|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nombre de la práctica** | **Lamina 15** | | | **No.** | **5** |
| **Asignatura:** | **Métodos numéricos** | **Carrera:** | **Ing. Sistemas Computacionales** | **Duración de la práctica (Hrs)** |  |

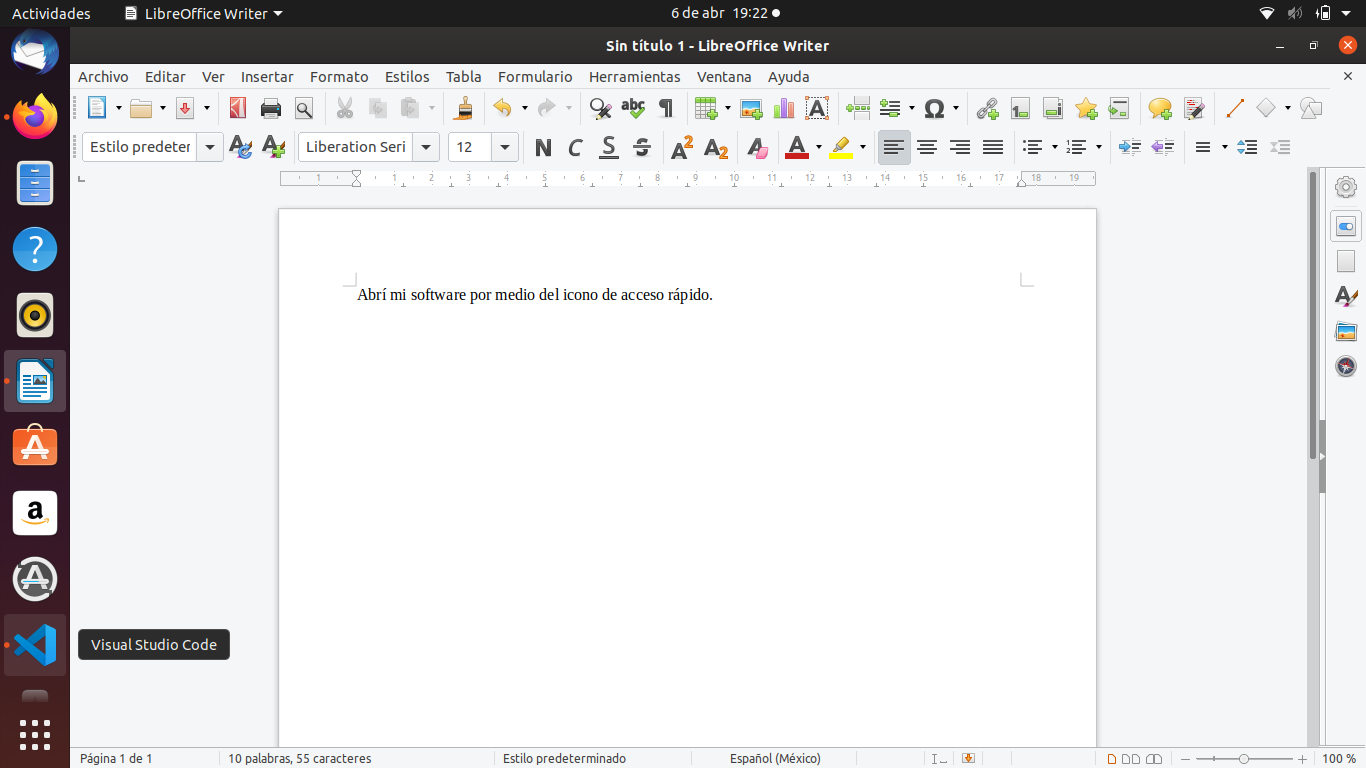
**II. Lugar de realización de la práctica (laboratorio, taller, aula u otro):** Otro

**III. Material empleado:** laptop y Visual Studio Code

**IV. Desarrollo de la práctica:**

# Lamina 15 Apuntadores

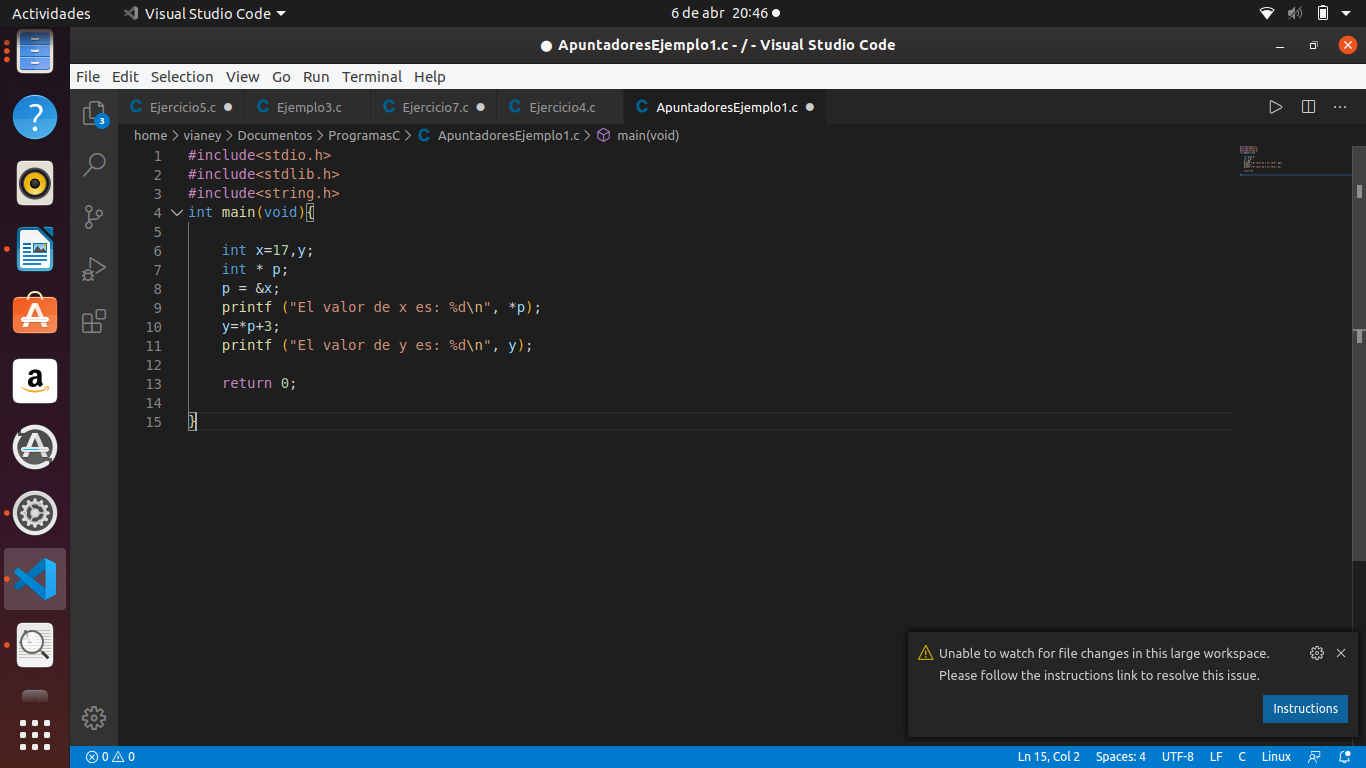
Abrí mi software por medio del icono de acceso rápido.



Posteriormente doy clic sobre la barra frontal en File>New file >File>Save As>nombre.c>save



En el presente código lo que se realiza es declarar 2 variables, en las cuales como podemos ver x tiene un valor ya definido (17), después de clara un apuntador de acuerdo a su sintaxis el tipo de dato, asterisco y el identificador de este, después tenemos la asignación de este valor, donde p toma un valor de x, es importante tomar en cuenta que toma el valor que posee dicha dirección, mas no el valor de la dirección , para después imprimir este valor, mientras que y toma el mismo valor que posee el apuntado y le suma 3 para finalmente imprimirlo.

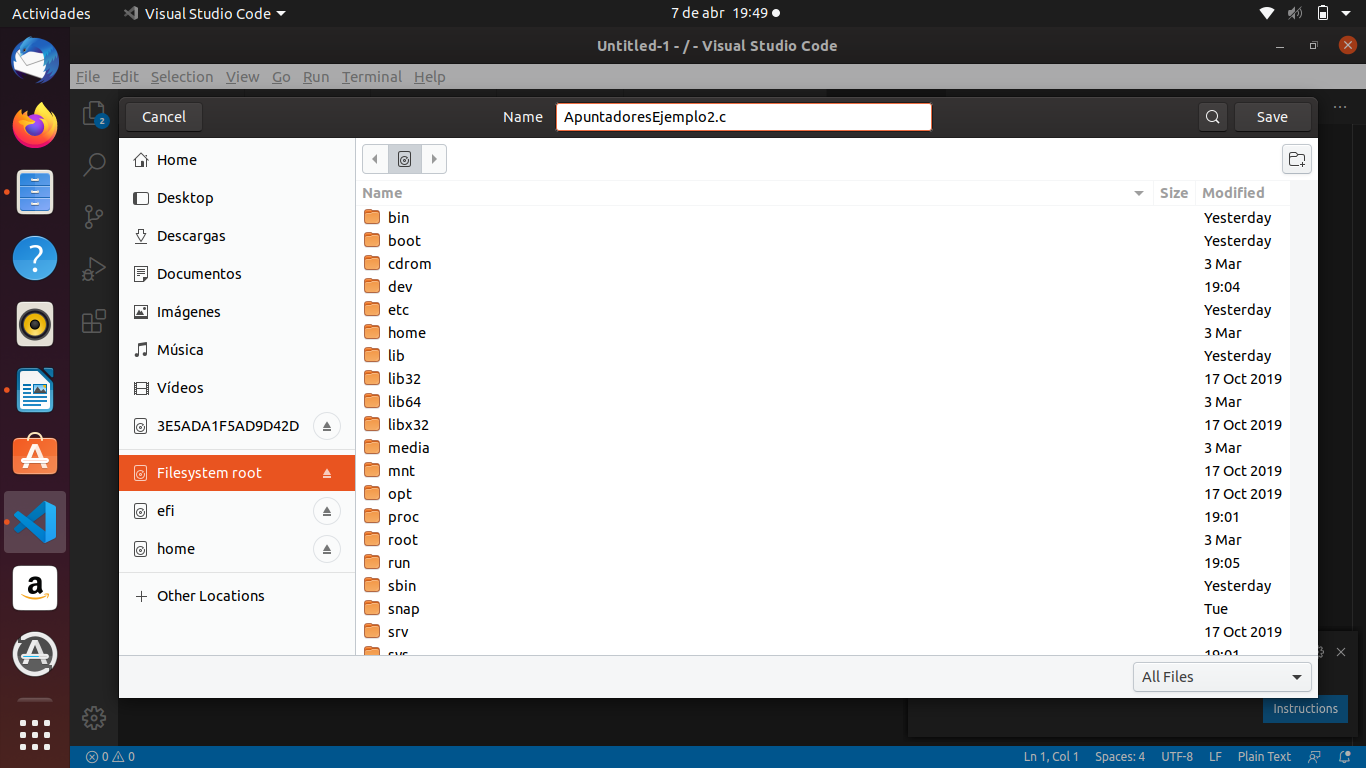


Finalmente al compilar el programa este es el resultado acorde a la explicación.



El

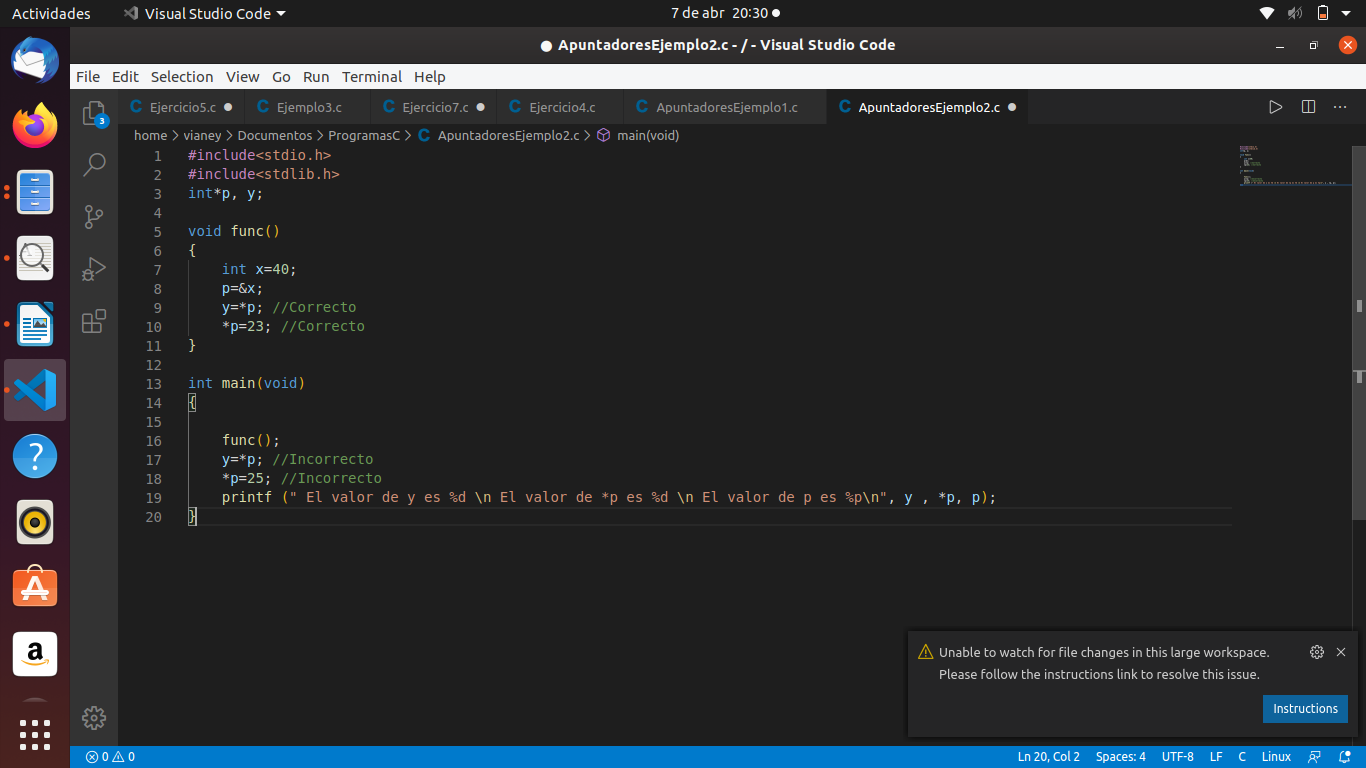
siguiente ejemplo muestra como se pueden asignar nuevos valores a una variable por medio de apuntadores, para lo cual cree el segundo programa.



