|  |
| --- |
| **Análisis de Algoritmos y Estructura de Datos** |
| **Informe laboratorio nº4**  **Traductor Little C’s** |
|  |

Roberto Lillo Toloza

Profesor: Mario Inostroza P.

Ayudantes: Javiera Torres M.

Sebastián Garay P.

Nicolás Romero F.

|  |  |  |
| --- | --- | --- |
|  | Santiago - Chile |  |
|  | 2-2018 |  |

Tabla de Contenidos

[Índice de Figuras 4](#_Toc489647714)

[CAPÍTULO 1. Introducción 5](#_Toc489647716)

[CAPÍTULO 2. Descripción de la solución 6](#_Toc489647717)

[2.1 Marco tórico 6](#_Toc489647718)

[2.2 Algoritmos y estructuras de datos 7](#_Toc489647719)

[2.3 Análisis de los resultados 14](#_Toc489647723)

[CAPÍTULO 3. Conclusiones 19](#_Toc489647733)

[CAPÍTULO 4. Manual de Usuario 21](#_Toc489647740)

[4.1 Introducción 21](#_Toc489647723)

[4.2 Cómo compilar y ejecutar el programa 21](#_Toc489647723)

[4.3 Funcionalidades del programa 25](#_Toc489647723)

[4.4 Formas de uso y errores conocidos 25](#_Toc489647723)

Índice de Figuras

2

4

5

7

7

8

8

# Introducción

Luego de todas las situaciones por las que pasó la coordinación, decidieron nuevamente volver a Las Vegas, pero esta vez con el fin de abrir su propio casino en la ciudad. No obstante, se encuentran con el problema de que ninguno de ellos sabe hablar inglés, por lo que se pide a los alumnos implementar un traductor inglés-español.

Este traductor se implementa mediante el uso de árboles binarios ordenados, específicamente un solo árbol para la totalidad del traductor. En este árbol se deben encontrar las palabras en inglés y español, como también las conexiones correspondientes a esta estructura, es decir, padre e hijos de cada nodo. Para la construcción del árbol se otorga un archivo de texto el cual contiene las palabras en español junto a su traducción al inglés.

Ya con estos datos, se decide poner en práctica lo aprendido en cátedra, usando el lenguaje de programación C, perteneciente al paradigma de programación Imperativo Procedural y todo el concepto relacionado a la estructura de datos árbol. Para la implementación del programa en C, se decidió trabajar con el editor de código Visual Studio Code y usar el compilador GCC en las versiones 4.2.1 en MacOS y 6.3.0 en Microsoft Windows.

# Descripción de la solución

**2.1 Marco teórico**

Lenguaje de programación: lenguaje formal que contiene un conjunto de símbolos, reglas sintácticas y semánticas que definen una estructura con la que se pueden especificar instrucciones a un computador.

Paradigma de programación: enfoque particular para diseñar soluciones en programación. Marco de referencia.

C: Lenguaje de programación perteneciente al paradigma Imperativo Procedural, desarrollado entre 1969 y 1972, destacado en su tiempo por necesitar pocas instrucciones en lenguaje máquina.

Imperativo-Procedural: describir términos computacionales a través de declaraciones y seguir estas indicaciones paso a paso.

Arreglos: medio para guardar un conjunto de datos de la misma clase.

Punteros: dato cuyo valor indica la posición en memoria donde se encuentra otro dato.

Árboles: tipo de dato abstracto que imita la jerarquía de un árbol, esto se refiere a tener raíz, subárboles y hojas. Se implementa mediante el enlazamiento de nodos los cuales contienen los datos y las conexiones.

**2.2 Algoritmos y estructuras de datos**

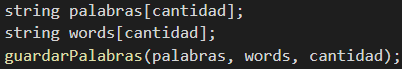
Como se mencionó anteriormente, se provee al estudiante con un único archivo de texto llamado “traductor.in”, el cual contiene una serie de palabras en español acompañadas de su traducción al inglés. Visualmente el archivo se ve de la siguiente forma:



*Figura 1: Formato archivo “traductor.in”.*

Como se puede apreciar en la imagen, cada palabra está separada por un espacio de su traducción al inglés, así mismo todas comienzan con mayúsculas y además la última palabra no presenta un salto de línea.

