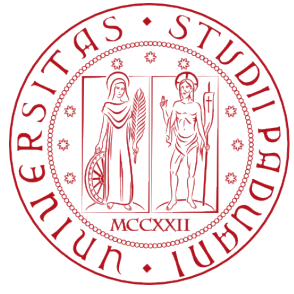
Automi e Linguaggi Formali

a.a. 2018/2019

LT in Informatica
3 Maggio 2019



Università
degli Studi
di Padova

Questo strumento può produrre in automatico il codice relativo a:

- Lexer: parte lessicale dove si definiscono i token e i separatori da ignorare
- Parser: parte grammaticale contenente le regole di produzione
- Listener: parte semantica che definisce le azioni da eseguire sul testo

Struttura di un Listener

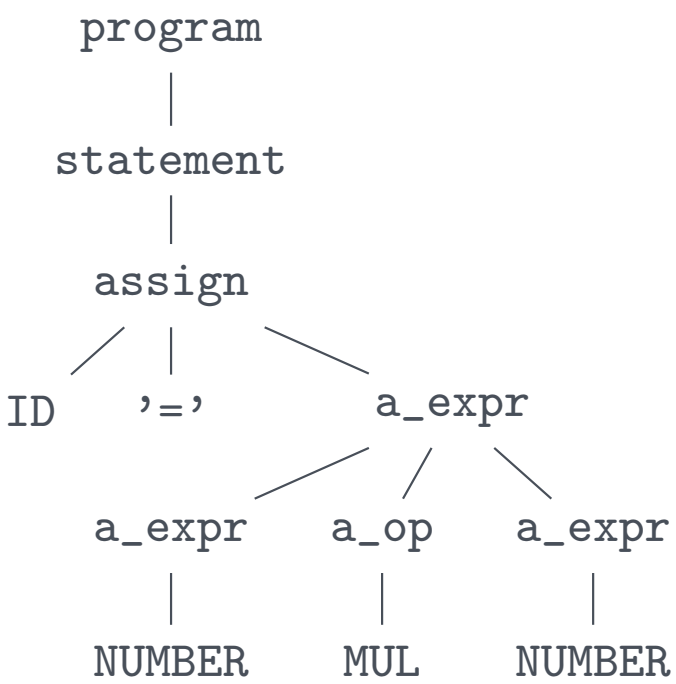
- Un listener è un oggetto che percorre l'albero sintattico generato dal parser ed esegue del codice
- La sua struttura è definita nel file `tinyrexxBaseListener.h` generato da `antlr4`
- Nel listener ci sono due metodi per ogni regola del parser:
  - un metodo che viene eseguito quando si entra in un nodo
  - un metodo che viene eseguito quando si esce da un nodo
- I metodi hanno come parametro il contesto della regola:
  - il contesto contiene un metodo per ogni simbolo terminale e nonterminale della regola
- I diversi tipi di contesti sono definiti nel file `tinyrexxParser.h` generato da `antlr4`

Esempio di Listener

- Programma: `n = 100 * 2`

Esempio di Listener

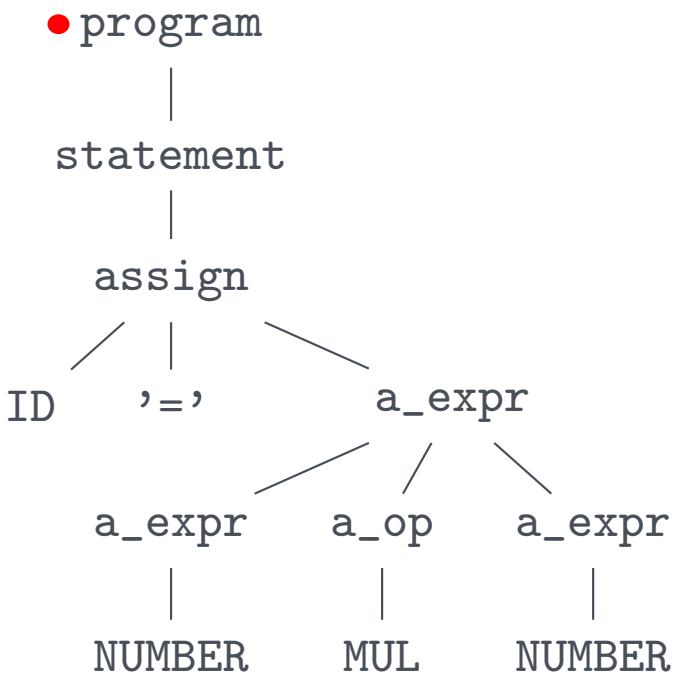
- Programma: `n = 100 * 2`
- Albero sintattico



