|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nombre de la práctica** | **Apuntadores** | | | **No.** | **5** |
| **Asignatura:** | **Métodos Numéricos** | **Carrera:** | **ISIC** | **Duración de la práctica (Hrs)** | **5 Hrs** |

**Alumno:**

* **Maria del Rosario Valentín Montiel**
* **3402**

**I. Competencia(s) específica(s):**

**II. Lugar de realización de la práctica (laboratorio, taller, aula u otro):**

* Laboratorio independiente

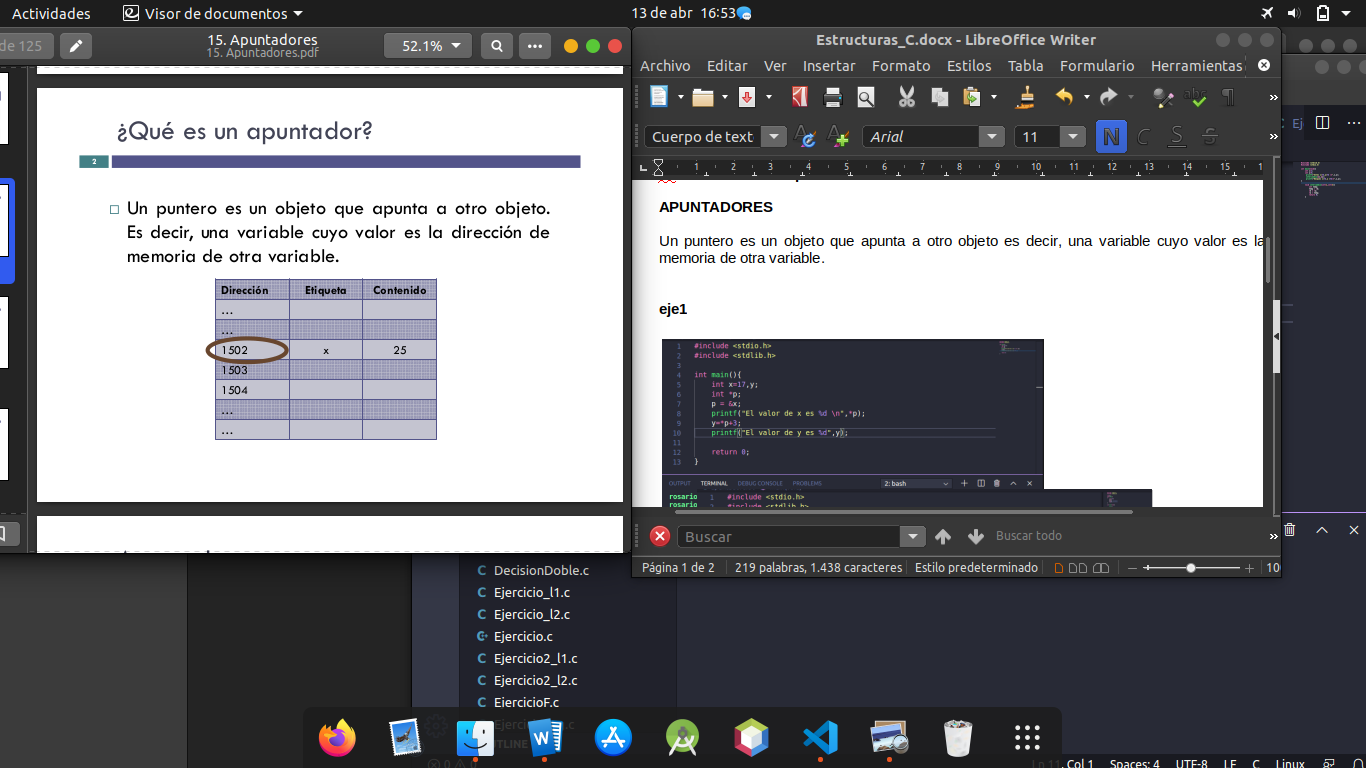
**III. Material empleado:**

* lenguaje C
* Visual Studio Code

**IV. Desarrollo de la práctica:**

**APUNTADORES**

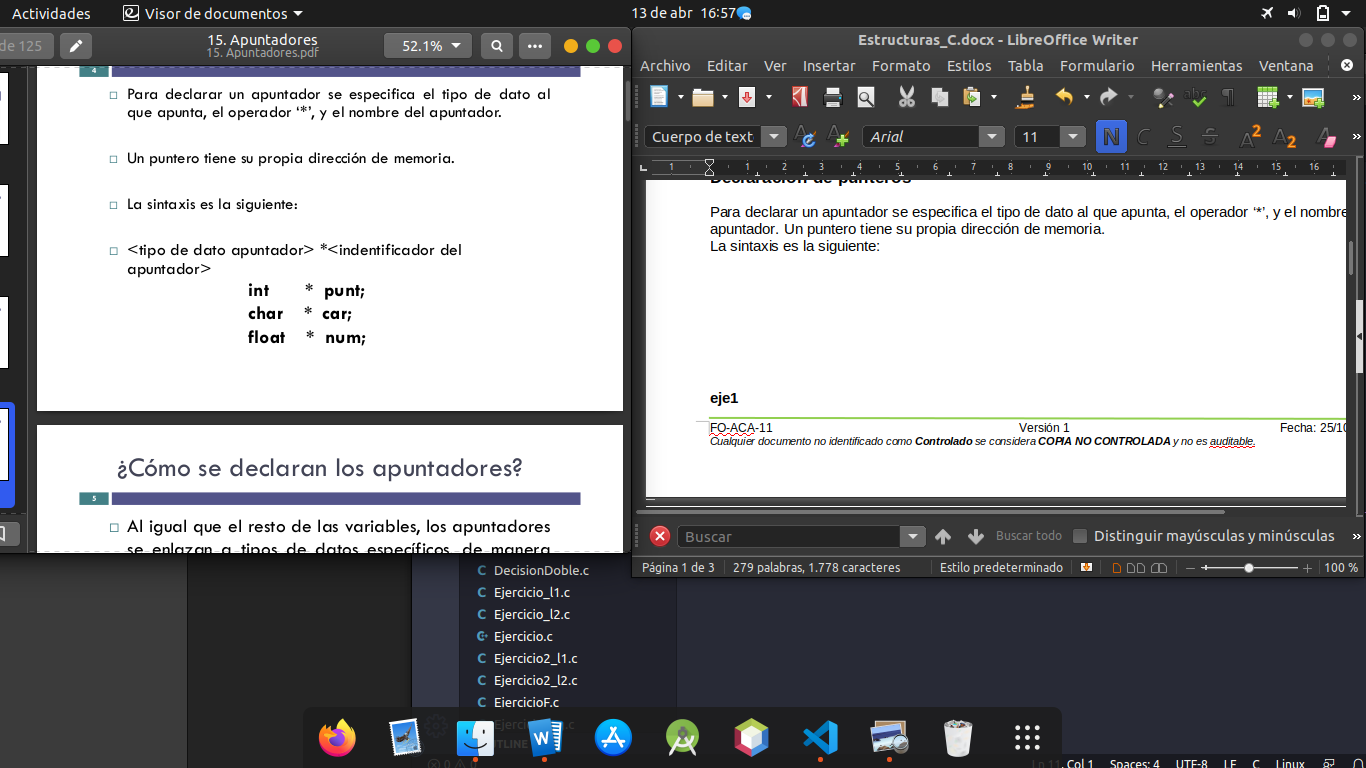
Un puntero es un objeto que apunta a otro objeto es decir, una variable cuyo valor es la dirección de memoria de otra variable. En C no se debe indicar numéricamente la dirección de la memoria, si no que se usa una etiqueta que conocemos como variable.



