Traductor de un lenguaje procedural mediante JFlex y Java CUP

Ingeniería Técnica de Informática de Sistemas

Alumno: Jesús Velayos Pastrana
Tutor: Jesús López Sánchez
Junio 2017

Precedentes

- Años 2004-2007
- Asignatura Compiladores e Intérpretes
- Totalmente válido y útil actualmente para el estudio de teoría de compiladores e intérpretes

Objetivos

- Finalidad formativa
- Implementación con lenguaje orientado a objetos: Java
- Uso de herramientas JFlex y Java CUP
- Interfaz gráfico que ayude a la finalidad formativa

JFlex (I)

- Herramienta generadora de analizadores léxicos en lenguaje Java
- Desarrollada originalmente por Gerwin Klein y actualmente mantenida bajo su propio proyecto
- Basada en JLex que a su vez está basado en lex/flex
- Funcionamiento basado en autómatas finitos deterministas
- Preparado para trabajar en conjunto con Java CUP
- Licencia BSD

JFlex (II)

```
declaraciones java
%%
declaraciones Jflex
%{
código Java
%}
declaración de macros
%%
reglas y acciones
```

Java CUP (I)

- Herramienta generadora de analizadores sintácticos LALR en Java (nombre actual CUP)
- Desarrollada originalmente por Scott E. Hudson modificada por varios autores posteriormente y mantenida actualmente por Technical University of Munich
- Basado en yacc/bison
- Define un runtime, con el que debe trabajar el resto del software

Java CUP (II)

- Define un runtime, con el que debe trabajar el resto del software:
  - clase java Symbol para instanciar objetos con las piezas sintácticas
  - interfaz java Scanner cuya especificación es la necesaria para la implementación de un scanner compatible con Java CUP

Java CUP (III)

```
declaraciones java

código de usuario

listas de símbolos

declaraciones de precedencia y asociación

gramática (producciones y acciones)
```

Definición del lenguaje JSPascal (I)

- Subconjunto Pascal:
  - tipos primitivos **entero, real y boolean**
  - uso de **tablas** de una dimensión
  - uso de **registros** con al menos un campo
  - definición de **literales nominativos** y **constantes enteras y reales**
  - procedimientos predefinidos **READ** y **WRITE**
  - **procedimientos con parámetros por valor y referencia**

Definición del lenguaje JSPascal (II)

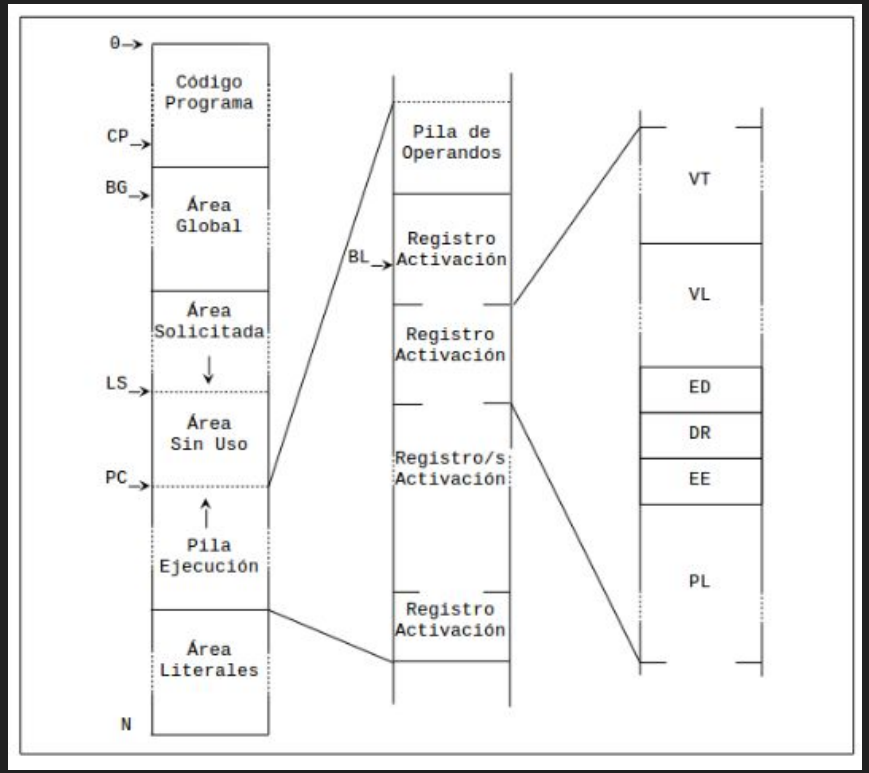
- sentencia **IF-ELSE**
- bucles **WHILE** y **REPEAT**
- operadores condicionales >, <, <=, >=, = y <>
- **operadores aritméticos** +. **-**, *, DIV y MOD
- **operadores lógicos** AND, OR y NOT
- se permite el **uso de recursividad**
- se permite el **anidamiento ilimitado de subprogramas**

