|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **NOMBRE DE LA PRÁCTICA** | **Prácticas de C ++** | | | | **No.** | | **1** |
| **ASIGNATURA:** | **Métodos numéricos** | **CARRERA:** | **INGENIERIA EN SISTEMAS COMPUTACIONALES** | **PLAN:** | | **ISIC-2010-224** | |

**VERONICA CRUZ OSORIO**

**I. COMPETENCIA(S) ESPECÍFICA(S): Aprender c**

**II. MATERIAL EMPLEADO:**

**visual studio code**

**sistema operativo ubuntu 19.**

**COMPUTADORA**

**III.DESARROLLO DE LA PRÁCTICA:**

**Apuntadores**

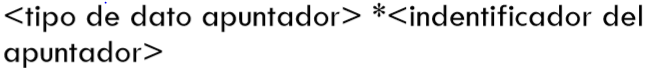
Un puntero es un objeto que apunta a otro objeto. Es decir una variable cuyo valores la dirección de memoria de otra variable.

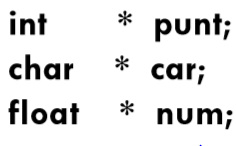
**¿Cómo se declaran los apuntadores?**

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:**





**¿Cómo se declaran los apuntadores?**

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

**Tipos de apuntadores**

Se puede también declarar apuntadores a estructuras más complejas.

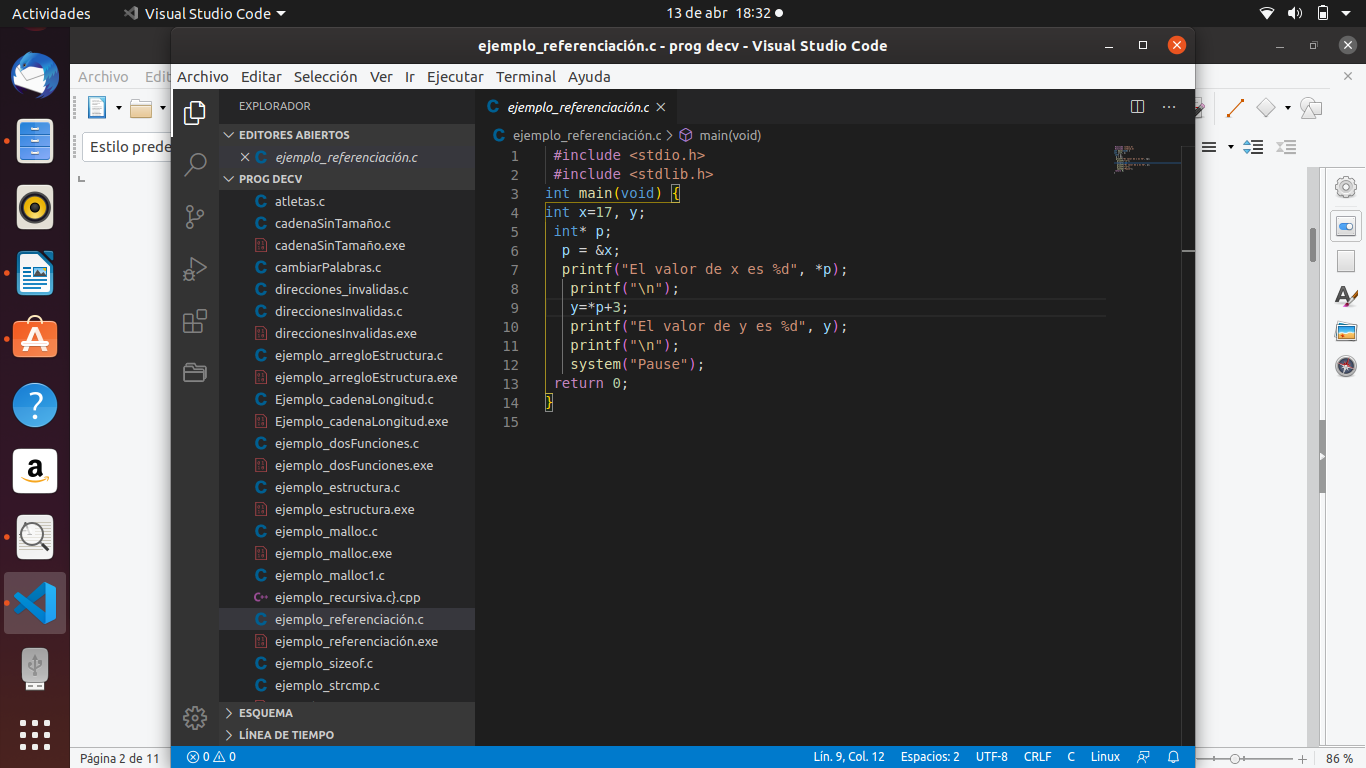
Funciones, Struct , Ficheros. Y también se pueden declarar punteros vacíos o nulos.

**¿Qué es la referenciación?**

La referenciación es obtener la dirección de una variable, a través del operador ‘&’, aplicado a la variable a la cual se desea saber su dirección.

Ejemplo

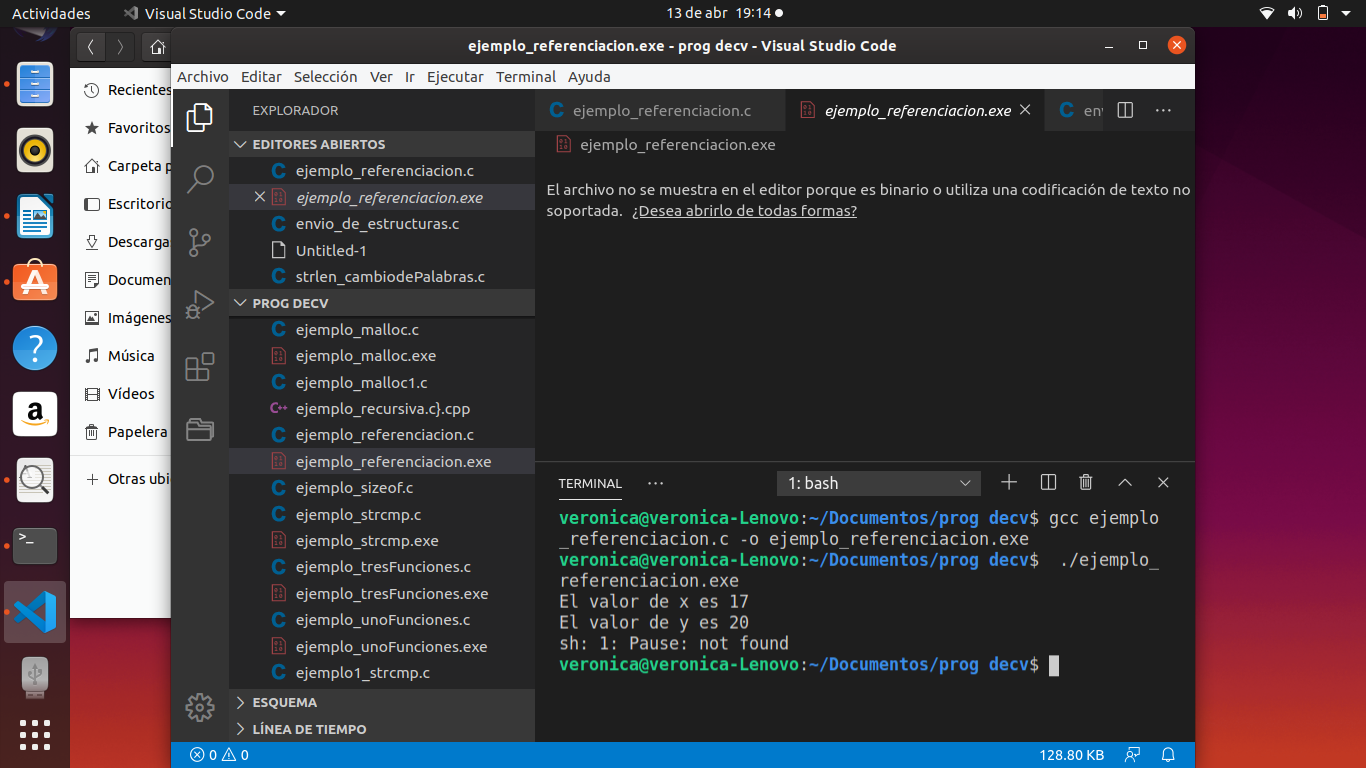
Bibliotecas



El método main, se declara 2 variables de tipo entero donde x vale 17 y no tiene valor

Se declra que **p**  va a hacer igual al valor que tiene x. Se manda un mensaje con ayuda de **printf,** + el texto + el valor de p=17.

