Trabalho 2 - Análise Sintática
MATA61 - Compiladores

O grupo deve criar um analisador sintático, através do Flex e Bison, para uma linguagem de programação criada no Trabalho 1, lembrando que tal linguagem deve seguir as seguintes regras:

- Deve ser imperativa e compatível com o ASCII.
- Deve ser possível inicializar variáveis do tipo numérico, seja inteiro ou de número flutuante, com tipagem estática, além de vetores do tipo numérico.
- Deve ser possível fazer acessos e atribuições às variáveis inicializadas, incluindo os elementos individuais dos vetores. As atribuições devem ser expressões aritméticas que podem conter números pré-definidos e/ou variáveis e/ou funções.
- Deve ser possível criar desvios condicionais simples e compostos, com a condição sendo feita com uma operação de comparação.
- Deve ser possível criar um comando de repetição, com a condição para a repetição sendo feita com uma operação de comparação.
- Deve ser possível criar funções que, a partir de um número definido de parâmetros (com tipagem estática), retorne, um número inteiro, ponto flutuante ou não retorne nada. O tipo de retorno deve ser explicitado na definição da função. Após a criação da função, é possível chamá-la no código.

O executável obtido ao compilar os códigos do Bison e Flex devem mostrar a árvore sintática obtida através da entrada, de forma similar ao exemplo abaixo. Tokens que possuem atributos variados devem ser explicitados na árvore sintática, de forma que deve ser possível ver o nome dos identificadores e valores numéricos na árvore.
É necessário também fazer uma documentação explicando o motivo do uso das regras de produção. É possível fazer alterações da linguagem utilizada no Trabalho 1 para o Trabalho 2, porém é necessário comentar as mudanças na documentação.

Exemplo:
O seguinte código:

```
int x;
x = 5;
if (x < 3) {
   x = 1;
} else {
   x = 0;
}
```

Deve gerar uma árvore sintática similar à:

```
program
        statement_list
                statement_list
                        statement_list
                                statement
                                        declaration
                                                INT
                                                x
                        statement
                                attribution
                                        x
                                        5
                statement
                        if_statement
                                condition
                                        x
                                        relop
                                                <
                                        3
                                statement_list
                                        statement
                                                attribution
                                                        x
                                                        1
                                statement_list
                                        statement
                                                attribution
                                                        x
                                                        0
```

Também será considerado a qualidade da integração entre o analizador léxico e o analisador sintático, com a criação de padrões concisos e de fácil uso pelo analisador sintático (não é necessário documentar a mudança nos padrões).
A entrega deve ser feita em um arquivo .zip com todos os arquivos a serem analisados.

Tamanho do Grupo: Até 4 (quatro) pessoas.
Correção:

- **Código Bison: 4 pontos.**
- **Documentação: 4 pontos.**
- **Integração entre Flex e Bison: 2 pontos.**

Data de Entrega: até 10/12/2025.

Local de Entrega: Moodle.