Abstract Grammar

Paula Suárez Prieto – UO269745

Diseño de Lenguajes de Programación, curso 2023-2024

Grupo PL-02

### CATEGORIES

sentence;

expression;

type;

### NODES

**program** -> name:string types:structDefinition\* vars:varDefinition\* builders:functionBuilder\* features:functionDefinition\* runCall:runCall ;

**runCall** -> name:string args:expression\* ;

**structDefinition** -> name:structType fields:fieldDefinition\* ;

**functionDefinition** -> name:string params:varDefinition\* returnType:type? vars:varDefinition\* sentences:sentence\* ;

**fieldDefinition** -> name:string tipo:type ;

**varDefinition** -> name:string tipo:type ;

**functionBuilder** -> name:string;

**functionCallSent**: sentence -> name:string args:expression\* ;

**assignment**: sentence -> left:expression right:expression ;

**loop**: sentence -> from:assignment\* until:expression body:sentence\* ;

**ifElse**: sentence -> condition:expression trueBlock:sentence\* falseBlock:sentence\* ;

**read**: sentence -> input:expression\* ;

**print**: sentence -> op:string input:expression\* ;

**return**: sentence -> value: expression? ;

**intConstant**: expression -> value:string;

**realConstant**: expression -> value:string;

**charConstant**: expression -> value:string;

**variable**: expression -> name:string;

**castExpr**: expression -> castType:type value:expression ;

**arithmeticExpr**: expression -> op1:expression operator:string op2:expression ;

**logicalExpr**: expression -> op1:expression operator:string op2:expression ;

**comparationExpr**: expression -> op1:expression operator:string op2:expression ;

**minusExpr**: expression -> op:expression ;

**notExpr**: expression -> op:expression ;

**functionCallExpr**: expression -> name:string args:expression\*;

**fieldAccess**: expression -> root:expression field:string ;

**arrayAccess**: expression -> array:expression index:expression ;

**intType**: type -> ;

**doubleType**: type -> ;

**charType**: type -> ;

**voidType**: type -> ;

**structType**: type -> name:string;

**arrayType**: type -> dimension:intConstant tipo:type;

**booleanType**: type -> ;

### ATTRIBUTE GRAMMAR Identification

varDefinition -> [inh] Scope;

variable -> definition:varDefinition;

functionDefinition -> isBuilder: boolean;

functionCallSent -> definition: functionDefinition;

functionCallExpr -> definition: functionDefinition;

runCall -> definition: functionDefinition;

structType -> definition: structDefinition;

fieldDefinition -> [inh] fieldOwner: structType;

### ATTRIBUTE GRAMMAR TypeChecking

functionDefinition -> hasReturn: boolean;

sentence -> [inh] owner: functionDefinition;

expression -> lvalue: boolean;

expression -> type;

### ATTRIBUTE GRAMMAR MemoryAllocation

fieldDefinition -> [inh] address: int ;

varDefinition -> [inh] address: int ;

### CODE SPECIFICATION Mapl

run[program]

generate[functionDefinition]

execute[runCall]

execute[sentence]

value[expression]

address[expression]

metadata[program]

metadata[varDefinition]

metadata[structDefinition]

metadata[fieldDefinition]

metadata[functionBuilder]