Práctica 2 - LEX

David Infante Casas - 75165296P - 3ºC / C1

1.- Explicación del Problema

El programa analiza el texto de entrada para darle un mejor formato mediante la adición de mayúsculas, espacios, puntos, etcétera donde corresponda. También elimina errores como números o mayúsculas en mitad de palabras donde no corresponde.
Además, si se escribe una operación matemática simple entre números enteros (suma+, resta-,multiplicación* o división/) entre símbolos #, en la salida se mostrará el resultado.
Ejemplos:
Entrada: #25+25 # #50 -25#
Salida: 50 25

2.- Limitaciones del Programa

Estas son algunas limitaciones del programa:

El tamaño máximo del texto de entrada es de 9999 caracteres aunque puede ser aumentado el tamaño de la variable "cadena" en la sección de Declaraciones. Lo mismo sucede con el número de operaciones que se almacenan.

El programa no reconoce caracteres extraños como vocales con tildes, la interrogación: ¿, la

exclamación ¡ o la ñ.

Al hacer uso de los signos + - * / para hacer operaciones, el programa no los escribirá y los tomará como signos de operación.

El programa tiene dificultades para arreglar palabras en las que aparezcan muchos números y mayúsculas mezclados.

3.- Método de Compilación

Se debe extraer el fichero "plantilla.l" y un fichero de texto simple que contendrá el texto de prueba en una misma carpeta. Entonces, haciendo uso de la terminal, haremos:

- lex plantilla.l (si no se tiene lex, usar flex)

- gcc lex.yy.c –o prog –ll (gcc lex.yy.c –o prog –lfl en el caso de usar flex)
Se nos mostrarán 3 warnings por pantalla, pero no afectarán a la ejecución del programa.

- ./prog <Nombre_Fichero_Que_Contiene_La_Prueba>

4.- Explicación del código

Sección de Declaraciones:

Hacemos la inclusión de las librerías básicas para trabajar en C.

<u>Variables:</u>

- Cadena con el texto de salida: char cadena[9999] = "";
- Cadena con los resultados de las operaciones: int operaciones[9999];
- Cadena que copia del yytext de entrada solo los números para operar con ellos: char copia[99] = "";
- Contador de celdas utilizadas de cadena: int cont;
- Contador de celdas utilizadas de operaciones: int cont_o;
- Contador de celdas utilizadas de copia: int cont_c;
- Indice básico para iterar en bucles: int i;
- Variables que almacenan los datos para operar sobre ellos: int dato1, dato2, resultado;
- "Enum" que indicará el tipo de operación matemática que se requiere: int operacion;
- Función que muestra por pantalla el texto de salida: void escribir_texto(char *, int, int *);

Sección de Reglas:

Contiene las reglas que harán que el programa:
- Haga operaciones básicas entre números enteros.
- Ponga una mayúscula al acabar una frase con un signo de puntuación.
- Ponga un espacio después de un signo de puntuación si no está puesto.
- Quitar mayúsculas y números en mitad de palabras.
- Poner un punto para acabar una frase si no hay otro signo de puntuación.
- Capturar el resto del texto.

Sección de Procedimientos:

Incluye el main y la implementación de la función escribir_texto.

5.- Hardware utilizado

Hardware: Raspberri Pi 3, Model B
Sistema Operativo: Linux raspberrypi 4.9.24-v7+

6.- Observaciones

El programa solo colocará puntos cuando haya un salto de linea(\n) y una mayúscula, si hay una mayúscula en mitad de una línea, el programa la interpretará como fuera de posición y la eliminará.

He encontrado dificultades a la hora de aplicar distintas reglas sobre palabras con pocas letras, por ejemplo, eliminar una mayúscula, un número y poner un punto de fin de frase.

En la carpeta se incluye un fichero de prueba.