## Exercises

Translate the following Z#-Code to IL-Code.

```
a = 1;
b = 23;

while (b > a) {
  b = b - c;
}
```

Assume:
- a is a local variable (adr: 0)
- b is a method argument (adr: 5)
- c is a global variable
- a, b and c have type int

# Exercises

Are the following grammars regular?

```
N = c { c | d }.
```

```
X = a | b Y.
Y = c.
```

```
X = a | b Y c.
Y = d X.
```

# Exercises

Attribute the following grammar such that the result of Number will be the represented number.

```
Number<out int x> =
  ( '#' 'b' BinNumber
  | DecNumber
  )
.

BinNumber =
  ( '0' | '1' ) { '0' | '1' }
.

DecNumber =
  DecDigit { DecDigit }
.

DecDigit =
  '0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9'
.
```

## Exercises

Write syntax analyzing methods for the following grammar.

```
Log =
  'BEGIN' { LogEntry } 'END'
.

LogEntry =
  [ Severity ] ( string | number)
.

Severity =
  'WARNING' | 'ERROR'
.
```

## Exercises

Calculate the first an follow sets of the nonterminal symbols of the following grammar. Is the grammar LL1? If no, can it be made LL1? If yes do so, otherwise explain why not.

```
Config =
  ConfEntries
.

ConfEntries =
  ConfEntry ';'
  | ConfEntries ConfEntry ';'
.

ConfEntry =
  ( 'set' ident number
  | 'set' 'default' number
  | '#' string
  )
.
```

# Exercises

Translate the following Z#-Code to IL-Code.

```
a = 1;
b = 23;

while (b > a) {
  b = b - c;
}
```

Assume:
- a is a local variable (adr: 0)
- b is a method argument (adr: 5)
- c is a global variable
- a, b and c have type int

```
ldc.i4.1
stloc.0
ldc.i4 23
starg.s 5

while:
ldarg.s 5
ldloc.0
ble end;
ldarg.s 5
ldsfld Tc
sub
starg.s 5
br while;
end:
```

## Exercises

Are the following grammars regular?

```
N = c { c | d }.
```

```
X = a | b Y.
Y = c.
```

```
X = a | b Y c.
Y = d X.
```

Yes
Yes
No (central recursion)

# Exercises

Attribute the following grammar such that the result of Number
will be the represented number.

```
Number<out int x> =
  ( '#' 'b' BinNumber
  | DecNumber
  )
.

BinNumber =
  ( '0' | '1' ) { '0' | '1' }
.

DecNumber =
  DecDigit { DecDigit }
.

DecDigit =
  '0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9'
.
```

```
Number<out int x> =
  ( '#' 'b' BinNumber<out x>
  | DecNumber<out x>
  )
.

BinNumber<out int x>    (. x = 0; .)
=
  ( '0' | '1'           (. x = 1; .)
  )
  {                     (. x = x * 2; .)
    ( '0'
    | '1'               (. x = x + 1; .)
    )
  }
.

DecNumber<out int x> =
  DecDigit<out x>
  { DecDigit<out y>     (. x = x * 10 + y; .)
  }
.

DecDigit<out int x> =
  ('0'|'1'|'2'|'3'|'4'
  |'5'|'6'|'7'|'8'|'9') (. x = t.val;/* pseudo code but ok */ .)
.
```

## Exercises

Write syntax analyzing methods for the following grammar.

```
Log =
  'BEGIN' { LogEntry } 'END'
.

LogEntry =
  [ Severity ] ( string | number)
.

Severity =
  'WARNING' | 'ERROR'
.
```

```
public void Log() {
  Check(Token.BEGIN);
  while (firstLogEntry[la]) { LogEntry(); }
  Check(Token.END);
}

public void LogEntry() {
  if (firstSeverity[la]) { Severity(); }
  if (la == Token.STRING) { scan(); }
  else if (la == Token.NUMBER) { scan(); }
  else { Error("Invalid log entry."); }
}

public void Severity() {
  if (la == Token.WARNING) { scan(); }
  else if (la == Token.ERROR) { scan(); }
  else { Error("Invalid Severity."); }
}
```

## Exercises

Calculate the first an follow sets of the nonterminal symbols of the following grammar. Is the grammar LL1? If no, can it be made LL1? If yes do so, otherwise explain why not.

```
Config =
  ConfEntries
.

ConfEntries =
  ConfEntry ';'
  | ConfEntries ConfEntry ';'
.

ConfEntry =
  ( 'set' ident number
  | 'set' 'default' number
  | '#' string
  )
.
```

First(Config) = First(ConfEntries)

First(ConfEntries) = First(ConfEntry)

First(ConfEntry) = 'set', '#'

Follow(Config) = eof

Follow(ConfEntries) = First(ConfEntriy) + eof

Follow(ConfEntry) = ';'