

1. Introdução: Visão Geral do Projeto

Cabeçalho e documentação online

<https://gsdvl.github.io/DocumentacaoLinguagem/>

Disciplina: Engenharia de Linguagens - DIM0548

Integrantes do Grupo: - Gabriel Victor de Lima Pimentel

- Zeus Justino de Lima
- Lucas Nogueira Cortez
- Gabriel Soares de Vasconcelos Lira
- Débora Noemy de Alcântara Valentim

Visão geral

Este documento detalha o projeto e a implementação da G2DL, uma nova linguagem de programação imperativa desenvolvida para a disciplina DIM0548 - Engenharia de Linguagens da Universidade Federal do Rio Grande do Norte. Com foco particular na área de Ciência de Dados e Aprendizado de Máquina, a linguagem foi cuidadosamente planejada para equilibrar legibilidade, facilidade de escrita, confiabilidade e custo, visando um sistema robusto e eficiente para seu domínio de aplicação.

A G2DL apresenta uma sintaxe clara e expressiva, suporta tipagem estática (com tipos determinados em tempo de compilação para maior segurança), e oferece suporte a tipos de dados abstratos, como dataframes e tensores, com sobrecarga de operadores para manipulação eficiente. Sua concepção híbrida é inspirada na sintaxe limpa e expressividade do Python, na eficiência e robustez de C e GoLang, e nas estruturas de dados primitivas do R. O sistema de implementação idealizado é a interpretação, crucial para a exploração rápida de dados e prototipagem no domínio de Ciência de Dados.

Atualmente, o design da linguagem impõe certas limitações deliberadas para otimização de recursos e clareza. Não há intenção de implementar mecanismos para alocação explícita de variáveis de escopo local na heap, e o tratamento de aliasing é delegado à responsabilidade do programador para poupar recursos. O tratamento completo de exceções é uma funcionalidade que ainda está em desenvolvimento. O projeto abrange as fases principais de um compilador, incluindo a análise léxica e a análise sintática, esta última com a construção de um analisador LALR bottom-up.

Palavras Reservadas da Linguagem G2DL

Palavras reservadas são identificadores especiais que **não podem ser usados como nomes de variáveis, funções ou objetos**, pois possuem um **significado fixo** na linguagem. Elas fazem parte da **gramática principal** e são reconhecidas diretamente pelo analisador léxico.

Abaixo está a lista completa das palavras reservadas da linguagem G2DL, junto com uma descrição do papel de cada uma.

Controle de Fluxo

Palavra	Função
<code>if</code>	Inicia uma estrutura condicional. Executa um bloco se a condição for verdadeira.
<code>else</code>	Bloco alternativo executado se a condição do <code>if</code> for falsa.
<code>while</code>	Laço que executa um bloco enquanto a condição for verdadeira.
<code>for</code>	Estrutura de repetição com inicialização, condição e incremento definidos.
<code>break</code>	Interrompe imediatamente a execução do laço atual (<code>for</code> ou <code>while</code>).

Funções e Retornos

Palavra	Função
<code>function</code>	Define uma função com nome, parâmetros e corpo.
<code>return</code>	Encerra a função e opcionalmente retorna um valor.

Estrutura da AST (`ast.c`)

Este módulo implementa as funções de construção e gerenciamento da Árvore de Sintaxe Abstrata (AST) da linguagem. Ele é responsável por representar a estrutura lógica do código-fonte após a análise sintática, permitindo que fases posteriores como análise semântica e interpretação/execução sejam realizadas.

Estrutura Geral

- **Arquivo:** `ast.c`
- **Cabeçalhos:** `#include "ast.h"`
- **Uso de memória:** Todas as alocações so feitas via `malloc`, `realloc`, `strdup`, e liberadas com `free`.
- **Origem da linha:** Cada n da AST marcado com a varivel global `yylineno` (linha atual do lexer/parser).

Criação de Nós da AST

Todos os nós são derivados de `AstNode` e criados com uma função base: `static AstNode* create_base_node(AstNodeType type, size_t size);` Ela aloca memoria, atribui o tipo do nó e o número da linha.

Tipos de Nós Literais

Função	Tipo de Nó	Descrição
<code>create_int_literal_node(int)</code>	<code>NODE_TYPE_INT_LITERAL</code>	Representa um número inteiro.
<code>create_float_literal_node(float)</code>	<code>NODE_TYPE_FLOAT_LITERAL</code>	Representa um número de ponto flutuante.

Verificações realizadas

(1) Estrutura Sintática

- **(A)** Introdução e detecção de erros léxicos e sintáticos
- **(B)** Como o programa/estrutura pode ser derivado das regras sintáticas?
- **(C)** Gerar fontes que incluam instruções que originam conflito *shift-reduce*. Ações corretas?

(2) Variáveis

- **(A)** Variáveis omitidas — detectar erros
- **(B)** Variáveis duplicadas no mesmo escopo — detectar erros
- **(C)** Variáveis duplicadas em escopos distintos e não-aninhados — testar acessos
- **(D)** Variáveis duplicadas em escopos distintos e aninhados — testar acessos

(4) Passagem de Parâmetros

- **(A)** Corretamente transmitidos ao subprograma chamado?
- **(B)** Alterados/inalterados corretamente após retorno do subprograma chamado?

Execução

1. Instruções de Uso do Compilador

Comandos Make Disponíveis

Você pode usar os seguintes comandos com make:

Comando	O que faz
make ou make all	Compila todo o projeto e gera bin/g2dl_interpreter
make clean	Remove todos os arquivos gerados, incluindo build/, bin/ e temporários
make debug	Não implementado (mas pode ser usado para recompilar com -g)
make help	Exibe instruções de uso dos comandos do Makefile

Para gerar e utilizar o compilador G2DL, siga os passos abaixo: Passo 1: Gerar o Compilador
Para compilar o código-fonte da linguagem e gerar o executável do compilador, navegue até o diretório raiz do projeto no seu terminal e execute o comando make: Bash \$ make

Esse comando executa os seguintes passos automaticamente:

- Gera o analisador léxico com Flex
- Gera o analisador sintático com Bison
- Compila todos os arquivos .c
- Organiza os .o em build/
- Cria o executável final em bin/g2dl_interpreter

Passo 2: Executar o Compilador Após a compilação, o executável do compilador será gerado. Você pode utilizá-lo para processar um arquivo de código-fonte da linguagem G2DL. Por exemplo, para compilar o arquivo input.txt, utilize o seguinte comando: Bash \$./compiler_g2dl input.txt

Certifique-se de substituir input.txt pelo nome do arquivo de código-fonte que deseja compilar.
Passo Extra: Limpar o Compilador Para remover os arquivos gerados durante o processo de