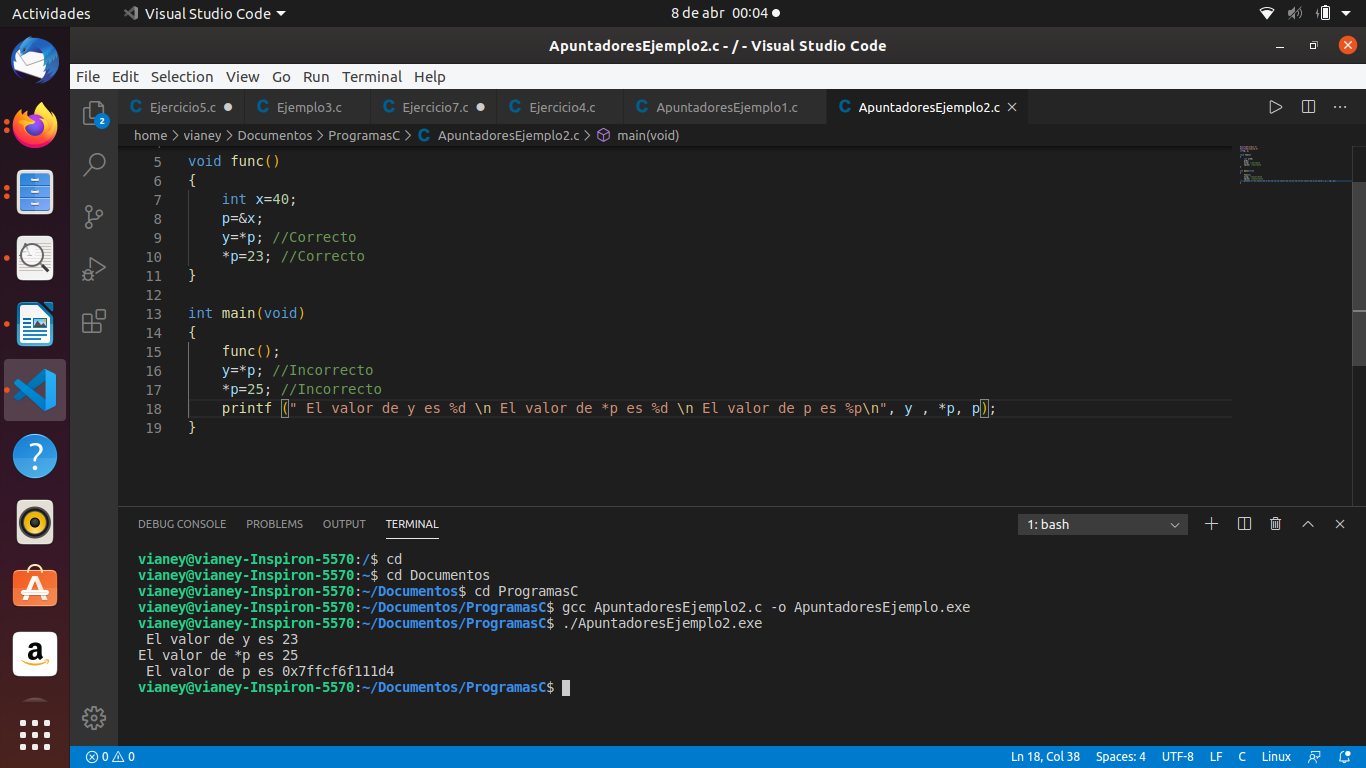
Como se muestra en el siguiente código tenemos una función el la cual se declara el valor del argumento x el cual es 40, posteriormente p toma el valor de la dirección de x y y el valor del apuntador, por lo tanto como el apuntador apunta al valor de la dirección de x su valor de y es el mismo que el d x y finalmente en la función asignamos el valor de 23 a la dirección que tiene el puntero, por lo tanto ahora el valor de x es 23.

Ahora una vez dentro de la función principal.

Una vez dentro del método principal se manda a llamara a la función y toma el valor al que apunta el apuntador, por lo tanto ahora p toma el valor de x, entonces ahora y vale 23 y después indica que la dirección a la que hace referencia el apuntador ahora almacenará el valor de 25



De acuerdo a la explicación este es el valor obtenido es el siguiente.

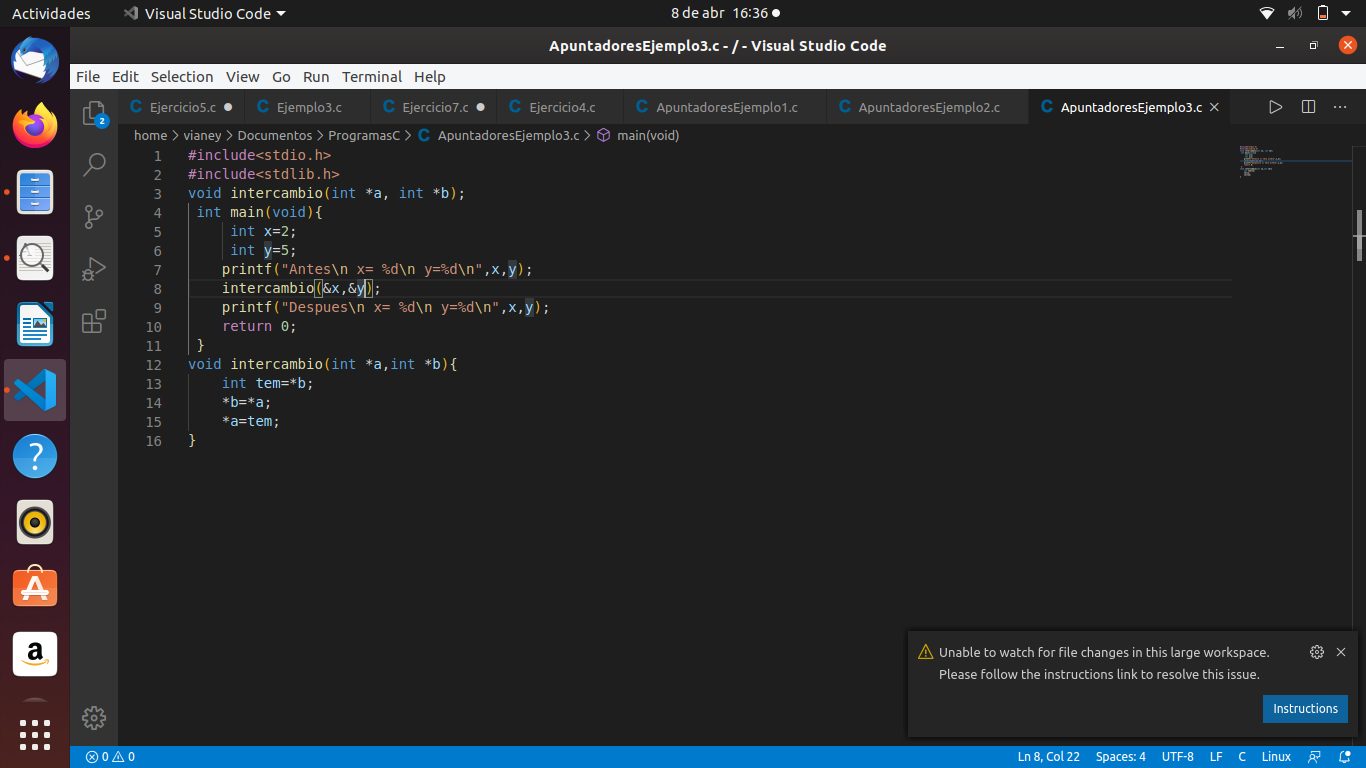


En este presente proyecto mostraremos como asignar parámetros por referencia, y nombre de la siguiente manera mi proyecto.

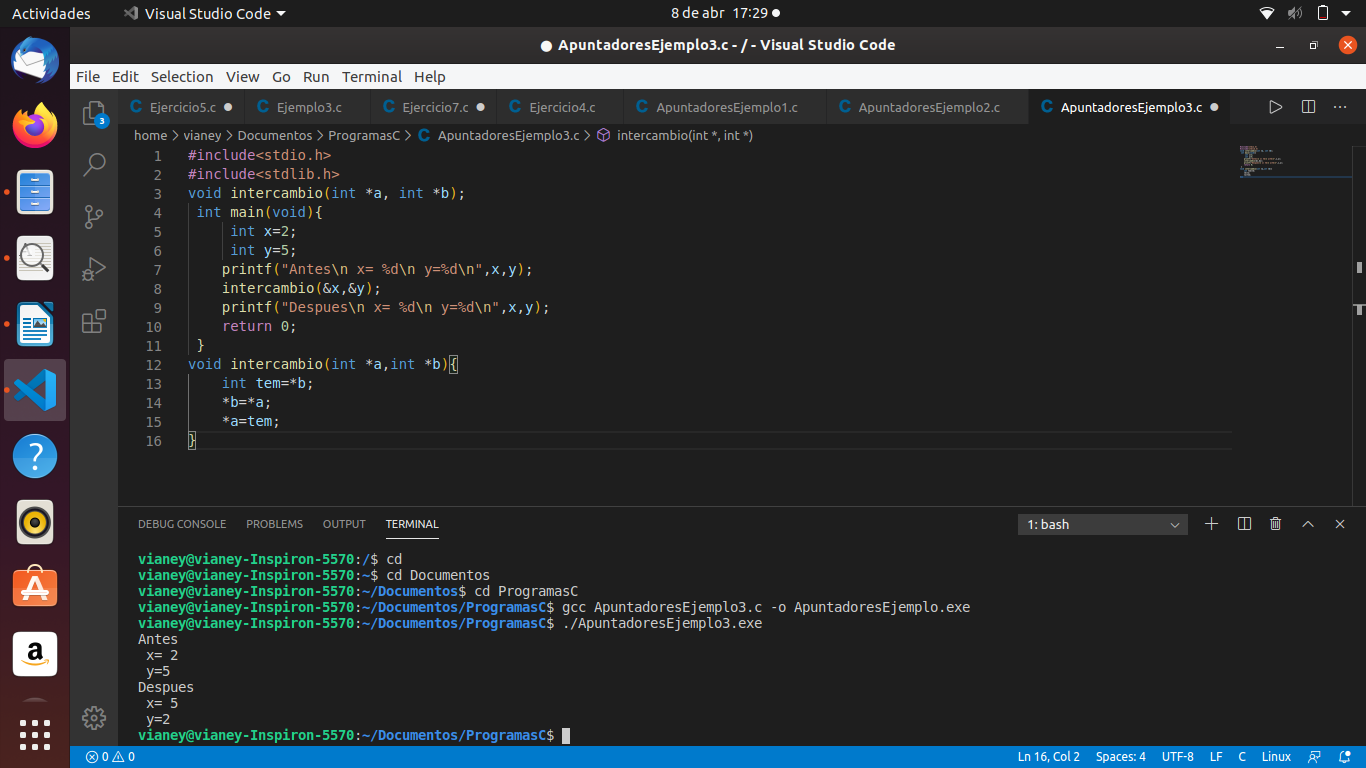


En cuanto al código, como se muestra en la captura tenemos una función declarada, pero comenzamos con el método main con el cual tenemos 2 argumentos x y y con sus respectivos valores de 2y 5, los cuales se imprimen directamente, posteriormente se manda a llamar la función

