ST0244-032
Clase 19

J.F. Cardona

Universidad EAFIT

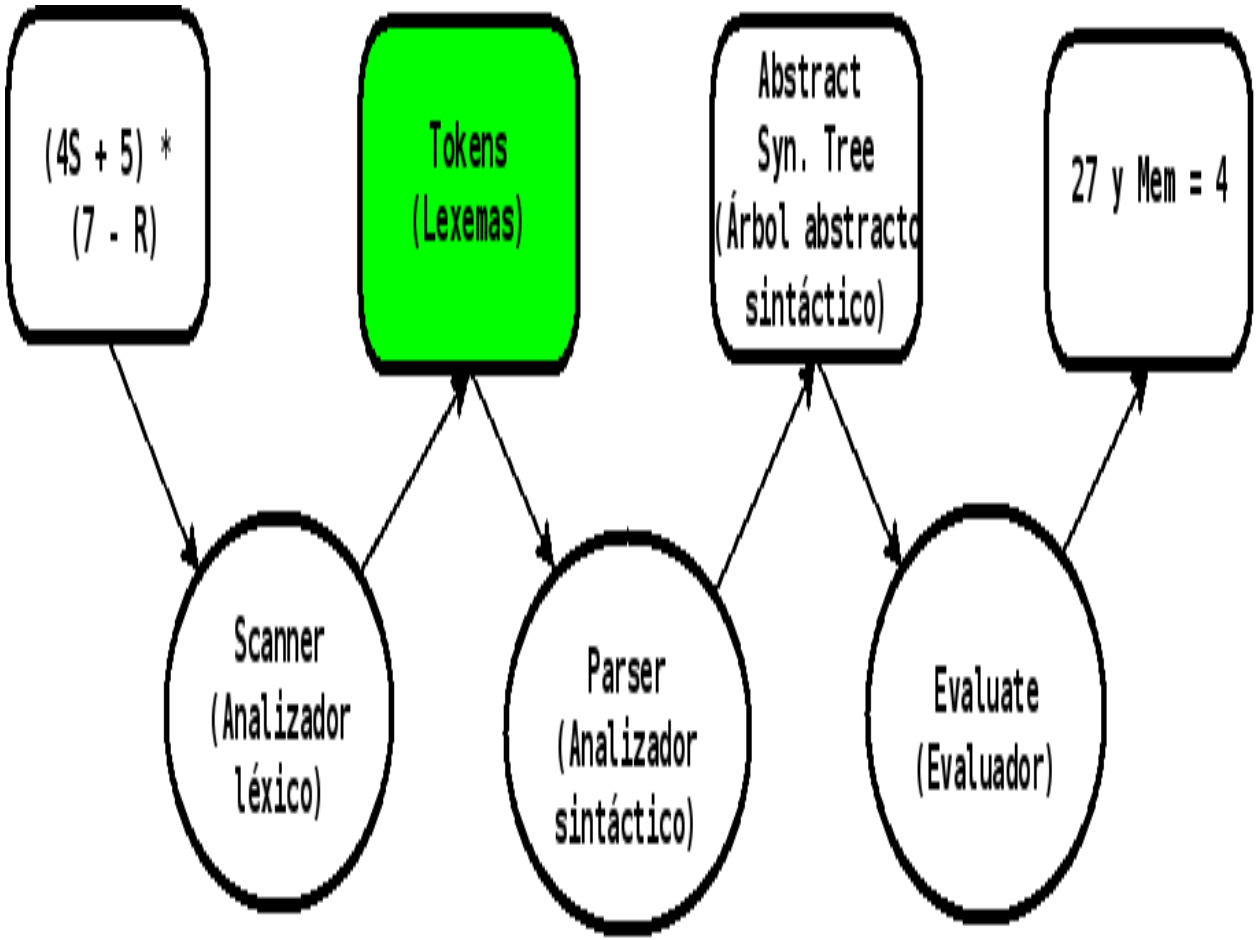
23 de septiembre de 2015

Agenda

1. Capítulo 3. Programación orientada a objeto con C++
   - Aplicación
   - La clase Token
     - Implementación de una clase
     - Estructuras versus clases
     - Un vistazo histórico al paso de parámetros
     - Constantes en C++

