Universidad Simón Bolívar
Departamento de Computación y
Tecnología de la Información
CI-3825: Sistemas de Operación I

Prof. Fernando Torre Mora

Tarea Enero – Marzo 2021

El Efecto ELIZA

1 Introducción

En 1966, Joseph Weizenbaum creó un programa de reconocimiento de texto para la IBM 7094 el cual denominó ELIZA. Este programa estimulaba al usuario a participar fingiendo ser un psicoterapeuta. El programa tuvo éxito sin precedentes ya que los usuarios lo trataban como una persona real, a pesar de tener muy pocas respuestas programadas. A esta percepción se le llamó "El Efecto ELIZA".

Tristemente, las tarjetas perforadas originales se han perdido, por lo que se le pide que realice una recreación del programa en C que pueda correr en las computadoras del LDC. El programa debe quedar en un archivo único ejecutable, sin requerir el uso de archivos externos (es decir, todos los mensajes deben estar cableados en el código). La entrada y la salida debe escribirse en ASCII (sin acentos).

2 Requerimientos del programa

Escriba un programa que tome la entrada de STDIN e imprima por STDOUT. Todos los mensajes del programa deben ser definidos por su equipo y deben ser conversacionales. El programa debe imprimir, al iniciar, un saludo indicando el inicio de la sesión de terapia.

Cada vez que reciba una entrada, el programa debe buscar si alguna palabra coincide contra una de las **palabras claves**. Las palabras claves se presentan en este documento ordenadas desde más a menos importantes (es decir, si la entrada contiene palabras claves de más de una categoría, la respuesta será la de la categoría más importante)

Adicionalmente, debe **recopilar información** sobre el usuario, incluyendo lugar de origen y los nombres de su familia y amigos, para propósitos del expediente. Si ninguna de las palabras clave es reconocida, el programa debe preguntar por algún dato faltante.

Finalmente, si todos los datos han sido recopilados, el programa debe darle al usuario uno de varios **mensajes conversacionales**.

2.1 Palabras clave

Usando ingeniería inversa, se han detectado las siguientes palabras claves que reconoce ELIZA:

1. El programa debe imprimir un **mensaje de despedida** y **terminar la ejecución** al recibir una entrada que contenga cualquiera de las siguientes palabras o frases:
   - Adiós
   - Chao
   - Me voy
   - Hora de irse
   - Hora de irme
   - Debo irme
   - Tengo que irme
   - Acabó el tiempo
   - Hasta luego

2. El programa debe imprimir un **mensaje de pésame** al recibir una entrada que contenga cualquiera de las siguientes palabras o frases:
   - Murió
   - Muere
   - Pasó a mejor vida
   - Estiró la pata
   - Falleció
   - Fallece
   - Perdió la vida
   - Costó la vida
   - Cobró la vida

3. El programa debe imprimir un **mensaje de freudiano** (invitando al usuario a hablar sobre su familia) al recibir una entrada que contenga cualquiera de las siguientes palabras o frases:
   - Mamá
   - Madre
   - Papá
   - Padre
   - Hermano
   - Hermana
   - Tío
   - Tía
   - Primo
   - Prima
   - Abuelo
   - Abuela
   - Es mi *
   - Se trata de mi *

Donde * representa la palabra que indica la relación cuando no es ninguna de las anteriores (para efectos de la recopilación de información). Cabe destacar que identificar una de estas relaciones también requiere almacenar la existencia de esta persona.

4. El programa debe imprimir un **mensaje de exploración** (animando al usuario a entrar en detalle) al recibir una entrada que contenga cualquiera de las siguientes palabras o frases:
   - Creo
   - Cree
   - Pienso
   - Piensa
   - Recuerdo
   - Recuerda
   - Digo
   - Dice
   - Hace
   - Hago
   - Igual
   - Iguales
   - Parece
   - Parecen
   - Parecidos
   - Lo mismo
5. El programa debe imprimir un **mensaje de espacio seguro** (animando al usuario a expresarse) al recibir una entrada que contenga cualquiera de las siguientes palabras o frases:
   - Puede
   - Perdón
   - Disculpa
   - Debo
   - Debería
   - Lo Siento
   - Lamento
   - Maldito
   - Desgraciado

6. El programa debe imprimir un **mensaje de exploración de sentimientos** (preguntándole al usuario qué siente al respecto) al recibir una entrada que contenga cualquiera de las siguientes palabras o frases:
   - Pasó
   - Ocurre
   - Ocurrió
   - Cayó
   - Lanzó
   - Aventó
   - Hizo
   - Hice
   - Logró
   - Logré
   - Llegó
   - Llegué
   - Soy
   - Estoy
   - Era
   - Fue
   - Perdío
   - Perdí
   - Ganó
   - Gané
   - Abuso
   - Abusa
   - Exagera
   - Trauma
   - Karma
   - Dolor
   - Duele

