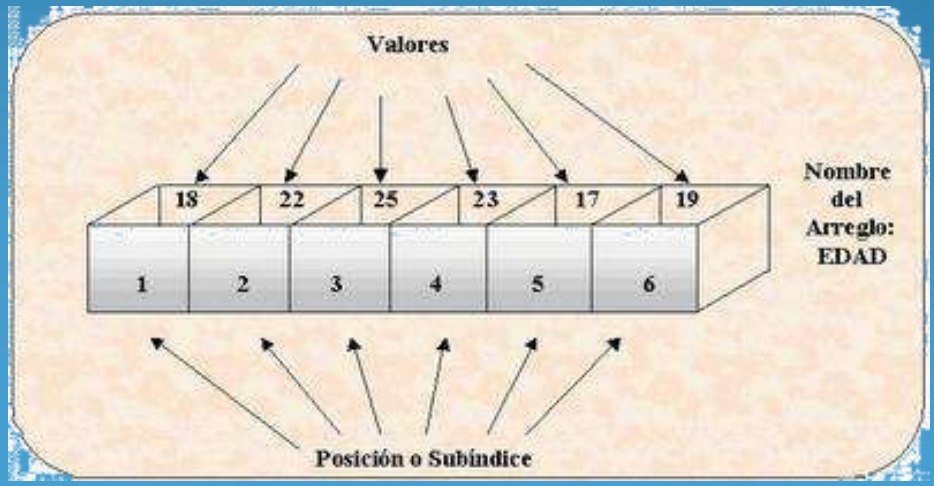
Estructuras

Almacenamiento estático



Arreglos

Conjunto

- ***Finito: Tiene un tamaño definido***
- ***Ordenado: Significa que el elemento primero, segundo, hasta el n-ésimo de un arreglo puede ser identificado.***
- ***Homogéneo significa que todos los elementos de un arreglo son del mismo tipo de datos.***

que se referencían por un identificador común (nombre).

clasificación

- ***Unidimensionales (Vectores).***
- ***Bidimensionales (Tablas o Matrices)***
- ***Multidimensionales.***

- ***Nota: En C todos los arreglos empiezan en posición o (cero)***

Unidimensional

- ***El subíndice o índice de un elemento [1, 2, ..,i,.., n] designa su posición en el orden del vector.***