Programación orientada a objetos con C++

Aplicación



Programación orientada a objetos con C++

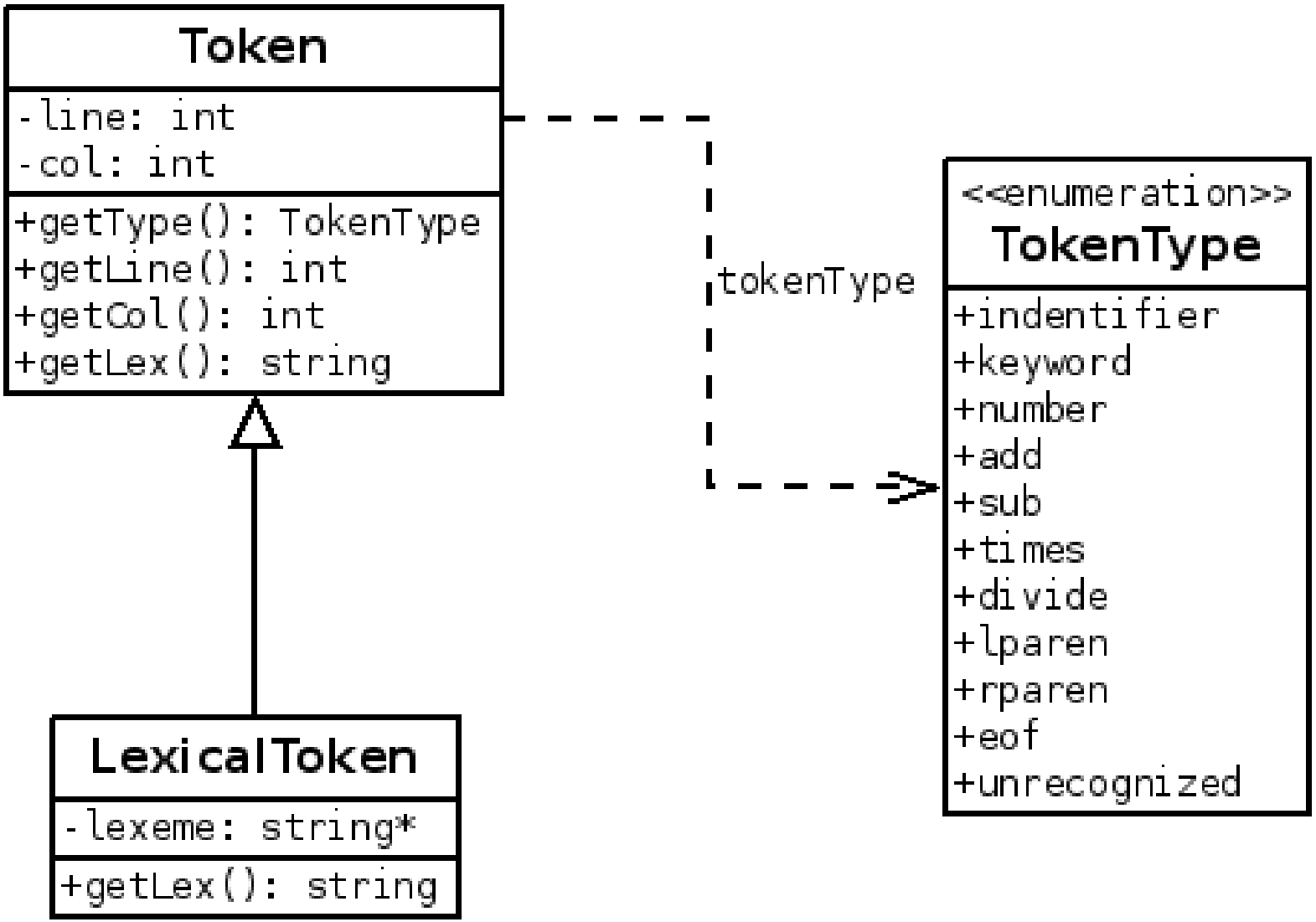
La clase Token

- El lenguaje tiene varios tipos de lexemas (*tokens*).
- La lista completa es: número, $+$, $-$, $\times$, $/$, $($, $)$, S y R.
- Es necesario tener una representación interna de tales lexemas (*tokens*) a través de una enumeración:

```
1 enum TokenType {
      identifier ,keyword ,number ,add ,sub ,times ,divide ,
3     lparen ,rparen ,eof ,unrecognized
  };
```

Programación orientada a objetos con C++

La clase Token



Programación orientada a objetos con C++

La clase Token - Tipos de datos enumerados

- Un tipo de dato enumerado es un tipo que permite almacenar un conjunto de valores definidos por el usuario.
- Estos valores están asociados a un valor entero que pueden ser definidos implícitamente o explícitamente.
- Ejemplos:

```
enum diaSemana { DOMINGO, LUNES, MARTES, MIERCOLES,
                 JUEVES, VIERNES, SABADO };
enum meses { ENE, FEB, MAR, ABR, MAY, JUN, JUL, AGO,
             SEP, OCT, NOV, DEC };
enum E1 { DIA, NOCHE };
enum E2 { S1 = 1, S20 = 20 };
enum E3 { MIN = -20, MAX = 1000 };
```

Programación orientada a objetos con C++

La clase Token

```
class Token {
public:

  Token();
  Token(TokenType typ, int line, int col);
  virtual ~Token();

  TokenType getType() const;
  int getLine() const;
  int getCol() const;
  virtual string getLex() const;

private:

  TokenType type;
  int line,col;
};
```

Programación orientada a objetos con C++

La clase Token

```
1 class LexicalToken: public Token {
  public:
3   LexicalToken(TokenType typ, string* lex, int line, int col);
    ~LexicalToken();
5
    virtual string getLex() const;
7
  private:
9   string* lexeme;
};
```

Programación orientada a objetos con C++

La clase Token - Implementación

```