**Declaración de punteros**

Para declarar un apuntador se especifica el tipo de dato al que apunta, el operador ‘\*’, y el nombre del apuntador. Un puntero tiene su propia dirección de memoria.

La sintaxis es la siguiente:

****

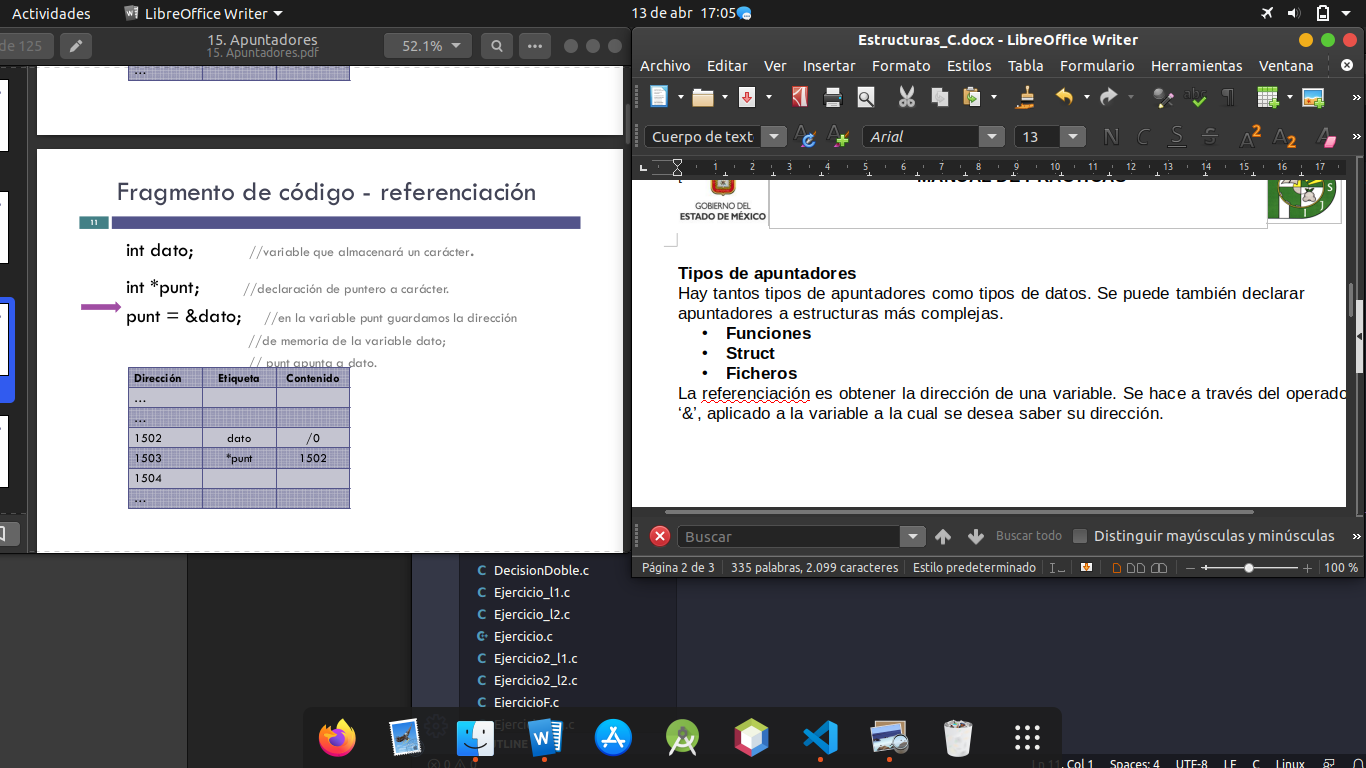
**Tipos de apuntadores**

Hay tantos tipos de apuntadores como tipos de datos. Se puede también declarar apuntadores a estructuras más complejas.

* **Funciones**
* **Struct**
* **Ficheros**

**Referenciación**

La referenciación 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.

