



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

# Laboratorios de computación salas A y B

---

*Profesor: Alejandro Esteban Pimentel Alarcon*

*Asignatura: Fundamentos de programación*

*Grupo: 3*

*No de Práctica(s): 6*

*Integrante(s): Rivera Sosa Arlethe*

*No. de Equipo de cómputo empleado:*

*No. de Lista o Brigada: 317083033*

*Semestre: 2020-1*

*Fecha de entrega: 30/septiembre/19*

*Observaciones:*      Muy bien

**CALIFICACIÓN: 10**

## Entorno en C

El lenguaje C es un lenguaje estructurado, en el mismo sentido que lo son otros lenguajes de programación tales como el lenguaje Pascal, el Ada o el Modula-2, pero no es estructurado por bloques, o sea, no es posible declarar subrutinas (pequeños trozos de programa) dentro de otras subrutinas, a diferencia de como sucede con otros lenguajes estructurados tales como el Pascal. Además, el lenguaje C no es rígido en la comprobación de tipos de datos, permitiendo fácilmente la conversión entre diferentes tipos de datos y la asignación entre tipos de datos diferentes, por ejemplo, la expresión siguiente es válida en C:

```
float a; /*Declaro una variable para números reales*/
```

```
int b; /*Declaro otra variable para número enteros*/
```

```
b=a; /*Asigno a la variable para entera el número real*/
```

Todo programa de C consta, básicamente, de un conjunto de funciones, y una función llamada main, la cual es la primera que se ejecuta al comenzar el programa, llamándose desde ella al resto de funciones que compongan nuestro programa.

Desde su creación, surgieron distintas versiones de C, que incluían unas u otras características, palabras reservadas, etc. Este hecho provoco la necesidad de unificar el lenguaje C, y es por ello que surgió un standard de C, llamado ANSI-C, que declara una serie de características, etc., que debe cumplir todo lenguaje C. Por ello, y dado que todo programa que se desarrolle siguiendo el standard ANSI de C será fácilmente portable de un modelo de ordenador a otro modelo de ordenador, y de igual forma de un modelo de compilador a otro, en estos apuntes explicaremos un C basado en el standard ANSI-C.

Objetivo: Conocer y usar los ambientes y herramientas para el desarrollo y ejecución de programas en Lenguaje C, como editores y compiladores en diversos sistemas operativos.

Actividad 1:

- ★ Txt: Documentos de texto plano guardados en formato TXT se pueden crear, abrir y editar utilizando una amplia variedad de programas de procesamiento de texto y de edición de textos desarrollados para sistemas Linux, ordenadores y plataformas Mac Microsoft basado en Windows. El contenido de estos .txt. constituye el formato básico, pues solamente presenta texto, sin ningún tipo de “enriquecimiento” visual más como tipos o tamaños de letra, ni permite mezclar imágenes con el texto, ni siquiera funcionalidades tan simples para los demás procesadores y formatos de texto como el justificar a la derecha. Sencillamente son archivos de tamaño reducido.
  
- ★ Markdown: es una forma sencilla de agregar formato a textos web y crear páginas web sin tener experiencia en HTML. Markdown nos permite aprovechar las ventajas del texto plano, sin tener que renunciar al formato, pero sí a todas las complicaciones derivadas del mismo que pueden surgir cuando lo trasladamos a la web. Se pueden hacer listas de tareas que se pueden pasar y que son más sencillas de hacer que teniendo que usar las viñetas de Word. Por otro lado, también podemos usarlo con una flexibilidad inusitada. No tenemos que preocuparnos por compatibilidad de formatos porque este tipo de textos se pueden leer sin importar el programa que usemos para abrirlos. Por otro lado, se pueden guardar en diferentes formatos para hacer diferentes cosas.
  
- ★ Html: es un lenguaje de marcado que se utiliza para el desarrollo de páginas de Internet. Se trata de la sigla que corresponde al Lenguaje de Marcas de Hipertexto, que podría ser traducido como Lenguaje de Formato de Documentos para Hipertexto. Se encarga de desarrollar una descripción sobre los contenidos que aparecen como textos y sobre su estructura, complementando dicho texto con diversos objetos como fotografías, animaciones, etc.