Por ejemplo

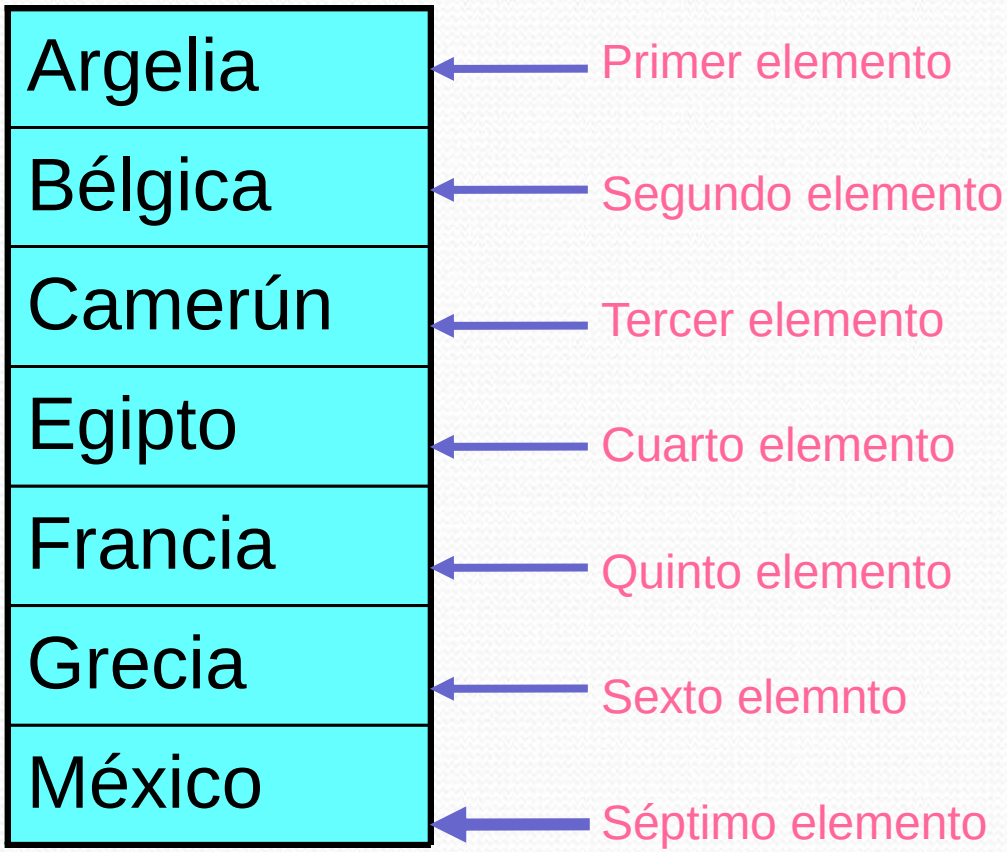
paises

que consta de 7 elementos

Vector

Los identificadores no pueden llevar acento

paises



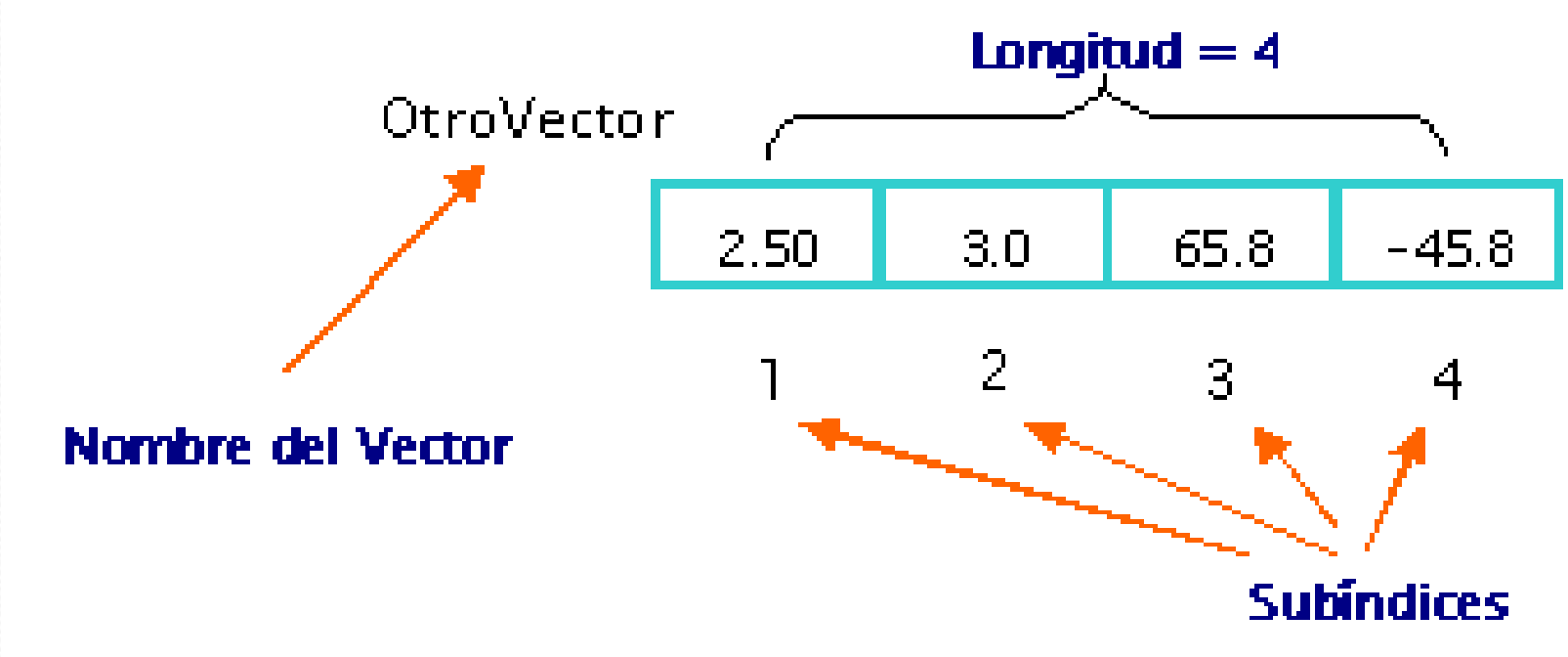
Vector

- ***Los elementos del vector productos se representan con la siguiente notación:***

paises[1] almacena Argelia

paises[2] almacena Bélgica

paises[3], …, productos[7]


Longitud = 4
OtroVector
2.50
3.0
65.8
-45.8
1
2
3
4
Nombre del Vector
Subíndices

Unidimensionales

Sintaxis: Declaración

tipo nombre [tamaño];

- **Tipo:** Es uno de los tipos predefinidos por el lenguaje, es decir int, float, etc.
- **Nombre:** Es un identificador que nombra el arreglo .
- **Tamaño:** Es una constante que especifica el numero de elementos del arreglo.

Ejemplo

- char x[20]; /* cadena de 20 caracteres */
- float z[40]; /* arreglo de 40 reales*/
- int vector[100]; /* Arreglo de enteros*/

Ejemplo: Sumar dos vectores

```
#include <stdio.h>
main( )
{
 int n;            /* donde n<=100 */
 int a[100], b[100], c[100];      /* Arreglos */
 int i;
 a[100]=b[100]=c[100]={0}; /* limpia a,b,c*/
 printf (“Numero de elementos a sumar: “);
 scanf(“%d”,&n);
 printf (“Elementos del vector a \n”);
```

Ejemplo

```
for (i=0; i<n; i++)
   scanf("%d",&a[i]);