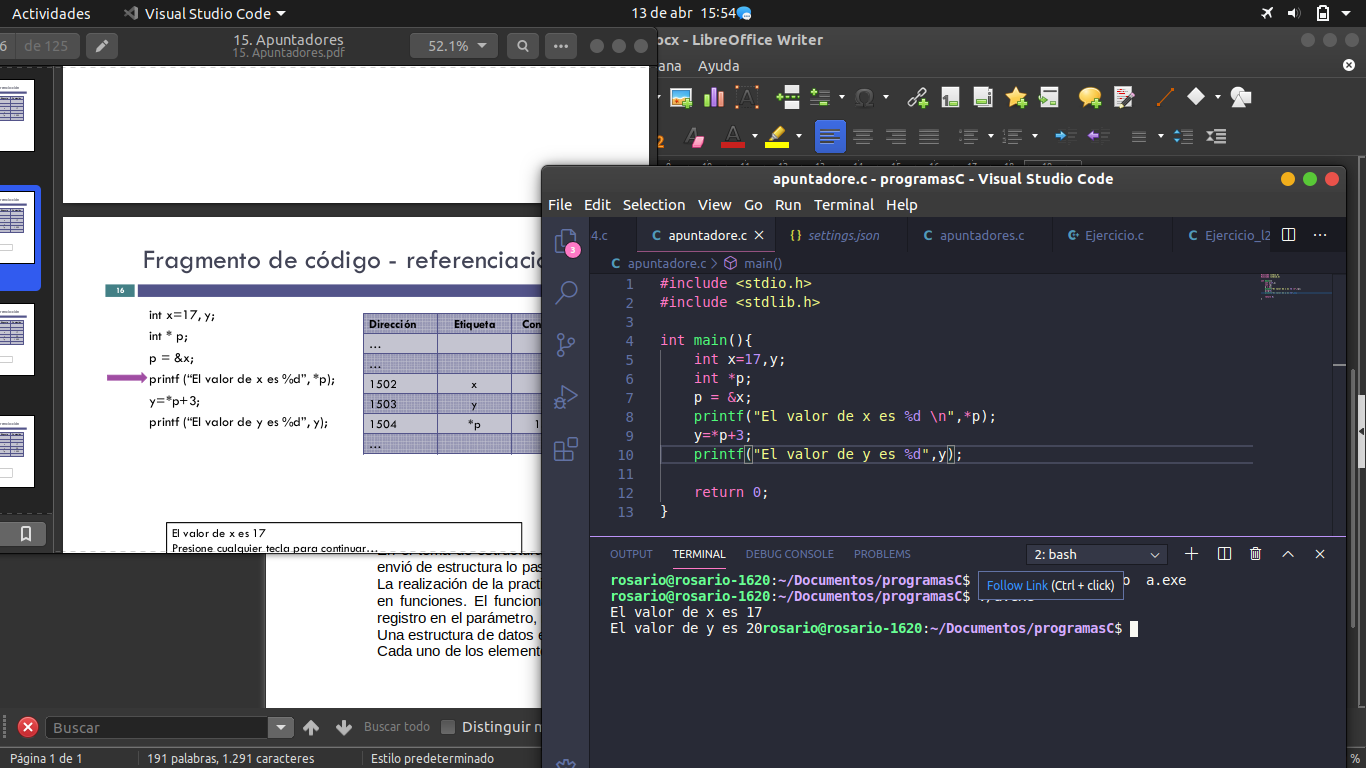


**Desreferenciación**

Es la obtención del valor almacenado en el espacio de memoria donde apunta un apuntador. Se hace a través del operador ‘\*’, aplicado al apuntador que contiene la dirección del valor.

