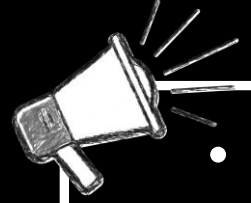
ARRAYS FUNCIONES

Oscar Paniagua @2023

array@funciones

- Pasar un SOLO elemento del array como REFERENCIA
- Pasar un SOLO elemento del array como VALOR y retornar un valor para asignar a un elemento del array
- Pasar un array COMPLETO a una función
- Pasar un array COMPLETO a una función, pero de solo LECTURA y documentar la función

array@funciones



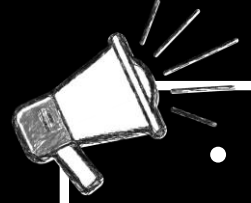
- Un array <u>únicamente</u> puede pasarse por referencia.

```
int arr[3];
init_int_array(arr);

void init_int_array(int *p){
    int i;

    for(i=0;i<sizeof(p);i++){
        p[i] = 0;
    }
}
```

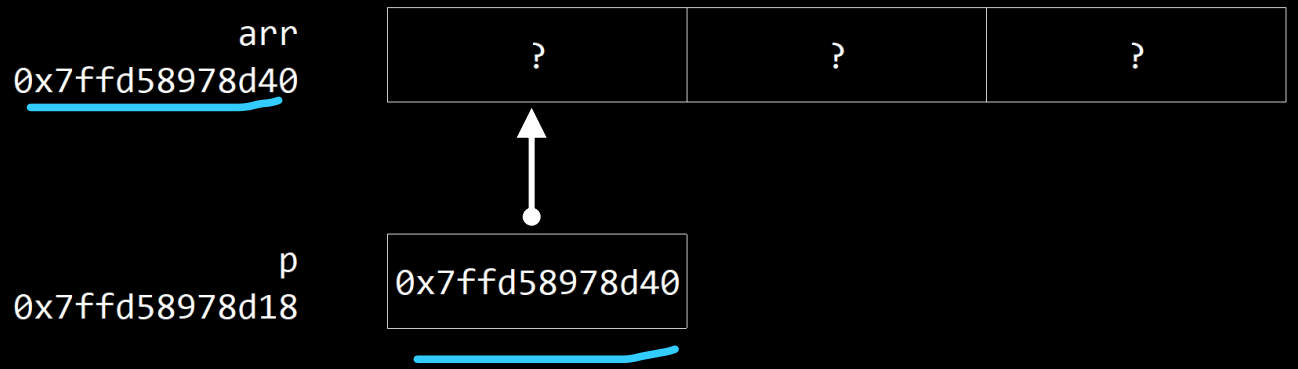
sizeof() valido únicamente dentro del scope del array

array@funciones

```c
// array
#define ARRAY_INT_SIZE  5

void init_int_array(int *p, int size);

 int main(){
    int arr[ARRAY_INT_SIZE];

    init_int_array(arr,ARRAY_INT_SIZE);
    return 0;
 }

 void init_int_array(int *p, int size){
     int i;
     for(i=0;i<size;i++){
         p[i] = 0;
     }
 }
```



- Un array <u>únicamente</u> puede pasarse por referencia.
- Debe indicarse el tamaño o algún valor que indique cual es el ultimo elemento



Array to Pointer

Podemos *decir, que el nombre del array, puede ser utilizado como un puntero al primer elemento del array (base address)*

En muchas situaciones, el compilador genera implícitamente un puntero al primer elemento del array con el nombre del array, por ejemplo, int arr[5], arr es un array de enteros, entonces el compilador genera un puntero int * si lo pasamos a una función.

arr -> int *

decay

En 3 excepciones no sucede:
1. sizeof(arr)
2. &arr
3. cuando el array es un string literal, ejemplo char msg[] = “hola”

Cuando pasamos el nombre del array 1D a una función, se convierte el nombre del array a un puntero al primer elemento

Conversión Array a Puntero

```
// el argumento de la función es un array de enteros,
// cuando se pasa el argumento automáticamente se ajusta
// (se lo suele llamar “decays”)
// de int [] (array de enteros) a int * (puntero a entero)

// array de entero
void arr_func(int a[])

// punter a entero
void arr_func(int *a)
```

int a[] es diferente a int *

```
cdecl> explain int arr[]
declare arr as array of int

cdecl> explain int *arr
declare arr as pointer to int
```

Resumen

- Un array se implementa en un espacio de memoria continua
- No se chequean limites
- No se inicializa (basura)
- Usualmente lo utilizamos como puntero al primer elemento(salvo las 3 excepciones)
- Con funciones solo por referencia (por diseño)
- Solo se puede utilizar `sizeof()` dentro del `scope` del propio array
- Si bien las ultimas versiones de C permiten arrays de tamaño dinámica(tiempo de ejecución), NO ES una buena practica.

A programar…**Operaciones con Arrays**

traverse	imprime todos los elementos, uno por uno
insert	agrega un elemento en la posición indicada
delete	elimina un elemento en la posición indicada
find	busca un elemento por la posición o por el valor
update	modifica un elemento en la posición indicada
cmp	compara dos arrays de la misma dimensión
copy	copia dos arrays de la misma dimensión
sum	suma dos arrays de la misma dimensión
init_rand	inicializa un array con valores random