INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Teoría Computacional

Reporte Práctica # 4

Profesora: Luz María Sánchez García

Grupo: 2CM1

Alumno: Mendoza Parra Sergio.

Boleta: 2015630300

MEXICO, D.F. a 28 de Abril del 2017

**Introducción:**

En este programa se verá el funcionamiento de una gramática libre de contexto. En este caso se dará una gramática ya establecida con símbolos terminales y no terminales y después de eso el usuario tendrá que escoger las opciones y al llegar a un símbolo terminal se tendrá que imprimir el árbol o las derivaciones de cómo se fue construyendo la cadena.

Se dará la explicación de la solución al igual que la implementación en código (java) y el funcionamiento en CMD y los resultados obtenidos.

**Planteamiento del Problema:**

El programa deberá leer desde un ***archivo*** o desde el ***teclado*** una GLC y derivado de él producir las cadenas que reconoce el lenguaje.

* Él programa debe leer:
* Los No Terminales a usar en la gramática. (S, A, B, etc.)
* Los Terminales (a, b, etc.)
* El símbolo inicial (S)
* Las reglas de producción (S->aA, A-> bB, B->b, etc.)
* Se puede preguntar al usuario cuantas cadenas quiere visualizar en pantalla producto de la derivación de sus producciones.

**Diseño y funcionamiento de la solución:**

Para la solución de este problema se tiene con la siguiente gramática libre de contexto que es:

S -> β | 0 | 1 | 1S1 | 0S0

En este caso lo que se hizo fue poner un arreglo en el cual se ingresaría todas las cadenas que se empiezan a realizar con forme a la expresión por ejemplo:

1. Primeros Casos (Símbolos Terminales): En este caso no hay necesidad de ponerlo en un arreglo ya que solo se imprime la opción del símbolo terminal, si se escogiera el lambda o el 0 o el 1 solo se imprimen esos valores y en ese momento terminaría el programa.
2. Segundo Caso(Símbolos no Terminales): En este caso como son símbolos que tienen recursividad se empiezan a poner en el arreglo teniendo lo siguiente:

|  |  |  |
| --- | --- | --- |
| 1 | S | 1 |

Posiciones: 0 1 2

De este modo si se empiezan a ingresar símbolos no terminales el arreglo se tiene que recorrer dependiendo de la posición de la letra “S”. Teniendo lo siguiente:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 0 | S | 0 | 1 |

Posiciones: 0 1 2 3 4

Para recorrer el arreglo e insertar la cadena del símbolo no terminal hay una función que nos permite insertar la cadena dentro un símbolo que nosotros le demos (como se muestra en la imagen).

