	Centro Universitário Nobre de Feira de Santana		
	Curso: ENGENHARIA DA COMPUTAÇÃO		**Professores:** Thiago de Lima Mariano
	Habilitação:		**Turno:** VESPERTINO
	Período letivo: 25.1	**Turma/Disciplina:** 6º e 7º Semestres	**Disciplina:** Compiladores

TRABALHO DISCENTE EFETIVO - TDE

Descrição: Desenvolvimento de uma linguagem compilada com seus analisadores léxicos e sintático.

Equipes de trabalho: O trabalho será desenvolvido por equipes formadas com 4 alunos. Haverá uma equipe com 5 alunos.

Objetivo

O objetivo deste trabalho é proporcionar aos alunos uma experiência prática com as **primeiras fases de um compilador**, definindo sua própria **linguagem de programação procedural** e desenvolvendo os módulos de **análise léxica e sintática**.

Descrição do Trabalho

Os alunos deverão:

1. **Projetar uma linguagem de programação procedural**, definindo sua sintaxe e regras básicas.
2. **Implementar um analisador léxico** que converte o código-fonte em uma sequência de tokens.
3. **Implementar um analisador sintático** que valida a estrutura do código e gera a árvore sintática abstrata (AST).

O trabalho poderá ser desenvolvido em Python ou Java.

Parte 1: Definição da Linguagem

Cada grupo deverá definir os elementos da linguagem, incluindo:

1. **Palavras-chave**
2. **Expressões matemáticas com e sem parênteses**
3. **Operadores Aritméticos (**soma, multiplicação, divisão, subtração e resto da divisão**)**
4. **Operadores Lógicos (**and lógico, or lógico, negação**)**

5. **Operadores de comparação (**maior que, menor que, maior ou igual a, menor ou igual a, diferente de e igual a**)**
6. **Tipos de dados (**inteiro, cadeira de caracteres e números com ponto flutuante**)**
7. **Regras para declaração e atribuição de variáveis**
8. **Estruturas de controle de fluxo (**com opção caso a condição seja satisfeita e caso contrário**)**
9. **Estrutura de Loop**
10. **Criação de procedures com e sem parâmetros (**Não tem retorno**)**
11. **Criação de funções com e sem parâmetros (**Existe retorno**)**
12. **Possibilidade de utilizar comentários no código fonte desenvolvido.**

OBS: É possível utilizarmos vários operadores de comparação e vários operadores lógicos nas condições da estrutura de fluxo e na estrutura de loop.

Parte 2: Implementação do Analisador Léxico

O analisador léxico deve converter o código-fonte em uma sequência de tokens, ignorando espaços em branco e comentários. A tabela de símbolos criada pelo analisador léxico deverá ser exportada em um arquivo html.

Exemplo de Tokens Gerados

Código Fonte	**Tokens Gerados**
inteiro x;	('TIPO', 'inteiro') ('ID', 'x') ('PONTO_VIRGULA', ';')
x = 10;	('ID', 'x') ('IGUAL', '=') ('NUMERO', '10') ('PONTO_VIRGULA', ';')

Se um símbolo desconhecido for encontrado, o analisador deve gerar um **erro léxico**.

OBS: Utilizar ferramentas como Flex (ou Lex) pode facilitar essa etapa.

Parte 3: Implementação do Analisador Sintático

Implementar um analisador sintático (parser) que verifica se a sequência de tokens está de acordo com a gramática da linguagem.

- Construir uma árvore sintática abstrata (AST) que representa a estrutura do programa. O compilador deverá criar um arquivo json contendo a AST gerada do código fonte.

O analisador sintático verificará a **estrutura do código**, identificando se a sequência de tokens forma **sentenças válidas** segundo a gramática da linguagem.

OBS: Ferramentas como Bison (ou Yacc) podem ser úteis.

Exemplo de Produções da Gramática

programa → declaracoes comandos
declaracoes → (tipo ID PONTO_VIRGULA)*

comandos → (atribuicao | estrutura_controle)*
atribuicao → ID IGUAL expressao PONTO_VIRGULA
estrutura_controle → condicional | repeticao
condicional → SE ABRE_PAREN expressao FECHA_PAREN ABRE_CHAVE comandos FECHA_CHAVE (SENAO ABRE_CHAVE comandos FECHA_CHAVE)?
repeticao → ENQUANTO ABRE_PAREN expressao FECHA_PAREN ABRE_CHAVE comandos FECHA_CHAVE
expressao → termo ((MAIS | MENOS) termo)*
termo → fator ((MULT | DIV) fator)*
fator → NUMERO | ID | ABRE_PAREN expressao FECHA_PAREN

Casos de Erro Sintático

Código Fonte	Erro Detectado
x = 5	**Erro:** Ponto e vírgula esperado no final da linha.
se x > 5 {	**Erro:** Parênteses () esperados após se.

Entrega do Trabalho

A entrega acontecerá em duas etapas: TDE1 (13/04/2025) e TDE2 (15/06/2025).

O TDE1 deverá conter a definição da linguagem e a implementação do analisador léxico e todas as suas especificações.

O TDE2 deverá conter a implementação do analisador sintático e todas as suas especificações.

Formato de Entrega

Os alunos deverão entregar:

- **Documento PDF** com a especificação da linguagem criada.
- **Código-fonte** do analisador léxico e sintático.
- **Exemplos de código** na nova linguagem, junto com a saída gerada.

Meios de Envio

- **GitHub** (repositório público ou privado compartilhado).
- **Arquivo ZIP** enviado pelo sistema acadêmico.