**\*p ; //El contenido de p**

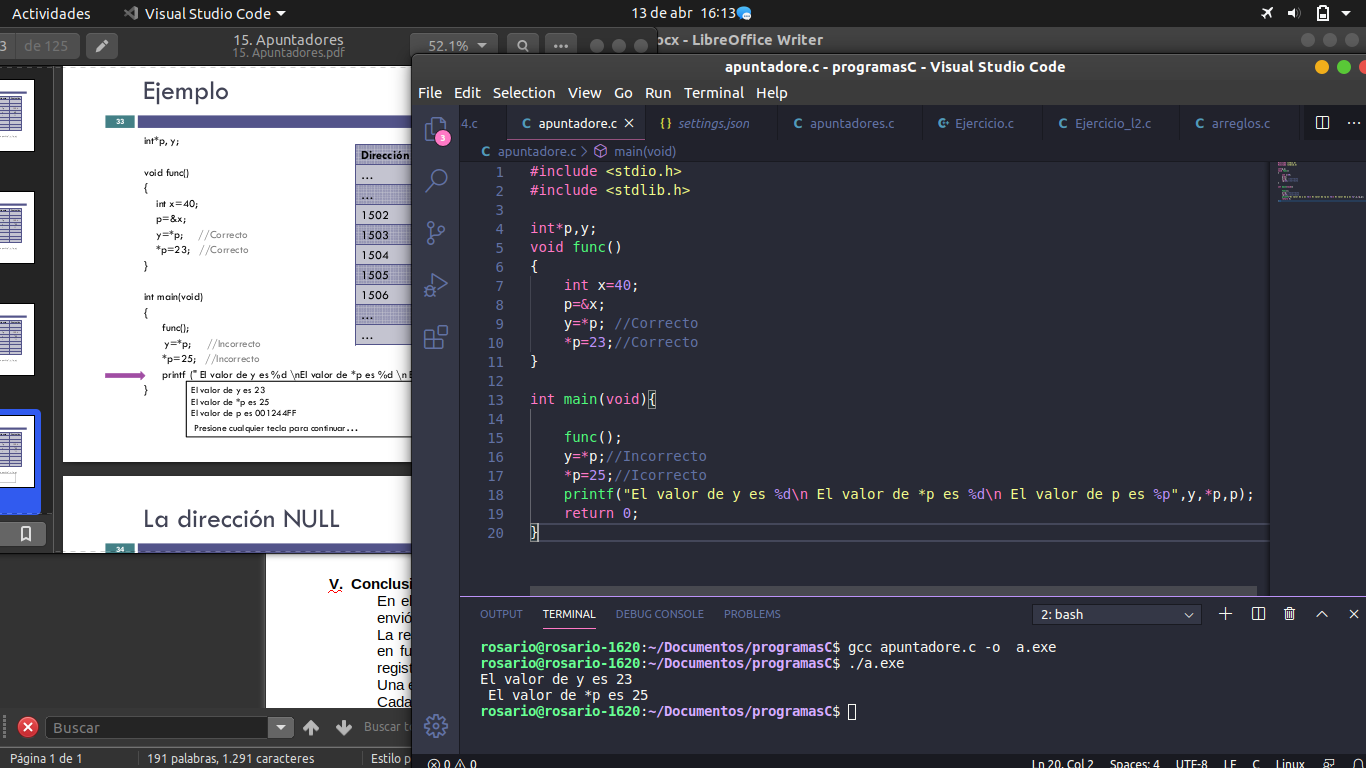
**Ejemplo 1**

****

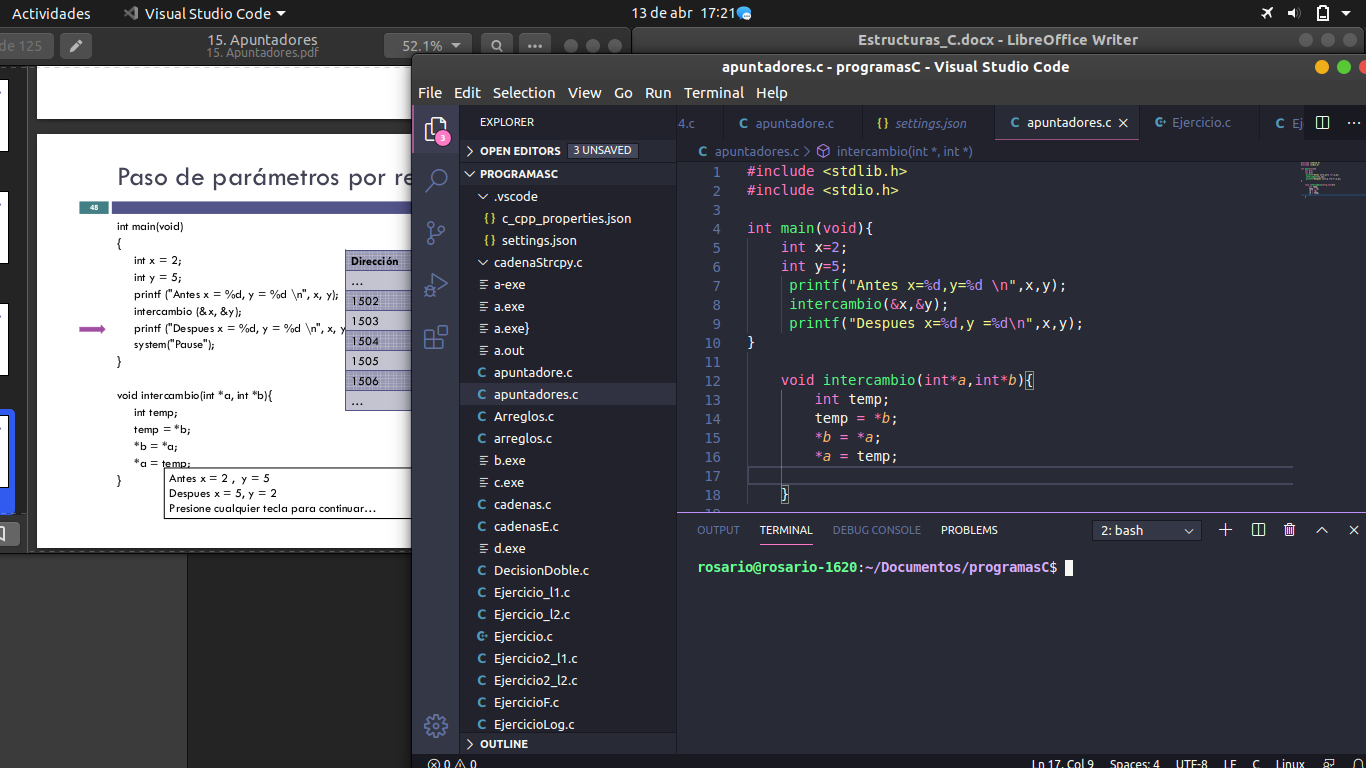
**Asignación de apuntadores**

A un apuntador se pueden asignar direcciones de variables a través del operador de referenciación (‘&’) o direcciones apuntadores.

**Ejemplo 2**

****

**Ejemplo 3**



**La función sizeof ()**

Devuelve el tamaño en bytes que ocupa un tipo o variable en memoria.

char cadena [10];

printf (“un int ocupa %d bytes”, sizeof(int));

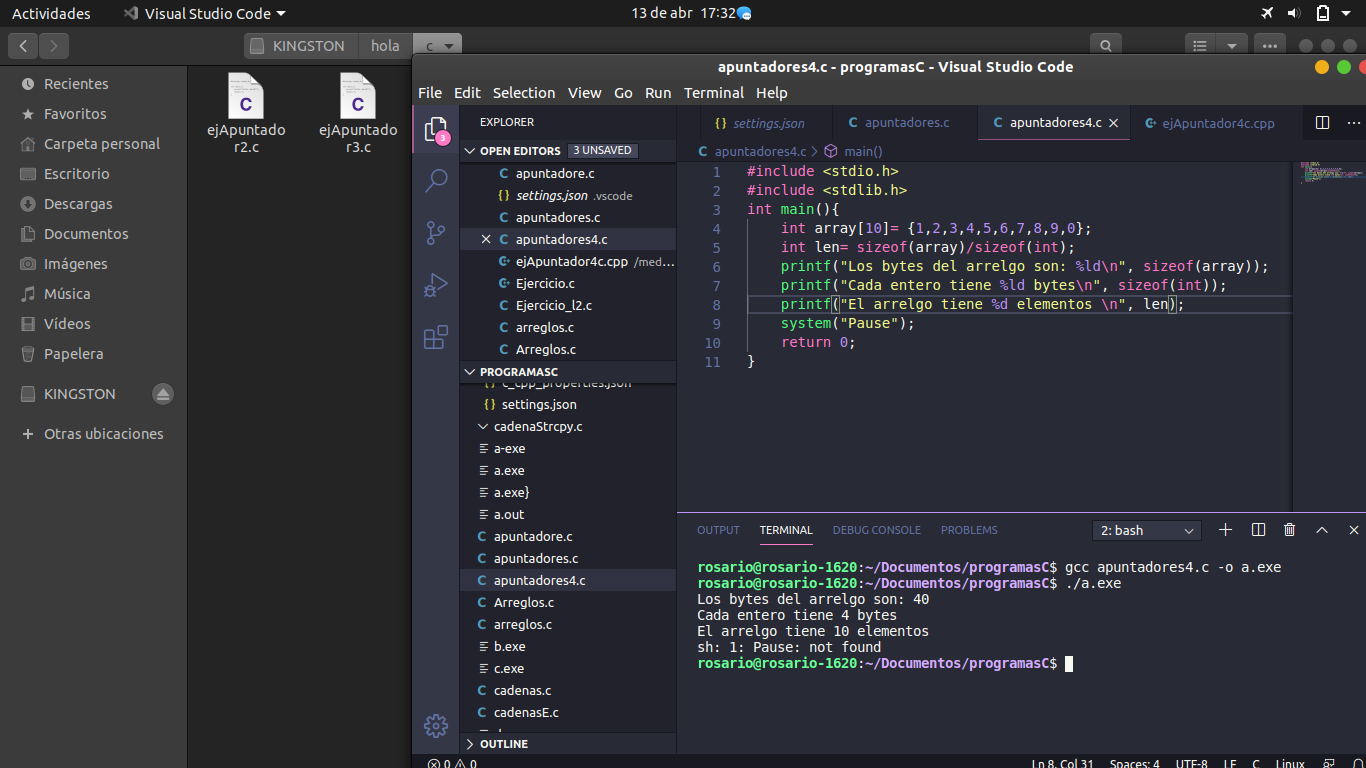
printf (“un char ocupa %d bytes”, sizeof(char));

printf (“un float ocupa %d bytes”, sizeof(float));

printf (“un double ocupa %d bytes”, sizeof(double));

printf (“cadena ocupa %d bytes”, sizeof(cadena));

