



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Alejandro Pimentel

Asignatura: Fundamentos de Programación

Grupo: 3

No de Práctica(s): Practica 6

Integrante(s): Garcés Gallardo Julio César

*No. de Equipo de
cómputo empleado:*

No. de Lista o Brigada: 14

Semestre: Primer Semestre

Fecha de entrega: 30 de septiembre del 2019

Observaciones:

CALIFICACIÓN: _____

Tema: Entorno de 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.

Introducción: Se implementará un nuevo lenguaje, este lenguaje C se ocupa para crear programas a través de algoritmos. Este es un lenguaje orientado a la implementación de sistemas operativos. C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistema, aunque también se utiliza para crear aplicaciones.

Desarrollo:

Actividad 1: Hacer una investigación muy somera acerca de los siguientes tipos de archivos:

- **Txt:** La extensión TXT representa "textfile" (archivo de texto), que sustituyó a su antiguo nombre "flatfile" (archivo sin formato). Este archivo informático estructura series de líneas de texto. El final del archivo se identifica habitualmente con un carácter especial definido como un marcador "end-of-file" (final de archivo), ubicado a continuación de la última línea de texto. 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 archivos de texto ASCII sin formato es que se pueden guardar como .txt documentos en archivos de tamaño reducido.
- **Markdown:** Markdown es un lenguaje de marcado que facilita la aplicación de formato a un texto empleando una serie de caracteres de una forma especial. En principio, fue pensado para elaborar textos cuyo destino iba a ser la web con más rapidez y sencillez que si estuviésemos empleando directamente HTML. Y si bien ese suele ser el mejor uso que podemos darle, también podemos emplearlo para cualquier tipo de texto, independientemente de cual vaya a ser su destino. Como explica "John Gruber": <http://daringfireball.net/projects/markdown/>, uno de sus creadores, Markdown es realmente dos cosas: por un lado, el lenguaje; por otro, una herramienta de software que convierte el lenguaje en HTML válido.
- **Html:** HTML significa Hypertext Markup Language, y estos archivos HTML se implementan en su mayoría en forma de páginas estáticas de sitios web. HTML se puede utilizar para integrar ciertos atributos de formato y las especificaciones de diseño en el contenido de estas páginas web. Hay una gran cantidad de aplicaciones que se pueden utilizar para crear y editar estos archivos HTML, y muchos de los navegadores web se pueden usar para abrir y ver el contenido de archivos en él .html formato. Los códigos HTML implementados en estos .html archivos son analizados por los navegadores web, lo que significa que está oculto a los espectadores. Microsoft Bloc de notas se puede utilizar para crear archivos HTML y también abrir y ver su código fuente.

- **LaTeX:** La extensión de archivo LATEX se utiliza para un sistema de documentos de preparación destinada a la composición tipográfica de alta calidad. Por lo general se prefiere para medianas y grandes documentos científicos o técnicos, sino también utilizado para la publicación de cualquier forma. Este sistema de documento se ofrece de forma gratuita a cualquier autor, pero se recomienda leer primero la LPPL antes de crear cualquier trabajo y modificado. Esto permite a los autores que utilizan este sistema para dar información básica, como el texto del artículo, título, nombre del autor, la fecha y el título del documento.
- **Csv:** La extensión de archivo CSV significa Comma Separated Values (Valores separados por comas). El formato es utilizado en muchos programas de bases de datos, hojas de cálculo y gestores de contactos para almacenar listas de información. Como un archivo de texto, el formato es ampliamente compatible. Un fichero CSV es un archivo de texto que contiene una serie de valores separados por comas. Los valores pueden ser cualquier cosa, desde números de un presupuesto de una hoja de cálculo, hasta nombres y descripciones de una lista de clientes de un negocio.

Actividad 2: Seguir el tutor de vim.

```

MINGW64:/c:/Users/Invitado/Downloads
---> Esto provoca la salida del editor SIN guardar ningún cambio que se haya
      hecho. Si quiere guardar los cambios y salir escriba:
              :wq <INTRO>

3. Cuando vea el símbolo del sistema, escriba el mandato que le trajo a este
   tutor. Este puede haber sido: vimtutor <INTRO>
   Normalmente se usaría:      vim tutor <INTRO>

---> 'vim' significa entrar al editor, 'tutor' es el fichero a editar.

4. Si ha memorizado estos pasos y se se siente con confianza, ejecute los
   pasos 1 a 3 para salir y volver a entrar al editor. Después mueva el
   cursor hasta la Lección 1.3.
-----
Lección 1.3: EDICIÓN DE TEXTO - BORRADO

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

1. Mueva e 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.

NOTA: A medida que vaya avanzando en este tutor no intente memorizar, aprenda
practicando.

-----
Lección 1.4: EDICIÓN 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.

vim-tutor.txt[+] [unix] (16:04 30/09/2019)