Definición del lenguaje JMPascal

- Lenguaje intermedio con instrucciones para realizar:
  - lectura y escritura de memoria
  - lectura desde los registros de la máquina virtual
  - operaciones aritméticas
  - conversión de operandos
  - gestión de direccionamiento
  - lectura de la entrada estándar
  - escritura a la salida estándar

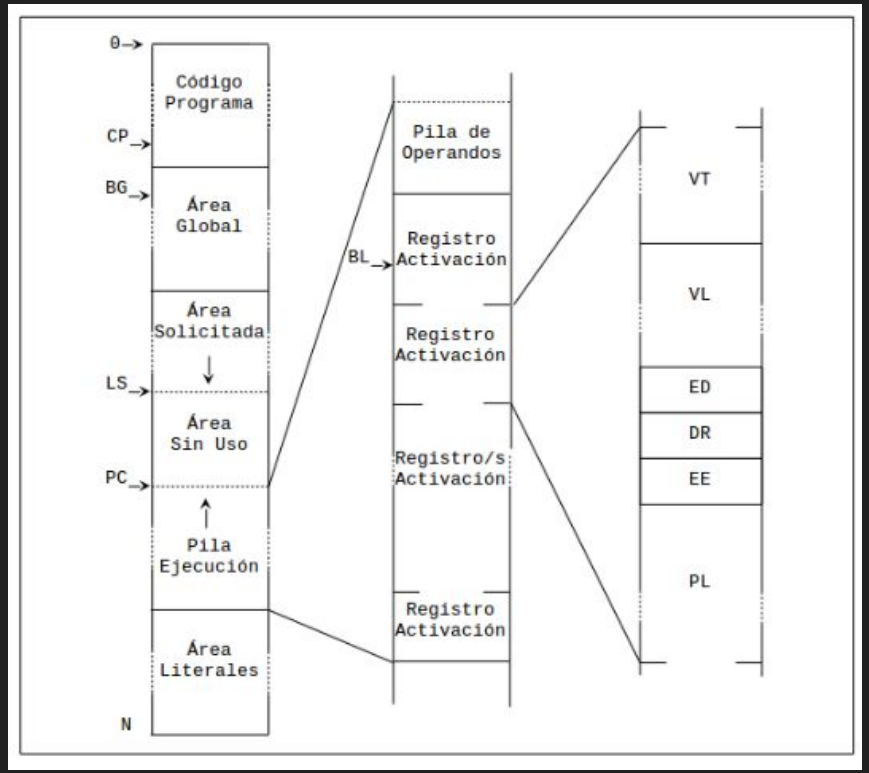
Máquina Virtual JMPascalVM (I)

- Tamaño memoria fijo
- Áreas de memoria:
  - Área de código
  - Área global
  - Área solicitada
  - Pila de ejecución
  - Área de literales