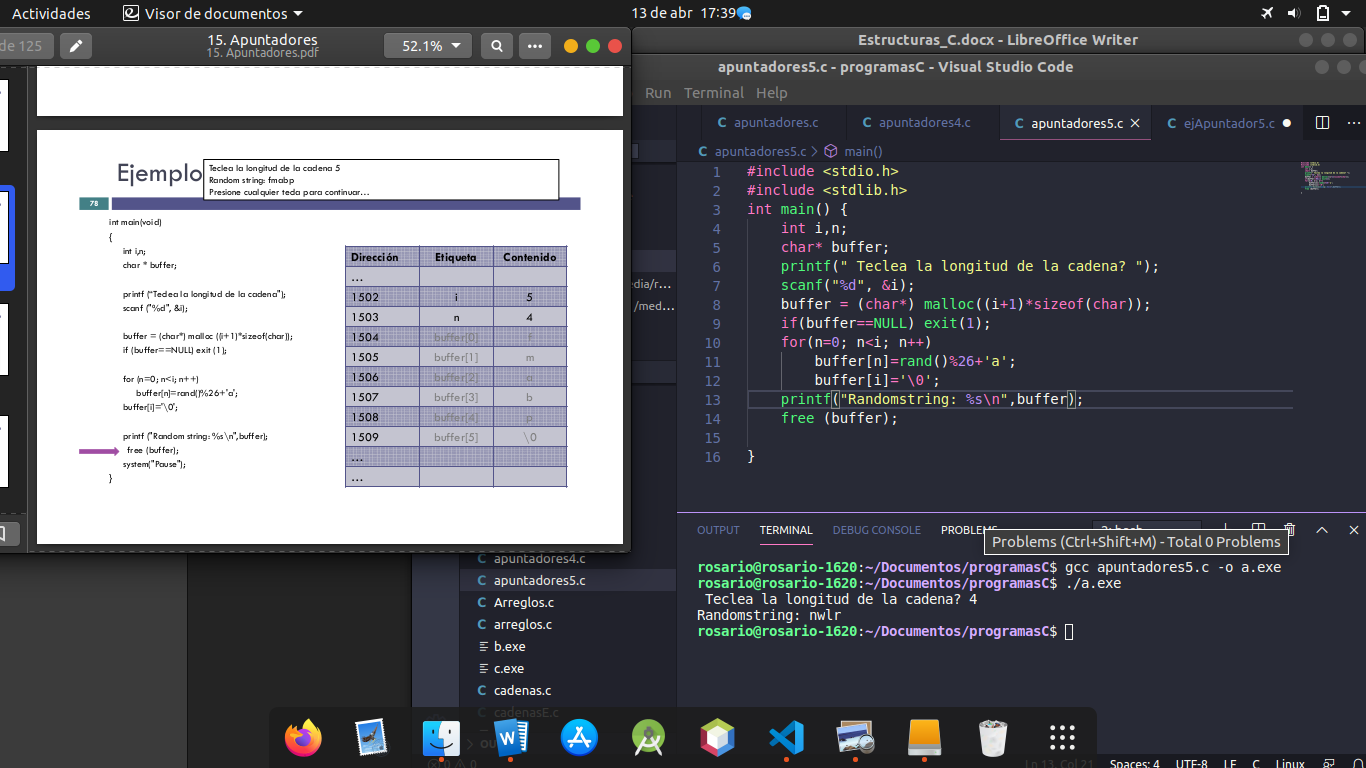
**Ejemplo 4**

****

**Asignación dinámica de memoria**

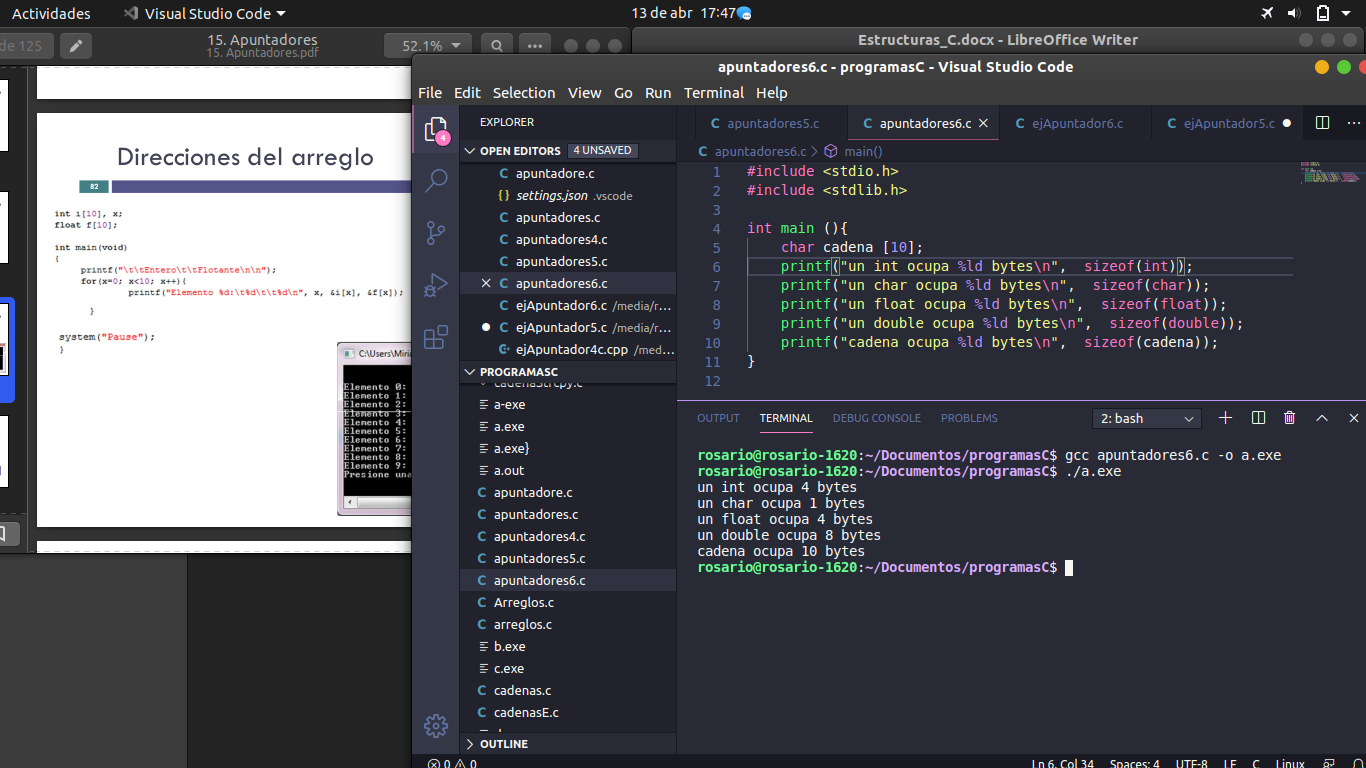
Para asignar memoria dinámicamente se utilizan las funciones malloc() y free(), definidas típicamente en el archivo stdlib.h.