Es un lenguaje muy simple y general que sirve para definir otros lenguajes que tienen que ver con el formato de los documentos. El texto en él se crea

a partir de etiquetas, también llamadas tags, que permiten interconectar diversos conceptos y formatos.

Para la escritura de este lenguaje, se crean etiquetas que aparecen especificadas a través de corchetes o paréntesis angulares. Entre sus componentes, los elementos dan forma a la estructura esencial del lenguaje, ya que tienen dos propiedades el contenido en sí mismo y sus atributos.

★ LaTeX: es un sistema de composición de textos que está orientado especialmente a la creación de documentos científicos que contengan fórmulas matemáticas, cuadros y tablas. Además, también se pueden crear otros tipos de documentos, que pueden ser desde cartas sencillas hasta libros completos. LATEX está organizado sobre TEX. LATEX está disponible para la mayoría de los miniordenadores y microordenadores, desde IBM PC en adelante. LATEX es un paquete de macros que le permite al autor de un texto componer e imprimir su documento con la mayor calidad tipográfica, empleando para ello patrones previamente definidos, es un sistema de composición de textos que está formado mayoritariamente por órdenes (macros) construidas a partir de comandos de Tex un lenguaje «de bajo nivel

★ Csv: es un archivo de texto que almacena los datos en forma de columnas, separadas por coma y las filas se distinguen por saltos de línea. El formato CSV es muy sencillo y no indica un juego de caracteres concreto, ni cómo van situados los bytes, ni el formato para el salto de línea. Estos puntos deben indicarse muchas veces al abrir el archivo, por ejemplo, con una hoja de cálculo.

Además, el término "CSV" también denota otros formatos de valores separados por delimitadores que usan delimitadores diferentes a la coma como los valores separados por tabuladores. Un delimitador que no está presente en los valores de los campos, mantiene el formato simple. Estos archivos separados por delimitadores alternativos reciben en algunas ocasiones la extensión, aunque este uso sea incorrecto.

## Actividad 2:

MINGW64/c/Users/Familia/documents

```
=====
  B i e n v e n i d o   a l   t u t o r   d e   V I M   -   V e r s i ó n   1.4   =
=====

Vim es un editor muy potente que dispone de muchos mandatos, demasiados
para ser explicados en un tutor como éste. Este tutor está diseñado
para describir suficientes mandatos para que usted sea capaz de
aprender fácilmente a usar Vim como un editor de propósito general.

El tiempo necesario para completar el tutor es aproximadamente de 25-30
minutos, dependiendo de cuanto tiempo se dedique a la experimentación.

Los mandatos de estas lecciones modificarán el texto. Haga una copia de
este fichero para practicar (con «vimtutor» esto ya es una copia).

Es importante recordar que este tutor está pensado para enseñar con
la práctica. Esto significa que es necesario ejecutar los mandatos
para aprenderlos adecuadamente. Si únicamente se lee el texto, se
olvidarán los mandatos.

Ahora, asegúrese de que la tecla de bloqueo de mayúsculas no está
activada y pulse la tecla j lo suficiente para mover el cursor
de forma que la Lección 1.1 ocupe completamente la pantalla.

Lección 1.1: MOVIMIENTOS DEL CURSOR

** Para mover el cursor, pulse las teclas h,j,k,l de la forma que se indica. **
vim-tutor.txt[+] [unix] (21:25 29/09/2019)
```

```
Familia@DESKTOP-NE4QAFJ MINGW64 ~ (master)
$ cd documents

Familia@DESKTOP-NE4QAFJ MINGW64 ~/documents (master)
$ vim vim-tutor.txt

Familia@DESKTOP-NE4QAFJ MINGW64 ~/documents (master)
$ |
```

```
=====
  B i e n v e n i d o   a l   t u t o r   d e   V I M   -   V e r s i ó n   1.4   =
=====

Vim es un editor muy potente que dispone de muchos mandatos, demasiados
para ser explicados en un tutor como éste. Este tutor está diseñado
para describir suficientes mandatos para que usted sea capaz de
aprender fácilmente a usar Vim como un editor de propósito general.

El tiempo necesario para completar el tutor es aproximadamente de 25-30
minutos, dependiendo de cuanto tiempo se dedique a la experimentación.

Los mandatos de estas lecciones modificarán el texto. Haga una copia de
este fichero para practicar (con «vimtutor» esto ya es una copia).

Es importante recordar que este tutor está pensado para enseñar con
la práctica. Esto significa que es necesario ejecutar los mandatos
para aprenderlos adecuadamente. Si únicamente se lee el texto, se
olvidarán los mandatos.
```

