SER 502- Languages and Programming Paradigm

bSharp Programming Language

Team 16:

Harika Kolli

Pradeep Ambalam Jawaharlal

Sneha Lakshminarasimhan

Arizona State University

Key Components

- Grammar
- Parse Tree
- Compiler
- Runtime
- Sample Programs

Language Features:

Simple Syntax

Loops using while

Datatypes: Double, Boolean

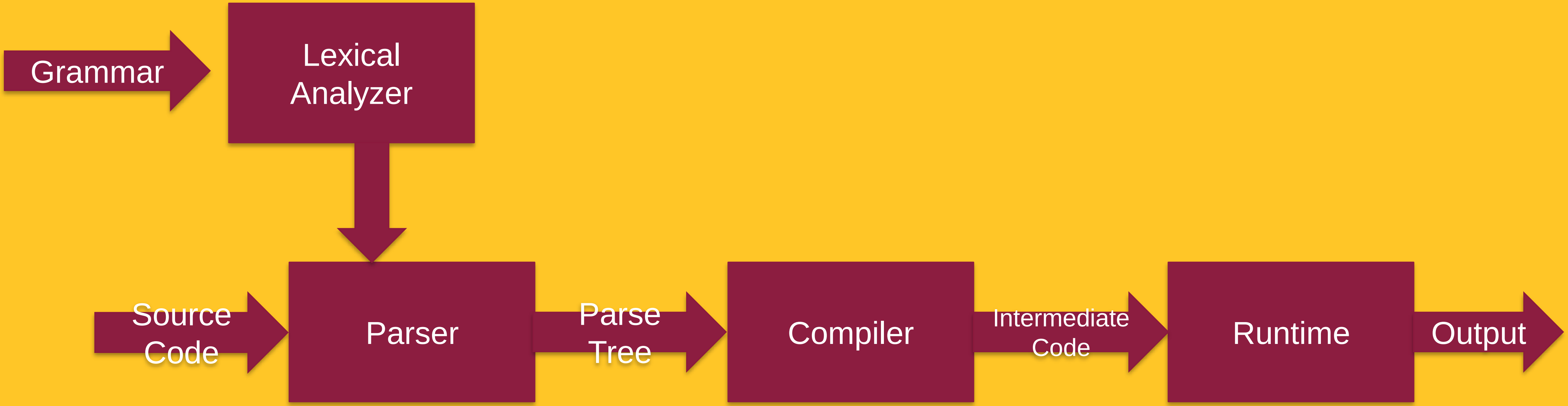
Comments

Operators: Arithmetic Logical Relational Boolean

Expressions evaluation with operator Precedence

Decisions based on If-else Nested If

Workflow:


Grammar
Lexical Analyzer
Source Code
Parser
Parse Tree
Compiler
Intermediate Code
Runtime
Output

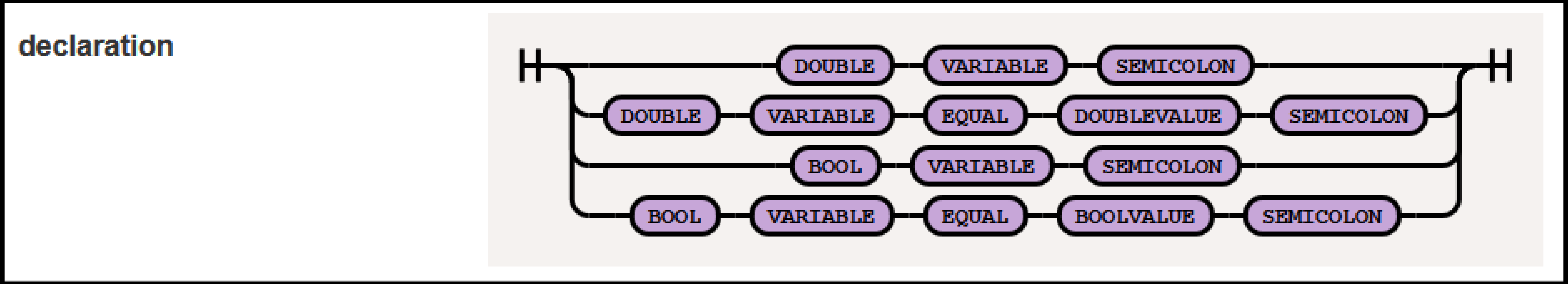
Grammar

- Datatypes
- Assignment
- Arithmetic Expressions
- Boolean Expressions
- Relational Expression
- Loops
- Miscellaneous

