



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): 06

Integrante(s): Gómez Mendoza Jonan

*No. de Equipo de
cómputo empleado:* 34 Perú

No. de Lista o Brigada: 18

Semestre: 2020-1

Fecha de entrega: 30/09/19

Observaciones: Muy bien, sin embargo la entrega se hizo tarde.

CALIFICACIÓN: 8

Introducción

En la actualidad la necesidad de programar todo tipo de códigos es muy alta por lo tanto se han creado diversos editores, tipos de texto, lenguajes, compiladores para programar estos a través de los años a adquirido una alta profesionalidad.

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

Investigación de tipos de archivos

Txt:

Es un archivo de texto sencillo sin caracteres especiales justificado ni nada además de texto

Markdown:

Tipo de texto que transforma lo que este adentro en texto XHTML para paginas web

HTML:

Es un tipo de texto que indica en que ubicación del espacio de una pagina web debe de ir un elemento este elemento puede ser una imagen, video, texto, etc.

LaTeX:

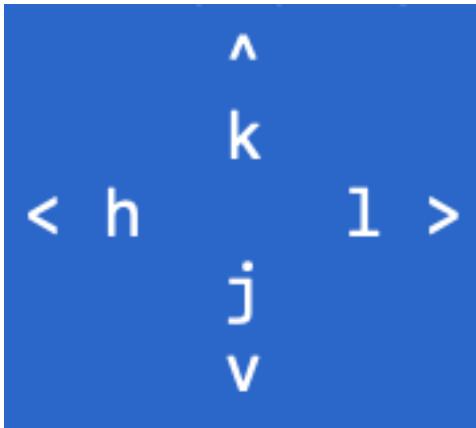
Editor de texto plano que lo pasa a caracteres especiales sobretodo usado para las matemáticas tales como x^2 seria x^2

CSV:

Texto plano usado para representar tablas indicando sus columnas con una coma

Tutor de vim

1.1 como moverte



Se usan estas letras para moverte o bien las flechas del teclado, pero así es mas rápido

1.2 salir del archivo



Así se sale

```
:wq <INTRO>
```

Y así se sale guardando el archivo

Es muy importante poner los :

1.3 borrar letras

```
---> La vvaca saltóó soobree laa luuuuna.
```

Con x puedes borrar cualquier carácter

```
---> La vaca saltó sobre la luna.
```

1.4 insertar texto

```
---> Flta texto en esta .  
---> Falta algo de texto en esta línea.
```

Con la letra i puedes empezar a insertar texto

```
-- INSERT --
```

```
---> Falta algo de texto en esta línea.  
---> Falta algo de texto en esta línea.
```

2.1 borrar palabras

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

Se pueden borrar palabras completas con **dw**

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

2.2 borrar desde el cursor hasta el final de la línea

```
---> Alguien ha escrito el final de esta línea dos veces. esta línea dos veces.
```

Se puede borrar todo lo que este enfrente del cursor con **d\$**

```
---> Alguien ha escrito el final de esta línea dos veces.
```

2.3 uso del comando para borrar el número es el número de veces que se va a ejecutar la acción

[número]	d	objeto	o	d	[número]	objeto
w	-	desde el cursor hasta el final de la palabra, incluyendo el espacio.				
e	-	desde el cursor hasta el final de la palabra, SIN incluir el espacio.				
\$	-	desde el cursor hasta el final de la línea.				

2.4 borrar una línea entera

```
1) Las rosas son rojas,  
2) El barro es divertido,  
3) El cielo es azul,  
4) Yo tengo un coche,  
5) Los relojes marcan la hora,  
6) El azucar es dulce,  
7) Y así eres tu.
```

Con **dd** se puede borrar una línea completa sin importar la ubicación del cursor

```
1) Las rosas son rojas,  
3) El cielo es azul,  
5) Los relojes marcan la hora,  
6) El azucar es dulce,  
7) Y así eres tu.
```

2.5 Undo

```
---> Corrrija los errores dee esttta línea y vuuelva a ponerlos coon deshacer.
```

Con **u** se deshace la ultima acción realizada con **U** se deshace los cambios de una línea entera

```
---> Coriija los errores dee esttta línea y vuuelva a ponerlos coon deshacer.
```

Corregimos algunos errores con **x**

```
1 change; before #19 13 seconds ago
```

Después usamos **u**

```
---> Corriija los errores dee esttta línea y vuuelva a ponerlos coon deshacer.
```

