|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Nombre de la práctica** | **Ejemplos y Ejercicios En lenguaje C** | | | | **No.** | | **….** |
| **Asignatura:** | **Métodos Numéricos** | **Carrera:** | **ISIC** | **Duración de la práctica (Hrs)** | | **5hrs** | |

**I.-DESARROLLO:**

**Implementando los ejemplos y ejercicios que vienen las laminas de la materia de Métodos números , programan dolos en Lenguaje C.**

**II.-Material empleado:**

1. **Lap top**
2. **PDF**
3. **Vscode**
4. **Linux**

**III.-Desarrollo de la practica :**

**APUNTADORES**

**¿Que es un apuntador?**

-Un apuntador es un objeto que apunta a otro objeto. Es decir, una variable cuyo valor es la dirección de memoria de otra variable.

En C no se debe indicar numéricamente la dirección de la memoria, si no que se usa una etiqueta que conocemos como **variable**. La dirección de memoria depende de la arquitectura del ordenador y de la gestión que el sistema operativo haga de ella.

**¿Como se declaran los apuntadores?**

Para declarar un apuntador se especifica el tipo de dato al que apuntar, el operador : “ \* “, y el nombre del apuntador.

* Un puntero tiene su propia dirección de memoria,
* La sintaxis es la siguiente:

<tipo de dato apuntador> \* < identificador del apuntador>

**int \* punt;**

**char \* car;**

**float \* num;**

Al igual que el resto de las variables, los apuntadores se enlazan a tipos de datos específicos, de manera que a un apuntador solo se le puede asignar dirección de viables del tipo especificado en la declaración.

**TIPOS DE APUNTADORES**

Hay tantos tipos de apuntadores como tipos de datos. Se pueden tambien declarar apuntadores a estructuras mas complejas.

1.-Funciones

2.-Struct

3.-Ficheros

Se pueden declarar punteros vacíos o nulos.

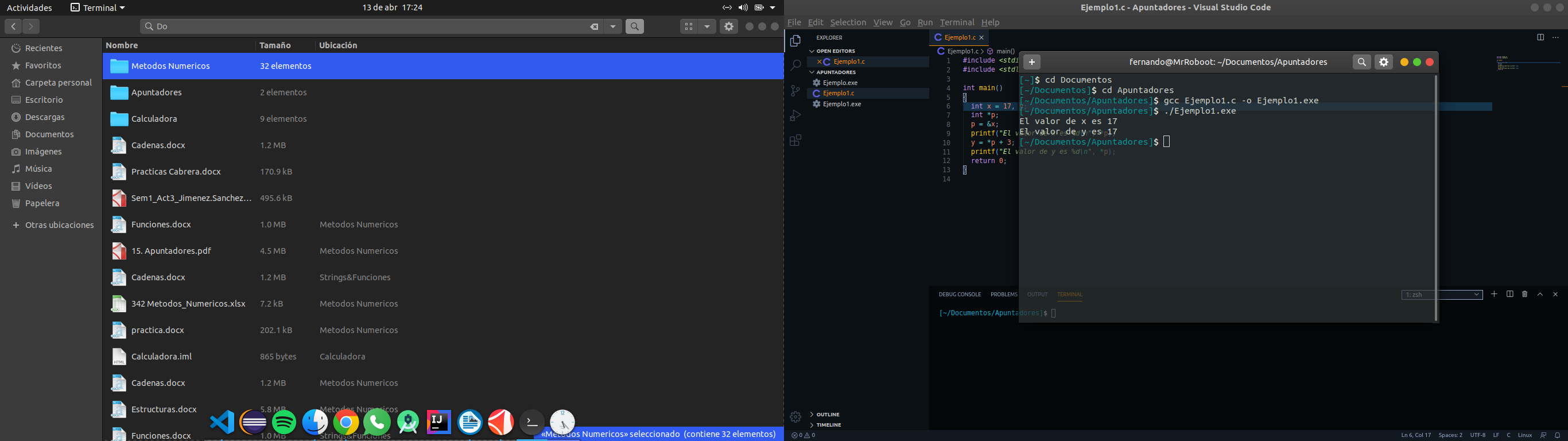
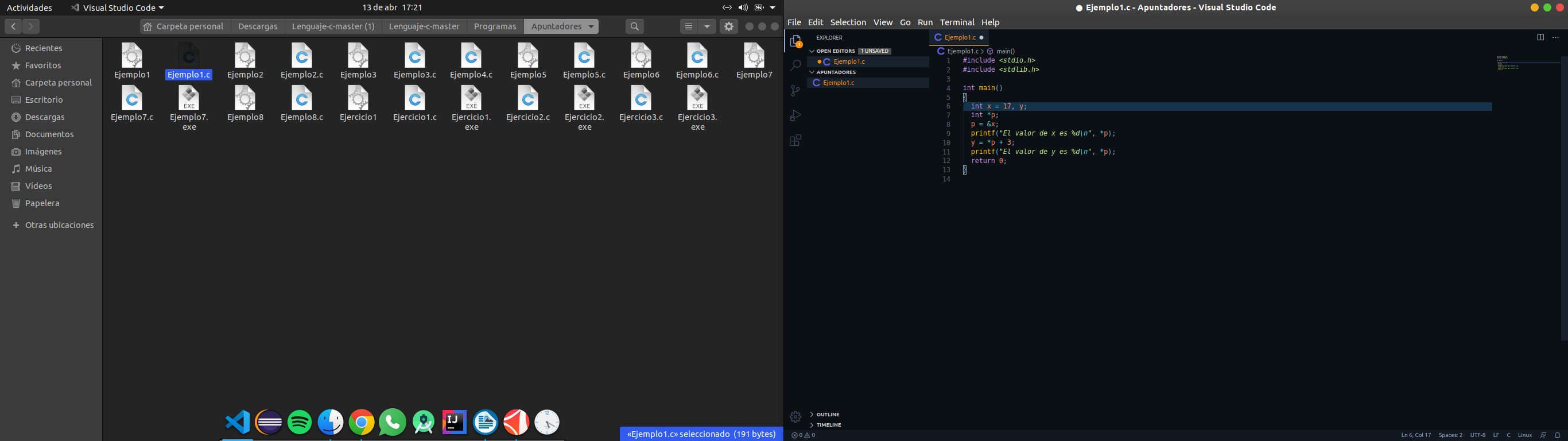
**¿Que es la referenciación?**