7. El programa debe imprimir un **mensaje de profundización de sentimientos** (preguntándole al usuario por qué se siente así) al recibir una entrada que contenga cualquiera de las siguientes palabras o frases:
    - Me siento
    - Siento que
    - Estoy sintiendo
    - Molesto
    - Molesta
    - Molestia
    - Ira
    - Cólera
    - Coraje
    - Aguanto
    - Aguanta
    - Aguantar
    - Contento
    - Solo
    - Sola
    - Feliz
    - Felicidad
    - Alegre
    - Alegría
    - Gusta
    - Triste
    - Tristeza
    - Disgusto
    - Desagrado
    - Agrado
    - Agrada
    - Bueno
    - Buena
    - Divertido
    - Divertida
    - Náuseas
    - Asco
    - Pena
8. El programa debe imprimir un **mensaje de positivismo** al recibir una entrada que contenga cualquiera de las siguientes palabras o frases:
    - No
    - Contrario
    - Malo
    - Mala
    - Pésimo
    - Pésima
    - Terrible
    - Horrible
    - Aguante
    - Aguanto
    - Aguanta
    - Resisto
    - Resiste
    - Resistirá
    - Chingada
9. El programa debe imprimir un **mensaje de deflexión** (tratando de regresar la conversación al tema del usuario) al recibir una entrada que contenga cualquiera de las siguientes palabras o frases:
    - Tu
    - Usted
    - Vos
    - Eres
    - Sos
    - Sois
    - Estás
    - Él es
    - Ella es
    - Qué es
    - Quién
    - Cómo
    - Cuándo
    - Dónde
    - Por qué
    - Cuál
    - Cuáles
10. El programa debe imprimir un **mensaje de ayuda** (indicando cómo usar el programa) al recibir una entrada que contenga cualquiera de las siguientes palabras o frases:
    - No sé
    - Cómo funciona
    - Qué debo hacer
    - Qué hacer
    - Qué hago
11. El programa debe imprimir un **mensaje de exploración** (animando al usuario a entrar en detalle, que puede ser el mismo mensaje №4) al recibir una entrada que contenga cualquiera de las siguientes palabras o frases:
    - Para
    - Hacia
    - Hasta
    - Desde
    - Por
    - Mediante
    - Según
    - Mientras

12. El programa debe imprimir un **mensaje de delimitación** (pidiéndole al usuario que especifique) al recibir una entrada que contenga cualquiera de las siguientes palabras o frases:
    - Nosotros
    - Nos
    - Ellos
    - Todos
    - Todas
    - Muchos
    - Muchas
    - Abrumador
    - Sobrecogedor
    - Lleva
    - Tan

2.2 Recopliación de Información

El programa debe tratar de almacenar información sobre varias personas, incluyendo el usuario, específicamente

- Nombre
- Lugar de origen
- Relación con el usuario

Para reconocer nombres, se toma la palabra que sigue a alguna de las siguientes palabras luego de preguntar. Se presume que los nombres no pueden contener espacios:

- Es
- Llama
- Llamo
- Dicen
- Digo
- Decimos

Si no se encuentran ninguna de estas palabras, se presume que la primera palabra, luego de preguntar, es el nombre.

Para reconocer lugares, se toma la palabra que sigue a alguna de las siguientes luego de preguntar:

- De
- En

Nótese que el usuario podría responder con un gentilicio (como una nacionalidad). Estos se reconocen con las palabras:

- Es
- Es un
- Parece

Al encontrarse nuevamente con el nombre de una persona mencionada anteriormente, o una relación mencionada anteriormente, el programa debe imprimir la información sobre esa persona. Esto tiene prioridad por sobre todos los mensajes anteriores, ya que ayuda al usuario a sentirse escuchado. Si hay información faltante, la pregunta. Así, por ejemplo:

ELIZA: Hola, soy ELIZA, tu psicoterapeuta. ¿Hay algo que te esté perturbando?
Usuario: Se trata de mi jefe
ELIZA: ¿Cómo se llama?
Usuario: Carlos, por qué?
ELIZA: Gracias, continua.
Usuario: Es que es un desgraciado!
ELIZA: Adelante, estás en un espacio seguro.
Usuario: No quiero hablar de eso
ELIZA: No hay que ser tan negativos.
Usuario: Es que se supone que es mi jefe!