Y para obtener el valor de y se realiza la operación de **\*p**+3=20

****

Se muestra en pantalla el resultado de printf

**¿Qué es la 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.



**Direcciones inválidas**

Un apuntador puede contener una dirección inválida por:

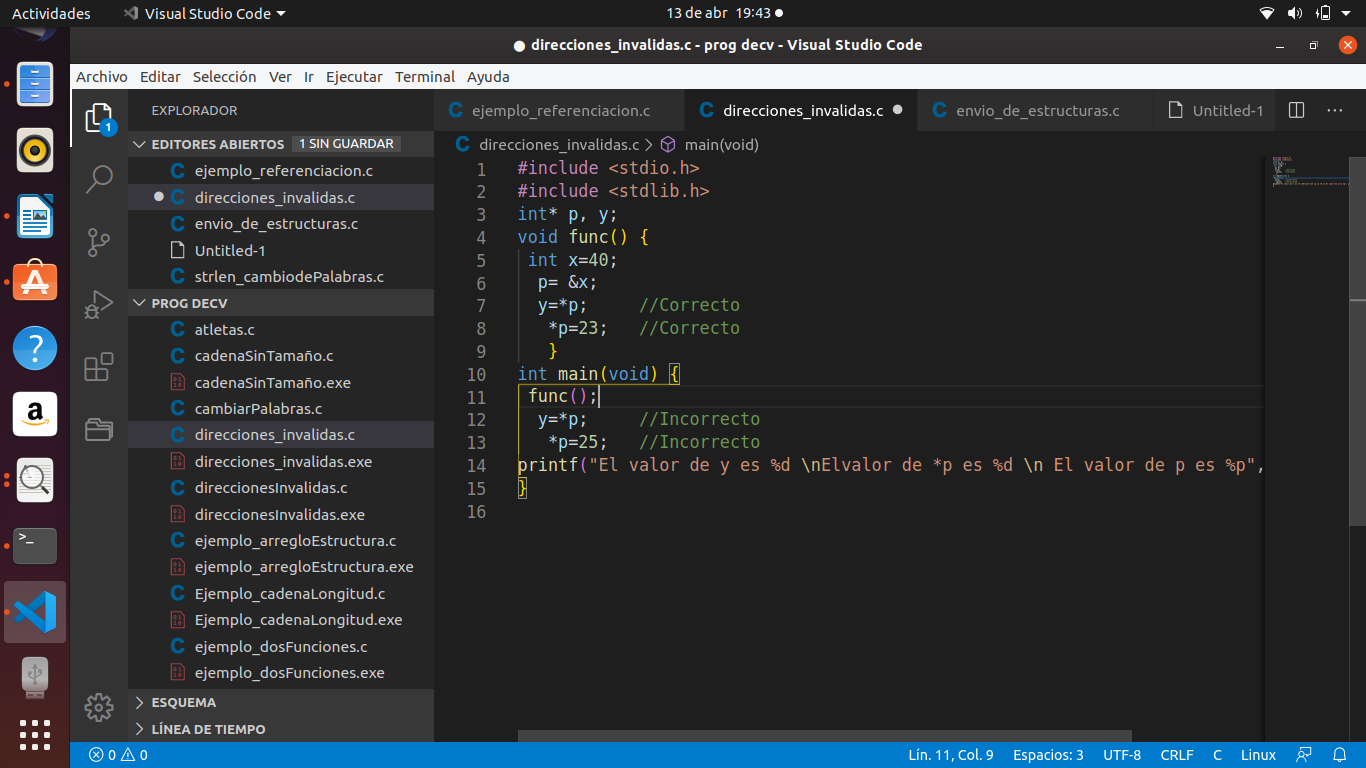
Cuando se declara un apuntador, posee un valor cualquiera que no se puede conocer con antelación.

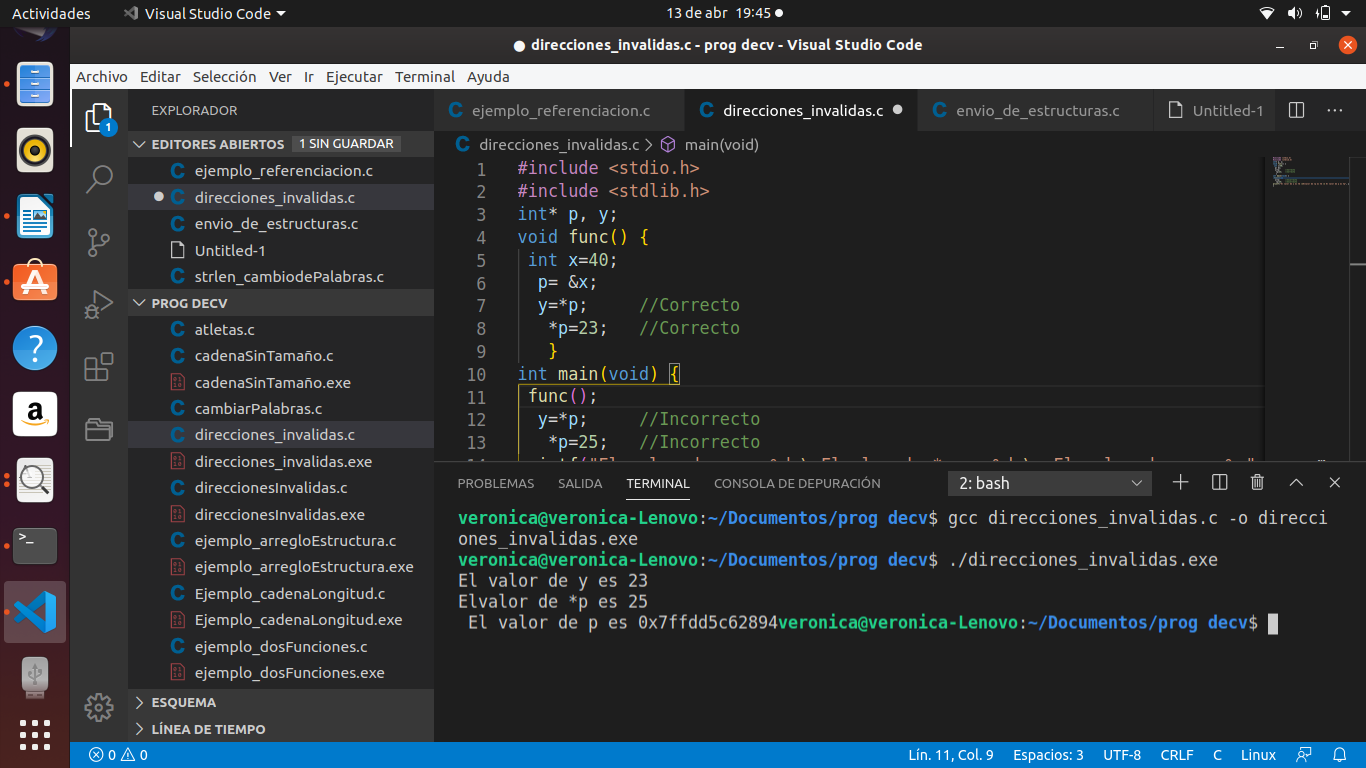
Después de que ha sido inicializado, la dirección que posee puede dejar de ser válida por que la variable asociada termina su ámbito o por que ese espacio de memoria fue reservado dinámicamente.

Se importan las bibliotecas.

Declara la variable **\*p**,**y**  sin valores y primero entramos al método principal que es main donde manda a llamar a la **void func** que se declara al iniciony donde se declarala variable **x=40**  de tipo entero. **P s**e se le asigna una dirección a través del operador de referencicion que es **&**  que son direcciones almacenada en otros apuntadores que en este caso es a la variable  **y** que va a tener el valor de **x=40,**  y después a **p=23 y recordemos que que p=&x donde se le asigna a x el 23. nos vamos al metodo main donde y=p**  en este caso el valor de 23 y \*==25  donde se sustituye el valor de x que era 23 a 25.

Ejemplo



****

Y lo que va a mostrar en pantalla son los valores q de **y=23**

**Y \*P=25,**  con la ayuda de un prontf que el que manda a imprimir + el texto que se va a mostrar y se concatenan laas variables que se van a imprimir

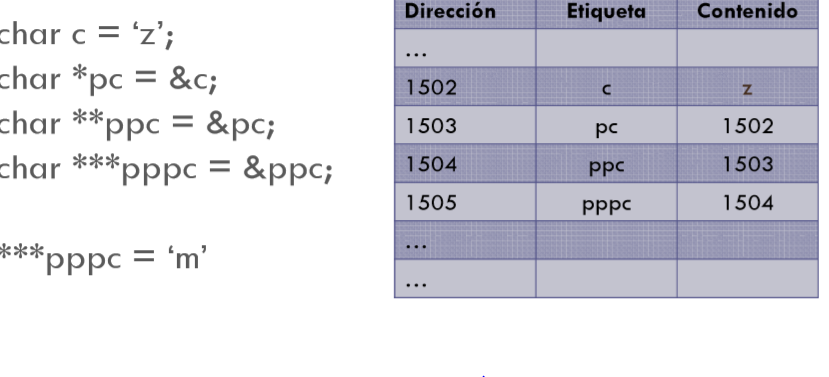
dado

**La dirección NULL**

* Cuando no se desea que el apuntador apunte a algo, se le suele asignar el valor de NULL, en cuyo caso se dice que el apuntador es nulo (no apunta a nada).
* NULL es una macro típicamente definida en archivos de cabecera como **stdef.hy stdlib.h.**
* Se utiliza para proporcionar a un programa un medio de conocer cuándo un apuntador contiene una dirección inválida.

