# CodeZilla

**Team - 11**

**Amudhan Manisekaran – Kartik Mathpal – Mayank Rawat - Sandya Manoharan**

## Compiler and Virtual Machine for a Programming Language

## SER 502

**Arizona State University, Tempe, AZ, USA**

## Data Types:

- Integer
- Float
- String
- Boolean

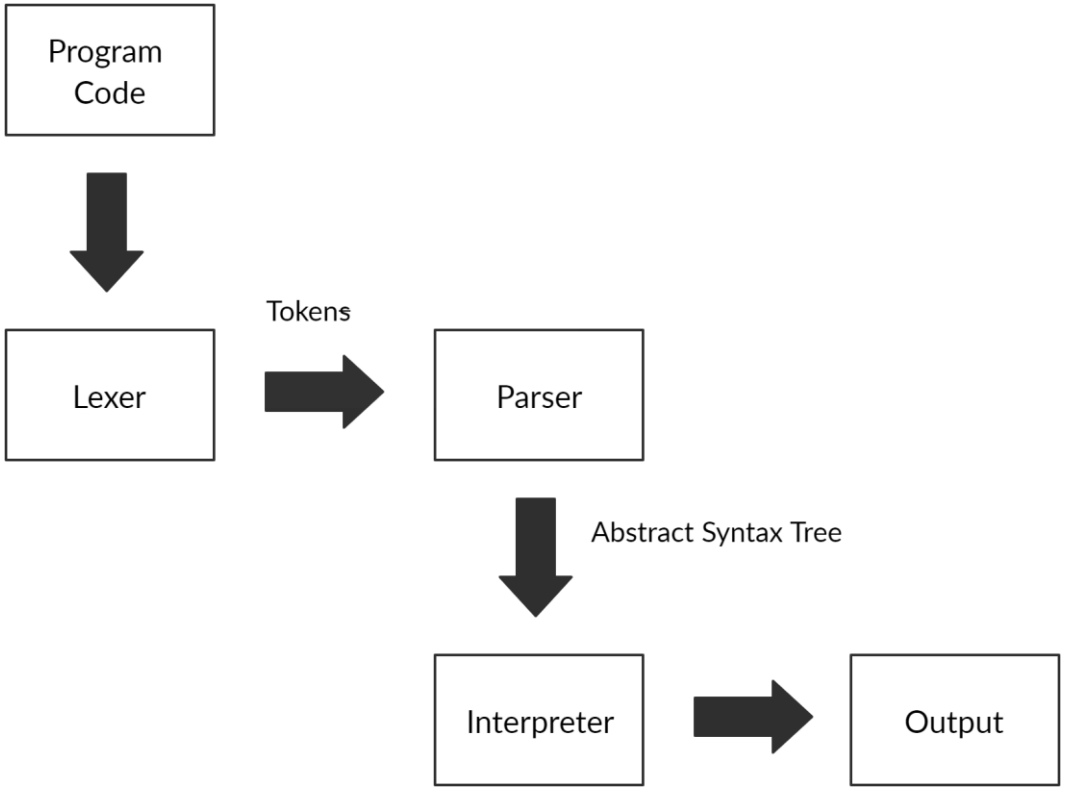
## Operations:

- Addition ( + )
- Subtraction ( - )
- Multiplication ( * )
- Division ( / )
- Comparison ( <, >, <=, >=, == )
- AND, OR, NOT (Boolean)
- Assignment ( = )

## Conditionals & Looping Structures:

- if then
- if then else
- Ternary ( $ )
- while loop
- Traditional for loop
- Range for loop
- read
- show

# Language Design



## Lexer

- In Python
- Read .cz
- Break down code
- Return .tok

## Parser

- In Prolog
- Read .tok
- Check syntax
- Return parse tree

## Interpreter

- In prolog
- Read .pt
- Check semantics
- Evaluate output

# Grammar Rules

```
PROGRAM          ::= start; BLOCK end;
BLOCK            ::= DECLARELIST; CONDITIONLIST;
DECLARELIST      ::= DECLARE; DECLARELIST | DECLARE
DECLARE          ::= var  ID | str  ID
CONDITIONLIST    ::= CONDITION; CONDITIONLIST | CONDITION
CONDITION        ::=  ASSIGN | IF | TERNARY | LOOPS | show DATA endshow| read ID endread
```

**Conditional Operations:**

```
ASSIGN    ::= ID = EXPSET | ID = STRING | ID = BOOL
IF        ::= if  BOOL  then  CONDITIONLIST  endif | if  BOOL  then  CONDITIONLIST
                else  CONDITIONLIST  endif
TERNARY   ::= ID = BOOL  $  EXPSET / EXPSET endternary
LOOPS     ::= while  BOOL  do  CONDITIONLIST  endwhile | for  (ID = VALUE ;  BOOL ; EXPSET):
                CONDITIONLIST endfor | for  ID  inrange (VALUE : VALUE):  CONDITIONLIST endfor
```

**Boolean Expressions:**

```
BOOL      ::= true | false | BOOL  and  BOOL | BOOL  or  BOOL | not  (BOOL) | BOOL LESSTHAN BOOL |
              BOOL GREATERTHAN BOOL | BOOL LESSTHANEQUAL BOOL |
              BOOL GREATERTHANEQUAL BOOL | BOOL EQUAL BOOL.
```

