D lang Lexer

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 :)

Tokens

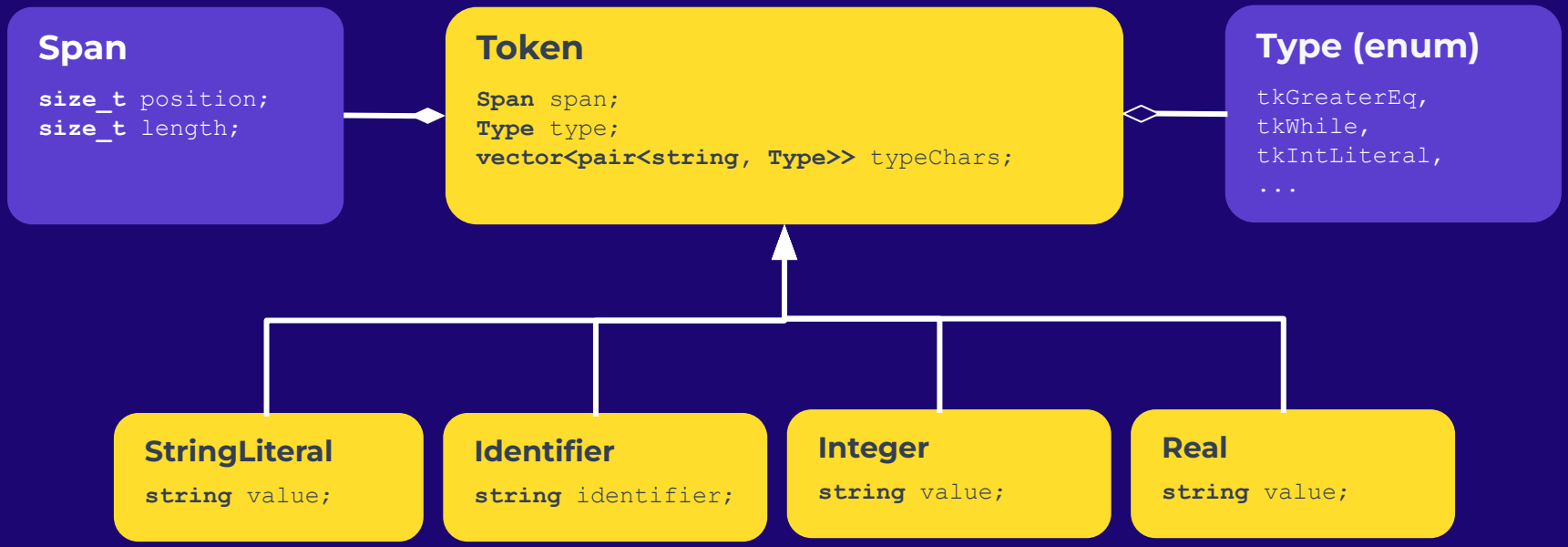
Token

```
Span span;
Type type;
vector<pair<string, Type>> typeChars;
```

Tokens


Span
size_t position;
size_t length;
Token
Span span;
Type type;
vector<pair<string, Type>> typeChars;
Type (enum)
tkGreaterEq,
tkWhile,
tkIntLiteral,
...

Tokens


Span
size_t position;
size_t length;
Token
Span span;
Type type;
vector<pair<string, Type>> typeChars;
Type (enum)
tkGreaterEq,
tkWhile,
tkIntLiteral,
...
StringLiteral
string value;
Identifier
string identifier;
Integer
string value;
Real
string value;

Lexer

Lexer

```
vector<Token*> tokenize(string& code);

checkComments();              // smth
checkStringLiterals();        "smth"
checkNumbers();               12.34
checkToken();                 int
checkIdentifier();            x
```

Lexer

Lexer

```
vector<Token*> tokenize(string& code);

checkComments();              // smth
checkStringLiterals();        "smth"
checkNumbers();               12.34
checkToken();                 int
checkIdentifier();            x
```

Longest Token First

```
"int"  over  "in"
".."   over  "."
"=>"   over  "="
">="   over  ">"
"<="   over  "<"
"/="   over  "/"
```

Lexer

Lexer

```
vector<Token*> tokenize(string& code);

checkComments();              // smth
checkStringLiterals();        "smth"
checkNumbers();               12.34
checkToken();                 int
checkIdentifier();            x
```

Longest Token First

```
"int"  over  "in"
".."   over  "."
"=>"   over  "="
">="   over  ">"
"<="   over  "<"
"/="   over  "/"
```

Identifier Restrictions

- only latin letters, digits, or underscores
- does not start with digit

Lexer

Lexer

```
vector<Token*> tokenize(string& code);

checkComments();              // smth
checkStringLiterals();        "smth"
checkNumbers();               12.34
checkToken();                 int
checkIdentifier();            x
```

Longest Token First

```
"int" over  "in"
".." over   "."
"=>" over   "="
">=" over   ">"
"<=" over   "<"
"/=" over   "/"
```

Identifier Restrictions

- only latin letters, digits, or underscores
- does not start with digit

No NewLine in strings

'\n' is **not** allowed
"\n" is allowed

Examples

```
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
```