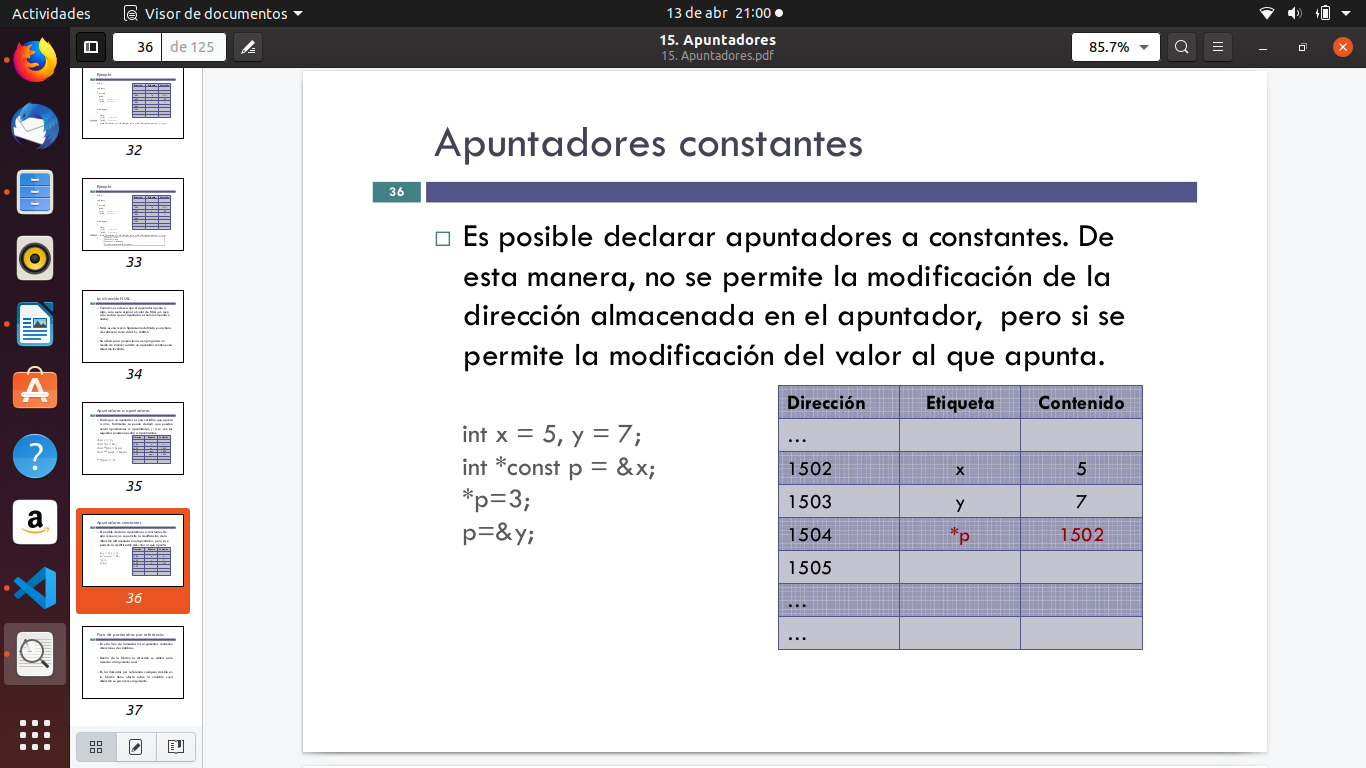
Dado que un apuntador es una variable que apunta a otra, fácilmente se puede deducir que pueden existir apuntador es a apuntadores, y asu vez los segundos pueden apuntar apuntadores.

**Apuntadores a apuntadores**



Donde se declara las variables de tipo char para poder ingresar caracteres y donde si no existe un puntero puede ocasionar un desbordamiento de memoria. Donde la variable c de tipo char tiene la lera **z** , en donde **\*pc=&c**  donde **&**  es un puntero se identifica con un operador de referenciacion y donde se el asigna la direccion de c=1502 y **\*\*ppc=&pc**  se le asigna la direcion de **pc=1530**  y asi en la siguiente variable **\*\*\*ppc=&ppc**  se le asigna la direccion 1504

**Apuntadores constantes**

****

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.

Donde se declara la variable x de tipo entero que tiene un valor de 5 y donde y=7 tambien de tipo entero, y se declara otra **\*const p=&x**  se le asigna la direccion 1502 a la variable **\*p** y donde **\*p=3** y **p** va a tener el valor de **y=7**

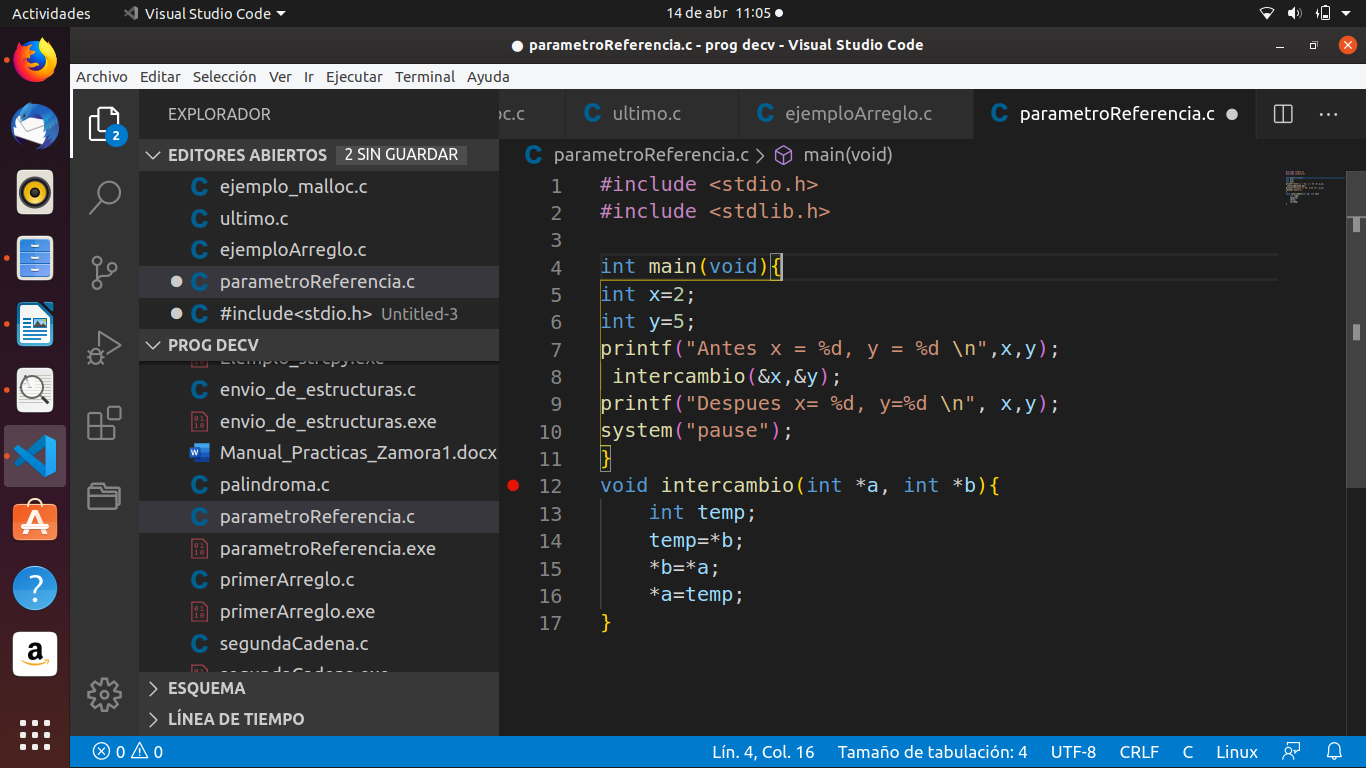
**Paso de parámetros por referencia**

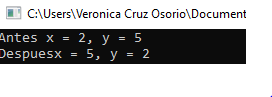
* En este tipo de llamadas los argumentos contienen direcciones de variables.
* Dentro de la función la dirección se utiliza para acceder al argumento real.
* En las llamadas por referencia cualquier cambio en la función tiene efecto sobre la variable cuya dirección se pasó como argumento.

**Paso de parámetros por referencia**

Se declaran las bibliotecas y en el metodo maui se declaran 2 variables de tipo entero que tiene valores **x=2, y=5**  y se manda a imprimir un mensaje donde muestra el valor de x & y antes que son los valores que se le asiganron y despues se cambian los valores de **x a y**