Esempio di Listener

- Programma: `n = 100 * 2`
- Albero sintattico ed esecuzione del listener:

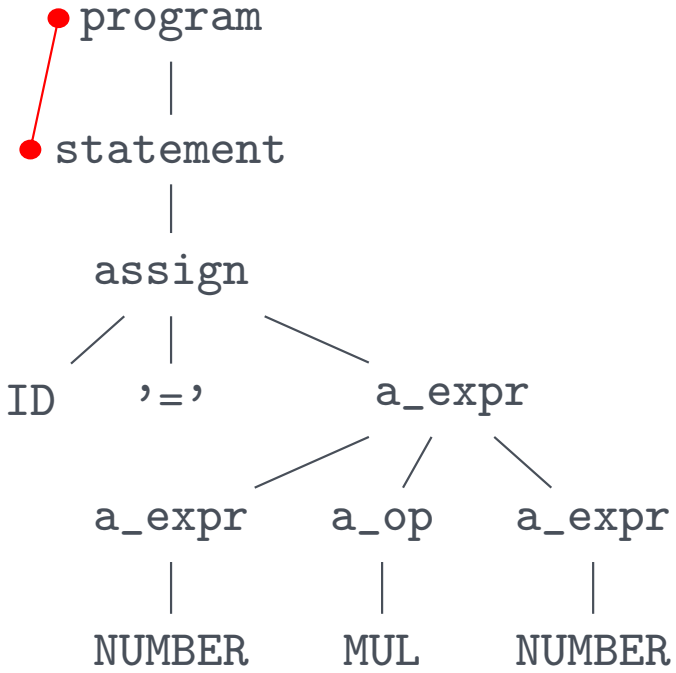
```
enterProgram(ProgramContext * ctx)
```



Esempio di Listener

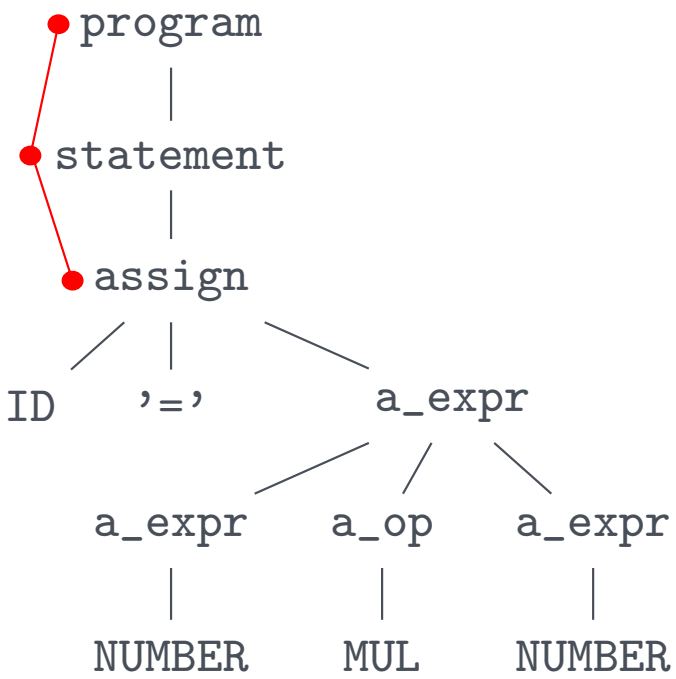
- Programma: `n = 100 * 2`
- Albero sintattico ed esecuzione del listener:

```
enterProgram(ProgramContext * ctx)
enterStatement(StatementContext * ctx)
```