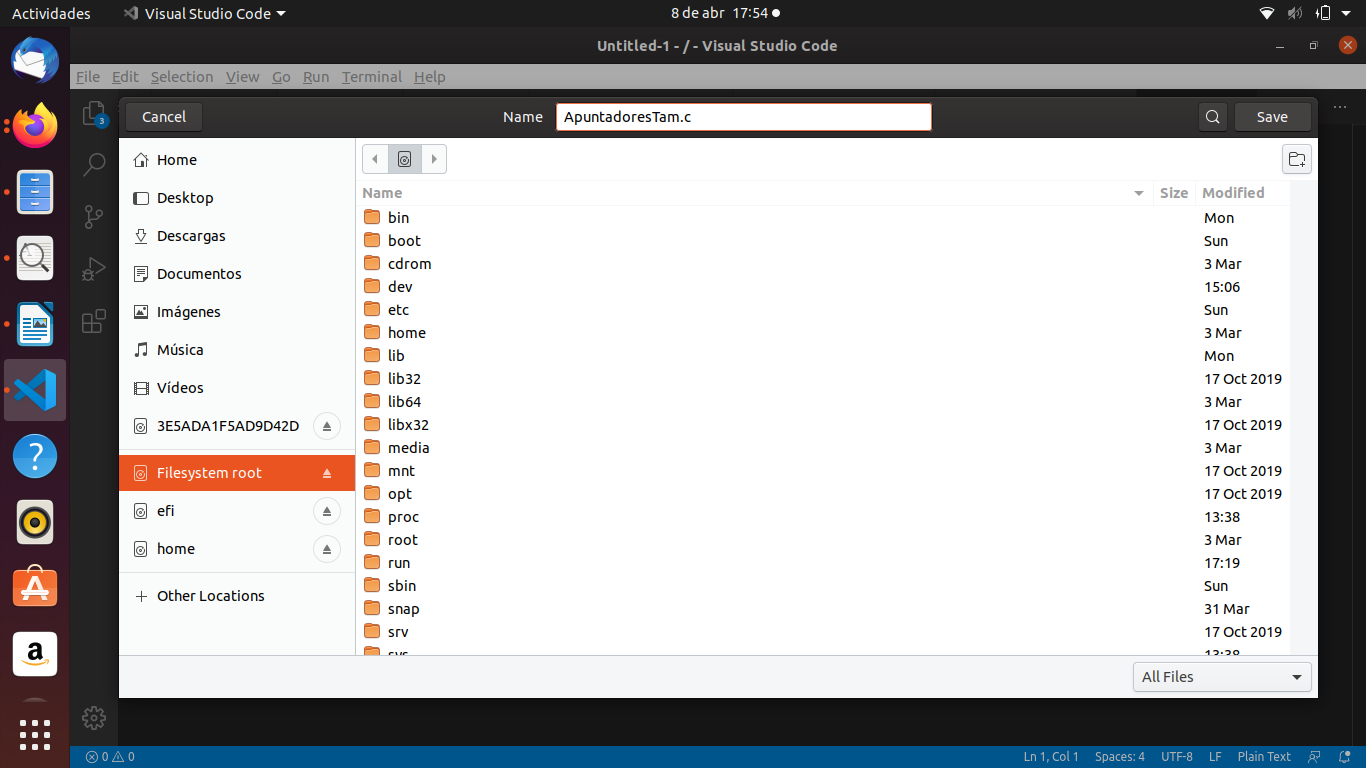
La función como se puede observar pasa dichos valores de x, y como parámetros, por medio de su dirección, en la función se tiene una variable tem, la cual almacena el valor que posee la dirección de b que en ese momento es 5, posteriormente la dirección de b almacenara el valor que almacena la dirección de a entonces ahora b vale 2, posteriormente la dirección de a almacenara el valor de la variable tem, por lo tanto ahora a vale 5, finalmente esto se imprime en el método main



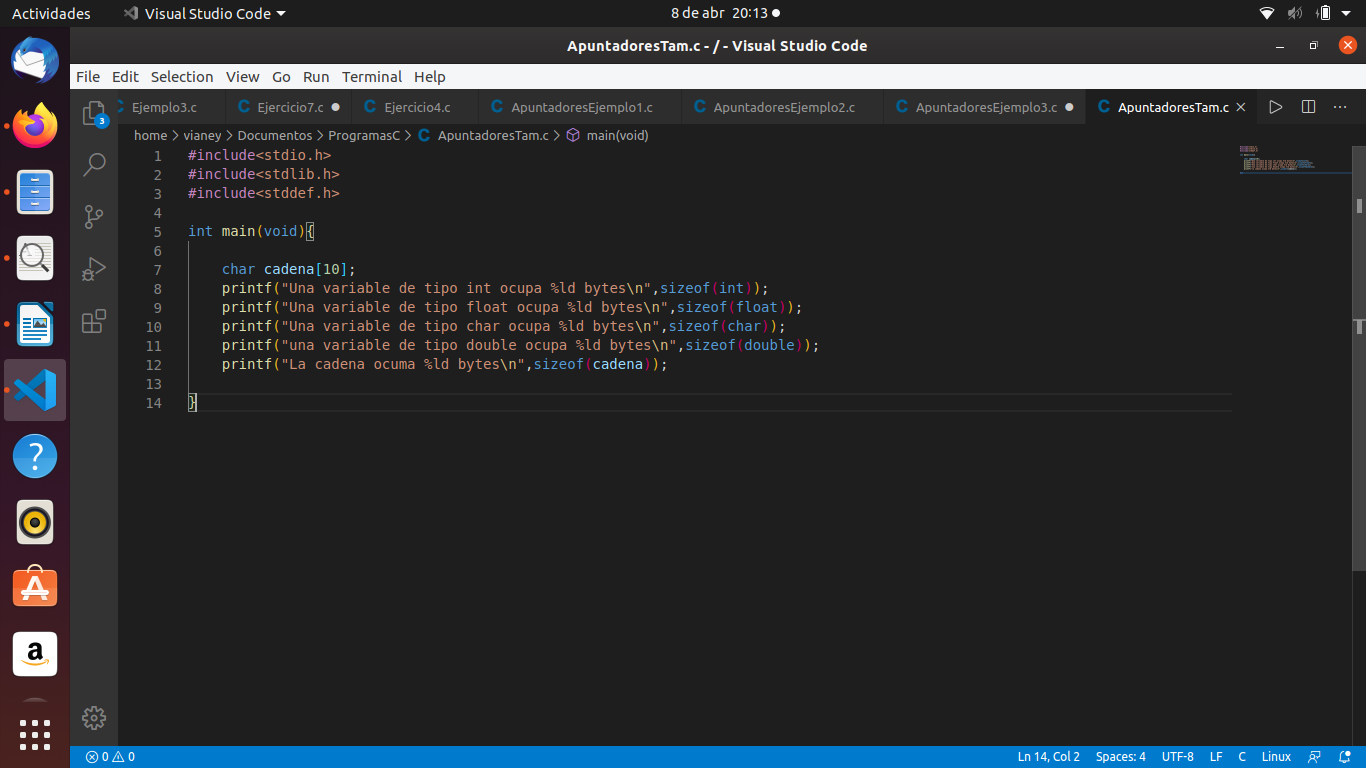
Por lo tanto los resultados se invierten al llamara a la función, siendo estos los resultados.



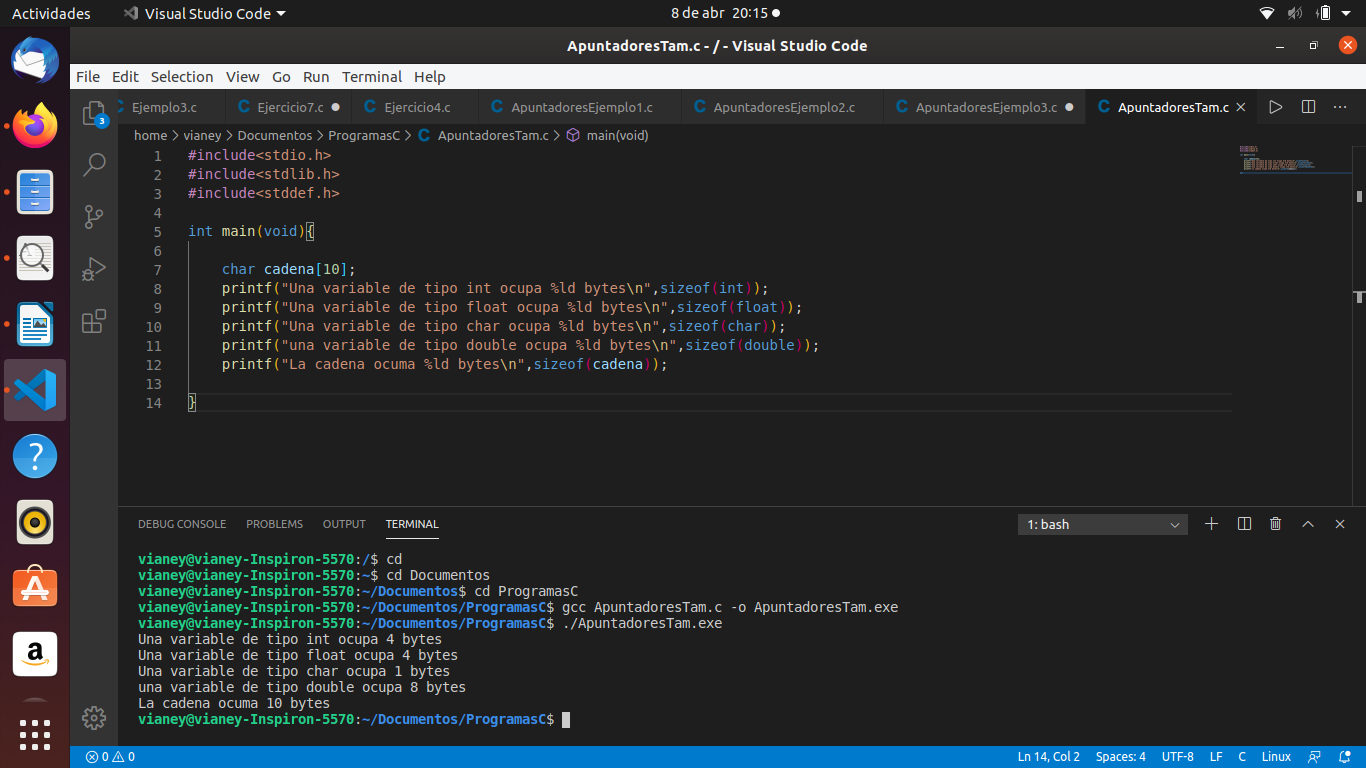
El siguiente proyecto lo nombre de la siguiente manera.



En base al código, por medio de la función sezeof(), vamos a poder saber los bytes que ocupa una variable en memoria según su tipo de dato, por lo tanto el tipo de dato se coloca dentro de dicha función y lo imprimo, otra de las partes a considerar es que para imprimir este resultado es %ld.

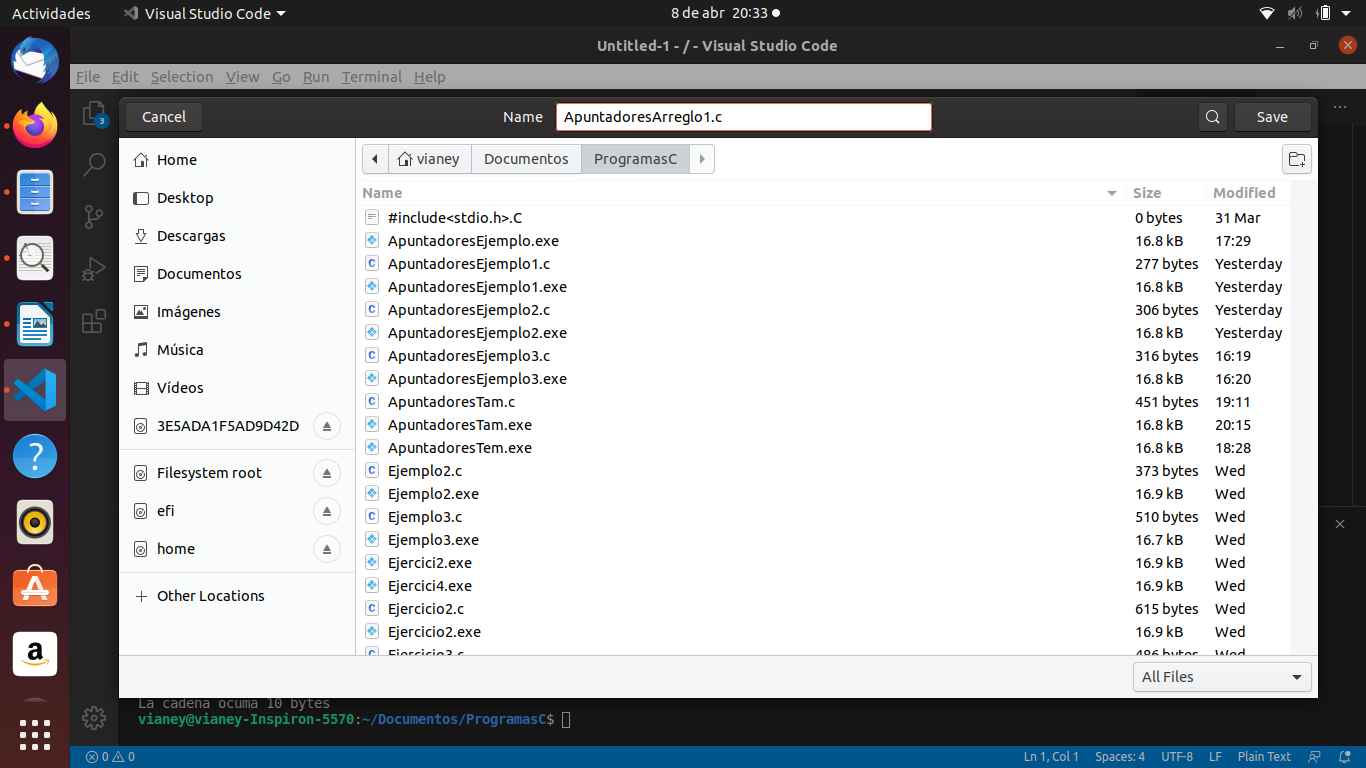


Mientras que el resultado que se muestra es el siguiente, en cuanto a la cadena como podemos ver tiene el tamaño asignado.

