

# Programming Assignment 4

Leo Kivikunnas 525925, Jaakko Koskela 526050

March 2020

## 1 Augmented LL(1) Grammar

---

```
Program -> Declaration-list #END
Declaration-list -> Declaration Declaration-list EPSILON
Declaration -> Declaration-initial Declaration-prime
Declaration-initial -> Type-specifier #PID ID
Declaration-prime -> #FUNCTION Fun-declaration-prime #VARIABLE Var-declaration-prime
Var-declaration-prime -> #PARAM ; #ARRAY [ NUM ] ;
Fun-declaration-prime -> #BEGINSCOPE #START_PARAM_COUNTER Params #STOP_PARAM_COUNTER Compound-stmt
Type-specifier -> int void
Params -> int #PID #PARAM ID Param-prime Param-list void Param-list-void-abtar
Param-list-void-abtar -> #PID #PARAM ID Param-prime Param-list EPSILON
Param-list -> , Param Param-list EPSILON
Param -> Declaration-initial Param-prime
Param-prime -> #ARRAY #ARRAY_PARAM [ ] EPSILON
Compound-stmt -> { Declaration-list Statement-list #ENDSCOPE }
Statement-list -> Statement Statement-list EPSILON
Statement -> Expression-stmt #BEGINSCOPE Compound-stmt Selection-stmt Iteration-stmt Return-stmt Switch-stmt
Expression-stmt -> Expression ; #CONTINUE continue ; #BREAK break ; ;
Selection-stmt -> if Expression Statement else Statement
Iteration-stmt -> while #ENTER_WHILE Expression Statement #EXIT_WHILE
Return-stmt -> return Return-stmt-prime
Return-stmt-prime -> ; Expression ;
Switch-stmt -> switch #ENTER_SWITCH_CASE Expression { Case-stmts Default-stmt } #EXIT_SWITCH_CASE
Case-stmts -> Case-stmt Case-stmts EPSILON
Case-stmt -> case NUM : Statement-list
Default-stmt -> default : Statement-list EPSILON
Expression -> Simple-expression-zegond #USE_PID #START_ARGUMENT_COUNTER #TYPE_CHECK #START_TYPE_CHECK ID B
B -> = Expression #CANCEL_TYPE_CHECK [ Expression ] H Simple-expression-prime
H -> = Expression G D C
Simple-expression-zegond -> Additive-expression-zegond C
Simple-expression-prime -> Additive-expression-prime C
C -> Relop Additive-expression #CANCEL_TYPE_CHECK EPSILON
Relop -> < ==
Additive-expression -> Term D
Additive-expression-prime -> Term-prime D
Additive-expression-zegond -> Term-zegond D
D -> Addop Term D EPSILON
Addop -> + -
Term -> Factor G
Term-prime -> Factor-prime G
```

```

Term-zegond -> Factor-zegond G
G -> Factor -> Expression #USE_PID #START_ARGUMENT_COUNTER #TYPE_CHECK #START_TYPE_CHECK ID Var-call-prime #T
Var-call-prime -> #CANCEL_TYPE_CHECK Args #STOP_ARGUMENT_COUNTER Var-prime
Var-prime -> [ #CANCEL_TYPE_CHECK Expression ] EPSILON
Factor-prime -> #CANCEL_TYPE_CHECK Args #STOP_ARGUMENT_COUNTER EPSILON
Factor-zegond -> Expression #TYPE_CHECK NUM
Args -> Arg-list EPSILON
Arg-list -> #ARGUMENT Expression Arg-list-prime
Arg-list-prime -> , #ARGUMENT Expression Arg-list-prime EPSILON

```

---

## 2 Error Detection

Our program can detect all 8 error types listed in the assignment.