## Edición de texto -borrado

```
Lección 1.3: EDICIÓN DE TEXTO - BORRADO

** Estando en modo Normal pulse x para borrar el carácter sobre el cursor. **j

1. Mueva el cursor a la línea de abajo señalada con --->.
2. Para corregir los errores, mueva el cursor hasta que esté bajo el
   carácter que va a ser borrado.
3. Pulse la tecla x para borrar el carácter sobrante.
4. Repita los pasos 2 a 4 hasta que la frase sea la correcta.
---> La vaca saltó sobre la luna.

5. Ahora que la línea esta correcta, continúe con la Lección 1.4.
```

## Edición de texto -inserción

```
Lección 1.4: EDICION DE TEXTO - INSERCIÓN

** Estando en modo Normal pulse i para insertar texto. **

1. Mueva el cursor a la primera línea de abajo señalada con --->.
2. Para que la primera línea se igual a la segunda mueva el cursor bajo el
   primer carácter que sigue al texto que ha de ser insertado.
3. Pulse i y escriba los caracteres a añadir.
4. A medida que sea corregido cada error pulse <ESC> para volver al modo
   Normal. Repita los pasos 2 a 4 para corregir la frase.
---> Falta algo de texto en esta línea.
---> Falta algo de texto en esta línea.

5. Cuando se sienta cómodo insertando texto pase al resumen que esta más
   abajo.
```

## Mandatos para borrar

```
Lección 2.1: MANDATOS PARA BORRAR

** Escriba dw para borrar hasta el final de una palabra **

1. Pulse <ESC> para asegurarse de que está en el modo Normal.
2. Mueva el cursor a la línea de abajo señalada con --->.
3. Mueva el cursor al comienzo de una palabra que desee borrar.
4. Pulse dw para hacer que la palabra desaparezca.

NOTA: Las letras dw aparecerán en la última línea de la pantalla cuando
      las escriba. Si escribe algo equivocado pulse <ESC> y comience de nuevo.

---> Hay algunas palabras que no pertenecen a esta frase.
```

## Más mandatos para borrar

```
Lección 2.2: MÁS MANDATOS PARA BORRAR

** Escriba d$ para borrar hasta el final de la línea. **

1. Pulse <ESC> para asegurarse de que está en el modo Normal.
2. Mueva el cursor a la línea de abajo señalada con --->.
3. Mueva el cursor al final de la línea correcta (DESPUES del primer . ).
4. Escriba d$ para borrar hasta el final de la línea.
--> Alguien ha escrito el final de esta línea dos veces.
```

## Una excepción al mandato-objeto

```
Lección 2.4: UNA EXCEPCION AL 'MANDATO-OBJETO'

** Escriba dd para borrar una línea entera. **

Debido a la frecuencia con que se borran líneas enteras, los diseñadores
de Vim decidieron que sería más fácil el escribir simplemente dos des en
una fila para borrar una línea.

1. Mueva el cursor a la segunda línea de la lista de abajo.
2. Escriba dd para borrar la línea.
3. Muévase ahora a la cuarta línea.
4. Escriba 2dd (recuerde número-mandato-objeto) para borrar las dos
   líneas.

1) Las rosas son rojas,
3) El cielo es azul,
6) El azúcar es dulce,
7) Y así eres tu.
```