La diferenciación es obtener la dirección de una variable, Se hace a través del operador &, aplicado a la variable a la cual se desea saber su dirección.

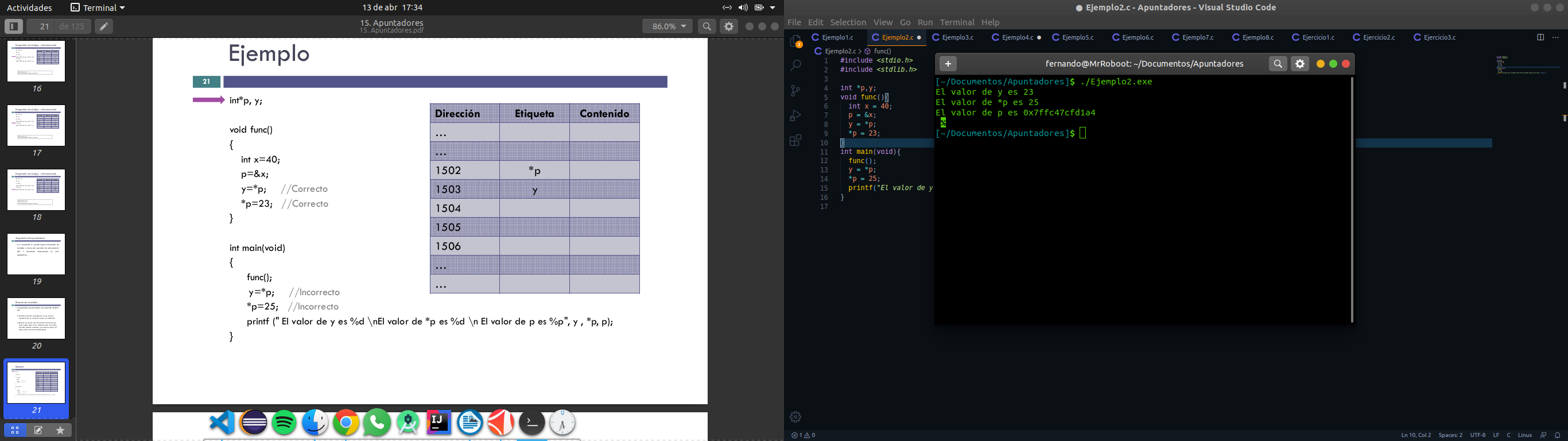
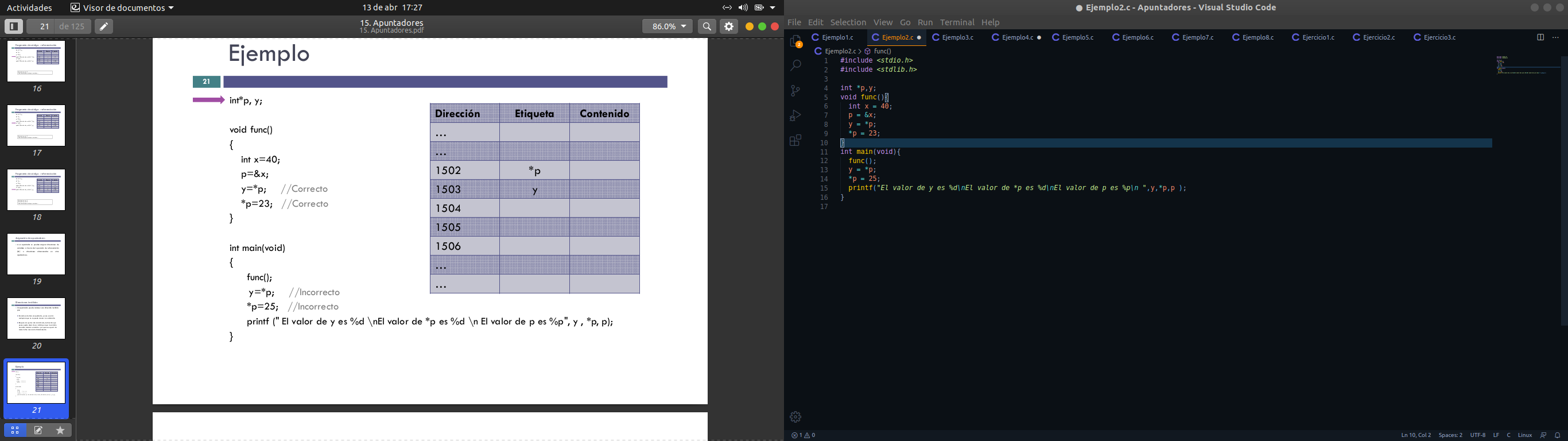
&x ; //La dirección de la variable

