ENTREGA 1 - PROYECTO

LUIS FELIPE AYALA URQUIZA
NICOLÁS ESTEBAN BAYONA ORDOÑEZ
JUAN SEBASTIÁN HERRERA GUAITERO

ANDREA DEL PILAR RUEDA OLARTE
PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
ESTRUCTURAS DE DATOS
BOGOTA D.C
20 DE FEBRERO DE 2021

Índice

Descripción del funcionamiento del programa

1. Inicia la ejecución del programa
2. El sistema muestra un carácter "$"
3. El sistema espera a que el usuario ingrese una instrucción
4. El sistema espera a que el usuario finalice de ingresar el comando que desea ejecutar y pulse la tecla enter
5. El sistema analiza la entrada y decide a qué función debería dirigirse la entrada (separada por el carácter ' ' como comando y argumento(opcional))
6. El repite indefinidamente las instrucciónes (2-5) hasta que el usuario ingrese la palabra "salir", allí el sistema decide que debe de llamar al método "exit" y termina la ejecución del programa.

Descripción de entradas:

Las entradas deben coincidir con el formato [comando] [argumento] para que el sistema funcione bien. No obstante, el sistema está diseñado teniendo en consideración casos en los que esto no necesariamente se cumpla, para cumplir con este fin, el sistema tiene varios mecanismos para asegurar su correcto funcionamiento en todo momento.

Descripción de salidas:

El sistema retorna las salidas de los diferentes comandos que puede ejecutar, para la entrega 0 serían:

1. inicializar [diccionario.txt]
   POSIBLES SALIDAS
- (Diccionario ya inicializado): El diccionario ya ha sido iniciado.
- (Archivo no existe): El archivo no existe o no puede ser leído.
- (Resultado exitoso): El diccionario se ha iniciado correctamente.

2. iniciar_inverso [diccionario.txt]
   POSIBLES SALIDAS:
- (Diccionario ya inicializado): El diccionario inverso ya ha sido iniciado.
- (Archivo no existe): El archivo diccionario no existe o no puede ser leído.
- (Resultado exitoso): El diccionario se ha iniciado correctamente.

3. puntaje [palabra]
   POSIBLES SALIDAS:
- (Palabra no existe): La palabra no existe en el diccionario.
- (Letras inválidas): La palabra contiene símbolos inválidos.
- (Resultado exitoso): La palabra tiene un puntaje de [puntaje2]

4. salir

→ Si el usuario ingresa un comando diferente, pueden ocurrir dos cosas:

1. Que el comando indicado sea uno de los comandos que hasta el momento no se han implementado, en cuyo caso se mostrará la ayuda del comando especificado.

2. Que el usuario haya escrito algo mal, en cuyo caso se mostrará un mensaje de comando inválido.

Descripción de condiciones:

Para el procedimiento principal no debería de cumplirse ninguna condición específica, puesto que el sistema verifica primero que pueda asegurar su correcto funcionamiento con la entrada ingresada por el usuario para poder otorgar una respuesta correcta para la petición que se realiza.

Descripción de condiciones para los procesos auxiliares:

Para los procedimientos auxiliares no debe de cumplirse ninguna condición específica pues, la función “decide” se encarga de verificar que los argumentos puedan llegar sin complicaciones a las funciones necesarias.

TAD ScrabbleClass

Conjunto mínimo de datos:

- dictionary, lista de cadenas de caracteres, lista de palabras que existen.
- inverse_dictionary, lista de cadenas de caracteres, lista de palabras que existen en orden inverso.
- file_name, cadena de caracteres, guarda el nombre del archivo del cual se tomaron las palabras para inicializar el campo “dictionary”.
- tree, TreeClass, arbol para ordenar las palabras por prefijo o sufijo según sea el caso.
- graph, GraphClass, grafo para organizar las diferentes palabras.