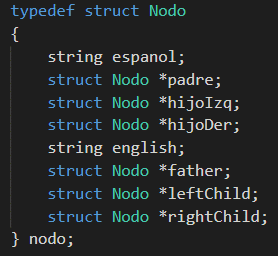
Como la parte fundamental del programa es ingresar los datos desde el archivo de texto para posteriormente realizar las traducciones, se procede a leer el archivo dos veces, la primera lectura tiene la finalidad de reconocer cuantas líneas tiene el archivo, lo cual debido a su formato se traduce en la cantidad de palabras que se deben guardar. Posterior al proceso de contar se implementan dos arreglos del tamaño de la cantidad de las palabras, con el fin de guardar las palabras en español e inglés en ellos por separado. Finalmente se lee nuevamente el archivo y se guardan las palabras en los arreglos para su posterior uso.



*Figura 2: Arreglos contenedores de las palabras.*

Como se puede apreciar en la figura anterior, se decidió implementar el tipo de dato “string”, este consiste en un arreglo de caracteres de tamaño máximo 50. Se escogió este número debido a que la palabra más larga en inglés que se encuentre en un diccionario académico consta de 45 caracteres, por lo que se decidió subir a 50 para evitar posibles errores.

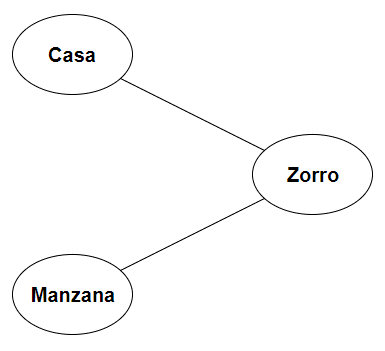
Posterior a la lectura de las palabras se procede a implementar el árbol binario, esto se hace a través de nodos los cuales posee la siguiente estructura.



*Figura 3: Estructura nodo.*

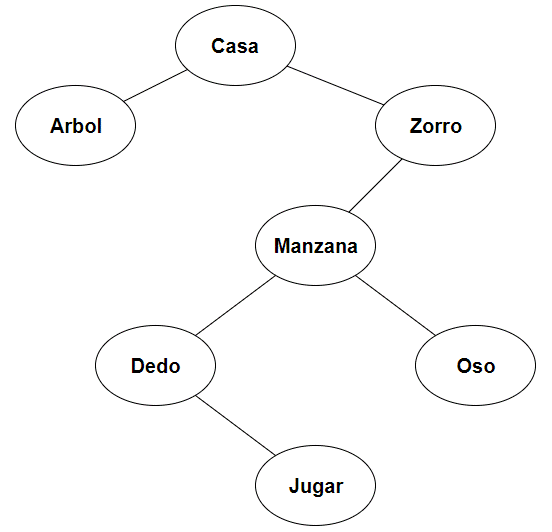
Donde los strings “espanol” y “english” contienen las palabras en sus respectivos idiomas. “Padre”, “hijoIzq” e “hijoDer” corresponden a los punteros a los nodos padre, hijo izquierda e hijo derecha de cada nodo para el idioma español, caso similar con “father”, “leftChild” y “rightChild” que corresponden a las mismas conexiones, pero para el idioma en inglés.

El árbol inicialmente se construye y ordena con las palabras en español, esto se hace con las palabras en el mismo orden con el que vienen en el archivo, usando como raíz la primera palabra de este y ordenando las siguientes palabras según comparación mediante la función strcmp.



