INSTITUTO POLITÉNICO NACIONAL

Escuela Superior de Cómputo

Práctica 2 “BYACC y java grafico”.

Alumno: Rodríguez Martínez Heber Emanuel

27 DE OCTUBRE DE 2021

COMPILADORES 3CM15

Docente: Roberto Tecla Parra

Índice

Introducción

Berkeley Yacc (byacc) es un generador de analizador sintáctico Unix diseñado para ser compatible con Yacc. Fue escrito originalmente por Robert Corbett y lanzado en 1989. Debido a su licencia liberal y porque era más rápido que AT&T Yacc, rápidamente se convirtió en la versión más popular de Yacc. Tiene las ventajas de estar escrito en ANSI C89 y ser un software de dominio público.

Algunas de las ventajas de BYACC son:

I. *Solo se incluyen uno o dos archivos de clase. Si solo necesita un solo tipo o una clase de objeto, se genera un archivo de clase. Si necesita un tipo genérico simple, se genera una clase de datos simple para usted, creando otro archivo pequeño.*
II. *No se requieren bibliotecas de tiempo de ejecución adicionales. El código fuente generado es el analizador completo.*
III. *Puede analizar las gramáticas YACC existentes, lo que permite 'Javanizar' de una gran base instalada de código fuente de YACC (las 'acciones' deben estar en Java).*
IV. *BYACC / J se puede ejecutar desde Makefiles e IDE existentes.*
V. *BYACC / J está codificado en C, por lo que la generación de código Java es extremadamente rápida.*
VI. *El código de bytes resultante es pequeño, a partir de aproximadamente 11 kilobytes.*

Desarrollo

Para el desarrollo de esta práctica se usará la carpeta *grafibasi1*, de la carpeta *conmaquina* dentro de la carpitea principal *grafibasico*.

El directorio contiene los archivos necesarios para hacer uso de las funciones de dibujo que nos brinda canvas para Java, no obstante, se encuentra en un estado básico, que sólo nos permite realizar un dibujo en el origen, por lo que es necesario que se modifique para corregir ese detalle.

Para comenzar la especificación definiremos los tokens necesarios para identificar los comandos que serán ingresados.

```
%token NUMBER LINE CIRCULO RECTANGULO COLOR PRINT
%start list
```

A continuación, es importante definir cada una de las reglas gramaticales que integran el analizador, en el cual, se usarán las funciones de “maquina.java”, archivo donde se ubican las acciones para cada tipo de comando que se acepta por el analizador.

```
list :
     | list ';'
     | list inst ';'   {
                       | maq.code("print"); maq.code("STOP"); return 1 ;
     }
     ;
inst:  NUMBER  { ((Algo)$$.obj).inst=maq.code("constpush");
                 maq.code(((Algo)$1.obj).simb);
               }
      | RECTANGULO NUMBER NUMBER NUMBER NUMBER {
                    maq.code("constpush");
                    maq.code(((Algo)$2.obj).simb);
                    maq.code("constpush");
                    maq.code(((Algo)$3.obj).simb);
                    maq.code("constpush");
                    maq.code(((Algo)$4.obj).simb);
                    maq.code("constpush");
                    maq.code(((Algo)$5.obj).simb);
                    maq.code("rectangulo");
               }
      | LINE NUMBER NUMBER NUMBER NUMBER {
                    maq.code("constpush");
                    maq.code(((Algo)$2.obj).simb);
                    maq.code("constpush");
                    maq.code(((Algo)$3.obj).simb);
                    maq.code("constpush");
                    maq.code(((Algo)$4.obj).simb);
                    maq.code("constpush");
                    maq.code(((Algo)$5.obj).simb);
                    maq.code("line");
               }
```

```
    | CIRCULO NUMBER NUMBER NUMBER {
                maq.code("constpush");
                maq.code(((Algo)$2.obj).simb);
                maq.code("constpush");
                maq.code(((Algo)$3.obj).simb);
                maq.code("constpush");
                maq.code(((Algo)$4.obj).simb);
                maq.code("circulo");
              }
    | COLOR NUMBER { maq.code("constpush");
                maq.code(((Algo)$2.obj).simb);
                maq.code("color");
              }
    ;
```

