1. ANALISI LESSICALE

Solo gli identificatori verranno inseriti nella tabella dei simboli, che è stata implementato nella classe SymbolTable<String,Identifier> che estende la classe HashMap. Viene utilizzato il lessema stesso come chiave per accedere ai singoli elementi del hashmap. Ogni identificatore ha i seguenti campi:

- Id: nome dell'identificatore.
- Type: tipo dell'identificatore.
- Return type: che puo essere IN, OUT, INOUT, FUNCTION(nel caso si tratti del nome della funzione)

Sono stati considerati solo i numeri positivi in questa fase in quanto l'istruzione *i=i-5*, non sarebbe stata ritenuta valida per grammatica. Il motivo è che l'analizzatore cerca sempre di catturare il lessema più lungo, e quindi sarebbe stato restituito il token <int_const, -5>, invece che i token <minus> e <int_const, 5>. Così l'istruzione precendete avrebbe prodotto i token: <id,i> <ASSIGN> <id,i> <int_const, -1> che non è una sequenza di token adatti per la nostra grammatica.

2. ANALISI SINTATTICA

Sono state apportate le seguenti modifiche alla gramamtica:

- Sono state eliminate le produzioni: `arith_op, bool_op, rel_op.`
  La code delle seguenti produzioni è stata aggiunta alle produzioni di `expr`

```
expr ::=        | expr:e1 GT expr:e2 {: RESULT = new GtOP(e1,e2); :}
                | expr:e1 GE expr:e2 {: RESULT = new GeOP(e1,e2); :}
                | expr:e1 LT expr:e2 {: RESULT = new LtOP(e1,e2); :}
                | expr:e1 LE expr:e2 {: RESULT = new LeOP(e1,e2); :}
                | expr:e1 EQ expr:e2 {: RESULT = new EqOP(e1,e2); :}
                | expr:e1 PLUS expr:e2 {: RESULT = new AddOP(e1,e2); :}
                | expr:e1 MINUS expr:e2 {: RESULT = new DiffOP(e1,e2); :}
                | expr:e1 DIV expr:e2 {: RESULT = new DivOP(e1,e2);  :}
                | expr:e1 TIMES expr:e2 {:RESULT = new MulOP(e1,e2);  :}
                | expr:e1 OR expr:e2 {:RESULT = new OrOP(e1,e2); :}
                | expr:e1 AND expr:e2 {: RESULT = new AndOP(e1,e2); :}
                | expr:e1 DIV expr:e2 {: RESULT = new DivOP(e1,e2);  :}
                | expr:e1 TIMES expr:e2 {:RESULT = new MulOP(e1,e2);  :}
                | expr:e1 OR expr:e2 {:RESULT = new OrOP(e1,e2); :}
                | expr:e1 AND expr:e2 {: RESULT = new AndOP(e1,e2); :}
```

Questo per ovviare hai problemi che venivano causati dal sistema di gestione delle precedenze di CUP. Un possibile esempio può essere il seguente:

Con Expr ArithOp Expr nell'handle e "+" nel lookahead avviene uno shift invece che una reduce. Questo rappresenta un errore se "ArithOp" era relativo ad una "" o "/" perché bisognava ridurre.*

Questo succede perché il lookahead *"+",* ha precedenza maggiore di *Expr ArithOp Expr(Java CUP assegna come valore di precedenza ad una produzione la precedenza del terminale più a destra quindi non avendo terminale nel suo corpo ha precedenza 0).*

- È stata inoltre aggiunta una nuova produzione a `args`.

```
args ::= expr:e COMMA args:a
       |ID:id COMMA args:a
       | expr:e

stat ::= args:a WRITE SEMI
```

Questo perché la grammatica come era stata precedentemente pensata dava problemi nella *writeOp.* Un possibile esempio può essere il seguente:

Con un ID nell'handle e "," nel lookahead avviene uno shift invece che una reduce, perché il COMMA ha precedenza maggiore del ID, e cosi l'ID non verrà mai più ridotto ad EXPR che è quello che la WRITE si aspetterebbe. Assegnare una precedenza maggiore all'ID rispetto al COMMA, permetterebbe di ridurre tutti gli ID ad EXPR prima che venga shiftato il COMMA e questo avrebbe dato problemi con la READ, in quanto questa produzione si aspetta una serie di ID. Per questo motivo è stata aggiunta la produzione evidenziata sopra, permettendo di avere come argomenti di una WRITE non solo EXPR, ma anche ID.

