Arrays

Disclaimer

Los códigos fuente utilizados para la explicación del funcionamiento del “stack” son solo a fines didácticos y no representa una óptima solución al problema a resolver.

Variables

- Primitive Types in C
- Pointers
- Arrays

Tipos de datos (repaso)

- Enteros
  - char, int
- Decimales
  - float, double
- Modificadores
  - short [int]
  - long [int, double]
  - signed [char, int]
  - unsigned [char, int]

C Data Type	32-bit	64-bit	printf
char	1	1	%c
short int	2	2	%hd
unsigned short int	2	2	%hu
int	4	4	%d / %i
unsigned int	4	4	%u
long int	4	8	%ld
long long int	8	8	%lld
float	4	4	%f
double	8	8	%lf
long double	12	16	%Lf
pointer	4	8	%p

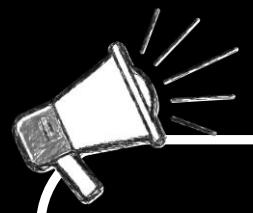
32 ó 64 bit

```
$ arch
$ echo $MACHTYPE
$ lscpu
```

¿ es posible resolver ?

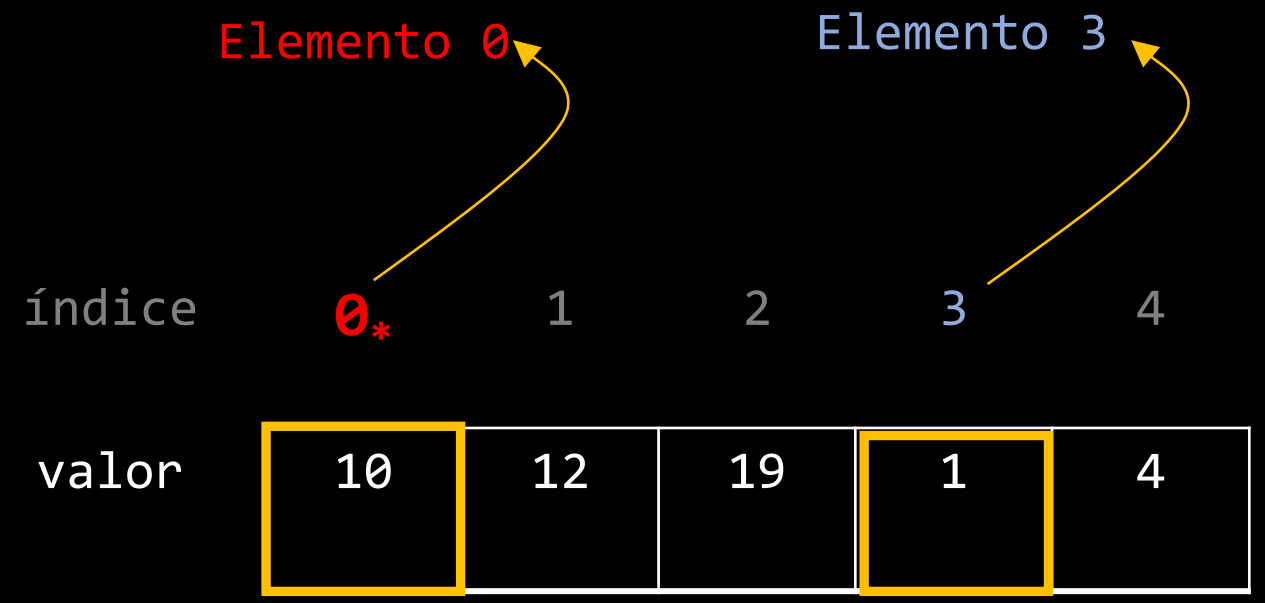
```
Ingrese 5 valores enteros:

Valor 1: 10
Valor 2: 12
Valor 3: 19
Valor 4: 1
Valor 5: 4

Promedio = 9.2

Valores superiores al promedio = 3
```

XX: entrada por teclado



array@definición

- Variable con un conjunto de datos del “**mismo tipo**”
- Se ubican en direcciones “**consecutivas de memoria**”



* índice “0-based int”