Declaration Block

Grammar Rule and Flow

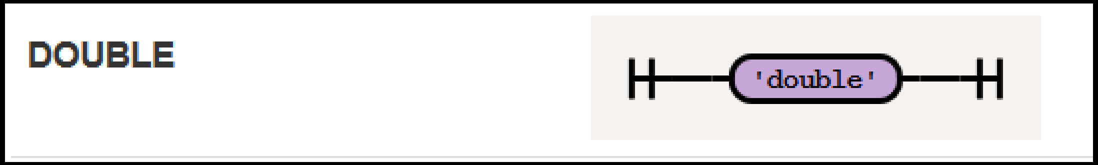
```
declaration          : DOUBLE VARIABLE SEMICOLON
                     | DOUBLE VARIABLE EQUAL DOUBLEVALUE SEMICOLON
                     | BOOL VARIABLE SEMICOLON
                     | BOOL VARIABLE EQUAL BOOLVALUE SEMICOLON;
```

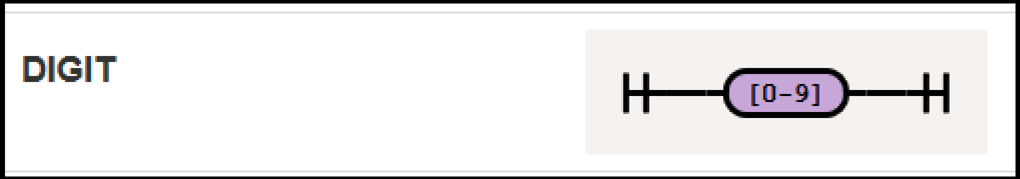


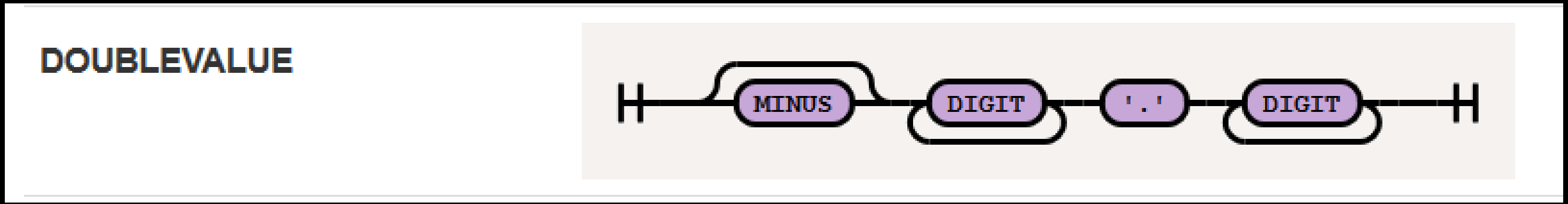
Datatypes - Double

Grammar Rule And Flow

```
DOUBLE              : 'double';
DOUBLEVALUE   : MINUS? DIGIT+ '.' DIGIT+ ;

fragment DIGIT         : [0-9];
fragment MINUS        : '-';
```