## El mandato deshacer

```
Lección 2.5: EL MANDATO DESHACER

** Pulse u para deshacer los últimos mandatos,
   U para deshacer una línea entera. **

1. Mueva el cursor a la línea de abajo señalada con ---> y sitúelo bajo el
   primer error.
2. Pulse x para borrar el primer carácter erróneo.
3. Pulse ahora u para deshacer el último mandato ejecutado.
4. Ahora corrija todos los errores de la línea usando el mandato x.
5. Pulse ahora U mayúscula para devolver la línea a su estado original.
6. Pulse ahora u unas pocas veces para deshacer lo hecho por U y los
   mandatos previos.
7. Ahora pulse CTRL-R (mantenga pulsada la tecla CTRL y pulse R) unas
   pocas veces para volver a ejecutar los mandatos (deshacer lo deshecho).
--> Corrija los errores de esta línea y vuelva a ponerlos coon deshacer.

8. Estos mandatos son muy útiles. Ahora pase al resumen de la Lección 2.
```

---

cambio; antes #24 57 seconds ago

## El mandato PUT

```
Lección 3.1: EL MANDATO «PUT» (poner)

** Pulse p para poner lo último que ha borrado después del cursor. **

1. Mueva el cursor al final de la lista de abajo.
2. Escriba dd para borrar la línea y almacenarla en el buffer de vim.
3. Mueva el cursor a la línea que debe quedar por debajo de la
   línea a mover.
4. Estando en mod Normal, pulse p para restituir la línea borrada.
5. Repita los pasos 2 a 4 para poner todas las líneas en el orden correcto.

   a) Las rosas son rojas,
   b) Las violetas son azules,
   c) La inteligencia se aprende,
   d) ¿Puedes aprenderla tu?
```

## El mandato replace

```
Lección 3.2: EL MANDATO «REPLACE» (reemplazar)

** Pulse r y un carácter para sustituir el carácter sobre el cursor. **

1. Mueva el cursor a la primera línea de abajo señalada con --->.
2. Mueva el cursor para situarlo bajo el primer error.
3. Pulse r y el carácter que debe sustituir al erróneo.
4. Repita los pasos 2 y 3 hasta que la primera línea esté corregida.
--> ¡Cuando esta línea fue escrita alguien pulsó algunas teclas equivocadas!
--> ¡Cuando esta línea fue escrita alguien pulsó algunas teclas equivocadas!
```

## Mandato Change

```
Lección 3.3: EL MANDATO «CHANGE» (cambiar)

** Para cambiar parte de una palabra o toda ella escriba cw . **

1. Mueva el cursor a la primera línea de abajo señalada con --->.
2. Sitúe el cursor en la u de lubrs.
3. Escriba cw y corrija la palabra (en este caso, escriba 'ínea').
4. Pulse <ESC> y mueva el cursor al error siguiente (el primer carácter
   que deba cambiarse).
5. Repita los pasos 3 y 4 hasta que la primera frase sea igual a la segunda.
--> Esta línea tiene unas pocas palabras que corregir usando el mandato change.
--> Esta línea tiene unas pocas palabras que corregir usando el mandato change.
```



## Más cambios usando C

```
Lección 3.4: MÁS CAMBIOS USANDO c

** El mandato change se utiliza con los mismos objetos que delete. **

1. El mandato change funciona de la misma forma que delete. El formato es:

    [número] c objeto      0      c [número] objeto

2. Los objetos son también los mismos, tales como w (palabra), $ (fin de
   la línea), etc.

3. Mueva el cursor a la primera línea de abajo señalada con --->.

4. Mueva el cursor al primer error.

5. Escriba c$ para hacer que el resto de la línea sea como la segunda
   y pulse <ESC>.

---> El final de esta línea necesita ser corregido usando el mandato c$.
---> El final de esta línea necesita ser corregido usando el mandato c$.
```

