

Sesión 05: *Arrays y Strings*

Programación para Sistemas

Ángel Herranz

Otoño 2018

Universidad Politécnica de Madrid

En capítulos anteriores. . .

Sesión 0: Presentación

Sesión 1: Contacto C

- Hay un error en las transparencias: gcc -l (la opción de linkado no existe, gcc linka por defecto)

Sesión 2: Ejecutando C

Sesión 3: Tipos básicos

Sesión 4: Módulos

En el capítulo de hoy...

- Vectores (*Arrays*)
- Cadenas de caracteres (*Strings*) = arrays de caracteres

Variables de tipo *array* i (longitud fija)

- Sintaxis i:


$$T \ a[N];$$

- Esa definición crea un espacio de **memoria contigua** para almacenar N elementos de tipo T ,
- **tan grande como lo que indican** N y **`sizeof(T)`**,
- elementos **accesibles usando la expresión**

$$a[i]$$

- donde i deberá estar **entre 0 y $N - 1$**

lcg3.c: modificar el programa lcg2.c

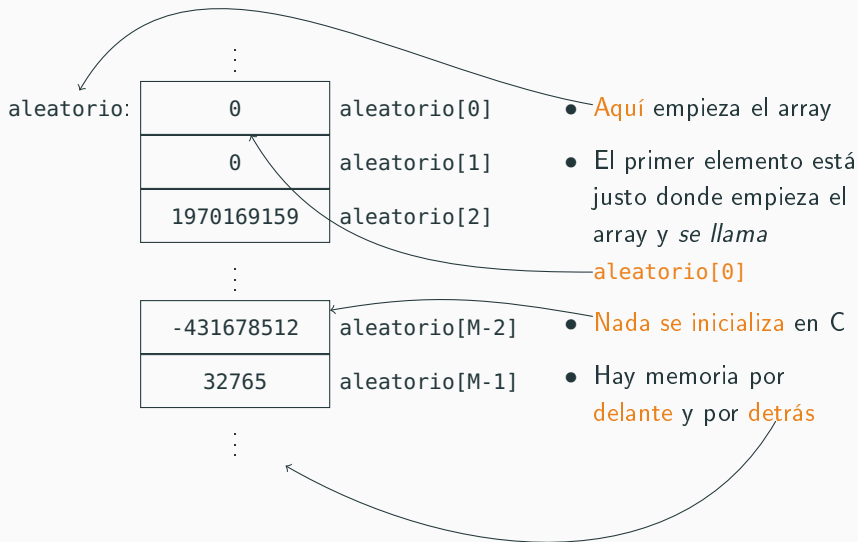
 Almacenar M^1 datos en un array y luego imprimirlos.

```
#include <stdio.h>
#include "generador_lcg.h"
int main() {
    int i;
    int aleatorio[M];
    for (i = 0; i < M; i++) {
        aleatorio[i] =
            generar_aleatorio();
    }
```

```
    for (i = 0; i < M; i++) {
        printf(
            "%i -> %i\n",
            i,
            aleatorio[i]);
    }
    return 0;
}
```

¹ $M = 11$ en las transparencias

Así están las cosas antes del primer for



¿Qué pasa si *me salgo* del array?


```
for (i = -M; i <= M; i++) {  
    printf(  
        "%i -> %i\n",  
        i,  
        aleatorio[i]);  
}
```

 ¡Explicar!

Al final de la ejecución ¿aleatorio[-1] == 12?


	⋮	
(?):	15775231	(aleatorio[-2])
i:	12	(aleatorio[-1])
aleatorio[0]:	0	
aleatorio[1]:	1	
aleatorio[2]:	8	
	⋮	
aleatorio[9]:	3	
aleatorio[10]:	0	
(?):	0	(aleatorio[11])
	⋮	

¿Longitud de un array?

 Escribir un programa que imprima la longitud de del array

²Veremos ejemplos

¿Longitud de un array?

 Escribir un programa que imprima la longitud de del array




sizeof

(tamaño en bytes de cualquier expresión)

²Veremos ejemplos

¿Longitud de un array?

