

ARREGLOS

Vectores y matrices

Vectores - Declaración

□ Sintaxis

tipo_base nombre[cant_de_elementos]

□ Ejemplos

int numeros[20];



cantidad de elementos
del vector

int datos[50], cantidades[22];

- El primer elemento tiene índice 0.
- Puede declararse más de un vector en una misma línea.

Inicialización

- Los elementos de un arreglo no se inicializan por defecto.
- Es posible asignarles valor al momento de su declaración indicando una lista de valores encerrados entre llaves y separados por comas.
- **Ejemplo**

```
int vector[5] = {12, 34, 7, 2, 89};
```

Inicialización

- Si la cantidad de valores es inferior al tamaño del vector, el resto se inicializa con cero.

- **Ejemplo**

`int PocosValores[7] = {2, 3};`

`int TodoNulo[100] = {0};` 

Al menos debe inicializarse con cero el primer elemento para que todo el vector quede en cero

Inicialización

- Si la cantidad de valores excede el tamaño del vector se producirá un error en compilación.
- Si se omite el tamaño del vector se utilizará la cantidad de elementos para obtener su dimensión.

- **Ejemplos**

Da error de compilación

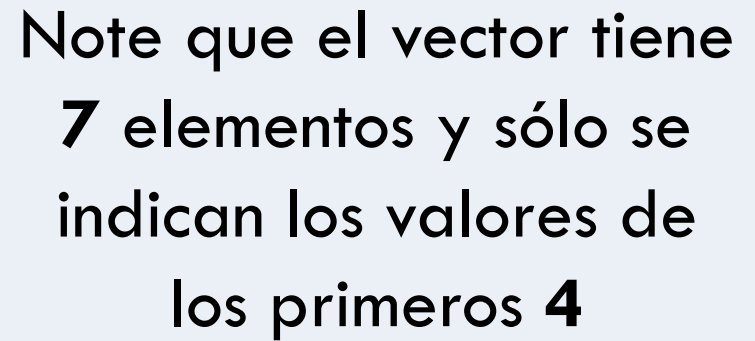
`int MuchosValores[3] = {1, 2, 3, 4, 5, 6, 7, 8};`

`int SinDimension[] = {23, 12, 67, 22};`

Crea un arreglo de 4 elementos

Qué imprime?

Note que el vector tiene
7 elementos y sólo se
indican los valores de
los primeros 4



```
#include <stdio.h>
int main()
{
    int nros[7]= {5, 3, 20000, 8};
    int i;
    printf("Indice      Valor\n");
    for (i=0; i<7; i++) {
        printf(" i= %d      ", i);
        printf("nros[%d]= %d\n", i, nros[i]);
    }
    return 0;
}
```

Definición de valores constantes

```
#include <stdio.h>
int main()
{
    int nros[7] = {5, 3, 20000, 8};
    int i;
    printf("Indice      Valor\n");
    for (i=0; i<7; i++) {
        printf("  i= %d      ", i);
        printf("nros[%d]= %d\n", i, nros[i]);
    }
    return 0;
}
```

Sería conveniente definir este valor de manera constante, no?

Directiva **#define**

- La directiva de preprocesador **#define** permite definir *constantes simbólicas*.
- Por ejemplo, la directiva

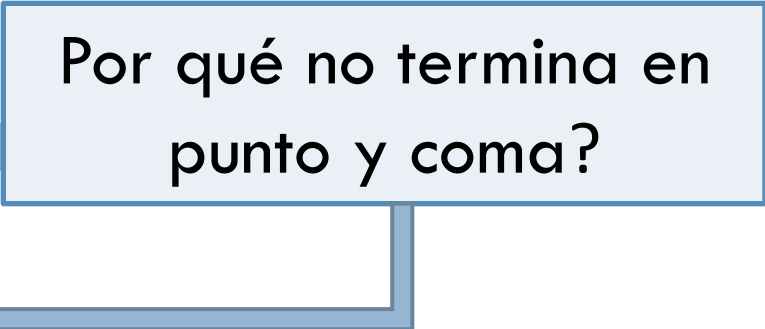
#define CUANTOS 7

define la constante simbólica **CUANTOS** cuyo valor es **7**.