Comportamiento (Operaciones):

- start (diccionario.txt), inicializa el sistema a partir del archivo diccionario.txt, que contiene un diccionario de palabras aceptadas en el idioma inglés (idioma original del juego). El comando debe almacenar las palabras del archivo de forma que sea fácil recuperarlas posteriormente. Las palabras deben ser verificadas para no almacenar aquellas que incluyen símbolos inválidos (como guiones, números y signos de puntuación).
- inverse_start (diccionario.txt), inicializa el sistema a partir del archivo diccionario.txt, que contiene un diccionario de palabras aceptadas en el idioma inglés (idioma original del juego). A diferencia del comando inicializar, este comando almacena las palabras en sentido inverso (leídas de derecha a izquierda), teniendo en cuenta que sea fácil recuperarlas posteriormente. Las palabras también deben ser verificadas para no almacenar aquellas que incluyen símbolos inválidos (como guiones, números y signos de puntuación).
- score (palabra), el comando permite conocer la puntuación que puede obtenerse con una palabra dada, de acuerdo a la tabla de puntuación de cada letra presentada anteriormente. Sin embargo, el comando debe verificar que la palabra sea válida, es decir, que exista en el diccionario (tanto original como en sentido inverso), y que esté escrita con símbolos válidos.
- exit (), termina la ejecución de la aplicación.
- start_tree (diccionario.txt), inicializa el sistema a partir del archivo diccionario.txt, que contiene un diccionario de palabras aceptadas en el idioma inglés (idioma original del juego). A diferencia del comando start, este comando almacena las palabras en uno o más árboles de letras (como se considere conveniente). Las palabras deben ser verificadas para no almacenar aquellas que incluyen símbolos inválidos (como guiones, números y signos de puntuación).
- start_inverse_tree (diccionario.txt), inicializa el sistema a partir del archivo diccionario.txt, que contiene un diccionario de palabras aceptadas en el idioma inglés (idioma original del juego). A diferencia de los comandos “start_inverse” e “start_tree”, este comando almacena las palabras en uno o más árboles de letras, pero en sentido inverso (leídas de derecha a izquierda). Las palabras también deben ser verificadas para no almacenar aquellas que incluyen símbolos inválidos (como guiones, números y signos de puntuación).
- words_by_preffix (prefijo), dado un prefijo de pocas letras, el comando recorre el(los) árbol(es) de letras (construído(s) con el comando “start_tree”) para ubicar todas las

palabras posibles a construir a partir de ese prefijo. A partir del recorrido, se presenta al usuario en pantalla todas las posibles palabras, la longitud de cada una y la puntuación que cada una puede obtener.

- words_by_suffix (sufijo), dado un sufijo de pocas letras, el comando recorre el(los) árbol(es) de letras (construído(s) con el comando “start_inverse_tree”) para ubicar todas las palabras posibles a construir que terminan con ese sufijo. A partir del recorrido, se presenta al usuario en pantalla todas las posibles palabras, la longitud de cada una y la puntuación que cada una puede obtener.
- word_graph (), con las palabras ya almacenadas en el diccionario, el comando construye un grafo de palabras, en donde cada palabra se conecta a las demás si y sólo sí difieren en una única letra (con las demás letras iguales y en las mismas posiciones).
- decide (comando), esta función se encarga de escoger a que función debería dirigirse la ejecución del programa, junto a los argumentos que necesita.
- possible_words (letras), dadas ciertas letras en una cadena de caracteres (sin importar su orden), el comando debe presentar en pantalla todas las posibles palabras válidas a construir, indicando la longitud de cada una y la puntuación que se puede obtener con cada una. En las letras de la cadena de caracteres, puede admitirse un único símbolo comodín (?), el cual representará una letra desconocida y permitirá generar mayores posibilidades de palabras a construir. Para este propósito, el comando debe hacer uso del grafo de palabras construído con el comando “word_graph”.
- check_character (palabra), dada una palabra, verifica que todos los caracteres de esta sean letras de la letra “a” a la letra “z”.
- sumScore (palabra), dada una palabra, me retorna el puntaje específico de esta palabra como número.
- inverse_characters (palabra), me retorna una palabra en orden inverso.
- find_in_dictionaries (palabra), me retorna si una palabra está en ambos diccionarios.

TAD TreeClass

...

TAD GraphClass

…

Diagrama de relación de TAD’s