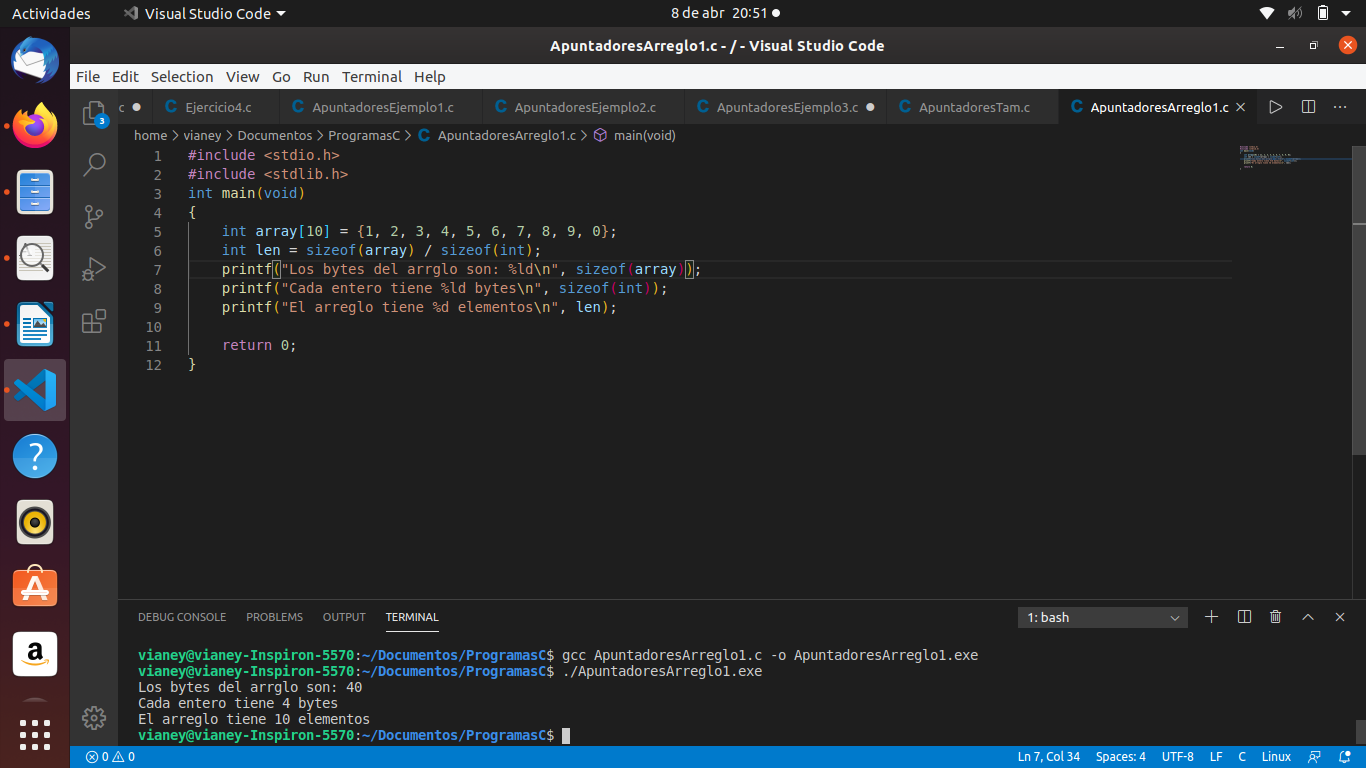


El

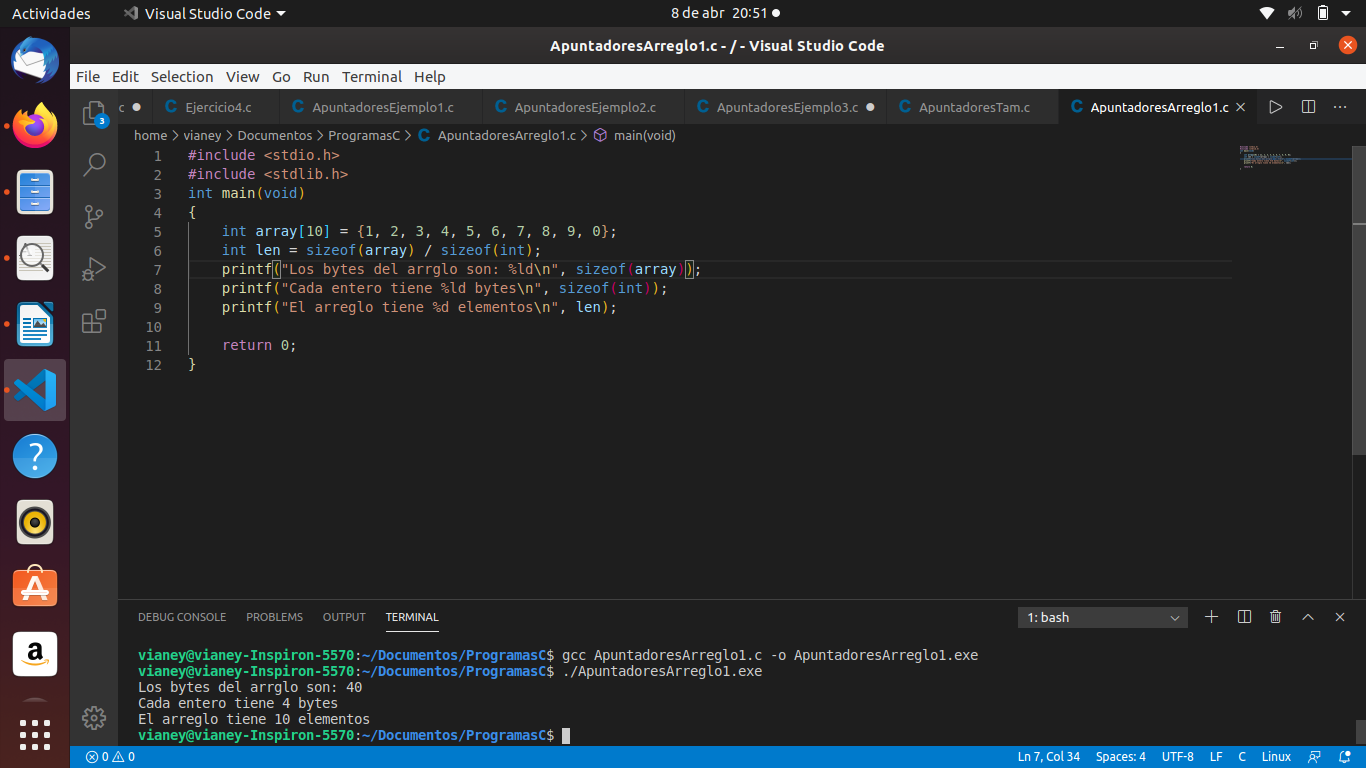
presente arreglo tendrá el siguiente nombre.



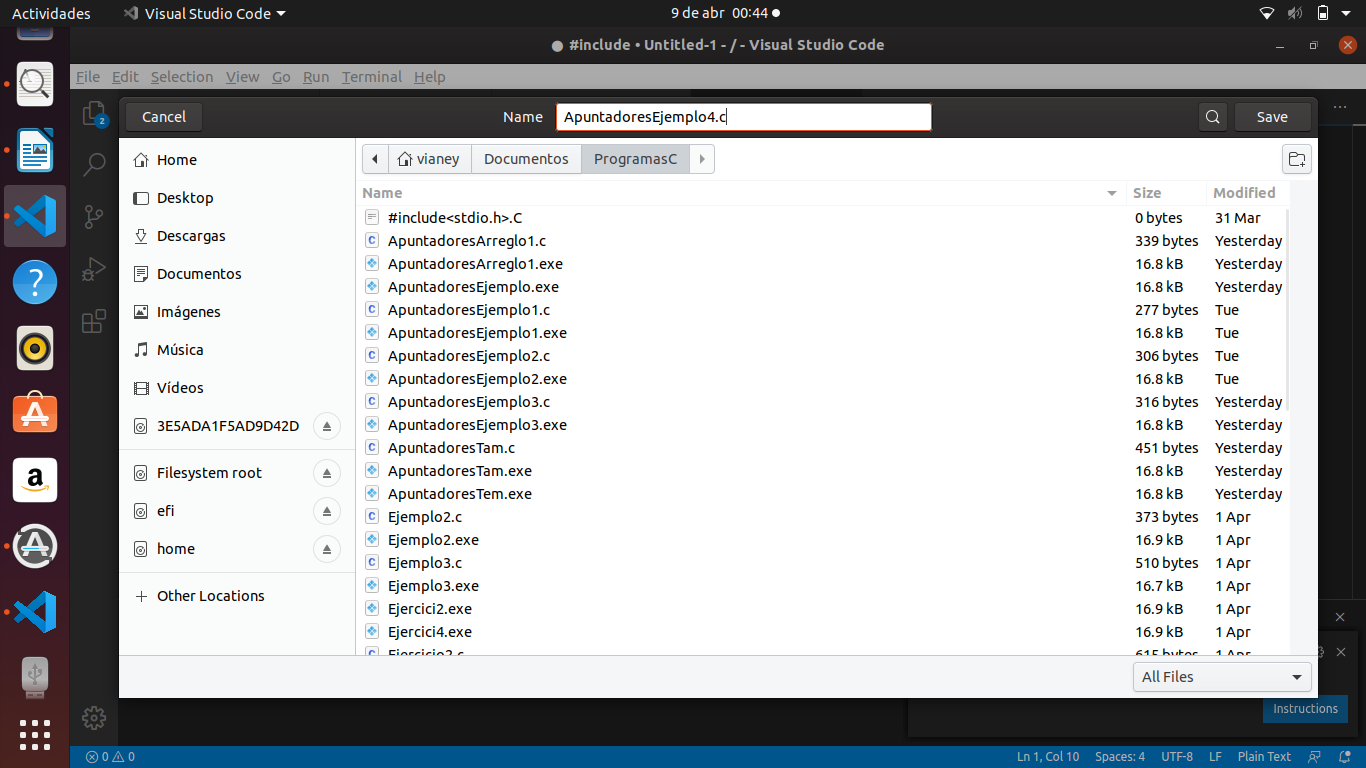
Lo primero que se realiza es declarar el arreglo como podemos ver de tamaño 10 y en cada posición se almacena un valor, después se encuentra la variable len la cual tiene el valor correspondiente a la división del tamaño que ocupa en memoria el arreglo entre el numero de bytes que ocupa cada posición del arreglo, posteriormente se manda a imprimir el espacio que ocupa en memoria el arreglo, posteriormente el número de bytes que ocupa cada posición y finalmente el numero de elementos que componen el arreglo, que corresponde al valor almacenado en la variable len.



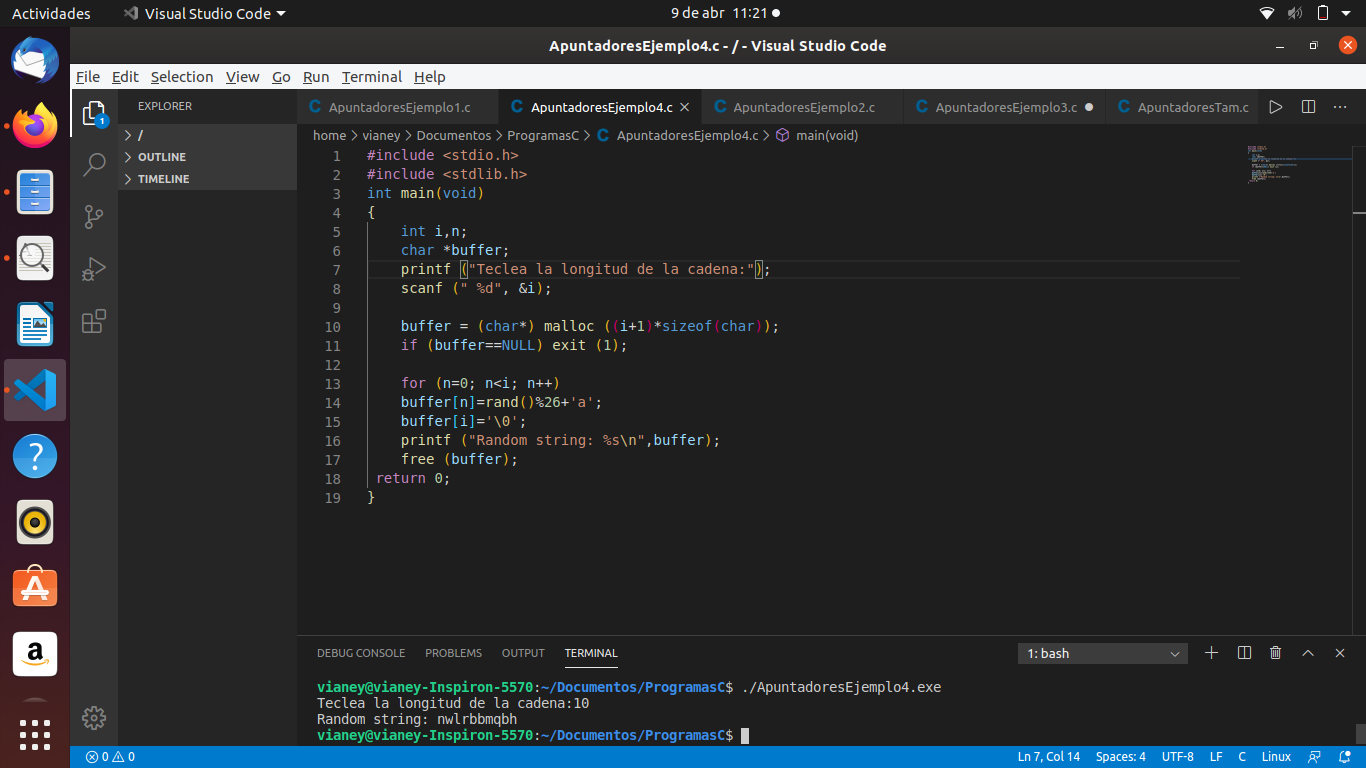
Finalmente este es el resultado que se muestra en consola.