- Antes de compilar todas las apariciones de **CUANTOS** son reemplazadas por el preprocesador con el texto de reemplazo **7**

Directiva #define

Por qué no termina en
punto y coma?



```
#include <stdio.h>
#define CUANTOS 7
int main()
{
    int nros[CUANTOS]= {5, 3, 20000, 8};
    int i;
    printf("Indice      Valor\n");
    for (i=0; i<CUANTOS; i++) {
        printf(" i= %d      ", i);
        printf("nros[%d]= %d\n", i, nros[i]);
    }
    return 0;
}
```

Palabra reservada **const**

- Este calificador permite indicar que el valor de una variable no puede ser modificado

- **Ejemplo**

`const int duracion = 120;`

- En cierta forma simula una constante.
- El valor de la variable NO puede aparecer a la izquierda de una asignación (salvo en su declaración donde recibe valor).

```
/* Palabra reservada const */  
#include <stdio.h>  
int main()  
{   int cambiante = 5;  
    float Sueldo;  
  
    const int ValorFijo = 23;  
  
    const SinTipo = 4;  
  
    const int Vacio;  
  
    const float ConDecimales = 3.14;  
  
    printf("%d  %d  %d  %d\n",  
           cambiante, ValorFijo, Vacio, SinTipo);  
  
    Sueldo = 4563.23;  
    printf("%4.2f  %5.1f", ConDecimales, Sueldo);  
  
    return 0;  
}
```

Estas variables pueden recibir valor en su declaración o dentro del programa

```
/* Palabra reservada const */
```

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

```
int main()
```

```
{   int cambiante = 5;
```

```
    float Sueldo;
```

Estas variables pueden recibir valor SOLO en su declaración

```
    const int ValorFijo = 23;
```

```
    const SinTipo = 4; ←
```

Si falta el tipo se asume int

```
    const int Vacio; ←
```

Nunca podrá tomar un valor

```
    const float ConDecimales = 3.14;
```

```
    printf("%d %d %d %d\n",  
           cambiante, ValorFijo, Vacio, SinTipo);
```

```
    Sueldo = 4563.23;
```

```
    printf("%4.2f %5.1f", ConDecimales, Sueldo);
```

```
    return 0;
```

```
}
```

#define y const

- Note que **#define** no es lo mismo que el calificador **const**.
- **#define** es una directiva para el precompilador que reemplaza el identificador por el texto correspondiente ANTES de compilar.
- La palabra clave **const** evita que el nombre de la variable se modifique en su alcance. Este chequeo se hace en compilación.
- Con **#define** NO se define una variable.

Ejercicio

- Se dispone de un mazo de 50 cartas españolas (4 palos de 12 cartas c/u y dos comodines)
- Se extraen las cartas de a una hasta completar los 4 palos de un mismo número.
- Indique cuál es ese número.

- En la definición del arreglo
 - ▣ utilice **#define**
 - ▣ puede utilizarse el identificador **const**? Justifique

Arreglos y el especificador **static**

- Cuando se utilizan arreglos como variables locales a funciones puede resultar de utilidad declararlos de manera estática (*static*) para que no sean creados e inicializados cada vez que se llame a la función.

static int valores[30];

- Los arreglos que se declaran *static* se inicializan automáticamente una única vez en compilación. Sino se le asigna valor inicial será inicializado con cero.

```
#include <stdio.h>
void ArregloStatic(void);
int main()
{
    printf("ArregloStatic - 1ra. vez\n");
    ArregloStatic();

    printf("\n\n\nArregloStatic - 2da. vez\n");
    ArregloStatic();
    return 0;
}

void ArregloStatic(void)
{
    static int nros[3];
    int i;

    printf("Valores al entrar\n");
    for (i=0; i<3; i++)
        printf("nros[%d]=%d  ", i, nros[i]);

    printf("\n\nSumamos 5 a c/u y salimos\n");
    for (i=0; i<3; i++)
        printf("nros[%d]=%d  ", i, nros[i]+=5);
}
```

Qué imprime?

ArregloDinamico.c

```
#include <stdio.h>
void ArregloDinamico(void);
int main()
{   printf("ArregloDinamico - 1ra. vez\n");
    ArregloDinamico();

    printf("\n\n\nArregloDinamico - 2da. vez\n");
    ArregloDinamico();
    return 0;
}
```

Qué imprime?

```
void ArregloDinamico(void)
{   int nros[3] = {1,2,3};
    int i;

    printf("Valores al entrar\n");
    for (i=0; i<3; i++)
        printf("nros[%d]=%d  ", i, nros[i]);

    printf("\n\nSumamos 5 a c/u y salimos\n");
    for (i=0; i<3; i++)
        printf("nros[%d]=%d  ", i, nros[i]+=5);
}
```

Arreglos como parámetros

- Para pasar un arreglo como parámetro a una función, especifique sólo el nombre del arreglo.
- Ejemplo: Dado el arreglo

int precios[200];

puede ser pasado a la función de la siguiente forma

modificar(precios, 200);

