Reglas lógicas de inferencia de tipos

- Use chatGPT para poder entender bien cómo se hacían las reglas de inferencia, por que si las entiendo pero me cuesta trabajo construirlas.

1. **[Int] (constante numérica)**

```
──────────────── [Int]
⊢ n : int
```

2. **[Var] (variable)**

```
x:τ ∈ Γ
──────────────── [Var]
Γ ⊢ x : τ
```

3. **[Assign] (asignación)**

```
Γ ⊢ x : int    Ù    Γ ⊢ e : int
──────────────────────── [Assign]
Γ ⊢ (x = e) : int
```

4. **[Add] (aritmética: +, –)**

```
Γ ⊢ e₁ : int    Ù    Γ ⊢ e₂ : int
──────────────────────── [Add]
Γ ⊢ (e₁ + e₂) : int
```

- El mismo para restar

5. **[Mul] (aritmética: *, /)**

```
Γ ⊢ e₁ : int    Ù    Γ ⊢ e₂ : int
──────────────────────── [Mul]
Γ ⊢ (e₁ ∗ e₂) : int
```

6. **[Rel] (operadores relacionales >,<, ==, != >=, <=)**

```
Γ ⊢ e₁ : int    Ù    Γ ⊢ e₂ : int
──────────────────────────── [Rel]
Γ ⊢ (e₁ relop e₂) : int
```

7. **[Call] (llamada a función)**

```
f:(τ₁,…,τₙ)→τᵣ ∈ Γ    Ù    ∀i. Γ ⊢ eᵢ : τᵢ
————————————————————————————— [Call]
Γ ⊢ f(e₁,…,eₙ) : τᵣ
```

8. **[Return-int] (return en función int)**

```
Ret = int    Ù    Γ ⊢ e : int
————————————————————— [Return-int]
(validar) return e
```

9. **[Return-void] (return en función void)**

```
Ret = void    Ù    (no e)
————————————————————— [Return-void]
(validar) return
```

10. **[If] / [While] (condición de control)**

```
Γ ⊢ e : int
———————— [If]
(validar) if(e) …

Γ ⊢ e : int
———————— [While]
(validar) while(e) …
```

Explicación de la estructura de la tabla de símbolos

La tabla de símbolos se definió de la siguiente manera:

```
        self.name        = name
        self.kind        = kind
        self.type        = typ
        self.array_size  = array_size
        self.params      = params or []
        self.declared_at = declared_at
```

El contenido de esta tabla es un mapa.

Cada vez que entras en un nuevo bloque—por ejemplo la función main, creas un nuevo ámbito: empujas (push) un diccionario vacío sobre la pila. Cuando sales del bloque, haces pop y destruyes ese diccionario, recuperando el ámbito anterior.

FLUJO

1. Inicializas la tabla con un ámbito global y registras las funciones predefinidas.

2. Recorres (traverse) sólo los hijos de el programa para insertar todas las declaraciones globales (var_decl, fun_decl).
3. Para cada función:
    - Haces push_scope() (entras en su ámbito local).
    - Insertas sus parámetros como variables locales.
    - Recorres su AST (declaraciones locales y sentencias). Cada vez que encuentras un var_decl dentro, haces insert_symbol, y cada vez que evalúas una expresión o sentencia usas lookup_symbol para checar uso y tipos.

- Al terminar, pop_scope (cierras su ámbito).