Proposta de Projeto: Compilador Charmeleon

1. Descrição do Projeto

Este projeto consiste no desenvolvimento de um compilador completo para a linguagem de programação **Charmeleon**. A linguagem Charmeleon foi concebida para ser **simples, intuitiva e rápida**, combinando características de linguagens como Python e C#, com um foco inicial no **desenvolvimento web**. O compilador transpila o código-fonte escrito em Charmeleon para código Python executável, permitindo que programas Charmeleon sejam executados no vasto ecossistema Python.

O compilador implementa um pipeline de processamento que inclui as seguintes fases:

- **Análise Léxica:** Converte o código-fonte em uma sequência de tokens.
- **Análise Sintática:** Constrói uma Árvore de Sintaxe Abstrata (AST) a partir dos tokens, verificando a conformidade gramatical.
- **Análise Semântica:** Realiza a checagem de tipos e o gerenciamento de escopo, garantindo a correção lógica do programa.
- **Geração de Código Intermediário (IR):** Transforma a AST em uma representação de baixo nível, independente da plataforma.
- **Otimização de Código (DCE):** Aplica a técnica de Eliminação de Código Morto (Dead Code Elimination) para remover instruções desnecessárias, tornando o código mais eficiente.
- **Análise de Segurança Estática (SAST):** Identifica proativamente vulnerabilidades de segurança comuns, como exposição de dados sensíveis e injeção por concatenação de strings.
- **Geração de Código Alvo:** Transpila o IR otimizado para código Python, que pode ser executado diretamente.

2. Objetivos do Projeto

Os principais objetivos do desenvolvimento do compilador Charmeleon foram:

- **Criar uma Linguagem de Programação Acessível:** Desenvolver uma linguagem com sintaxe clara e fácil de aprender, que combine a simplicidade do Python com a estrutura do C#.

- **Construir um Compilador Funcional e Robusto:** Implementar todas as fases essenciais de um compilador, garantindo a correta transformação do código-fonte Charmeleon em código Python executável.
- **Melhorar a Eficiência do Código Gerado:** Incorporar otimizações, como a Eliminação de Código Morto, para produzir código Python mais performático e enxuto.
- **Promover a Segurança no Desenvolvimento:** Integrar funcionalidades de Análise de Segurança Estática (SAST) para identificar e alertar sobre potenciais vulnerabilidades de segurança desde as etapas iniciais do desenvolvimento.
- **Garantir a Qualidade através de Testes:** Desenvolver uma suíte de testes de ponta a ponta para validar a correção de cada fase do compilador e a funcionalidade geral da transpilação.
- **Fornecer Documentação Abrangente:** Elaborar uma documentação detalhada que descreva o design da linguagem, a arquitetura do compilador e o funcionamento de cada um de seus componentes.

3. Tecnologias e Ferramentas Utilizadas

O projeto foi desenvolvido utilizando as seguintes tecnologias e ferramentas:

- **Linguagem de Implementação:** Python 3.11
  - Utilizado para desenvolver todas as fases do compilador (lexer, parser, analisadores, geradores e otimizador).
  - A flexibilidade e o rico ecossistema do Python foram cruciais para o desenvolvimento rápido e eficiente.
- **Linguagem de Destino (Transpilação):** Python
  - O compilador Charmeleon transpila o código-fonte para Python, aproveitando a portabilidade e a vasta gama de bibliotecas e frameworks disponíveis no ecossistema Python para desenvolvimento web.
- **Ferramentas de Teste:** `unittest` (módulo padrão do Python)
  - Utilizado para criar e executar os testes de ponta a ponta, garantindo a validação e a qualidade do compilador.
- **Expressões Regulares:** Módulo `re` (Python)
  - Essencial para a fase de Análise Léxica, permitindo a definição e o reconhecimento de padrões de tokens no código-fonte.
- **Estruturas de Dados:** Listas, dicionários e classes personalizadas (para AST e IR)
  - Utilizadas para representar e manipular os tokens, a Árvore de Sintaxe Abstrata (AST), o Código Intermediário (IR) e as tabelas de símbolos.

Este conjunto de tecnologias e ferramentas permitiu a construção de um compilador robusto e funcional, capaz de transformar o código Charmeleon em aplicações Python eficientes e seguras.

4. Cronograma de Desenvolvimento (com marcos importantes)

- Agosto – Início do Projeto

• Definição do escopo detalhado do compilador.

• Estudo das linguagens de referência (Python e C#).

• Elaboração da gramática inicial da linguagem Charmeleon.

- Setembro – Primeira Fase Técnica

• Implementação da Análise Léxica (tokenização) e primeiros testes unitários.

• Desenvolvimento da Análise Sintática e construção da Árvore de Sintaxe Abstrata (AST).

• Implementação da Análise Semântica (checagem de tipos e escopo).

- Outubro – Segunda Fase Técnica

• Geração do Código Intermediário (IR) e início da implementação de estruturas de otimização.

• Implementação da Otimização de Código (DCE) e integração com o fluxo do compilador.

• Adição do módulo de Análise de Segurança Estática (SAST).

- Novembro – Finalização e Preparação

• Aprimoramento da transpilação para Python.

• Testes de ponta a ponta e ajustes finais.

• Elaboração da documentação completa.

• Preparação da apresentação oficial para 27 de novembro.

5. Distribuição de Responsabilidades

- Integrante 1 – Definição do escopo detalhado, estudo das linguagens de referência e elaboração da gramática inicial da linguagem Charmeleon.

- Integrante 2 – Implementação da Análise Léxica (tokenização) e primeiros testes unitários.

- Integrante 3 – Desenvolvimento da Análise Sintática e construção da Árvore de Sintaxe Abstrata (AST).

- Integrante 4 – Implementação da Análise Semântica (checagem de tipos e escopo).

- Integrante 5 – Geração do Código Intermediário (IR), otimização de código (DCE) e integração com o fluxo do compilador.

- Integrante 6 – Implementação da Análise de Segurança Estática (SAST), documentação completa e preparação da apresentação final.

Obs.: As etapas de testes e revisão final serão realizadas de forma colaborativa por todos os integrantes do grupo.