**Ejemplo 5**

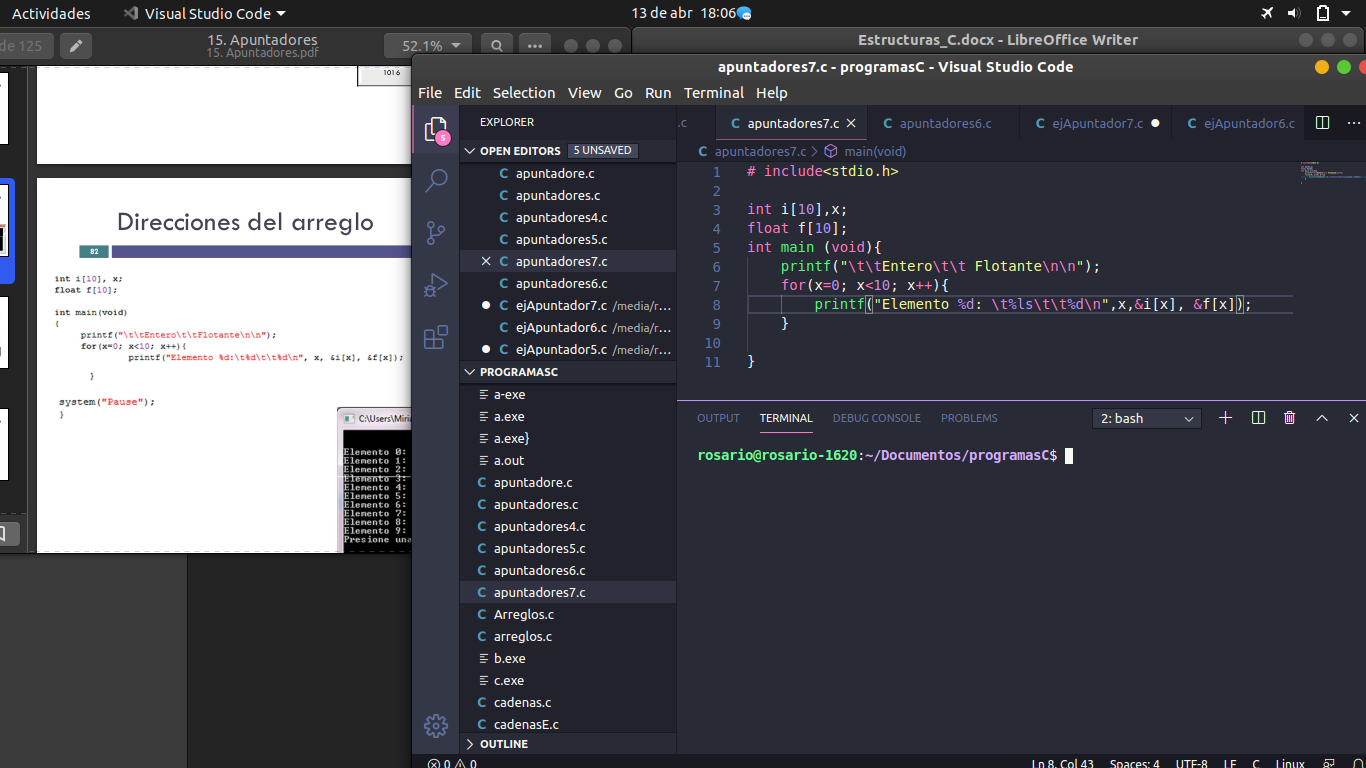
****

**Funciones sizeof()**

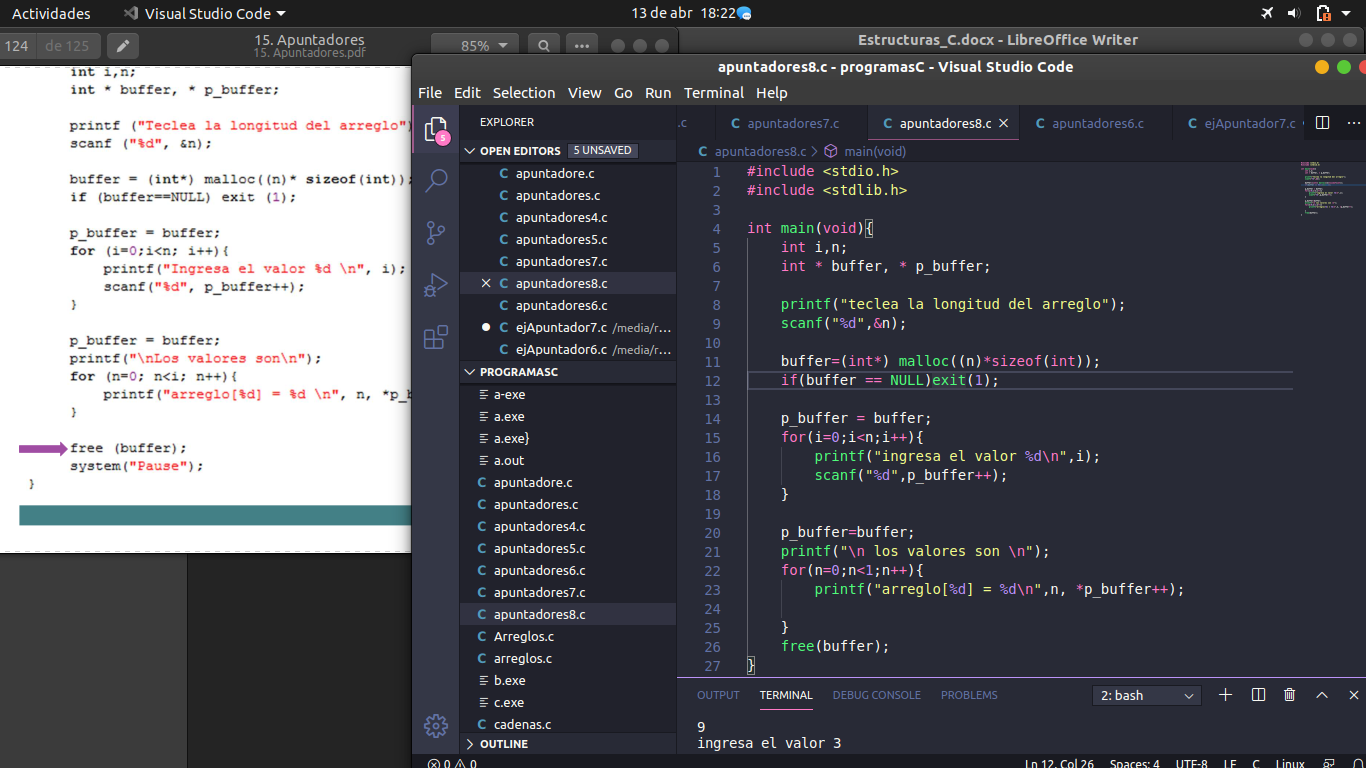
**Ejemplo 6**

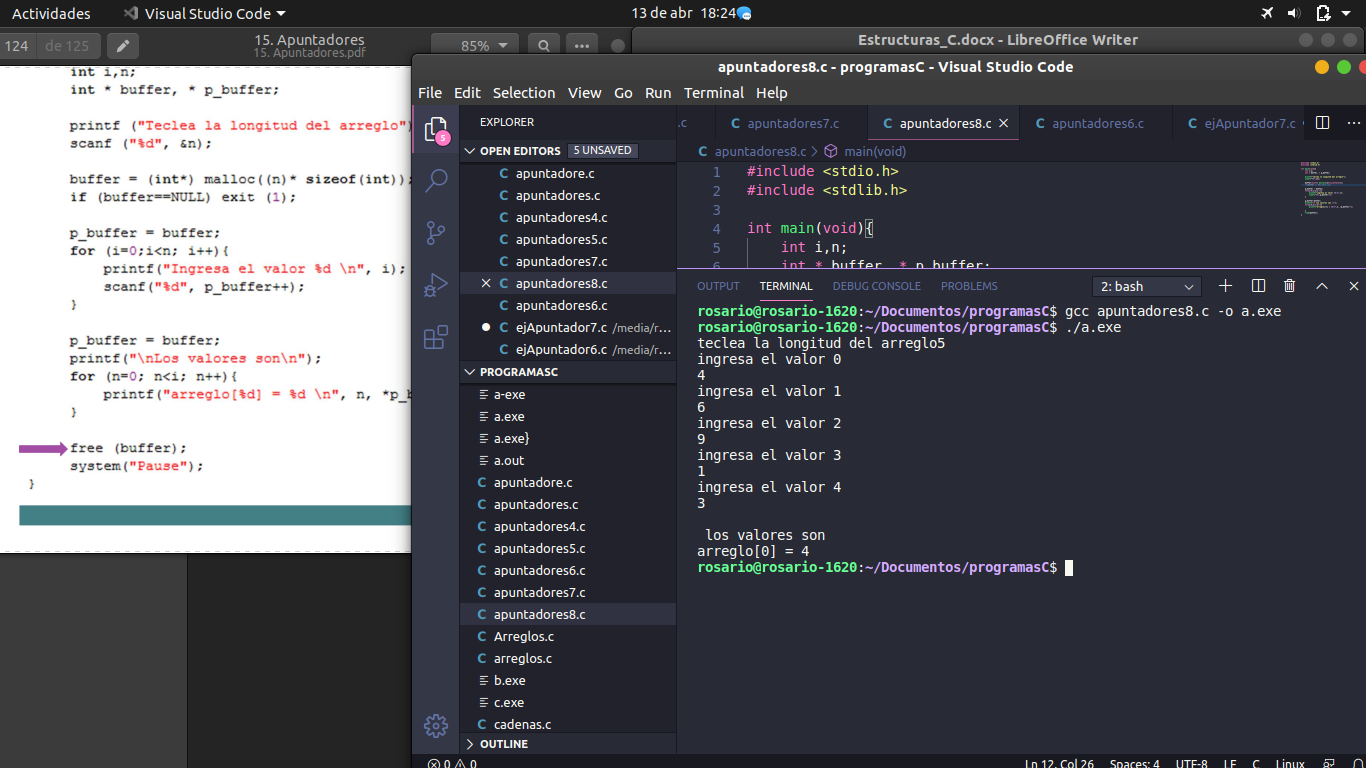
****

**Ejemplo 7**

****

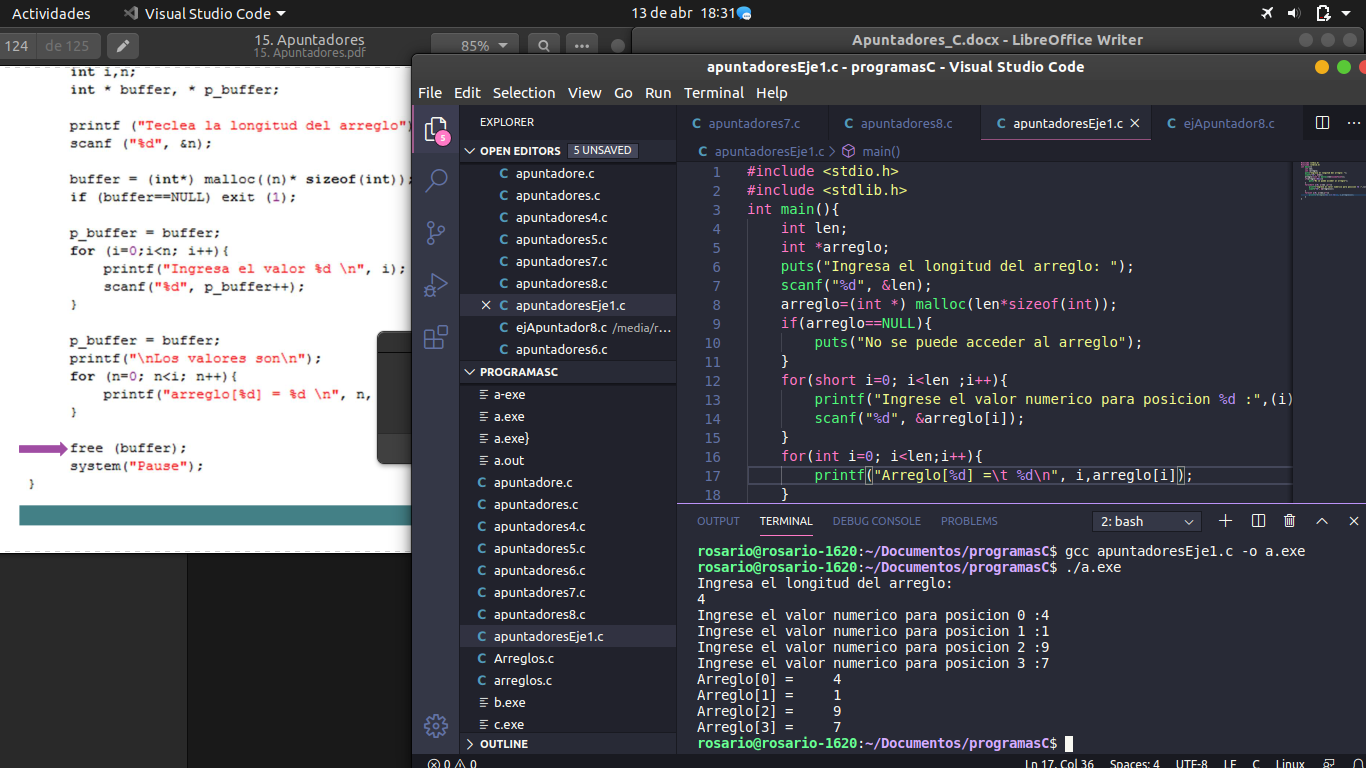
**Ejemplo 8**

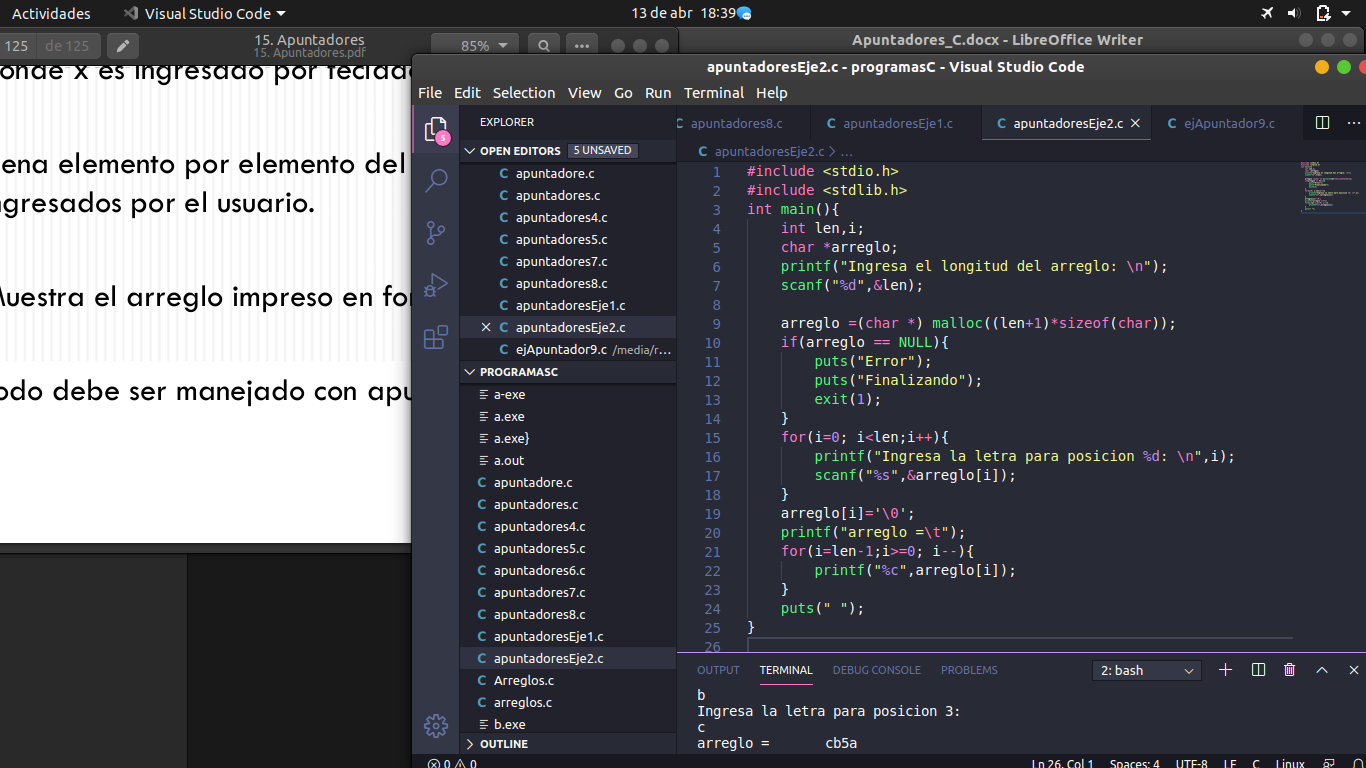