Esempio di Listener

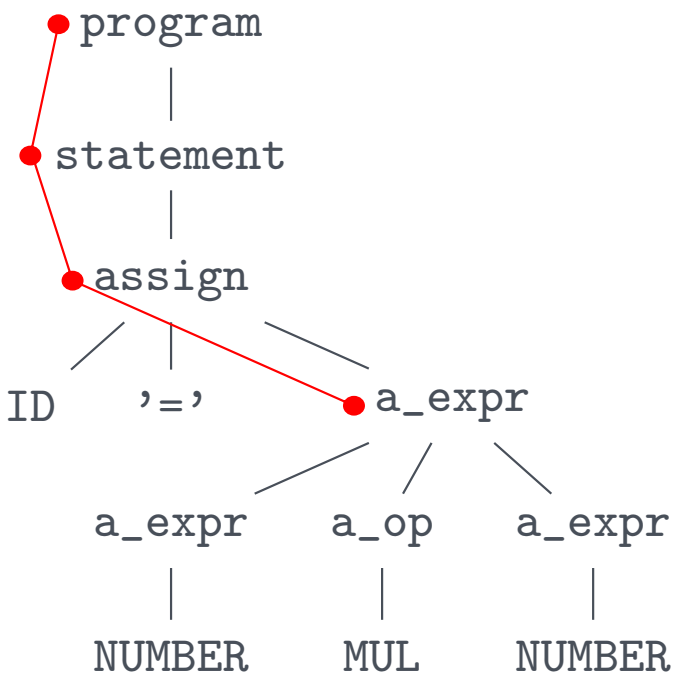
- Programma: `n = 100 * 2`
- Albero sintattico ed esecuzione del listener:




```
enterProgram(ProgramContext * ctx)
enterStatement(StatementContext * ctx)
enterAssign(AssignContext * ctx)
```

Esempio di Listener

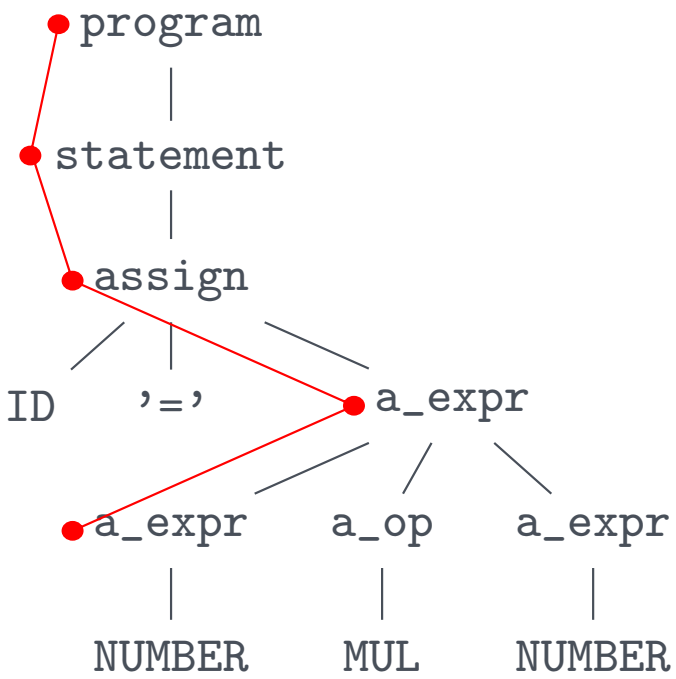
- Programma: `n = 100 * 2`
- Albero sintattico ed esecuzione del listener:




```
enterProgram(ProgramContext * ctx)
enterStatement(StatementContext * ctx)
enterAssign(AssignContext * ctx)
enterA_expr(A_exprContext * ctx)
```

