Atividade de Construção de Compiladores

Turma: 2025/2
Professor: Ed Wilson Tavares Ferreira
Objetivo final: Desenvolver um compilador funcional como requisito obrigatório para aprovação no curso.

Especificação da Gramática

Projete uma gramática que inclua no mínimo:

- **Tipos de dados:** Dois tipos primitivos (ex: `int`, `string`);
- **Entrada/saída:** Comandos `leia()` e `escreva()`;
- **Controle de fluxo:**
  - Condicional: `se...então...senão`;
  - Repetição: `enquanto...faça`;
- **Expressões:**
  - Aritméticas com parênteses e operadores `+, -, *, /`;
  - Lógicas com quatro operadores (ex: `&&`, `||`, `!`, `==`).

Sugestões:

- Utilize Gramática LL(1);
- Remova as ambiguidades;
- Remova recursão à esquerda;
- Aplique fatoração.

Importante! Cada dupla deverá indicar/escolher/receberá uma linguagem de programação para ser utilizada como base de sua gramática

---

Analisador Léxico

Implemente um scanner que:

1. **Processe arquivo-fonte** e gere tokens no formato: <Tipo do Token, Lexema, Linha, Coluna>;
2. **Registre logs** detalhados em arquivo texto e/ou tela;
3. **Reporte erros** com formato:
   `ERRO LÉXICO [Linha 5, Coluna 12]: Símbolo '#' inválido`.

Analisador Sintático

Desenvolva um parser que:

1. **Gere uma AST** em formato visualizável (ex: DOT/Graphviz);
2. **Valide a estrutura** do código-fonte conforme a gramática;
3. **Trate erros** com mensagens intuitivas:
   `ERRO SINTÁTICO [Linha 8, Coluna 3]: Esperado ';', encontrado '}'`.
   Requisito técnico:

- Deve consumir os tokens gerados pelo analisador léxico;

Critérios de Avaliação

Componente	Peso	Detalhes
Apresentação	40%	Apresentação em sala
Funcionalidade	40%	Casos de teste e execução correta
Qualidade do código	10%	Documentação e boas práticas
Tratamento de erros	10%	Mensagens claras e recuperação

Recomenda-se o uso de Git para versionamento e VS Code com extensões para ANTLR.

Casos de Teste

Implemente os programas abaixo na linguagem definida pela sua gramática, seguindo os requisitos:

1. Triângulo de Pascal

Objetivo: Validar estruturas de repetição, link.
Especificação:

- Ler um número inteiro `n` (≥1) como entrada;
- Gerar e exibir as primeiras `n` linhas do triângulo;
- **Exemplo para n=5:**
  1
  1 1
  1 2 1
  1 3 3 1
  1 4 6 4 1

Critérios de avaliação:

- Verificação de limites (ex: `n=0` deve gerar erro)
- Formatação correta do output

2. Classificação de Triângulos

Objetivo: Testar expressões lógicas e estruturas condicionais aninhadas.
Explicação
Especificação:

- Ler três valores decimais positivos (`a` , `b` , `c`);
- Verificar se formam um triângulo válido;
- Classificar como:
  Equilátero (a == b == c)
  Isósceles (apenas dois lados iguais)
  Escaleno (todos lados diferentes)
- **Exemplo de saída:**
  Entrada: 3, 4, 5 → "Triângulo escaleno válido"
  Entrada: 1, 1, 3 → "Medidas inválidas"
- **Casos extremos:**
  Valores negativos/zero (deve gerar erro)
  Ordem de entrada irrelevante (3,4,5 ≡ 5,3,4)