No hay que confundir una dirección de memoria con el contenido de esa dirección de memoria.

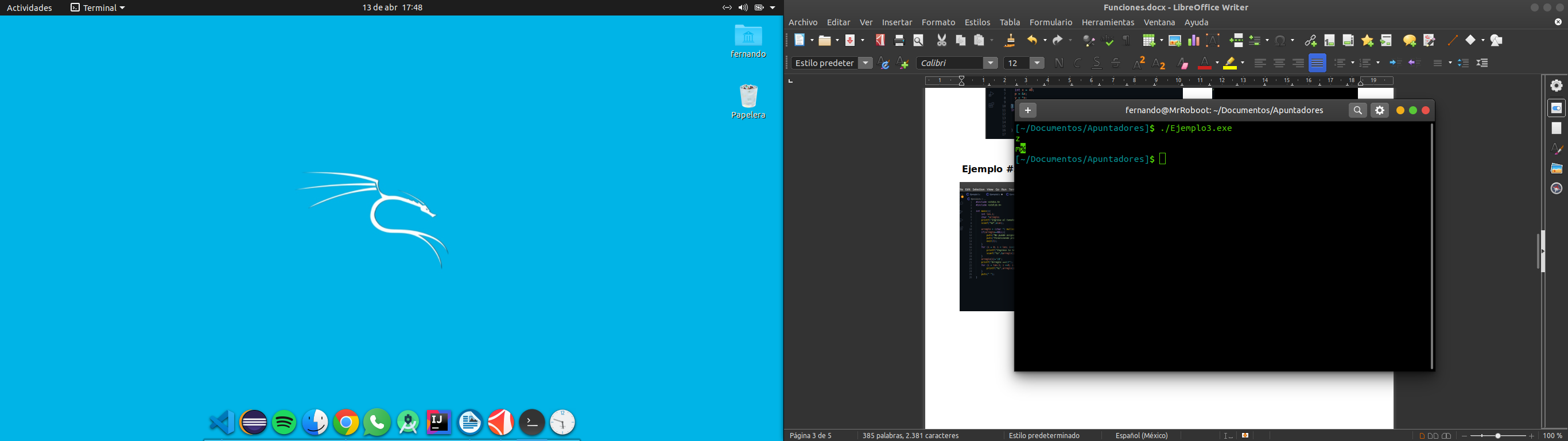