Esempio di Listener

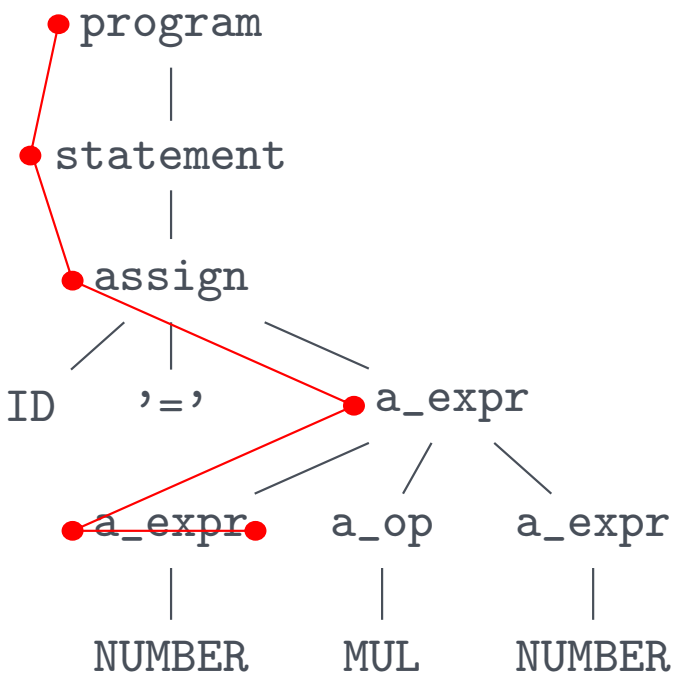
- Programma: `n = 100 * 2`
- Albero sintattico ed esecuzione del listener:




```
enterProgram(ProgramContext * ctx)
enterStatement(StatementContext * ctx)
enterAssign(AssignContext * ctx)
enterA_expr(A_exprContext * ctx)
enterA_expr(A_exprContext * ctx)
```

Esempio di Listener

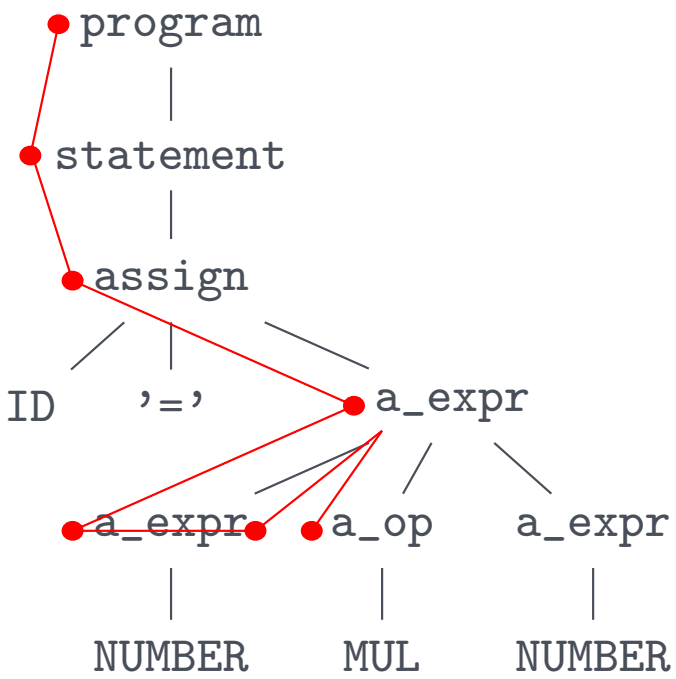
- Programma: `n = 100 * 2`
- Albero sintattico ed esecuzione del listener:




```
enterProgram(ProgramContext * ctx)
enterStatement(StatementContext * ctx)
enterAssign(AssignContext * ctx)
enterA_expr(A_exprContext * ctx)
enterA_expr(A_exprContext * ctx)
exitA_expr(A_exprContext * ctx)
```