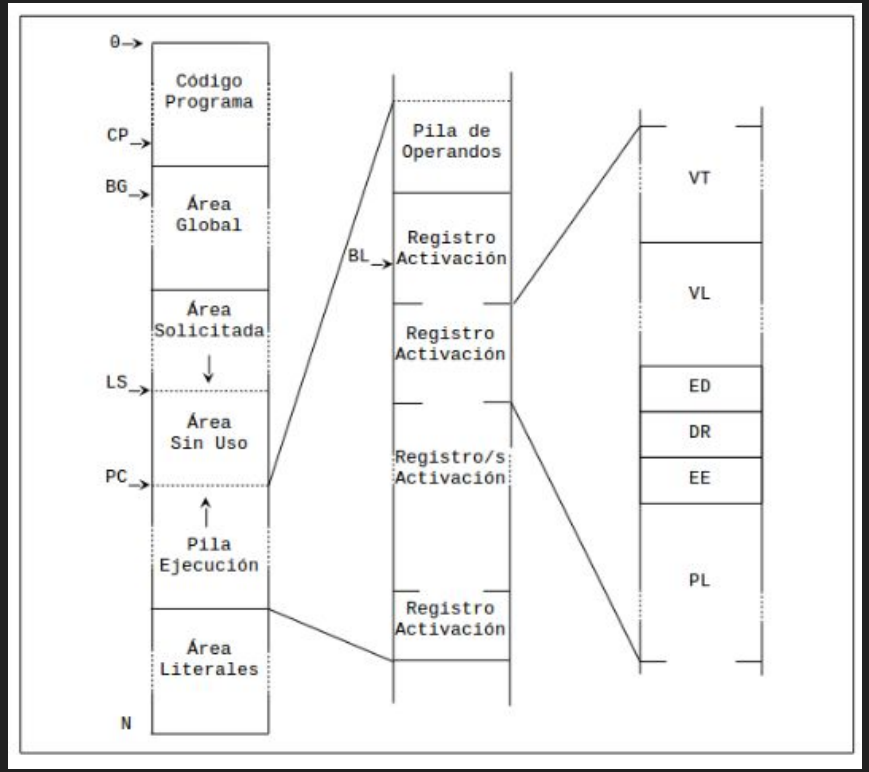
Máquina Virtual JMPascalVM (II)

- Registros internos:
  - CP: Contador Programa
  - PC: Puntero Cima
  - BG: Base Global
  - BL: Base Local
  - LS: Límite Solicitado