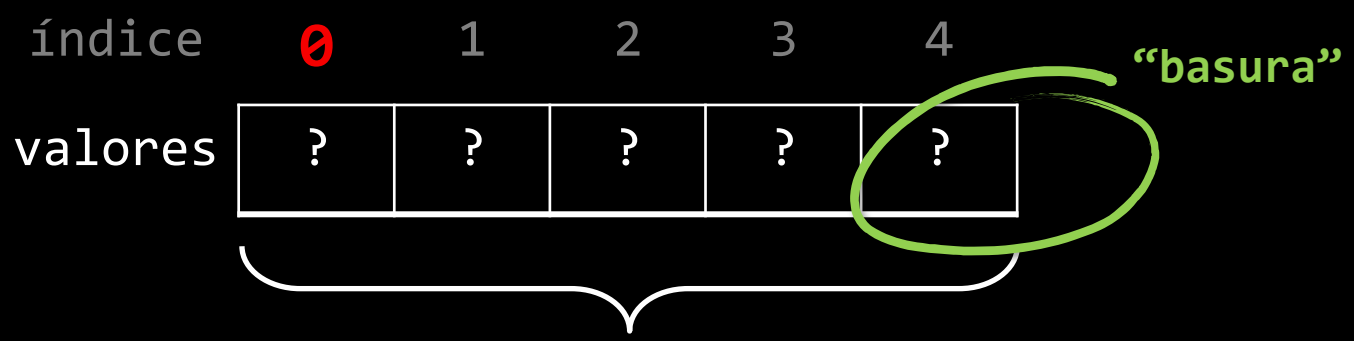
array@declaración

tipo nombre[tamaño];

- El tamaño en cantidad de elementos de “tipo”
- El tamaño se define en tiempo de compilación (*excepción C99, no recomendada)

- Ejemplo:

```
int valores[5];
```

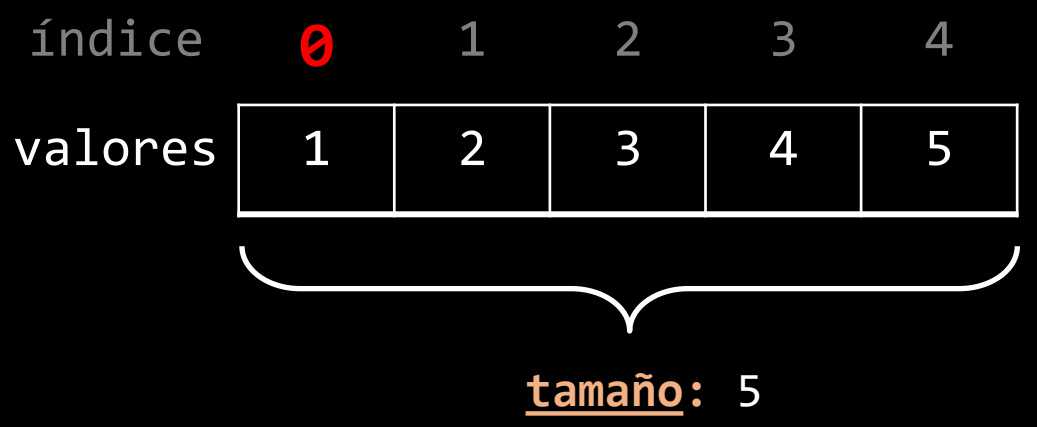


array@declaración e inicialización

tipo nombre[tamaño] = { valor, valor, …} ;

- Ejemplo:

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

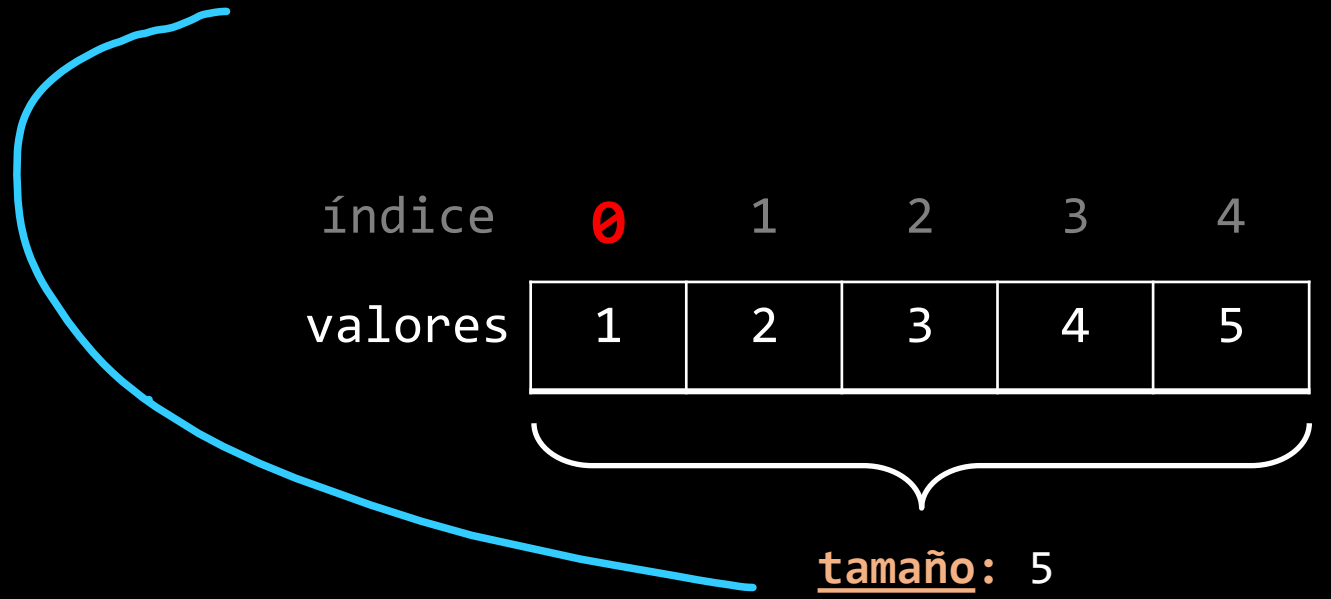


array@declaración e inicialización

tipo nombre[] = { valor, valor, …} ;

- Ejemplo:

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

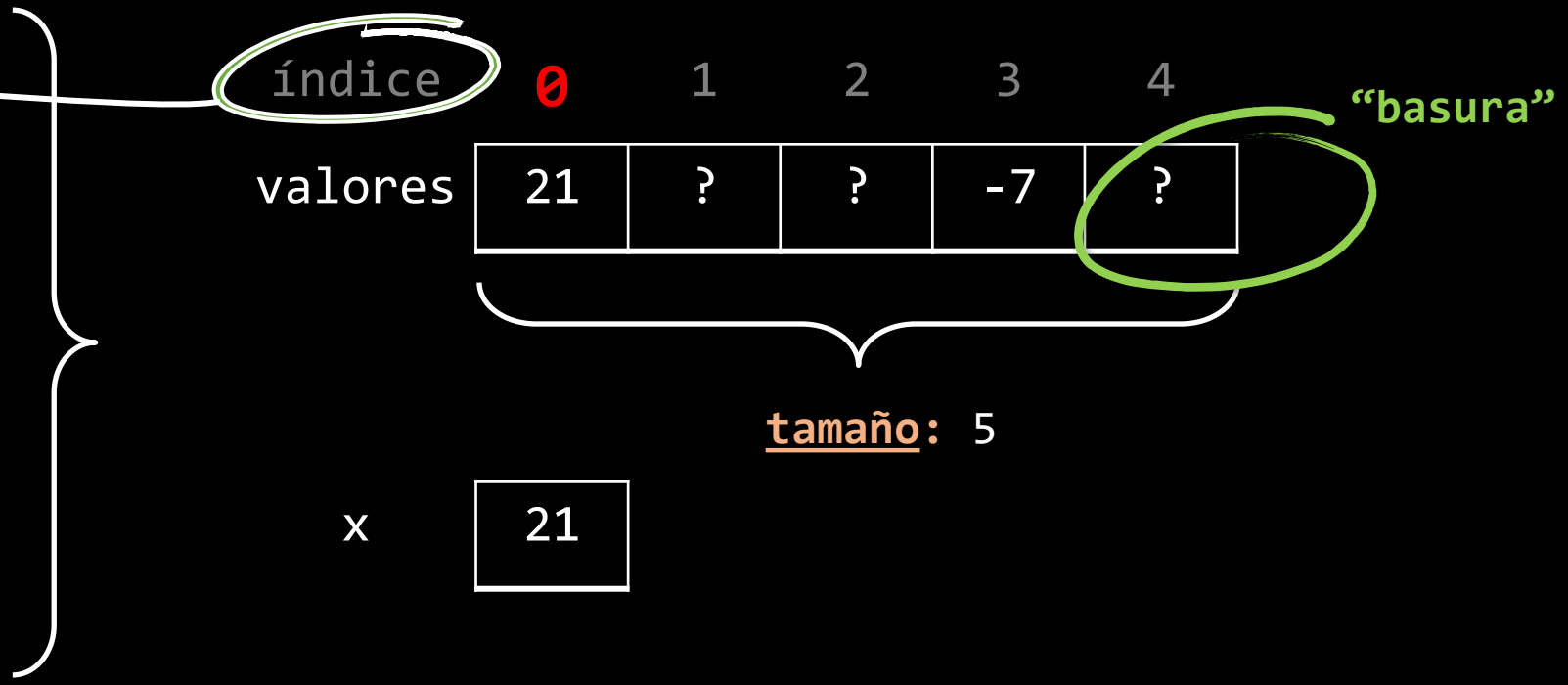


array@acceso a un elemento

- **nombre**[**indice**] // acceso
- **nombre**[**indice**] = **valor** // modificación

  - Ejemplo:

```
int x, valores[5];

valores[0] = 21;
valores[3] = -7;
x = valores[0];
```



array@limites

- Limites: entre **0** y el tamaño del array **- 1**.

  Leer o Escribir fuera de los limites es un error del programador, que no es verificado por el compilador(algunos compiladores emiten warning pero obviamente es una verificación estática)

- Ejemplo

```
int x, valores[5];

valores[1] = 21;   // Correcto
valores[5] = -7;   // Error, fuera de limite (0…4)
```

¿Qué sucede en este caso de error?

- Leer o Escribir en una posición de memoria no deseada
- “Segmentation fault”

completar… 5’ 

