Instituto Politécnico Nacional

Escuela Superior de Cómputo

Compiladores

Práctica #2
"Java"

3CM7

Alumno: Zepeda Flores Alejandro de Jesús

Profesor: Tecla Parra Roberto

INTRODUCCIÓN

HOC es un acrónimo para **High Order Calculator**, es un lenguaje de programación interpretado que fue usado en 1984 en el libro *"El Entorno de Programación de UNIX"* para demostrar como construir interpretes usando una herramienta llamda YACC y lenguaje C.

HOC fue desarrollado por Brian Kernighan y Rob Pike como una grandiosa calculadora interactiva. Su función básica es evaluar expresiones numéricas de puntos flotantes e.g. "1+2*sin(0.7)". Después variables fueron agregadas, expresiones condicionales, ciclos, funciones definidas por el usuario, simple entrada/salida y más, todo esto usando una sintaxis parecida a lenguaje C.

Hasta ahora, las 6 etapas de HOC son:

- HOC1: Calculadora Básica
- HOC2: Calculadora con 26 variables
- HOC3: Calculadora Científica
- HOC4: Máquina Virtual de Pila
- HOC5: Ciclos / Decisiones
- HOC6: Funciones y Procedimientos

OBJETIVO

En esta práctica, tomando el código que nos dio el profesor, y en lugar de sólo dibujar una figura, darle más parámetros y poder dibujas una figura personalizada.

DESARROLLO

Se modifico la sección de reglas:

```
inst:  NUMBER  {
    ((Algo)$$.obj).inst = maq.code("constpush");
    maq.code(((Algo)$1.obj).simb);
}
| RECTANGULO NUMBER NUMBER NUMBER NUMBER {
    //Push del primer símbolo gramatical         (  X   )
    maq.code("constpush");
    maq.code(((Algo)$2.obj).simb);
    //Push del segundo símbolo gramatical    (   Y   )
    maq.code("constpush");
    maq.code(((Algo)$3.obj).simb);
    //Push del tercer símbolo gramatical         ( ancho )
    maq.code("constpush");
    maq.code(((Algo)$4.obj).simb);
```

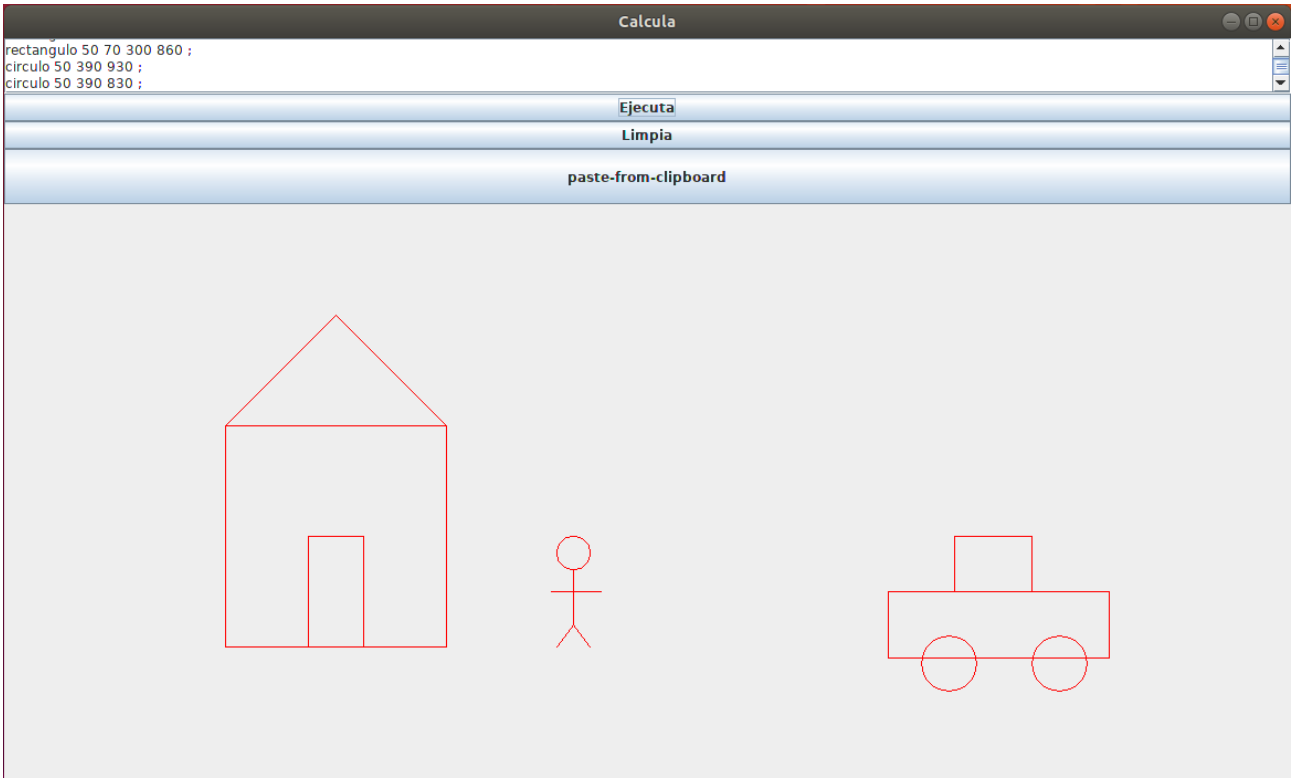
```
        //Push del cuarto símbolo gramatical        ( alto )
        maq.code("constpush");
        maq.code(((Algo)$5.obj).simb);
        maq.code("rectangulo");
    }
    | LINE NUMBER NUMBER NUMBER NUMBER {
        //Push del primer símbolo gramatical        (  X1   )
        maq.code("constpush");
        maq.code(((Algo)$2.obj).simb);
        //Push del segundo símbolo gramatical   (   Y1   )
        maq.code("constpush");
        maq.code(((Algo)$3.obj).simb);
        //Push del tercer símbolo gramatical        (   X2   )
        maq.code("constpush");
        maq.code(((Algo)$4.obj).simb);
        //Push del cuarto símbolo gramatical       (    Y2     )
        maq.code("constpush");
        maq.code(((Algo)$5.obj).simb);
        maq.code("line");
    }
    | CIRCULO NUMBER NUMBER NUMBER {
        //Push del primer símbolo gramatical        (  radio  )
        maq.code("constpush");
        maq.code(((Algo)$2.obj).simb);
        //Push del segundo símbolo gramatical   (   X   )
        maq.code("constpush");
        maq.code(((Algo)$3.obj).simb);
        //Push del tercer símbolo gramatical        (   Y   )
        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");
    }
    ;
    %%
```