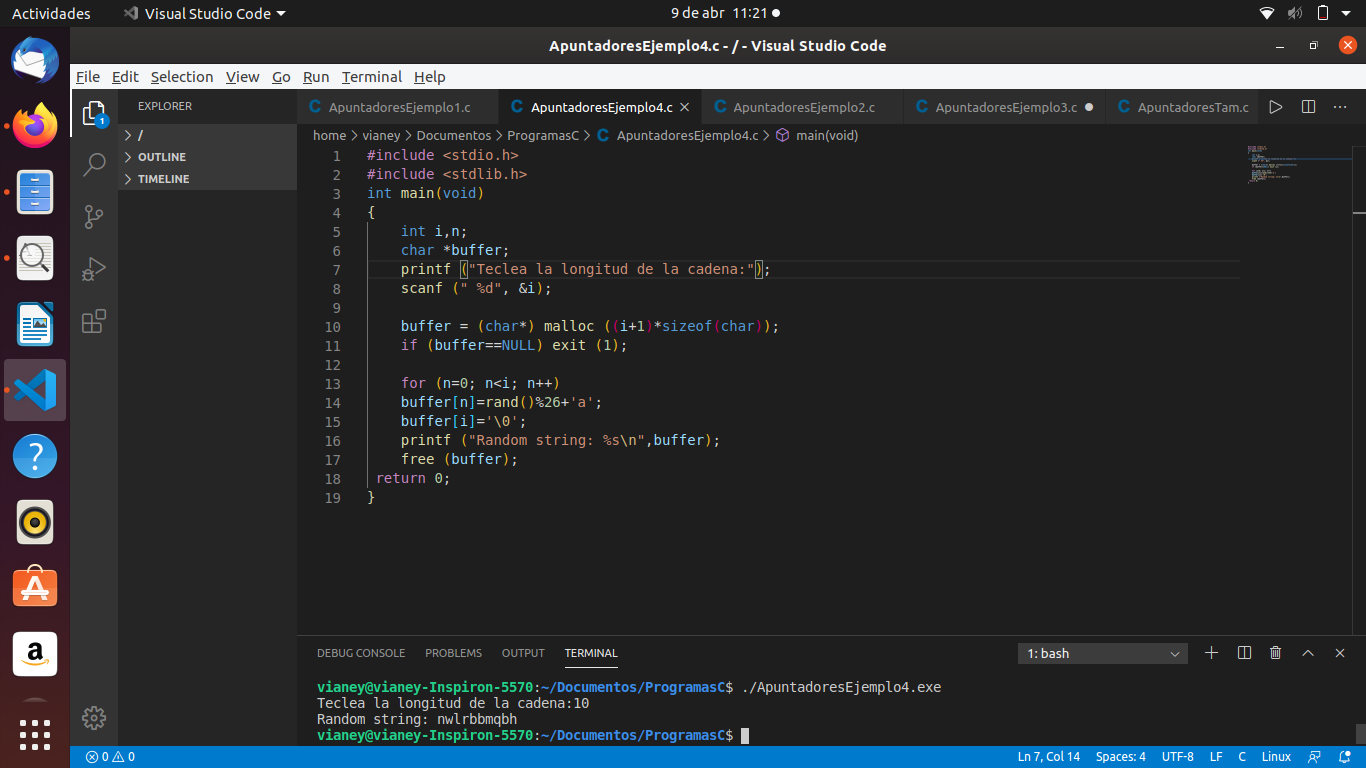
Para dar pauta creo por consecuencia el siguiente programa nombrado:



En el cual lo primero que se realiza es declarar nuestras variables, así como muestro apuntador, por medio del mensaje se indica al usuario la acción a realizar y una vez que el usuario introduce el valor este es asignado a la variable i, posteriormente este valor se reserva para el espacio en memoria que ocupara el puntero, ademas de que se le suma 1 para obtener en total 6 posiciones en la cadena buffer, de esta manera se le asigna el tamaño, posteriormente evaluamos si este es nulo, p se saldrá del programa y si no entonces en el for se llena la cadena con valores aleatorios del abecedario comenzando por la a, para ello se utiliza la función rand, y finalmente en su ultima posición la llenamos con un valor nulo con ‘\0’ , para después imprimir la cadena y por medio de la función free se librea ese espacio de memoria diatónicamente.

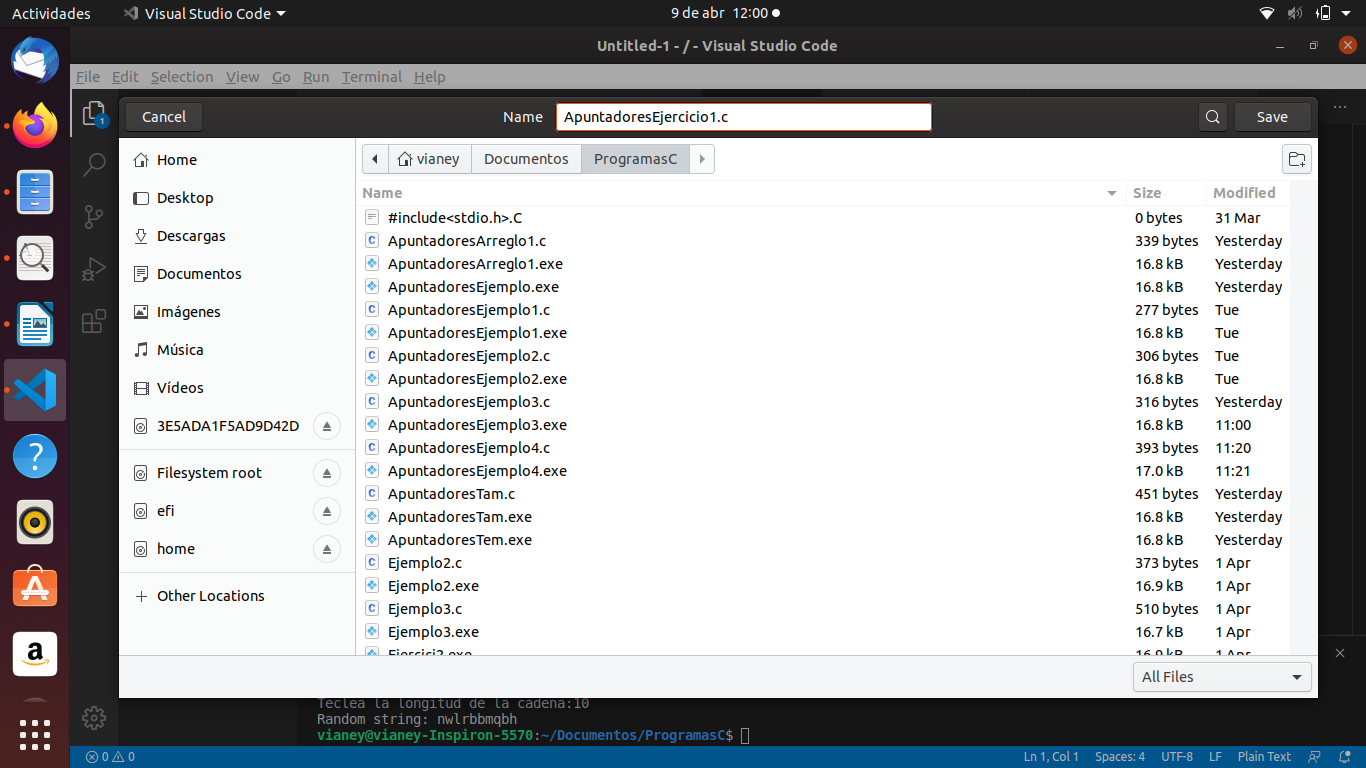


Por lo tanto al imprimir el resultado es el siguiente.

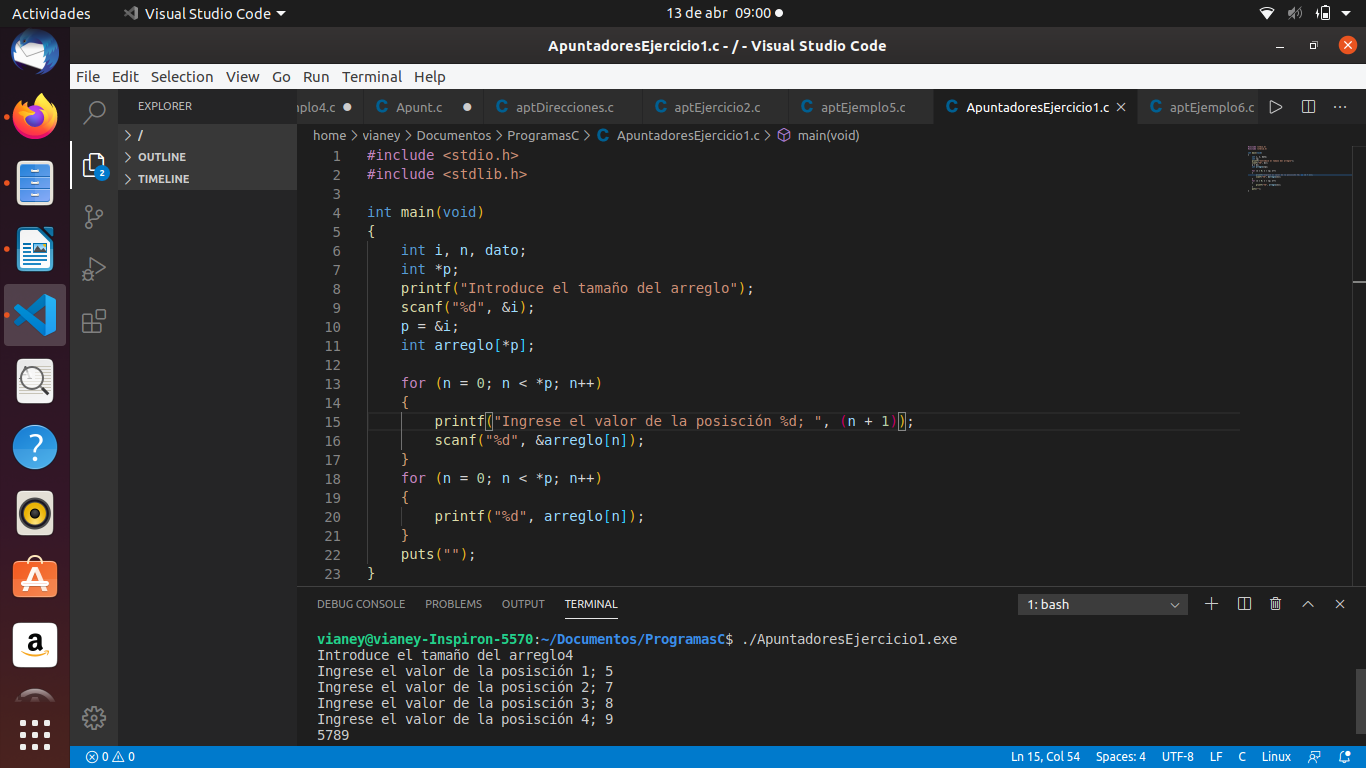


Crea un arreglo entero de tamaño x, en donde x es ingresado por teclado.

