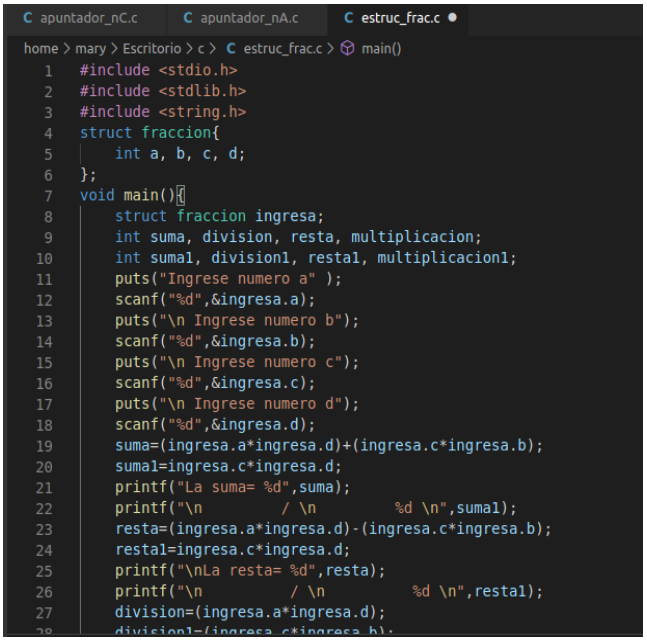
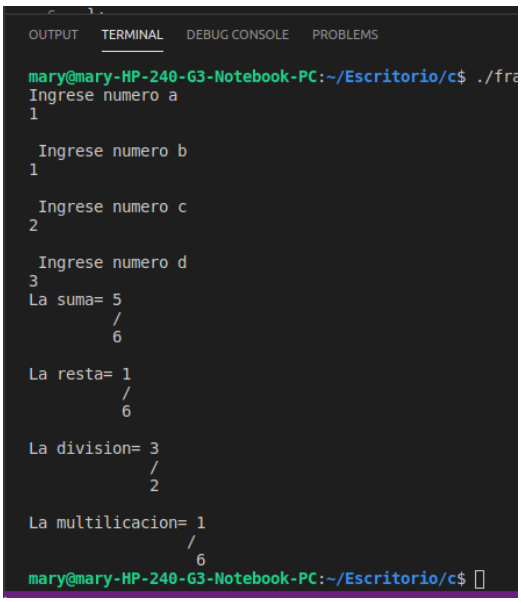
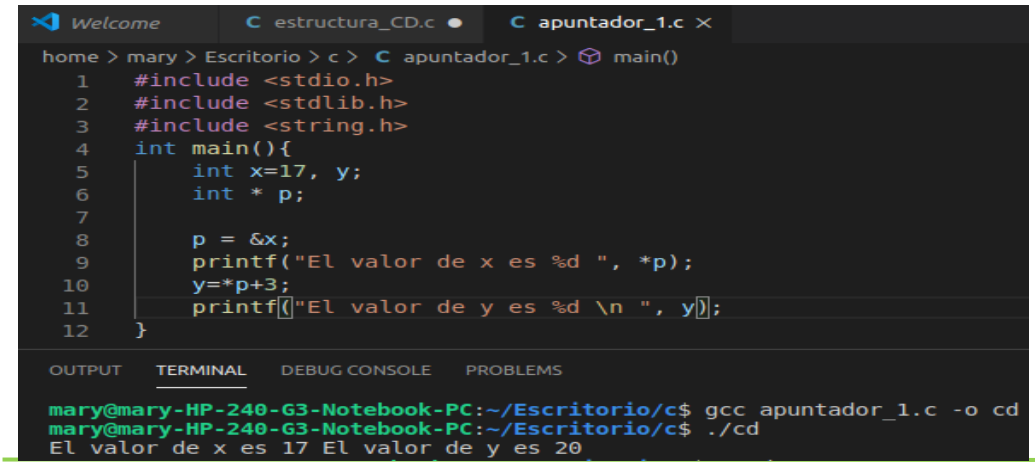
Nombre de la práctica	Apuntadores			No.	15
Asignatura:	MÉTODOS NUMÉRICOS	Carrera:	ISIC	Duración de la práctica (Hrs)	6

