Práctica 2 – Flex++

1. Descripción del problema.

Un personaje cualquiera de un juego de rol cualquiera dispone de un inventario donde almacena los objetos, de diferentes tipos, que ha ido obteniendo a lo largo de su recorrido por la historia. A la hora de llegar a cierta misión, el juego debe mostrar al jugador qué objetos puede o no portar a dicha misión según el siguiente criterio:

-De entre las armas disponibles en el inventario, solo podrá elegir las que superen cierto umbral determinado por la misión.
-Siempre podrá llevar objetos de tipo support o ayuda, cuyo poder será siempre 0.0.
-Existen ciertos objetos con un poder fuera del alcance del personaje, las armas legendarias. Solo podrá portar una por misión, independientemente de la cantidad disponible.

El juego, es decir, el programa, mostrará al jugador cuántos objetos contiene su inventario, cuántos y cuáles están o no disponibles para la misión según el criterio anterior y cuantas armas legendarias posee. Para cada objeto, se indicará el nombre, el tipo, el poder y si está o no disponible para la misión. Al final, se muestra una lista con todos los datos anteriormente mencionados.

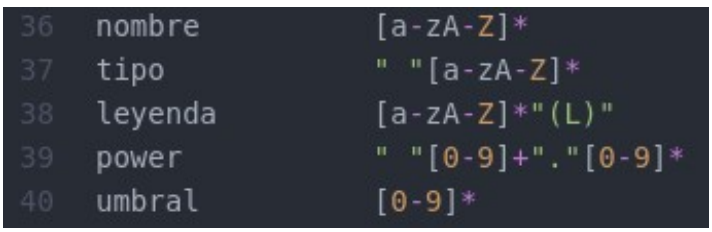
Lo que el programa lee es un fichero con extensión .txt que contiene en la primera línea un entero que representa el umbral de la misión y, en el resto de líneas, los objetos del inventario con el siguiente formato: Nombre Tipo Poder. Las armas legendarias tienen el siguiente: Nombre(L).

2. Secciones del fichero flex++.

Para ahorrar espacio, indicaré en este apartado lo que realizado en cada sección incluyendo únicamente capturas de los fragmentos mencionados, ya que el contenido al completo se encuentra en el propio fichero que contendrá la entrega, el cual será especificado más adelante.

2.1. Sección de declaraciones.

En esta sección he incluido, como se indica, el contenido del principio del código c++ que generé. Concretamente, inclusión de librerías y declaración de variables. Además, he creado un bloque de alias para las reglas con la siguiente estructura:



-Línea 36 → nombre: Palabras formadas por letras de la ‘a’ a la ‘z’ en mayúscula o minúscula.
Ejemplos: Arturo, Axxxxccvvv, Excalibur, Coche, etc.

-Línea 37 → tipo: Palabras formadas de la misma forma que los nombres, precedidas de un carácter de espacio (“ “).
Ejemplos: Lanza, Espada, Casa, sdknvla, etc. (Hay un espacio extra delante de cada una).

-Línea 38 → leyenda: Palabras formadas de la misma forma que los nombres, acabando siempre con los caracteres ‘(‘, ‘L’ y ‘)’ en ese orden.
Ejemplos: Arturo(L), E(L), Alfombra(L), etc.

-Línea 39 → power: Palabras formadas por una cadena de dígitos del 1 al 9, el carácter ‘.’ y, de nuevo, otra cadena de dígitos del 1 al 9. Representa un número decimal. Siempre precedidas por un espacio.
Ejemplos: 1989.487, 1.1, 8888.7777, etc.

-Línea 40 → umbral: Palabras formadas por una cadena de dígitos del 1 al 9. Representa un número entero.
Ejemplos: 16, 191651, 1, etc;

2.2. Sección de reglas.

Para cada alias existe una regla con el siguiente formato:

{alias} {Código c++}.

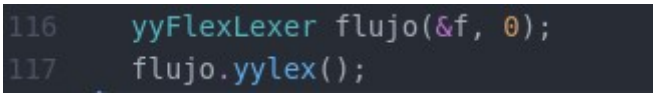
Debido a su extensión, no incluiré capturas de las mismas. Sin embargo, a continuación, indico las líneas donde se encuentran las reglas de cada alias en el fichero flex++:

-{umbral} → Línea 46.
-{nombre} → Línea 52.
-{tipo} → Línea 58
-{leyenda} → Línea 62
-{power} → Línea 70
-\n → Línea 100

He añadido una regla para el carácter de salto de línea ‘\n’. De esta forma puedo contar el número de líneas total del fichero.