Esempio di Listener

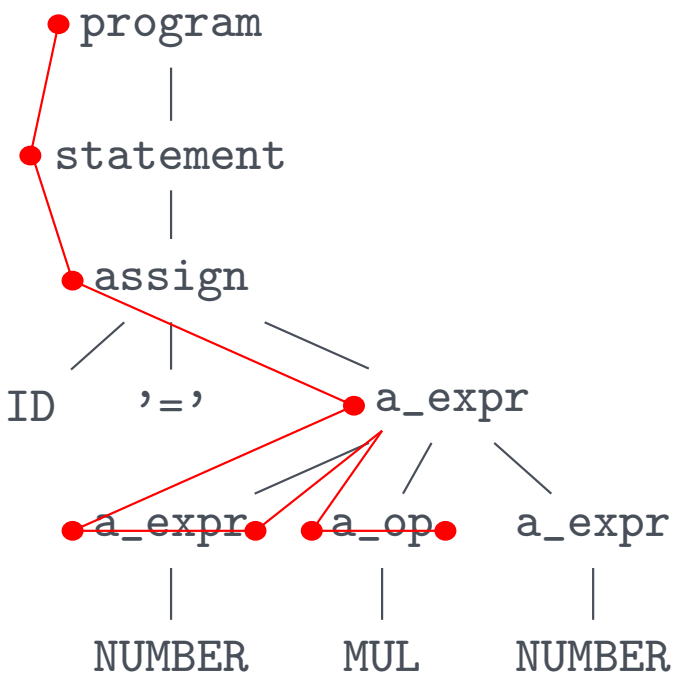
- Programma: `n = 100 * 2`
- Albero sintattico ed esecuzione del listener:




```
enterProgram(ProgramContext * ctx)
enterStatement(StatementContext * ctx)
enterAssign(AssignContext * ctx)
enterA_expr(A_exprContext * ctx)
enterA_expr(A_exprContext * ctx)
exitA_expr(A_exprContext * ctx)
enterA_op(A_opContext * ctx)
```

Esempio di Listener

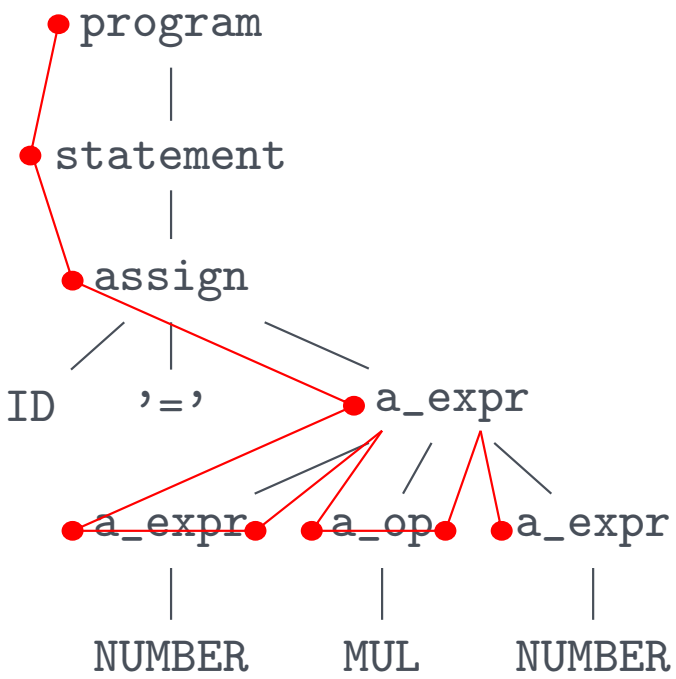
- Programma: `n = 100 * 2`
- Albero sintattico ed esecuzione del listener:




```
enterProgram(ProgramContext * ctx)
enterStatement(StatementContext * ctx)
enterAssign(AssignContext * ctx)
enterA_expr(A_exprContext * ctx)
enterA_expr(A_exprContext * ctx)
exitA_expr(A_exprContext * ctx)
enterA_op(A_opContext * ctx)
exitA_op(A_opContext * ctx)
```