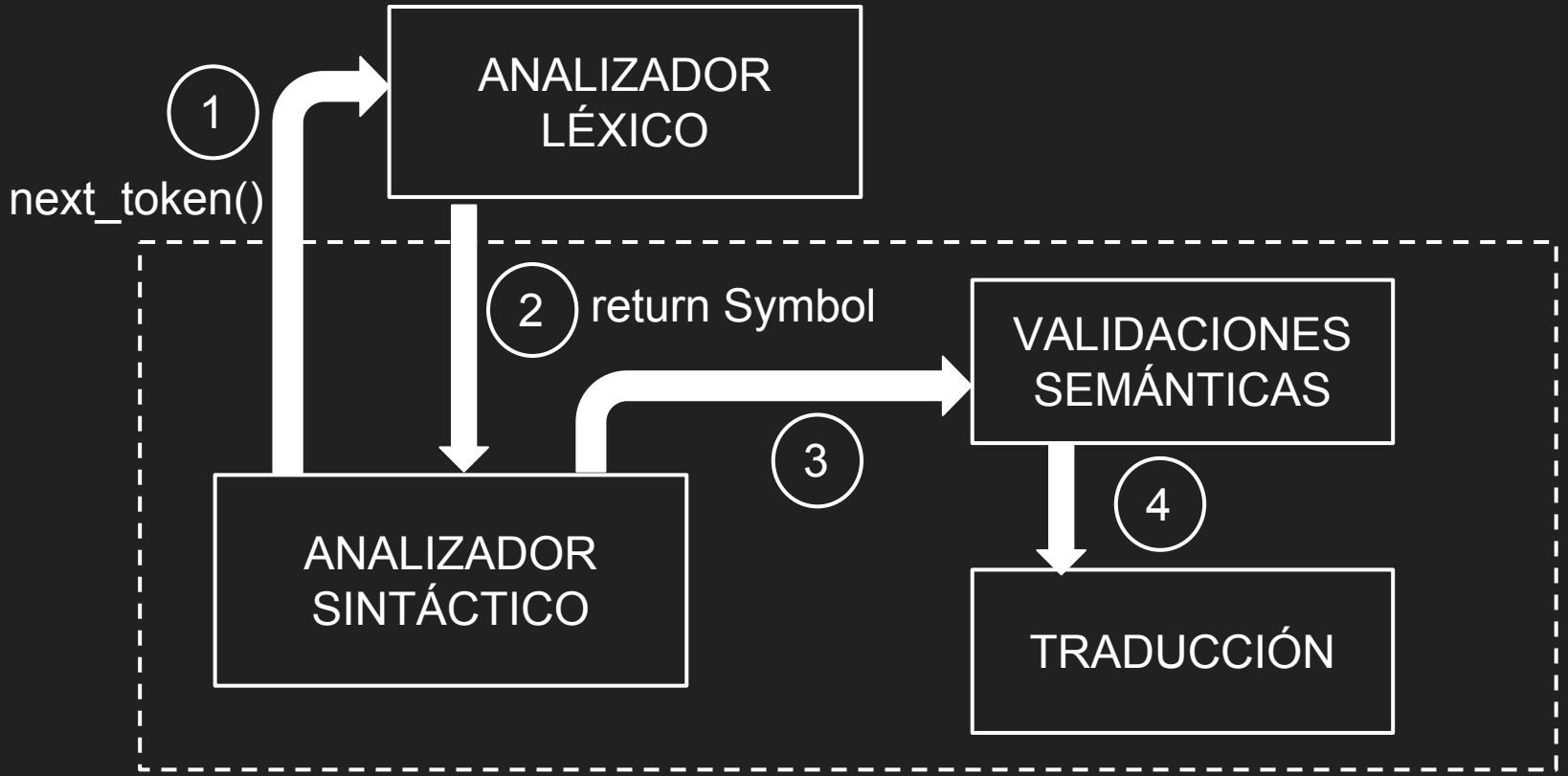


Máquina Virtual JMPascalVM (III)

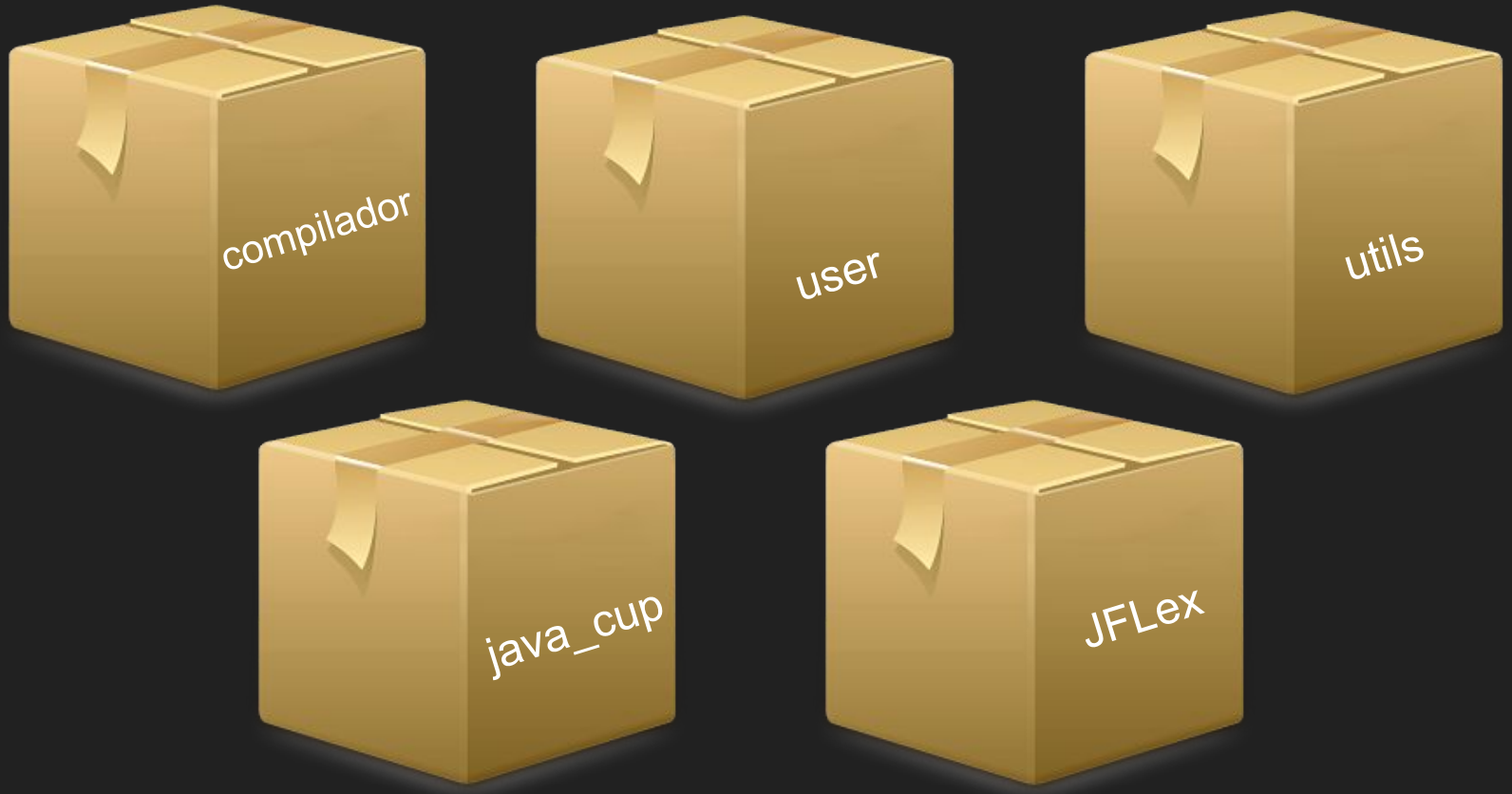
- Registro de activación
  - PL: parámetro locales
  - EE: Enlace estático
  - DR: Dirección de retorno
  - ED: Enlace dinámico
  - VL: Variables locales
  - VT: Variables temporales



