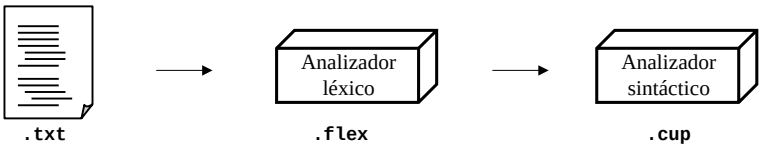
Cup y el análisis semántico

Hasta ahora...

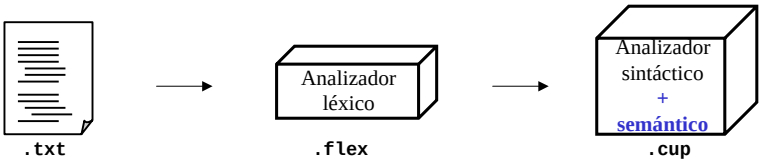


Fichero fuente (tablas y consultas)

Lee lexemas, asocia a patrones, genera tokens

Comprueba validez de la cadena de tokens

Desde ahora...



Fichero fuente (tablas y consultas)

Lee lexemas, asocia a patrones, genera tokens

Comprueba validez de la cadena de tokens

Comprueba tipos y existencias, se ayuda de la tabla de símbolos

Especificación Cup

```
import ...;

action code {: ... :}
parser code {: ... :}
init with {: ... :}
scan with {: ... :}

terminal ...;
non terminal ...;

precedence ...;

gramática
```

Especificación Cup

```
import ...;

action code {: ... :}
parser code {: ... :}
init with {: ... :}
scan with {: ... :}

terminal ...;
non terminal ...;

precedence ...;

gramática
```

Importación de paquetes necesarios para poder desarrollar el código Java

Especificación Cup

```
import ...;

action code {: ... :}
parser code {: ... :}
init with {: ... :}
scan with {: ... :}

terminal ...;
non terminal ...;

precedence ...;

gramática
```

Son áreas opcionales.

Action code: código que contiene métodos auxiliares y variables empleados por el código incrustado en la gramática, este código se incrusta en una clase embebida del parser.

Parser code: código que flexibiliza el uso del parser, este código se incrusta directamente en la clase **`parser`**.

Especificación Cup

```
import ...;

action code {: ... :}
parser code {: ... :}
init with {: ... :}
scan with {: ... :}

terminal ...;
non terminal ...;

precedence ...;

gramática
```

Son áreas opcionales.

Init with: El parser ejecutará el código aquí introducido antes de pedir el primer token. Inicializaciones, instanciaciones...

Scan with: código que devolverá símbolos.

Especificación Cup

```
import ...;

action code {: ... :}
parser code {: ... :}
init with {: ... :}
scan with {: ... :}

terminal ...;
non terminal ...;

precedence ...;

gramática
```

Área donde definir todos los símbolos que aparecerán en la gramática.

Se les puede asignar un tipo (clase) para ajustarlos a las necesidades del analizador.

Para evitar ambigüedades, se deben definir precedencias.

Especificación Cup

```
import ...;

action code {: ... :}
parser code {: ... :}
init with {: ... :}
scan with {: ... :}

terminal ...;
non terminal ...;

precedence ...;

gramática
```

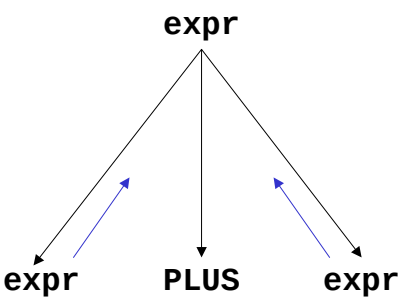
Se define la especificación sintáctica y **se incluyen los atributos y las acciones semánticas** que permiten manejar los símbolos leídos y realizar las acciones oportunas (comprobaciones, inserciones...) sobre la **tabla de símbolos**

G.I.C. con atributos + A.Semánticas

= Traductor dirigido por la sintaxis

Especificación Cup

```
expr::= expr PLUS expr
        {: acción semántica :}
```



-Uso de identificadores (¿declarados?)

