# Projeto de Compiladores
# Parte 2

A segunda parte do projeto consistirá na implementação de alguns métodos de análise semântica dos códigos da linguagem descrita pela gramática implementada na parte 1 do projeto.
Será fornecido um arquivo *Grammar.g4* com a implementação da gramática.
Vocês devem implementar o visitor da gramática de forma a implementar um *type checker*, seguindo os requisitos abaixo:

**Requisitos:**

- Siga as seguintes regras para atribuições ou operações equivalentes (passagem de parâmetros, retorno de funções, etc):
  - **int = FloatExpression**.
    - Esta operação irá gerar um ***warning*** notificando possível perda de informação na atribuição da variável int e indicando a linha e coluna da ocorrência.
  - Atribuições do tipo **int = string** ou **float = string** devem gerar um ***erro***, indicando a linha e coluna do ocorrido.
  - Expressões do tipo **void** não podem ser atribuídas a nenhuma variável nem colocadas como operadores em expressões aritméticas, gerando um **erro** caso isto aconteça, indicando a linha e coluna da ocorrência.
  - Da mesma forma, funções com tipo **void** não podem retornar expressões de nenhum tipo.
    - Um erro indicando o retorno de um valor ***non-void*** em uma função ***void*** deve ser emitido caso isto ocorra, indicando a linha e coluna da ocorrência.

        - ***Arrays*** devem seguir as mesmas regras indicadas acima, porém seu ***erro*** ou ***warning*** emitido deve indicar também o índice no qual houve a ocorrência do mesmo, considerando que só teremos **arrays** estáticos na linguagem. (Se houver mais de um problema na definição do mesmo array, cada problema deve gerar a saída (**erro** ou **warning**) equivalente).
    - Operações aritméticas entre expressões do tipo **int** e **float** devem retornar uma expressão do tipo **float**.
    - Expressões que sejam o índice de um array devem ser do tipo **int**, gerando um **erro**, indicando linha e coluna, caso contrário.
    - Caso uma variável ou função não definida seja usada no programa, deve ser gerado um **erro** indicando que a variável/função não foi declarada.
        - Porém, não precisa checar se uma variável ou função já foi declarada: todos os testes utilizam nomes únicos para cada variável ou função declarada.
    - Também deve ser checado se o número de parâmetros passados para determinada chamada de função é correto.
    - Lembre-se que toda função definida possui um tipo, e sua chamada (***function_call***) retorna um valor daquele tipo.
    - As saídas geradas utilizando os arquivos de entrada em *inputs* devem corresponder aos arquivos presentes em *outputs*.

Vocês precisarão editar o arquivo *GrammarCheckerVisitor.py* disponibilizado no diretório *projeto2* para implementar os requisitos pedidos.

**DICA:**

- Para quem vai rodar no Windows, leia o Readme disponibilizado no diretório.

- Leiam os comentários feitos no arquivo para algumas informações sobre a implementação em Python.
- Os arquivos de saída dão uma boa noção do que está sendo pedido acima. Consulte-os enquanto estiver implementando o visitor.
- Se forem usar a gramática própria, modifiquem os nomes das regras *file* e *type* para *fiile* e *tyype* na gramática, pois os nomes *file* e *type* são palavras reservadas da linguagem Python. Lembre-se também de gerar um novo visitor e renomeá-lo para *GrammarCheckerVisitor.py*.