Esempio di Listener

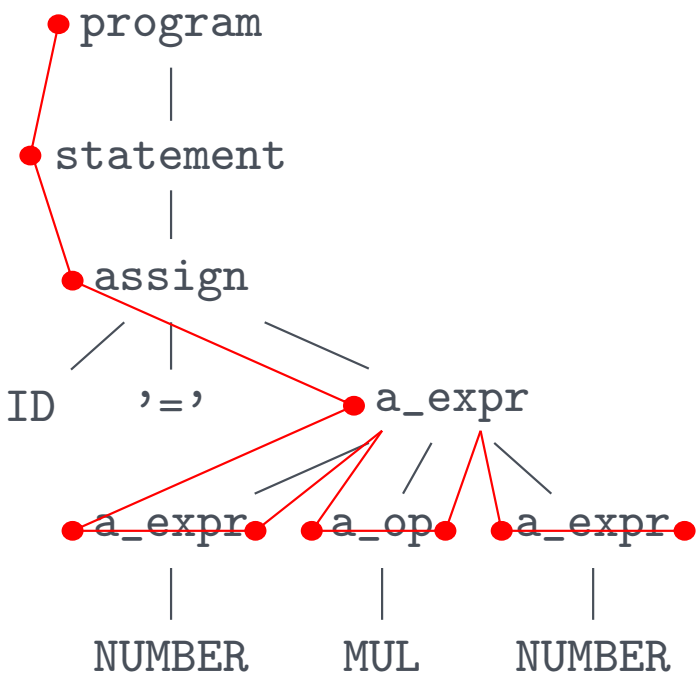
- Programma: `n = 100 * 2`
- Albero sintattico ed esecuzione del listener:




```
enterProgram(ProgramContext * ctx)
enterStatement(StatementContext * ctx)
enterAssign(AssignContext * ctx)
enterA_expr(A_exprContext * ctx)
enterA_expr(A_exprContext * ctx)
exitA_expr(A_exprContext * ctx)
enterA_op(A_opContext * ctx)
exitA_op(A_opContext * ctx)
enterA_expr(A_exprContext * ctx)
```

Esempio di Listener

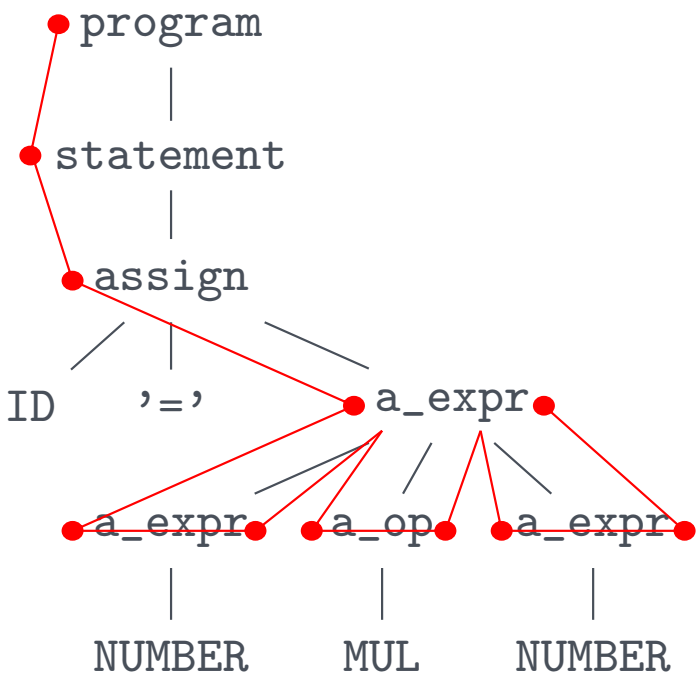
- Programma: `n = 100 * 2`
- Albero sintattico ed esecuzione del listener:




```
enterProgram(ProgramContext * ctx)
enterStatement(StatementContext * ctx)
enterAssign(AssignContext * ctx)
enterA_expr(A_exprContext * ctx)
enterA_expr(A_exprContext * ctx)
exitA_expr(A_exprContext * ctx)
enterA_op(A_opContext * ctx)
exitA_op(A_opContext * ctx)
enterA_expr(A_exprContext * ctx)
exitA_expr(A_exprContext * ctx)
```