## El mandato search

```
** Escriba / seguido de una frase para buscar la frase. **

1. En modo Normal pulse el carácter /. Fíjese que tanto el carácter /
   como el cursor aparecen en la última línea de la pantalla, lo mismo
   que el mandato :.

2. Escriba ahora errroor <INTRO>. Esta es la palabra que quiere buscar.

3. Para repetir la búsqueda, simplemente pulse n.
   Para buscar la misma frase en la dirección opuesta, pulse Mayu-N.

4. Si quiere buscar una frase en la dirección opuesta (hacia arriba),
   utilice el mandato ? en lugar de /.

---> Cuando la búsqueda alcanza el final del fichero continuará desde el
   principio.

«errroor» no es la forma de deletrear error; errroor es un error.

-----
Lección 4.3: BÚSQUEDA PARA COMPROBAR PARENTESIS

** Pulse % para encontrar el paréntesis correspondiente a ),] o }. **
vim-tutor.txt[+] [unix] (22:06 29/09/2019)
/, errroor
```

## Búsqueda para comprobar paréntesis

```
Lección 4.3: BÚSQUEDA PARA COMPROBAR PARENTESIS

** Pulse % para encontrar el paréntesis correspondiente a ),] o }. **

1. Sitúe el cursor en cualquiera de los caracteres ),] o } en la línea de
   abajo señalada con --->.

2. Pulse ahora el carácter % .

3. El cursor debería situarse en el paréntesis (, corchete [ o llave {
   correspondiente.

4. Pulse % para mover de nuevo el cursor al paréntesis, corchete o llave
   correspondiente.

---> Esto es una línea de prueba con (, [, ], {, y } en ella. ).

Nota: ¡Esto es muy útil en la detección de errores en un programa con
   paréntesis, corchetes o llaves dispares.
```

## Una forma de cambiar errores

```
Lección 4.4: UNA FORMA DE CAMBIAR ERRORES

** Escriba :s/viejo/nuevo/g para sustituir 'viejo' por 'nuevo'. **

1. Mueva el cursor a la línea de abajo señalada con --->.

2. Escriba :s/laas/las/ <INTRO> . Tenga en cuenta que este mandato cambia
sólo la primera aparición en la línea de la expresión a cambiar.

--> Las mejores épocas para ver las flores son las primaveras.

4. Para cambiar todas las apariciones de una expresión ente dos líneas
escriba :#,#s/viejo/nuevo/g donde #,# son los números de las dos
líneas. Escriba :%s/viejo/nuevo/g para hacer los cambios en todo
el fichero.
```

## Como ejecutar un mandato externo

```
Lección 5.1: COMO EJECUTAR UN MANDATO EXTERNO

** Escriba :! seguido de un mandato externo para ejecutar ese mandato. **

1. Escriba el conocido mandato : para situar el cursor al final de la
pantalla. Esto le permitirá introducir un mandato.

2. Ahora escriba el carácter ! (signo de admiración). Esto le permitirá
ejecutar cualquier mandato del sistema.

3. Como ejemplo escriba ls después del ! y luego pulse <INTRO>. Esto
le mostrará una lista de su directorio, igual que si estuviera en el

~$INGLES.docx
12 HIGHLIGHTS OF MARK ZUCKERBERG.docx
análisis la lista de schindler civica.docx
Armas nucleares presentacion.pptx
Armas Nucleares...docx
Armas Nucleares..docx
Armas Nucleares.docx
armas nucleares.pdf
Azulejos musulmanes y su geometría.docx
Blocs de notas de OneNote
vim-tutor.txt[+] [unix] (22:06 29/09/2019)
```

## Más sobre guardar ficheros

```
Lección 5.2: MAS SOBRE GUARDAR FICHEROS

** Para guardar los cambios hechos en un fichero,
escriba :w NOMBRE_DE_FICHERO. **

1. Escriba :!dir o :!ls para ver una lista de su directorio.
Ya sabe que debe pulsar <INTRO> después de ello.

2. Elija un nombre de fichero que todavía no exista, como TEST.

3. Ahora escriba :w TEST (donde TEST es el nombre de fichero elegido).

4. Esta acción guarda todo el fichero (Vim Tutor) bajo el nombre TEST.
Para comprobarlo escriba :!dir de nuevo y vea su directorio.

--> Tenga en cuenta que si sale de Vim y entra de nuevo con el nombre de
fichero TEST, el fichero sería una copia exacta del tutor cuando lo
ha guardado.
```

```
tertunias\ poemas.docx
TEST
textos\ caro.docx
triptico\ azulejos\ musulmanes.docx
triptico\ del siberico\ 13 de mayo.docx
```