Corregimos todos los errores

```
---> Corija los errores de esta línea y vuelva a ponerlos con deshacer.
```

Usamos **U**

```
---> Corriija los errores dee esttta línea y vuuelva a ponerlos coon deshacer.
```

3.1 Put

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

Con **p** podemos pegar lo ultimo que borramos

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

3.2 Replace

```
---> ¡Cuendo esta línea fue rscrita alguien pulso algunas teclas equibocadas!  
---> ¡Cuando esta línea fue escrita alguien pulsó algunas teclas equivocadas!
```

Con **r** podemos sustituir el carácter sobre el que estamos

```
---> ¡Cuando esta línea fue escrita alguien pulso algunas teclas equivocadas!  
---> ¡Cuando esta línea fue escrita alguien pulsó algunas teclas equivocadas!
```

3.3 Change

```
---> Esta lubrs tiene unas pocas pskavtad que corregir usem el mandato change.  
---> Esta línea tiene unas pocas palabras que corregir usando el mandato change.
```

Con **cw** podemos remplazar todos los caracteres desde el cursor hasta un espacio

```
---> Esta línea tiene unas pocas palabras que corregir usando el mandato change.  
---> Esta línea tiene unas pocas palabras que corregir usando el mandato change.
```

3.4

```
---> El final de esta línea necesita alguna ayuda para que sea como la segunda.  
---> El final de esta línea necesita ser corregido usando el mandato c$.
```

Al igual que el comando para borrar el comando para cambiar utilizan objetos

[número]	c	objeto	O	c	[número]	objeto
---		El final de esta línea necesita	a		alguna ayuda para que sea como la segunda.	
---		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 final de esta línea necesita			ser corregido usando el mandato c\$.	

4.1

```
"vim-tutor.txt" [Modified] line 398 of 766 --51%-- col 1
```

Con **ctrl g** nos muestra la línea en la que estamos

```
mandatos.

Para lecturas y estudios posteriores se recomienda el libro:
    Learning the Vi Editor - por Linda Lamb
    Editorial: O'Reilly & Associates Inc.
Es un buen libro para llegar a saber casi todo lo que desee hacer con Vi.
La sexta edición incluye también información sobre Vim.

Este tutorial ha sido escrito por Michael C. Pierce y Robert K. Ware,
Colorado School of Mines utilizando ideas suministradas por Charles Smith,
Colorado State University.
E-mail: bware@mines.colorado.edu.

Modificado para Vim por Bram Moolenaar.

~~~~~

Traducido del inglés por:

Eduardo F. Amatria
Correo electrónico: eferna1@platea.pntic.mec.es

~~~~~
```

Con **Mayus g** vamos al final del archivo

```
** Pulse CTRL-g para mostrar su situación en el fichero y su estado.
   Pulse MAYU-G para moverse a una determinada línea del fichero. **

Nota: ¡¡Lea esta lección entera antes de ejecutar alguno de los pasos!!

1. Mantenga pulsada la tecla Ctrl y pulse g . Aparece una línea de estado
   al final de la pantalla con el nombre del fichero y la línea en la que
   está situado. Recuerde el número de la línea para el Paso 3.

2. Pulse Mayu-G para ir al final del fichero.

3. Escriba el número de la línea en la que estaba y después Mayu-G. Esto
   le volverá a la línea en la que estaba cuando pulsó Ctrl-g.
   (Cuando escriba los números NO se mostrarán en la pantalla).

4. Si se siente confiado en poder hacer esto ejecute los pasos 1 a 3.

~~~~~
                          Lección 4.2: EL MANDATO «SEARCH» (buscar)
~~~~~

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

Luego volvemos escribiendo el numero de la línea seguido ponemos **Mayus g**
4.2 Search

```
/.errroor
```

Al escribir esto se busco la palabra error

```
«errroor»
```

Y así se busca para arriba

```
? .errroor
```

4.3 Búsqueda de corchetes

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

Si nos situamos en el final de un corchete o paréntesis y ponemos % nos mandara al principio

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

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

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

4.4 Cambiar errores