*Figura 4: Ejemplo 3 primeras palabras del archivo.*

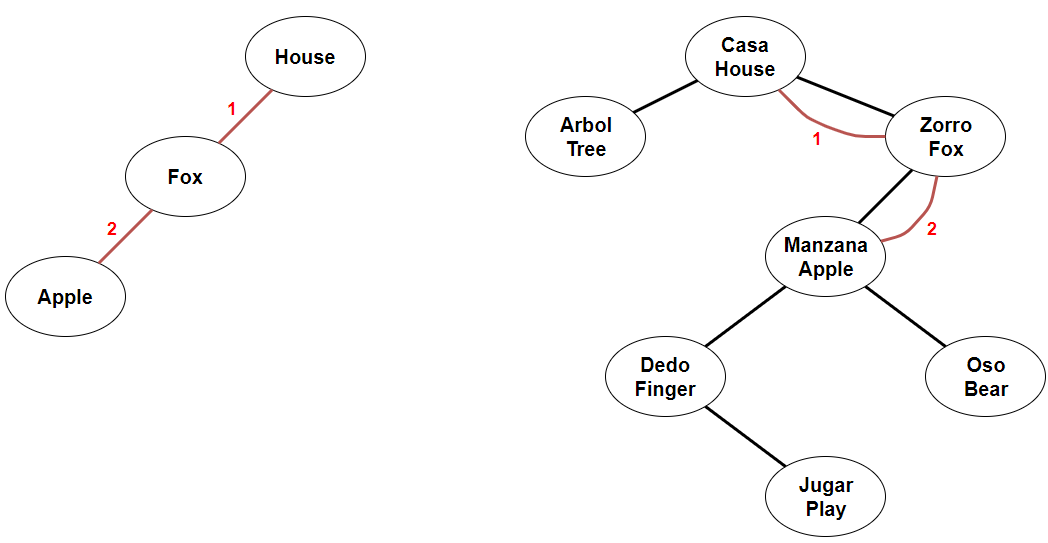
Para el mismo caso del ejemplo el árbol que se tiene al final del proceso es el siguiente.



*Figura 5: Árbol para el archivo de ejemplo.*

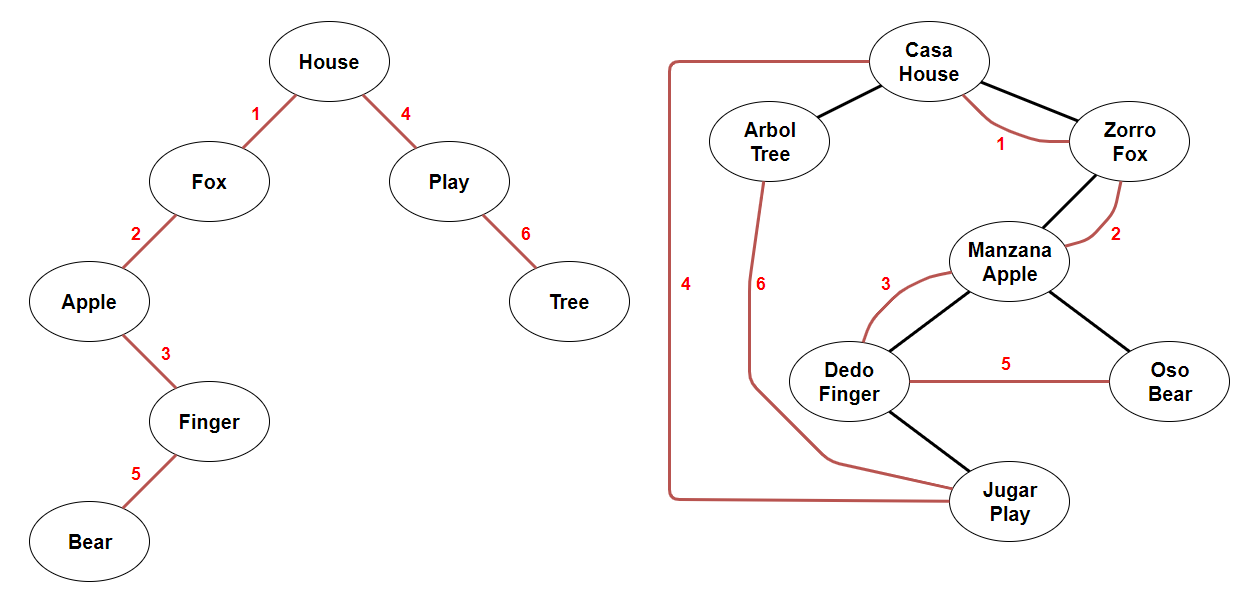
Hasta este punto los nodos solo cuentan con conexiones para los punteros del idioma español (padre, hijoIzq e hijoDer) y las palabras en español e inglés.

Posterior a este trabajo, se procede a realizar lo mismo para las palabras en inglés. Nuevamente se utilizan las palabras en el mismo orden que fueron ingresadas para armar el árbol en inglés esta vez de forma “imaginaria”, pero cada vez que se encuentre un lugar donde debería ingresarse un nodo, se hace una búsqueda dentro del árbol ya construido para encontrar el nodo correspondiente a la palabra, de esta forma se actualizan los punteros que corresponden a la palabra en inglés de este (father, leftChild y rightChild).