Implementación (I)



Implementación (II)



Implementación (III)

- Implementa el **interfaz gráfico**
- Implementado con **Java Swing**
- Implementa el método principal para la ejecución del progama Java en la clase principal Screen que **hereda de JFrame**
- Dispone de otras clases que implementan cada parte del interfaz gráfico y que igualmente heredan de clases de contenedores de Java Swing (JPanel, JInternalFramey jDesktopPane)
- Controla/dispara la ejecución del resto del software mediante eventos

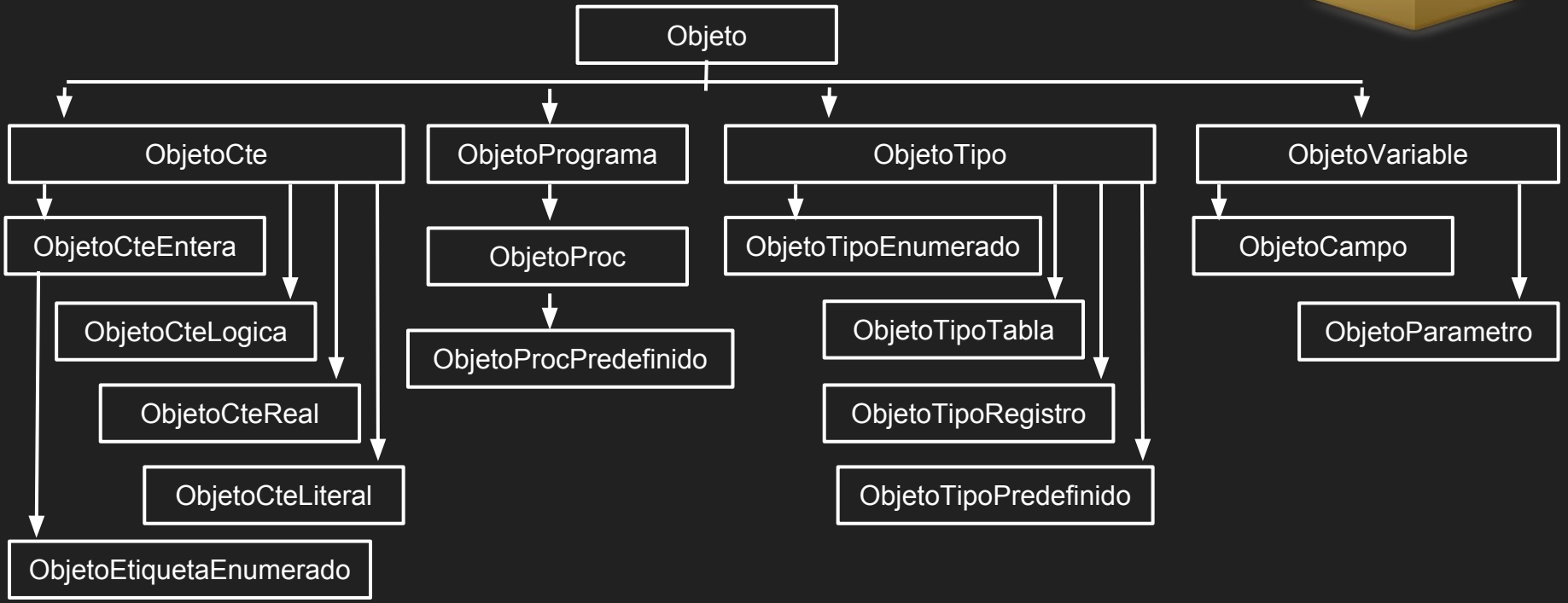
Implementación (IV)

