

ANALISI LESSICALE

- Aggiunta di pattern per commenti
 - - tradizionali (`/* */`)
 - - di fine linea (`//`)
 - - di documentazione (`/** */`)

ANALISI SINTATTICA

- E' stata aggiunta una produzione in **Args**:
 - ID COMMA Args, per far accettare una sequenza di ID all'interno di una operazione di WRITE. Es. `(x,y->)`

ANALISI SEMANTICA

- Type checking
 - **ReadOp**: controllo della esistenza della variabile (sia nella current table che nella global); il controllo del tipo viene inferito
 - **CalOp**: controllo dell'esistenza del nome della funzione, controllo del tipo di variabili dati in input alla funzione, controllo del numero di parametri dati in input alla funzione
- Gestione degli errori
 - Se all'interno del programma vi è un errore semantico, il compilatore lancia una eccezione e si arresta (`System.exit(1)`).
- All'interno del body di una funzione, le variabili di tipo **out**, non possono essere utilizzate, cioè la variabile di tipo **out** (caso assegnazione) deve sempre trovarsi nella parte sinistra di una assegnazione.
- Le tabelle dei simboli delle funzioni vengono memorizzate in una lista la quale, insieme allo stack, viene passata al generatore del codice.
- Inoltre per la **CalOP**, solo nei parametri di tipo **in** ci può essere una espressione (es.somma) oppure una costante (stringa,intero,double).

Tabelle di compatibilità

Tabella per AssignOp

I figlio/Il figlio	Integer	Double	String	Char	Bool
Integer	void	Error	Error	Error	Error
Double	void	void	Error	Error	Error

String	Error	Error	void	Error	Error
Char	Error	Error	Error	void	Error
Bool	Error	Error	Error	Error	void

Tabella per GTOp, GEOp, LTOp, LEOp

I figlio/II figlio	Integer	Double	String	Char	Bool
Integer	Boolean	Boolean	Error	Error	Error
Double	Boolean	Boolean	Error	Error	Error
String	Error	Error	Error	Error	Error
Char	Error	Error	Error	Error	Error
Bool	Error	Error	Error	Error	Error

Tabella per EQOp

I figlio/II figlio	Integer	Double	String	Char	Bool
Integer	Boolean	Boolean	Error	Error	Error
Double	Boolean	Boolean	Error	Error	Error
String	Error	Error	Boolean	Error	Error
Char	Error	Error	Error	Boolean	Error
Bool	Error	Error	Error	Error	Boolean

Tabella per AddOp, MulOp, DivOp, DiffOp

I figlio/II figlio	Integer	Double	String	Char	Bool
Integer	Integer	Double	Error	Error	Error
Double	Double	Double	Error	Error	Error
String	Error	Error	Error	Error	Error
Char	Error	Error	Error	Error	Error
Bool	Error	Error	Error	Error	Error

Tabella per UminusOp

Tipo figlio	Tipo
--------------------	-------------

Integer	Integer
Double	Double
String	Error
Char	Error
Bool	Error

Tabella per CallOp

Tipo parametro/Parametro funzione	Integer	Double	String	Char	Bool
Integer	void	Error	Error	Error	Error
Double	Error	void	Error	Error	Error
String	Error	Error	void	Error	Error
Char	Error	Error	Error	void	Error
Bool	Error	Error	Error	Error	void

GENERAZIONE DEL CODICE

- Prende in input lo stack e le tabelle dei simboli per le funzioni
- Per le funzioni
 - ai parametri di **out** e **inout** viene aggiunto "*" all'interno della funzione per indicare che sono puntatori
 - nella chiamata di una funzione, ai parametri di **out** e **inout** viene aggiunto "&"
 - se viene stampato un valore di **inout**, viene aggiunto "*" nella printf
 - all'interno di una assegnazione, viene aggiunto "*" al valore di **out** o **inout**
 - se viene letto un valore di **inout**, viene aggiunto "&*" nella scanf
- Le stringhe vengono tradotte in array di caratteri (char [])
- All'interno della **printf** ci possono essere più valori; lo stesso equivale per la **scanf**