```
Laas mejores épocas para ver laas flores son laas primaveras.
```

Escribimos esto para cambiar “laas” por “las” pero solo la primera aparición

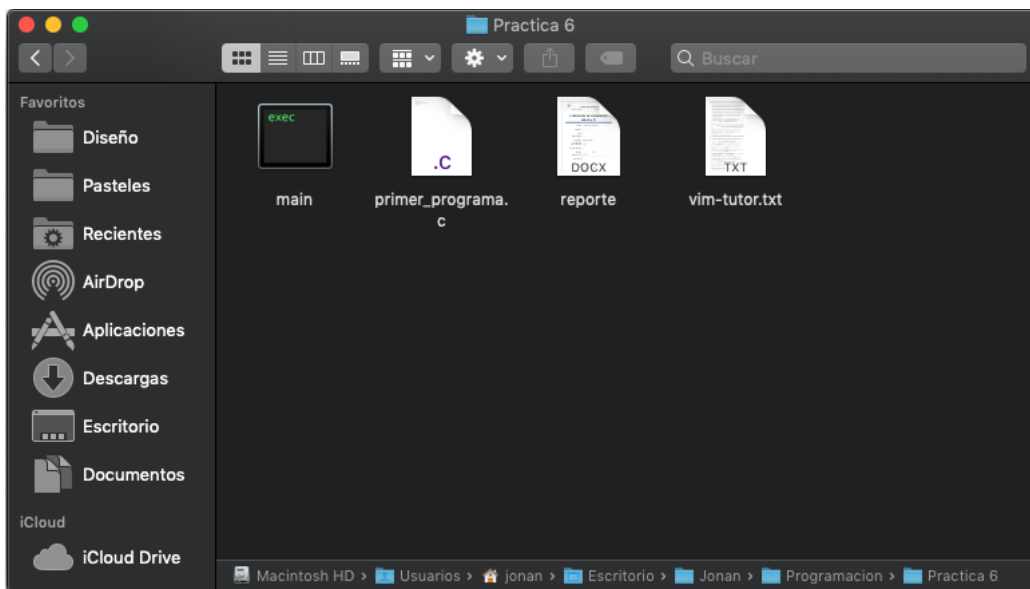
```
:s/laas/las/
```