Como se uso una Maquina Virtual de pilas, se agregó este código:

```
    void line(){
        double X1, Y1, X2, Y2;
        //Obtenemos el primer valor, haciendo pop de la pila
        X1 = ((Double)pila.pop()).doubleValue();
        //Obtenemos el segundo valor, haciendo pop de la pila
        Y1 = ((Double)pila.pop()).doubleValue();
        //Obtenemos el tercer valor, haciendo pop de la pila
        X2 = ((Double)pila.pop()).doubleValue();
        //Obtenemos el cuarto valor, haciendo pop de la pila
        Y2 = ((Double)pila.pop()).doubleValue();

        //Los gráficos no deben ser nulos para poder dibujar
        if(g!=null){
            //Creamos un objeto Linea con los datos obtenidos de la pila
            (   new Linea((int)X1, (int)Y1, (int)X2, (int)Y2)   ).dibuja(g);
        }
    }
    void circulo(){
        double radio, X, Y;
        //Obtenemos el valor del radio haciendo pop de la pila
        radio = ((Double)pila.pop()).doubleValue();
        //Obtenemos el valor de la posición  X haciendo pop de la pila
        X = ((Double)pila.pop()).doubleValue();
        //Obtenemos el valor de la posición Y haciendo pop de la pila
        Y = ((Double)pila.pop()).doubleValue();
        //Para poder dibujar la variable g no debe ser nula
        if(g!=null){
            //Creamos un nuevo objeto circulo
            (   new Circulo((int)radio, (int)X, (int)Y)    ).dibuja(g);
        }
    }
    void rectangulo(){
        double X, Y, ancho, alto;
        //Obtenemos el valor de la posición en X haciendo pop de la pila
        X = ((Double)pila.pop()).doubleValue();
        //Obtenemos el valor de la posición en Y haciendo pop de la pila
        Y = ((Double)pila.pop()).doubleValue();
        //Obtenemos el valor de la anchura del rectangulo haciendo pop de la pila
        ancho = ((Double)pila.pop()).doubleValue();
        //Obtenemos el valor de la altura dle rectangulo haciendo pop de la pila
        alto = ((Double)pila.pop()).doubleValue();
        if(g!=null){
            (   new Rectangulo((int)X, (int)Y, (int)ancho, (int)alto )
).dibuja(g);
        }
    }
```

Pruebas de funcionamiento:



CONCLUSIONES

Aunque al principio me costó un poco de trabajo de identificar un error que tuve, esta práctica me resulto muy interesante y me puso a pensar en las cosas que se podrían desarrollar con YACC. Mucho más porque es sencillo, y aunque YACC sólo esta para Linux, el código generado puedo ser usado en otros sistemas operativos, como en Windows.