Tecnicatura Universitaria

en Desarrollo de Software

Arreglos

Datos estructurados

Cuando hablamos de estructuras de datos hacemos referencia a un conjunto de datos que poseen un mismo nombre, que pueden ser caracterizados por su organización y por las operaciones que podemos hacer sobre ellos.

- Tipos simples: int, float, double, char
- Tipos estructurados estáticos: Arreglos y registros o estructuras

Arreglos

- Las estructuras de datos estáticas, son aquellas en el que el tamaño que ocuparán las mismas se define antes que el programa se ejecute y el mismo no puede ser modificado durante la ejecución.
- Los tipos de datos que vimos hasta ahora, son tipos simples, cada variable representa un elemento.
- Los tipos estructurados tienen la particularidad que con un **identificador** pueden representarse múltiples datos individuales y a la vez cada uno de ellos puede ser referenciado por separado.

Arreglos

Un arreglo se define como una colección **finita**, **ordenada** y **homogénea** de elementos.

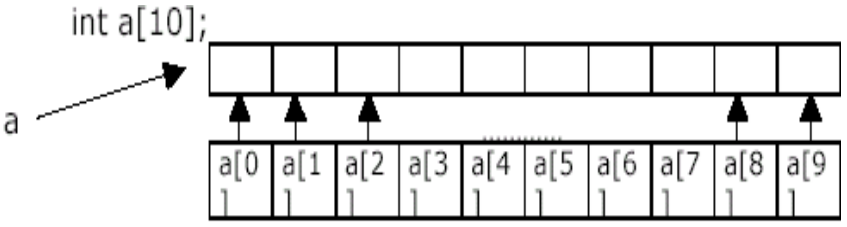
- **Finita**: todo arreglo tiene un límite, debe determinarse cual será el máximo de elementos que contendrá.
- **Ordenada**: se debe determinar cual es el primer elemento, el segundo, el enésimo.
- **Homogénea**: todos los elementos son del mismo tipo, tales como enteros, booleanos, etc.

Un arreglo tiene la característica de poder almacenar N elementos de un mismo tipo, distinguimos 2 aspectos:

- Los elementos o componentes (valores guardados en cada posición del arreglo),
- Los índices (permiten hacer referencia a un elemento)

Arreglos

Graficamente declaramos un arreglo a de 10 elementos lo representaremos



Donde a[i] es el i-ésimo elemento del arreglo.

La forma general de la declaración es:

```
<tipo> <variable de array>[<nro_elementos>][[<nro_elementos>]...];
Ej. double temperaturas [31];
```

En C no se puede operar con un arreglo como una unidad. Hay que tratar los elementos por medio de iteraciones de tipo **for o while** que

- Los elementos si se operan como cualquier variable. Ejemplos:

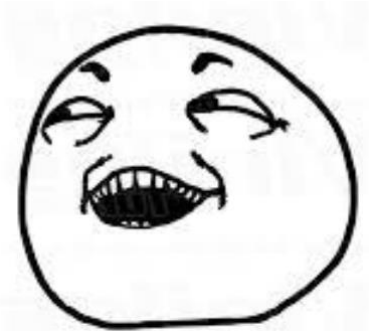
```
int edades[10], i=0;
edades[3]=18;    edades[4]=edades[3];
edades[5]=edades[4]+1;  edades[6]=15*3+1;  edades[i]=100;
```

En este ejemplo i es el índice y edades[i]contiene el valor 100 que se le asignó.

Particularmente la cadena de caracteres es un vector de tipo char:
`char Nombre[20]=“Juan”` donde se utiliza los cuatro primeros lugares mas un quiento para indicar el fin de la cadena.

- Necesitamos calcular el promedio de edades de 20 alumnos de un curso

```
…
int edad[20], i;
float tot=0;
for (i=0;i<20;i++)
   {cout << “ingrese una edad”<<endl;
   cin >> edad[i];
   tot=tot+edad[i];}

tot=tot/i;
cout << “El promedio de edades es “<<tot;
…
```



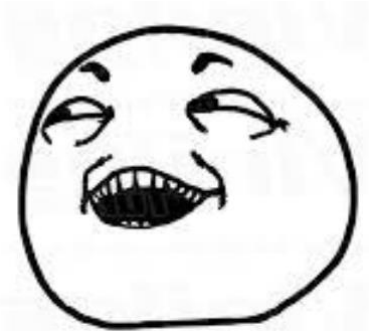
- Los elementos de un arreglo también deben ser inicializados, podemos hacerlo en la declaración del mismo usando la siguiente asignación

Ej. `char vocales[5]={“a”,“e”,“i”,“o”,“u”};`

También podemos inicializarlo recorriéndolo.

