D lang
Syntax Analyzer

Gleb Popov, Timur Usmanov
Compiler Construction
Innopolis University
Fall 2025

Project Context

Dynamic Lang

object types are not specified and can change while program execution

the language assumes **interpretation**

C++ Language

the implementation language is C++

it provides extensive memory management and optimization features

Personal parser

hand-written parser

if you want a thing done well, do it yourself :)

Recall: Lexer

```
var x := 5
print x
```

```
tkVar tkIdent(”x”) tkAssign tkIntLiteral(5) tkNewLine
tkPrint tkIdent tkNewLine
```

```
var t := {x:=1}
t := t + {y:=2}
```

```
tkVar tkIdent(t) tkAssign tkOpenCurlyBrace tkIdent(“x”)
tkAssign tkIntLiteral(1) tkClosedCurlyBrace tkNewLine
tkIdent(“t”) tkAssign tkIdent(“t”) tkPlus tkOpenCurlyBrace
tkIdent(“y”) tkAssign tkIntLiteral(2) tkClosedCurlyBrace
```

```
var x := 3
if x < 10 then
    print "small"
else
    print "big"
end
```

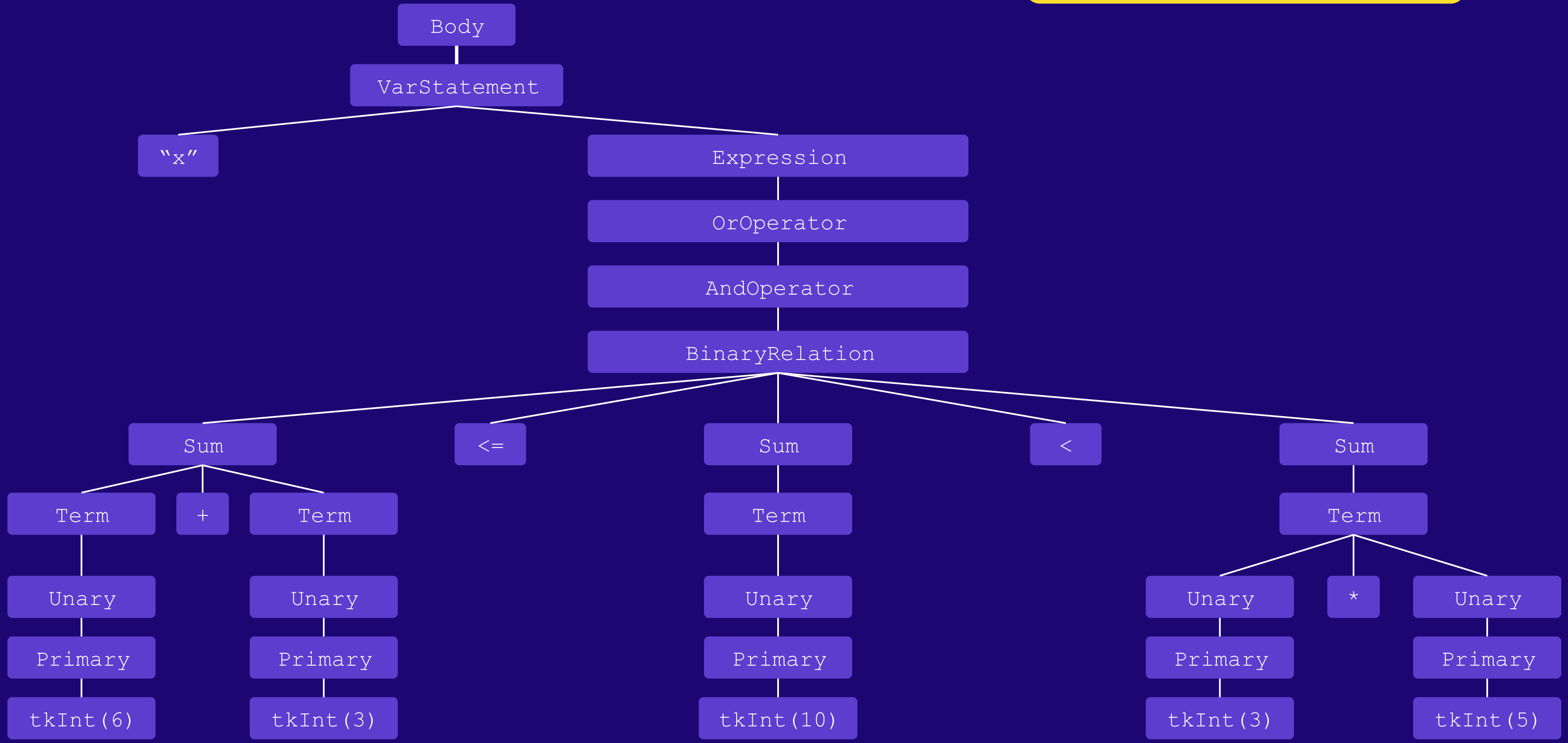
```
tkVar tkIdent(“x”) tkAssign tkIntLiteral(3) tkNewLine tkIf
tkIdent(“x”) tkLess tkIntLiteral(10) tkThen tkNewLine
tkPrint tkStringLiteral(“small”) tkNewLine tkElse tkNewLine
tkPrint tkStringLiteral(“big”) tkNewLine tkEnd tkNewLine
```

