

## Regras semânticas da linguagem CalculaDin

A linguagem tem apenas dois tipos: **real** (implementado como float) e **bool** (implementado como bool).

**Variáveis devem declaradas antes do uso**, i.e., atribuições, expressões e as funções input/output aceitam apenas variáveis previamente declaradas.

Variáveis podem ser declaradas em qualquer ponto do programa principal, **exceto dentro de blocos de comandos de if-then-else**. Dessa forma, todas as variáveis terão escopo global. Declarações dentro de blocos if devem gerar erro semântico.

Nas **atribuições**, o tipo do lado direito deve ser o igual ao lado esquerdo.

Nos condicionais, a **condição deve ser do tipo booleano**.

Os cálculos aritméticos e as expressões lógicas podem utilizar tanto constantes numéricas ou booleanas quanto variáveis, desde que os tipos sejam compatíveis.

As **regras de compatibilidade de tipos** para os cálculos/expressões são as seguintes:

| Tipo de operador                       | Regras                                     |
|--|--|
| Operadores aritméticos (+, -, *, /)    | (real, real) → real                        |
| Operadores relacionais (<, <=, >, >=): | (real, real) → bool                        |
| Operadores de diferença (==, !=):      | (real, real) → bool<br>(bool, bool) → bool |
| Operadores lógicos binários (and, or): | (bool, bool) → bool                        |
| Operador lógico unário (not):          | (bool) → bool                              |

## Implementação da verificação semântica

Foram fornecidos três módulos que auxiliarão na implementação da verificação semântica:

- o módulo `defs_cldin2` contém **definições de tipos** importantes para anotação da AST e população da tabela de símbolos;
- o módulo `sem_cldin2_v0` contém uma implementação de **tabela de símbolos** que manipula apenas variáveis globais e calcula o deslocamento das variáveis na pilha de execução (útil para a geração de código). **É nesse módulo que você escreverá o seu VerificadorSemantico**;
- por fim, o módulo `ast_cldin2` contém as definições da AST já com os **campos necessários para a anotação** semântica.

O verificador semântico será um **Visitador**, assim como a `ImpressoraAST`. Cada método de visita poderá:

- adicionar símbolos à **tabela de símbolos**;
- verificar **regras semânticas** pertinentes ao nó;
- incluir informações adicionais aos símbolos da tabela de símbolos;
- **anotar a AST**.

As **anotações à AST** necessárias para a etapa posterior (i.e., geração de código) são as seguintes:

- o nó Programa deve ser anotado com o **total de variáveis declaradas**;
- os nós do tipo `CalcId` devem ser anotados com os respectivos objetos **Símbolo**;
- os nós do tipo `Calculo` devem ter seu campo `tipo` anotado com um objeto **Tipo**.