printf ("Elementos del vector b \n");
for (i=0; i<n; i++)
   scanf("%d",&b[i]);
printf ("Suma de vectores \n");
for (i=0; i<n; i++)
    c[i]=a[i] + b[i];
printf ("Resultados \n");
for (i=0; i<n; i++)
    printf("%d",c[i]);
}
```

Cadenas

- Son arreglos unidimensionales de tipo char.
- Una cadena en C se termina con el centinela de fin de cadena o carácter nulo **'\0'**

Ejemplo:

Longitud = 13

e	s		u	n	a		c	a	d	e	n	a	\0

o	t	r	a	2	"	[	{	}	d	e	n	.	\0

Ejemplo: calcula el # de caracteres

```
#include<stdio.h>
#include<string.h>
main ( )
{ char cadena[20];
  int tam,i;
  printf( " dame la cadena " );
  gets ( cadena );
  for ( i = 0; cadena [ i ]! = ' \0 '; i + +); /*strlen(cadena)*/
    tam = i; / * tam= largo de una cadena * /
  puts(cadena);
  printf("tiene%delementos", tam);
 }
```

Librería string.h

strlen. Calcula la longitud de una cadena.

Sintaxis: longitud=strlen(cadena);

Strcpy. Copia el contenido de una cadena sobre otra.

Sintaxis: strcpy(copia , original);

Strcat. Concatena dos cadenas.

Sintaxis: strcat(cadena1 , cadena2);

Strcmp. Compara el contenido de dos cadenas.

Si **cadena1 < cadena2** retorna un número negativo.

Si **cadena1 > cadena2**, un número positivo

si **cadena1** es igual que **cadena2** retorna **0** (o **NULL**).