# Grammar Rules

**Arithmetic Expressions:**

```
EXPSET    ::= EXP | ID = EXPSET
EXP       ::= EXP + TERM | EXP – TERM | TERM
TERM      ::= TERM / FACTOR | TERM * FACTOR | FACTOR
FACTOR    ::= VALUE | ( EXPSET )
VALUE     ::= ID | INT | FLOAT
```

**Identifiers:**

```
ID        ::= ^ [a-z] $
INT       ::= ^ [0-9]+ $
FLOAT     ::= ^ [0-9]+ \ . [0-9] $
STRING    ::=  << ^[a-z] [a-zA-Z]+ $ >>
```

**Keywords:**

```
DATA                ::=  STRING | ID
LESSTHAN            ::=  <
LESSTHANEQUAL       ::=  <=
GREATERTHAN         ::=  <
GREATERTHANEQUAL    ::=  <=
EQUAL               ::=  ==
```

# CodeZilla (factorial.cz) file

factorial.cz

```
#program to calculate factorial
start;
var f;
var n;
var i;
f = 1;

read n endread;
for (i = 1; i <= n; i = i + 1): f = f * i endfor;

show << factorial is >> endshow;
show f endshow;

end;
```

# Lexer

C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 10.0.18363.778]
(c) 2019 Microsoft Corporation. All rights reserved.

D:\Uni\Spring 20-SER 502\project\SER502-Spring2020-Team11\src>python lexer.py

****************  CodeZilla - LEXER  ****************

Enter .cz File : factorial.cz

*************  tokens.tok file generated  *************


D:\Uni\Spring 20-SER 502\project\SER502-Spring2020-Team11\src>
```

## factorial.tok

[start, semicolon, var, f, semicolon, var, n, semicolon, var, i, semicolon, f, equal, 1, semicolon, read, n, endread, semicolon, for, open_para, i, equal, 1, semicolon, i, less_thanequal, n, semicolon, i, equal, i, +, 1, close_para, colon, f, equal, f, *, i, endfor, semicolon, show, less_than, less_than, factorial_is, greater_than, greater_than, endshow, semicolon, show, f, endshow, semicolon, end, semicolon].

# Parser

| | | | |
|---|---|---|---|
| compiler | 4/28/2020 12:34 PM | PL File | 4 KB |

SWI-Prolog -- d:/Uni/Spring 20-SER 502/project/SER502-Spring2020-Team11/src/compiler/co...

File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.0.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- codezilla('factorial.tok').
true .

?-
```

## factorial.pt

```
t_parser(t_program(t_block(t_declarationLINE(t_declarationVAR(t
_id(f)),t_declarationLINE(t_declarationVAR(t_id(n)),t_declarationL
INE(t_declarationVAR(t_id(i))))),t_commandLINE(t_command_assi
gn(t_assign_var(t_id(f),t_expr(t_fact(t_value(t_int(1))))))),t_comm
andLINE(t_command_read(t_read(t_id(n))),t_commandLINE(t_co
mmand_loops(t_loops(t_trad_for(t_id(i),t_value(t_int(1)),t_lessth
anequal(t_expr(t_fact(t_value(t_id(i)))),t_expr(t_fact(t_value(t_id(
n))))),t_assign(t_id(i),t_expr(t_add(t_fact(t_value(t_id(i))),t_fact(t_
value(t_int(1))))))),t_commandLINE(t_command_assign(t_assign_v
ar(t_id(f),t_expr(t_mul(t_fact(t_value(t_id(f))),t_fact(t_value(t_id(i
)))))))))),t_commandLINE(t_command_show(t_show(t_data(t_str(
factorial_is)))),t_commandLINE(t_command_show(t_show(t_data
(t_id(f)))))))))))).
```

# Interpreter

| | | | |
|---|---|---|---|
| interpreter | 4/28/2020 12:34 PM | PL File | 9 KB |

SWI-Prolog -- d:/Uni/Spring 20-SER 502/project/SER502-Spring2020-Team11/src/runtime/inter...

File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.0.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- codezilla('factorial.pt').

please enter value for:n
|: 5.

factorial_is
f = 120
true.

?-
```

## Changes in Grammar from Milestone 1 :

- ***For loop*** *(traditional and range) structure changed based on feedback from TA and Professor.*
- *Changed* ***bool*** *structure in grammar for easy access.*
- *Newly added* ***assignment*** *inside expressions,* ***boolean value assignment*** *to variables.*
- *Newly added* ***endshow***, ***endread***, ***endternary***, ***endfor*** *to identify termination.*
- ***Space*** *keyword in grammar* ***removed*** *since Lexer was made to check it inherently.*
- *Regular expression for* ***id*** *and* ***string*** *modified.*

# Experience

- *Hand-construct a lexical analyzer and parser using Definite Clause Grammars (DCG), to recognize and translate a language into an intermediate form.*

- *Understand and design of language runtime environments.*

- *Use regular and context-free language specifiers and recognizers.*

# Thanks!