TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE LÁZARO CÁRDENAS

LENGUAJES Y AUTÓMATAS I

"TAREA 13: PROGRAMA QUE VALIDA NUESTRO LENGUAJE DISEÑADO"

CATEDRÁTICA:

Araceli Galván Montelongo

ALUMNOS:

Luis Fernando Rojas González

Owen Hassan López González

52T

CONTENIDO

I. CÓDIGO DISEÑADO

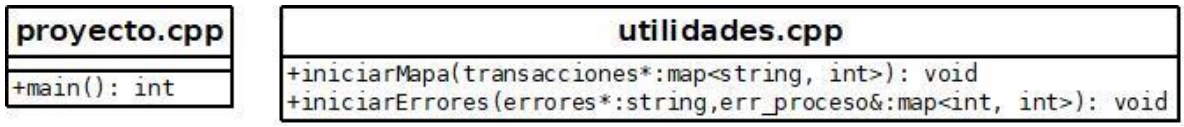
El programa se hizo en el lenguaje C++. Para la elaboración del programa se crearon dos clases:

proyecto.cpp

utilidades.cpp

La clase **proyecto** contiene la clase principal donde se desarrolla todo el procedimiento que permite al programa validar el lenguaje de programación.

La clase **utilidades** contiene dos métodos públicos y estáticos donde se llena la tabla de transiciones y la tabla de errores que serán utilizadas en la clase principal para poder validar el lenguaje



int main(): Única función en la clase principal. El copilador utiliza la función **main** contenida en la clase **proyecto.cpp** para iniciar el programa. Aquí se hace la lectura del archivo que se va a compilar, se analiza lo que se lee y utilizando diferentes capas de comprobaciones se determina que tipo de instrucción está siendo leída

El código diseñado realiza una serie de operaciones cuya importancia es absoluta para que se pueda conseguir el comportamiento deseado. Dicha serie de instrucciones son:

a) Crear el diccionario de palabras reservadas utilizadas (línea 21 y 31)
b) Crear el diccionario de símbolos utilizados (línea 20 y 36)
c) Crear la tabla de transiciones (línea 41)
d) Crear la tabla de errores (línea 42)
e) Leer el archivo que se desea compilar (línea 47)
f) Iniciar un stream para leer todos los caracteres contenidos en el archivo (línea 49)
g) Hacer un ciclo que recorrerá uno a uno todos los caracteres contenidos en el archivo (línea 59)
h) Cerrar el stream de lectura (línea 228)

i) Determinar el estado de compilación del código (línea 230 y 321)

En el inciso g) es donde se realizan todas las operaciones de concatenación, identificación y análisis de los caracteres que conformarán símbolos, números, cadenas de caracteres y palabras reservadas del mismo lenguaje de programación

El algoritmo para realizar lo descrito en el párrafo anterior se compone de diferentes capas de análisis contextual. De la forma mas general se presentan tres casos posibles diferentes

a) ¿Estamos en el estado de aceptación? (línea 61)
b) ¿Estamos leyendo un conjunto de números? (línea 65)
c) ¿Estamos leyendo otra cosa que no son números? (línea 102)

Para el inciso a) basta con terminar el ciclo y continuar con el código

Para el inciso b) se plantean dos posibilidades

a) ¿Estamos leyendo más números? (línea 67)
b) ¿Estamos leyendo otra cosa que no son números? (línea 74)

Para el inciso c) se plantean cuatro posibilidades

a) ¿Empezamos a leer números? (línea 103)
b) ¿Lo que leímos coincide con una palabra reservada? (línea 114)
c) ¿Lo que leímos coincide con un símbolo? (línea 135)
d) Análisis detallado del carácter (línea 199)