**Ejemplo #1 :**

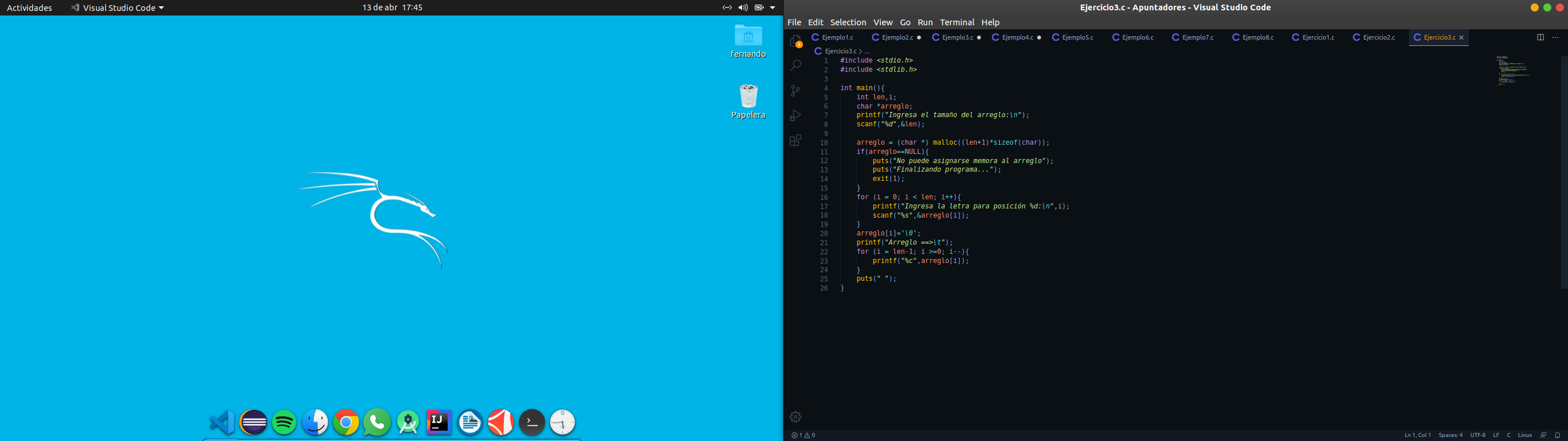


**Ejemplo #2:**

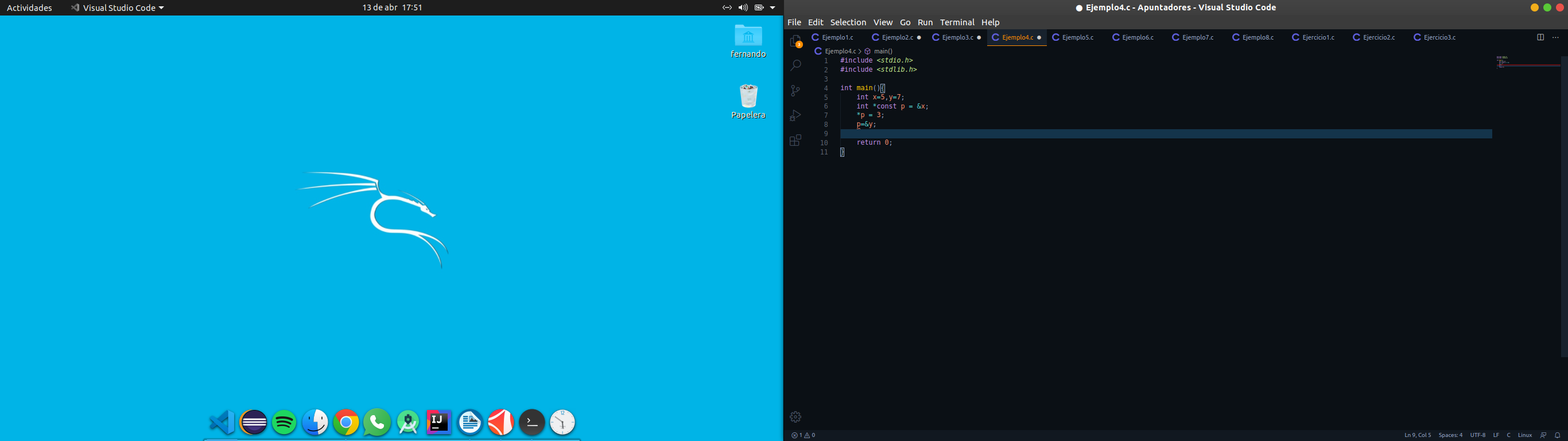


