```
Tipo -> int {Tipo.trad = ":integer"}
Tipo -> float{Tipo.trad = ":float"}
Var -> id {Var.trad = id.lexema}
Var -> amp id {Var.trad = "var " || id.lexema"}
A -> Tipo Var { A.trad = Var.trad || Tipo.trad}
MasArgs -> A { MasArgs.trad = A.trad}
MasArgs -> MasArgs coma A{ MasArgs.trad = MasArgs.trad || ";" || A.trad}
LArgs -> e { LArgs.trad = ""}
LArgs -> MasArgs { LArgs.trad = MasArgs.trad}
Args -> pari LArgs pard {IF LArgs.trad == "" then; Args.trad = ""; else Args.trad = "(" || LArgs.trad || ")" }
T -> numentero { T.trad = numentero.lexema, T.tipo = ENTERO}
T -> numreal { T.trad = numreal.lexema, T.tipo = FLOAT}
T -> id { T.trad = id.lexema, T.tipo = id.tipo}
E -> T {E.trad = T.trad, E.tipo = T.tipo}
E -> E opas T {IF E.tipo == ENTERO && T.tipo == ENTERO then;
                  E.trad = E.trad || " " || opas.lexema || "i " || T.trad;
                  E.tipo = ENTERO;
               ELIF E.tipo == ENTERO && T.tipo == REAL then;
                  E.trad = "itor(" || E.trad || ") " || opas.lexema || "r " || T.trad;
                  E.tipo = REAL;
               ELIF E.tipo == REAL && T.tipo == ENTERO then;
                  E.trad = E.trad || " " || opas.lexema || "r itor(" || T.trad || ") ";
                  E.tipo = REAL;
               ELIF E.TIPO == REAL && T.tipo == REAL then;
                  E.trad = (E.trad || " " || opas.lexema || "r " || T.trad);
                  E.tipo = REAL; }
Instr -> Tipo id pyc {Instr.trad = "", Instr.decl = get_fun_id()||get_profun()|| id.lexema || Tipo.trad || "\n"}
Instr -> if pari E pard Instr { IF Instr.trad == "" then;
                                   Instr.trad = "if " || E.trad || " then\ndoNothing;"; Instr.decl = "";
                                ELSE
                                   Instr.trad = "if" || E.trad || " then\n" || Instr.trad; Instr.decl = ""}
Instr -> id asig E pyc { IF id.tipo == REAL && E.tipo == ENTERO then;
                            Instr.trad = id.lexema || " := itor(" || E.trad || ")"; Instr.decl = "";
                         ELIF id.tipo == ENTERO && E.tipo == REAL then;
                             ERROR TIPOS INCOMPATIBLES;
                         ELSE
                             Instr.trad = id.lexema || " := " || E.trad ; Instr.decl = ""}
Instr -> Bloque { Instr.trad = Bloque.trad; Instr.trad= Bloque.decl}
SInstr -> Instr { SInstr.trad = Instr.trad; SInstr.decl=Instr.decl}
SInstr -> SInstr Instr { IF Instr.trad == "" then;
                            SInstr.trad = SInstr.trad; SInstr.decl = SInstr.decl
                         ELIF SInstr.trad == "" && Instr.trad != "" then;
                            SInstr.trad = Instr.trad; SInstr.decl = Instr.decl
                         ElIF SInstr.trad == "" && Instr.trad == "" then;
                            SInstr.trad = ""; SInstr.decl = ""
                         ELSE
                            SInstr.trad = SInstr.trad || ";" || Instr.trad; SInstr.decl = SInstr.decl + Instr.decl }
Bloque -> llavei SInstr llaved { IF SInstr.trad == "" Then;
                                    Bloque.trad = "begin\ndoNothing\nend\n";Bloque.decl = "";
                                 ELSE
                                    Bloque.trad = "begin\n" || SInstr.trad || "end\n"; Bloque.decl = SInstr.decl}
Fun -> Tipo id Args Bloque { Fun.trad = "function " || id.lexema || Args.trad || Tipo.trad||";\nvar  "||Bloque.decl ||Bloque.trad;}
S -> Fun { S.trad = Fun.trad}
```