I. Competencia(s) específica(s):
II. Lugar de realización de la práctica (laboratorio, taller, aula u otro):
III. Material empleado:
Visual studio code
IV. Desarrollo de la práctica:

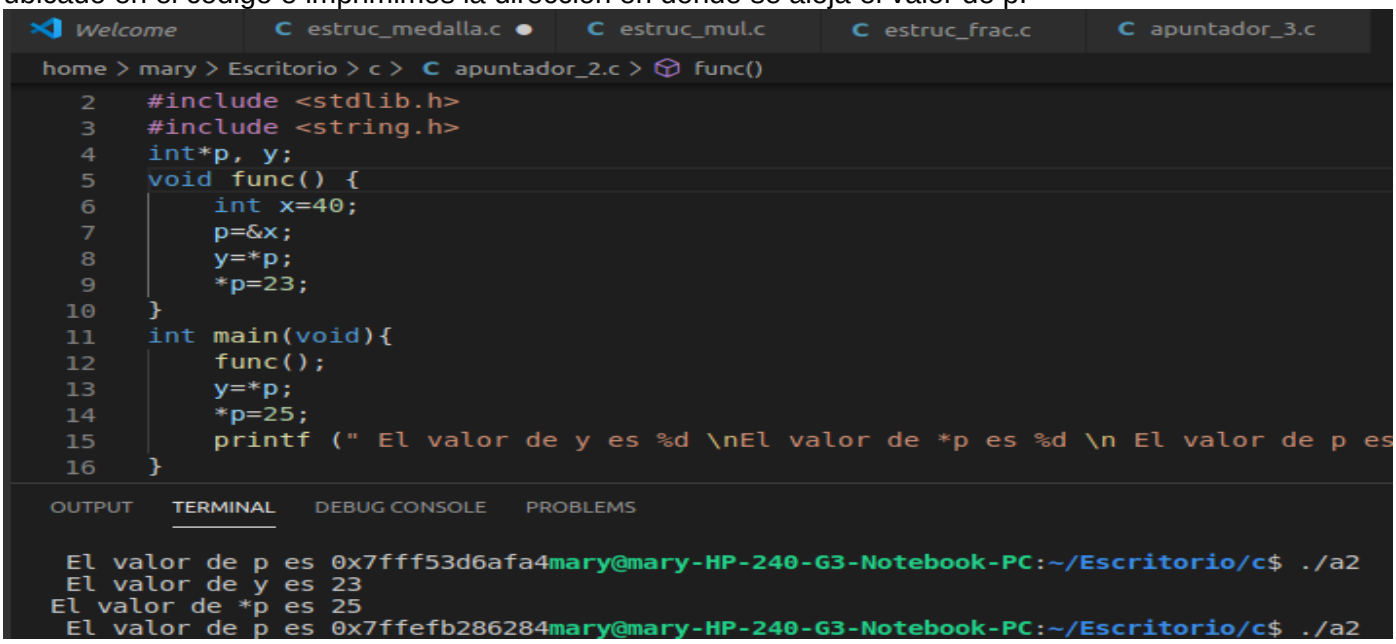




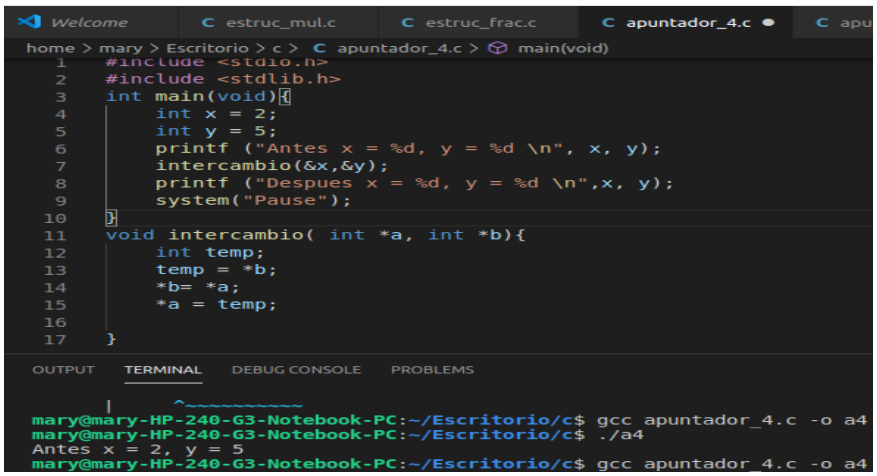
Un puntero es un objeto que apunta a otro objeto. Es decir, una variable cuyo valor es la dirección de memoria de otra variable. Para declarar un apuntador se especifica el tipo de dato al que apunta, el operador '*', y el nombre del apuntador. Creamos un apuntador de tipo entero entero, donde a este le asignamos un valor de una variable de igual entero, imprimimos un mensaje en pantalla en el que el apuntador es 17 puesto que antes se declara el valor, después hacemos