2.3.Sección de procedimientos.

En esta sección se incluye el bloque de código c++ que se ejecutará. En este caso, únicamente he requerido de una función main con una estructura básica. En la siguiente imagen puede ver cómo he realizado la creación de la variable flujo necesaria para el programa, donde &f es una referencia al fichero de entrada:



3. Ejemplo de ejecución.

3.1 .Fichero de entrada Objetos.txt:

Objetos.txt

```
17
Brionac Lanza 19.5
Enfeeble Cetro 0.0
Belenus(L)
Talion Hacha 16.0
```

El umbral de la misión que los objetos deben superar es de 17. El resto de líneas representan objetos del inventario.

3.2. Ejemplo de compilación y ejecución.

```
arturo@arturo-VirtualBox:~/Escritorio/MC$ flex++ Inventario.l
arturo@arturo-VirtualBox:~/Escritorio/MC$ g++ lex.yy.cc -o Inventario
arturo@arturo-VirtualBox:~/Escritorio/MC$ ./Inventario Objetos.txt
Umbral de poder de la misión: 17

Inventario:

Brionac - Poder-> 19.5
Este objeto es de tipo Lanza y supera el umbral de poder para ser utilizada

Enfeeble - Poder-> 0
Este objeto es de apoyo, de tipo Cetro, siempre puede utilizarse

Belenus(L)
Se trata de un arma de categoría legendaria, solo podrás llevar una a la vez

Talion - Poder-> 16
Este objeto es un arma de tipo Hacha, pero no supera el umbral de poder de la misión


Número total de objetos en el inventario: 4
Objetos no permitidos para la misión final: 1
Total de armas que superan el umbral: 1
Total de objetos de ayuda o support: 1
Total de Armas legendarias (solo se puede portar una): 1
Total de objetos aptos para la misión final: 3
arturo@arturo-VirtualBox:~/Escritorio/MC$
```

3.3. Compilación y ejecución con/sin makefile.

-Para ejecutar tras el makefile, escribir “./Inventario Objetos.txt” en terminal.

-Para compilar y ejecutar sin makefile seguir los siguientes pasos:

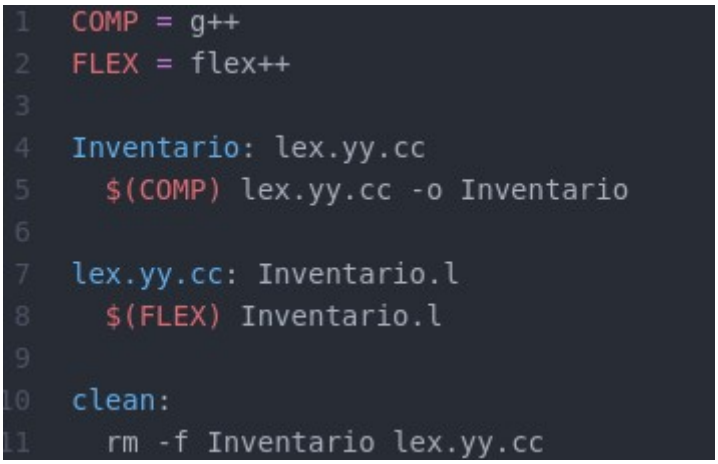
-Escribir “flex++ Inventario.l” en terminal.
-Escribir “g++ lex.yy.cc -o Inventario” en terminal.
-Escribir “./Inventario Objetos.txt” en terminal.

4. Ficheros creados en la práctica.

En la siguiente lista se indican los ficheros que irán incluidos en la entrega:

-Objetos.txt → Fichero con extensión txt con el contenido de un inventario.
-Inventario.l → Fichero con el código de la resolución del problema con flex++.
-Makefile → Fichero makefile para compilar y crear los siguientes archivos:
-Archivo lex.yy.cc.
-Archivo ejecutable denominado Inventario.
-Práctica 2 – flex++.pdf → Memoria de la práctica en formato pdf.

4.2. Estructura del Makefile.



```
COMP = g++
FLEX = flex++

Inventario: lex.yy.cc
  $(COMP) lex.yy.cc -o Inventario

lex.yy.cc: Inventario.l
  $(FLEX) Inventario.l

clean:
  rm -f Inventario lex.yy.cc
```

Nombre: Arturo Alonso Carbonero.

Grupo: 3ºA – A1