Esempio di Listener

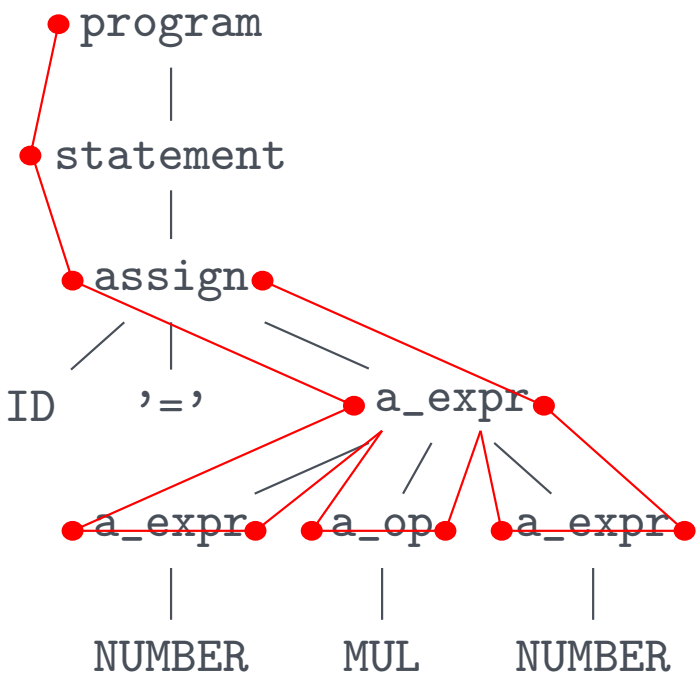
- Programma: `n = 100 * 2`
- Albero sintattico ed esecuzione del listener:




```
enterProgram(ProgramContext * ctx)
enterStatement(StatementContext * ctx)
enterAssign(AssignContext * ctx)
enterA_expr(A_exprContext * ctx)
enterA_expr(A_exprContext * ctx)
exitA_expr(A_exprContext * ctx)
enterA_op(A_opContext * ctx)
exitA_op(A_opContext * ctx)
enterA_expr(A_exprContext * ctx)
exitA_expr(A_exprContext * ctx)
exitA_expr(A_exprContext * ctx)
```

Esempio di Listener

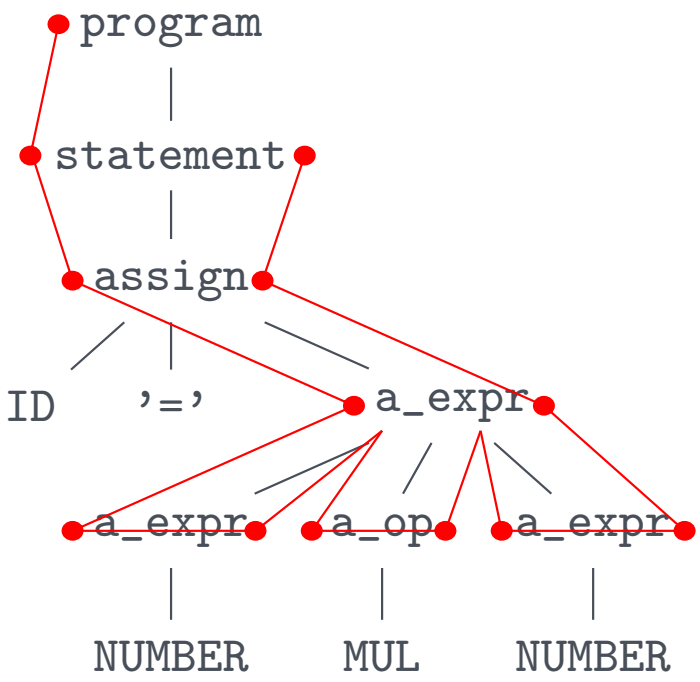
- Programma: `n = 100 * 2`
- Albero sintattico ed esecuzione del listener:




```
enterProgram(ProgramContext * ctx)
enterStatement(StatementContext * ctx)
enterAssign(AssignContext * ctx)
enterA_expr(A_exprContext * ctx)
enterA_expr(A_exprContext * ctx)
exitA_expr(A_exprContext * ctx)
enterA_op(A_opContext * ctx)
exitA_op(A_opContext * ctx)
enterA_expr(A_exprContext * ctx)
exitA_expr(A_exprContext * ctx)
exitA_expr(A_exprContext * ctx)
exitAssign(AssignContext * ctx)
```