- Implementa la **lógica del traductor JPascal a JMPascal**
- Integra el **parser generado por Java CUP y el scanner generado por JFlex** en las clases Parser.java y Lexer.java respectivamente
- Incluye también la clase Sym.java generada por Java CUP que se usa tanto en el parser como en el scanner para definir las constantes de los diferentes símbolos

Implementación (V)

- Incluye la **clase Traductor.java** que dispone del método que permite lanzar el proceso de traducción de un fichero fuente JSPascal
- El traductor se apoya en una **tabla de símbolos implementada en la clase TablaSimbolos.java**
- Los **objetos** de la tabla de símbolos se instancian a partir de clases definidas **mediante el uso de herencia**

Implementación (VI)



Interfaz


JSPascalator
Archivo Edición Herramientas Ayuda
MENÚ Y HERRAMIENTAS
MEMORIA
EDITOR
REGISTROS
CP
PC
S
BG
BL
E/S ESTÁNDAR
Inicializando Programa.... Inicializado
LOG

Conclusiones

- La implementación con un lenguaje orientado a objetos y concretamente con Java es totalmente viable y la herencia ayuda a generar una implementación de calidad aceptable
- La obligación de embeber código en la especificación Java CUP hace hace necesario orden en el código para su mantenimiento
- El interfaz gráfico permite conseguir una herramienta sencilla y evolucionable en el sentido de facilitar el análisis y por tanto el aprendizaje
- La implementación y la disponibilidad pública del proyecto ayudan a que pueda ser usado con fines formativos

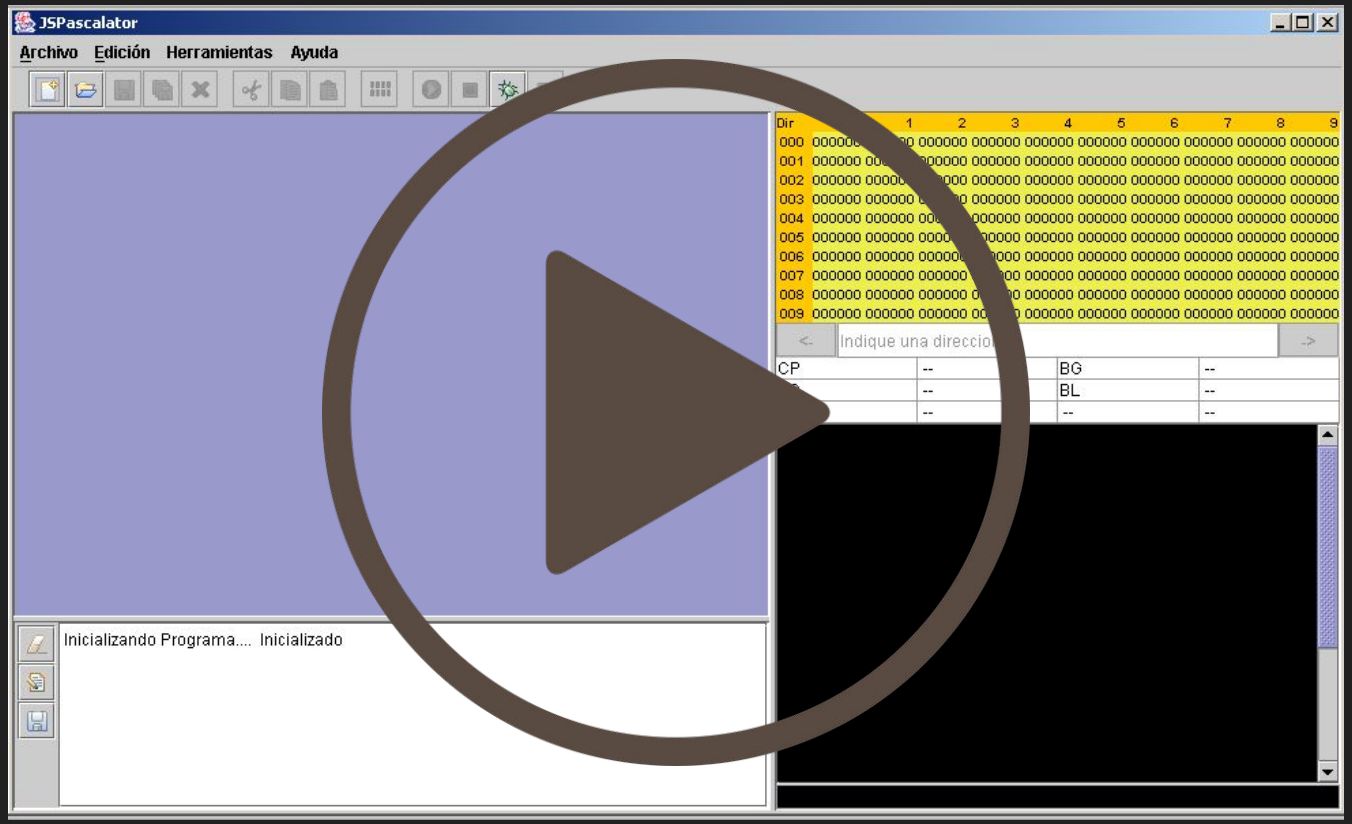
Ampliaciones (I)

