

Algoritmos y Estructuras de Datos II

TALLER - 23 de marzo 2021

Laboratorio 1: Arreglos, Archivos, Módulos

- 2021 Marco Rocchietti
- 2019 Gonzalo Peralta

Objetivos

1. Familiarizarse con vocabulario informático
2. Saber cómo compilar el programa
3. Tener manejo de las instrucciones básicas del lenguaje de programación C
4. Comenzar a manejar archivos como fuente de datos
5. Tener manejo de *standard input* y *standard output*.
6. Tener nociones del manejo de parámetros a través de la función principal `main()`
7. Trabajar con módulos en C

Ejercicio 1: Lectura de archivos

En la carpeta `ej1` vas a encontrar un archivo `main.c` y un directorio `input` que contiene varios archivos con *extensión* `.in`. Cada archivo tiene en su contenido un arreglo que ha sido guardado dentro de él. El arreglo (o *array*) se representa con su tamaño (*size*) y luego se enumera cada uno de los elementos separándolos con espacios. Por ejemplo, un *array* cuyos elementos son `[1,2,3,4,5]` es representado en el archivo como:

```
5
1 2 3 4 5
```

El archivo principal es `main.c`, donde vas a programar el ejercicio. Para compilarlo:

```
gcc -Wall -Werror -Wextra -pedantic -std=c99 -o reader main.c
```

Se pide que tu programa principal sea capaz de leer *arrays* de cualquiera de los archivos dentro del directorio `input` y que luego imprima por pantalla su contenido de la siguiente manera:

```
./reader input/example-easy.in
```

Se debe obtener la siguiente salida por pantalla:

```
[ 1, 2, 3, 4, 5]
```

Para ello deberás completar las definiciones de las funciones `array_from_file()` y la función `array_dump()`.

Te sugerimos que no esperes hasta el final para compilar, podés ir compilando a medida que completas alguna funcionalidad de tu programa.



Pueden serte útil las funciones `fopen()`, `fscanf()`, `fclose()`... se pueden consultar las páginas del manual de referencia de linux, ejecutando por ejemplo: "man fopen"

Ejercicio 2: Entrada Estándar

Modificar `main.c` (no borres el original) para que en lugar de leer un archivo de la carpeta `input`, lea el tamaño y cada uno de los miembros del `array` por teclado y luego los muestre por la pantalla.



Investiga sobre standard input: "man stdin"

Ejercicio 3: Módulos

Crea dos archivos nuevos: `array_helpers.h` y `array_helpers.c`. En el archivo `array_helpers.h` escribí los *prototipos* de las funciones `array_from_file()` y `array_dump()`. Luego, en el archivo `array_helpers.c` colocá las definiciones de las funciones sobre arreglos antes mencionadas e incluí los prototipos usando la directiva `#include "array_helpers.h"`. En `main.c` también tenés que incluir la librería `array_helpers.h`. Ahora compilá tu programa siguiendo los siguientes pasos:

```
gcc -Wall -Werror -Wextra -pedantic -std=c99 -c array_helpers.c
gcc -Wall -Werror -Wextra -pedantic -std=c99 -c main.c
```

y finalmente:

```
gcc -Wall -Werror -Wextra -pedantic -std=c99 array_helpers.o main.o -o reader
```

Ejercicio 4: Orden de elementos

Agrega la función `array_is_sorted()` a la librería `array_helpers` que tiene el siguiente prototipo:

```
bool array_is_sorted(int a[], unsigned int max_size);
```

que dado un `array` `a[]` y su tamaño `max_size` debe devolver `true` si y sólo si los elementos del arreglo `a[]` están ordenados de manera ascendente, es decir si:

$$a[0] \leq a[1] \leq \dots \leq a[\text{max_size}-1]$$

Cómo estamos utilizando el tipo `bool` el cual no es nativo del lenguaje C, recordá incluir la librería `stdbool.h`. Modificá `main.c` para que además de mostrarte el contenido del *array* leído desde el archivo especificado, también te informe si está ordenado o no. Un ejemplo sería:

```
./reader input/example-easy.in
[1, 2, 3, 4, 5]
El arreglo está ordenado
```

Otro ejemplo:

```
./reader input/example-unsorted.in
[2, -1, 3, 8, 0]
El arreglo no está ordenado
```

Ejercicio 5: Problemática de librerías

Aquí usaremos el mismo programa que construiste en el Ejercicio 4 pero en vez de usar el tipo `bool` de `stdbool.h` vamos a usar una definición casera de los *booleanos*. Como ya vimos en Algoritmos I, en C los enteros y los *booleanos* son muy parecidos. Podemos definir entonces en el archivo `boolean.h`:

```
typedef int boolean;
```

Recordá que `typedef` define sinónimos de tipos (como `type` en *Haskell*), por lo cual estamos diciendo que `boolean` es un sinónimo de `int` (son el mismo tipo). Ya que estamos, definimos las constantes `true` y `false`:

```
#define true 1
#define false 0

typedef int boolean;
```

En el archivo `test_boolean.c` podés ver ejemplos del uso de este tipo, allí vas a ver que es prácticamente lo mismo que el tipo `bool`. Para comparar además podés ver el archivo `test_bool.c` que utiliza `stdbool.h`.

Modifica `main.c`, `array_helpers.h` y `array_helpers.c` reemplazando al tipo `bool` por el tipo `boolean`. Acordate también de cambiar los `#include <stdbool.h>` por `#include "boolean.h"`.

Compilá tu programa cómo indica el ejercicio 3 *¿Por qué falla la compilación?* Tené en cuenta que cuando compilamos `test_boolean.c` todo funciona bien *¿Cómo se resuelve el problema?*

La solución tiene que ver con una modificación que debés hacer en **boolean.h**. Una vez resuelto el problema, hacer algo similar con **array_helpers.h** para evitar que pudiera generar el mismo problema.

Ejercicio 6*: *Bonus Track*

Usa como base el código que construiste para el Ejercicio 4 y agrega la función `array_swap()` a la librería **array_helpers** con el siguiente prototipo:

```
bool array_swap(int a[], unsigned int i, unsigned int j);
```

que dado un *array* `a[]` y su dos índices `i`, `j` debe intercambiar los valores de dichas posiciones.

Modificar **main.c** e invertir el *array* antes de mostrarlo por pantalla. El programa resultante debería comportarse de la siguiente manera:

```
./reader input/example-easy.in  
[5, 4, 3, 2, 1]  
El arreglo no está ordenado
```

Otro ejemplo:

```
./reader input/example-unsorted.in  
[0, 8, 3, -1, 2]  
El arreglo no está ordenado
```