Γ ⊢ true: bool Γ ⊢ false: bool Γ ⊢ float: float Γ ⊢ int: int Γ ⊢ string: string

Γ ⊢ intConst: int Γ ⊢ stringConst: string Γ ⊢ null: null

DICHIARAZIONI

$\frac{(x : \tau) \in \Gamma}{\Gamma \vdash x : \tau}$

STATEMENT

$\frac{\Gamma \vdash stmt1 \quad \Gamma \vdash stmt2}{\Gamma \vdash stmt1; stmt2;}$

WHILE

$\frac{\Gamma \vdash e: bool \quad \Gamma \vdash stmt}{\Gamma \vdash \text{while e do stmt od}}$

IF ELIF ELSE

$\frac{\Gamma \vdash e1: bool \quad \Gamma \vdash e2: bool \quad \Gamma \vdash stmt1 \quad \Gamma \vdash stmt2 \quad \Gamma \vdash stmt3}{\Gamma \vdash \text{if e1 then stmt1 elif e2 then stmt2 else stmt3 fi}}$

RETURN

$\frac{(\$ret1 : \tau1, .. , \$retn : \tau n) \in \Gamma \quad \Gamma \vdash e1 : k1 \quad \dots \quad \Gamma \vdash en : kn \quad isCompType(\tau1, k1) : \tau1, \dots , isCompType(\tau n, kn) : \tau n}{\Gamma \vdash \text{-> e1, .. ,en}}$

CALLPROC

$\frac{\Gamma \vdash f: \tau1, \dots, \tau n \text{ -> } T \quad \Gamma \vdash e1 : \tau1, \dots , \quad \Gamma \vdash en : \tau n \quad \Gamma \vdash T : T}{\Gamma \vdash f(e1, .., en) \text{ -> } T}$

ASSIGN

$\frac{\Gamma \vdash id1 : \tau1, \dots , \quad \Gamma \vdash idn : \tau n \quad \Gamma \vdash e1 : k1, \dots , \quad \Gamma \vdash kn : \tau n \quad isComp (op, \tau1, k1) : x1 , \dots , isComp (op, \tau n, kn) : xn}{\Gamma \vdash (id1, \dots , idn \ op \ e1, \dots en) : x1 , \dots , xn}$

OPERAZIONI UNARIE

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

OPERAZIONI BINARIE

$\frac{\Gamma \vdash e1 : \tau1 \quad \Gamma \vdash e2 : \tau2 \quad isCompatibleType(op2, \tau1, \tau2) = \tau}{\Gamma \vdash (e1\ op2\ e2) : \tau}$

isCompatibleType			
OP	FIRST OP	SECOND OP	RESULT
+ - * /	int	int	int
+ - * /	int	float	float
+ - * /	float	float	float
+	string	string	string
< = > <>	int	int	bool
< = > <>	int	float	bool
< = > <>	float	int	bool
< = > <>	float	float	bool
< = > <>	string	string	bool
&& ||	bool	bool	bool
:=	int	int	int
:=	int	float	float
:=	float	float	float
:=	string	string	string
:=	bool	bool	bool
-	int	/	int
-	float	/	float
!	bool	/	bool