- La ampliación clara y evidente del proyecto es la implementación del intérprete integrado en el mismo interfaz gráfico para mantener el objetivo formativo (ya implementado)
- Permitir la ejecución paso a paso en el intérprete descrito anteriormente (ya implementado)
- Incrementar el subconjunto de pascal implementado incluyendo, funciones, punteros, etc.

Ampliaciones (II)

- Incremento de las funcionalidades del interfaz gráfico:
  - visualización de los registros de activación
  - visualización de la tabla de símbolos
  - resaltado de sintaxis en el editor y mejoras visuales
  - establecimiento y resalte de puntos de parada

DEMO (I)


JSPascalator
Archivo Edición Herramientas Ayuda
Dir 1 2 3 4 5 6 7 8 9
Indique una direcci
CP -- BG --
-- BL --
-- --
Inicializando Programa.... Inicializado

DEMO (II)

- AINDVALOR (2)
- AREAL (3)
- ARESFUNC (4)
- BIFCOND (5)
- BIFINCOND (6)
- CCONSTANTE (7)
- CCONSTANTEREAL (8)
- CDIRGLOBAL (9)
- CDIRINTERM (10)
- CLITERAL (11)
- CDIRLOCAL (12)
- CINDVALOR (13)
- CONJUNCION (14)
- CREGISTRO (15)
- DCAMPO (16)
- DEFLITERAL (17)
- DELEMENTO (18)
- DEVOLUCION (19)
- DISTINTO (20)
- DISTINTOREAL (21)
- DIVISION (22)
- DIVISIONREAL (23)
- DISYUNCION (24)
- GRABACION (25)
- GRABACIONLITERAL (26)
- GRABACIONREAL (27)
- IGUAL (28)
- IGUALREAL (29)
- LECTURA (30)
- LECTURAREAL (31)
- LIMITACION (32)
- LLAMADA (33)
- MAYORIGU (34)
- MAYORIGUREAL (35)
- MAYOR (36)
- MAYORREAL (37)
- MENORIGU (38)
- MENORIGUREAL (39)
- MENOR (40)
- MENORREAL (41)
- MODULO (42)
- MODULOREAL (43)
- NEGACION (44)
- NEGATIVO (45)
- OCUPAESP (46)
- PRODUCTO (47)
- PRODUCTOREAL (48)
- RESERVAESP (49)
- RESTA (50)
- RESTAREAL (51)
- SUMA (52)
- SUMAREAL (53)
- TAMFUNCION (54)
- TAMGLOBAL (55)
- TERMINACION (56)

DEMO (III) - demo01.pas

- Prueba de análisis léxico

Encontrado símbolo: bEGin (4) en la línea: 0 en la columna: 0

….

Encontrado símbolo: x (47) en la línea: 11 en la columna: 3

….

