

# ***CFGM SISTEMAS MICROINFORMÁTICOS Y REDES***

**Sistemas Operativos Monopuesto**

**TEMA 04: Fundamentos de Programación II**

**Arrays**

# Contenido

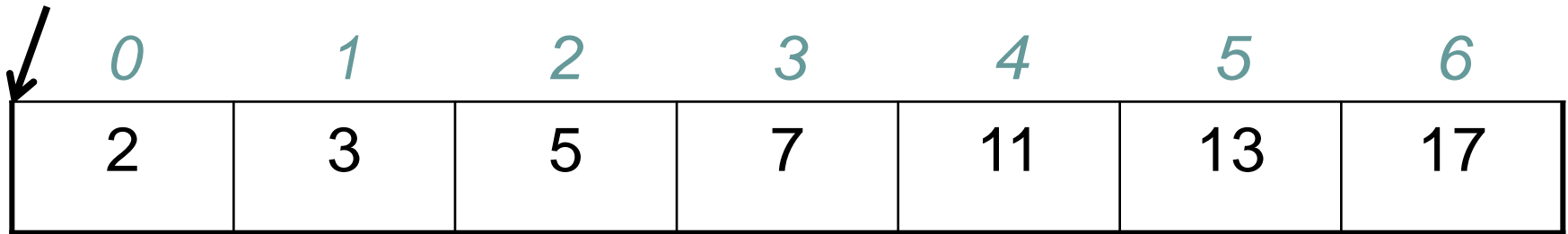
- Introducción. Concepto de array
- Declaración e inicialización de arrays
- Acceso a elementos
- Esquema de recorrido
- Esquema de búsqueda
- Ejercicios

# Concepto de array

- Un array unidimensional o vector es una secuencia finita de datos del mismo tipo.
- Los datos se llaman elementos y se enumeran consecutivamente 0, 1, 2, ...
- Es conveniente un vector para almacenar (p. ej.):
  - Las alturas de los alumnos de una clase.
  - Las temperaturas de cada día del mes de Enero.
  - Los 7 primeros números primos.
  - Los goles marcados por los equipos de primera división.
  - Los días de cada uno de los meses del año.

# Concepto de array. Acceso a elementos

primos



0	1	2	3	4	5	6
2	3	5	7	11	13	17

**¿Qué imprimen por pantalla las siguientes sentencias?**

```
printf("%d", primos[0]);
```

```
printf("%d", primos[4]);
```

```
printf("%d", primos[7]);
```

```
int i = 3; printf("%d %d", primos[i], primos[i+2]);
```

# Declaración de arrays

- La declaración reserva la memoria necesaria para el vector.

*tipo\_básico nombre\_vector[num\_elementos];*

- Ejemplos:

```
float alturas[28];  
double temperaturas[31];  
int primos[15];  
int goles[20];
```

- Es recomendable definir una constante con el número de elementos del vector:

```
#define NUM_ALUMNOS 28  
...  
float alturas[NUM_ALUMNOS];
```

# Inicialización de vectores

- La inicialización asigna valores a los elementos del vector.
- Frecuentemente se inicializa a 0 todos los elementos del vector.

```
int goles[20];  
int i;  
for (i = 0 ; i < 20 ; i++)  
    goles[i] = 0;
```

# Inicialización de vectores (II)

- Se puede utilizar una constante de inicialización para dar valores a los elementos de un vector en el propio programa. En ese caso no es necesario incorporar el número de elementos del vector en la declaración.

```
float primos[] = {2,3,5,7,11,13,17};
```

# Inicialización de vectores (III)

- También podemos utilizar `scanf` repetidamente para inicializar un vector con los valores que vayamos introduciendo por teclado.

```
#define NUM_ALUMNOS 28
...
float alturas[NUM_ALUMNOS];
int i;
for (i = 0 ; i < NUM_ALUMNOS ; i++) {
    printf("Introduzca la altura del alumno número %d\n",i);
    scanf("%f",&(alturas[i]));
}
```