Token::Token() : type(eof), line(0), col(0) { }

Token::~Token() { }

Token::Token(TokenType typ, int lineNum, int colNum) :
  type(typ),line(lineNum), col(colNum) { }

TokenType Token::getType() const { return type; }

int Token::getLine() const { return line; }

int Token::getCol() const { return col; }
```

Programación orientada a objetos con C++

La clase Token - Implementación

```
LexicalToken::LexicalToken(TokenType typ, string* lex,
                           int lineNum, int colNum) :
    Token(typ,lineNum,colNum), lexeme(lex) { }

LexicalToken::~LexicalToken() {
    try {
        delete lexeme;
    } catch (...) { }
}

string LexicalToken::getLex() const {
  return *lexeme;
}
```

Programación orientada a objetos con C++

Estructuras versus clases

- C++ cuenta también con otro tipo de dato para definir tipos de datos de usuario: *struct*.
- ¿Cuál es la diferencia de ambos?
- Ambos permiten definir funciones miembros y atributos.
- La diferencia es que las *struct* en C++ por omisión todos los métodos y los atributos son públicos, mientras que en las *class* por omisión todos los atributos son privados.
- Las *struct* permiten también herencia.
- Ejemplo.

Programación orientada a objetos con C++

Un vistazo histórico al paso de parámetros

```
1  procedure lookup_term(name:ident_type; var found:boolean;
                                          var place:integer);
3
   var k:integer;
5
   begin
7  found := false;
   for k:=1 to non_term_index do
9     if terminal[k].ident = name then
      begin
11       found := true;
         place := k;
13    end;
   end; (* PROCEDURE *)
15
```

Programación orientada a objetos con C++

Constantes en C++

- La palabra reservada `const` permite definir constantes:

```
const int maxVal = 100;
2
```

- También se utiliza para resolver el problema de paso de parámetros por copia.