Sintaxis: valor=strcmp(cadena1 , cadena2);

Práctica 5

En equipo de 3 diseña los programas

1. Hacer un programa que almacene 40 números enteros en un vector, imprimir cuantos son cero, cuantos negativos, cuantos positivos. Imprimir la suma de los positivos.
2. Llenar dos arreglos con 10 caracteres cada uno, posteriormente verificar si son iguales entonces imprimir “IGUALES” en caso contrario imprimir “DIFERENTES”

Práctica 5

3. Llena una cadena de tamaño 50 y posteriormente recórrela para contar cada una de las vocales encontradas

Tarea 7

- ***Investigar la sintaxis para 6 funciones de la librería string.h***
- ***Incluir 1 ejemplo para cada función***
- ***Incluir una prueba de escritorio para cada ejemplo***
- ***Incluir la referencia bibliográfica***

Tarea 8 y 9

Aplicaciones de arreglos

Búsqueda

- ***Proceso de determinar el elemento, o su posición, que cumple una condición.***

Búsqueda Secuencial

- ***Compara con cada uno de los elementos en forma secuencial.***
- ***Actúa sobre arreglos desordenados.***

VALOR a Buscar = 21

A[0]=1 <> VALOR

A = 1 11 21 25 26 33 38 40 42 48

i=0

Segunda iteración: A[1]=11 <> VALOR

A = 1 11 21 25 26 33 38 40 42 48

i=1

Tercera iteración: A[2]=21 = VALOR

A = 1 11 21 25 26 33 38 40 42 48

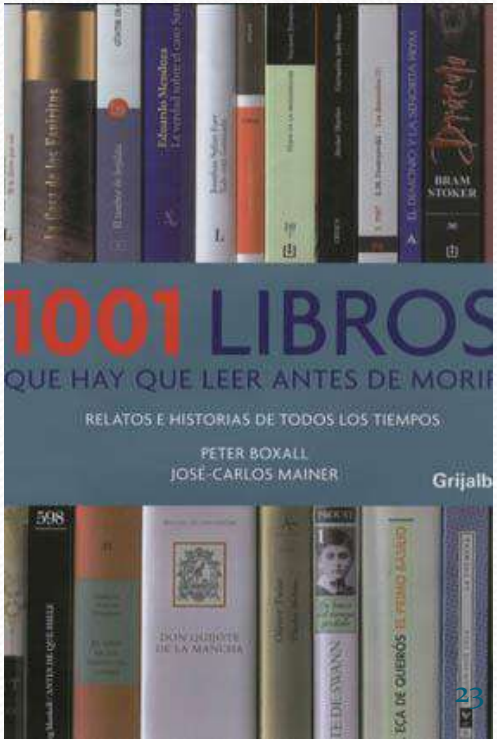
i=2

Programa

```
#include<stdio.h>
#define N 10        /*defino una constante*/
main()
{ int I,pos, num[N],valor;
  printf("Qué valor buscas?");
  scanf("%d", &valor);
   pos=0;  I=0;
  while( ( I<  N ) && ( num[ I ] != valor ) )
            I = I + 1;
      /* Determinar si encontró o no */
      if I < N
         { pos = I; printf(" se encontró el elemento en la posición",pos);}
      else printf("no se encontró el elemento");
}
```

Búsqueda Binaria

- ***El método requiere que la información sobre la cual se va a buscar este ordenada.***
- ***Al estar ésta ordenada puede descartarse la mitad que se sabe no es posible que este la información***