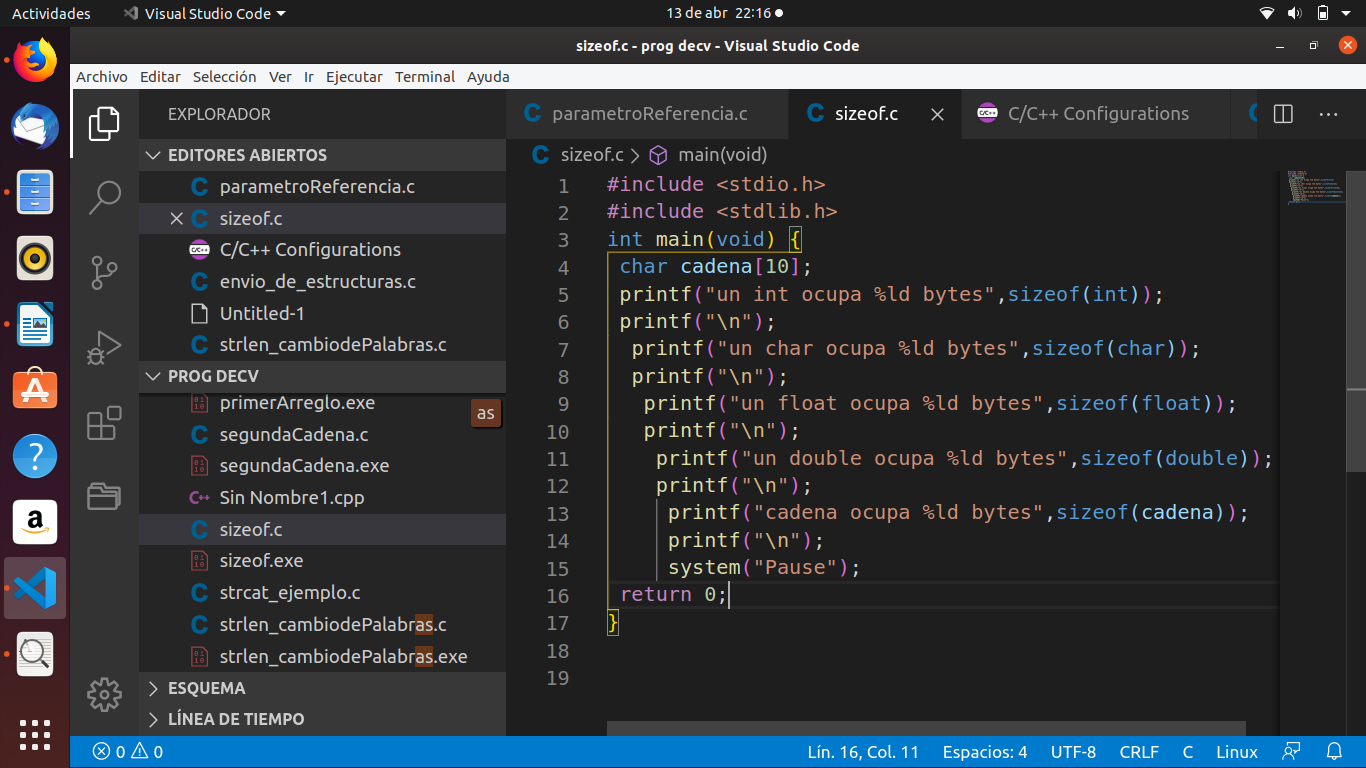
& **y a x**  pero antes de cambia los valores nos vamos al metodo intercambio donde tiene atributos de tipo entero que es \***a, \*b** estas variables son las wue va a almacenar los valores de \*b=5 y a=2





Se muestra los valores antes y despúes

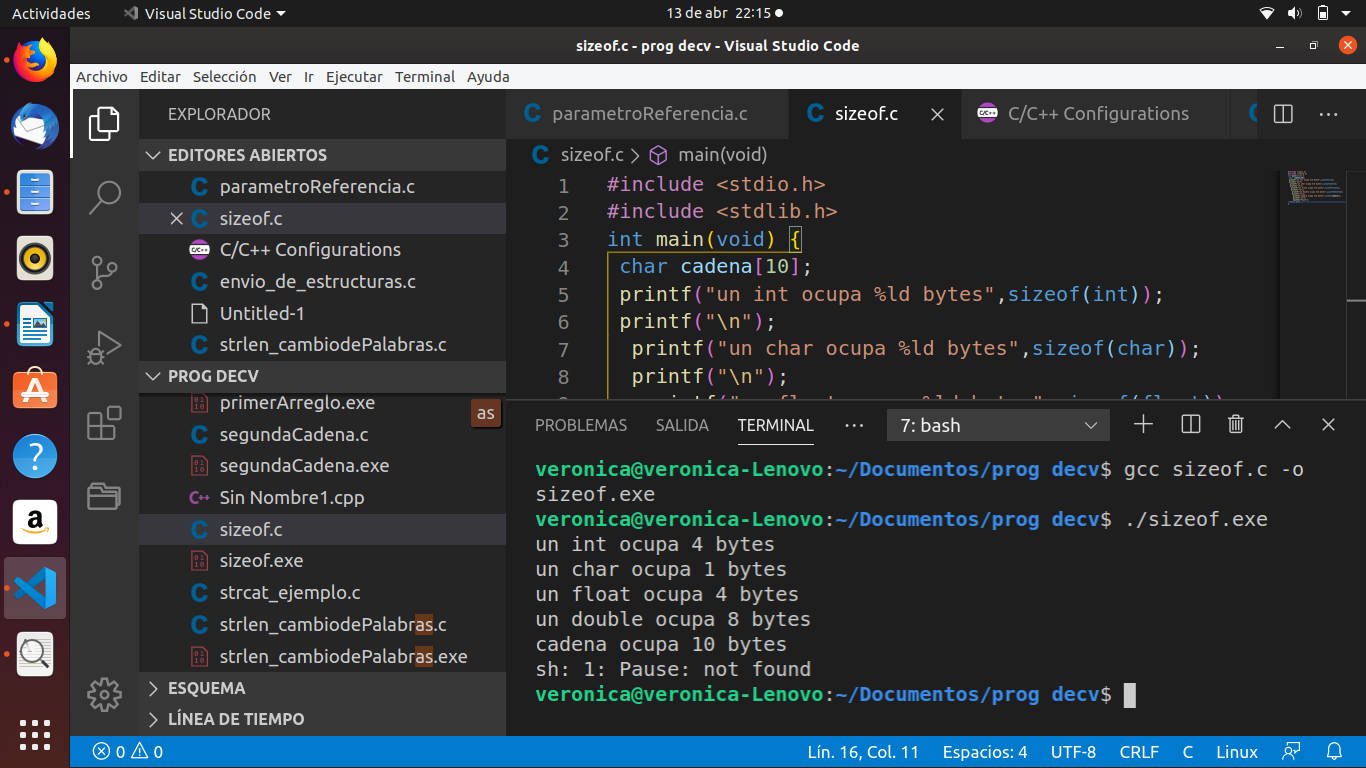
**La función sizeof()**



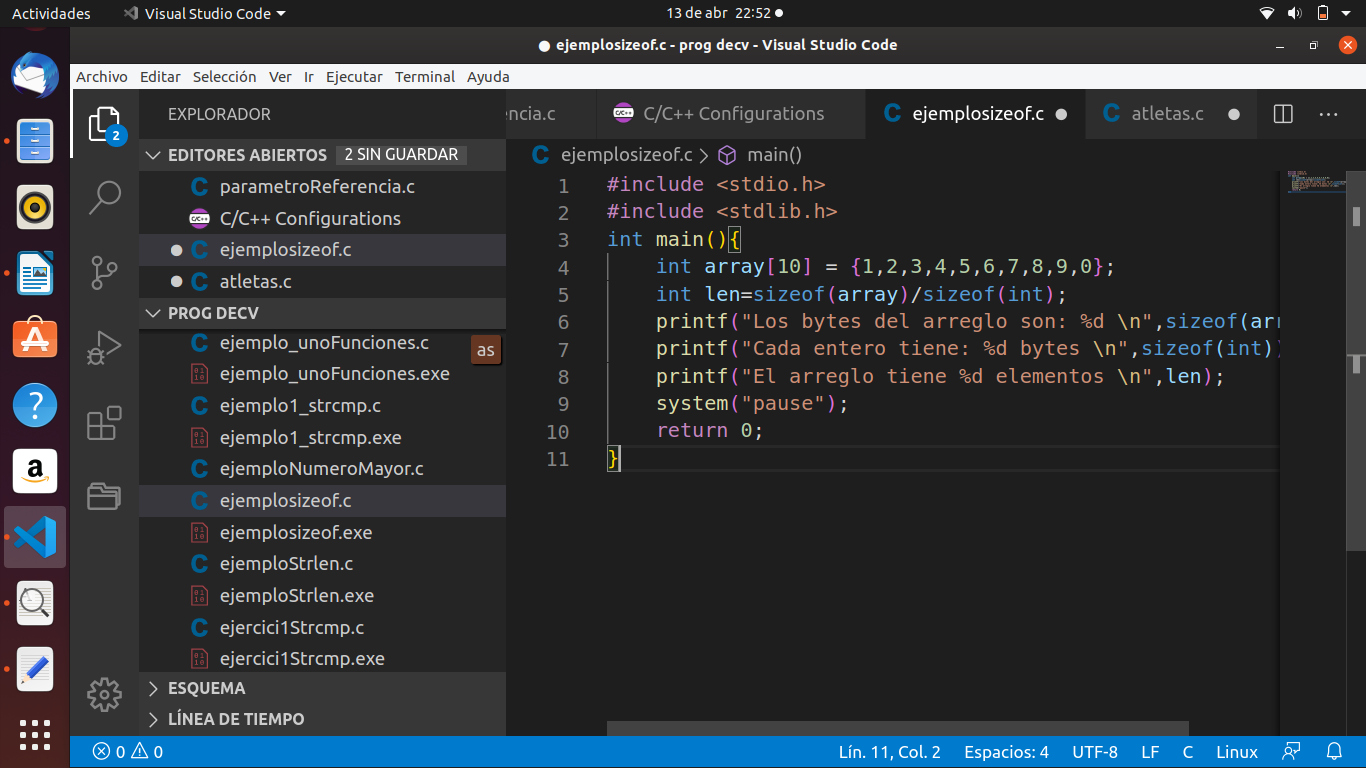
Sizeof que devuelve el tamaño en bytes que ocupa un tipo o variable en memoria.

Se declara las bibliotecas.