Cabe resaltar que se hace un uso dedicado de la función, “constpush”, ésta hace referencia a una pila en el archivo denominado como “maquina.java” de la cual se obtendrán los parámetros para ejecutar los comandos que se ingresan.

```
    void constpush() {
        Simbolo s;
        Double d;
        s = (Simbolo) prog.elementAt(pc);
        pc = pc + 1;
        pila.push(new Double(s.val));
    }
```

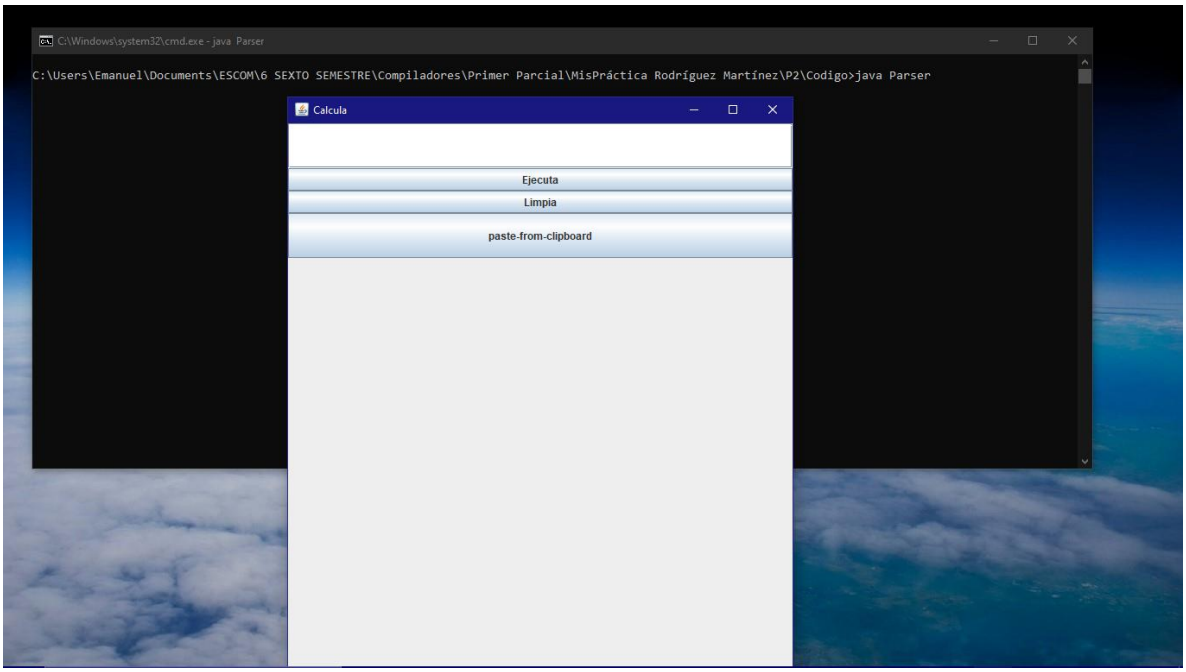
De acuerdo a la característica de la estructura de datos que se usa, se harán cambios en las funciones “line()”, “rectángulo()”, y “circulo()”, con la finalidad de obtener los parámetros de la cadena del último al primero (esto se debe al orden en que han sido ingresados a la pila); a continuación se ejecutara la creación, de acuerdo a cada caso.