## El mandato OPEN

```
Lección 6.1: EL MANDATO «OPEN» (abrir)

** Pulse o para abrir una línea debajo del cursor
y situarle en modo Insert **

1. Mueva el cursor a la línea de abajo señalada con --->.

2. Pulse o (minúscula) para abrir una línea por DEBAJO del cursor
y situarle en modo Insert.

3. Ahora copie la línea señalada con ---> y pulse <ESC> para salir del
modo Insert.

--> Luego de pulsar o el cursor se sitúa en la línea abierta en modo Insert.
Luego de pulsar o el cursor se sitúa en la línea abierta en modo Insert.
```

## El mandato APPEND

```
Lección 6.2: EL MANDATO «APPEND» (añadir)

** Pulse a para insertar texto DESPUES del cursor. **

1. Mueva el cursor al final de la primera línea de abajo señalada con --->
pulsando $ en modo Normal.

2. Escriba una a (minúscula) para añadir texto DESPUES del carácter
que está sobre el cursor. (A mayúscula añade texto al final de la línea).

Nota: ¡Esto evita el pulsar i , el último carácter, el texto a insertar,
<ESC>, cursor a la derecha y, finalmente, x , sólo para añadir algo
al final de una línea!

3. Complete ahora la primera línea. Nótese que append es exactamente lo
mismo que modo Insert, excepto por el lugar donde se inserta el texto.

--> Esta línea le permitirá practicar el añadido de texto al final de una línea.
--> Esta línea le permitirá practicar el añadido de texto al final de una línea.
```

## Otra versión de REPLACE

```
Lección 6.3: OTRA VERSIÓN DE «REPLACE» (reemplazar)

** Pulse una R mayúscula para sustituir más de un carácter. **

1. Mueva el cursor a la primera línea de abajo señalada con --->.
2. Sitúe el cursor al comienzo de la primera palabra que sea diferente
   de las de la segunda línea marcada con ---> (la palabra 'anterior').
3. Ahora pulse R y sustituya el resto del texto de la primera línea
   escribiendo sobre el viejo texto para que la primera línea sea igual
   que la primera.
-> Para hacer que esta línea sea igual que la siguiente escriba R y el texto.
-> Para hacer que esta línea sea igual que la siguiente escriba R y el texto.
4. Nótese que cuando pulse <ESC> para salir, el texto no alterado permanece.
```

## Fijar opciones

```
Lección 6.4: FIJAR OPCIONES

** Fijar una opción de forma que una búsqueda o sustitución ignore la caja **
(Para el concepto de caja de una letra, véase la nota al final del fichero)
1. Busque 'ignorar' introduciendo:
  /ignorar
  Repita varias veces la búsqueda pulsando la tecla n
2. Fije la opción 'ic' (Egnorar la caja de la letra) escribiendo:
  :set ic
3. Ahora busque 'ignorar' de nuevo pulsando n
  Repita la búsqueda varias veces más pulsando la tecla n
4. Fije las opciones 'hlsearch' y 'insearch':
  :set hls is
5. Ahora introduzca la orden de búsqueda otra vez, y vea qué pasa:
  /ignore

RESUMEN DE LA LECCIÓN 6

1. Pulsando o abre una línea por DEBAJO del cursor y sitúa el cursor en
   la línea abierta en modo Insert.
vim-tutor.txt[+] [unix] (22:06 29/09/2019)
/ignor
```