Valor buscado = 72

Primera iteración: A[m]=48 <> valor

A = 1 11 21 25 26 33 38 40 42 48 50 56 59 60 62 64 67 72 76 77

i=0 *m=10* *j=19*

valor > A[m] se descarta la primera mitad

Segunda iteración: A[m]=64 <> valor

A = 1 11 21 25 26 33 38 40 42 48 50 56 59 60 62 64 67 72 76 77

i=11 *m=15* *j=19*

valor > A[m] se descarta la primera mitad

Tercera iteración: A[m]=72 = valor

A = 1 11 21 25 26 33 38 40 42 48 50 56 59 60 62 64 67 72 76 77

i=16 m=17 j=19

Termina la búsqueda, valor se encuentra en posición: m= 17

```
#include<stdio.h>
#define  CANT 20
main()
{
  int izq, der, pos, mitad,  A[CANT],busco;
  mitad = CANT / 2;
  izq = 0; der = CANT-1;
  scanf("%d", &busco);
         /* Recorrido del arreglo buscando el valor */
 while ( ( A[ mitad ] != busco ) && ( izq < der ))
    {
       if A[mitad] > busco            /*mitad izquierda*/
         der = mitad – 1
       else
         izq = mitad + 1;             /*mitad derecha*/
       mitad = ( izq + der ) / 2;
    }
/* Determinar si encontró o no */
  if A[mitad] == busco
     pos = mitad;
}
```

Tarea 10

METODOS

DE

ORDENAMIENTO

Ordenamiento

- ***Significa mover los datos o sus referencias para que queden en una secuencia tal que represente un orden (bajo un criterio), el cual puede ser numérico, alfabético o incluso alfanumérico, ascendente o descendente.***

Tipos de Ordenamiento

Ordenamiento interno.

- ***Se lleva a cabo completamente en memoria principal.***

Ordenamiento externo.

- ***No cabe toda la información en memoria principal y es necesario ocupar memoria secundaria.***

Criterios de Eficiencia

- ***El número de pasos.***
- ***El número de comparaciones entre elementos p***
- ***El número de movimientos de elementos que se requieren para ordenar n registros.***

Métodos Simples

- ***Método de Burbujeo***
- ***Método de Selección***
- ***Método de Inserción***

Método de Burbujeo corregir

```
#define N 20
void main()
{   int A[N],  i,j,temp;
     for( i=0; i < N-1; i++ )
        for( j=N-1; j > i; j--)
           if( A[j] < A[j-1] ){
            temp=A[j];
            A[j]=A[j-1] ;
            A[j-1]=temp}
}
```

25	3	12	19	2	1	9	6
1	25	3	12	19	2	6	9
	2	25	3	12	19	6	9
		3	25	6	12	19	9
			6	25	9	12	19
				9	25	12	19
					12	25	19
						19	25