ELIZA: ¿Tu jefe Carlos? ¿De dónde es?
Usuario: Es Maracucho, tal vez sea por eso
ELIZA: Gracias, continua.
Usuario: Debería portarse como jefe!
ELIZA: ¿Tu jefe Carlos que es Maracucho?
Usuario: Sí! Cuál otro?
ELIZA: Tratemos de enfocarnos en ti.

Se presume que el usuario siempre tiene un padre y una madre, por lo que el programa siempre deberá pedirle al usuario que hable sobre su madre si no hay más nada de qué hablar.

2.3 Mensajes Conversacionales

El programa debe tener una serie de mensajes conversacionales de los cuales debe escoger uno al azar. Debe al menos tener mensajes para preguntar

- Los pensamientos actuales del usuario
- Qué ha hecho el usuario recientemente
- Psicoterapeutas anteriores
- Pasatiempos

Recuerde que estos mensajes sólo se deben mostrar si la información sobre todas las personas ya ha sido llenada.

3 Sugerencias

- Puede separar la entrada usando `strtok` de `strings.h`.
- Puede generar números aleatorios usando la función `rand` de `stdlib.h`, pero debe inicializar el generador usando `srand(time(NULL))`.
- Puede declarar todos los elementos de un arreglo con `{` y `}` (véase la sección 5.7 del libro de Kernighan & Ritchie para un ejemplo).
- Diseñe un `struct persona` que almacene información de las personas independientemente de si es el usuario, un familiar u otro. Almacene un arreglo dinámico pedido con malloc de cada una de estas tres categorías.
- Escriba solo 1 mensaje por cada una de las 12 categorías de palabras clave.
- Almacene una lista o arreglo de todos los nombres recibidos separado de todas las relaciones recibidas. Mantenga estas listas ordenadas y recórralas usando búsqueda binaria. Asegúrese de que, cuando crezca dinámicamente, ese crecimiento ocurra de forma correcta.
- Almacene las palabras claves en listas similares a la de los nombres.
- Subidivida las palabras claves de múltiples palabras en las palabras que la componen para poder distinguir entre ellas más fácilmente, y revise la siguiente palabra si hay alguna coincidencia. Puede almacenar estas en una estructura similar a un árbol. (De hecho, puede separar las palabras claves muy similares de la misma manera.)
- Si dos palabras clave producen el mismo mensaje, no importa cuál de las dos sea identificada.
- Use `union` para guardar el lugar y defina una función que imprima dependiendo de si almacena un lugar o un gentilicio.
- Intente recrear una sesión de terapia pre-ensayada al probar el programa.

- No intente usar el programa para terapia real.

4 Requerimientos de la entrega

Debe entregar un archivo `tar.gz` con

- Su programa principal en un `.c`
- Archivos complementarios en `.c` y `.h`
- Makefile para compilar su programa
- El resultado de al menos dos ejecuciones diferentes en archivos `.txt` mostrando cómo probaron el programa

El programa debe compilar sin errores al llamar al comando `make`. No incluya archivos `.o` ni archivos ejecutables en su entrega.

Debe documentar su código, incluyendo una breve definición al inicio de cada archivo, función y macro que indique qué es y para qué es cada parámetro; y para cualquier variable cuyo nombre no sea intuitivo.

El proyecto debe ser subido al Moodle de la materia en la asignación marcada como "Tarea" en la sección de "Semana 2" Sólo deberá efectuar una entrega por grupo.

5 Evaluación

El proyecto tiene una ponderación de 10 puntos. Se asignarán:

- 8 puntos por código
  - 2,5 puntos: Por cada uno de los 12 mensajes basados en palabras claves
    - 2 puntos por la forma de detección (0,16 c/u)
    - 0,5 puntos por la impresión de estos mensajes
  - 4,5 puntos: Por la información sobre las personas
    - 3 puntos: por la forma de almacenarla
    - 1 puntos por la forma de detectarla
    - 0,5 puntos por la forma de imprimirla
  - 1 punto por la generación de mensajes conversacionales
- 2 puntos por ejecución
  - 1 punto por priorizar correctamente los mensajes
  - 0,5 puntos por imprimir una respuesta adecuada para cada uno
  - 0,5 puntos por las ejecuciones de prueba incluidas

El programa debe correr sin errores. Se considerará funcionalidad adicional para puntos extra. Se penalizarán las entregas tardías. Se penalizarán fallas del estilo de programación de C en Unix (por ejemplo, la omisión del makefile)