una suma del apuntador con el valor de 17 mas 3 que este resultado se guarda en y e imprimimos ese resultado en otro mensaje.



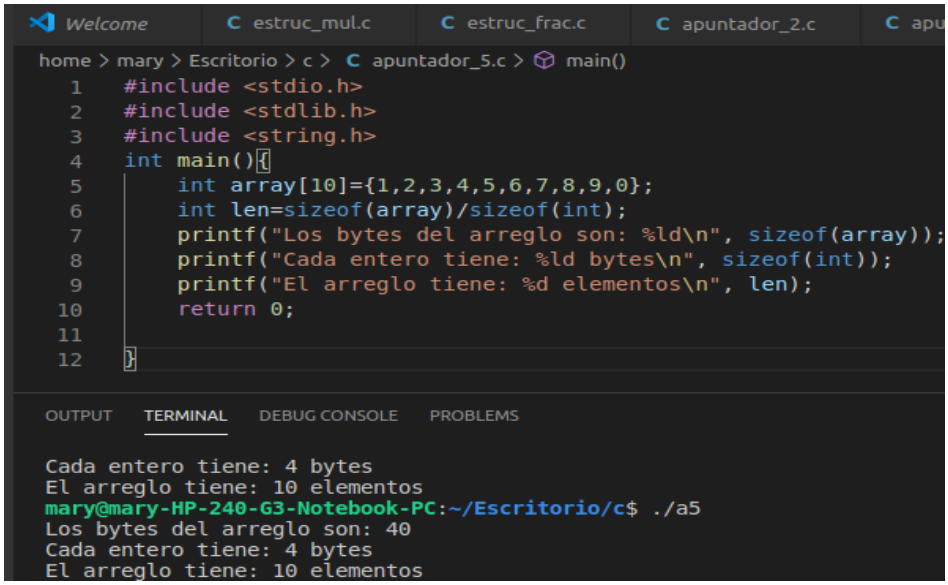
Cuando se declara un apuntador, posee un valor cualquiera que no se puede conocer con antelación. Después de que ha sido inicializado, la dirección que posee puede dejar de ser válida por que la variable asociada termina su ámbito o por que ese espacio de memoria fue reservado dinámicamente.
Declaramos el apuntador, lo que se muestra al usuario es el valor de y donde es igual al apuntador de valor 23, por eso se muestra el numero 23, el apuntador de nombre p es 25 por la parte en como esta ubicado en el código e imprimimos la dirección en donde se aloja el valor de p.