Se declara una cadena de tipo char que tiene un tamaño de 10, con e**l char** permite ingresar caracteres, y donde manada a imprimir la cantidad de bytes que ocupa un int, chart, float, double y nos debe de mostrar el tamaño de cadena.

****

Muestra el pantalla el tamaño de bayte que ocupa cada int, chart, float, double y la cadena ocupa 10 bytes

****

**Asignación dinámica de memoria**

* Los programas pueden crear variables globales 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.
* Las variables locales se almacenan en la pila(stack)y existen solo mientras están activas las funciones donde están declaradas.
* En ambos casos el espacio de almacenamientos reservan el momento de compilación del programa.
* Para asignar memoria dinámicamente se utilizan las funciones **malloc() y free()**, definidas típicamente en el archivo **stdlib.h**.

**free()**

La función free() permite liberar la memoria reservada a través de un apuntador.

**voidfree (void\* ptr);**ptres un puntero de cualquier tipo que apunta a un área de memoria reservada previamente conmalloc.

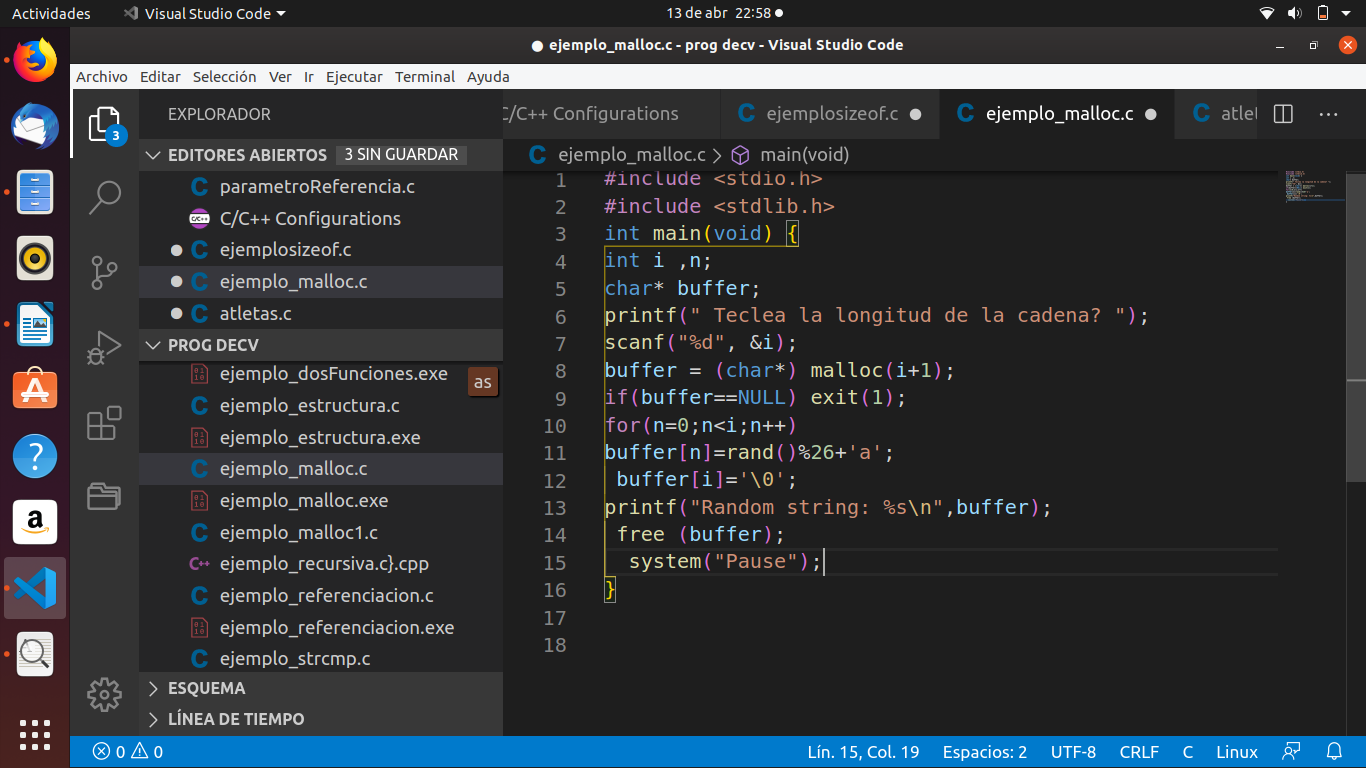
**malloc()**

La función **malloc()** reserva memoria y retorna su dirección, o retorna NULL en caso de no haber conseguido suficiente memoria.

**Void\*malloc(size\_ttam\_bloque)**

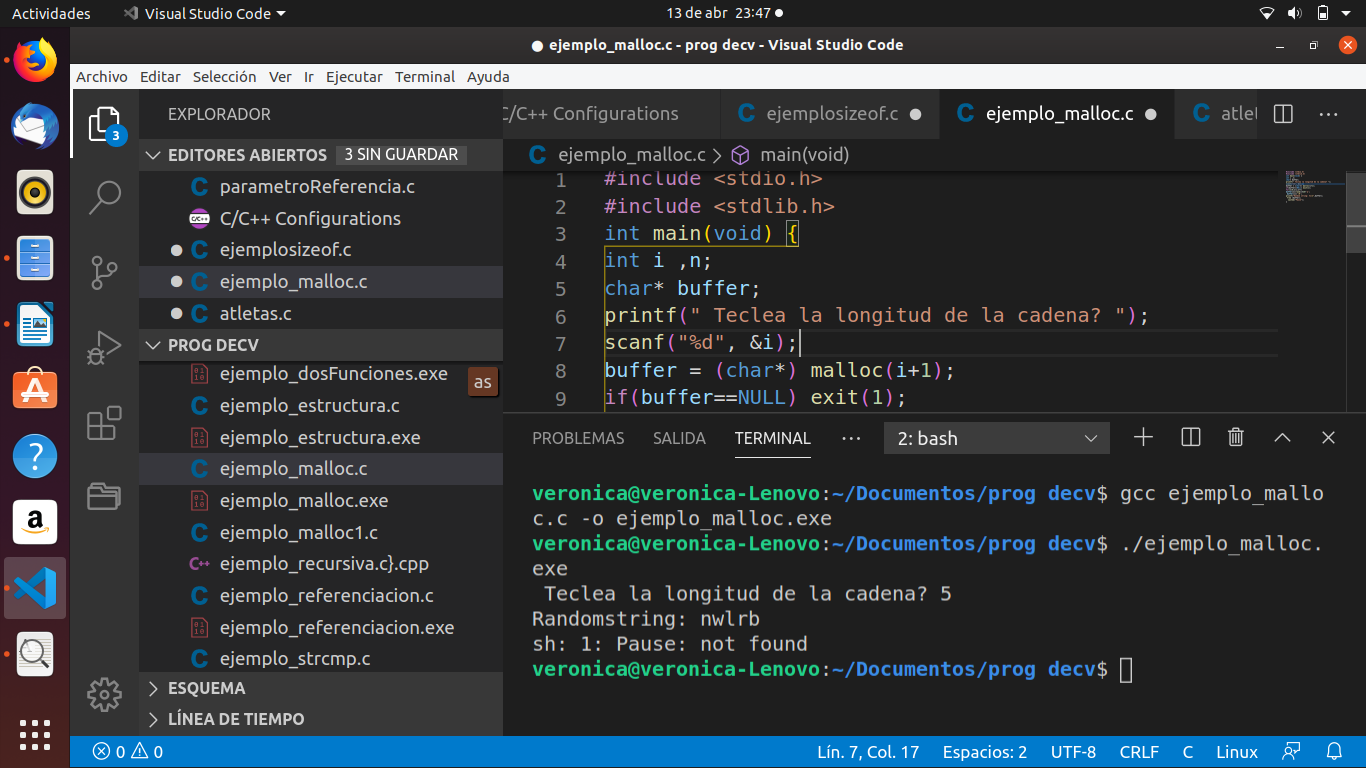
**malloc()** reserva memoria sin importar el tipo de datos que almacenará en ella.

**Ejemplo**



Declaración de bibliotecas.