```

```

Lección 1.4: EDICIÓN 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.
---> Flta texto en esta . Buenas tardes profe, saqueme 10 porfavor :(
---> Falta algo de texto en esta línea. muchas gracias

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

```

```

RESUMEN DE LA LECCIÓN 1

1. El cursor se mueve utilizando las teclas de las flechas o las teclas hjkl.
   h (izquierda)  j (abajo)    k (arriba)  l (derecha)
2. Para acceder a Vim (desde el símbolo del sistema %) escriba:
   vim FILENAME <INTRO>
3. Para salir de Vim escriba: <ESC> :q! <INTRO> para eliminar todos
   los cambios.
4. Para borrar un carácter sobre el cursor en modo Normal pulse: x
5. Para insertar texto en la posición del cursor estando en modo Normal:
   pulse i escriba el texto pulse <ESC>

NOTA: Pulsando <ESC> se vuelve al modo Normal o cancela un mandato no deseado
      o incompleto.

Ahora continúe con la Lección 2.

```

```

Lección 2.1: MANDATOS PARA BORRAR
vim-tutor.txt[+] [unix] (16:04 30/09/2019)
-- INSERTAR --

```

Ahora continúe con la Lección 2.

```

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 pásalo bien que no pertenecen a esta frase.

```

```

Lección 2.2: MAS 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.

```

```

vim-tutor.txt[+] [unix] (16:04 30/09/2019)

```

Lección 2.4: UNA EXCEPCIÓN 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.

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 con deshacer.

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

RESUMEN DE LA LECCIÓN 2

1. Para borrar desde el cursor hasta el final de una palabra pulse: dw
2. Para borrar desde el cursor hasta el final de una línea pulse: d\$

vim-tutor.txt[+] [unix] (16:04 30/09/2019)

1 cambio; antes #25 3 seconds ago



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?

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 rescrita alguien pulso algunas teclas equivocadas!

---> ¡cuando esta línea fue rescrita alguien pulsó algunas teclas equivocadas!

Lección 3.3: EL MANDATO «CHANGE» (cambiar)

vim-tutor.txt[+] [unix] (16:04 30/09/2019)

-- INSERTAR --



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.

Lección 3.4: MAS 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 O 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\$.

RESUMEN DE LA LECCIÓN 3

1. Para sustituir texto que ha sido borrado, pulse p . Esto pone el texto
 vim-tutor.txt[+] [unix] (16:04 30/09/2019)

-- INSERTAR --



Lección 4.2: EL MANDATO «SEARCH» (buscar)

**** 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 /errorroo <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.

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

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 desparejos.

Lección 4.4: UNA FORMA DE CAMBIAR ERRORES

vim-tutor.txt[+] [unix] (16:04 30/09/2019)

/errorroo <INTRO>



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 desaparejos.

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.

---> Laas mejores épocas para ver laas flores son laas 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.

RESUMEN DE LA LECCIÓN 4

vim-tutor.txt[+] [unix] (16:04 30/09/2019)



Escribe aquí para buscar



paréntesis, corchetes o llaves desaparejos.

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.

RESUMEN DE LA LECCIÓN 4

1. Ctrl-g muestra la posición del cursor en el fichero y su estado. Mayu-G mueve el cursor al final del fichero. Un número de línea seguido de Mayu-G mueve el cursor a la línea con ese número.
2. Pulsando / seguido de una frase busca la frase hacia ADELANTE. Pulsando ? seguido de una frase busca la frase hacia ATRÁS. Después de una búsqueda pulse n para encontrar la aparición siguiente en la misma dirección.
3. Pulsando % cuando el cursor esta sobre (,), [,], { o } localiza la pareja correspondiente.
4. Para cambiar viejo por nuevo en una línea pulse :s/viejo/nuevo
Para cambiar todos los viejo por nuevo en una línea pulse :s/viejo/nuevo/g
Para cambiar frases entre dos números de líneas pulse :#,#s/viejo/nuevo/g
Para cambiar viejo por nuevo en todo el fichero pulse :%s/viejo/nuevo/g
Para pedir confirmación en cada caso añada 'c' :%s/viejo/nuevo/gc

vim-tutor.txt[+] [unix] (16:04 30/09/2019)



Escribe aquí para buscar



versión del fichero.

RESUMEN DE LA LECCIÓN 5

1. `!!mandato` ejecuta un mandato externo.