# Acceso a los elementos de un vector

- Para acceder a los elementos de un vector utilizamos un índice, que siempre es un número entero entre 0 y el número de elementos del vector menos uno.

```
#define MAX 23  
...  
int v[MAX];  
v[3] = 4;  
v[4] = 13;  
printf("%d\n",v[3]);  
printf("%d\n",v[3]+v[4]);  
printf("%d\n",v[v[3]]);
```

# Esquema de recorrido

- Una de las tareas comunes a realizar con un vector es el **recorrido**. Éste supone pasar por todos y cada uno de los elementos del vector. Hemos visto ejemplos (inicializar a 0, leer desde teclado).

```
tipo nombre_vector[CTTE_MAX];  
int i;  
for (i = 0 ; i < CTTE_MAX ; i++) {  
    // Realizar tratamiento con  
    //nombre_vector[i]  
}
```

# Ejemplo 1

Programa que lee un vector de 5 enteros. Una vez ha leído el vector suma sus 5 elementos y nos muestra la suma.

## Ejemplo 2

Programa que inicializa un vector con los 7 primeros números primos y calcula y muestra el producto de éstos.

# Esquema de búsqueda

```
inicializar_secuencia;  
  
int encontrado = 0;  
while (!encontrado && !final_de_secuencia)  
{  
  
    if (cumple_condicion_busqueda)  
        encontrado = 1;  
    pasar_a_siguiente;  
}  
if (encontrado)  
    tratar_exito;  
else  
    tratar_fracaso;
```

**GENÉRICO**

```
int i = 0;  
int x;  
int encontrado = 0;  
while (!encontrado && (i < CTTE_MAX)) {  
    x = v[i];  
    if (x cumple_condicion_busqueda)  
        encontrado = 1;  
    i++;  
}  
if (encontrado)  
    tratar_exito;  
else  
    tratar_fracaso;
```

**SOBRE UN VECTOR**

# Ejemplo

Programa que lee un vector de 5 elementos y muestra el primer elemento del vector cuyo valor es menor que 10 ó dice que no hay ningún elemento menor que 10 en el vector.

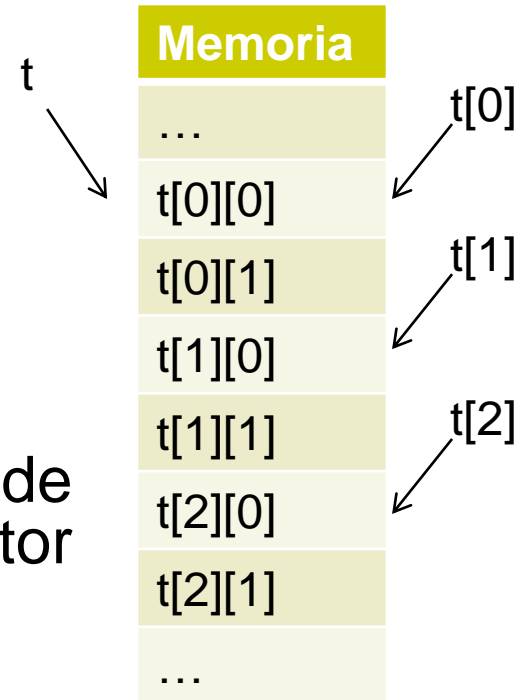
# Tablas multidimensionales

- En C podemos definir tablas de más de una dimensión:

```
int t[3][2];  
t[1][1] = 23;
```

- Debemos entender t como:

Un vector de 3 elementos y cada uno de estos elementos es a su vez un vector de 2 elementos



# Ejemplo

- La traza de una matriz cuadrada (mismo número de filas que de columnas) es la suma de los elementos de su diagonal. Crear un programa que:
  1. Defina una constante para el tamaño máximo de la matriz de 100.
  2. Lea el tamaño de la matriz teniendo en cuenta que si el usuario introduce un tamaño negativo o superior al máximo el programa debe volver a pedirle que introduzca el tamaño.
  3. Lea la matriz desde el teclado (llamando a un procedimiento *leer\_matriz* que deberemos implementar).
  4. Calcule la traza de la matriz (llamando a una función *traza* que deberemos implementar).