Università degli studi di Salerno

Curriculum: Software Engineering and IT Management

Compilatori

Documentazione NewLang

Docente
Prof. **Gennaro Costagliola**

Candidati
Davide Prezioso - 0522501237
Vincenzo Esposito - 0522501385

Anno Accademico 2022-2023

Indice

Capitolo 1

Token, lessemi e struttura

1.1 Token e lessemi

Il linguaggio NewLang è stato ideato considerando i seguenti lessemi/token:

Codice sorgente	Symbol	Descrizione
start:	Symbol(sym.MAIN)	Rappresenta la funzione main
;	Symbol(sym.SEMI)	Rappresenta il punto e virgola
,	Symbol(sym.COMMA)	Rappresenta la virgola
|	Symbol(sym.PIPE)	Rappresenta la pipe
var	Symbol(sym.VAR)	Rappresenta una variabile non tipizzata
integer	Symbol(sym.INTEGER)	Rappresenta il tipo integer
real	Symbol(sym.REAL)	Rappresenta il tipo real(float)
string	Symbol(sym.STRING)	Rappresenta il tipo string
boolean	Symbol(sym.BOOL)	Rappresenta il tipo boolean
char	Symbol(sym.CHAR)	Rappresenta il tipo character
void	Symbol(sym.VOID)	Rappresenta il tipo void

Codice sorgente	Symbol	Descrizione
def	Symbol(sym.DEF)	Rappresenta una definizione di funzione
out	Symbol(sym.OUT)	Rappresenta una variabile puntatore
for	Symbol(sym.FOR)	Rappresenta il ciclo for
if	Symbol(sym.IF)	Rappresenta una condizione if
then	Symbol(sym.THEN)	Rappresenta il corpo che segue la condizione if
else	Symbol(sym.ELSE)	Rappresenta la condizione else
while	Symbol(sym.WHILE)	Rappresenta il ciclo while
to	Symbol(sym.TO)	Rappresenta la condizione per ciclare nel for
loop	Symbol(sym.LOOP)	Rappresenta il loop
<–	Symbol(sym.READ)	Rappresenta una read
–>	Symbol(sym.WRITE)	Rappresenta una write
–>!	Symbol(sym.WRITELN)	Rappresenta una writeln

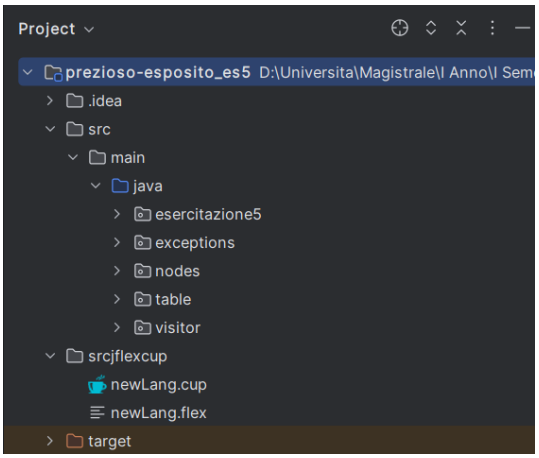
Codice sorgente	Symbol	Descrizione
(	Symbol(sym.LPAR)	Rappresenta la parentesi tonda aperta
)	Symbol(sym.RPAR)	Rappresenta la parentesi tonda chiusa
{	Symbol(sym.LBRACK)	Rappresenta la parentesi graffa aperta
}	Symbol(sym.RBRACK)	Rappresenta la parentesi graffa chiusa
:	Symbol(sym.COLON)	Rappresenta i due punti
«	Symbol(sym.ASSIGN)	Rappresenta un'assegnazione
return	Symbol(sym.RETURN)	Rappresenta una condizione di ritorno

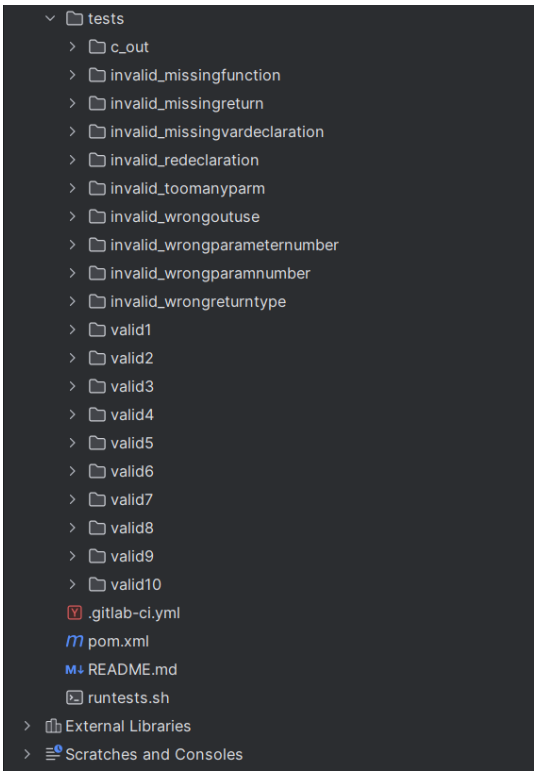
Codice sorgente	Symbol	Descrizione
true	Symbol(sym.TRUE)	Rappresenta il valore true
false	Symbol(sym.FALSE)	Rappresenta il valore false
+	Symbol(sym.PLUS)	Rappresenta l'operazione addizione
-	Symbol(sym.MINUS)	Rappresenta l'operazione sottrazione
*	Symbol(sym.TIMES)	Rappresenta l'operazione moltiplicazione
/	Symbol(sym.DIV)	Rappresenta l'operazione divisione
^	Symbol(sym.POW)	Rappresenta l'operazione potenza
&	Symbol(sym.STR_CONCAT)	Rappresenta una concatenazione
=	Symbol(sym.EQUALS)	Rappresenta un'uguaglianza
<>	Symbol(sym.NE)	Rappresenta una disuguaglianza
!=	Symbol(sym.NE)	Rappresenta una disuguaglianza
<	Symbol(sym.LT)	Rappresenta l'operatore minore
<=	Symbol(sym.LE)	Rappresenta l'operatore minore uguale
>	Symbol(sym.GT)	Rappresenta l'operatore maggiore
>=	Symbol(sym.GE)	Rappresenta l'operatore maggiore uguale
and	Symbol(sym.AND)	Rappresenta l'operatore logico and
or	Symbol(sym.OR)	Rappresenta l'operatore logico or
not	Symbol(sym.NOT)	Rappresenta l'operatore not