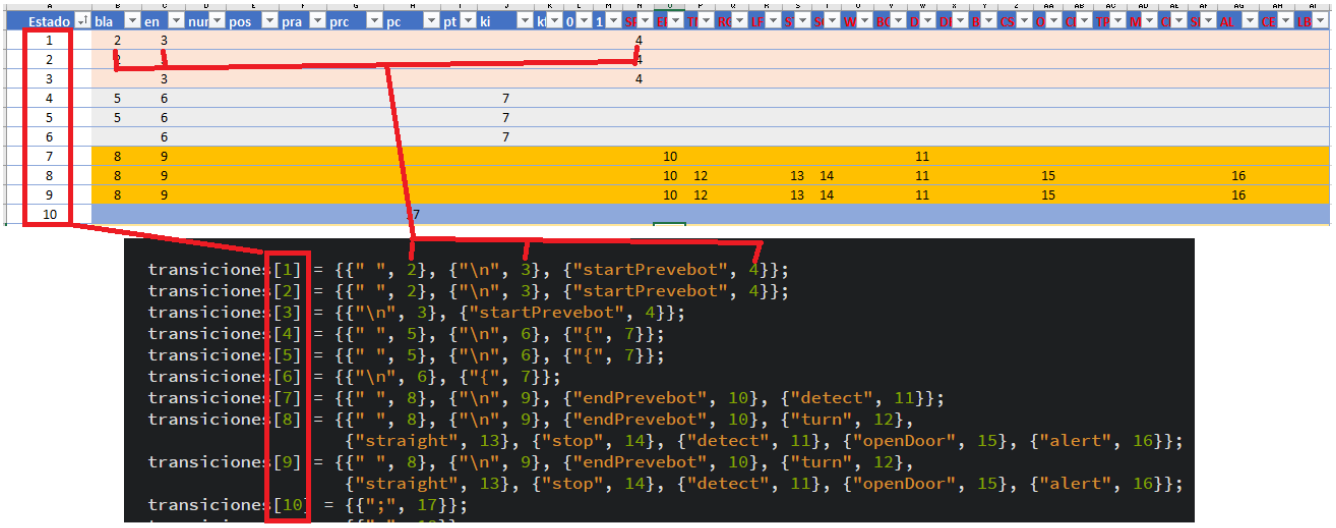
En el ultimo caso analizado (línea 199) se puede encontrar un comportamiento complejo de los caracteres porque hay múltiples posibilidades que se pueden presentar dependiendo del contexto. Este comportamiento complejo se puede optimizar con un análisis contextual más profundo. Pero para la naturaleza del problema resulta innecesario. En caso de seguir desarrollando el lenguaje de programación será absolutamente necesario.

II. RECURSIVIDADES DISEÑADAS

void iniciarMapa(): Función de la clase **utilidades.cpp**. Recibe como parámetro la siguiente estructura de datos:

map<**string**, **int**> transacciones []

Esta función es utilizada para llenar una estructura de datos que simula la tabla de transiciones del lenguaje diseñado. A continuación se muestra una comparación de la tabla de transiciones con el código contenido en la función **iniciarMapa**()



La estructura de datos utilizada es un arreglo de mapas. Cada mapa tiene la configuración key=string y value=int. Eso significa que al acceder a una cadena de caracteres determinada se podrá acceder a un numero entero asociado a esta. Ese numero entero corresponde a un nuevo estado. Este comportamiento es bastante familiar porque se está describiendo el funcionamiento completo de una tabla de transiciones, utilizando el numero mencionado anteriormente se puede acceder a otro estado y dicho estado provocará la repetición del procedimiento descrito en este párrafo.

Como se puede intuir, si se quiere acceder a algún estado de la tabla de transiciones, basta con poner entre corchetes el numero de estado que se desea. Al hacer eso se tiene acceso a todo el contenido de dicho estado.

```
transiciones[1] = {{" ", 2}, {"\n", 3}, {"startPrevebot", 4}};
transiciones[2] = {{" ", 2}, {"\n", 3}, {"startPrevebot", 4}};
transiciones[3] = {{"\n", 3}, {"startPrevebot", 4}};
transiciones[6] =
transiciones[7] = {{" ", 8}, {"\n", 9}, {"endPrevebot", 10}, {"detect", 11}};
transiciones[8] = {{" ", 8}, {"\n", 9}, {"endPrevebot", 10}, {"turn", 12},
                   {"straight", 13}, {"stop", 14}, {"detect", 11}, {"openDoor", 15}, {"alert", 16}};
transiciones[9] = {{" ", 8}, {"\n", 9}, {"endPrevebot", 10}, {"turn", 12},
                   {"straight", 13}, {"stop", 14}, {"detect", 11}, {"openDoor", 15}, {"alert", 16}};
transiciones[10] = {{";", 17}};
```