*Figura 6: Ejemplo árbol en inglés y sus conexiones en el árbol real.*

El árbol que se tiene finalmente para el caso de ejemplo es el siguiente, donde los números representan la iteración en la cual se generó esa conexión.



*Figura 7: Árbol en inglés y árbol real final.*

Luego de este proceso ya se terminó de construir el árbol completamente.

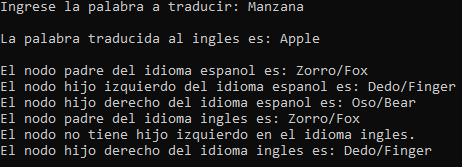
Posteriormente se procede a pedirle al usuario que ingrese la opción que quiera:

* Traducir desde el español al inglés
* Traducir desde el inglés al español
* Salir del programa.

Luego de cada traducción se vuelve a preguntar esto mismo con el fin de que pueda traducir varias palabras por ejecución y terminar el programa cuando lo desee.

Para cada traducción que solicita el usuario, se hace una búsqueda dentro del árbol la cual se ve afectada por el orden del árbol y también del orden de la traducción, ya sea español a inglés o viceversa.

La búsqueda siempre se realiza comparando la palabra ingresada por el usuario con un nodo, inicialmente el nodo raíz, si este nodo no contiene la palabra que se desea traducir, se compara para escoger si se debe procesar el subárbol izquierdo o el subárbol derecho hasta encontrar la palabra buscada, esto se hace mediante recursión lineal hasta que es encontrado el nodo que se busca. Una vez esto suceda, imprime por pantalla la traducción de la palabra como también la información relacionada a quien es el padre del nodo, el hijo izquierdo y el hijo derecho para ambos idiomas.



*Figura 8: Salidas del programa.*

También se debe mencionar que si el usuario ingresa una palabra que no se encuentre dentro del árbol se imprime un mensaje correspondiente y se vuelve a preguntar para que ingrese una opción para la traducción.

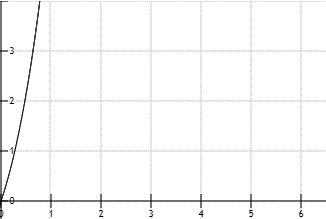
**2.3 Análisis de los resultados**

Finalmente, se logró completar el programa. En esta sección se procede a calcular el tiempo de ejecución promedio como también el orden de complejidad del programa.

1. Las funciones contarPalabras y guardarPalabras ambas cuentan con un   
   T(n) = n + c, donde n corresponde a la cantidad de líneas que contiene el archivo.
2. La función ingresarNodo es de tipo recursivo, en el caso base donde ingresa el nodo de inmediato cuenta con T(n) = c, para el peor caso su T(n) = n + c, donde n representa la cantidad de palabras que se encuentran en el archivo.
3. pointNode también es de tipo recursivo, donde T(n) = c en el mejor caso donde tenga que apuntar un nodo a otro de inmediato. Por otro lado, puede llegar a   
   T(n) = n + c, si tiene que recorrer completamente el árbol.
4. insertNode también es de tipo recursivo y hace uso de la función pointNode. En el mejor caso de ambas se puede llegar a T(n) = c, pero el peor caso es que llegue a T(n) = n2 + nc + c, debido a que primero busca donde colocar la palabra en inglés y luego en pointNode se busca el nodo ya existente para cambiar la dirección de los punteros.
5. imprimir es una función simple que solo imprime los datos del nodo, su   
   T(n) = c.
6. traducir es función recursiva y hace uso de imprimir. Si encuentra el nodo en la raíz su T(n) = c. Para el peor caso cuenta con un T(n) = n + c, igual que las anteriores ya que depende de la cantidad de nodos del árbol, lo que es la cantidad de palabras del archivo.
7. Dentro de la función main igualmente se encuentra un T(n) = 2n + X + c. Donde X corresponde a la sección en la que se pregunta al usuario que ingrese la opción que desea realizar. Esta se repite la cantidad de veces que el usuario quiera, por lo tanto, no es directamente estimable en comparación a n.