-Variables asignadas antes de usar

-Índices de array dentro de rango válido

-Operandos adecuados en expresiones

-Invocación correcta de métodos

-Tipo de valor de retorno adecuado en método

-...

Especificación Cup: gramática

```
...
terminal     int, float, id, semicolon, comma;
non terminal DECL, T, LID;
...
// Gramática
...
DECL ::= T LID semicolon;
T ::= int {: T.tipo = int :}
       | float {: T.tipo = float :} ;
LID ::= id comma LID
       | id ;
```

```
int v1, x, indice;
```


Especificación Cup: gramática

```
...
terminal     int, float, id, semicolon, comma;
non terminal DECL, LID;
non terminal Simbolo T;
...
// Gramática
...
DECL ::= T LID semicolon;
T ::= int {:  Simbolo miSimbolo=new Simbolo();
              miSimbolo.setTipo(Simbolo.T_INT);
              RESULT= miSimbolo; :}
       | float {:    Simbolo miSimbolo =new Simbolo();
                     miSimbolo.setTipo(Simbolo.T_FLOAT);
                     RESULT= miSimbolo;  :} ;
LID ::= id comma LID
       | id ;
```

```
int v1, x, indice;
```

Especificación Cup: gramática

```
...
terminal     int, float, id, semicolon, comma;
non terminal DECL, LID;
non terminal Simbolo T;
...
// Gramática
...
DECL ::= T LID semicolon;
T ::= int {: ... :}
      | float {: ... :} ;
LID ::= id comma LID {: LID.lista = LID.lista.add(id) :}
      | id {: LID.lista = nuevaLista(id) :} ;
```

```
int v1, x, indice;
```


Especificación Cup: gramática

```
...
terminal     int, float, semicolon, comma;
terminal String id;
non terminal DECL; non terminal ArrayList<String> LID;
non terminal Simbolo T;
...
// Gramática
...
DECL ::= T LID semicolon;
T ::= int {: ... :}
      | float {: ... :} ;
LID ::= id:ident comma LID:l {: l.add(ident);RESULT = l;:}
  | id:ident {: ArrayList<String> lista=new
                              ArrayList<String>();
                lista.add(ident);
                RESULT = lista; :} ;
```

```
int v1, x, indice;
```

Especificación Cup: gramática

```
...
terminal     int, float, semicolon, comma;
terminal String id;
non terminal  DECL; non terminal ArrayList LID;
non terminal  Simbolo T;
...
// Gramática
...
DECL ::= T LID semicolon {: // Introducir en tabla de
                            // símbolos realizando las
                           // comprobaciones necesarias :};
T ::= int {: ... :} | float {: ... :} ;
LID ::= id:ident comma LID:l {: ... :}
       | id:ident {: ... :} ;
```

EJERCICIO

Implementar la acción semántica:

- Comprobando que no hay repeticiones
- Realizando inserción en tabla de símbolos

Especificación Cup: gramática

```
...
terminal PLUS, MINUS, TIMES, DIV, AND, OR, CONCAT;
non terminal      Simbolo     expr;
...
// Gramática
...
expr::= expr:e1 PLUS expr:e2
        {:
          //...

        :} | expr1:e1 AND expr:e2
        {:
          //...

        :};
```

Clase “Simbolo.java”

```
public class Simbolo
{
  final int T_INT = 1;
  final int T_FLOAT = 2;
  ...
  int tipo;      // Entero, Float...
  int clase;     // Var. local, método, clase...

  public Simbolo()
  {
        //...
  }

  public void setTipo(int valorTipo)
  {
    this.tipo=valorTipo;
  }

  public int getTipo()
  {
    return this.tipo;
  }
  ...
}
```