UNIVERSIDADE TIRADENTES
CIÊNCIA DA COMPUTAÇÃO

ALAN REIS ANJOS
ANTHONY RAMOS DOS SANTOS
JOSÉ HENRIQUE OLIVEIRA DE CARVALHO

COMPILADOR QUIZLANG

Aracaju - SE
2025

ALAN REIS ANJOS
ANTHONY RAMOS DOS SANTOS
JOSÉ HENRIQUE OLIVEIRA DE CARVALHO

COMPILADOR QUIZLANG

RELATÓRIO DA IMPLEMENTAÇÃO DO COMPILADOR

ATIVIDADE sobre Compiladores apresentado como requisito parcial da avaliação da disciplina Compiladores, ministrada pela Prof. Layse, no 2º semestre de 2025.

Aracaju - SE
2025

Sumário

1 INTRODUÇÃO

A compilação possui 4 etapas: A análise léxica, análise sintática, análise semântica e geração de código. Ao longo da disciplina de compiladores, aprendemos os processos da compilação e seus algoritmos. Neste trabalho, implementamos um compilador para uma linguagem fictícia, a QuizLang.

2 JUSTIFICATIVA

A implementação de um compilador para a linguagem fictícia *QuizLang* justifica-se pela necessidade de aplicar, de forma prática, os conceitos teóricos aprendidos ao longo da disciplina de Compiladores. O desenvolvimento desse projeto permite compreender de maneira aprofundada as etapas que compõem o processo de compilação — análise léxica, sintática, semântica e geração de código —, além de reforçar habilidades em lógica e estrutura de dados. Dessa forma, o trabalho contribui para a consolidação do conhecimento sobre o funcionamento interno de linguagens de programação e a construção de ferramentas de tradução de código.

3 OBJETIVOS

3.1 OBJETIVO GERAL

Desenvolver um compilador funcional para a linguagem fictícia *QuizLang*, compreendendo e aplicando as principais etapas do processo de compilação — análise léxica, análise sintática, análise semântica e geração de código — de forma a consolidar o conhecimento teórico adquirido na disciplina de Compiladores.

3.2 OBJETIVOS ESPECÍFICOS

1. Implementar o analisador léxico responsável por analisar os tokens da linguagem;
2. Implementar o analisador sintático, responsável por classificar a estrutura gramatical da linguagem;
3. Implementar o analisador semântico, responsável por classificar os tipos, variáveis e escopos utilizados dentro da linguagem;
4. Implementar uma simulação de compilação para testar o código produzido pela *QuizLang*;
5. Documentar o processo

4 METODOLOGIA

A implementação do compilador para a **QuizLang** foi conduzida de forma incremental, abrangendo as fases léxica, sintática e semântica, com simulação prática e validação em código. O desenvolvimento foi realizado em **Python**, utilizando o **ANTLR4** para a geração automática dos analisadores.

4.0.1 Ferramentas e Estrutura de Desenvolvimento

A gramática da linguagem foi descrita no arquivo `QuizLang.g4` e processada pelo **ANTLR4**, que gerou automaticamente o lexer e o parser em Python. Essa abordagem permitiu concentrar os esforços na fase semântica, a mais complexa do compilador. O código foi modularizado em componentes distintos para cada fase da compilação, favorecendo manutenção e clareza de execução.

4.0.2 Análise Léxica

O analisador léxico foi projetado para percorrer o código-fonte caractere por caractere, agrupando símbolos em **tokens** válidos conforme as regras da gramática. Os principais tipos de tokens definidos foram:

- **Palavras-chave:** `quiz`, `titulo`, `tempo`, `secao`, `mcq`, `pergunta`, `opcoes`, `resposta`, `discursiva`, `palavras`, `numerica`, `intervalo`;
- **Identificadores:** `ID` (nomes de quizzes, seções e questões);
- **Literais:** `STRING` e `NUMBER`;
- **Delimitadores e operadores:** `{ }`, `[ ]`, `:`, `,`, `-`;
- **Comentários e espaços em branco:** ignorados pelo lexer.

A validação desta fase foi realizada através de arquivos de teste (`.txt`) contendo códigos QuizLang diversos, verificando a correta classificação e filtragem dos tokens.

4.0.3 Análise Sintática

Com base na sequência de tokens, o analisador sintático verificou se o código obedecia à estrutura definida pela gramática formal da QuizLang. Durante o processo, foram identificadas e corrigidas ambiguidades e estruturas inválidas na gramática inicial, como o uso incorreto de chaves {} para repetição e regras de lista pouco intuitivas.

Esses problemas foram resolvidos com operadores padrão do ANTLR:

- Substituição de {...} por `+` (um ou mais) e `*` (zero ou mais);
- Reescrita da regra de listas (`opcoes`) para `[ OPCAO (',' OPCAO)* ]`;
- Garantia de que cada **quiz** contenha pelo menos uma seção e cada **seção**, ao menos uma questão.