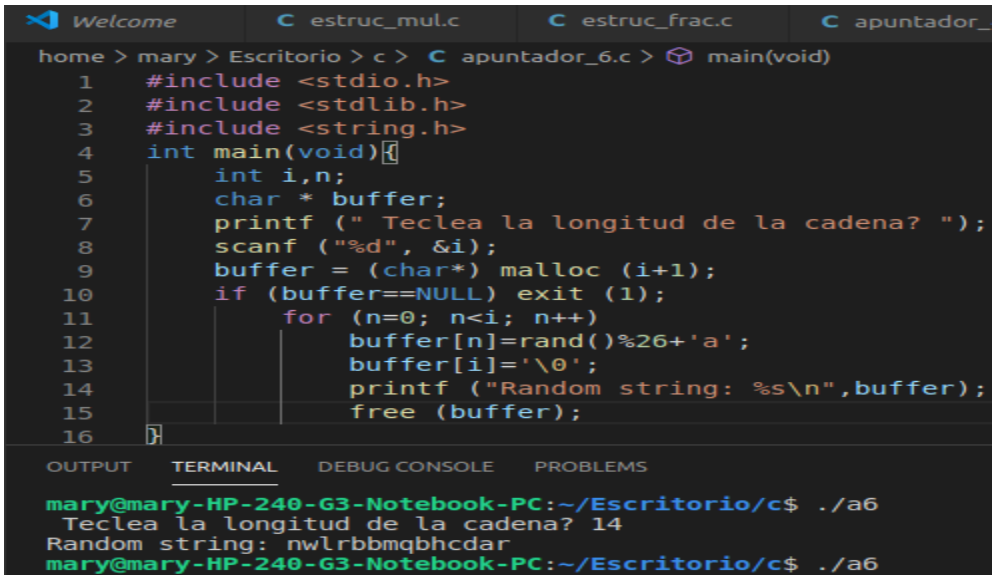
Dado que un apuntador es una variable que apunta a otra, fácilmente se puede deducir que pueden existir apuntadores a apuntadores, y a su vez los segundos pueden apuntar a apuntadores.
Es posible declarar apuntadores a constantes. De esta manera, no se permite la modificación de la dirección almacenada en el apuntador, pero si se permite la modificación del valor al que apunta.
Dentro de la función la dirección se utiliza para acceder al argumento real. En las llamadas por referencia cualquier cambio en la función tiene efecto sobre la variable cuya dirección se pasó como argumento.



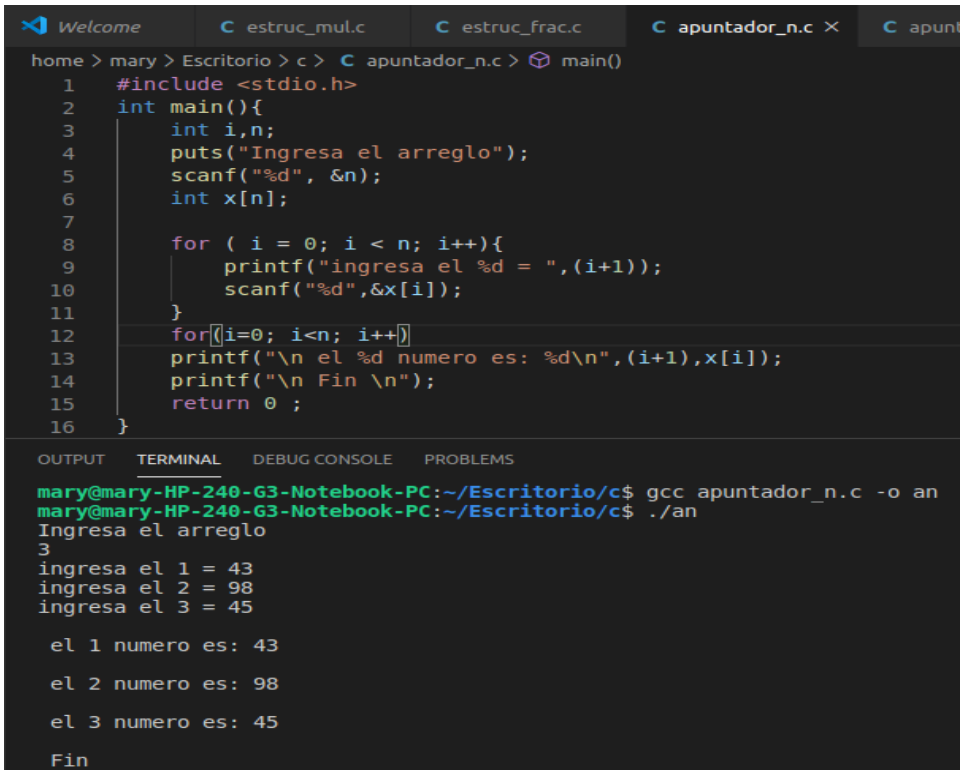
La función sizeOf() devuelve el tamaño en bytes que ocupa un tipo o variable en memoria.
Por ello declaramos un array con los valores declarados, tomamos la longitud en bytes del array entre los bytes que es un entero imprimimos los bytes del array, cuantos enteros tiene un byte, cuantos elemento tiene un array.