Syntax Analyzer: Output

```
var x := 6 + 3 <= 10 < 3 * 5
```

Syntax Analyzer: Output

```
var x := 6 + 3 <= 10 < 3 * 5
```

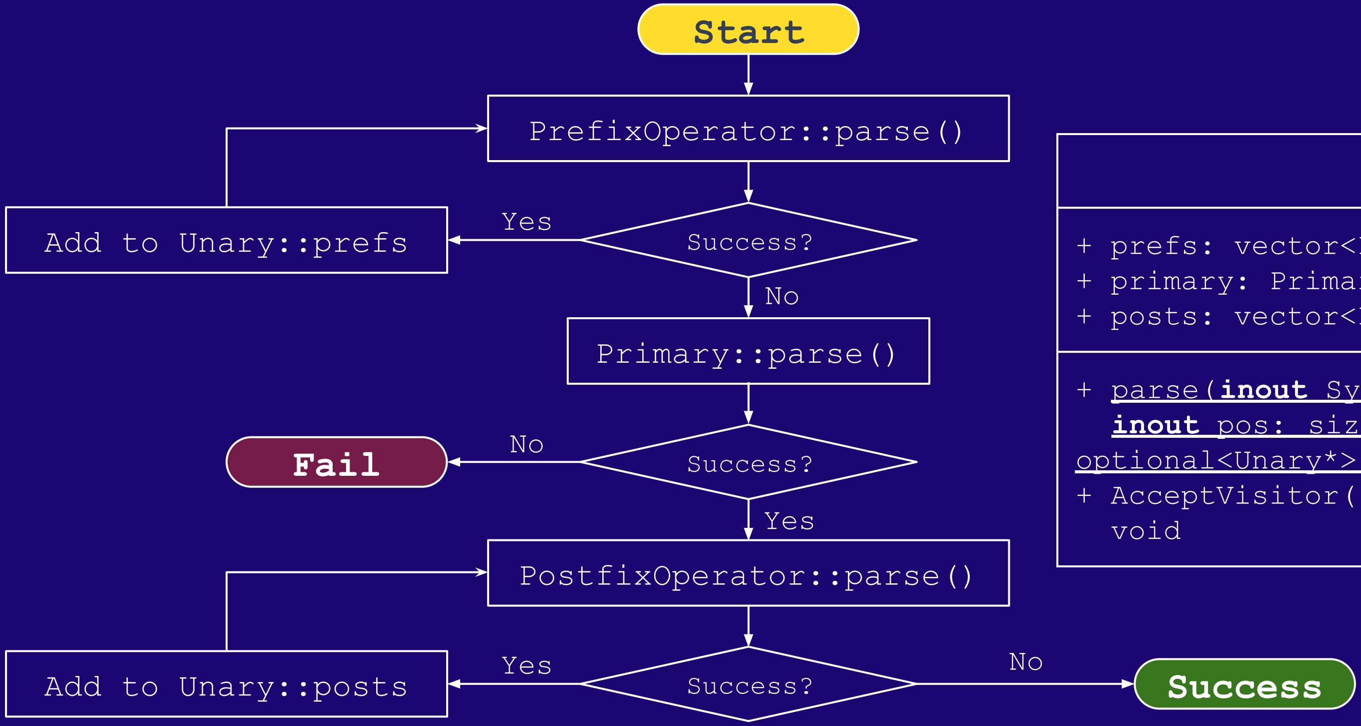


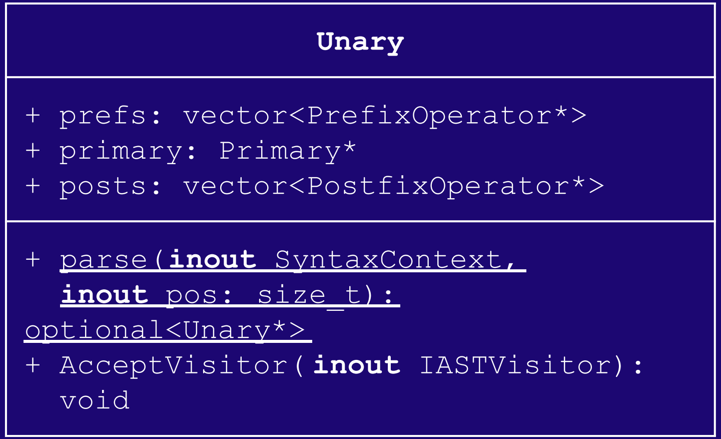
Syntax Analyzer: Implementation

- We have a class for every syntactic structure
- Every class has a *static* method `parse(SyntaxContext&, size_t& pos)`
- On **success**, `parse` returns the parsed syntax node *and* advances the token pointer
- On **failure**, `parse` returns nothing and logs a diagnostic message
- One class's `parse` may invoke other classes' `parse` for *nested structures*
- `parseProgram(...)` is the top method that parses the whole file through other `parse`s