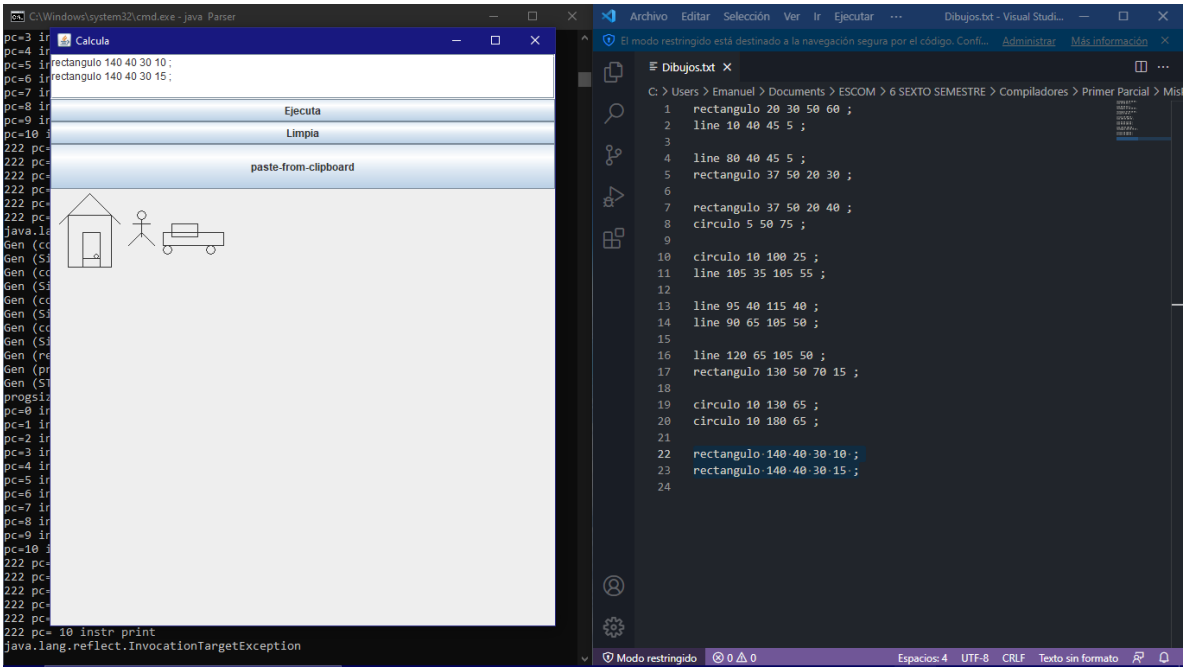
```
void line() {
    double d1 = 0, d2 = 0, d3 = 0, d4 = 0;
    d4 = ((Double) pila.pop()).doubleValue();
    d3 = ((Double) pila.pop()).doubleValue();
    d2 = ((Double) pila.pop()).doubleValue();
    d1 = ((Double) pila.pop()).doubleValue();
    if (g != null) {
        (new Linea((int) d1, (int) d2, (int) (d3), (int) (d4))).dibujar(g);
    }
}
```

```
void rectangulo() {
    double d1 = 0, d2 = 0, d3 = 0, d4 = 0;
    d4 = ((Double) pila.pop()).doubleValue();
    d3 = ((Double) pila.pop()).doubleValue();
    d2 = ((Double) pila.pop()).doubleValue();
    d1 = ((Double) pila.pop()).doubleValue();
    if (g != null) {
        (new Rectangulo((int) d1, ((int) d2), (int) d3, ((int) d4))).dibujar(g);
    }
}
```

```
void circulo() {
    double d1 = 0, d2 = 0, d3 = 0;
    d3 = ((Double) pila.pop()).doubleValue();
    d2 = ((Double) pila.pop()).doubleValue();
    d1 = ((Double) pila.pop()).doubleValue();
    if (g != null) {
        (new Circulo((int) d2, (int) d3, (int) d1)).dibujar(g);
    }
}
```

Finalmente comenzaremos con las pruebas para comprobar una buena ejecución del programa.





Conclusiones

Gracias al desarrollo de esta práctica he podido observar de mejor manera la forma en la que trabaja BYACC, siendo un complemento a la especificación que se tiene con el uso de archivos generados en tipo C, y gracias al uso de código Java, es posible extender las funcionalidades del intérprete. A tal grado de poder realizar dibujos como el de esta práctica.

Es importante recalcar que esta herramienta no representa una alternativa, por el contrario, un aditamento para realizar las especificaciones que requieran alguna función que es alcanzable en C.

Finalmente, quiero comentar que la estructura de BYACC es similar a la de YACC, pero el código debe de ser relacionado al lenguaje C o Java.