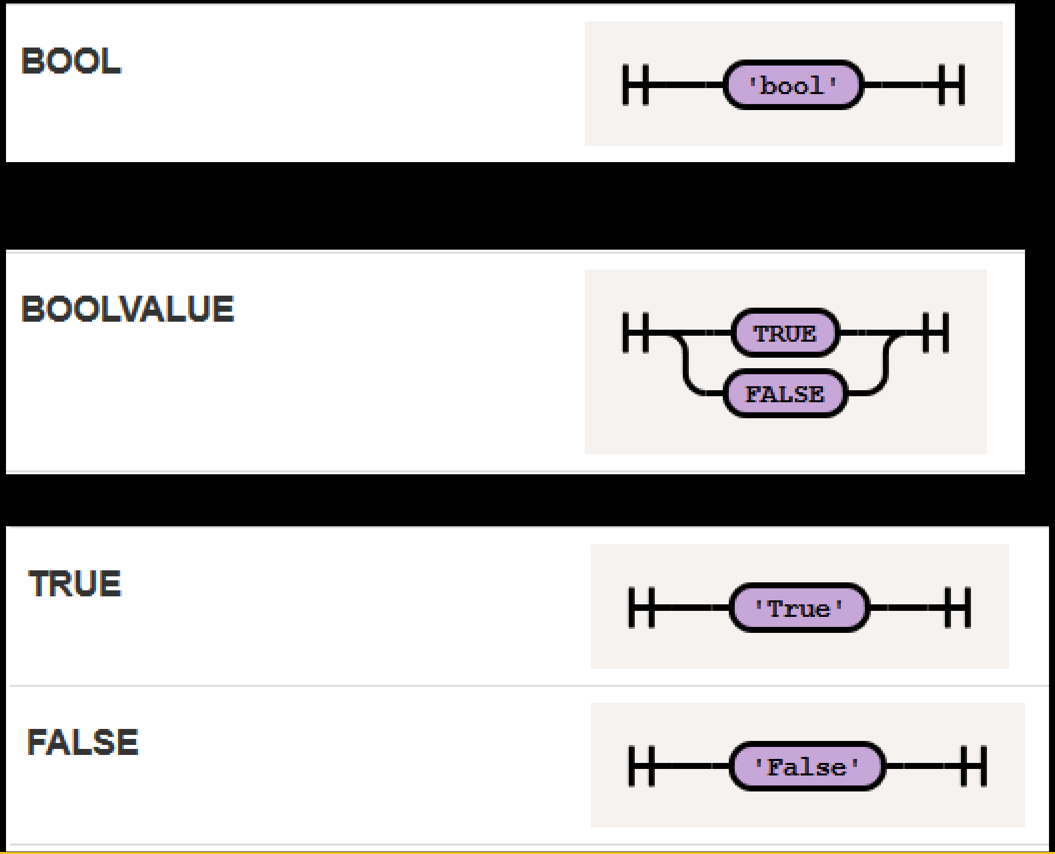
EXAMPLES:

double x;

Datatypes - Boolean

Grammar Rule and Flow

```
BOOL            : 'bool';
BOOLVALUE   : TRUE | FALSE ;

fragment TRUE           : 'True';
fragment FALSE          : 'False';
```



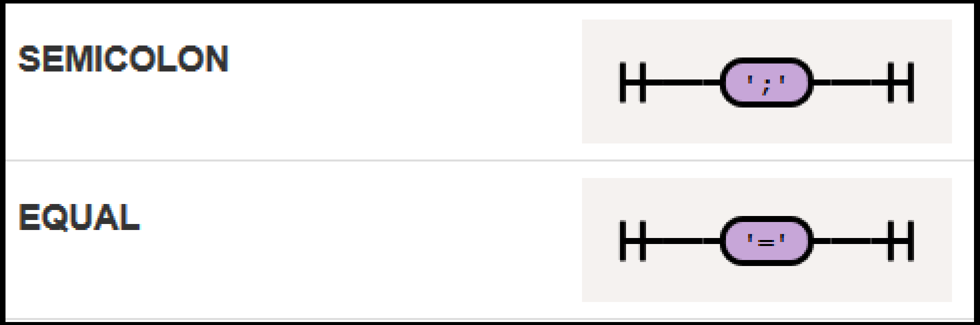
EXAMPLES:

```
bool case;
```

Assignment

Features

- Language uses ‘= ‘ for variable assignment
- Provides flexibility to the writer
- Has expressiveness and ensures Syntactic Sugar