**Ejemplo #3 :**

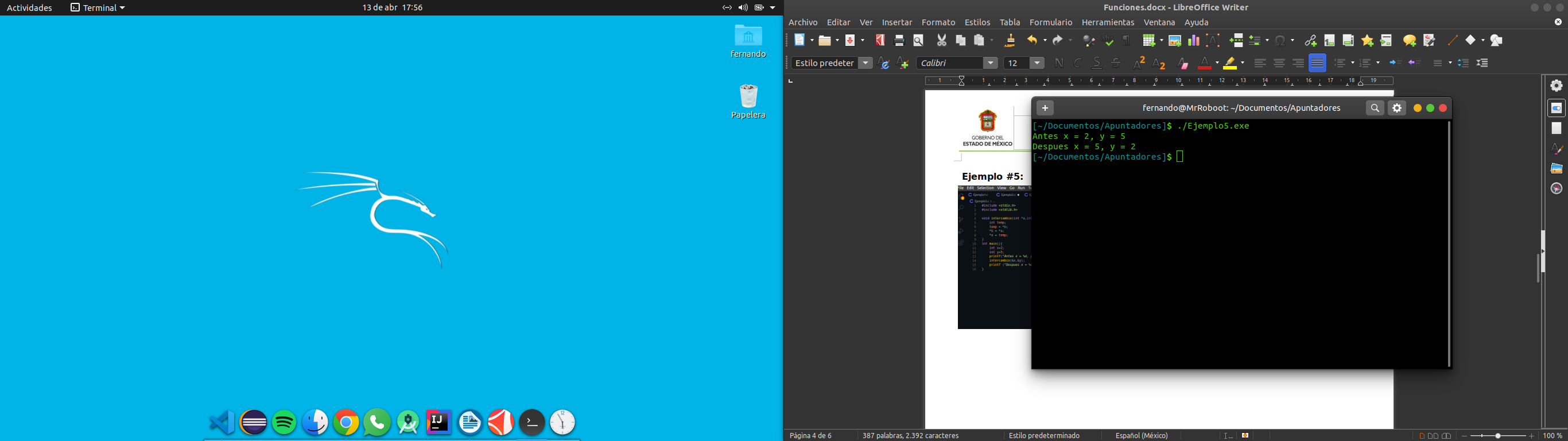
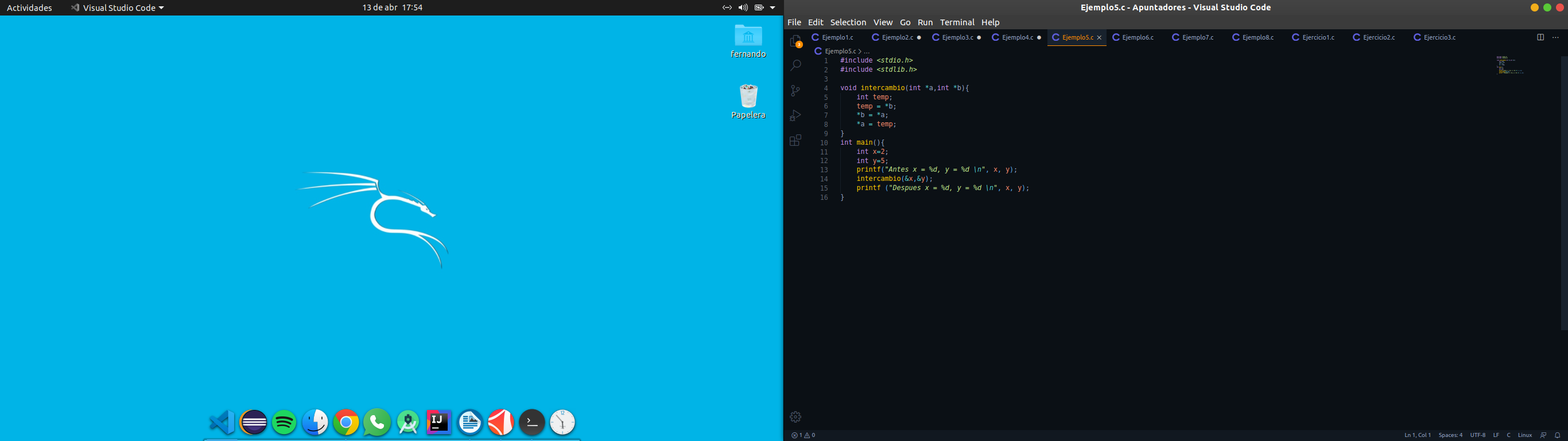