 Escribir un programa que imprima la longitud de del array



sizeof

(tamaño en bytes de cualquier expresión)



Este recurso no es válido en general²

²Veremos ejemplos

¿Inmutabilidad de las variables array?

- Intentemos estas dos asignaciones:

```
int a[10];
```

```
int b[10];
```

```
b = a;
```

 ¿Qué nos dice el compilador?

¿Inmutabilidad de las variables array?

- Intentemos estas dos asignaciones:

```
int a[10];
```

```
int b[10];
```

```
b = a;
```

 ¿Qué nos dice el compilador?

```
$ make
```

```
cc -Wall -g -pedantic -o arrays arrays.c
```

```
arrays.c: In function 'main':
```

```
arrays.c:15:5: error: assignment to expression with array type
```

```
    b = a;
```

```
    ^
```

Variables de tipo *array* ii (inicializando)

- Sintaxis ii:

$$T \ a[] = \{ e_0, e_1, \dots, e_{n-1} \};$$

- donde la inicialización es obligatoria.
- Esa definición crea un espacio de **memoria contigua** para almacenar n elementos de tipo T ,
- **tan grande como lo que indican** n y **`sizeof(T)`**
- elementos **accesibles usando la expresión**

$$a[i]$$

- donde i deberá estar **entre 0 y $n - 1$**

¿Longitud de un array?

Escribir un programa

- Con una variable de tipo array declarada por inicialización con los números de Fibonnaci menores de 100
- Imprimir todos los elementos
- Imprimir la longitud

Variables de tipo *array* iii (argumentos)

- Sintaxis iii:


```
tipo_return funcion(tipo arg[]) {  
  
    . . .  
  
}
```

- Esa definición **no** crea un espacio de memoria contigua,
- simplemente se pasa como argumento la **dirección de memoria del primer elemento** del array
- De nuevo, los elementos son accesibles usando la sintaxis

arg[i]


- donde *i* deberá estar entre 0 y **la longitud del array - 1**

¿Longitud de un array?

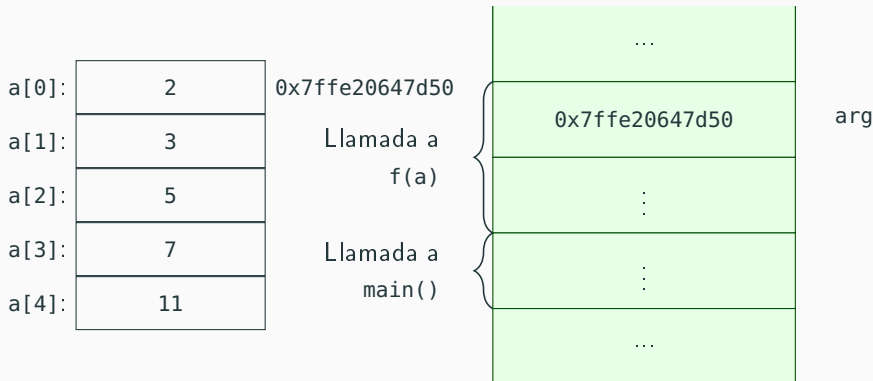
 Escribir una función que admita como argumento un array de enteros e imprima su longitud utilizando la técnica ya aprendida

 ¡Explicar!



¿Longitud de un array?

 Escribir una función que admita como argumento un array de enteros e imprima su longitud utilizando la técnica ya aprendida

 ¡Explicar!



Media del generador aleatorio i

-  Escribir un programa que imprima la media del generador de números aleatorios (generar $2 \cdot M$ datos) utilizando una función a la que se le pasa el array aleatorio
-  ¿Longitud del array?