O resultado foi a geração de uma **árvore sintática** (*Parse Tree*) representando a hierarquia do código-fonte, possibilitando a navegação e validação estrutural.

4.0.4 Análise Semântica

A análise semântica foi implementada com o padrão de projeto **Visitor**, utilizando a classe `QuizLangVisitor` para percorrer a árvore sintática e aplicar verificações de coerência lógica. Essa fase incluiu a implementação de uma **Tabela de Símbolos** (`SymbolTable`) para gerenciar escopos e identificadores.

A estrutura da tabela é um dicionário com chaves compostas por `(escopo, nome_do_símbolo)`, acompanhado de uma pilha de escopos (`scope_stack`) para controlar o contexto atual. As validações implementadas incluíram:

- **Redefinição de identificadores:** impede duplicações dentro do mesmo escopo;
- **Questões de múltipla escolha (mcq):** a resposta correta deve estar entre as opções, que não podem se repetir, e deve haver ao menos duas;
- **Questões numéricas (numerica):** o valor inicial do intervalo não pode ser maior que o final;
- **Questões discursivas (discursiva):** o limite de palavras deve ser um número positivo;
- **Avisos:** o compilador emite *warnings* para seções declaradas sem questões.

4.0.5 Simulação e Testes

Para validar o compilador, foram criados arquivos de teste em QuizLang contendo quizzes completos e exemplos de erro. Esses testes permitiram verificar o funcionamento conjunto das três fases, incluindo:

1. Identificação correta e categorização dos tokens;
2. Construção da árvore sintática de acordo com a gramática revisada;
3. Aplicação das validações semânticas via *Visitor*.

Durante as simulações, foram detectados e corrigidos problemas de ambiguidade e inconsistência gramatical, assegurando robustez e confiabilidade ao compilador.

4.0.6 Resultados da Implementação

A simulação demonstrou que o compilador da QuizLang é capaz de:

- Processar integralmente arquivos `.quizlang`;
- Detectar e relatar erros léxicos, sintáticos e semânticos;
- Emitir avisos sem interromper a execução;
- Produzir uma representação estruturada do quiz (árvore de dados em Python), passível de exportação para formatos como JSON, XML ou HTML.

Os resultados obtidos comprovam a eficácia do compilador desenvolvido e consolidam o aprendizado prático sobre os princípios de análise léxica, sintática e semântica na construção de linguagens de domínio específico.

5 RESULTADOS E DISCUSSÕES

A implementação do compilador para a linguagem **QuizLang** foi concluída com sucesso, contemplando as fases de análise léxica, sintática e semântica. Cada etapa foi validada de forma incremental, com testes unitários e simulações de código real, permitindo avaliar a robustez da gramática, a coerência do modelo semântico e a integração entre os componentes.

5.0.1 Resultados da Análise Léxica e Sintática

Durante o desenvolvimento do analisador léxico, foi possível identificar e classificar corretamente todos os tokens definidos na gramática `QuizLang.g4`. A ferramenta **ANTLR4** mostrou-se eficiente na geração do lexer e parser, reduzindo significativamente a complexidade de implementação manual.

Os testes realizados confirmaram que o lexer reconhece adequadamente palavras-chave, identificadores, literais numéricos e strings, além de ignorar corretamente espaços e comentários. Erros de tokenização foram tratados com mensagens de saída descritivas, indicando a linha e a posição do caractere inválido.

Na fase sintática, a validação da gramática foi fundamental para eliminar ambiguidades e permitir a correta geração da *Parse Tree*. A reescrita de regras com os operadores padrão `+` e `*` garantiu que o compilador reconhecesse a estrutura mínima obrigatória da linguagem ou seja, que todo quiz possua ao menos uma seção e que cada seção contenha uma ou mais questões.

Os testes de parsing demonstraram que a árvore sintática é gerada de forma hierárquica, refletindo a organização declarativa dos elementos da QuizLang. Entradas com estruturas incompletas ou malformadas geraram erros sintáticos claros e precisos, comprovando a confiabilidade do parser.

5.0.2 Resultados da Análise Semântica

A análise semântica foi o ponto mais sofisticado do projeto, responsável por verificar o significado lógico das declarações. A implementação da classe `QuizLangVisitor` e da `SymbolTable` possibilitou o controle preciso dos escopos e identificadores da linguagem.

O compilador foi capaz de identificar e relatar corretamente situações de inconsistência, tais como:

- Redefinição de identificadores dentro do mesmo escopo;
- Questões de múltipla escolha com respostas inválidas ou opções duplicadas;

- Questões numéricas com intervalos incorretos (valor inicial maior que o final);
- Questões discursivas com limite de palavras inválido;
- Seções declaradas sem questões, gerando *warnings* sem interromper a execução.