Se declaran 2 variables de tipo entero **i,n**  y otra de tipo char de nombre **\*buffer** se manda un mensaje en pantalla donde el usuario debe de ingresar el tamaño de la cadena. El valor que el usuario ingresa se va a almacenar en la variable **i**  entonces **buffer =\*char**  que reserva en memoria y retorna su direccion = **null**  donde se crean 4 espacion en memoria que inicia desde 0. donde en el buffer 0 se asigna la letra **f**  y donde n =1**,**  buffer 1 se asigna la letra **m**  y donde n ahora vale =2, buffer 2 se le asigna **a**  y donde n =3, buffer 3 y se le asigna **b**  y donde n=4 y buffer 4 se les asigna la letra **p**  y donde n ya no tiene otro tamaño y es /**0.**  donde llegar buffer [i] que i es el valor de 5 y llega hasta el numero 4 recordando que inica desde 0 y se manda a aimprimir  **Random string**

****

Se muestra en pantalla donde se ingresa el numero 5 el tamaño de la cadena y despues se muestra **Random string**

**Ejemplo**

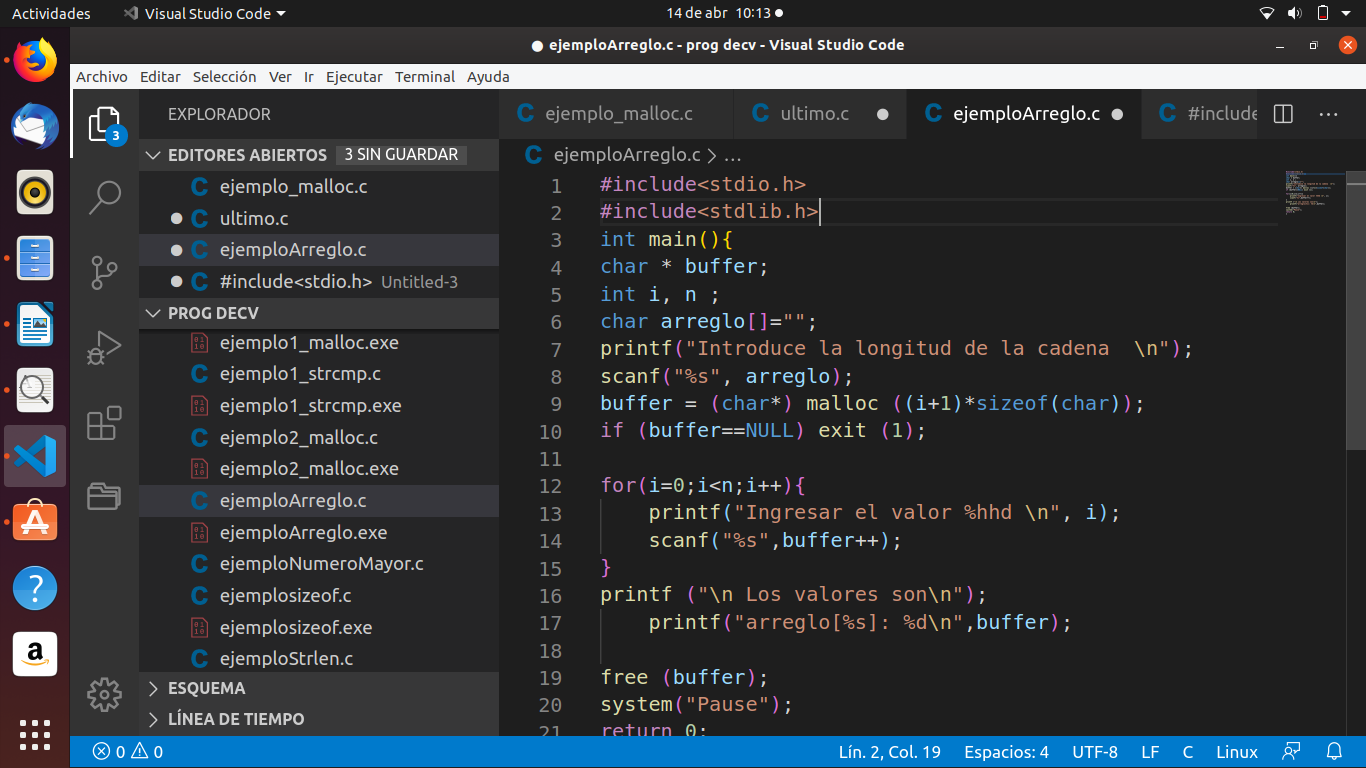
Crea un arreglo entero de tamaño x, en donde x es ingresado por teclado.

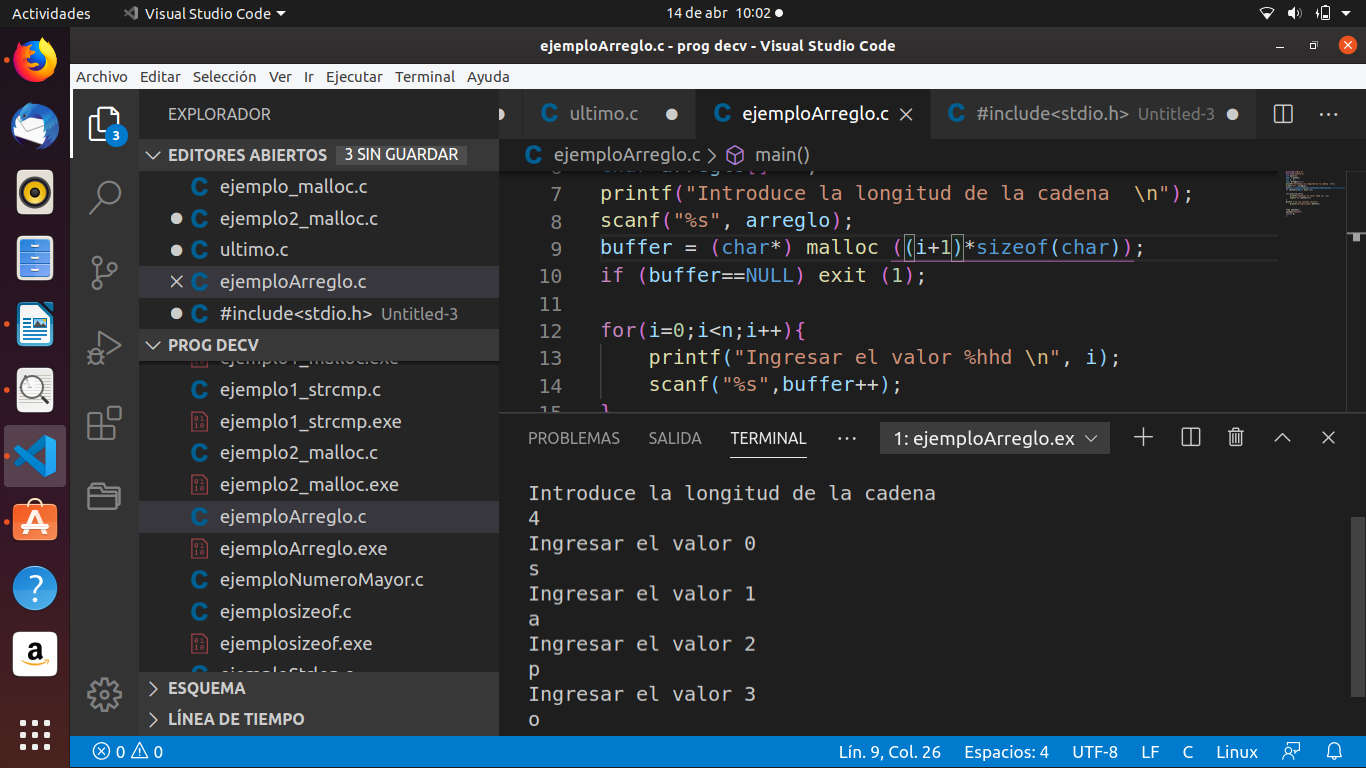
Llena todos los elementos del arreglo con datos ingresados por el usuario.

Se declaran las bibliotecas.

Se crea el metodo principal main y se declaran 2 variables de tipo entero **i,n**  y una de tipo **char \*buffer**, y se declara un arreglo de tipo char para que pueda ingresar caracteres, pero no tiene tamaño por eso es necesario que el usuario lo ingrese y para eso se muestra un mensaje en pantalla donde le pide al usuario que ingrese la longitud de la cadena y el valor que fue ingresado se almacena en la cadena **arreglo** y despues nos vamos con el cilo for donde se repetira el cilo las veces que el usuario ingreso el valor y se va mostrar en pantalla **ingresa el valor**  y va a iniciar de 0 hasta 4

Muestra los valores



****

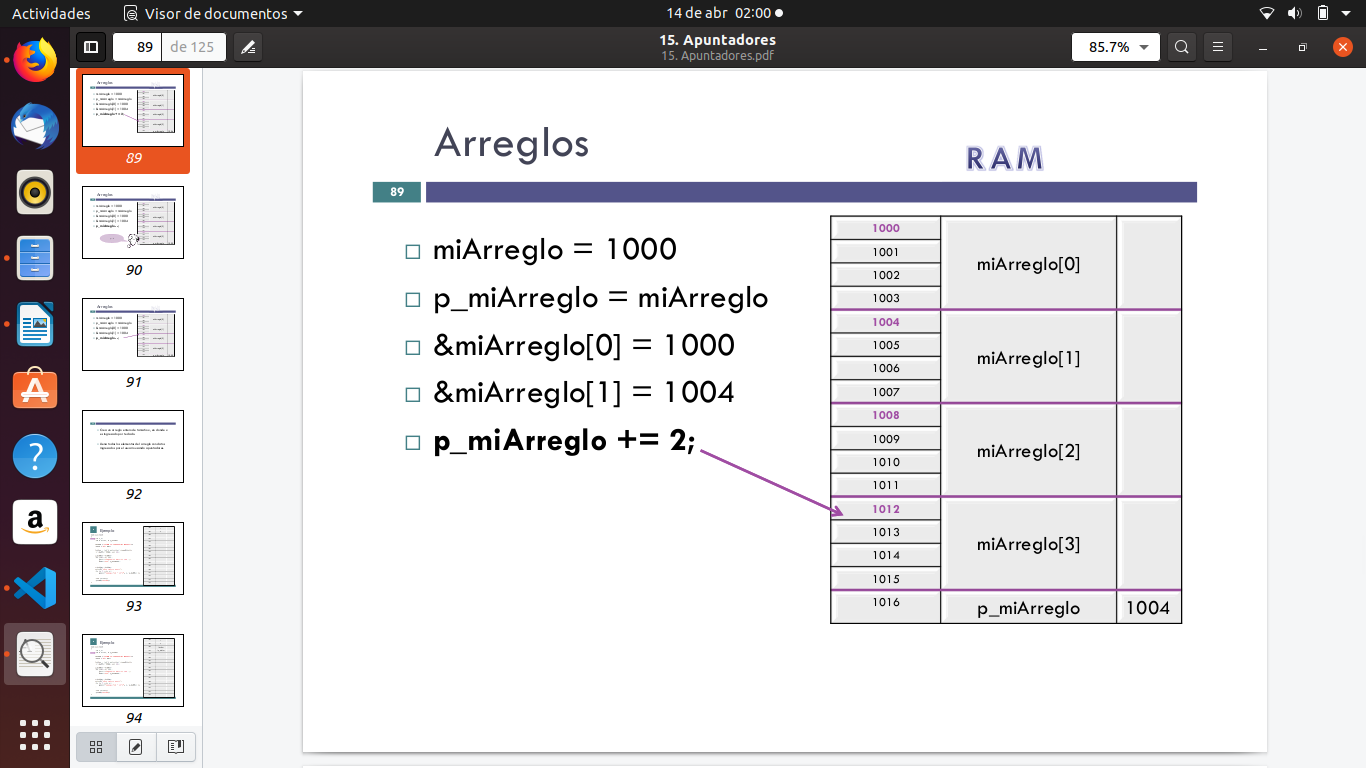
Se muestra los valores de la cadena y se muestra los mensajes para que el usuario ingrese los valores.