Syntax Analyzer: Implementation (example)

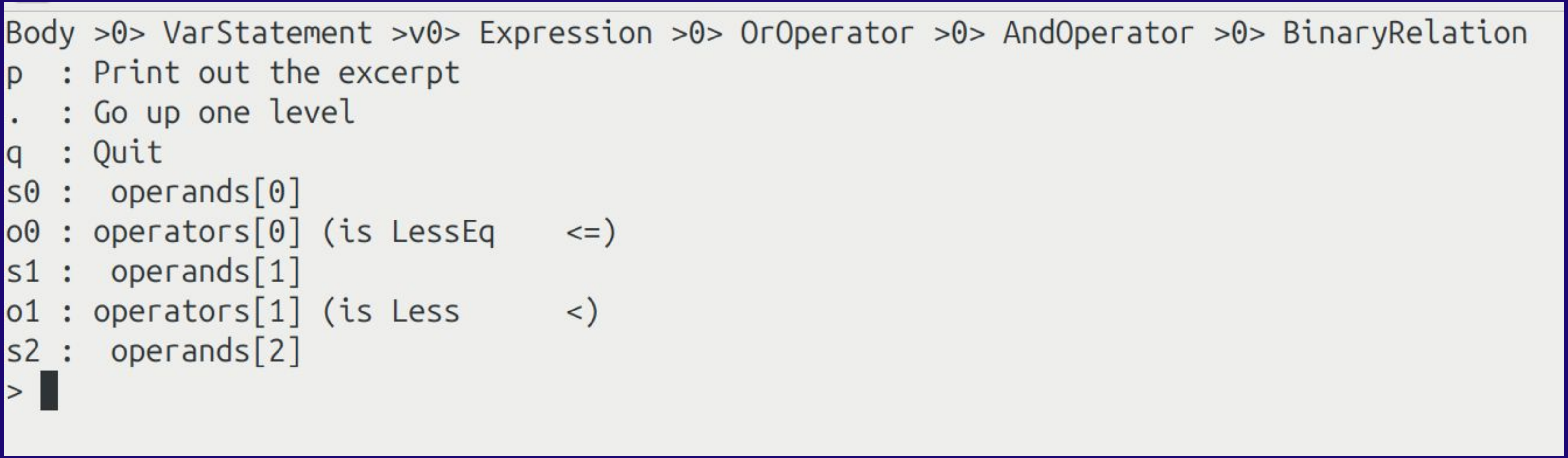
Unary -> {PrefixOperator} Primary {PostfixOperator}





Unary
+ prefs: vector<PrefixOperator*> + primary: Primary* + posts: vector<PostfixOperator*>
+ parse(**inout** SyntaxContext, **inout** pos: size_t): optional<Unary*> + AcceptVisitor(**inout** IASTVisitor): void

Syntax Analyzer: CLI



```
Body >0> VarStatement >v0> Expression >0> OrOperator >0> AndOperator >0> BinaryRelation
p  : Print out the excerpt
.  : Go up one level
q  : Quit
s0 :  operands[0]
o0 : operators[0] (is LessEq    <=)
s1 :  operands[1]
o1 : operators[1] (is Less      <)
s2 :  operands[2]
> 
```