Ej. Tenemos la declaración

```
int Arr[10];
For (int x=0; x<10;x++)
   Arr[x]=x;
```



Arreglos

- Declaración

```
int arreglo1[30]; //declara que arreglo1 es un arreglo que puede contener 30 enteros.

#define TAMANIO 100
int arreglo2[TAMANIO];  //declara que arreglo2 es un arreglo que puede contener TAMANIO enteros

int notas[8]; /* almacena ocho notas */
char nombre[21]; /* almacena nombres de largo menor o igual a 20 */
int multiplos[n]; /* donde n tiene un valor, declara un arreglo de tamaño n*/
```

Arreglos

- Inicialización

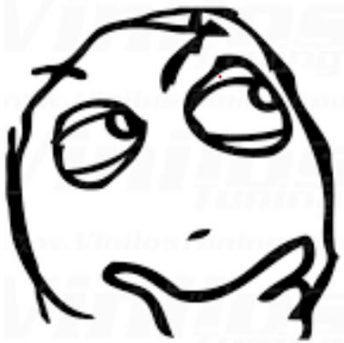
```
int n[10]={32, 27, 64, 18, 95, 24, 90, 70, 8, 3};

int n[10]={7}; //completas los demás valores con 0

int a1[] = {1,2,3,4,5}; //crea un arreglo de 5
elementos.
```

Arreglos

```
int arreglo2[TAMANIO];

/* cargo array con valor igual al indice mas 10*/
for (int i=0;i<TAMANIO;i++)
    arreglo2[i]=i+10;

/* imprimo contenido del arreglo */
for (int i=0;i<TAMANIO;i++)
    printf("Elemento %d del array es %d\n",i+1,arreglo2[i]);
}
```

Recorrer un Arreglo

```
for(i=0; i<N;i++)
      scanf(“%d”, &miarray[i]);
for(i=0; i<3;i++)
       printf(“%d”, mis_tres_numeros[i]);

int main() {
int vec[]= {1,2,-2,1,3,-1,5,10}, i, var;
 var=1;
 for( i = 0; i <= 7; ++i )
      if(vec[i]>0)
         var = var * vec[i];
 printf ( “ %d”, var);
}
```



- Las ventajas que presenta el uso de arreglos son:
  - Posibilita el uso de menor número de variables, dado que en un identificador agrupamos n variables, donde n es el tamaño del arreglo.
  - Los arreglos permiten manejar datos con cierto orden
  - Facilidad de acceso a la información, ya sea a un elemento en particular o recorrido, que puede hacerse del 1º al último o al revés.

Ejercicio: Resolver

En grupos resolver:

1. Guardar en el arreglo los valores que el usuario introduce.
2. Mostrar todo el contenido de un arreglo
3. Sumar todos los valores del arreglo.
4. Obtener el valor máximo del arreglo

Cadena de caracteres

- Una cadena de caracteres es un array de caracteres que contiene el ‘\0’ (el carácter nulo, o carácter fin de cadena)

```
char micadena1[10]={‘h’,’o’,’l’,’a’,’\0’};
char micadena2[50]=”que tal”;

puts(micadena1); //imprime : hola
puts(micadena2); //imprime : que tal
puts(“Introduzca su nombre por favor:”);
gets(micadena1);

srcat(micadena2,”estás? ”); //cadena2 = que tal estas?
srcat(micadena2, micadena1);

puts(micadena2); // que tal estas? jose
```

Ejercicio

1. Guardar en el arreglo los valores que el usuario introduce.

```
for(i=0; i<=9; i++){
        printf("\nEscriba el valor %d", i);
        scanf("%f", &grupo[i]);
}
```

2. Mostrar todo el contenido de un arreglo

```
for (i=0; i<10; i++){
  printf("En posicion %d el valor es %f\n", i, grupo[i]);
}
```

Ejercicio

3. Sumar todos los valores del arreglo.

```
float suma =0;
for(i=0; i<10; i++){
        suma+=grupo[i];
}
float media = suma / 10;
printf("La media es %.2f", media);
```

4. Obtener el valor máximo del arreglo

```
float max=grupo[0];
for(i=0; i<10; i++){
        if (grupo[i] > max)
                max=grupo[i];
}
printf("El valor máximo es %.2f", max);
```

MATRIZ: Arreglos de más de una dimensión

Llamamos matrices a los arreglos de más de una dimensión. En C, podemos declararlos utilizando un subíndice para cada dimensión.