**Apuntadores a arreglos**

El nombre de un arreglo es simplemente un apuntador constante al inicio del arreglo.

Se declara un arreglo de nombre **miArreglo,** y exixte un aumento de 4 en 4, pero p\_miArreglo = miArreglo que son 4 y al llenar [0] hasta [4] va a tener el valor de **1004, 1008, 1012**  con ayuda de p\_miArreglo++

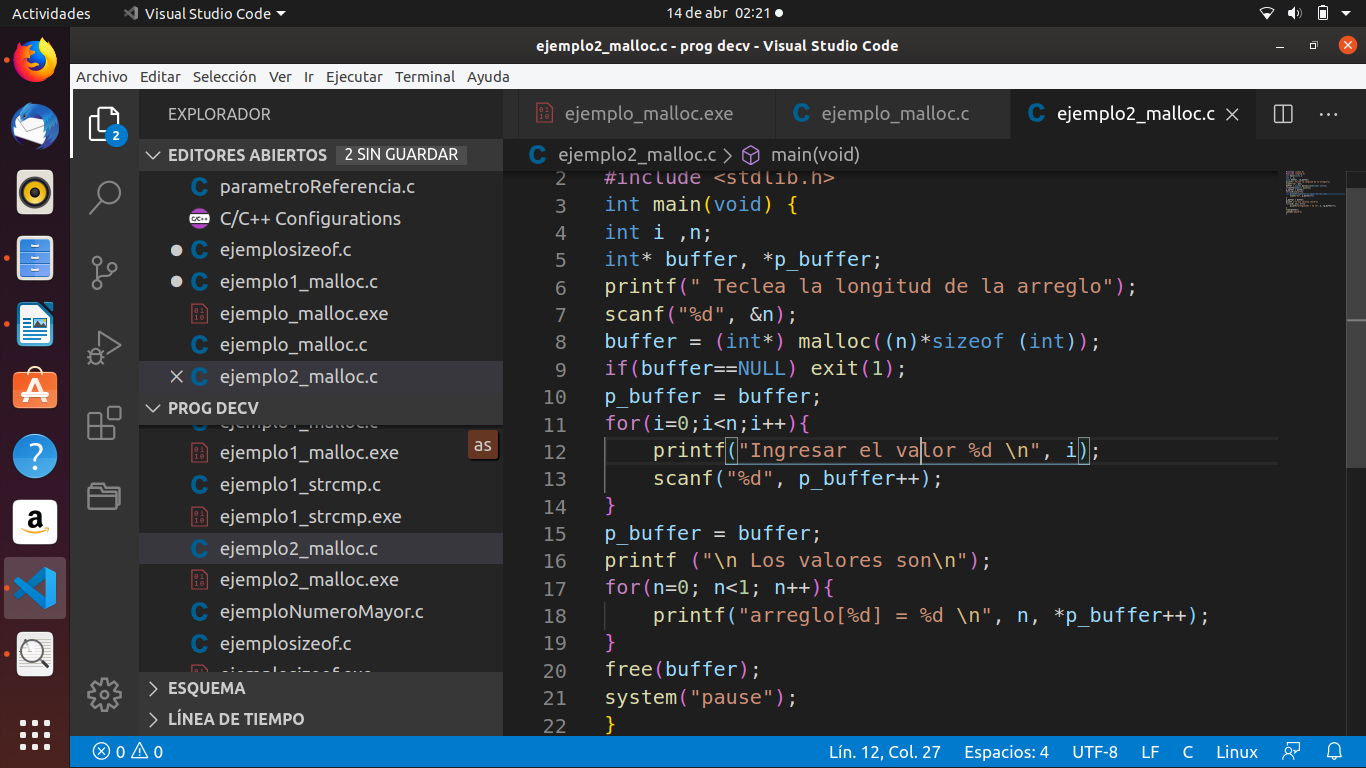
**Arreglos**

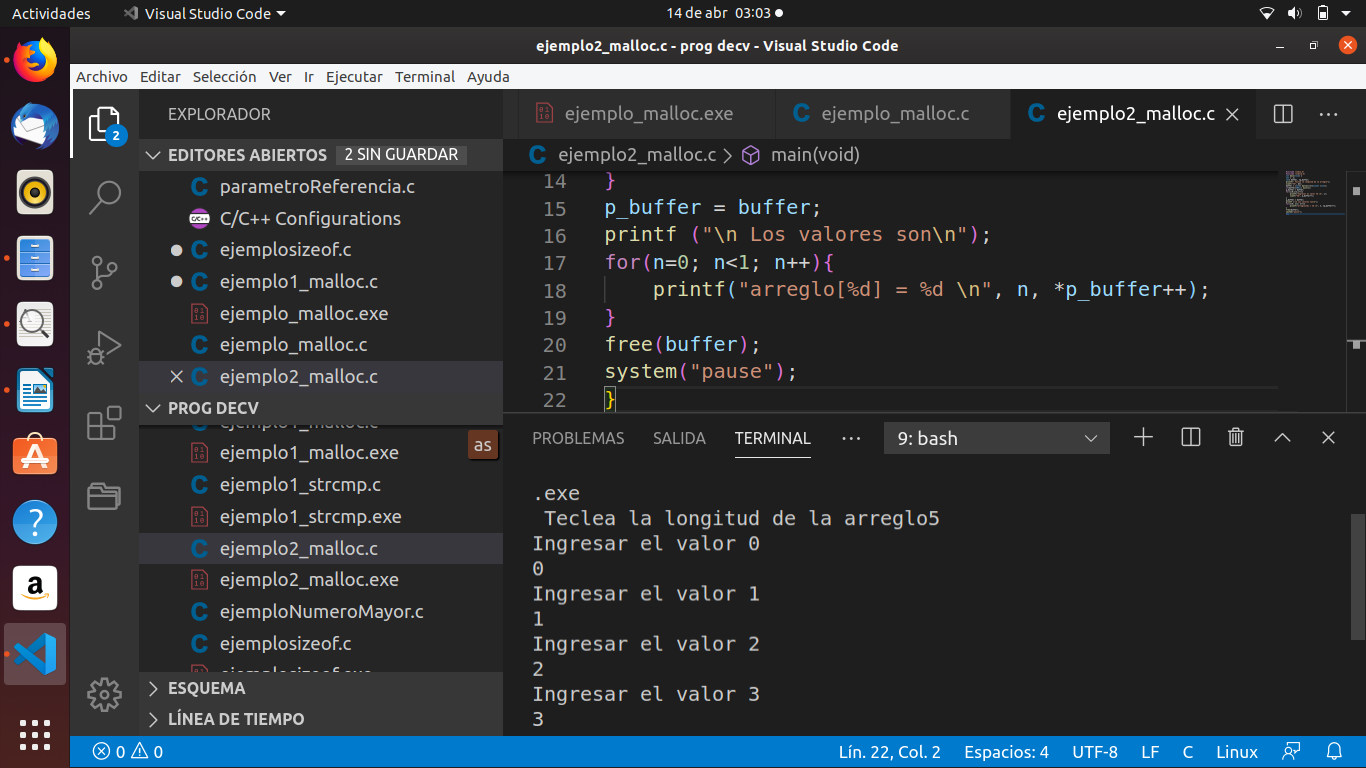
****

Se declara la biblioteca.

