UNIVERSIDADE FEDERAL DE LAVRAS

Bacharelado em Ciência da Computação - 10A

TRABALHO PRÁTICO DE COMPILADORES

ENTREGA 1 e 2

GUSTAVO COSTA ALMEIDA,
HENRIQUE CÉSAR SILVA SOARES,
PEDRO MILITÃO MELLO REIS

Lavras
2025

1 INTRODUÇÃO

A análise léxica é a etapa inicial do processo de compilação, responsável por transformar o código-fonte em uma sequência de unidades significativas chamadas tokens. Nesta primeira fase do trabalho prático da disciplina Compiladores, foi implementado um analisador léxico utilizando a ferramenta Flex, voltado para uma linguagem simplificada inspirada em C. Essa linguagem contempla declarações de tipos básicos, identificadores, literais inteiros, operadores relacionais, lógicos e aritméticos, além de palavras-chave específicas e símbolos de pontuação. O objetivo principal do projeto foi reconhecer corretamente os tokens válidos da linguagem, reportar erros léxicos de maneira clara indicando linha e coluna em que ocorrem, desconsiderar espaços em branco e comentários e construir uma tabela de símbolos organizada. Essa implementação tem como finalidade consolidar, na prática, os conceitos estudados sobre a etapa de análise léxica dos compiladores.

2 DECISÕES DO PROJETO

O projeto foi desenvolvido em Flex, explorando a expressividade das expressões regulares para a definição dos padrões de tokens. Optou-se por utilizar uma tabela de símbolos implementada como tabela hash com encadeamento, de forma a permitir inserções e buscas rápidas, evitando duplicações de identificadores. A saída do analisador foi organizada em um formato de tabela com uso de cores ANSI, mostrando em cada linha a posição (linha e coluna), o tipo do token e o lexema correspondente. Esse cuidado buscou tornar o resultado mais legível e próximo de ferramentas reais de compiladores. A detecção de erros léxicos também recebeu atenção: quando um número possui sufixo inválido, como no caso de “10abc”, o analisador não divide em múltiplos tokens, mas emite um erro único mais compreensível para o usuário. Já caracteres não reconhecidos, como “@” ou “#”, também geram mensagens de erro com a posição exata no código. Para lidar com a posição dos tokens, foi necessário complementar o contador de linhas fornecido pelo Flex com uma variável própria de coluna, atualizada a cada lexema lido. Comentários de linha única e múltiplas linhas foram tratados de forma a serem ignorados durante a análise. Todas essas escolhas tiveram como finalidade manter o analisador robusto, claro e próximo da prática profissional.

3 DIFICULDADES ENCONTRADAS

Durante o desenvolvimento do analisador léxico algumas dificuldades se destacaram. A primeira foi o controle da coluna, já que o Flex fornece diretamente apenas a linha. Foi preciso implementar manualmente uma lógica para rastrear a posição horizontal de cada lexema. Outra dificuldade ocorreu no reconhecimento de inteiros negativos. A definição da regra deveria permitir que o sinal fosse tratado como parte do número e não como um operador separado, o que exigiu ajustes nas expressões regulares. O tratamento de comentários de múltiplas linhas também trouxe desafios, pois foi necessário atualizar corretamente a contagem de linhas e colunas dentro deles. Além disso, a definição de regras

de erro específicas, como a detecção de números malformados em vez de apenas separar em tokens válidos, exigiu atenção para manter a clareza das mensagens ao usuário.

4 DIAGRAMAS DE TRANSIÇÃO

Arquivos anexados dentro da pasta diagramas do zip enviado.

5 ARQUIVO DE TESTE E SAÍDA

Arquivos anexados dentro da pasta testes do zip enviado.

6 CONCLUSÃO

A implementação do analisador léxico em Flex permitiu aplicar de forma prática os conceitos estudados na disciplina e compreender a importância dessa etapa no processo de compilação. O projeto atendeu aos requisitos propostos, reconhecendo todos os tokens definidos, ignorando espaços em branco e comentários, construindo e imprimindo uma tabela de símbolos e reportando erros léxicos de maneira clara e precisa. A elaboração dos diagramas de transição foi útil para visualizar e confirmar o funcionamento dos autômatos que dão suporte às regras de reconhecimento. As dificuldades enfrentadas durante o desenvolvimento, como o controle da coluna, a manipulação de inteiros negativos e o tratamento de comentários e erros, contribuíram para uma compreensão mais profunda dos desafios envolvidos na construção de compiladores. Os testes realizados confirmaram a robustez do analisador, tanto na identificação de tokens válidos quanto na detecção de entradas malformadas. Dessa forma, a Etapa 1 do trabalho prático cumpriu seu papel no aprendizado, consolidando os fundamentos da análise léxica e preparando terreno para as próximas fases do compilador.