Supongamos que queremos guardar la temperatura de todos los dias de un año; podemos usar una matriz de 12 filas (meses) y 31 columnas (días).

```
float temperatura[12][31]
```

Por convención el primer subíndice representa las filas, y el segundo subíndice las columnas

Si queremos acceder a la temperatura del 10 de marzo, accedemos:

```
temperatura[2][9]
```

- También podemos inicializar un arreglo multidimensional de la forma: `Int x [2][3]; x={{1,2,3},{4,5,6}}`

Ej. Imprima los elementos de una matriz de 2x3 elementos

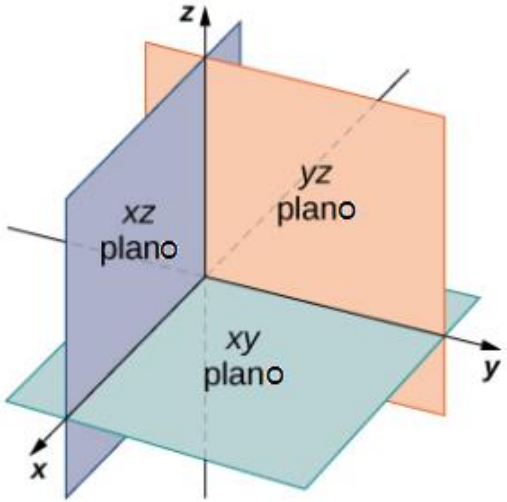
```
int arr[2][3]={{1,2,3},{4,5,6}};

for (int x=0; x<2;x++){
   for (int y=0; y<3;y++)
    cout<< arr[x][y]<<endl;
}
```

1	2	3
4	5	6

Matrices, arrays multidimensionales

```
int matriz1[5][3]; //matriz de cinco filas y tres columnas de
                   números enteros.
int arraydematrices[7][5][3];  //array de 7 matrices de las
                               anteriores
```

- El plano que contiene los ejes *x* e *y* se llama **plano *xy***.
- El plano que contiene los ejes *x* y *z* se llama **plano *xz***,
- y los ejes *y* y *z* definen el **plano *yz*.**



Matrices, arrays multidimensionales

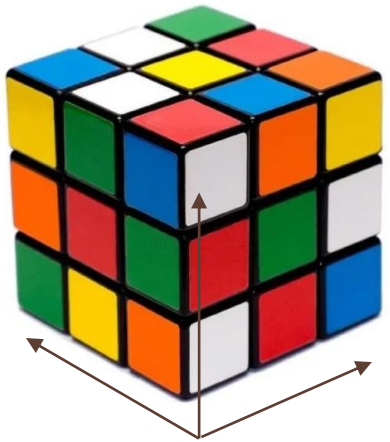
Es posible realizar una generalización para declarar y manipular arreglos *n-dimensionales* o de *n* dimensiones ya que:

Un **arreglo** es una sucesión contigüa de variables de un mismo tipo de datos.
•Un **arreglo de dos dimensiones** o **matriz** es un arreglo de arreglos.

•Un **arreglo de tres dimensiones** o **cubo** (si son iguales) es un arreglo de arreglos de dos dimensiones, o un arreglo de matrices.

•Un **arreglo de 4 dimensiones** o **hipercubo** (si son iguales) es un arreglo de arreglos de tres dimensiones, o un arreglo de cubos para el caso de los hipercubos.

•Un arreglo de ***n*** **dimensiones** es un arreglo de arreglos de *n-1* dimensiones.