****

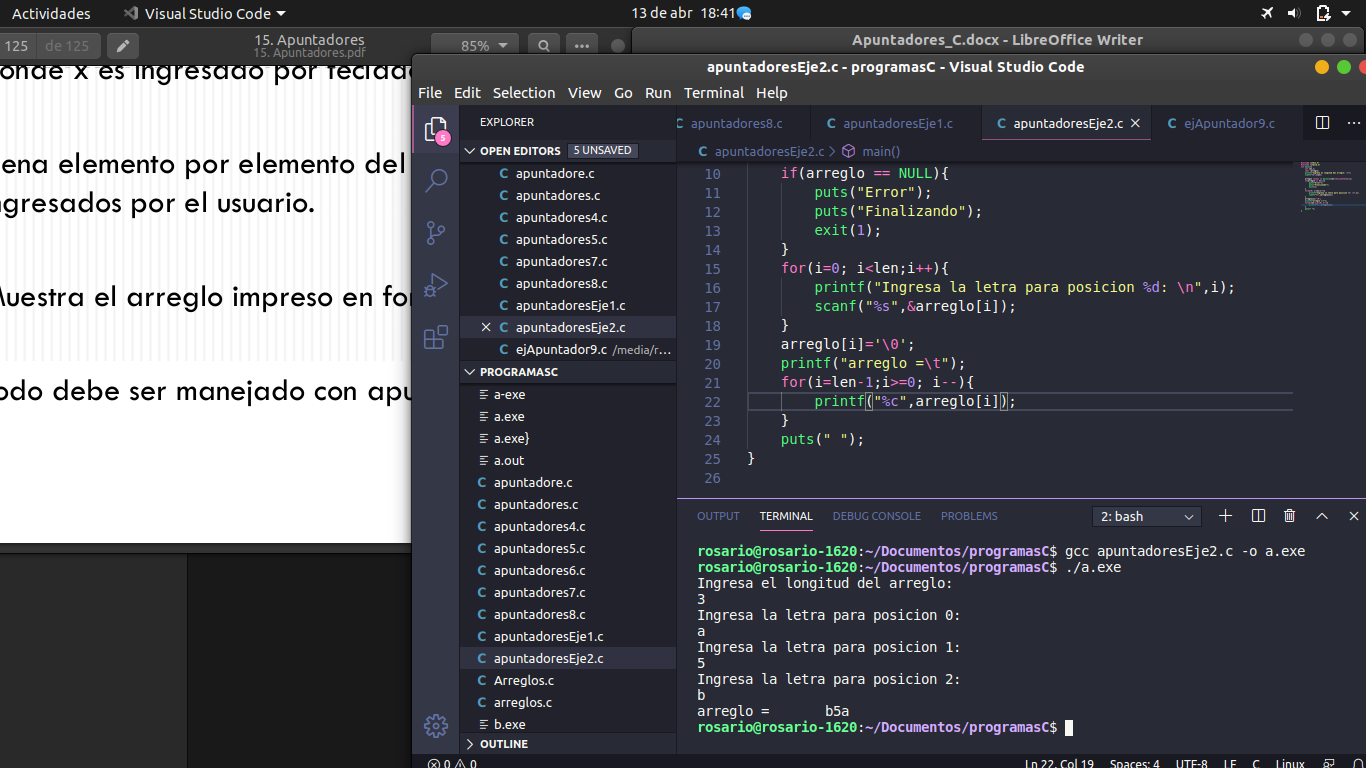
****

**Ejercicio 1**

* Crea un arreglo de tipo char de tamaño x, en donde x es ingresado por teclado.
* Llena elemento por elemento del arreglo con letras ingresados por el usuario.
* Muestra el arreglo impreso en forma inversa.
* Todo debe ser manejado con apuntadores.

****





**V. Conclusiones:**

En el tema de estructura es una manera de realizar un código de mejor forma ocupamos en los apuntadores donde es un puntero que apunta a otro en forma de objeto. En C no se debe indicar numéricamente la dirección de la memoria, si no que se usa una etiqueta que conocemos como variable.