

Progetto Compilatori

A.A. 2020/2021

Gaetano Antonucci

Alessio Romano

21 gennaio 2021

Indice

1	Grammatica Utilizzata	2
2	Regole di Type Checking implementate	4
2.1	Tipi Primitivi	4
2.2	Dichiarazioni di Variabili	4
2.3	Operazioni Unarie	4
2.4	Operazioni Binarie	4
2.5	Chiamata a Procedura	4
2.6	Statement	4
2.6.1	if-then	4
2.6.2	if-then-else	4
2.6.3	if-then-elif-else	4
2.6.4	while	4
2.6.5	while-return	5
2.6.6	readln	5
2.6.7	write	5
2.6.8	simple-assign	5
2.6.9	multiple-assign	5
2.6.10	return	5
2.7	Tabelle di Compatibilità	5

1 Grammatica Utilizzata

```
Program := VarDeclList ProcList
VarDeclList := VarDecl VarDeclList
              | /* empty */
VarDecl := Type IdListInit SEMI
Type := INT
        | BOOL
        | FLOAT
        | STRING
IdListInit := ID
              | IdListInit COMMA ID
              | ID ASSIGN Expr
              | IdListInit COMMA ID ASSIGN Expr
ProcList := Proc
           | Proc ProcList
Proc := PROC ID LPAR ParamDeclList RPAR ResultTypeList COLON ProcBody
       | PROC ID LPAR RPAR ResultTypeList COLON ProcBody
ProcBody := VarDeclList StatList RETURN ReturnExprs CORP SEMI
           | VarDeclList RETURN ReturnExprs CORP SEMI
ParamDeclList := ParamDecl
               | ParamDeclList SEMI ParamDecl
ParamDecl := Type ParIdList
ParIdList := ID
            | ParIdList COMMA ID
ResultTypeList := ResultType
                | ResultType COMMA ResultTypeList
ResultType := Type
            | VOID
StatList := Stat SEMI
           | Stat SEMI StatList
Stat := IfStat
       | WhileStat
       | ReadlnStat
       | AssignStat
       | CallProc
IfStat := IF Expr THEN StatList ElifList Else FI
ElifList := Elif ElifList
           | /* empty */
```

```

Elif := ELIF Expr THEN StatList
Else := ELSE StatList
      | /* empty */
WhileStat := WHILE StatList RETURN Expr DO StatList OD
          | WHILE Expr DO StatList OD
ReadlnStat := READ LPAR IdList RPAR
IdList := ID
        | IdList COMMA ID
WriteStat := WRITE LPAR ExprList RPAR
AssignStat := IdList ASSIGN ExprList
CallProc := ID LPAR ExprList RPAR
          | ID LPAR RPAR
ReturnExprs := ExprList
            | /* empty */
ExprList := Expr
          | Expr COMMA ExprList
Expr := NULL
      | TRUE
      | FALSE
      | INT_CONST
      | FLOAT_CONST
      | STRING_CONST
      | ID
      | MINUS Expr
      | Expr PLUS Expr
      | Expr MINUS Expr
      | Expr TIMES Expr
      | Expr DIV Expr
      | NOT Expr
      | Expr AND Expr
      | Expr OR Expr
      | Expr GT Expr
      | Expr GE Expr
      | Expr LT Expr
      | Expr LE Expr
      | Expr EQ Expr
      | Expr NE Expr
      | CallProc

```

2 Regole di Type Checking implementate

2.1 Tipi Primitivi

$$\begin{array}{l} \Gamma \vdash \text{null} : \text{null} \quad \Gamma \vdash \text{true} : \text{boolean} \quad \Gamma \vdash \text{false} : \text{boolean} \\ \Gamma \vdash \text{int} : \text{int} \quad \Gamma \vdash \text{float} : \text{float} \quad \Gamma \vdash \text{string} : \text{string} \quad \Gamma \vdash \text{bool} : \text{boolean} \end{array}$$

2.2 Dichiarazioni di Variabili

$$\frac{(x : \tau) \in \Gamma}{\Gamma \vdash x : \tau}$$

2.3 Operazioni Unarie

$$\frac{\Gamma \vdash e : \tau_1 \quad \text{optype1}(op, \tau_1) = \tau}{\Gamma \vdash (op \ e) : \tau}$$