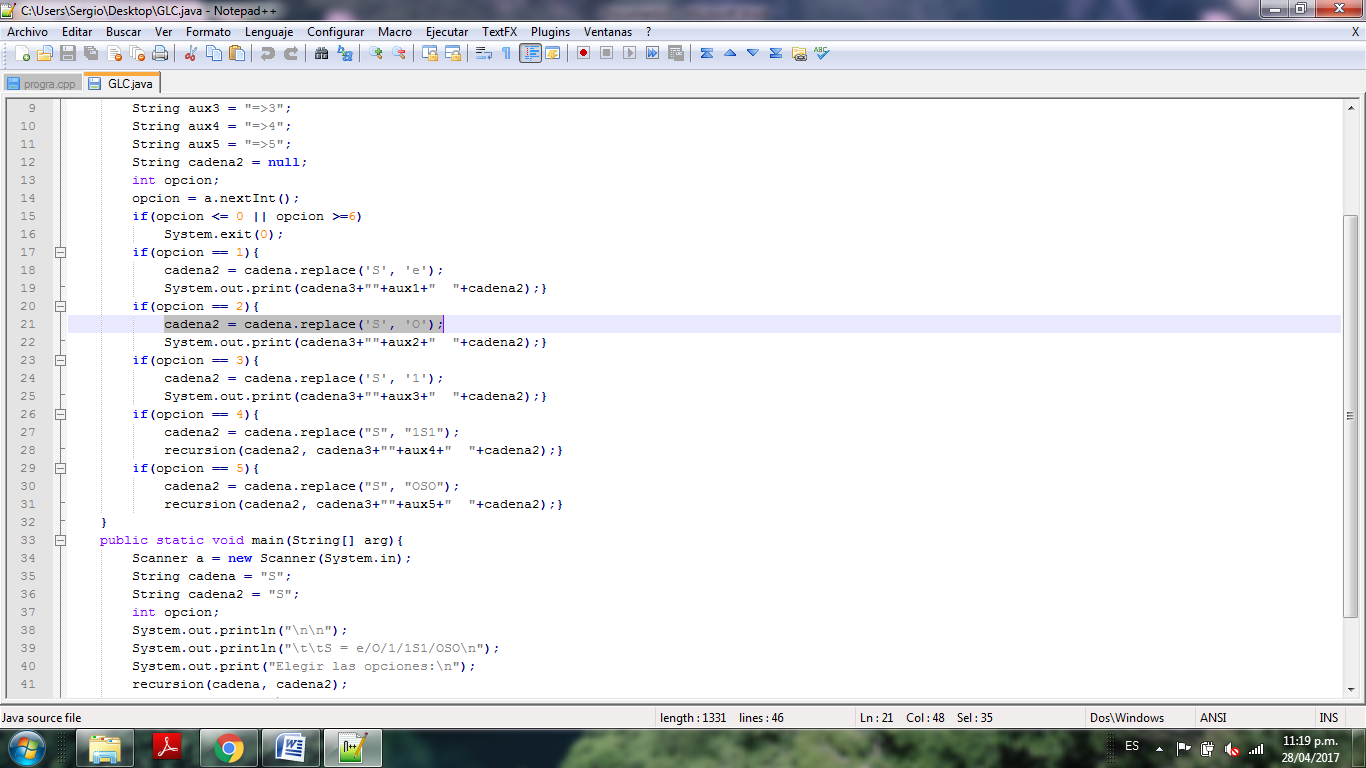
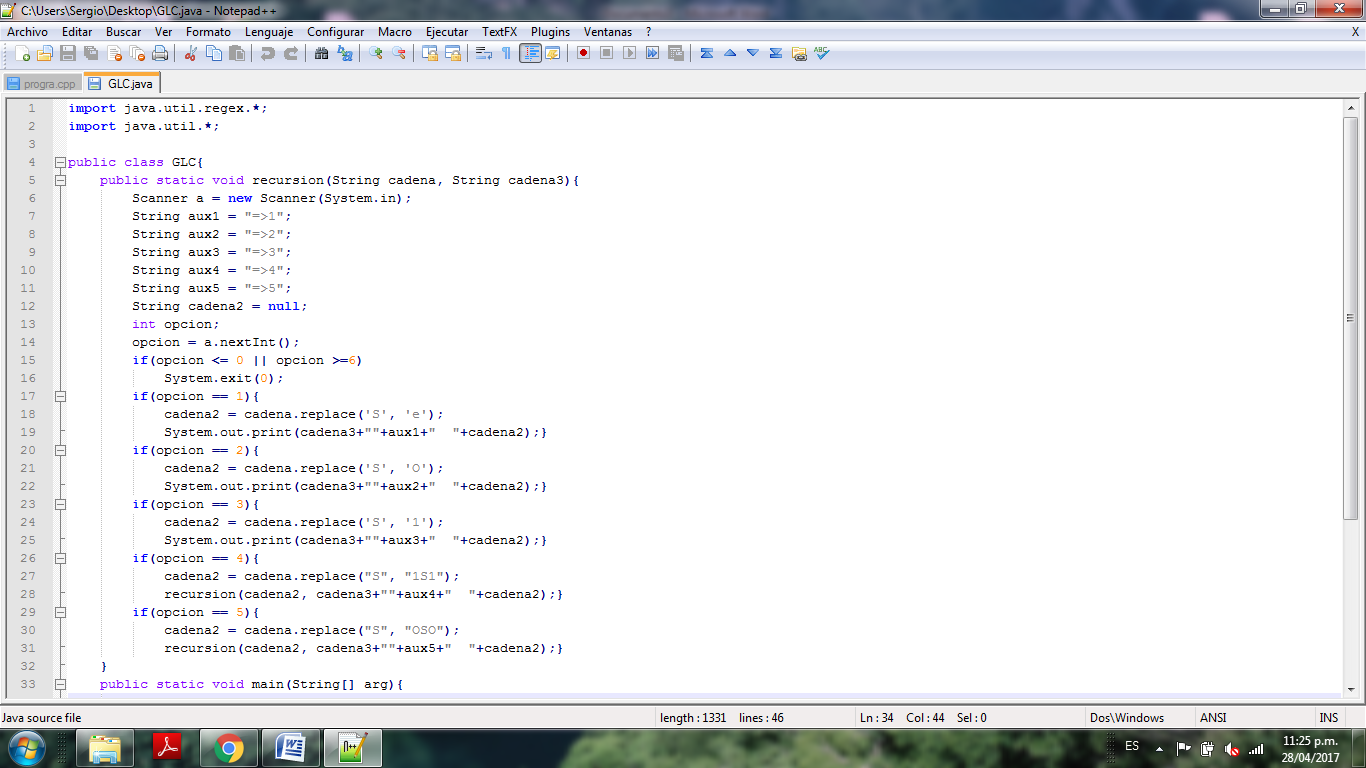