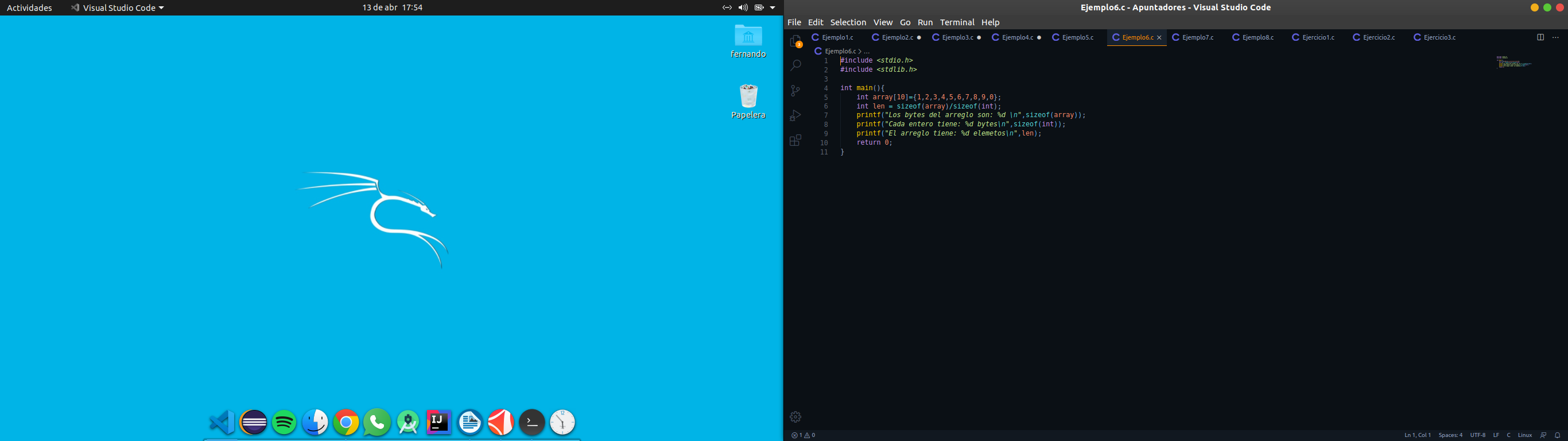
**Ejemplo #4:**



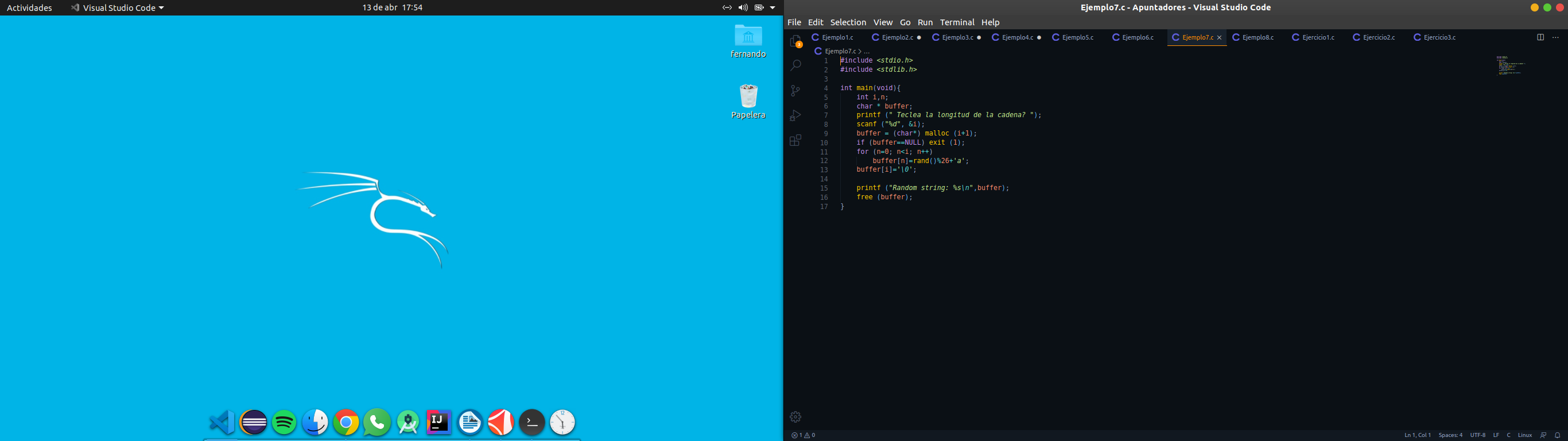
**Ejemplo #5:**



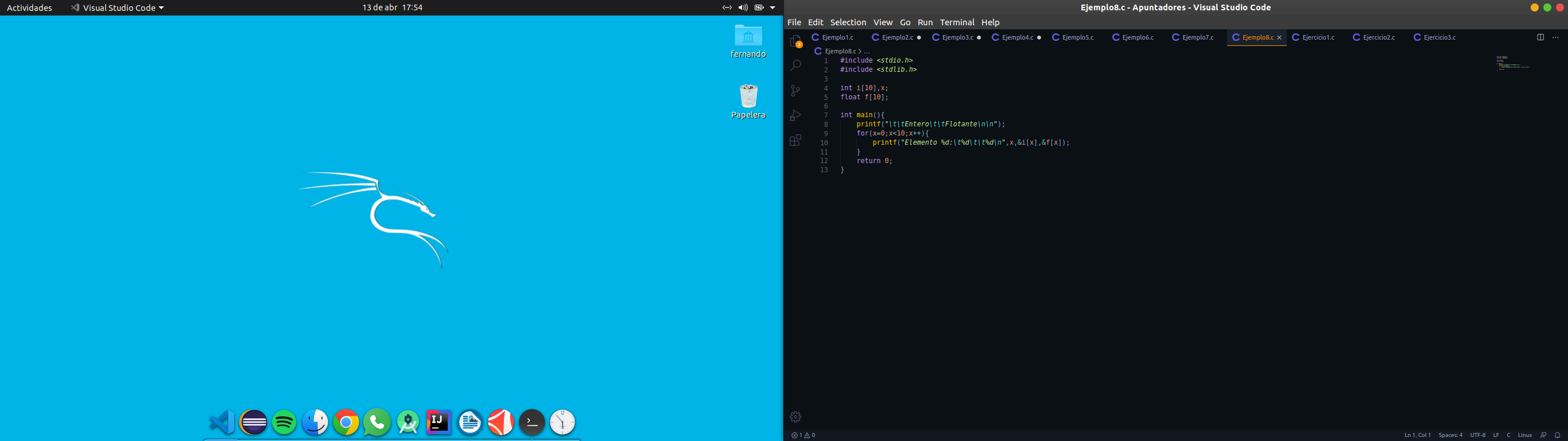