`int x[5]`

1	3	7	14	19

20

`int y[5]`

23	25	28	12	0

40

`int z[5]`

8	31	-10	14	98

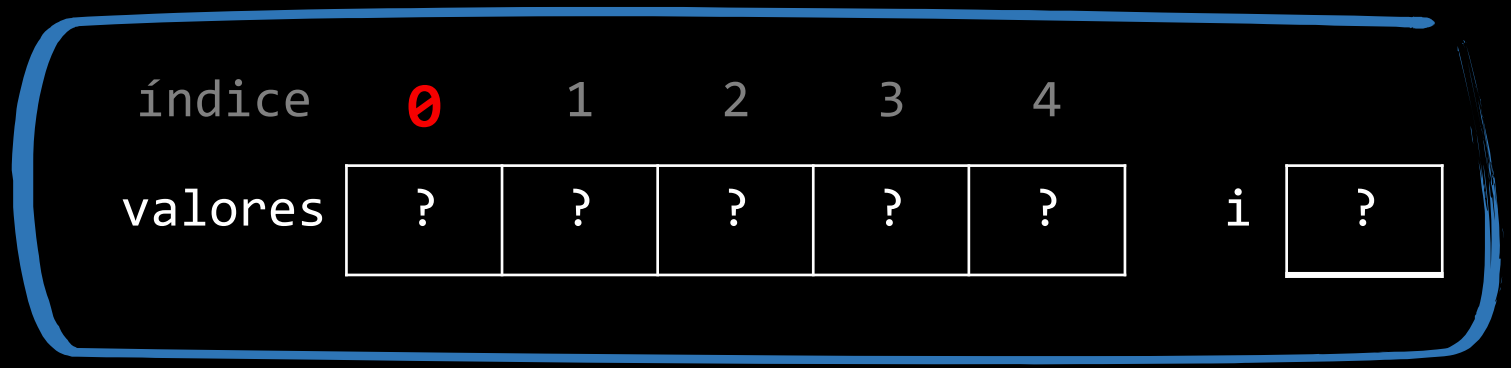
60

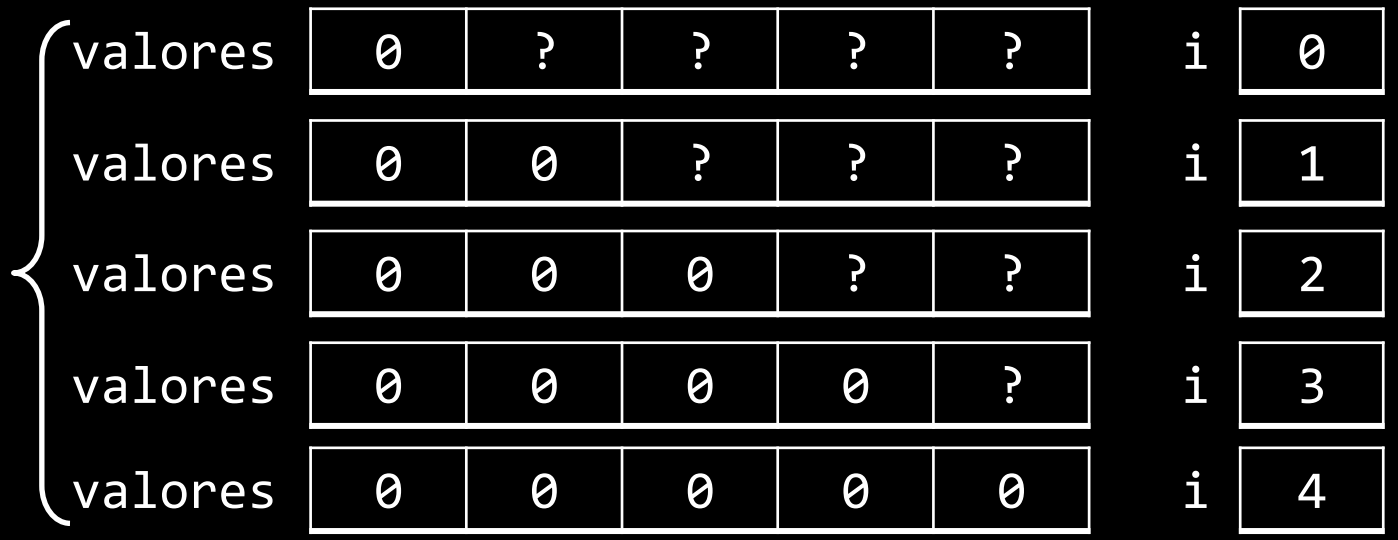
Referencia	Dirección	Valor
y[3]		
y[6]		
y[-1]		
x[15]		

array@inicializar

Buena práctica de programación el uso de #define para el tamaño del array

```
#define MAX 5

 int main(){
    int valores[MAX];
    int i;

   for(i=0;i<MAX;i++){
       valores[i] = 0;
   }
```

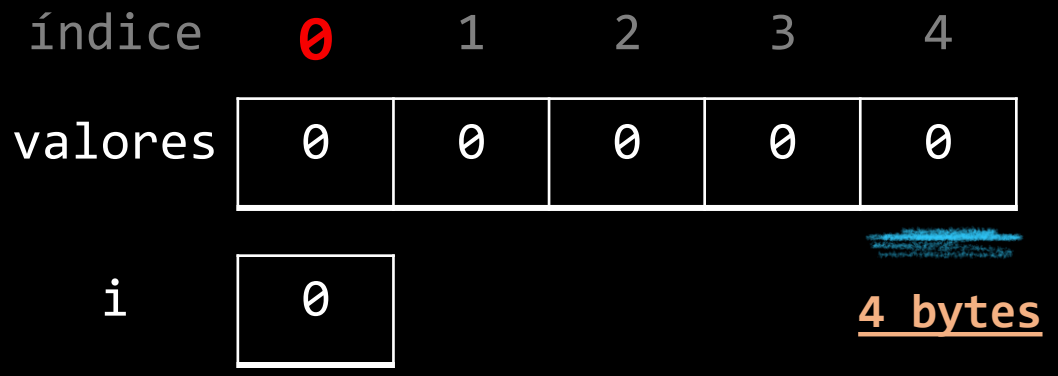




array@sizeof()

sizeof() valido únicamente dentro del scope del array

```c
#define MAX 5

 int main(){
    int valores[MAX];
    int i;

    for(i=0;i<MAX;i++){
        valores[i] = 0;
    }

    printf("Tamaño del array en bytes es: %ld\n", sizeof(valores));

    printf("Tamaño del array en elementos es: %ld\n", sizeof(valores)/sizeof(int));
```



a programar… 15’

```
Ingrese 5 valores enteros:

Valor 1: 10
Valor 2: 12
Valor 3: 19
Valor 4: 1
Valor 5: 4

Promedio = 9.2

Valores superiores al promedio = 3
```

XX: entrada por teclado