## Mandatos para ayuda en línea

```
help.txt      For Vim version 8.1. Last change: 2019 Jul 21

      VIM - main help file

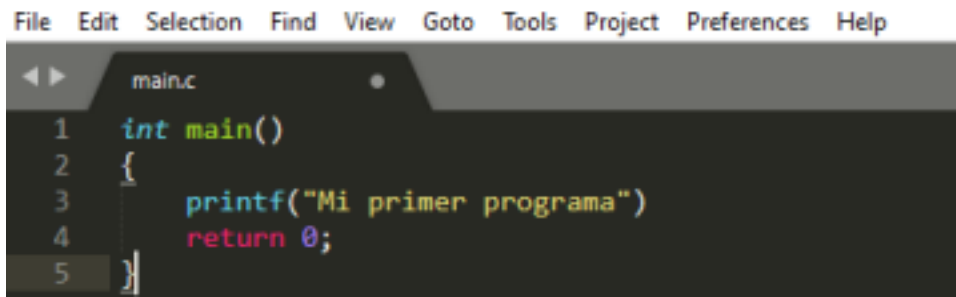
Move around: Use the cursor keys, or "h" to go left,          h  l
              "j" to go down, "k" to go up, "l" to go right.    j
Close this window: Use ":q<Enter>".
Get out of Vim: Use ":q!<Enter>" (careful, all changes are lost!).

Jump to a subject: Position the cursor on a tag (e.g. bars) and hit CTRL-J.
With the mouse:   ":set mouse=a" to enable the mouse (in xterm or GUI).
                  Double-click the left mouse button on a tag, e.g. bars.
Jump back: Type CTRL-O. Repeat to go further back.

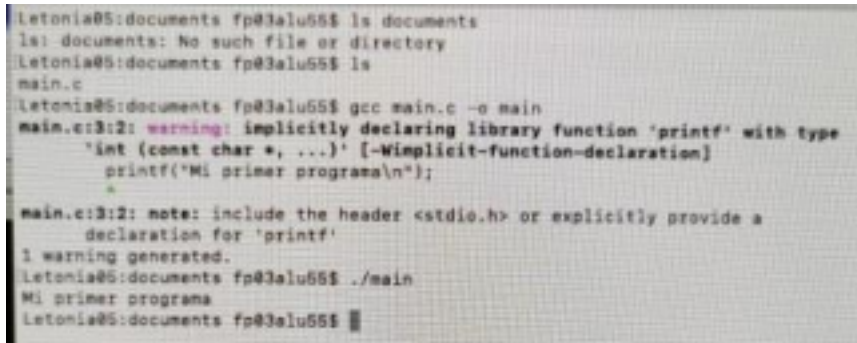
Get specific help: It is possible to go directly to whatever you want help
on, by giving an argument to the :help command.
Prepend something to specify the context: help-context

      WHAT      PREPEND  EXAMPLE
Normal mode command      :help x
Visual mode command      v_  :help v_u
Insert mode command      i_  :help i_<ESC>
Command-line command     :   :help :quit
Command-line editing     c_  :help c_<Del>
```

### Actividad3:



```
File Edit Selection Find View Goto Tools Project Preferences Help
main.c
1  int main()
2  {
3      printf("Mi primer programa")
4      return 0;
5  }
```



```
Letonia@5:documents fp@3alu55$ ls documents
ls: documents: No such file or directory
Letonia@5:documents fp@3alu55$ ls
main.c
Letonia@5:documents fp@3alu55$ gcc main.c -o main
main.c:3:2: warning: implicitly declaring library function 'printf' with type
      'int (const char *, ...)' [-Wimplicit-function-declaration]
      printf("Mi primer programa\n");
      ^
main.c:3:2: note: include the header <stdio.h> or explicitly provide a
      declaration for 'printf'
1 warning generated.
Letonia@5:documents fp@3alu55$ ./main
Mi primer programa
Letonia@5:documents fp@3alu55$
```

Conclusión: Vim nos ayuda a no depender de un mouse, haciendo el programa más práctico. Los comandos que utiliza son fáciles de poner en práctica y generan funciones muy útiles. Mientras que Sublime text nos ayuda a llevar acabo una función, compilar y correrla en un programa compilador.

### Bibliografía:

- <https://www.definicionabc.com/tecnologia/formatos-texto.php>
- <https://hipertextual.com/archivo/2013/04/que-es-markdown/>
- <https://definicion.de/html/>
- <https://www.ecured.cu/LaTeX>