**Ejemplo #6:**

****

**Ejemplo #7:**

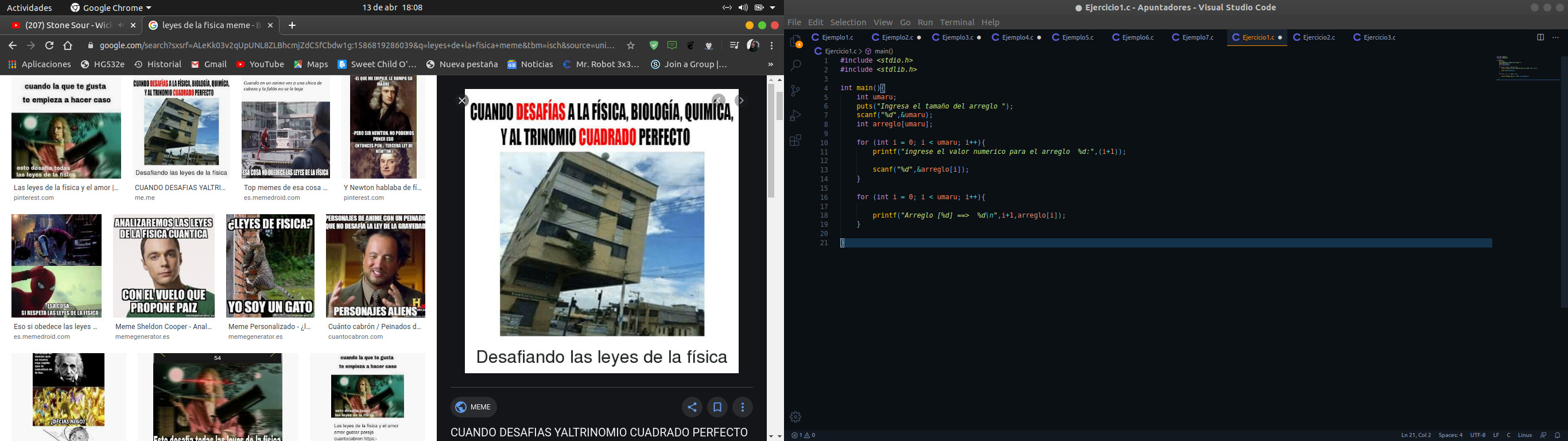


**Ejemplo #8:**

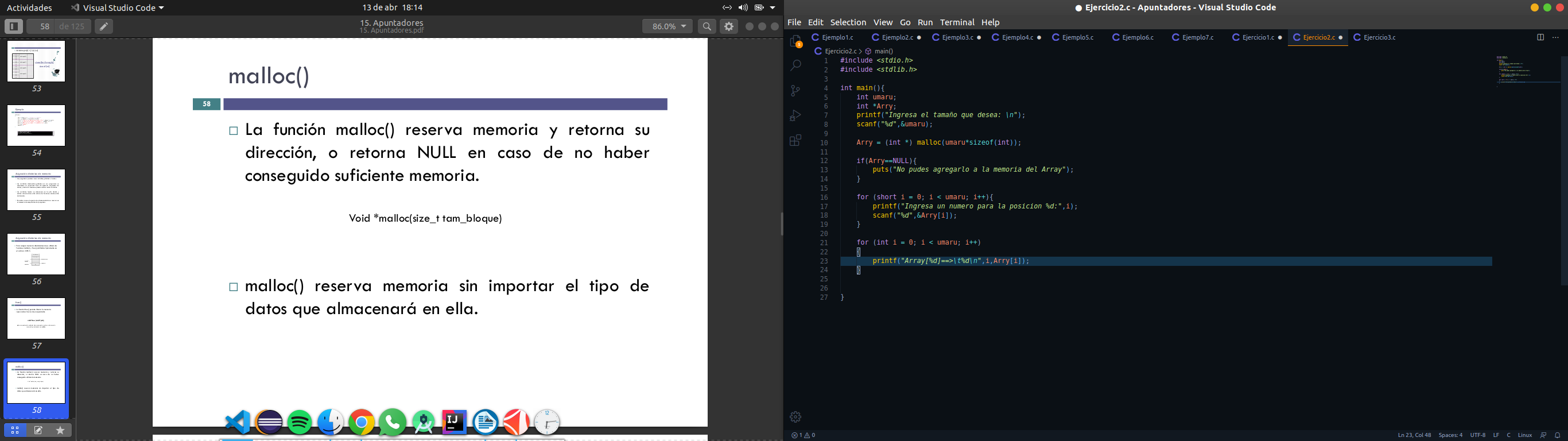