Posiciones en el arreglo

Valores almacenados en una estructura tipo mapa

-> map<string,int> <-

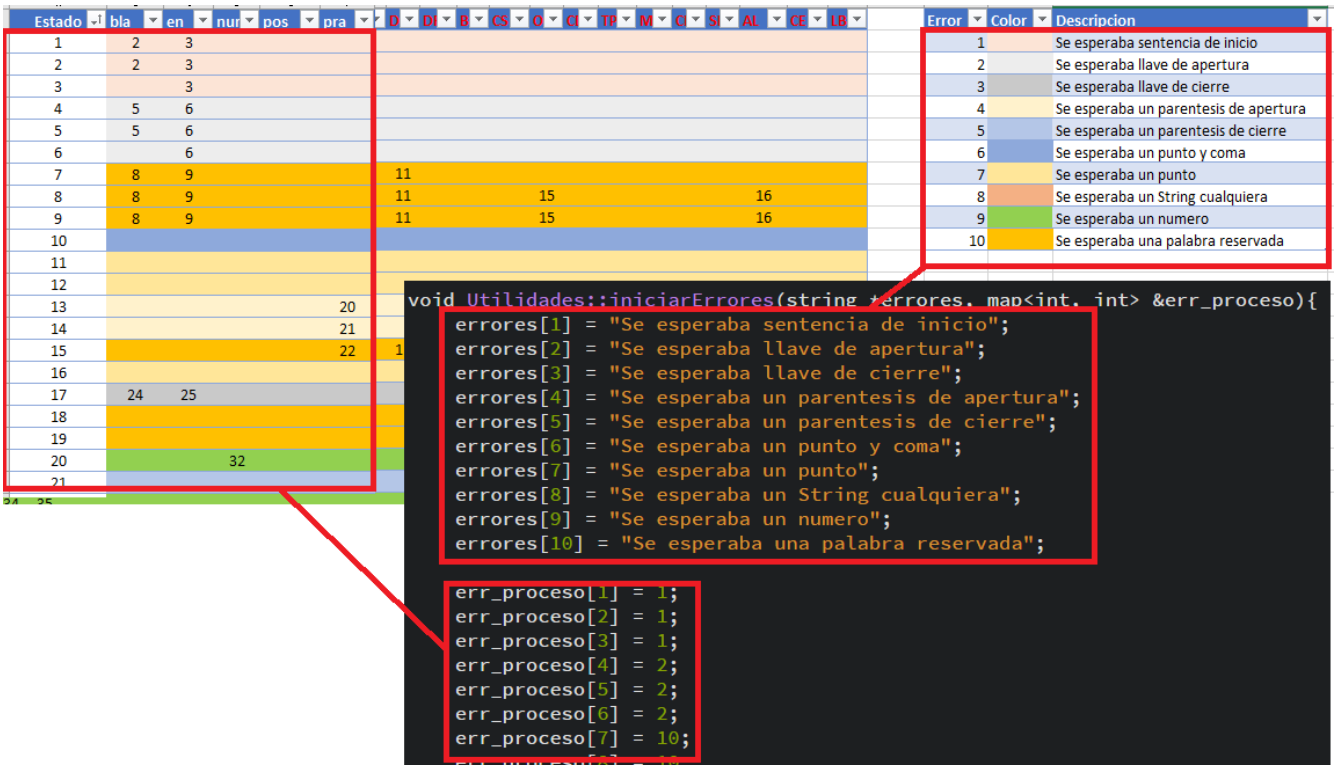
III. ERRORES DISEÑADOS

void iniciarErrores(): Función de la clase **utilidades.cpp**. Recibe como parámetro las siguientes estructuras de datos:

string errores[]

map<**int**, **int**> err_proceso

Esta función es utilizada para llenar dos estructuras de datos que juntas simulan la tabla de errores del lenguaje diseñado. A continuación, se muestra una comparación de la tabla de errores con las estructuras de datos utilizadas



Como se puede ver en la imagen anterior, la tabla de errores está contenida en el arreglo de strings y la tabla de errores está contenida en el mapa que tiene la configuración key=int y value=int. La key del mapa representa el numero de estado al que dicho error pertenece. El value del mapa corresponde al índice del arreglo de strings donde especifica que error le corresponde