Os resultados confirmam a coerência da abordagem baseada no padrão **Visitor**, que permitiu percorrer a árvore sintática e aplicar validações semânticas de forma organizada e extensível. A Tabela de Símbolos demonstrou eficiência na gestão de escopos aninhados, prevenindo conflitos de nomes entre seções e questões.

5.0.3 Simulações e Casos de Teste

Foram criados diversos arquivos de entrada com estruturas válidas e inválidas da QuizLang, de modo a testar o comportamento completo do compilador. Nos casos válidos, o compilador percorreu todas as etapas sem interrupções, gerando uma representação estruturada do quiz em memória, pronta para exportação. Nos casos inválidos, foram emitidas mensagens detalhadas indicando a natureza e a localização do erro, seja ele léxico, sintático ou semântico.

Essas simulações mostraram que o compilador responde corretamente a diferentes cenários, reforçando sua robustez e aderência às regras da gramática. Além disso, a execução das validações semânticas garantiu integridade lógica ao conteúdo dos quizzes, assegurando que apenas estruturas válidas sejam aceitas.

5.0.4 Discussão dos Resultados

A análise dos resultados evidencia que o compilador atingiu plenamente os objetivos definidos no projeto. A combinação do **ANTLR4** com a linguagem **Python** proporcionou uma arquitetura flexível, escalável e fácil de depurar. A definição modular das fases do compilador permitiu uma evolução incremental e reduziu significativamente o tempo de desenvolvimento.

Os principais desafios enfrentados estiveram na definição precisa da gramática e no tratamento de ambiguidades, especialmente durante a transição entre as fases sintática e semântica. Contudo, a solução adotada com uso de operadores formais e gerenciamento explícito de escopos eliminou os problemas e resultou em uma análise semântica confiável.

Em síntese, o compilador para QuizLang apresentou desempenho satisfatório, identificando e relatando erros de forma clara, e validando corretamente os elementos da linguagem. A metodologia empregada se mostrou eficaz e replicável em projetos futuros que envolvam o desenvolvimento de linguagens de domínio específico.

6 CONSIDERAÇÕES FINAIS

6.1 Considerações Finais

O desenvolvimento do compilador para a linguagem **QuizLang** proporcionou uma compreensão aprofundada dos princípios teóricos e práticos envolvidos no processo de compilação. A implementação das fases léxica, sintática e semântica permitiu vivenciar, de forma aplicada, os conceitos de análise de linguagem, construção de gramáticas e verificação de consistência semântica.

A utilização do **ANTLR4** em conjunto com o **Python** demonstrou-se uma escolha acertada, pois simplificou a geração automática dos analisadores e possibilitou uma implementação modular e extensível. A adoção do padrão de projeto *Visitor* foi essencial para a clareza e a escalabilidade da fase semântica, permitindo uma abordagem organizada para percorrer e validar a árvore sintática.

Durante o desenvolvimento, os principais desafios estiveram relacionados à definição precisa da gramática e ao tratamento de ambiguidades, que impactaram diretamente a fase de parsing. A resolução desses problemas fortaleceu a robustez do compilador e garantiu a consistência entre as fases. Além disso, a criação e manutenção da *Symbol Table* mostraram-se fundamentais para o gerenciamento de escopos, reforçando a importância das boas práticas de organização e validação semântica em compiladores.

Os resultados obtidos evidenciam o sucesso do projeto em atender aos requisitos propostos: o compilador é capaz de identificar tokens, validar a estrutura sintática e garantir a coerência semântica de programas escritos em QuizLang, emitindo erros e avisos de forma precisa e compreensível.

Como trabalhos futuros, destaca-se a possibilidade de expandir o compilador para incluir uma fase de **geração de código**, permitindo exportar quizzes para formatos estruturados como JSON, XML ou HTML. Também é viável a criação de uma interface gráfica que possibilite o carregamento e validação de arquivos QuizLang de forma interativa, ampliando o alcance e a aplicabilidade da ferramenta.

Em suma, o projeto consolidou conhecimentos essenciais sobre o funcionamento interno de compiladores e reforçou a relevância das técnicas de análise léxica, sintática e semântica na construção de linguagens formais. A experiência prática obtida contribuiu significativamente para o domínio de conceitos fundamentais da engenharia de software e para a capacidade de projetar linguagens de domínio específico de maneira estruturada e eficiente.

7 REFERÊNCIAS

7.1 referencias

AHO, Alfred V.; LAM, Monica S.; SETHI, Ravi; ULLMAN, Jeffrey D. **Compiladores: princípios, técnicas e ferramentas**. 2. ed. São Paulo: Pearson, 2009.

PARR, Terence. **The Definitive ANTLR 4 Reference**. 2. ed. Dallas: Pragmatic Bookshelf, 2013. Disponível em: `https://www.antlr.org/`. Acesso em: 13 nov. 2025.

ANTLR. **ANTLR 4 Documentation**. Disponível em: `https://github.com/antlr/antlr4/blob/master/doc/index.md`. Acesso em: 13 nov. 2025.