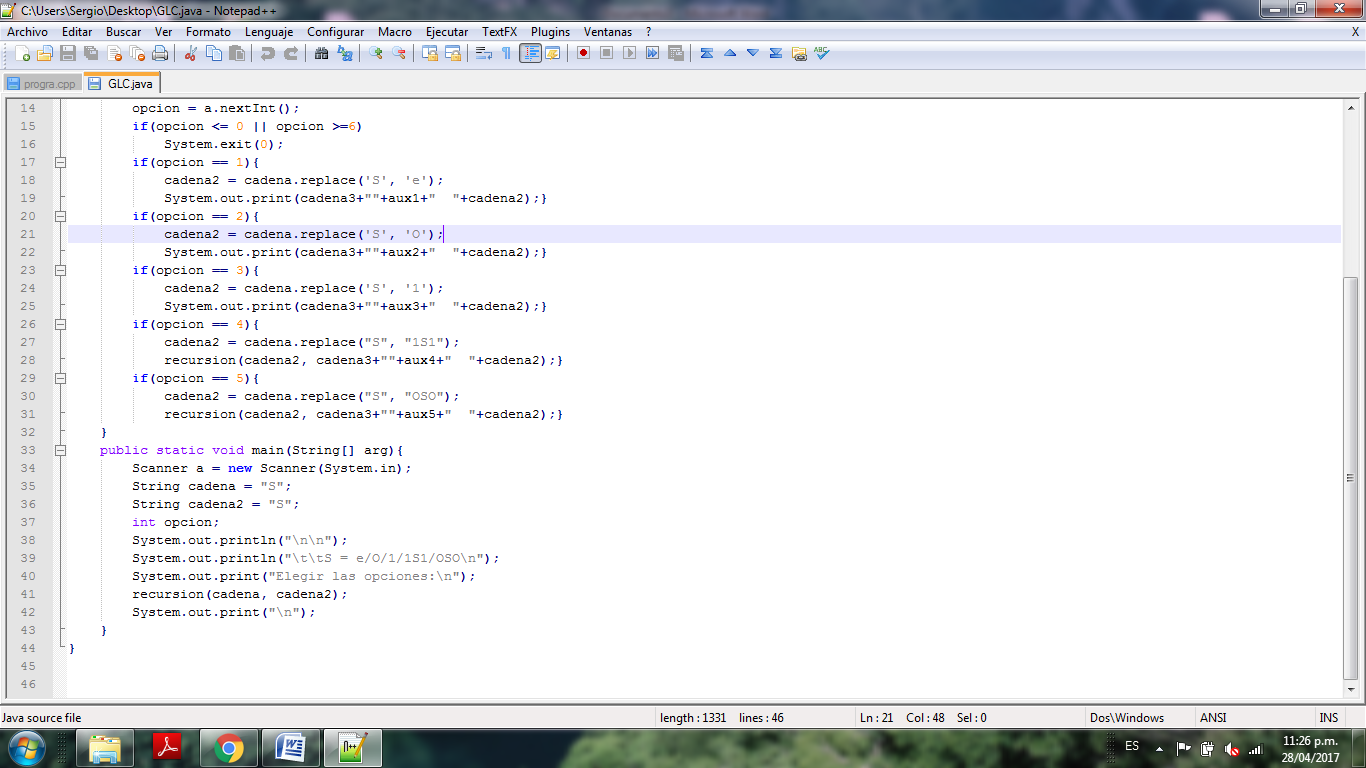
Imagen: Función replace.