Finalmente, dentro de la ejecución del programa se puede observar un   
T(n) = n3 + 2n2 + 2n + Xn + C, lo que lleva al programa a tener un orden de complejidad O(n3). El n3 se atribuye a la sección correspondiente a asignar los punteros del árbol ya creado para ingresar las conexiones del supuesto árbol en inglés, la función pointNode corresponde a un n, insertNode usa pointNode resultando en un n2 y este proceso se repite n veces debido a la cantidad de palabras en el archivo de entrada, resultando en el n3

El resultado anterior es observable en el siguiente gráfico.



*Figura 9: Gráfico T(n).*

Donde claramente se observa que el tiempo de ejecución puede aumentar rápidamente a mayor cantidad de palabras ingresadas dentro del archivo de texto.

# Conclusiones

Finalmente, se pudo completar el programa. Tiene un par de situaciones que podrían ser mejorables, como el hecho de que las traducciones son sensitivas a mayúsculas y minúsculas, por lo tanto, las palabras ingresadas por el usuario para poder ser reconocidas deben ser ingresadas de igual forma como se encuentran dentro del archivo de texto. Para solucionar esto se podría implementar una forma de bajar todas las letras a minúsculas al momento de hacer la comparación.

Respecto a la estructura de datos árbol, se encontró bastante entretenida de usar e implementar, las búsquedas al ser un árbol binario también fue fácil reconocer la manera de hacerlo mediante recursión y así mismo el ingreso de nuevos nodos al árbol. Probablemente si hubiese sido un árbol AVL no lo hubiese encontrado tan entretenido ya que tendrían que implementarse las rotaciones y una serie de restricciones distintas a que solo trabajar con un árbol binario ordenado.

Para concluir, creo que los laboratorios fueron una buena forma de implementar la materia de la cátedra, a pesar de que no pude implementar correctamente el laboratorio 2 y 3 comprendo bien el manejo de las estructuras correspondientes a esas entregas. Además ya manejo de mucho mejor forma los punteros y funciones como malloc, cosa que no sucedía al inicio del semestre.

**Referencias**

EcuRed (2011). *Arreglos informática*. <https://www.ecured.cu/Arreglos_(Inform%C3%A1tica)>

Wikipedia (2018). *C (lenguaje de programación)*. <https://es.wikipedia.org/wiki/C_(lenguaje_de_programaci%C3%B3n)>

Wikipedia (2018). *Lista enlazada*. <https://es.wikipedia.org/wiki/Lista_enlazada>

Wikipedia (2018). *Matriz (matemáticas).* <https://es.wikipedia.org/wiki/Matriz_(matem%C3%A1ticas)>

Wikipedia (2018). *Paradigmas de programación.* <https://es.wikipedia.org/wiki/Paradigma_de_programaci%C3%B3n>

Wikipedia (2018). *Puntero (informática).* <https://es.wikipedia.org/wiki/Puntero_(inform%C3%A1tica)>

Wikipedia (2018). *Árbol (informática).* <https://es.wikipedia.org/wiki/%C3%81rbol_(inform%C3%A1tica)>

# Manual de usuario

**4.1 Introducción**

Este programa está diseñado para ser utilizado como traductor inglés-español, mediante un archivo de texto plano que se encuentra en la misma carpeta del programa en el cual se encuentran las palabras y sus respectivas traducciones al otro idioma.

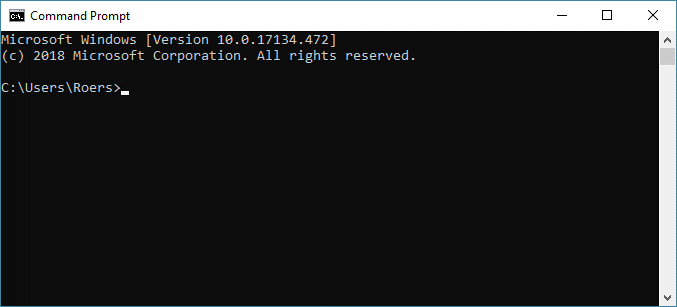
**4.2 Cómo compilar y ejecutar el programa**

Primero que todo, debe tener en una carpeta 2 archivos importantes:

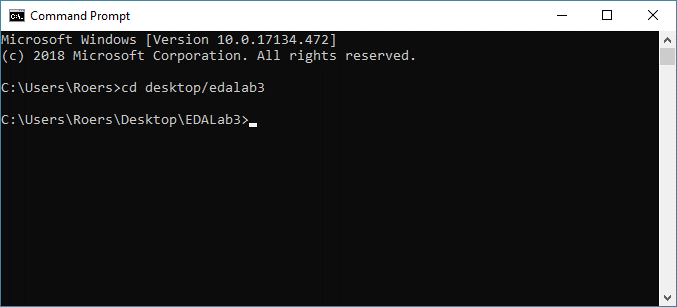
* El archivo del programa .c
* El archivo del programa .h
* El archivo de texto traductor.in

**Para compilación en Windows:**

Presiones la tecla Windows de su teclado y escriba la palabra cmd y luego presione la tecla Enter, al hacer esto debe encontrarse con la siguiente ventana:



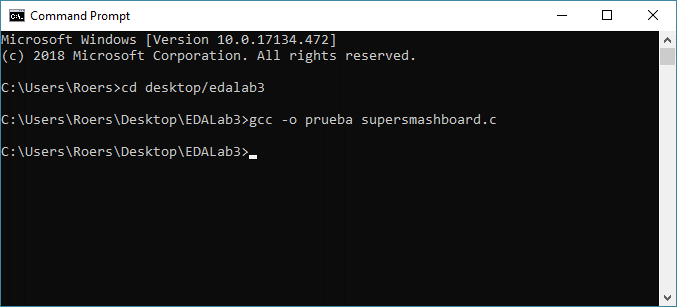
Una vez aquí, haciendo uso del comando cd (change directory) e ingresar la ruta de la carpeta donde se encuentra el programa:



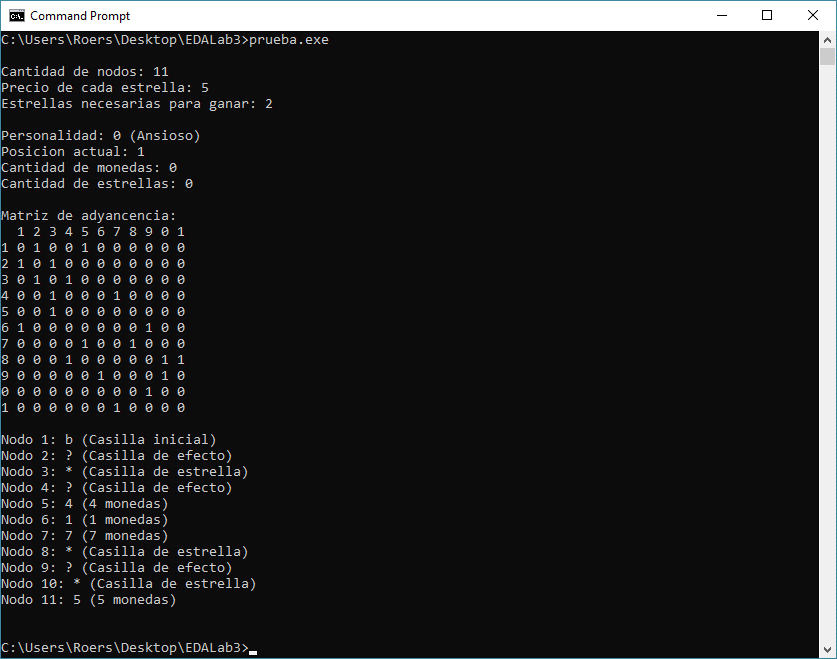
Ya ubicado en la carpeta correspondiente, antes de querer ejecutar el programa debe compilarlo, para esto debe ingresar la siguiente línea:

gcc -o “nombreSalida” “nombrePrograma.c”

donde nombreSalida representa el nombre que usted desee darle al programa compilado, mientras nombrePrograma.c al programa que va a compilar, incluyendo el .c:



Cuando ya quiera ejecutar el programa, debe ingresar el nombre que le dio de salida seguido de .exe



**Para compilación en Linux/MacOS:**

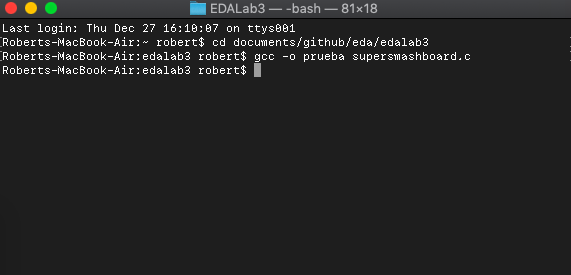
Primero necesita abrir la terminal, una vez se encuentre en esta, haciendo uso del comando cd (change directory) debe dirigirse a la dirección en donde tiene la carpeta con el programa y los archivos:



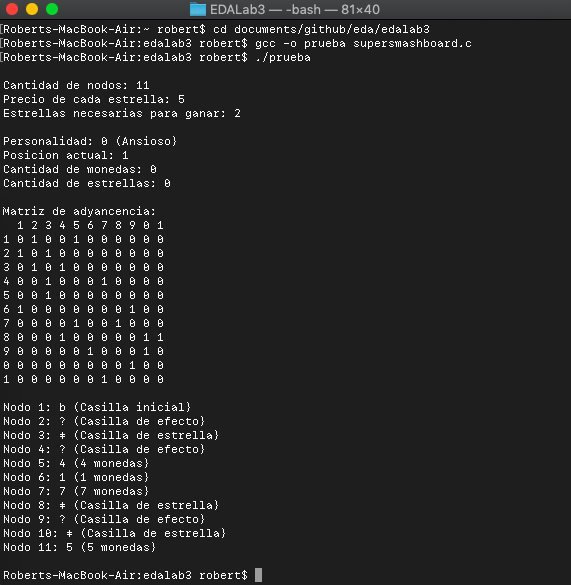
Ya ubicado en la carpeta correspondiente, antes de querer ejecutar el programa debe compilarlo, para esto debe ingresar la siguiente línea:

gcc -o “nombreSalida” “nombrePrograma.c”

donde nombreSalida representa el nombre que usted desee darle al programa compilado, mientras nombrePrograma.c al programa que va a compilar, incluyendo el .c:



Cuando ya quiera ejecutar el programa, debe ingresar ./ seguido inmediatamente del nombre que le asignó al programa:



**4.3 Funcionalidades del programa**

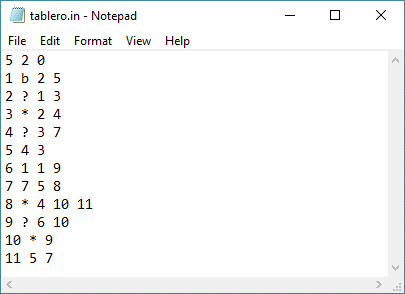
Actualmente, este programa tiene la capacidad de:

1. Leer el archivo “tablero.in”
2. Guardar todos los datos del archivo (Matriz de adyacencia, cantidad de nodos, etc.)
3. Imprimir por pantalla todos los datos obtenido

Los demás requisitos no están implementados en esta versión.

**4.4 Formas de uso y errores conocidos**

Se adjunta la siguiente imagen del archivo de texto que es necesario, este es sólo un ejemplo de formato:



Este archivo de texto debe encontrarse en la misma carpeta que el programa cuando se vaya a ejecutar.

El archivo que usted va a utilizar debe ser igual en formato al anteriormente mostrado, donde los números 5 y 2 indican el precio de una estrella y la cantidad de estrellas necesarias para ganar respectivamente, mientras el 0 representa la personalidad del bot, la cual debe ser 0 ó 1.

Las siguientes líneas corresponde a todos los nodos del grafo/tablero, donde el primer número es el número del nodo/casilla, el segundo símbolo corresponde a el tipo de casilla que es el nodo, mientras todos los números siguientes corresponden a los nodos a los cuales se puede llegar desde el nodo actual.

Para los tipos de casillas debe recordar que: un número mayor o igual a 1 corresponde a la cantidad de monedas que entrega esa casilla, una b indica la posición inicial del tablero, un signo ? indica una casilla de efecto y un \* significa una casilla donde se puede comprar una estrella.

Una vez con el archivo en orden puede compilar y ejecutar el programa como se explicaba anteriormente.

Entre los errores conocidos se tiene que, si usted ingresa un salto de línea extra al final del archivo, provocará que dentro del programa se genere una casilla extra replicando la última, lo cual lleva a una mala representación del tablero.

También debe recordar que si desea colocar casillas en las que se entreguen dinero, el número debe ser mayor o igual a 1.

Además. preocúpese de no ingresar demasiados nodos conectados a un solo nodo, el programa solamente lee 100 caracteres por línea.