Algunos ejemplos útiles son:

- `:!dir` - muestra el contenido de un directorio.
 - `!!del NOMBRE_DE_FICHERO` - borra el fichero `NOMBRE_DE_FICHERO`.
2. `:#,#w NOMBRE_DE_FICHERO` guarda desde las líneas `#` hasta la `#` en el fichero `NOMBRE_DE_FICHERO`.
 3. `:r NOMBRE_DE_FICHERO` recupera el fichero del disco `NOMBRE_DE_FICHERO` y lo inserta en el fichero en curso a partir de la posición del cursor.

/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.
Pulsando una `O` mayúscula se abre una línea SOBRE la que está el cursor.
2. Pulse una `a` para insertar texto DESPUÉS del carácter sobre el cursor.
Pulsando una `A` mayúscula añade automáticamente texto al final de la línea.
3. Pulsando una `R` mayúscula se entra en modo Replace hasta que, para salir, se pulse `<ESC>`.
4. Escribiendo `[:set xxx]` fija la opción «xxx»

Lección 7: MANDATOS PARA LA AYUDA EN LÍNEA

**** Utilice el sistema de ayuda en línea ****

Vim dispone de un sistema de ayuda en línea. Para activarlo, pruebe una de estas tres formas:

- pulse la tecla `<AYUDA>` (si dispone de ella)
- pulse la tecla `<F1>` (si dispone de ella)
- escriba `:help <INTRO>`

Escriba `:q <INTRO>` para cerrar la ventana de ayuda.

Puede encontrar ayuda en casi cualquier tema añadiendo un argumento al mandato `[:help]` mandato. Pruebe éstos:

```
:help w <INTRO>
:help c_<T <INTRO>
:help insert-index <INTRO>
```

Aquí concluye el tutor de Vim. Está pensado para dar una visión breve del editor Vim, lo suficiente para permitirle usar el editor de forma bastante sencilla. Está muy lejos de estar completo pues Vim tiene muchísimos más

Vim-tutor.txt[+] [unix] (16:04 30/09/2019)

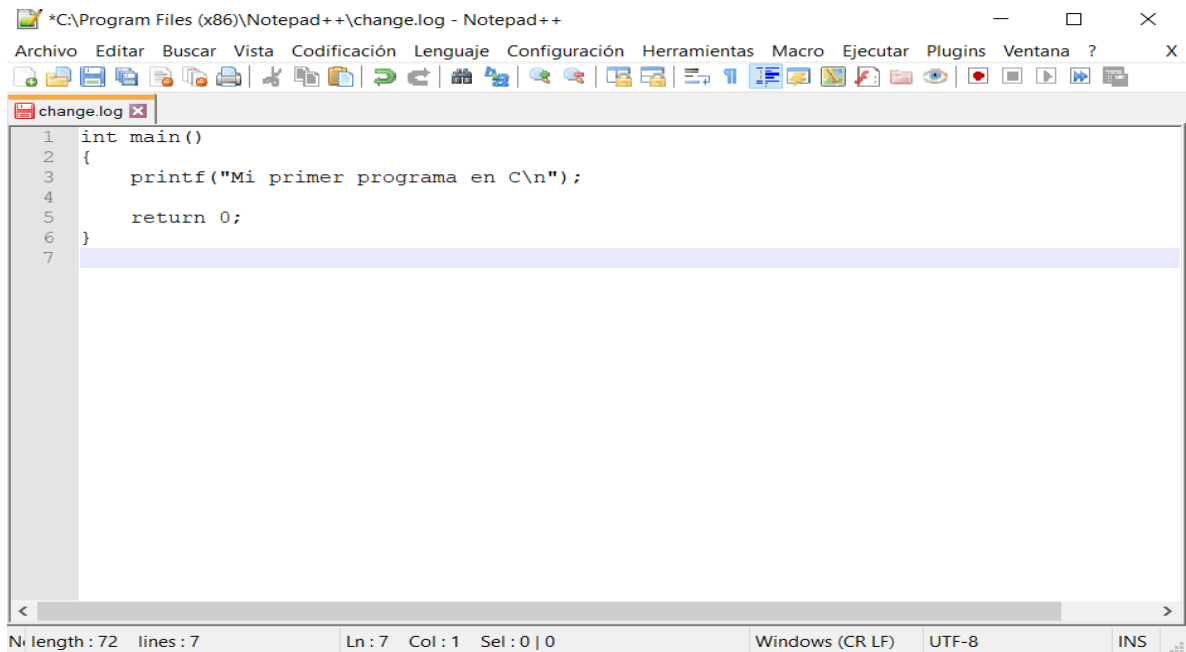


Escribe aquí para buscar



Actividad 3: Después de conocer los diferentes editores de texto, intenté probar algunos, pero la verdad solo pude ocupar Notepad++ y seguí los pasos de la práctica y logré lo siguiente:

1. Primero coloque los pasos en el editor de textos.

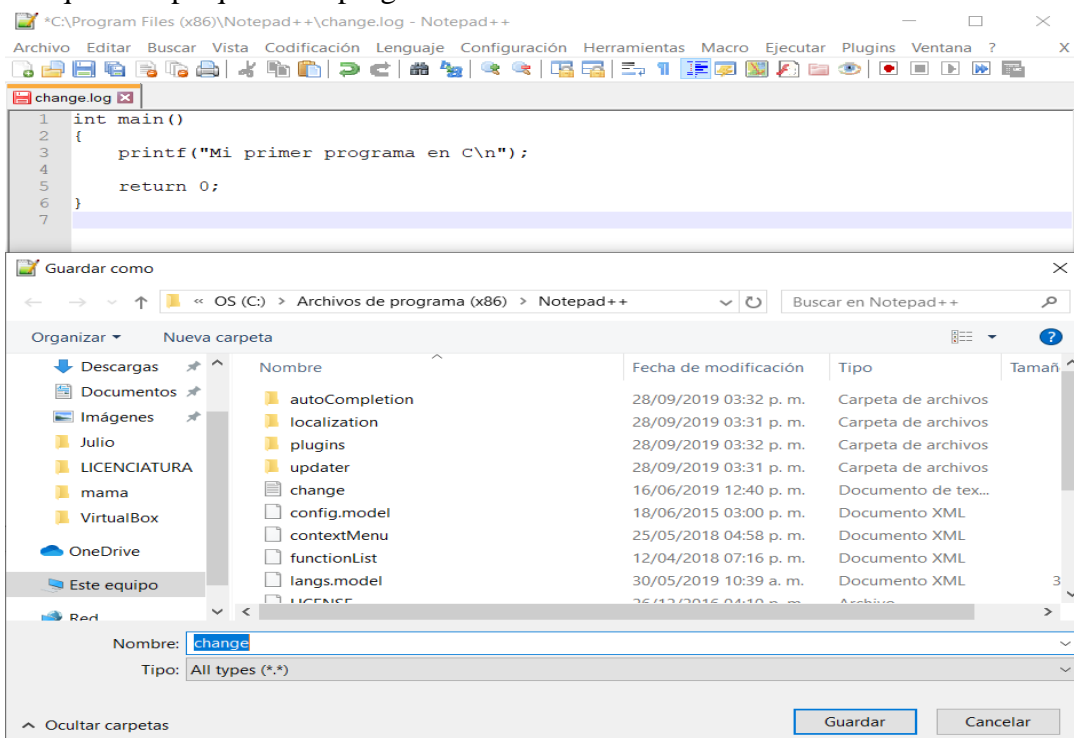


The screenshot shows the Notepad++ application window with the file `change.log` open. The code is as follows:

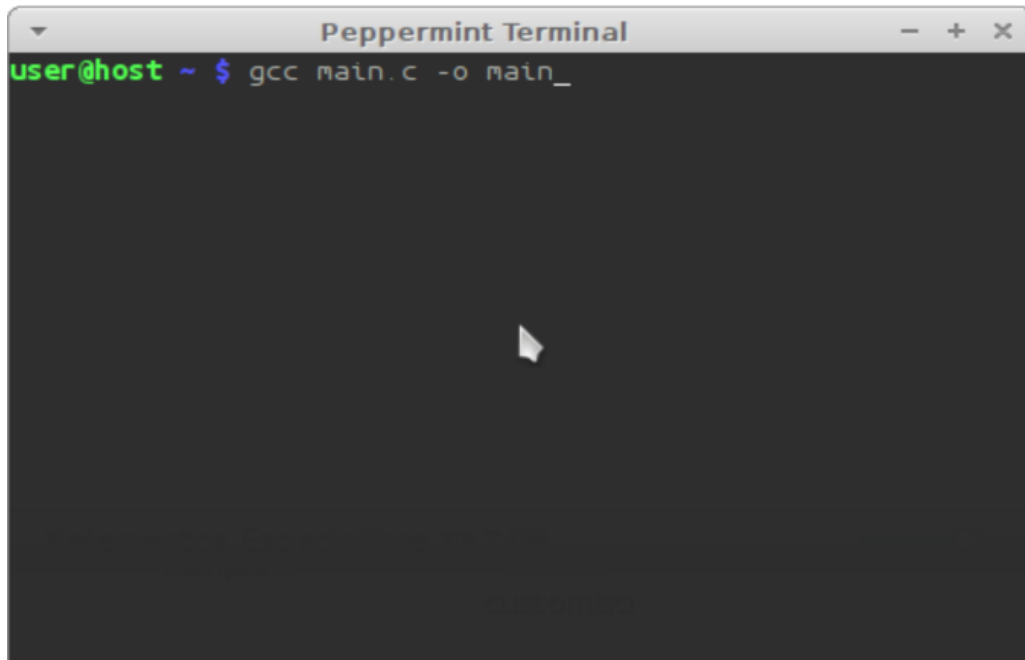
```
1 int main()  
2 {  
3     printf("Mi primer programa en C\n");  
4  
5     return 0;  
6 }  
7
```