De este modo el arreglo se recorre las posiciones necesarias para ingresar la cadena. De esta manera también se empieza a hacer las derivaciones de la cadena

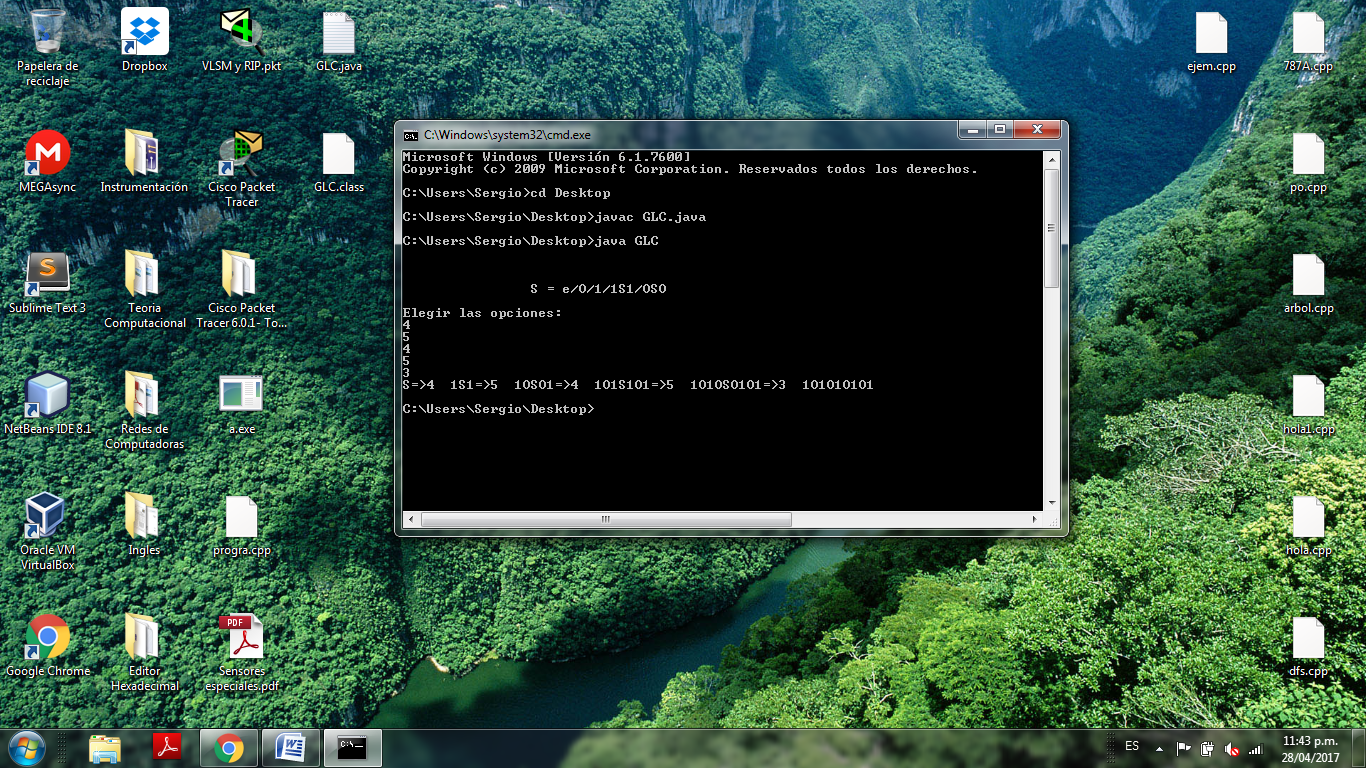
Y teniendo las derivaciones se tienen los resultados deseados (ver implementación en código).

**Solución en código:**





**Código compilado y mostrando la pantalla de salida.**

****

En este caso se eligieron algunas opciones de símbolos no terminales y al final se puso uno terminal dando como resultado la derivación de cómo se fue construyendo la cadena.

**Errores detectados**

Ninguno.

**Posibles mejoras**

En este código yo creo que no haría ninguna mejora porque si utilizo alguna estructura de datos la complejidad del algoritmo seria mayor y el tiempo de ejecución también por lo que yo dejaría así como está el programa.

**Conclusión**

Las GIC (Gramáticas Independientes del Contexto) o GLC (Gramáticas Libres del Contexto) son llamadas también “Gramática en la Forma de Backus-Naur (BNF)” (usado para describir lenguajes de programación). Las GIC se usan para inferir si ciertas cadenas están en el lenguaje expresado por la gramática. Hay 2 tipos de inferencia:

•Inferencia recursiva (cuerpo a cabeza/de cadenas a variables).

•Derivación (cabeza a cuerpo, expansión de producciones).

Por otro lado esta práctica se me hizo interesante ya que se pudo producir la cadena que se obtiene al empezar a escoger tanto símbolos terminales como no terminales y de este modo ver las cadenas validas que acepta la gramática.