```
Laas mejores épocas para ver las flores son laas primaveras.
```

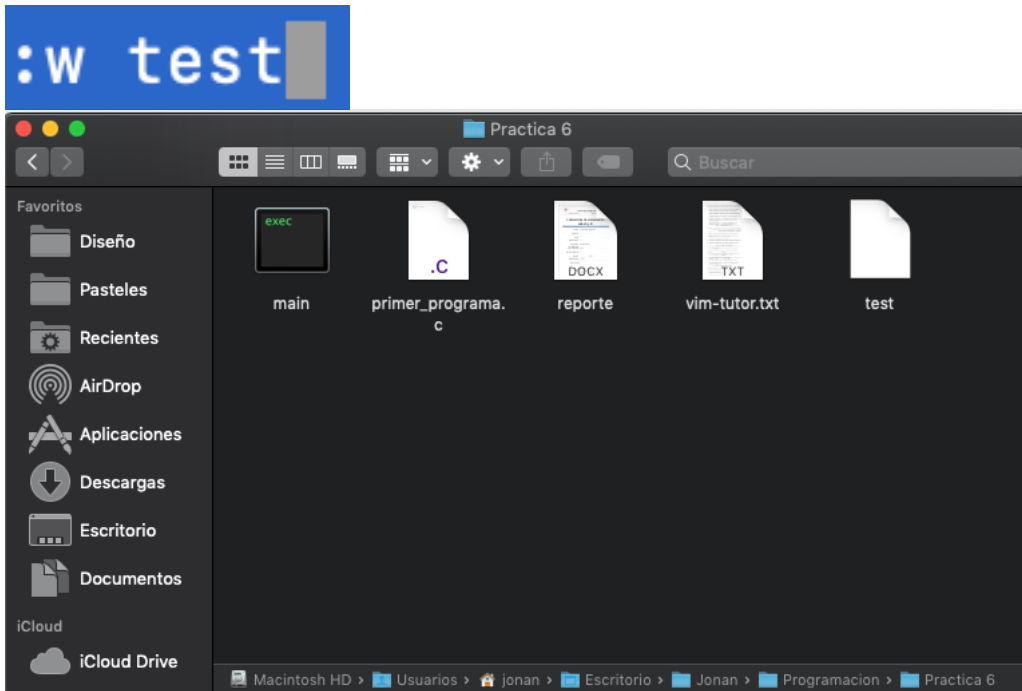
5.1

Con :! Nos deja usar cualquier comando de la computadora

5.2 Guardar archivo como



Aquí tengo la carpeta donde esta el archivo vimtutor

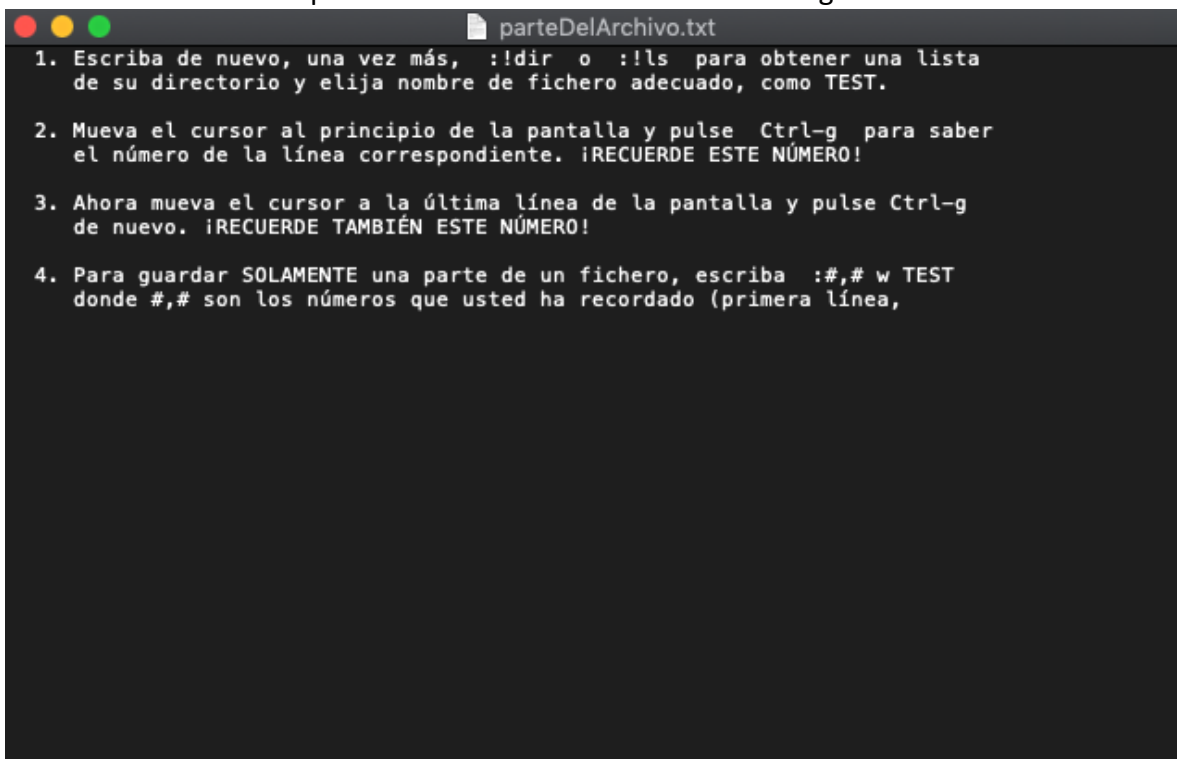


Luego de ejecutar ese comando se observa que se guardo el archivo con el nuevo nombre

5.3 Guardar solo algunas lineas



Los dos numeros corresponden al numero de lineas asi solo se guardaron esas lineas



6.1 Abrir líneas

Con **o** podemos abrir una línea debajo del cursor y con **O** una línea arriba del cursor

6.2 Append

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

Si pulsamos **A** nos mandara al final de la línea para añadir texto si solo pulsamos **a** sera al carácter siguiente donde nos mande

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

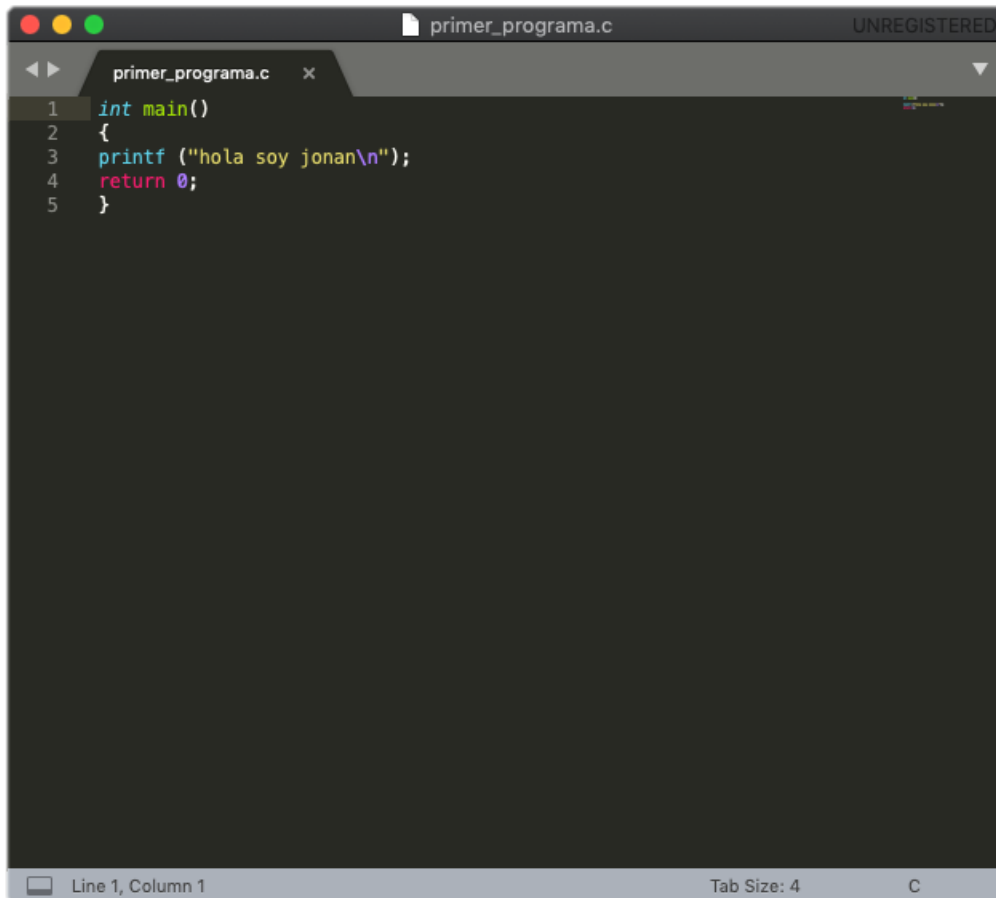
6.3