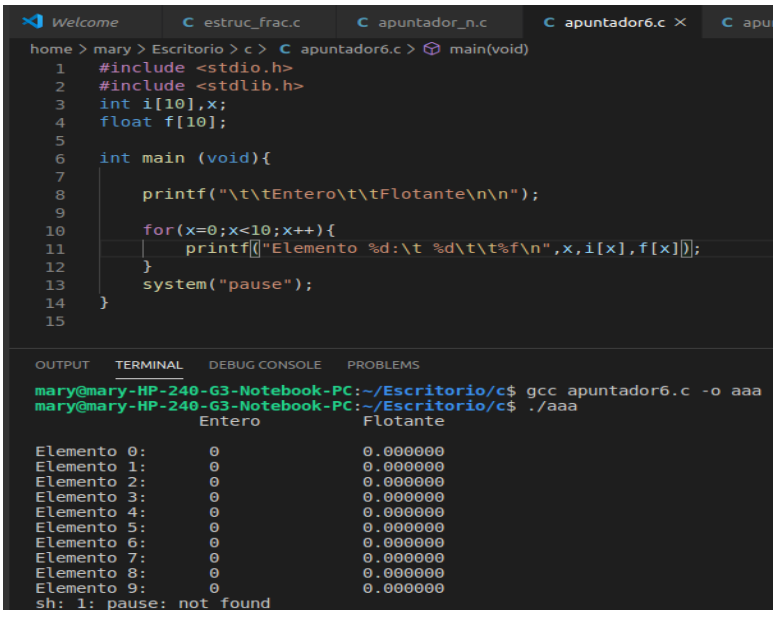
Los programas pueden crear variables globales o locales. Las variables declaradas globales en sus programas se almacenan en posiciones fijas de memoria (segmento de datos) y todas las funciones pueden utilizar estas variables.
La función free() permite liberar la memoria reservada a través de un apuntador.
void free (void* ptr);
La función malloc() reserva memoria y retorna su dirección, o retorna NULL en caso de no haber conseguido suficiente memoria.
Void *malloc(size_t tam_bloque)



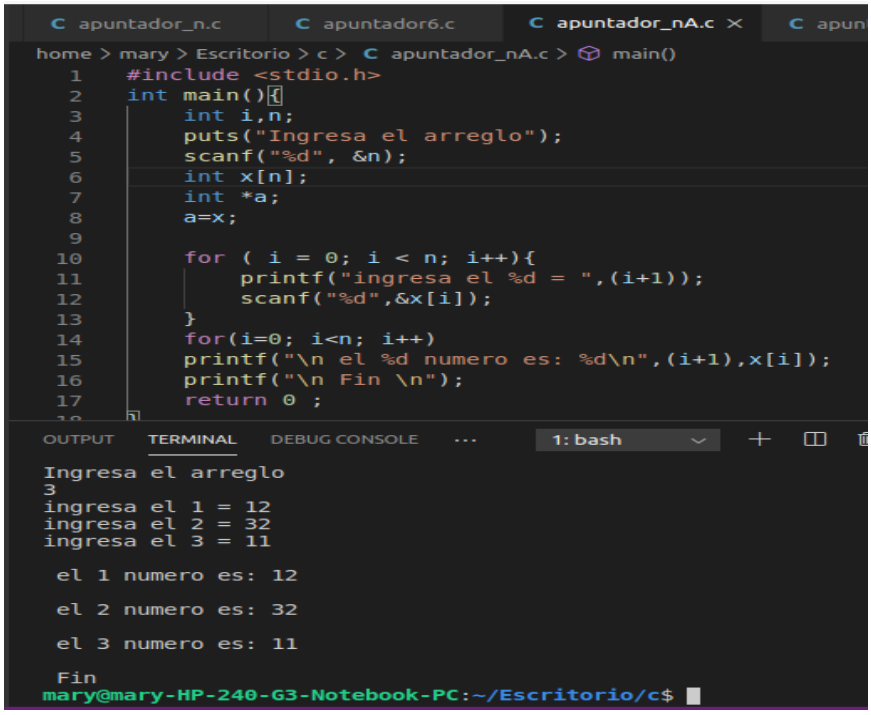
Crea un arreglo entero de tamaño x, en donde x es ingresado por teclado.
Llena todos los elementos del arreglo con datos ingresados por el usuario.
Muestra los valores