1.2 Struttura

Il progetto presenta la seguente struttura:

- package src/main/java contenente il main, le eccezioni, i nodi, utilities e i visitor
- package srcjflexcup contenente il cup e flex
- cartella test_files
- altri file utili





Capitolo 2

Gestione dello scoping

Per la gestione dello scoping, abbiamo inizialmente identificato tutti i costrutti che conducono alla creazione di un nuovo livello di scoping, e successivamente, abbiamo implementato la classe MyScope-Visitor.

Uno scope viene generato da tutti quei costrutti che contengono un blocco di istruzioni (Body), come ad esempio If, Else, For e While. Oltre a questi costrutti, che generano un nuovo livello di scoping, abbiamo considerato anche il nodo radice (ProgramNode), che consente di creare una tabella globale all'interno della quale vengono registrati i nomi dei metodi presenti nel programma e tutte le variabili globali contenute in esso.

Anche le funzioni, generando un nuovo livello di scoping, sono stae gestite, poiché possiedono un proprio blocco di istruzioni (Body).

Capitolo 3

Gestione della tipizzazione

Riportiamo alcune delle regole di tipo del linguaggio NewLang:

Identificatore

$$\frac{\Gamma(id) = \tau}{\Gamma \vdash id : \tau}$$

Costanti

$\Gamma \vdash int\_const : integer$
$\Gamma \vdash string\_const : string$
$\Gamma \vdash true : boolean$
$\Gamma \vdash false : boolean$

Operatori unari (vedi Table 1):

$$\frac{\Gamma \vdash e : \tau_1 \quad optype1(op_1, \tau_1) = \tau}{\Gamma \vdash op1\ e : \tau}$$

Operatori binari (vedi Table 2):

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2 \quad optype2(op_2, \tau_1, \tau_2) = \tau}{\Gamma \vdash e_1\ op_2\ e_2 : \tau}$$

Blocco dichiarazione-istruzione
(il type environment dell'istruzione stmt viene esteso con la dichiarazione di id, prima di fare il controllo di tipo dell'istruzione stmt):

$$\frac{\Gamma[id \mapsto \tau] \vdash stmt : notype}{\Gamma \vdash \tau\ id; stmt : notype}$$

Istruzione while

$$\frac{\Gamma \vdash e : boolean \quad \Gamma \vdash body : notype}{\Gamma \vdash \textbf{while}\ e\ \textbf{loop}\ body\ : notype}$$

Istruzione for

$$\frac{\Gamma \vdash e_1 : int\_const \quad \Gamma \vdash e_2 : int\_const \quad \Gamma[id \mapsto integer] \vdash body : notype}{\Gamma \vdash \textbf{for}\ id << e_1\ \textbf{to}\ e_2\ \textbf{loop}\ body\ : notype}$$

Istruzione if-then

$$\frac{\Gamma \vdash e : boolean \quad \Gamma \vdash body : notype}{\Gamma \vdash \textbf{if}\ e\ \textbf{then}\ body\ : notype}$$

Capitolo 4

Tabelle operatori

Riportiamo le tabelle per le operazioni unarie e binarie del linguaggio NewLang:

op1	operando	risultato
MINUS	integer	integer
MINUS	float	float
NOT	boolean	boolean

Figura 4.1: Operazioni Unarie

op	operando1	operando2	risultato
PLUS, TIMES, ...	integer	integer	integer
PLUS, TIMES, ...	integer	float	float
PLUS, TIMES, ...	float	integer	float
PLUS, TIMES, ...	float	float	float
STR_CONCAT	string	string	string
AND, OR	boolean	boolean	boolean
LT, LE, GT, ...	integer	integer	boolean
LT, LE, GT, ...	float	integer	boolean
LT, LE, GT, ...	integer	float	boolean
LT, LE, GT, ...	float	float	boolean

Figura 4.2: Operazioni Binarie