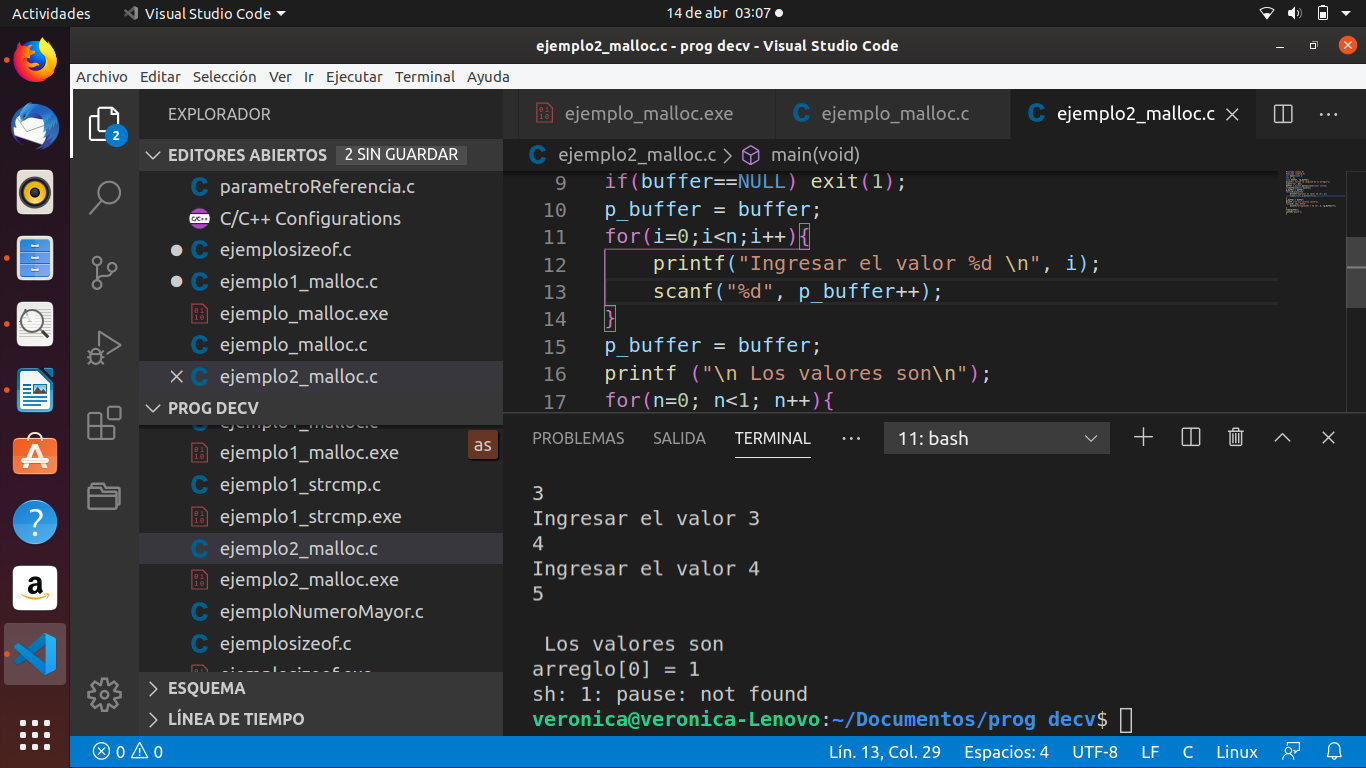
Se declara 2 variables de tipo entero **i,n n,** y **\*buffer** y **\*p\_buffer,** y se manda un mensaje en pantalla para que el usuario ingrese lq longitud del arreglo, y se al amacena en  la variable **n**  , buffer almacena 1016 que es el limite sabiendo que sizeof(), donde n=3 y la siguiente variable es **\*p\_buffer**  y tiene el valor mismo 1016 y donde i= y donde buffer[0] equivale a 5 que es lo que ingreso el usuario y que **\*p\_buffer**  cambia el valor a 1020 y i=1 y buffer[1] vale a 6 y cambia **\*p\_buffer**  1024 y i cambia a 2 y cambia **\*p\_buffer**  a 1028 y buffer[2] equivale a 7 y i=2, i=3 y **\*p\_buffer?1016 y n=0.**

**Direcciones del arreglo**

****

****

Se manda a imprimir en pantalla el mensaje los valores son 1,2,3,4, los que se han alamacenados en cadena **\*p\_buffer=1020 y n=1 y \*p\_buffer**  va cambiando 2 en 2, 1020-1024 y **n=2, 1028-n=3 y**  se manda a imprimir el arreglo.

****

Incremento de operadores

Cuando se incrementa un apuntador se está incrementando su valor.

Si incrementamos en 1 el valor del apuntador, C sabe el tipo de dato al que apunta e incrementa la dirección guardada en el apuntador en el tamaño del tipo de dato.

**I**

**Ejercicios**

Crea un arreglo de tipo charde 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

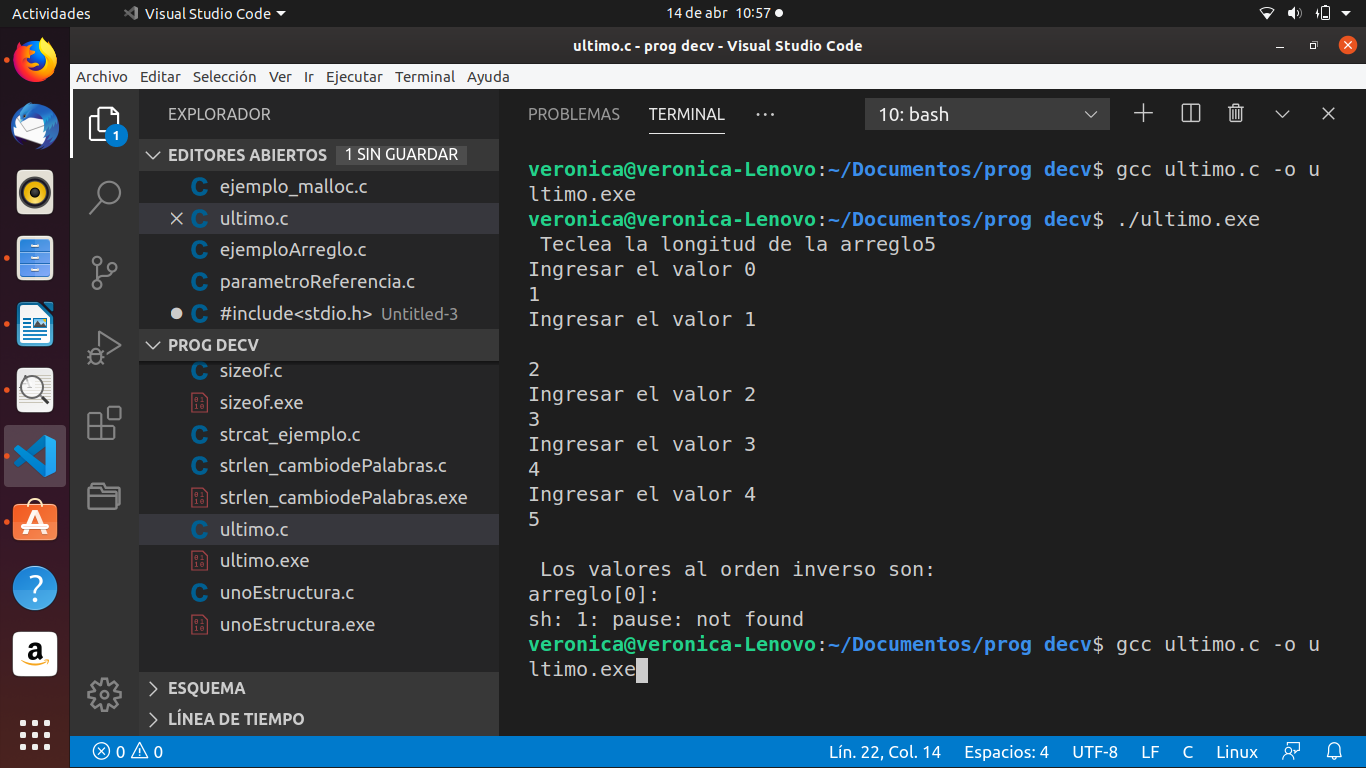


Se declara las bibliotecas.

En el metodo main se declara 2 variables de tipo char  **\*buffer, \*p\_buffer** y 2 de tipo entero **i,n**  se muestra un mensaje en pantalla donde le pide al usuario que ingrese la longitud del arregloy el valor que es ingresado se almacena en la variable n se utilizasizeof que devuelve el tamaño en bytes que ocupa una variable en memoria y malloc para asignar en memoria dinamica

Se crea un ciclo for donde i=0 y la condicion es que i < al valor de n y exixte un aunmento a **i y**  se muestra un mensaje en pantalla donde se debe de ingresar los valores de la cadena hasta que llque sea menor o igual a valor de n y estos valores son alamacenado en ña variable \*p\_buffer y se va a mostrar en oder inverso entonces es necesario que se ocupe un ciclo for donde el valor de **n**  es igual **i-1**  por que va a ir de abajo hacia arriba y el valor de n menor o igual a **0**  y un decremento en los valores en  **n**

Se muestra en pantalla el mensaje de la longitud del arreglo y es 5 y te muestra las 5 veces el llegando del errego y despues te muestra los valores



**conclusiones**

El entender como funciona los apuntadores se me complico, aun no esta dominado del tema pero con la practica y estudiando y revisando de nuevo las laminas podre comprender mas, por ejemplo, nos permiten pasar argumentos (o parámetros) a una función y modificarlos. También permiten el manejo de cadenas y de arrays. ... Un puntero no sólo sirve para apuntar a una variable, también sirve para apuntar una dirección de memoria determinada. Un Apuntador es una variable que contiene una dirección de memoria, la cual corresponderá a un dato o a una variable que contiene el dato. Cada variable que se utiliza en una aplicación ocupa una o varias posiciones de memoria. Estas posiciones de memoria se accesan por medio de una dirección.