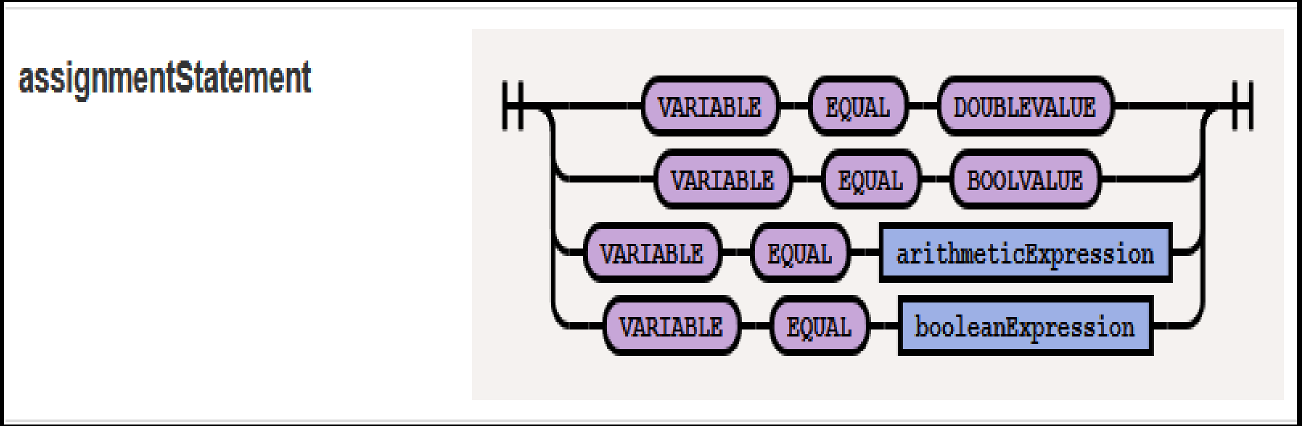


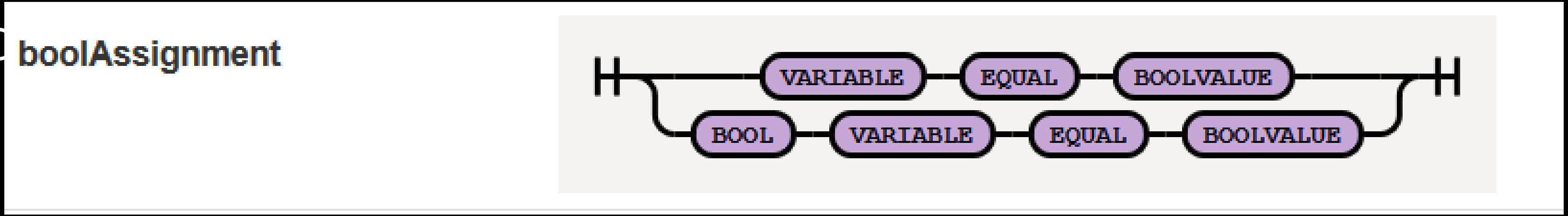
Assignment Examples

Grammar Rule and Flow

- assignmentStatement :
  VARIABLE EQUAL DOUBLEVALUE
  | VARIABLE EQUAL BOOLVALUE

- boolAssignment :
  VARIABLE EQUAL BOOLVALUE
  | BOOL VARIABLE EQUAL BOO





EXAMPLES:
double value = 10.09;
bool case = False;

Operators

Supported Operations:

- Arithmetic Operations: +, -, *, /
- Logical Operations: <, >, <=, >=, ==, !=
- Boolean Operations: &&, ||

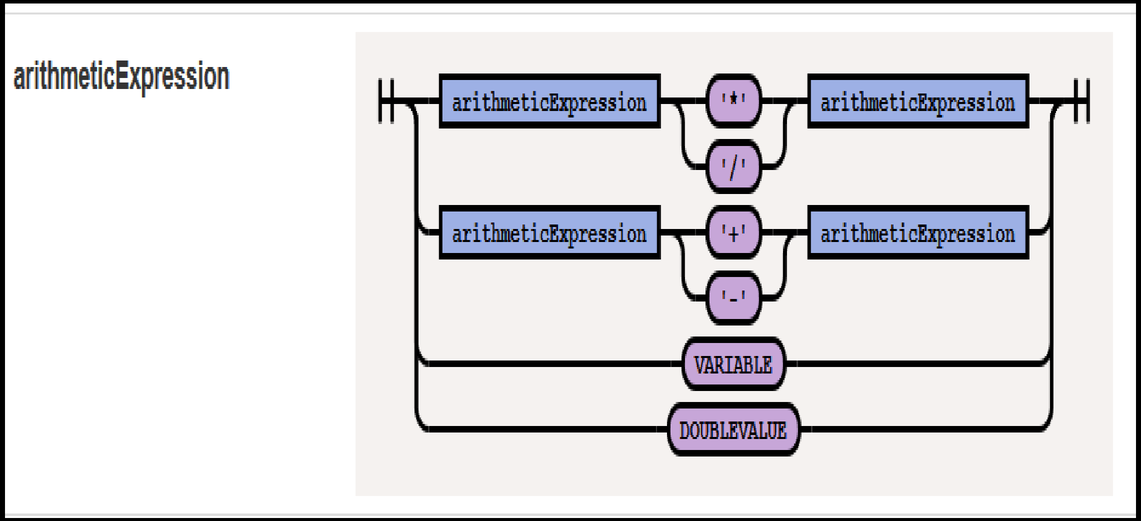
Operator Precedence:

- Multiply and Divide have greater precedence over addition and subtraction.
- Operator on the left is given greater priority over the one on the right, when it encounters same priority operator (* and /, + and -)

Arithmetic Expressions

Grammar Rule and Flow

```
arithmeticExpression :
        left=arithmeticExpression op=('*' | '/' )
        right=arithmeticExpression
       | left=arithmeticExpression op=('+' | '-' )
         right=arithmeticExpression
       | VARIABLE
       | DOUBLEVALUE;
```



Examples:

Val = 45.0*2.8-9.0/8.3;

Val = a/b-c*d+e

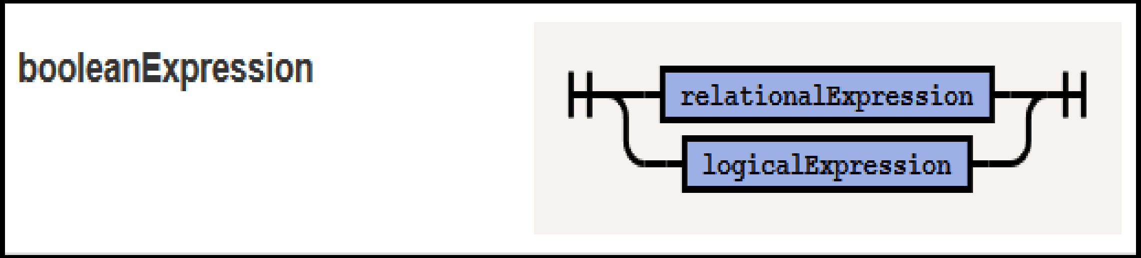
Boolean Expressions

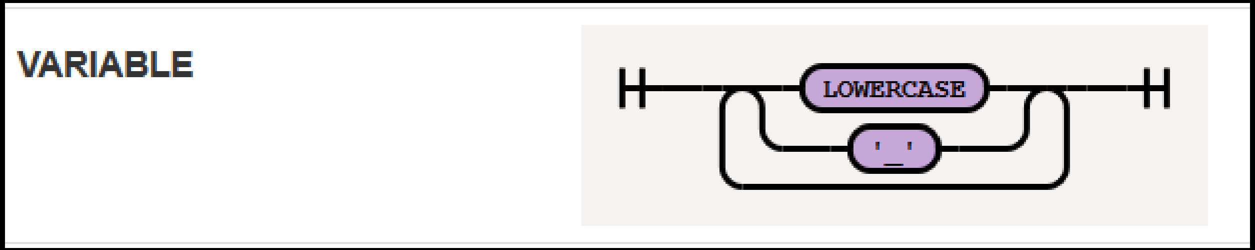
Grammar Rule

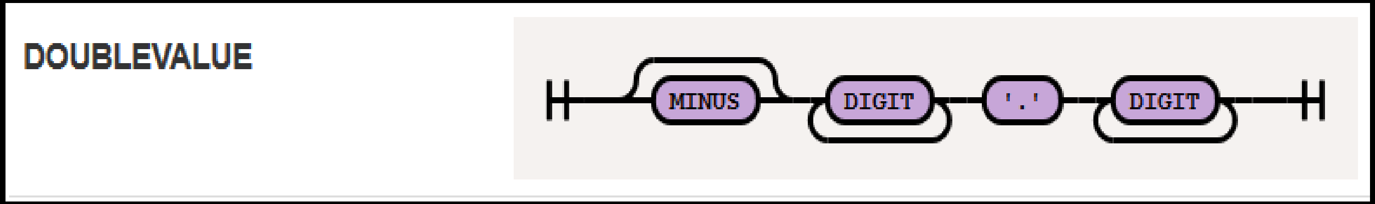
booleanExpression : relationalExpression | logicalExpression;

logicalExpression : arithmeticExpression
| left=logicalExpression op=logicalOperator right=logicalExpression
| VARIABLE | BOOLVALUE;

relationalExpression : VARIABLE | DOUBLEVALUE
| arithmeticExpression
| left=relationalExpression op=relationalOperator right=relationalExpression;