The status bar at the bottom indicates: `Ln: 7 Col: 1 Sel: 0 | 0`, `Windows (CR LF)`, `UTF-8`, and `INS`.

2. Después guarde el archivo. Aquí es importante poner el `.c` después del nombre para que se sepa que es un programa.



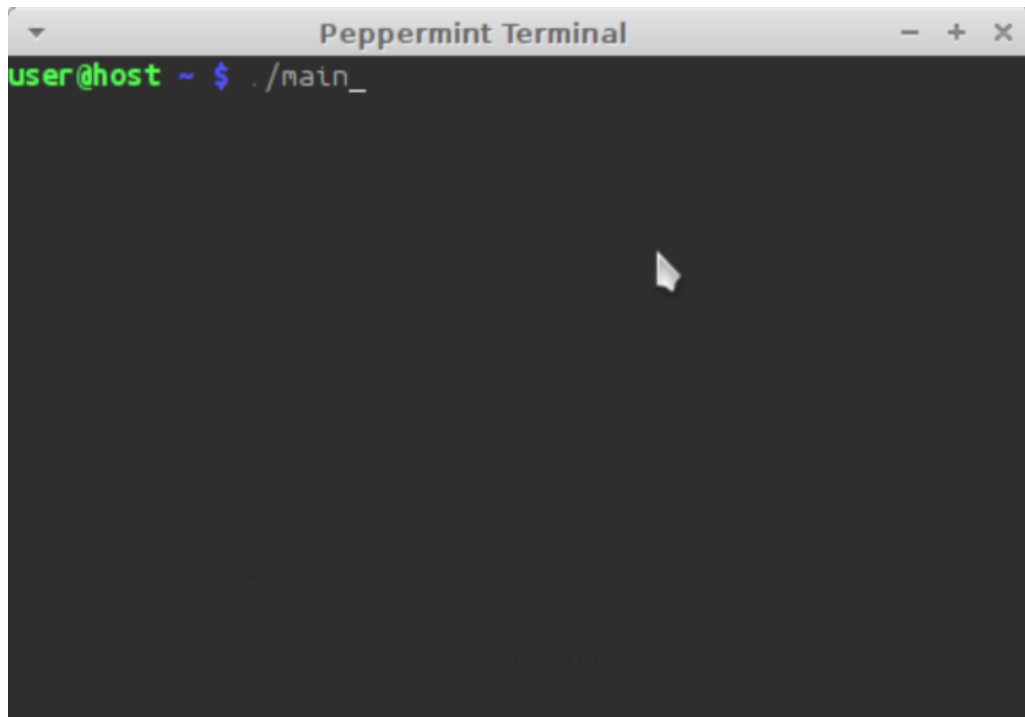
3. Después de guárdalo se debe colocar en la terminal y seguir la siguiente estructura, esto para compilar el programa.



A screenshot of a terminal window titled "Peppermint Terminal". The prompt is "user@host ~ \$". The command entered is "gcc main.c -o main_". The terminal is dark with green text for the prompt and blue text for the command.

```
user@host ~ $ gcc main.c -o main_
```

4. Para terminar, se debe correr el programa y para esto se debe seguir la siguiente estructura.



A screenshot of a terminal window titled "Peppermint Terminal". The prompt is "user@host ~ \$". The command entered is "./main_". The terminal is dark with green text for the prompt and blue text for the command.

```
user@host ~ $ ./main_
```

Conclusiones: Para mi este tipo de algoritmos son muy útiles y muy buenos, ya que tuve la oportunidad de ejecutar uno de estos y la verdad nunca pensé poder hacer eso de una forma tan fácil, literal solo metía un dato y el programa me imprimía lo que había puesto, pero obviamente para desarrollar algoritmos complejos el nivel de dificultad sube, pero como ya dije, para algoritmos cortos, es fácil realizar entorno c.