O objetivo final desse trabalho é a implementação de um compilador que gere *bytecodes Java* para a linguagem definida pela gramática abaixo, a saída deve ser um arquivo texto com mnemônicos que representem as instruções. O arquivo gerado deve ser montado pelo *Jasmin.*

2ª Fase – Geração de Código Intermediário

Na segunda fase do trabalho deve ser gerada a árvore sintática abstrata (AST) para o código, o programa será representado como uma lista de funções, incluindo o bloco principal.

- Cada função deve ser representada por uma estrutura de dados contendo: nome da função, tipo do retorno, lista de parâmetros e tipos, tabela de símbolos (para as variáveis locais) e o bloco de instruções.
- O bloco de instruções pode ser uma lista de ponteiros para estruturas que representam a árvore sintática abstrata.
- A árvore sintática pode conter até dois nós filhos (para poder representar operandos de operadores binários) e até dois blocos de instruções (para poder representar um comando *if* com *else*), deve conter também campos para armazenamento de constantes e identificadores de variáveis ou funções.

Também deve ser fornecida a implementação de uma função que percorra a estrutura gerada e imprima um código correspondente ao código original.

3ª Fase – Análise Semântica (Verificação de Tipos)

O analisador semântico deve receber como entrada a AST, fazer a verificação de tipos e retornar uma AST correspondente incluindo as coerções de tipos, erros e advertências deverão ser emitidos no processo. As regras para coerção de tipos e emissão de mensagens de erro são:

- Em expressões binárias aritméticas ou relacionais quando um dos operandos for do tipo *int* e o outro for do tipo *double* o operando do tipo *int* deve ser convertido a *double*.
- Quando uma variável declarada como *double* receber o valor de uma expressão de tipo *int*, o resultado da expressão deve ser convertido para o tipo *double*. Isso é válido para comandos de atribuição, passagem de parâmetros em chamadas de funções e para o retorno de funções.
- Quando uma variável declarada como *int* receber o valor de uma expressão de tipo *double*, o resultado da expressão deve ser convertido para o tipo *int*, nesse caso deve ser emitida uma mensagem de advertência. Isso é válido para comandos de atribuição, passagem de parâmetros em chamadas de funções e para o retorno de funções.
- O tipo *string* pode ocorrer apenas em expressões relacionais, os dois operandos devem ser do mesmo tipo, caso contrário uma mensagem de erro deve ser emitida.

- Expressões com tipos incompatíveis devem emitir mensagens de erro.
- Chamadas de funções com número de parâmetros errados ou com parâmetros formais e reais com tipos conflitantes devem ocasionar a emissão de mensagens de erro.
- Atribuição de variáveis ou retorno de funções com tipos conflitantes devem ocasionar a emissão de mensagens de erro.
- O uso de variáveis não declaradas deve informado com uma mensagem de erro.
- Chamada de funções não declaradas deve ocasionar a emissão de uma mensagem de erro.
- A existência de variáveis multiplamente declaradas em uma mesma função deve ocasionar a emissão de uma mensagem de erro.
- A existência de funções multiplamente declaradas deve ocasionar uma mensagem de erro.