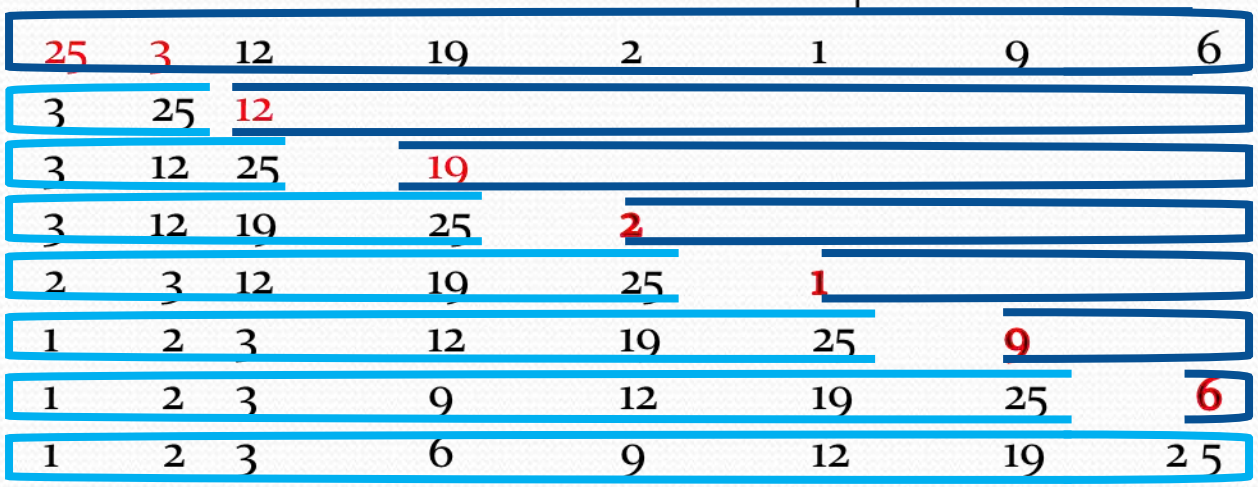
Método de Selección corregir

25	3	12	19	2	1	9	6
1	3	12	19	2	25	9	6
	2	12	19	3	25	9	6
		3	19	12	25	9	6
			6	12	25	9	19
				9	25	12	19
					12	25	19
						19	25

```
#define N 20
void  main()
{   int A[N], i, j, imin, temp;
    for( i=0; i < N-1; i++ )
        {  imin = i;
           for( j = i+1; j < n; j++ )
                if( A[j] < A[imin] ) imin = j;
                temp=A[i];
                A[i]=A[imin] ;
                A[imin]=temp;}
         }
}
```

Método de Inserción

```
#define N 20
void main()
 { int A[N],  i,j,temp;
  for( i=1; i < N; i++ ) {
      j:= i;
       while( j > 0 && A[j] < A[j-1] ) {
           temp=A[j];
          A[j]=A[j-1] ;
          A[j-1]=temp;}
        j--
       }
    }
   }
```



Tarea 11

Bidimensionales

Sintaxis: Declaración

- **Tipo nombre[renglones][columnas];**
- **Tipo:** Es uno de los tipos predefinidos por el lenguaje, es decir int, float, etc.
- **Nombre:** Es un identificador que nombra el arreglo .
- **Renglones:** Indica el # de renglones de la matriz
- **Columnas:** Indica el # de columnas de la matriz

Matriz m de 12 enteros

int m[3][4];

Columnas

Filas

	0	1	2	3
0	5	-3	5	9
1	3	6	8	-2
2	-4	6	7	9

Es el elemento[1][2] = 8

Ejemplo: cuenta +,- y 0

```
 #include <stdio.h>
/*Cuenta positivos, negativos y ceros*/
main()
{
  int ma, cp,cn,cc;
  int num, ma[10][10];
  cp=cn=cc=0;
  do{                /* validar #_filas y #columnas*/
    printf("Da el numero de renglones: ");
    scanf("%d",&n);
    printf("Da el numero de columnas: ");
    scanf("%d",&m);
  }while((n>10)||(m>10));
```

Ejemplo

```
printf("Lectura de la Matriz : \n")
for (r=0 ; r<n;r++)
        for(c=0; c<m;c++)
            scanf("%d",&ma[r][c]);

       /* Recorre para contar */
for (r=0 ; r<n;r++)
        for(c=0; c<m;c++)
            {
             if(ma[r][c] > 0)        cp=cp+1;
             if(ma[r][c] < 0)        cn=cn+1;
             if(ma[r][c] == 0)       cc=cc+1;
            }
printf("%d %d %d",cp,cn,cc);
}
```

Práctica 6

En equipo de 4

1. Elaborar un programa que llene una matriz por columnas y la imprima por filas. Solicitando al usuario N_filas y N_columnas.
2. Para el ejercicio anterior, encuentre el elemento mayor y el menor, así como sus posiciones, en caso de que alguno de los elementos (mayor y/o menor) esté varias veces en la matriz, indíquele esta situación al usuario mediante un mensaje.

3. Elabore un programa que llene una matriz de un tamaño definido por el usuario. Luego intercambie el contenido de la primera y la última columna, de la segunda y la penúltima y así hasta completar cambios que den una matriz con columnas invertidas. Imprima ambas matrices.

Práctica 6

4- En las elecciones para alcalde del PUEBLO ÚNICO se han presentado tres candidatos (A,B,C). el pueblo está dividido en 5 zonas de votación.
El reporte de votos de las zonas se recibe en orden: primero la zona 1, la 2, etc. Elabore un programa que:
• Forme una matriz de 5 filas y 3 columnas que contenga, en cada fila, los votos reportados por las zonas para cada uno de los tres candidatos.
• Encuentre el total de votos obtenidos por cada candidato y el porcentaje que éste representa.
• Escriba un mensaje declarando ganador a un candidato, si éste obtuvo más del 50% de la votación, en caso de "empate", notifíquelo mediante un mensaje.