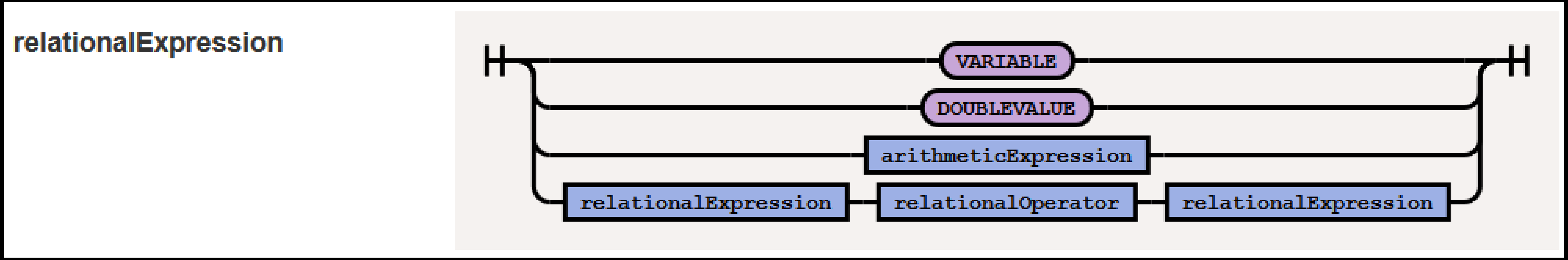




Examples:

result1 = a >b;
result2 = a !=b;
result3 = a<= 5.0;
result4 = a==b;
result5 = a && b;
result6 = a != True;

Relational Expressions



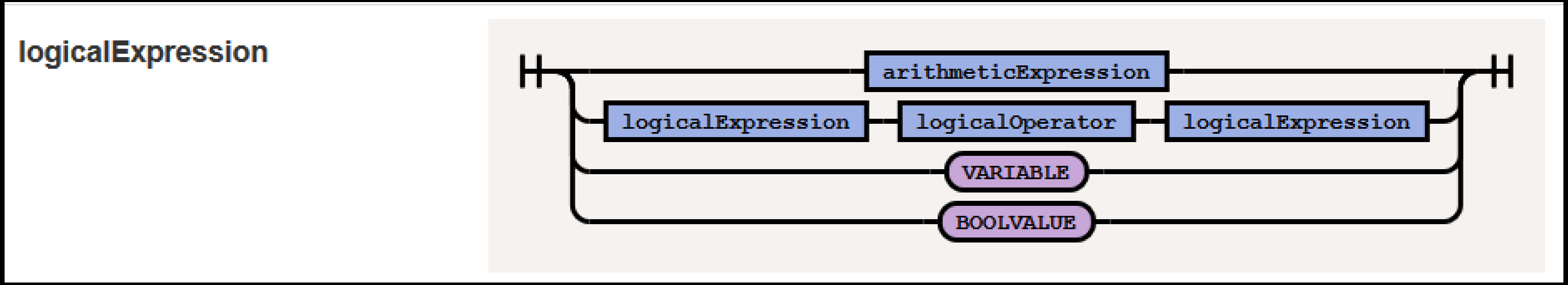
Example

```
double a;
number = 10.0;
a = number + 3.66;
result = a >= number;
```

Grammar Rule

```
relationalExpression   : VARIABLE
                       | DOUBLEVALUE
                       | arithmeticExpression
                       | left=relationalExpression
op=relationalOperator
right=relationalExpression;
```

Logical Expressions



Example

```
double a;
number = 10.0;
a = number + 3.66;
result = a >= number;
```