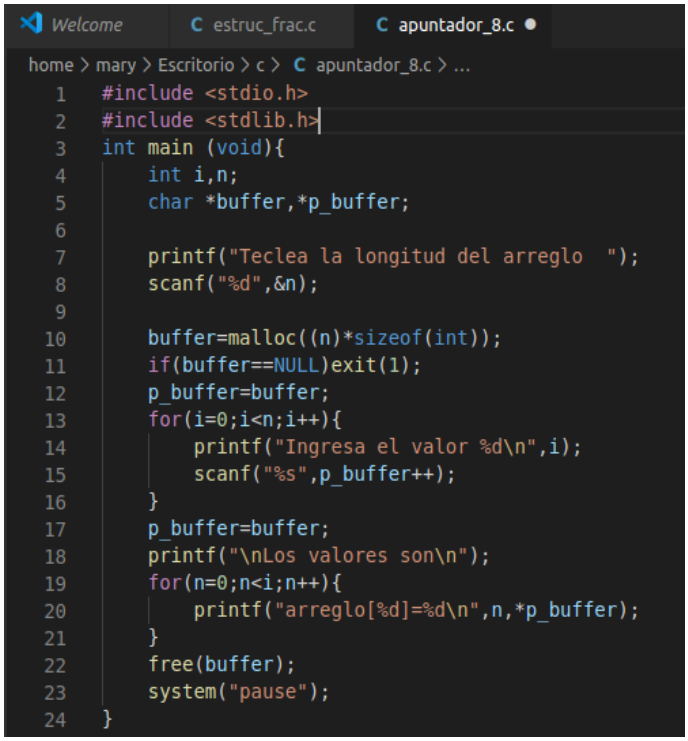
En este código tenemos una muestra donde se declara un arreglo de tipo entero en el que se imprime hasta que se 10, ya que muestra su tipo de dato entero y flotante.
Por ello el mensaje lo muestra recordemos que para ello los valores deben de estar de acuerdo a su tipo de dato %d de tipo entero, %f tipo flotante.

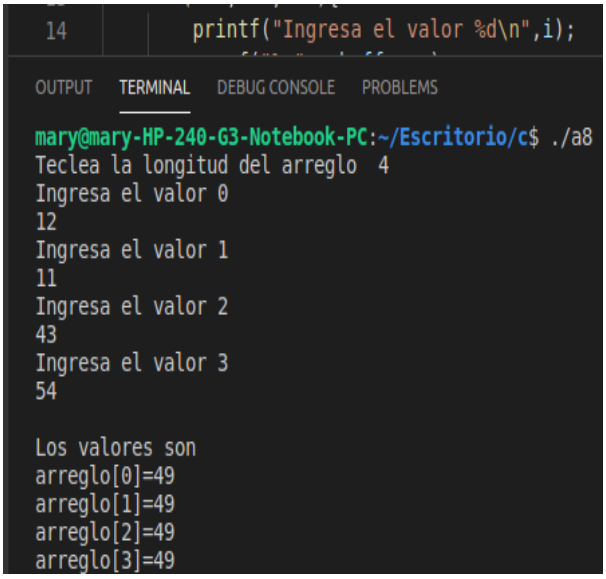


Crea un arreglo entero de tamaño x, en donde x es ingresado por teclado.
Llena todos los elementos del arreglo con datos ingresados por el usuario usando apuntadores.