```
---> Para hacer que esta línea sea igual que la anterior use las teclas.
---> Para hacer que esta línea sea igual que la siguiente escriba R y el texto.
```

Si nos situamos en una palabra y pulsamos **R** lo que vamos a hacer es reemplazar todo en adelante

```
---> 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.
```

Primer programa en c

A screenshot of a code editor window titled 'primer_programa.c' with a 'UNREGISTERED' watermark. The editor shows a C program with the following code:

```
1 int main()
2 {
3     printf ("hola soy jonan\n");
4     return 0;
5 }
```

The status bar at the bottom indicates 'Line 1, Column 1', 'Tab Size: 4', and 'C'.

Esta es la estructura básica de un programa en C

```
Practica 6 — -bash — 80x24
Last login: Tue Oct  1 20:34:26 on ttys000
iMac-J:~ jonan$ cd Desktop/
iMac-J:Desktop jonan$ cd Jonan/
iMac-J:Jonan jonan$ Programacion/
-bash: Programacion/: is a directory
iMac-J:Jonan jonan$ cd Programacion/
iMac-J:Programacion jonan$ cd Practica\ 6/
iMac-J:Practica 6 jonan$ gcc primer_programa.c -o main
primer_programa.c:3:1: warning: implicitly declaring library function 'printf'
      with type 'int (const char *, ...)' [-Wimplicit-function-declaration]
printf ("hola soy jonan\n");
^
primer_programa.c:3:1: note: include the header <stdio.h> or explicitly provide
      a declaration for 'printf'
1 warning generated.
iMac-J:Practica 6 jonan$
```

Se compila con esa instrucción

```
iMac-J:Practica 6 jonan$ ./main
hola soy jonan
iMac-J:Practica 6 jonan$
```

Y se corre de esta manera

Conclusión

Para poder programar en C tenemos que manejar texto plano todo lo que queramos va a ser en puro texto debemos de aprender a ser mas eficientes a la hora de escribir además de que con una herramienta como la terminal podemos compilar y ejecutar el programa.