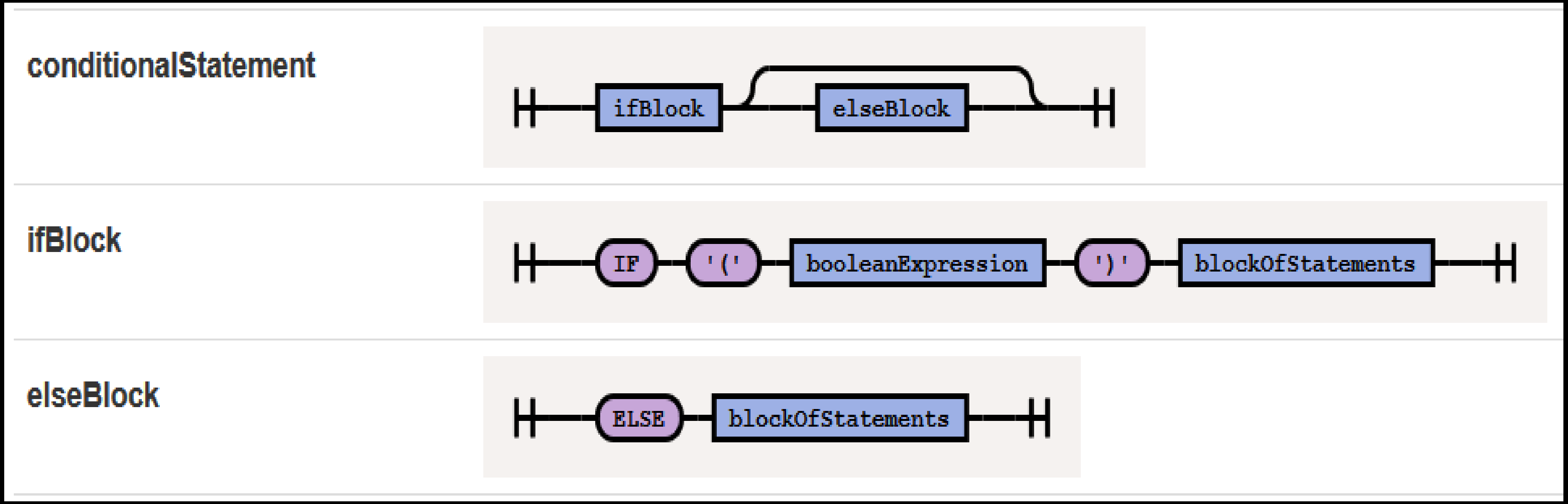
Grammar Rule

```
logicalExpression  : arithmeticExpression
                   | left=logicalExpression
op=logicalOperator right=logicalExpression
                   | VARIABLE
                   | BOOLVALUE;
```

Decisions

- Supports If-Else decision statements.
- Also supports Nested-If conditions.
- If the ‘if’ condition is met, the block of code inside the ‘if’ block is executed.
- When it is not, the ‘else’ block is executed.
- For multiple ‘if’, it sequentially executes one if after the other, evaluating them if the corresponding order.

