

1. Introdução: Visão Geral do Projeto

Este documento detalha o projeto e a implementação de uma nova linguagem de programação imperativa, desenvolvida para a disciplina DIM0548 - Engenharia de Linguagens da Universidade Federal do Rio Grande do Norte. O principal objetivo do projeto é conceber e implementar uma linguagem de programação, com foco particular na área de Ciência de Dados e Aprendizado de Máquina.

A linguagem foi projetada para ter uma sintaxe clara e expressiva, visando a legibilidade do código. Ela oferecerá suporte a tipagem dinâmica ou estática opcional, provendo flexibilidade e segurança. Um aspecto crucial é o suporte a tipos de dados abstratos, como dataframes e tensores, com sobrecarga de operadores para manipulação eficiente, o que é essencial para o domínio de aplicação.

O sistema de implementação ideal para a linguagem proposta é a interpretação, dada a importância da execução interativa para o desenvolvimento e experimentação em Ciência de Dados e Aprendizado de Máquina. Essa abordagem permite a exploração rápida de dados e o desenvolvimento de modelos sem a necessidade de recompilar a cada modificação, acelerando o processo de prototipagem e ajustes.

A linguagem busca ser um híbrido, herdando características de linguagens como C e GoLang, mantendo um propósito semelhante ao LISP, e se posicionando próxima ao Python na árvore genealógica das linguagens de programação. A inspiração do Python se reflete na sintaxe limpa e expressiva, além do suporte à tipagem estática opcional para controle de desempenho e segurança. O projeto visa equilibrar facilidade de uso e expressividade com a eficiência e robustez computacional de linguagens compiladas, tornando-a atrativa tanto para cientistas de dados quanto para desenvolvedores que necessitam de alto desempenho.

O projeto aborda as fases principais de um compilador, incluindo a análise léxica e a análise sintática, com a construção de um analisador sintático bottom-up (LALR). A linguagem também define suas vinculações e sistema de tipos, com foco em tipagem estática e forte.

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.

Sintaxe

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.

G2DL - Interpretador de Linguagem

Compilação

Para compilar o projeto, utilize o comando:

```
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

Execução

Após a compilação, o executável estará disponível em:

bin/g2dl_interpreter

Para executar um programa da linguagem G2DL, use:

```
./bin/g2dl_interpreter arquivo.g2dl
```

Certifique-se de que o main.c está preparado para ler o arquivo de entrada.

Organização

```
├─ Makefile
├─ src/
│   ├── main.c
│   ├── ast/
│   └── lexer/
```