Tarea 12

Estructuras(Registro)

- Es una colección de datos heterogéneos, lógicamente relacionados.
- Define un nuevo tipo de datos.
- Cada elemento de una estructura se denomina miembro o campo

Declaración

```
struct nombre_estructura
    {
      tipo nombre_variable;
      tipo nombre_variable;
      tipo nombre_variable;
       ...
    } variables_estructura;
```

Descripción

nombre_estructura

Identificador que nombra el nuevo tipo definido.

variables_estrutura

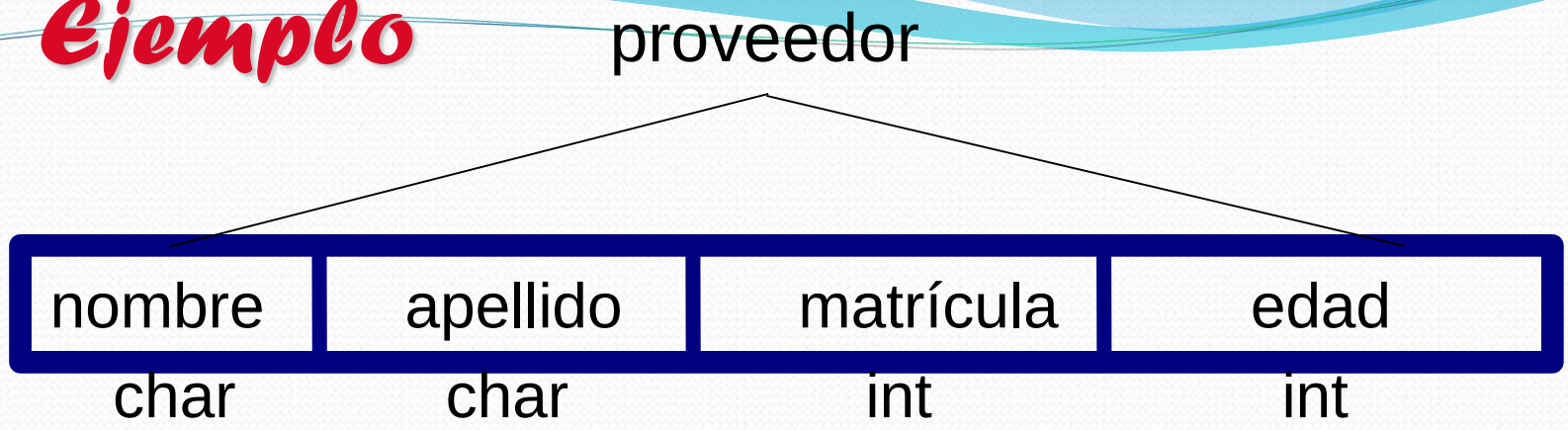
son identificadores para acceder a los campos de la estructura.