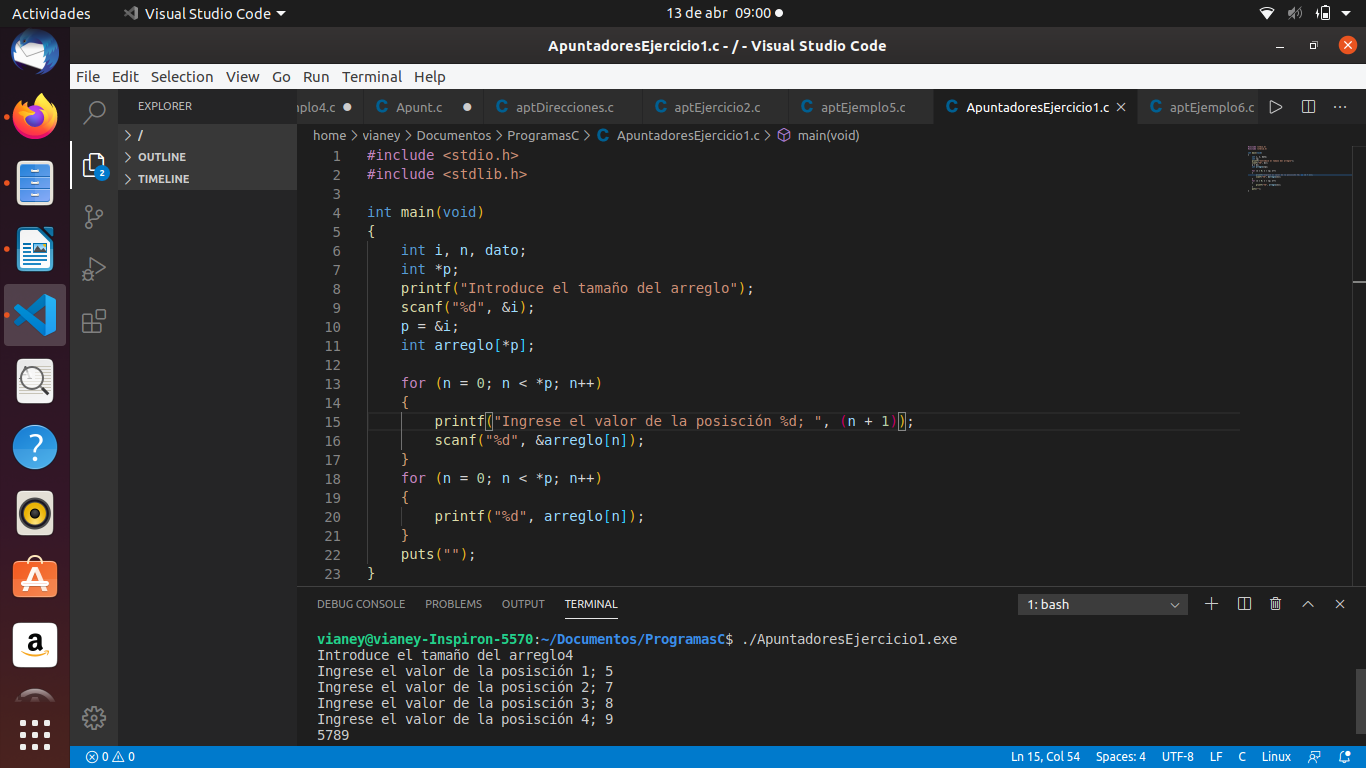
* Llena todos los elementos del arreglo con datos ingresados por el usuario.
* Muestra los valores



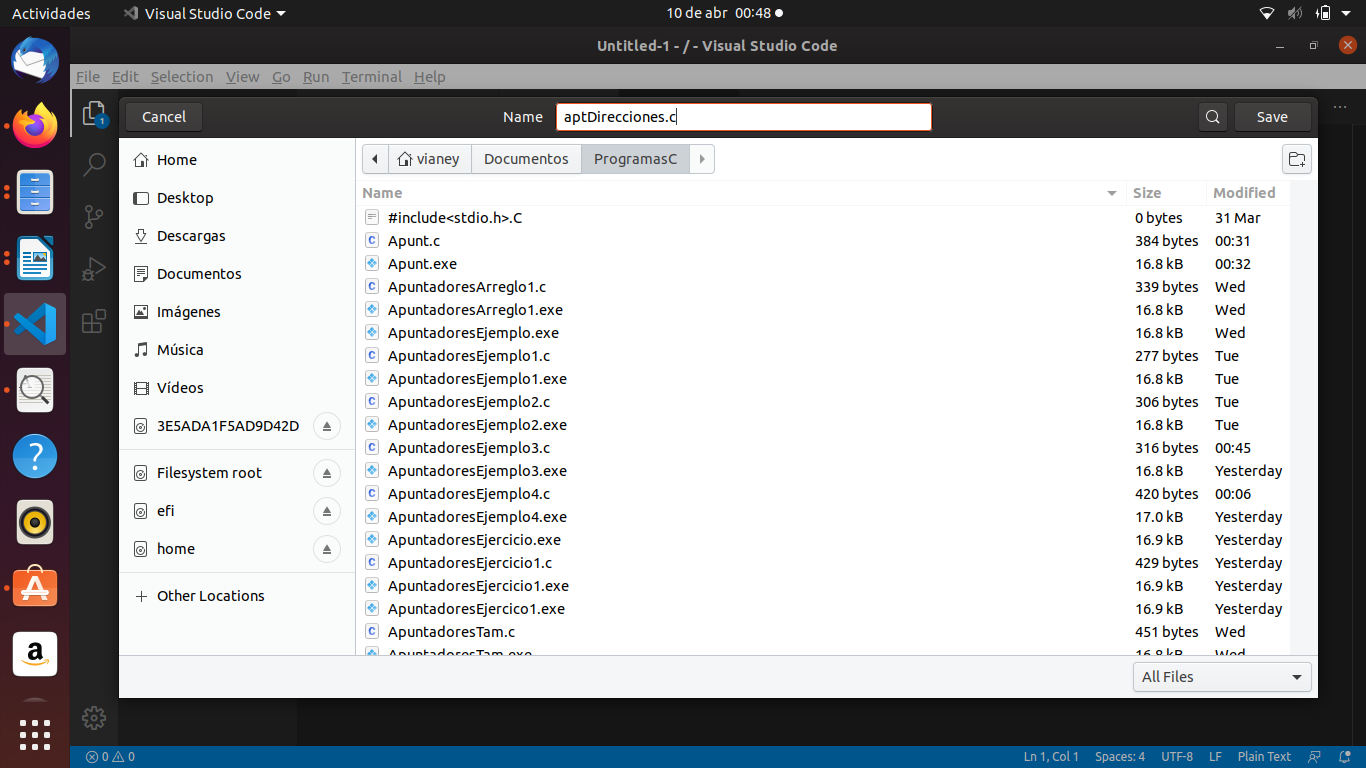
Como podemos apreciar en el código el apuntador apunta ala dirección de a variable i, por lo tanto el apuntador tiene ese valor que posee i y es por ello que en el for se coloca como parámetro, pero no se coloca la dirección, mas bien el parámetro es el valor que posee, por ello es que lleva es \* y así es como se recorre el determinado numero de veces que indica el usuario el ciclo para llenar los valores, mientras que en el segundo for se van imprimiendo.



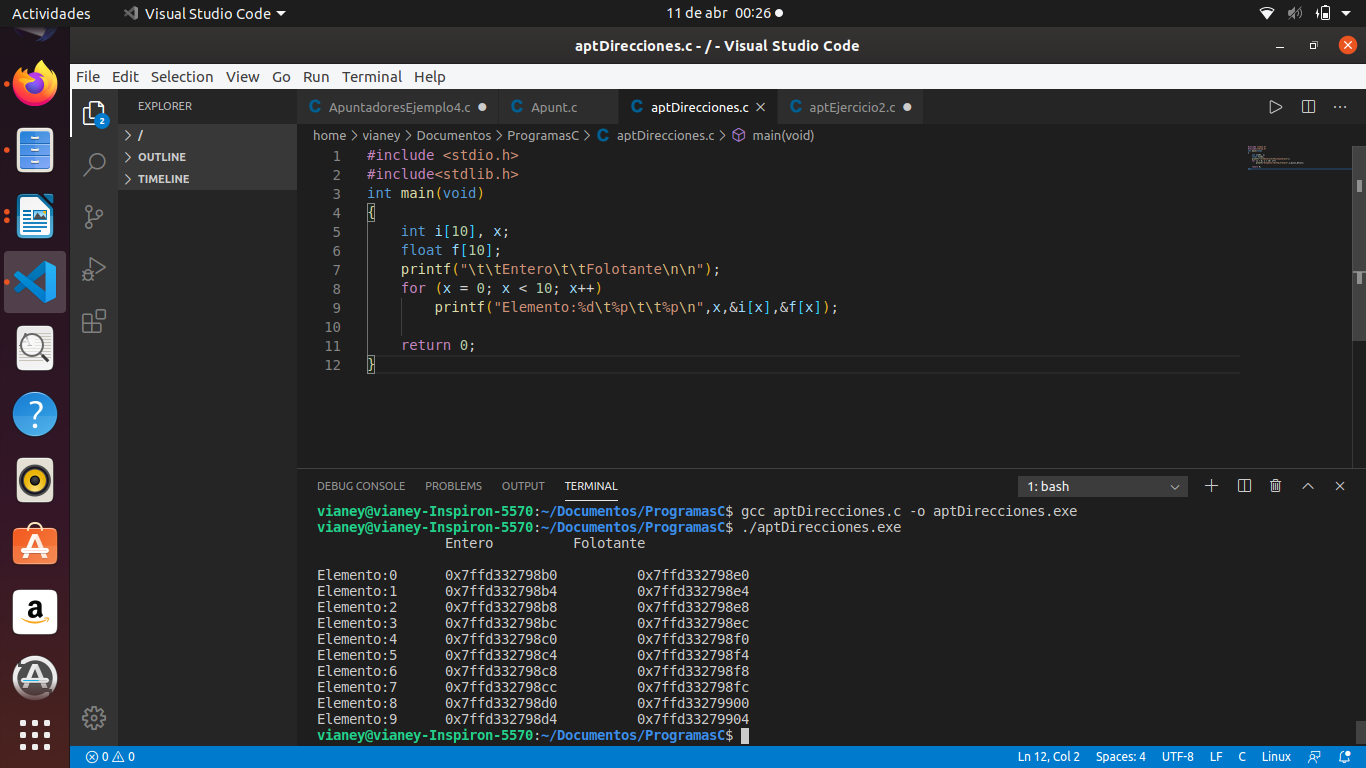
El resultado es el siguiente.



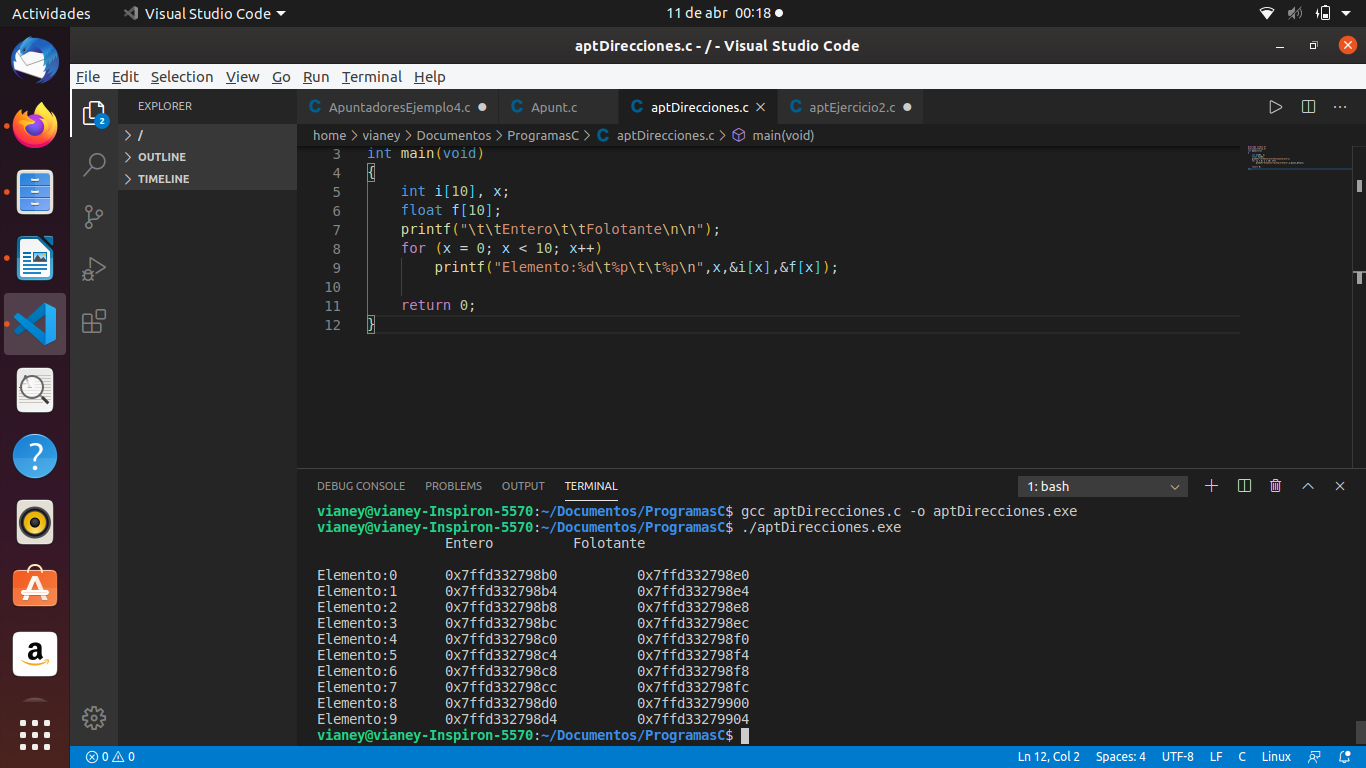
El siguiente programa toma el nombre en referencia a su función



Lo primero que se realiza es declarar los arreglos necesarios, como podemos observar tanto i como f se conforman de 10 elementos, posteriormente se manda a imprimir un mensaje para indicar las direcciones de memoria reservada para los elementos del arreglo entero y flotante, por medio del ciclo for se recorre posición por posición, empezando por la posición 0 que en este caso se utiliza la variable x con ella indicaremos el elemento que se se esta evaluando, posteriormente como se observa las posiciones de los arreglos son impresos con el amperson, puesto que estamos imprimiendo sus direcciones tanto del arreglo float como el el int.