Encontrado símbolo: 56.985 (50) en la línea: 43 en la columna: 15

DEMO (IV) - demo02.pas

```
program demo02;

const

   N=5;

   N2=N*N;

   FACTOR1=0.8;

   FACTOR2=FACTOR1+N2;

   EXITO=TRUE;

   LITERAL1='Demo PFC';
```

```
var

   primera,segunda: integer;

   tercera: real;

   cuarta: boolean;

begin

end.
```

DEMO (V) - demo02.pas

```
00 #DEFLITERAL 'Demo PFC'

01 *0

02 #TAMGLOBAL 5

03 #TAMFUNCION 0

04 TERMINACION
```

DEMO (VI) - demo03.pas

```
program demo03;

const

   N=5;

   N2=N*N;

   FACTOR1=0.8;

   FACTOR2=FACTOR1+N2;

   EXITO=TRUE;

   LITERAL1='Demo PFC';
```

```
var

   primera,segunda: integer;

   tercera: real;

   cuarta: boolean;

begin

   primera:=N

end.
```

DEMO (VII) - demo03.pas

```
00 #DEFLITERAL 'Demo PFC'
01 *0
02 #TAMGLOBAL 5
03 #TAMFUNCION 0
04 CCONSTANTE 5
05 CDIRGLOBAL 0
06 AINDVALOR 1
07 TERMINACION
```

DEMO (VIII) - demo04.pas

```
program demo04;

const

   cadena='El valor de PI es ';

   PI=3.14159;

   procedure uno(x:real);

      procedure dos(x:real);

         procedure tres(x:real);

         begin

            WRITE(cadena);

            WRITE(x)

         end;

      begin

         tres(x);

      end;

   begin

      dos(x);

   end;

begin

   uno(PI)

end.
```

DEMO (IX) - demo04.pas

```
00 #DEFLITERAL 'El valor de PI es '
01 *3
02 RESERVAESP 0
03 CLITERAL 18 0
04 GRABACIONLITERAL
05 CDIRLOCAL 3
06 CINDVALOR 2
07 GRABACIONREAL
08 DEVOLUCION 3
```

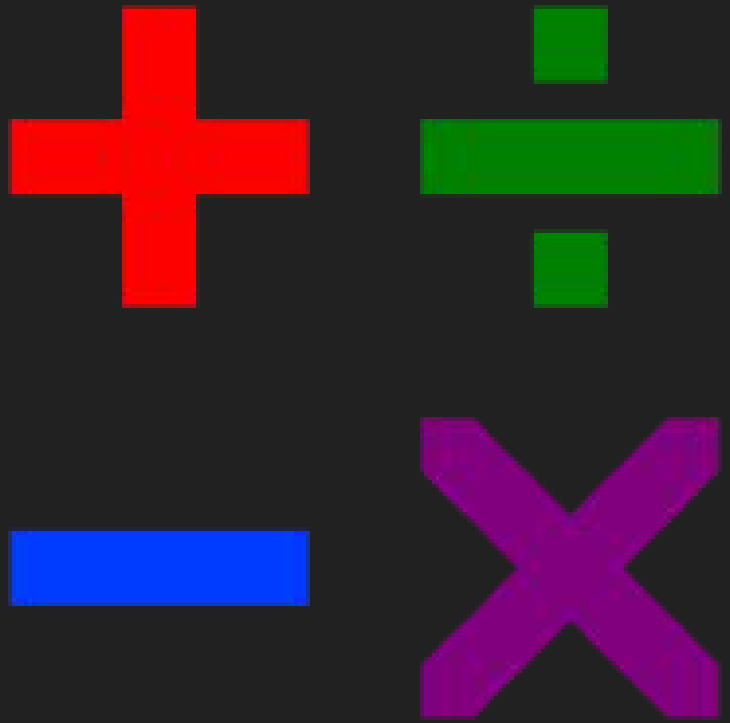
```
09 *2
10 RESERVAESP 0
11 CDIRLOCAL 3
12 CINDVALOR 2
13 CDIRLOCAL 0
14 LLAMADA 3
15 DEVOLUCION 3
```

DEMO (X) - demo04.pas

```
16 *1
17 RESERVAESP 0
18 CDIRLOCAL 2
19 CINDVALOR 2
20 CDIRLOCAL 0
21 LLAMADA 2
22 DEVOLUCION 2
```

```
23 *0
24 #TAMGLOBAL 0
25 #TAMFUNCION 0
26 CCONSTANTEREAL 3.14159
27 LLAMADA 1
28 TERMINACION
```

DEMO (XI) - demo05.pas



GRACIAS

 **Proyecto en github**

 **Diapositivas**

 **Memoria PFC**

 **@txetxuvel**