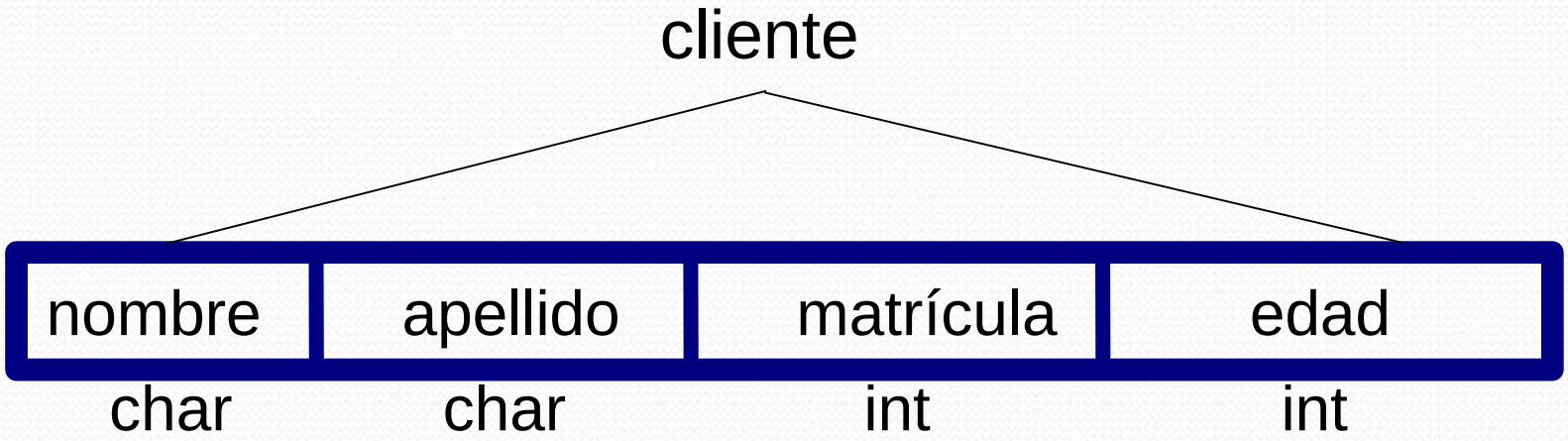
Ejemplo

Ejemplo:

```
        struct ficha
                {
                  char nombre[20];
                  char apellidos[40];
                  int matricula;
                  int edad;
                }provedor, cliente;
            ó
struct ficha proveedor,cliente;
```

Ejemplo


proveedor
nombre
apellido
matrícula
edad
char
char
int
int


cliente
nombre
apellido
matrícula
edad
char
char
int
int

Acceso a los campos

Sintaxis

nombre_var_estructura . nombre_campo

Ejemplo

cliente . nombre= “Juan”;

proveedor . nombre= “Carlos”;

cliente . edad=22;

Ejemplo: Eres mayor de edad?

```
#include <stdio.h>
struct ife
        {
           char nombre[20];
           char apellidos[40];
           int folio;
           int edad;
        }ciudadano;
```

Ejemplo

```
main()
{
  printf(“dame los datos del ciudadano”);
  gets(ciudadano.nombre);
  gets(ciudadano.apellido);
  scanf(“%d”, &ciudadano.folio);
  scanf(“%d”, &ciudadano.edad);
  if (ciudadano.edad > 18)
    printf(“ERES MAYOR DE EDAD ”);
  else
    printf(“ERES MENOR DE EDAD ”);
}
```

Arreglo de estructuras

Sintaxis:

struct
nombre_var_estructura[tamaño];

Ejemplo:

0		Juan	López	10098877	34
1		Lola	Ramos	123456	56
2		Tere	Ramos	123456345	24
.					
.					
19					

struct tipo_ciudadano[20];

Ejemplo: Eres mayor de edad?

```
#include <stdio.h>
struct ife
        {
           char nombre[20];
           char apellidos[40];
           int folio;
           int edad;
        }ciudadano[20];
```

Ejemplo

```
main()
{ int nc;
   struct ife ciudadano[20];
  for(nc = 0; nc < 20; nc ++)
    {
     printf("dame los datos del ciudadano");
     gets(ciudadano[nc].nombre);
     gets(ciudadano[nc].apellido);
     scanf("%d", &ciudadano [nc].folio);
     scanf("%d", &ciudadano [nc].edad);
     if (ciudadano[nc].edad > 18)
        printf("ERES MAYOR DE EDAD ");
     else
       printf("ERES MENOR DE EDAD ");
    }
```

Práctica 7

En equipo de 4

1. Hacer un programa que lea información para 20 estudiantes. Dicha información consiste de: nombre, edad, matricula, calif1,calif2,calif3

Para cada estudiante deberá calcular el promedio y deberá imprimir el porcentaje de estudiantes aprobados(promedio mayor o igual a 6) y porcentaje de reprobados (promedio menor a 6). Se deben validar los datos de entrada.