If-Else Block Grammar Flow



If-Else Block Examples

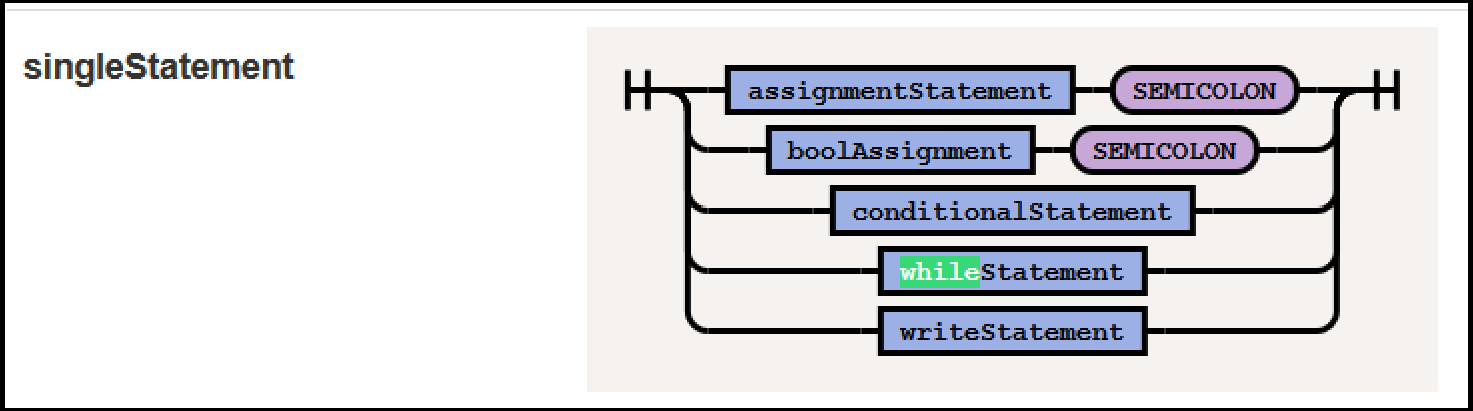
Examples

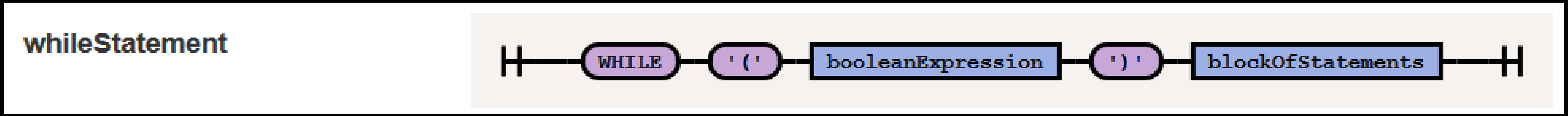
```
double x = 10.0;
double y = 12.0;
double z = 15.0;
if(x < y) {
        if (y < z) {
                write("z is the greatest");
        } else {
                write("y is the greatest");
        }
}
else {
        if (x < z) {
                write("z is the greatest");
        } else {
         write("x is the greatest");
         }
 }
```

Grammar Rule

```
conditionalStatement : ifBlock (elseBlock)?;

ifBlock   : IF '(' booleanExpression ')'
trueBlock=blockOfStatements;

elseBlock  : ELSE
falseBlock=blockOfStatements;
```

While Loop Grammar Flow





While Loop Example

Examples:

```
while (x >y){
     .....
}

while (True){
     .....
}
```

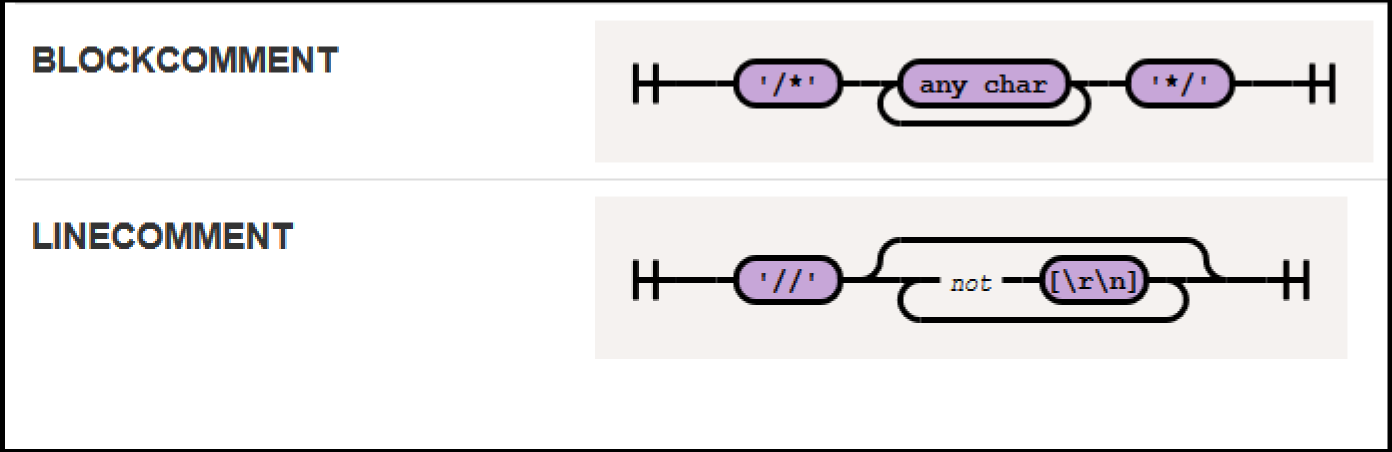
Grammar Rule:

```
whileStatement     :
WHILE '(' booleanExpression ')'
blockOfStatements;

booleanExpression :
     relationalExpression
   | logicalExpression;

WHILE      : 'while';
```

Comments

- bSharp language supports Comments of ‘/* */’ and ‘//’ form.

Grammar Flow:



Comments

Example

```
write("I'm being printed");
//write("Do not print");
write("I am getting printed, too");
write("Print Me");
/* double a = 10.78;
double b = 2.333;
double c;
c = a+b;
*/
print(c);
```

Grammar Rule

```
BLOCKCOMMENT
    :   '/*' .*? '*/'
        -> skip
    ;

LINECOMMENT
    :   '//' ~[\r\n]*
        -> skip
    ;
```

Thank you!

Try bSharp!