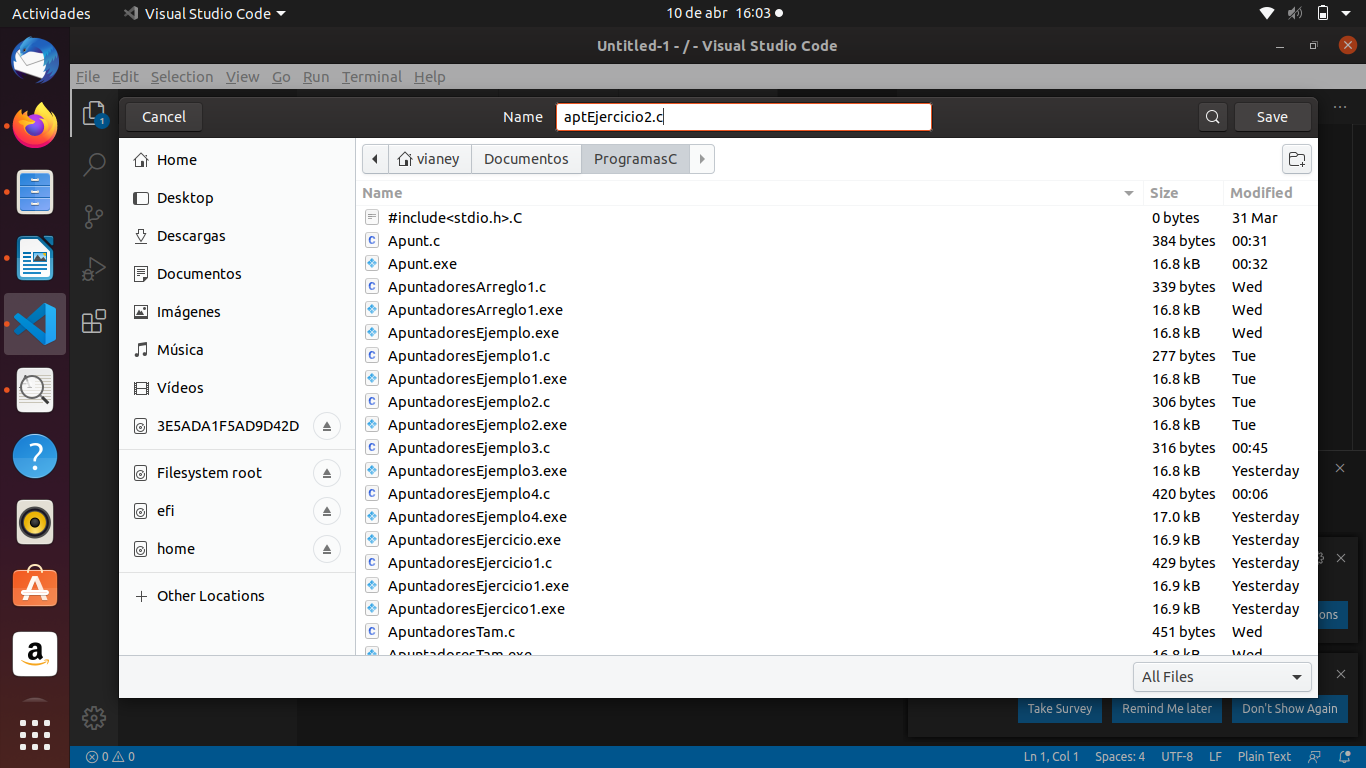
Por lo tato al imprimirse el resultado es el siguiente.



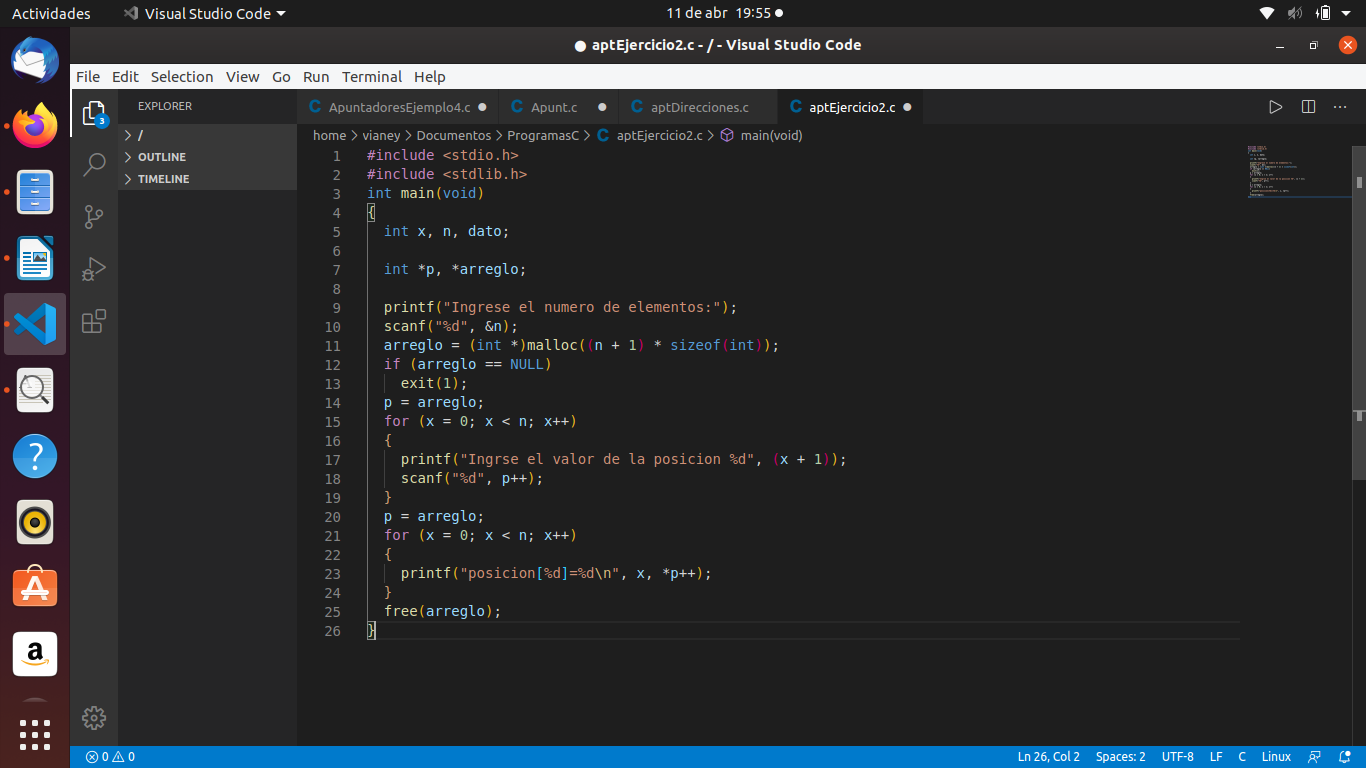
Ejercicio2

* Crea un arreglo entero de tamaño x, en donde x es ingresado por teclado.
* Llena todos los elementos del arreglo con datos ingresados por el usuario usando apuntadores.

Nombro mi proyecto.



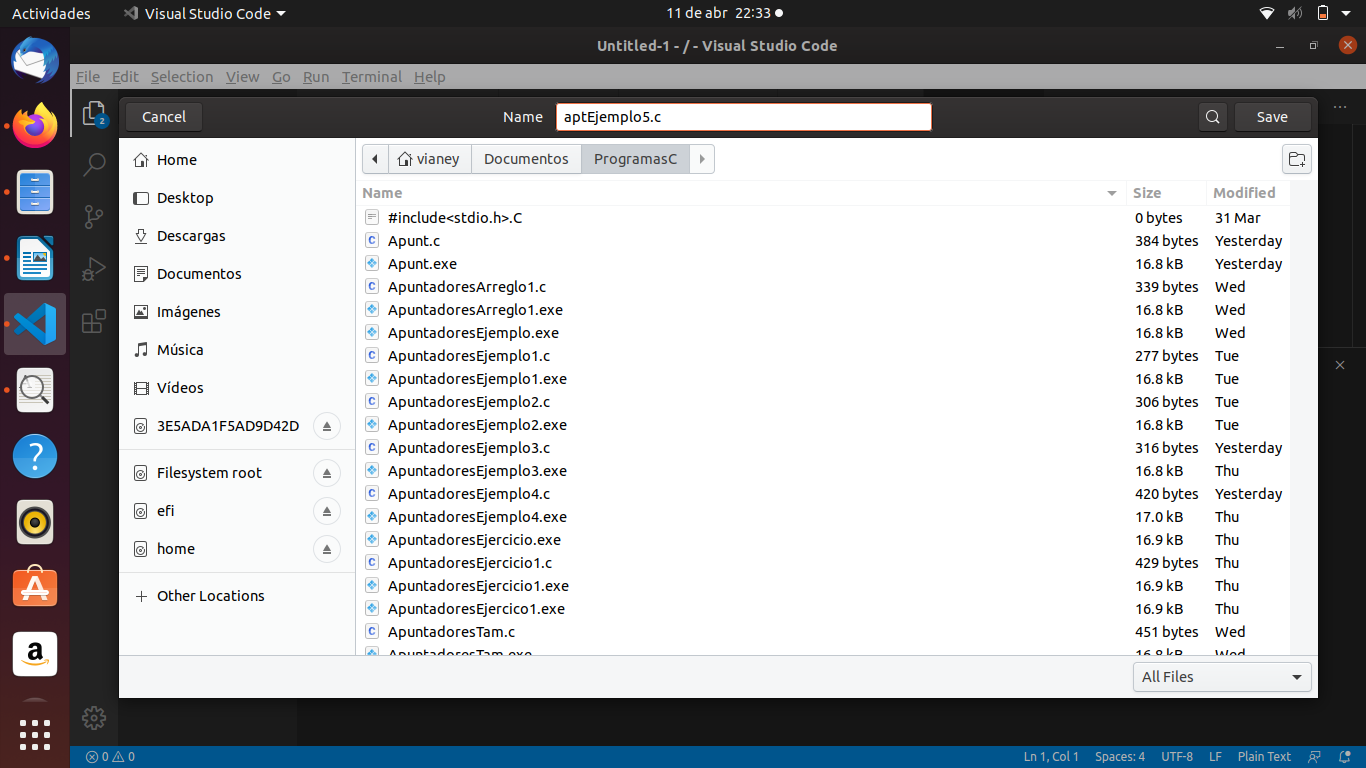
Lo primero es declara las variables y punteros, como se puede observar se solicita el numeró de elementos que desea almacenar el usuario, posteriormente este valor se asigna al apuntador llamado arreglo pero es multiplicado por el número de bytes que ocupa cada elemento, por lo tanto este es el espacio en memoria que se reserva diatónicamente, después asignaremos que el puntero recorrerá estas pociones por medio del ciclo for, como podemos ver se solicita el valor a ingresar y lo almacena, posteriormente avanza una posición y así sucesivamente, mientras que en el segundo for se imprimen los valores ingresados en esa dirección y finalmente liberamos el espacio reservado en memoria para el puntero llamado arreglo.



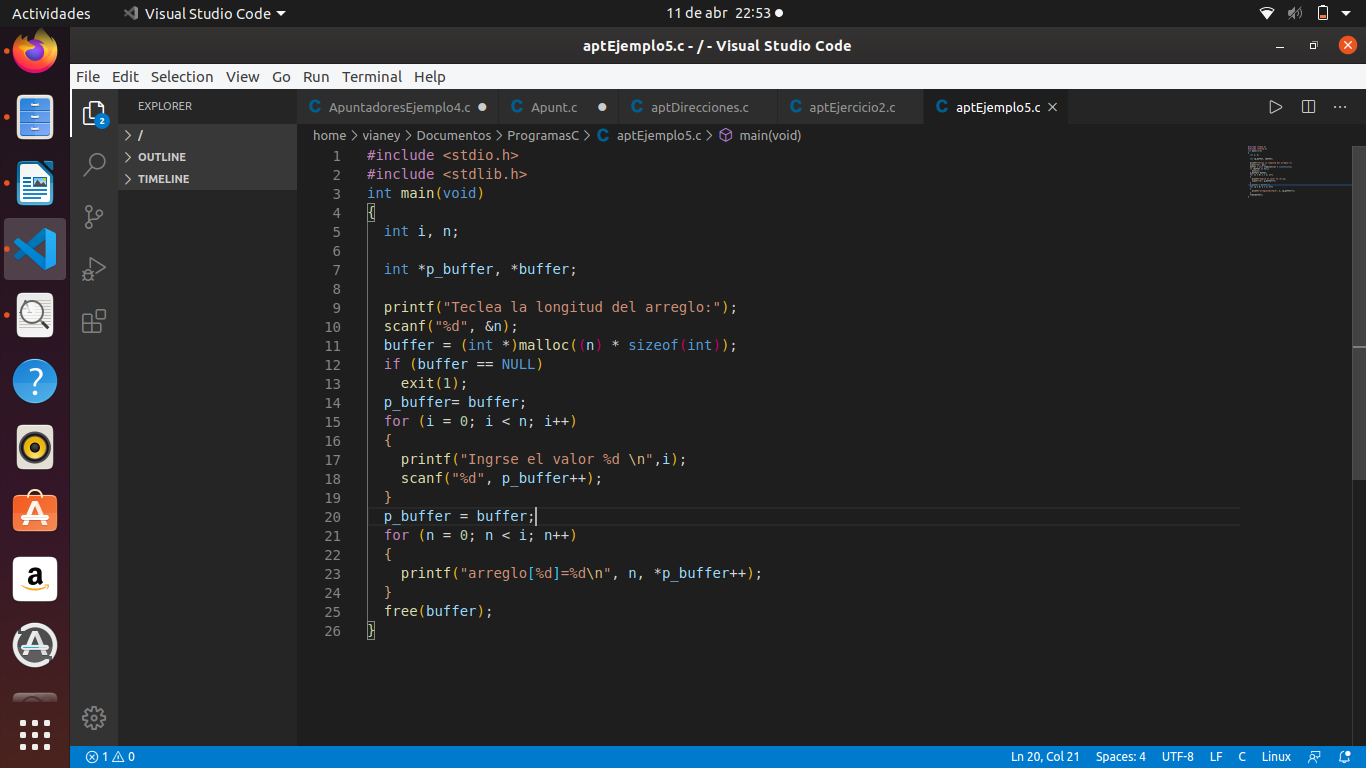
Al compilar el código se ejecutan las instrucciones programadas mostrando los siguientes resultados.



