1 Instruções Importantes

Nesta seção são apresentadas diversas informações relevantes referentes a entrega do trabalho e orientações a serem seguidas durante a implementação do mesmo. Leia atentamente antes de começá-lo.

1.1 Equipe de Desenvolvimento

O trabalho será desenvolvido individualmente ou em dupla.

1.2 Escolha da Linguagem de Programação

O trabalho poderá ser desenvolvido em qualquer uma das linguagens a seguir, a seu critério: Java, Haskell, Scala, Elixir, Scheme, Racket, Rusty, Go, C ou C++. No que concerne a implementação, todas as estruturas de dados referente a modelagem e desenvolvimento do trabalho deverão ser implementadas. Poderão ser usadas apenas bibliotecas de estruturas de dados elementares (listas, filas, pilhas, árvores, etc.), leitura de strings e arquivos.

1.3 Artefatos a Serem Entregues

Os artefatos a serem entregues são:

- código fonte do programa;
- arquivo de *build* para compilação do fonte (makefile, ant, shell script, etc);
- relatório do trabalho em formato pdf

Antes de enviar seu trabalho para avaliação, assegure-se que:

1. seu código compila e executa em ambiente Unix / Linux. Programas que não compilam receberão nota zero;
2. todos os fontes a serem enviados têm, em comentário no início do arquivo, nome e matrícula do(s) autor(es) do trabalho;
3. arquivo do relatório tenha a identificação do(s) autor(es) do trabalho.

1.4 Critérios de Avaliação

A avaliação será feita mediante análise do código fonte, relatório e apresentação do trabalho (entrevista). Os seguintes fatores serão observados na avaliação do código fonte: corretude do programa, estrutura do código, redigibilidade e legibilidade. A corretude se refere à implementação correta de todas as funcionalidades especificadas, i.e., se o programa desenvolvido está funcionando corretamente e não apresenta erros. Os demais fatores avaliados no código fonte são referentes a organização e escrita do trabalho. Nesse quesito, será observado:

- estruturas de dados bem planejadas;
- código modularizado em nível físico (separação em arquivos) e lógico;
- legibilidade do código, i.e., nomes mnemônicos de variáveis, funções e comentários úteis no código;

O relatório do código deve conter informações relevantes para compilar, executar e auxiliar no entendimento da estratégia adotada para resolução do problema e do código fonte. Ressalta-se que no relatório não deve conter cópias do fonte – afinal o seu fonte é um dos artefatos entregue –, porém trechos de códigos podem ser incluídos caso facilite a explicação e elucidação das estratégias implementadas. O relatório deve apresentar as decisões de projetos tomadas: i) analisador léxico – expressões regulares e autômatos para os *tokens* da linguagem, estruturas de dados usadas, estratégia de implementação do analisador léxico e de "bufferização" da entrada; ii) analisador sintático – ferramentas utilizada com um breve descrição e estratégia de implementação do analisador sintático (LL, LR, PEG e separação entre léxico e sintático); iii) analisador semântico – estruturas de dados usadas para análise e verificação de tipos e estratégia usada para verificar a semântica estática da linguagem (*static semantics*); iv) interpretador – estratégia de interpretação, estruturas de dados usadas e representação da árvore de sintaxe abstrata; v) geração de código – representação intermediária usada para gerar código objeto, estruturas de dados auxiliares, estratégia usada na geração de código, dentre outras informações. Faça uso de exemplos para apresentar a sua solução.

O trabalho deverá ser apresentado ao professor da disciplina e, só serão avaliados após a realização da entrevista, i.e., trabalhos que não forem apresentados não terão nota. Na entrevista, o discente deverá elucidar, ao menos, como modelou e resolveu o problema, os resultados e conclusões obtidas. A entrevista também tem a finalidade de avaliar a confiabilidade e segurança do autor do código em explicar pontos relevantes do trabalho desenvolvido.

Assim, a entrevista influenciará na avaliação dos artefatos entregues. Portanto, a nota final será dada a partir da avaliação do conjunto do código fonte, relatório e entrevista. **É de responsabilidade do discente solicitar a marcação do dia e horário da entrevista com o professor da disciplina após a entrega do trabalho**.

Atrasos serão penalizados por uma função exponencial de dias de atrasos, i.e., será reduzido do percentual da nota a exponencial na base 2 dos dias de atraso. A tabela a seguir mostra a nota em função dos dias de atraso:

Dias de Atraso	Nota
1	n*0.98
2	n*0.96
3	n*0.92
4	n*0.84
5	n*0.68
6	n*0.36
7	0

Observe que a partir do 7º dia de atraso seu trabalho não será mais avaliado.

2 Especificação Técnica do Trabalho

Deverá ser implementado um compilador para a linguagem *lang*. O software desenvolvido deverá ser executado via linha de comando com 2 parâmetros: uma diretiva indicando a ação do compilador e o caminho do arquivo contendo o programa *lang*.

A Tabela 1 lista cada uma das diretivas do compilador, bem como a descrição de sua função e formato da saída.

O desenvolvimento e, consequentemente, o código do compilador deve ser estruturado na arquitetura clássica, contendo as seguintes fases:

- analisador léxico e sintático;

Diretiva	Descrição	Saída
-syn	Executa a análise sintática do programa	**accept** para programa corretos e **reject** para inválidos
-i	Interpreta o programa	saída conforme o programa de entrada
-t	Executa a verificação de tipos do programa	**accept** para programa bem-tipados e **reject** para inválidos
-src	Gera código de alto nível	programa equivalente na linguagem alvo
-gen	Gera código para uma arquitetura alvo	assembler na arquitetura alvo

Tabela 1: Diretivas do compilador

- interpretador;
- analisador semântico;
- geração de código *source-to-source*; e
- geração de código de baixo nível.

Para a implementação dos analisadores léxico/sintático pode ser usado qualquer gerador de analisadores sintáticos, podendo o léxico e sintático serem integrados. Na geração *source-to-source* (diretiva -src) poderá ser usada qualquer linguagem de alto nível como alvo. A geração de código de baixo nível deverá ser realizada para a Java Virtual Machine (JVM) via Jasmin. Ressalta-se que o código gerado (arquivo de saída), em ambas as gerações, devem ser programas válidos, i.e., compilam e executam.

3 Entrega do Trabalho e Apresentação

A entrega do trabalho será dividida em duas: a primeira com a implementação do analisador sintático e o interpretador (diretivas *-syn* e *-i*); e a segunda com a implementação da fases restantes, analisador semântico e geração de código *source-to-source* e baixo nível (diretivas *-t*, *-src* e *-gen*). Todas as entregas devem satisfazer os requisitos descritos na Seção 1.3. Todas as entregas serão realizadas na plataforma do Google Classroom. A seguir estão os detalhes de cada uma das entregas:

- **1ª Entrega**
  - **Data da entrega:** 04 de julho de 2025
  - **Data limite da apresentação:** 18 de julho de 2025
  - **Valor:** 35
- **2ª Entrega**
  - **Data da entrega:** 15 de agosto de 2025
  - **Data limite da apresentação:** 25 de agosto de 2025
  - **Valor:** 40