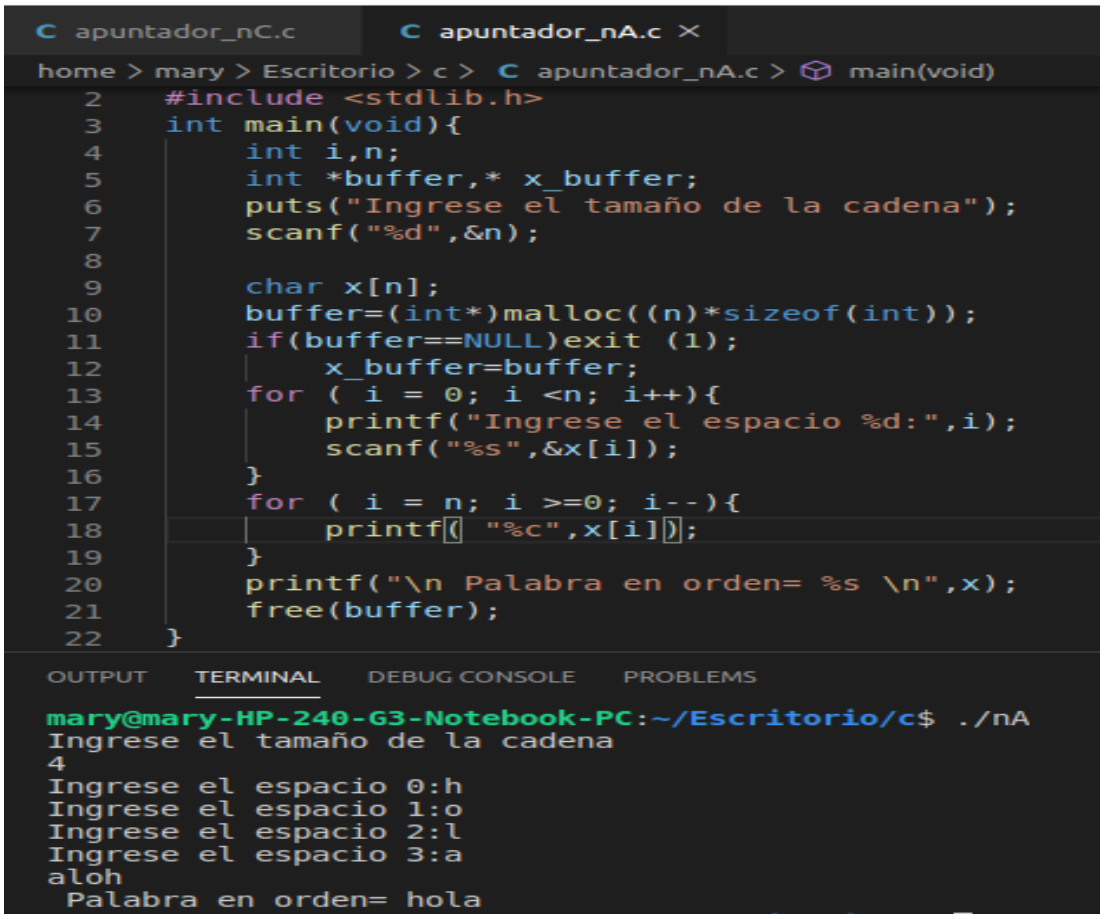


De todo lo visto anterior se crea un código donde hacemos uso de apuntadores, ademas de buffer free() permite liberar la memoria reservando a través de un apuntador ademas la función malloc() reserva memoria y retorna su dirección, el usuario ingresa la longitud del arreglo, se ocupan los valores anteriores, ingresa cada valor y esto se imprimen después.





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:

Los apuntadores funciona de una estructura totalmente diferente en el cual podemos seguir usando el código por partes para una mejor estructura ademas de que los datos se pueden pasar por parámetros, los apuntadores e declaran por un *p con el nombre del apuntador, cuentan con su propia dirección de memoria, antes esta el tipo de dato.
Se enlazan a los datos en específicos con su declaración, existe apuntadores de estructuras mas complejas como struct, ficheros, la dirección se hace por medio de &.
La función free() permute liberar espacio en memoria a través de un apuntador malloc() reserva memoria sin importar el tipo de datos que almacenará en ella.