Compliant Solution

- La solución al problema de no conocer la longitud de un array en C es sencilla:

Añadir un argumento con la longitud del array

```
#include <stddef.h> /* Para importar size_t */  
...  
tipo_return funcion(tipo arg[], size_t len) {  
    ...  
    for (i = 0; i < len; i++) {  
        ...arg[i]...  
    }  
}  
...  
...
```

size_t

- `size_t` es un tipo definido en las librerías estándares (ej. `#include <stdio.h>`)
- Se usa para `longitudes de arrays` y para `tamaño de datos`
- Internamente es un `unsigned`, probablemente `long`, pero no importa
- Usado en las `bibliotecas estándares`, por ejemplo:

```
$ man 3 strlen
```

```
STRLEN(3)          Linux Programmer's Manual          STRLEN(3)
```

```
NAME
```

```
    strlen - calculate the length of a string
```

```
SYNOPSIS
```

```
    #include <string.h>
```

```
    size_t strlen(const char *s);
```

```
...
```

Media del generador aleatorio ii

- 📄 Escribir un programa que imprima la media del generador de números aleatorios (generar $2 \cdot M$ datos) utilizando una función a la que se le pasa el array aleatorio y la longitud del array

Strings

- C **no tiene** tipo *String*
- Se usan **arrays de caracteres**³ para representar *strings*

 Transcribir el siguiente código

```
#include <stdio.h>

int main() {
    char s[] = "mundo";
    printf("El string es \"%s\"\n", s);
    printf("La longitud del array s es %lu\n",
           sizeof(s) / sizeof(s[0]));
    return 0;
}
```


³Es decir, arrays de enteros que caben en 1 byte

¿Longitud del array?

El string es "mundo".

La longitud del array s es 6

- ¿Otra vez con problemas con la longitud?


 Modifica el programa para que imprima el código ASCII de cada elemento del array

 ¿Encuentras alguna explicación?

NULL terminated

Convención


*las bibliotecas estándares de C asumen que los strings son **NULL terminated**, es decir, el string termina con el caracter '`\0`' (entero 0) (independientemente de la longitud del array)*

 ¿Qué implicaciones tiene dicha convención?

NULL terminated

Convención

*las bibliotecas estándares de C asumen que los strings son **NULL terminated**, es decir, el string termina con el caracter '**\0**' (entero 0) (independientemente de la longitud del array)*

 ¿Qué implicaciones tiene dicha convención?

- La **longitud del string está marcada** por la posición del caracter '**\0**'.
- La longitud del array tiene que tener un **hueco para el caracter '**\0**'**.

strings.h i

- Biblioteca para el manejo de strings: strlen, strcpy, etc.
- Puedes encontrar el API de strings.h en el *K&R*
- Usa el *manual*:

```
$ man 3 strcpy
```

```
STRCPY(3)          Linux Programmer's Manual          STRCPY(3)
NAME
```

```
strcpy, strncpy - copy a string
```





```
SYNOPSIS
```

```
#include <string.h>
char *strcpy(char *dest, const char *src);
char *strncpy(char *dest, const char *src, size_t n);
```

```
DESCRIPTION
```

```
The strcpy() function copies the string pointed to by
src, including the terminating null byte ('\0'), [...]
```

strings.h ii

-  Modifica el último programa para que imprima la longitud del string utilizando la función `strlen`.
-  Modifica el último programa para cambiar el caracter de terminación por otro (por ejemplo `'_'`) y luego pedir a `printf` que imprima el string
-  ¿Qué ocurre? ¿Puedes explicarlo?
-  ¿Qué diferencia hay entre estos dos strings?

```
char s6[] = "mundo";  
char s5[] = {'m', 'u', 'n', 'd', 'o'};
```

-  ¿Qué pasa cuando intentas imprimir los dos?

Arrays multidimensionales

- Sintaxis:

$$T \ m[N][M];$$

- Esa definición crea un espacio de **memoria contigua**,
- **tan grande** como para almacenar $N \times M$ datos del tipo T ,
- elementos **accesibles usando la expresión**

$$m[i][j];$$

- (elemento que ocupa la fila i y la columna j)
- donde i deberá estar **entre 0 y $N - 1$** y j **entre 0 y $M - 1$** .

Arrays multidimensionales: array de array

- Otra forma de ver un array multidimensional es entendiendo que la expresión

$m[i];$

- es un array de M elementos de tipo T .
- 💬 ¿Es posible pasarle a la función que calcula la media cada una de las filas de una matriz?