****

**EJERCICIOS**

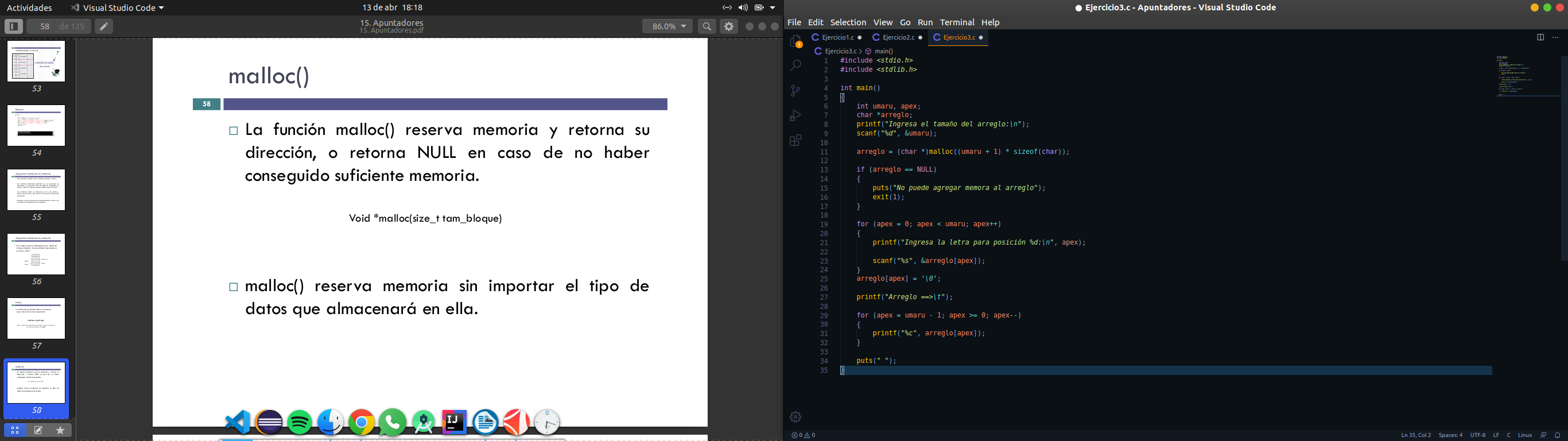
**Ejercicio #1:**

****

**Ejercicio #2:**

****

**Ejercicio #3**:



**Conclusión :**