2.4 Operazioni Binarie

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2 \quad \text{optype2}(op, \tau_1, \tau_2) = \tau}{\Gamma \vdash (e_1 \ op \ e_2) : \tau}$$

2.5 Chiamata a Procedura

$$\frac{\Gamma \vdash f : \tau_i^{i \in 1 \dots n} \rightarrow \tau_j^{j \in 1 \dots m} \quad \Gamma \vdash e_i : \tau_i^{i \in 1 \dots n}}{\Gamma \vdash f(e_i^{i \in 1 \dots n}) : \tau_j^{j \in 1 \dots m}}$$

2.6 Statement

2.6.1 if-then

$$\frac{\Gamma \vdash e : \text{boolean} \quad \Gamma \vdash \text{stmt}}{\Gamma \vdash \text{if } e \text{ then } \text{stmt} \text{ fi}}$$

2.6.2 if-then-else

$$\frac{\Gamma \vdash e : \text{boolean} \quad \Gamma \vdash \text{stmt}_1 \quad \Gamma \vdash \text{stmt}_2}{\Gamma \vdash \text{if } e \text{ then } \text{stmt}_1 \text{ else } \text{stmt}_2 \text{ fi}}$$

2.6.3 if-then-elif-else

$$\frac{\Gamma \vdash e_j^{j \in 1 \dots m} : \text{boolean} \quad \Gamma \vdash \text{stmt}_i^{i \in 1 \dots 3}}{\Gamma \vdash \text{if } e_1 \text{ then } \text{stmt}_1 \text{ (elif } e_j^{j \in 2 \dots m} \text{ then } \text{stmt}_2)_t^{t \in 1 \dots k} \text{ else } \text{stmt}_3 \text{ fi}}$$

2.6.4 while

$$\frac{\Gamma \vdash e : \text{boolean} \quad \Gamma \vdash \text{stmt}}{\Gamma \vdash \text{while } e \text{ do } \text{stmt} \text{ od}}$$

2.6.5 while-return

$$\frac{\Gamma \vdash e : \text{boolean} \quad \Gamma \vdash stmt_1 \quad \Gamma \vdash stmt_2}{\Gamma \vdash \text{while } stmt_1 \rightarrow e \text{ do } stmt_2 \text{ od}}$$

2.6.6 readln

$$\frac{(x_i^{i \in 1 \dots n} : \tau_i^{i \in 1 \dots n}) \in \Gamma}{\Gamma \vdash \text{readln}(x_i^{i \in 1 \dots n})}$$

2.6.7 write

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash \text{write}(e : \tau)}$$

2.6.8 simple-assign

$$\frac{(x : \tau) \in \Gamma \quad \Gamma \vdash e : \tau}{\Gamma \vdash x := e}$$

2.6.9 multiple-assign

$$\frac{(x_i^{i \in 1 \dots n} : \tau_i^{i \in 1 \dots n}) \in \Gamma \quad \Gamma \vdash e_j^{j \in 1 \dots n} : \tau_j^{j \in 1 \dots n}}{\Gamma \vdash x_i^{i \in 1 \dots n} := e_j^{j \in 1 \dots n}}$$

2.6.10 return

$$\frac{(\$ret : \tau) \in \Gamma \quad \Gamma \vdash e : \tau}{\Gamma \vdash \rightarrow e}$$

2.7 Tabelle di Compatibilità

op	operand	result
-	integer	integer
-	float	float
!	boolean	boolean

(a) optype1

op	first operand	second operand	result
+ - * /	integer	integer	integer
+ - * /	integer	float	float
+ - * /	float	integer	float
+ - * /	float	float	float
&&	boolean	boolean	boolean
< = > <= >= <>	integer	integer	boolean
< = > <= >= <>	integer	float	boolean
< = > <= >= <>	float	integer	boolean
< = > <= >= <>	float	float	boolean
< = > <= >= <>	string	string	boolean

(b) optype2

Figura 1: Relazioni di tipo per gli operatori primitivi. Gli operatori aritmetici lavorano sia su numeri interi sia su numeri in virgola mobile. Gli operatori logici ! && || (not, and e or) lavorano su boolean. Gli operatori di comparazione lavorano su tipi primitivi diversi da boolean.