Matrices, arrays multidimensionales

En general, la declaración de un arreglo de *n* dimensiones tiene la siguiente estructura en el lenguaje de programación C:

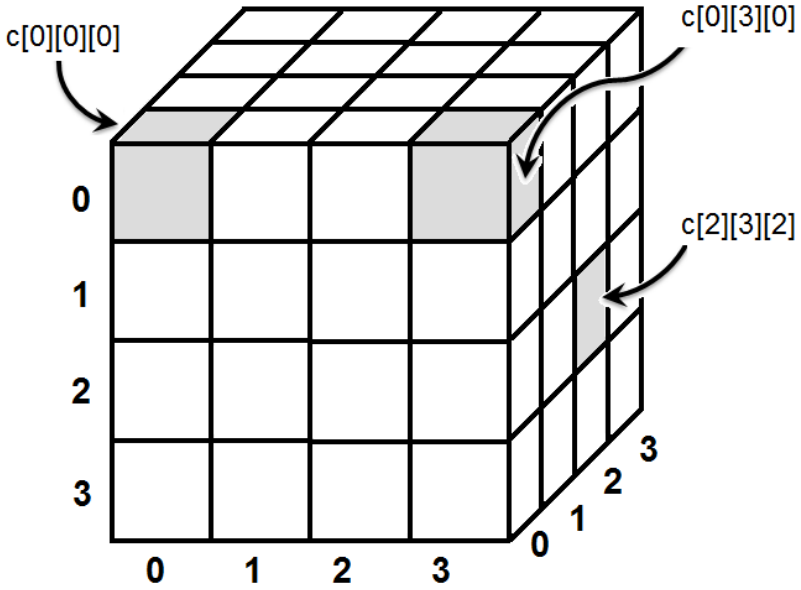
```
tipo_de_dato    arreglo_n_dimensional[TAMAÑO1][TAMAÑO2]...[TAMAÑO_N];
```

- **`tipo_de_dato`** es cualquier tipo de dato válido en C.
- **`arreglo_n_dimensional`** un identificador válido en C.

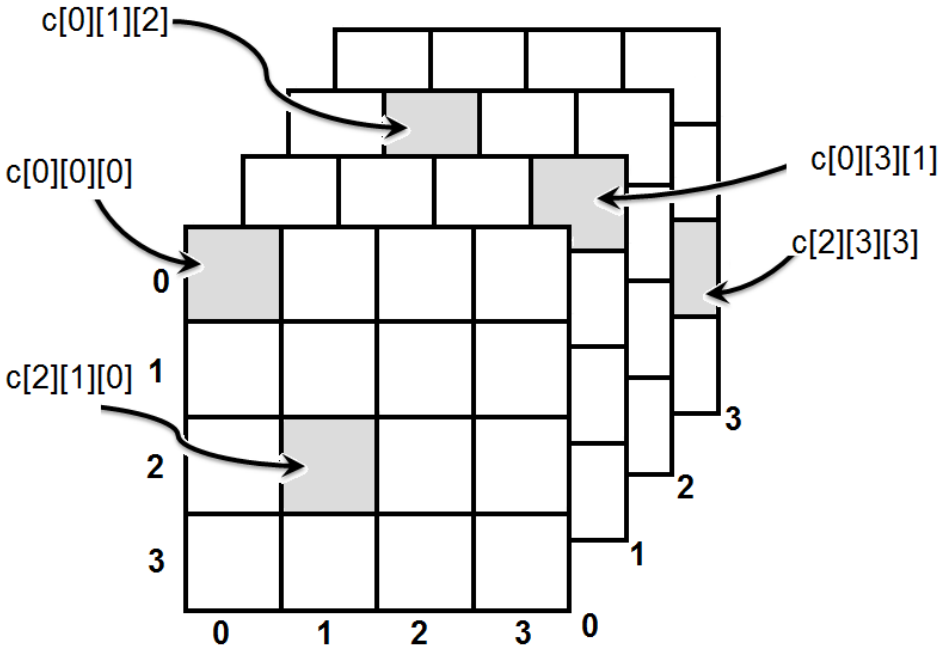
- **`TAMAÑO1`** es la longitud para la primera dimensión.
- **`TAMAÑO2`** es la longitud para la segunda dimensión.
- **`TAMAÑO_N`** es la longitud para la *n-ésima* dimensión.

Matrices, arrays multidimensionales

Las siguientes figuras muestran 2 posibles representaciones para un arreglo *c* de tres dimensiones de 4 x 4 x 4. Asegúrese de entender la ubicación de los elementos del arreglo tridimensional en ambas representaciones.



(a) Representación de un arreglo de tres dimensiones en forma de cubo.



(b) Representación de un arreglo de tres dimensiones en forma de malla.