Esempio di Listener

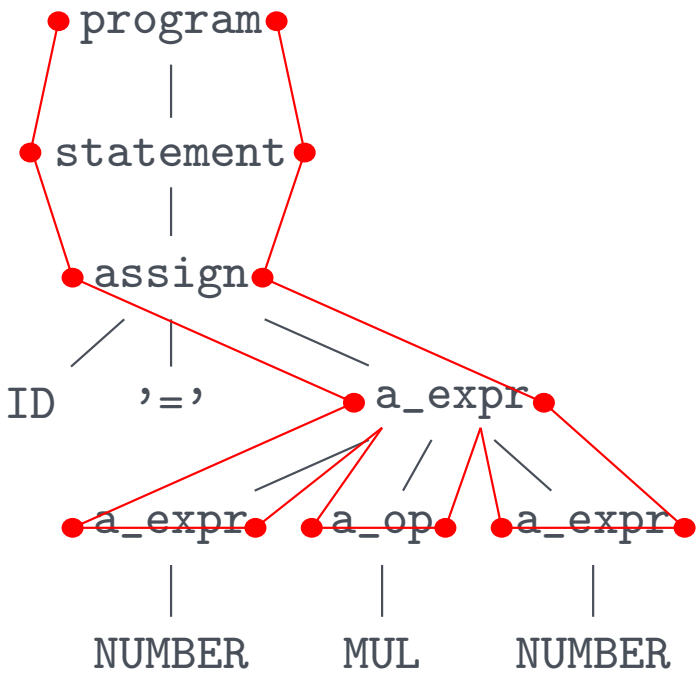
- Programma: `n = 100 * 2`
- Albero sintattico ed esecuzione del listener:




```
enterProgram(ProgramContext * ctx)
enterStatement(StatementContext * ctx)
enterAssign(AssignContext * ctx)
enterA_expr(A_exprContext * ctx)
enterA_expr(A_exprContext * ctx)
exitA_expr(A_exprContext * ctx)
enterA_op(A_opContext * ctx)
exitA_op(A_opContext * ctx)
enterA_expr(A_exprContext * ctx)
exitA_expr(A_exprContext * ctx)
exitA_expr(A_exprContext * ctx)
exitAssign(AssignContext * ctx)
exitStatement(StatementContext * ctx)
```

Esempio di Listener

- Programma: `n = 100 * 2`
- Albero sintattico ed esecuzione del listener:




```
enterProgram(ProgramContext * ctx)
enterStatement(StatementContext * ctx)
enterAssign(AssignContext * ctx)
enterA_expr(A_exprContext * ctx)
enterA_expr(A_exprContext * ctx)
exitA_expr(A_exprContext * ctx)
enterA_op(A_opContext * ctx)
exitA_op(A_opContext * ctx)
enterA_expr(A_exprContext * ctx)
exitA_expr(A_exprContext * ctx)
exitA_expr(A_exprContext * ctx)
exitAssign(AssignContext * ctx)
exitStatement(StatementContext * ctx)
exitProgram(ProgramContext * ctx)
```

Scrivere il codice del Listener

- `tinyrexxBaseListener` definisce dei metodi virtuali che non fanno nulla
- i file `MyListener.h` e `MyListener.cpp` istanziano i metodi
- definiscono una classe `MyListener` derivata da `tinyrexxBaseListener`
- definiscono il codice dei metodi che ci servono (non tutti)
- la classe `MyListener` possiede due attributi privati:
  - `indent` che contiene il numero di caratteri di indentazione
  - `vars` che contiene i nomi delle variabili usate dal programma

Esercizio

1. Primo gruppo di tre: avete 30 minuti per risolvere gli esercizi scritti nel foglio che vi verrà consegnato.
   - scrivete la soluzione sul foglio

2. Secondo gruppo di tre: raggruppati per il numero che c'è sul foglio, avete 15 minuti per spiegare la vostra soluzione agli altri membri del gruppo