El nombre del arreglo es la dirección a su primer elemento y por lo tanto, la función lo puede modificar!

```
#include <stdio.h>

void modificar(float [], int );
void mostrar(const float [], int );
#define SIZE 5

int main()
{
    float precios[SIZE] = {350, 110, 300, 210, 200};
    mostrar(precios, SIZE);
    modificar(precios, SIZE);
    mostrar(precios, SIZE);
    return 0;
}

void modificar(float V[], int cant)
{
    int i;
    for (i=0; i<cant; i++)
        V[i] = (V[i]>200? 0.8*V[i] : V[i]);
}

void mostrar(const float V[], int cant)
{
    int i;
    for (i=0; i<cant; i++)
        printf("  v[%d] = %5.1f", i, V[i]);
    printf("\n")    ;
}
```

```
#include <stdio.h>
void modificar(float [], int );
void mostrar(const float [], int );
#define SIZE 5
int main()
{
    float precios[SIZE] = {350, 110, 300, 210, 200};
    mostrar(precios, SIZE);
    modificar(precios, SIZE);
    mostrar(precios, SIZE);
    return 0;
}
```

```
void modificar(float V[], int cant)
{
    int i;
    for (i=0; i<cant; i++)
        V[i] = (V[i]>200? 0.8*V[i] : V[i]);
}
```

La función **modificar** puede alterar los valores del vector V. Estos cambios se verán reflejados sobre el vector precios al volver al programa principal.

```
#include <stdio.h>
void modificar(float [], int );
void mostrar(const float [], int );
#define SIZE 5
int main()
{
    float precios[SIZE] = {350, 110, 300, 210, 200};
    mostrar(precios, SIZE);
    modificar(precios, SIZE);
    mostrar(precios, SIZE);
    return 0;
}
```

Para asegurarse de que la función no modifique los valores del vector puede utilizar el calificador **const** delante de la declaración del parámetro.

```
void mostrar(const float V[], int cant)
{
    int i;
    for (i=0; i<cant; i++)
        printf("  v[%d] = %5.1f", i, V[i]);
    printf("\n")    ;
}
```

Salida del programa anterior

```
v[0] = 350.0  v[1] = 110.0  v[2] = 300.0  v[3] = 210.0  v[4] = 200.0  
v[0] = 280.0  v[1] = 110.0  v[2] = 240.0  v[3] = 168.0  v[4] = 200.0
```

- Puede verse que la función modificar puede cambiar los valores del vector.

ArregloConst.c

Ejercicio

- Escriba una función que reciba un vector de números y devuelva el promedio de sus valores.

Arreglos con múltiples subíndices

- En ANSI C los arreglos pueden tener como mínimo hasta 12 índices.

- Ejemplos:

/* Crea 3 matrices de 2 filas x 3 columnas */

```
int matriz1[2][3] = {{1,2,3}, {4,5,6}},
```

```
matriz2[2][3] = {1,2,4,6},
```

```
matriz3[2][3] = {{1,2}, {6}};
```

Watches

matriz1

1

2

3

4

5

6

matriz2

1

2

4

6

0

0

matriz3

1

2

0

6

0

0

MiniMatriz.c

```
#include <stdio.h>
#define COL 2

int mini(int[][COL], int);

int main()
{
    int M1[2][COL]={{10,2},{4,1}};
    int M2[4][COL]={{3,4},{12,2},{76,1}};


    printf("El menor valor de M1 es %d\n", mini(M1,2));
    printf("El menor valor de M2 es %d\n", mini(M2,4));

    return 0;
}

int mini(int M[][COL], int f)
{
    int i,j, menor=M[0][0];

    for(i=0; i<f; i++) /* recorre las filas */
        for (j=0; j<COL; j++) /*recorre las columnas */
            if (menor>M[i][j])
                menor = M[i][j];

    return (menor);
}
```



Si se trata de una matriz,
sólo se puede omitir la
cantidad de valores que
toma el 1er. índice

Ejercicio

- El dueño de un restaurante entrevista a 5 clientes de su negocio y les pide que califiquen de 1 a 10 los siguientes aspectos (1 es pésimo y 10 es excelente)
 - ▣ Atención de parte de los empleados
 - ▣ Calidad de la comida
 - ▣ Justicia del precio (el precio que pagó le parece justo?)
 - ▣ Ambiente (muebles cómodos?, música adecuada?, iluminación suficiente?, decoración, etc.)
- Escriba un algoritmo que pida las calificaciones de los 5 clientes para c/u de estos aspectos y luego escriba el promedio obtenido en cada uno de ellos.

Encuentre el error y corríjalo

```
#define SIZE 100;  
int muchos[SIZE];
```

```
int pares[4]={2,4,6,8,10,12};  
pares[1]=3;
```

Encuentre el error y corríjalo

```
int b[10]={0}, i;  
for (i=1; i<=10; i++) b[i]=i;
```

```
const int j, i, suma;  
for (i=1, suma=0; i<10; suma+=i++);
```

Encuentre el error y corríjalo

```
int matriz[3][2] = {{3,2,3}, {7,11,17}};  
matriz[3][2]=2;
```

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