Grammatica EBNF Astratta di Simpla (Albero Astratto)

program → *var-decl-list* *func-decl-list* *stat-list*
var-decl-list → { *var-decl* }
var-decl → *id-list* **type** in cui **type** qualificato come: `INTEGER` | `REAL` | `STRING` | `BOOLEAN` | `VOID`
id-list → { **id** }$^{+}$
func-decl-list → { *func-decl* }
func-decl → **id** *opt-param-list* **type** *var-decl-list* *stat-list*
opt-param-list→ { *param-decl* }
param-decl → **id type**
stat-list → {*assign-stat* | *if-stat* | *while-stat* | *for-stat* | *return-stat* | *read-stat* | *write-stat* | *func-call* | **break**}$^{+}$
assign-stat → **id** **EXPR**
if-stat → **EXPR** *stat-list* [*stat-list*]
while-stat → **EXPR** *stat-list*
for-stat → **id** *expr* *expr* *stat-list*
return-stat → [**EXPR**]
read-stat → *id-list*
write-stat → **write-op** *expr-list* in cui **write-op** qualificato come: `WRITE` | `WRITELN`
expr-list → { **EXPR** }$^{+}$

in cui **EXPR** = *logic-expr* | *rel-expr* | *math-expr* | *neg-expr* | **id** |
intconst | **realconst** | **strconst** | **boolconst** |
func-call | *cond-expr* | *casting*

in cui la qualifica è stabilita nel seguente modo:

logic-expr : `AND` | `OR`
rel-expr : `EQU` | `NEQ` | `'>'` | `GEQ` | `'<'` | `LEQ`
math-expr : `'+'` | `'-'` | `'*'` | `'/'`
neg-expr : `'-'` | `NOT`
casting : `INTEGER` | `REAL`

logic-expr → **EXPR** **EXPR**
rel-expr → **EXPR** **EXPR**
math-expr → **EXPR** **EXPR**
neg-expr → **EXPR**
func-call → **id** { **EXPR** }
cond-expr → **EXPR** **EXPR** **EXPR**
casting → **EXPR**

Note:

- Ogni nodo è identificato da un `tipo` (terminale o nonterminale).
- L'albero astratto non contiene nodi generici **EXPR**, ma solo nodi specifici (*logic-expr*, *rel-expr*, ..., *casting*).
- Ove richiesto, il qualificatore (`INTEGER`, ..., `AND`, `OR`, ...,) può essere memorizzato nel campo `ival` del campo `valore` (di tipo `union`) del nodo.