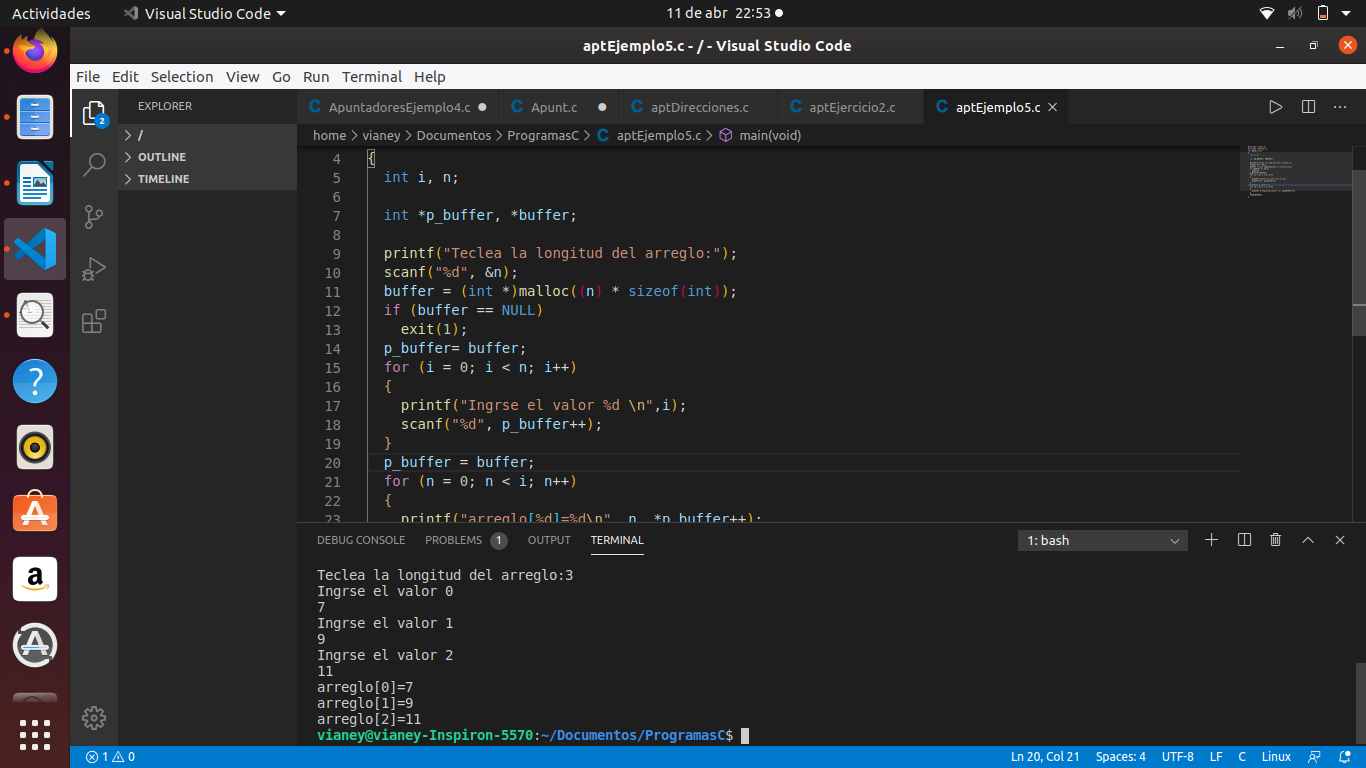
El siguiente ejemplo es nombrado de la siguiente manera.



en el se presenta realiza la misma función al anterior, a diferencia de algunos mensajes, pero como se puede observara también el arreglo se llena por medio de punteros, avanzando de posición en posición por medio de las direcciones a las que apunta el apuntador .

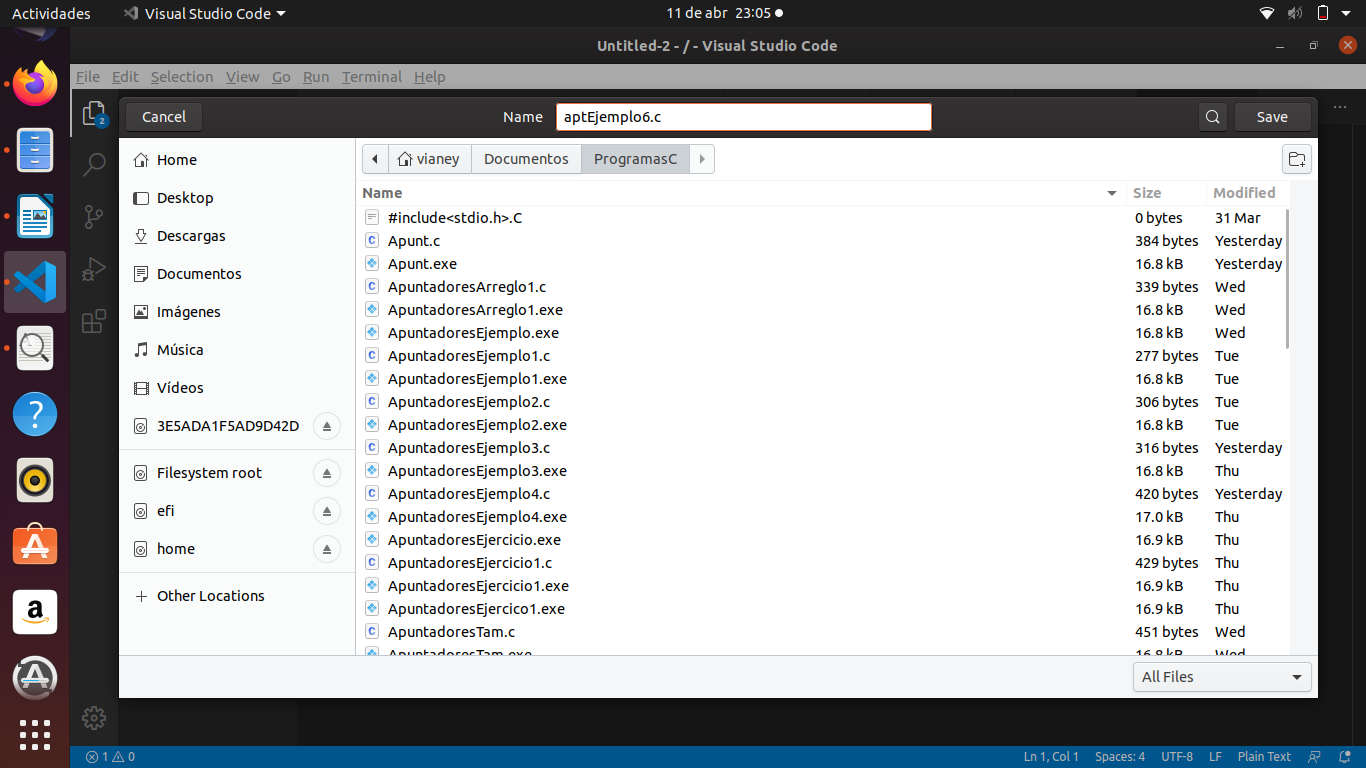


El resultante es el siguiente.



* 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.

Nombro mi archivo de tipo c de esta manera.

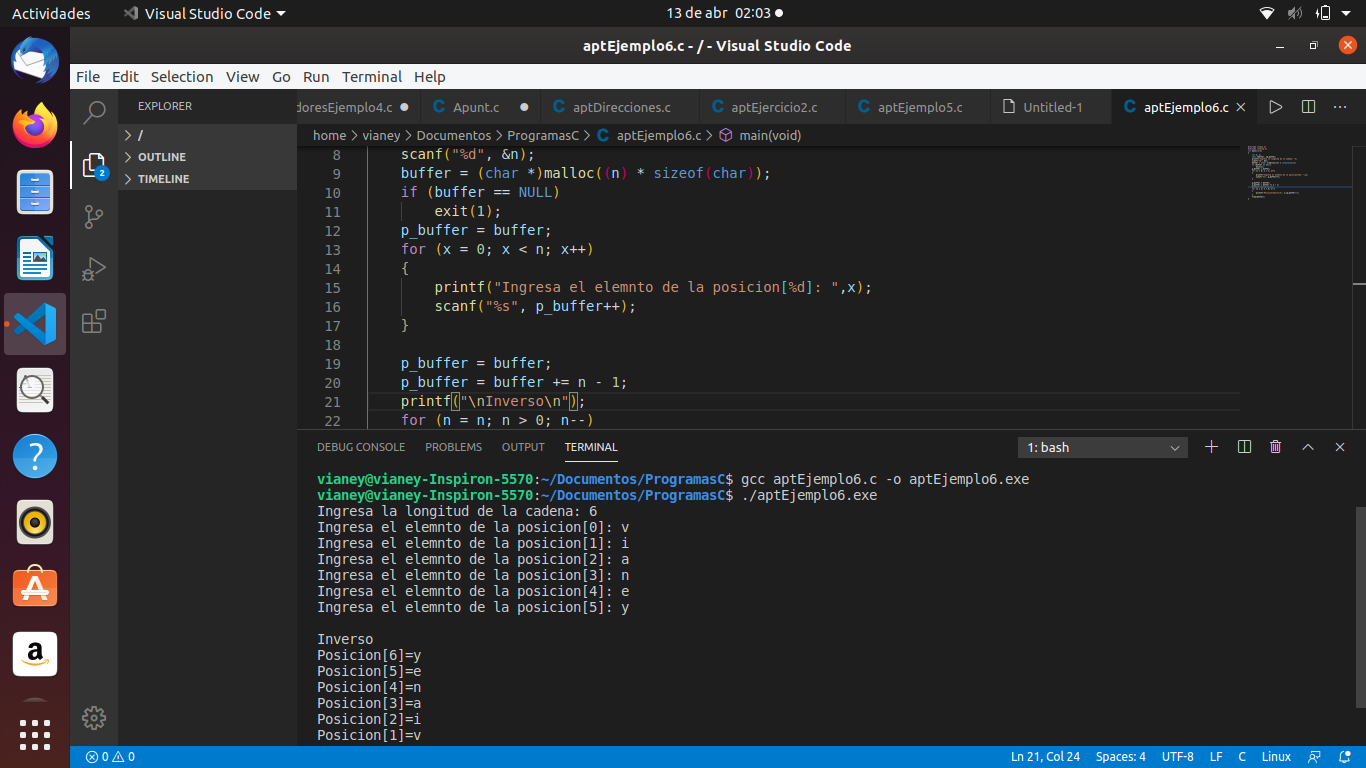


Primero declaro los apuntadores que utilizare, así como las variables, como se puede apreciar, posteriormente se solicita la longitud de dicha cadena, este valor es asignado a la variable n, después dicho valor se multiplica por el numero de bytes que se requieren para un dato de tipo char, y dicho valor se reserva en la memoria para el apuntador buffer, evaluamos por medio del if para saber si es nulo o no, después de ello se asigna el mismo espacio en memoria al apuntador p\_buffer. El cual va recorriendo sus posiciones par llenarlas con los caracteres por medio del ciclo for, mientras que para el segundo for como podemos ver se coloca buffer+=n-1, lo cual indica que se comenzara en la penúltima posición a imprimir n , esto debido a que se comienza a imprimir desde atrás.

Este for al imprimir de forma inversa, n ira decrementando, y así mismo el puntero, ira decrementando de posición,así finalmente el resultado se imprime indicando la posicion y el elemento.



Finalmente al ejecutar e introducir los caracteres se obtiene la cadena inversa



**V. Conclusiones:**

Por medio de la presente lamina, aprendimos para que sirve un apuntador, de igual manera su función, al tener la dirección de otra variable, incrementos y decrementos, conocimos la sintaxis para poder utilizarlos. La parte mas relevante a mi parecer fue utilizar un apuntador para recorrer distintas direcciones, como lo hicimos en los ejercicios de los arreglos, ademas de ello la reserva de memoria para almacenar datos, ya que aprendí la función de malloc para reservar diatónicamente memoria y después liberar ese espacio por medio de free().

A demás de ello un apuntador al tener la dirección de memoria de otra variable, también le es posible modificar el valor de dicha variable, esto lo puede lograr de igual manera dentro de una función, como lo fue en el ejemplo de la función intercambio.