Programación III

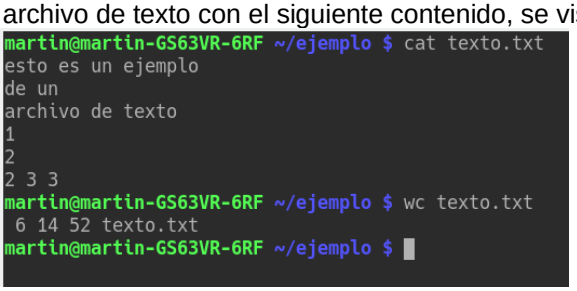
Proyecto Segundo parcial

Introducción:

El comando 'wc' en linux sirve para obtener estadísticas de un archivo de texto, se utiliza escribiendo ***wc archivo.txt***, donde archivo.txt es el nombre del archivo.

La salida en pantalla son 3 numeros que representan la cantidad de letras, palabras y lineas del texto.

Por ejemplo, si se le pasa un archivo de texto con el siguiente contenido, se visualiza lo siguiente:



Se plantea realizar un programa similar y mas potente que permita realizar mejores análisis de textos.

Enunciado:

Realizar una aplicación informática que realice las funciones básicas del comando "wc" de linux y las mejoras planteadas a continuación:

1. Las funciones básicas son:
    1. contar letras
    2. contar palabras
    3. contar líneas
2. Indicar cuantas palabras diferentes hay en el texto.
3. Mostrar las palabras diferentes en orden alfabético.
4. Mostrar las palabras diferentes en orden de mas ocurrencias. El usuario puede informar un grupo de palabras a ignorar mediante otro archivo de texto.
5. Informar la cantidad de veces que aparece una o un grupo de palabras solicitadas por el usuario.

Manual de uso:

Uso normal:

wce.exe file.txt

Argumentos posibles:

-palabras [n] Mostrará las **n** primeras palabras en orden alfabético. Si n no es ingresado o vale 0, mostrará todas las palabras.

-ocurrencias [n] Mostrará las **n** palabras y la cantidad de ocurrencias de cada palabra ordenadas de mayor a menor por cantidad de ocurrencias. Si n no es ingresado o vale 0, mostrará todas las palabras

-mostrar [palabra],[palabra] Mostrará la o las palabras pasadas como argumento ordenadas por ocurrencia.

-excluir [palabra],[palabra] Modifica los comandos ocurrencias y palabras haciendo que no muestren las palabras pasadas como argumentos.

-excluirf [archivo.txt] Modifica los comandos ocurrencias y palabras haciendo que no muestren las palabras contenidas en el archivo.txt

Ejemplo de uso:

// **file.txt** **el veloz murciélago hindú**
comía feliz cardillo y kiwi.
La cigüeña tocaba el saxofón
detrás del palenque de paja

// **ign.txt**
cardillo

```
# wce.exe file.txt
```
lineas: 4
palabras: 19
letras: 119
palabras diferenes: 18

```
# wce.exe -palabras 3 file.txt
```
cardillo
cigüeña
comía

```
# wce.exe -mostrar el file.txt
```
el 2

```
# wce.exe -mostrar “el, cardillo” file.txt
```
2 el
1 cardillo

```
# wce.exe -ocurrencias file.txt
```
2 el
1 cardillo
1 cigüeña
...

```
# wce.exe -ocurrencias -excluir cardillo file.txt
```
2 el
1 cigüeña
...

```
# wce.exe -ocurrencias -excluirf ign.txt file.txt
```
2 el
1 cigüeña
…

Detalles de la implementación:

- Se considera una palabra a letras encerradas entre dos espacios o símbolos distintos a letras, estas deben contener solamente letras (incluidas la ñ, acentos, diereis, etc), los símbolos o signos de puntuación no se contemplan como parte de una palabra.(ej: ‘*¿/#Cigüeña@:’* se cuenta como la palabra ‘*cigüeña’*)
- Dos palabras iguales que solo difieran en letras mayúsculas o minúsculas se considera la misma palabra. (ej: La == la)
- No se pueden utilizar las estructuras de datos de la librería estandar de C++ (STL) ni librerías que realicen operaciones propias a la solución.

Objetivo:

- Ejercitar y afianzar el uso de estructuras de datos mediante una aplicación concisa y de uso general.
- Afianzar la importancia del uso de las estructuras correctas para mejorar el rendimiento de las aplicaciones.
- Tener en cuenta el impacto en performance introducidos por el uso de estructuras equivocadas o mala implementación en código.

Formas de presentación:

- **Defensa del trabajo:** Explicación de lo realizado y defensa de la elección de las estructuras de datos.
- **Informe del desarrollo:** Informe explicando que estructuras se utilizaron, porque se decidió su utilización. También incluir detalles de pruebas, tareas, complicaciones, retrabajos que hayan ocurrido durante el desarrollo del trabajo.
- **Código fuente y comentarios:** Se corregirá el código fuente desde el repositorio de **GitHub** del grupo. El código debe ser prolijo y bien comentado (no sobrecomentar)

Forma de trabajo: Grupos de no más de 2 personas.

Forma de evaluación: Se evaluara de acuerdo a los siguientes criterios:

- Puntualidad en la forma de entrega del trabajo.
- Presentación (interfaz, código, prolijidad)
- Solución óptima al problema dado – velocidad de ejecución, uso de memoria, etc.
- Utilización y adaptación de los códigos proporcionados en clase y del libro.
- Conocimiento de la/las estructura/s de datos utilizadas.
- Exposición y defensa del trabajo.
- Preguntas sobre las estructuras utilizadas a cada integrante.
- No se tolerará la deshonestidad académica que comprenda el plagio de código, lo cual implicará la reprobación del parcial de forma irrevocable.

Cada grupo tendrá 15 minutos para presentar lo desarrollado el día de la entrega.

La plantilla y creación de grupo esta disponible en el siguiente REPO:

https://classroom.github.com/g/Ccj-ubtM