3. ANALISI SEMANTICA

REGOLE TYPE SYSTEM

TABELLA PER OPERAZIONI ADDOP

	INTERO	**DOUBLE**	**STRINGA**	**CHAR**	**BOOLEANO**
INTERO	INTERO	DOUBLE	STRINGA (Concat STRING)	INTERO	INTERO
DOUBLE	DOUBLE	DOUBLE	STRINGA (Concat STRING)	DOUBLE	DOUBLE
STRINGA	STRINGA (Concat STRING)	STRINGA (Concat STRING)	STRINGA (Concat STRING)	STRINGA (Concat STRING)	
CHAR	INTERO	DOUBLE	STRINGA (Concat STRING)	STRINGA (Concat STRING)	INTERO
BOOLEANO	INTERO	DOUBLE		INTERO	INTERO

TABELLA PER OPERAZIONI ARITMETICI (MULOP, DIFFOP, DIVOP)

	INTERO	**DOUBLE**	**STRINGA**	**CHAR**	**BOOLEANO**
INTERO	INTERO	DOUBLE		INTERO	INTERO
DOUBLE	DOUBLE	DOUBLE		DOUBLE	DOUBLE
STRINGA					
CHAR	INTERO	DOUBLE		INTERO	INTERO
BOOLEANO	INTERO	DOUBLE		INTERO	INTERO

TABELLA PER OPERATORI RELAZIONALI (GT,GE,LT,LE)

	INTERO	**DOUBLE**	**STRINGA**	**CHAR**	**BOOLEANO**
INTERO	BOOLEANO	BOOLEANO		BOOLEANO	

DOUBLE	BOOLEANO	BOOLEANO		BOOLEANO	
STRINGA			BOOLEANO (strcmp)		
CHAR	BOOLEANO	BOOLEANO		BOOLEANO	
BOOLEANO					

TABELLA PER OPERATORI EQOP

	INTERO	**DOUBLE**	**STRINGA**	**CHAR**	**BOOLEANO**
INTERO	BOOLEANO	BOOLEANO		BOOLEANO	
DOUBLE	BOOLEANO	BOOLEANO		BOOLEANO	
STRINGA			BOOLEANO (strcmp)		
CHAR	BOOLEANO	BOOLEANO		BOOLEANO	
BOOLEANO					BOOLEANO

TABELLA PER OPERATORI RELAZIONALI (AND , OR)

	INTERO	**DOUBLE**	**STRINGA**	**CHAR**	**BOOLEANO**
INTERO					
DOUBLE					
STRINGA					
CHAR					
BOOLEANO					BOOLEANO

TABELLA PER OPERAZIONI NOTOP

	INTERO	**DOUBLE**	**STRINGA**	**CHAR**	**BOOLEANO**
RESULT	BOOLEANO	BOOLEANO		BOOLEANO	BOOLEANO

TABELLA PER UMINUS

	INTERO	**DOUBLE**	**STRINGA**	**CHAR**	**BOOLEANO**
RESULT	INTERO	DOUBLE			

TABELLA PER ASSEGNAMENTO

	INTERO	**DOUBLE**	**STRINGA**	**CHAR**	**BOOLEANO**
INTERO	VOID	VOID		VOID	VOID
DOUBLE	VOID	VOID		VOID	
STRINGA			VOID		
CHAR	VOID	VOID		VOID	
BOOLEANO	VOID				VOID

TABELLA PER OPERAZIONI CALLOP

RICHIESTO/PASSO	**INTERO**	**DOUBLE**	**STRINGA**	**CHAR**	**BOOLEANO**
INTERO	VOID	VOID		VOID	VOID
DOUBLE	VOID	VOID		VOID	VOID
STRINGA			VOID		

CHAR	VOID	VOID		VOID	VOID
BOOLEANO	VOID				VOID

*I PARAMETRI DI TIPO OUT E INOUT, NON POSSONO ESSERE COSTANTI MA SOLO RIFERIEMNTI A IDENTIFICATORI

TABELLA PER OPERAZIONI READOP

	INTERO	**DOUBLE**	**STRINGA**	**CHAR**	**BOOLEANO**
RESULT	VOID	VOID	VOID	VOID	

TABELLA PER OPERAZIONI WRITEOP

	INTERO	**DOUBLE**	**STRINGA**	**CHAR**	**BOOLEANO**
RESULT	VOID	VOID	VOID	VOID	VOID

TABELLA PER OPERAZIONI CONDITIONALOP

	INTERO	**DOUBLE**	**STRINGA**	**CHAR**	**BOOLEANO**
RESULT	VOID				VOID