Universidade Tuiuti do Paraná

Credenciada por Decreto Presidencial de 07 de julho de 1997
D.O.U. nº 128, de 08 de julho de 1997, Seção 1, página 14295

Bacharelado em Ciência da Computação
Trabalho de Conclusão de Curso

Proposta de Tema

(Apêndice CC1)

1. Título

Desenvolvimento de um interpretador com um ambiente de programação da linguagem D+.

2. Tema

Os compiladores são essenciais para a verificação e a transformação do programa escrito em algo executável. Para ter uma maior interação com o desenvolvedor, é proposto criar uma forma de compilação dinâmica, criando um interpretador, que compile o código em tempo real, e uma interface. A interface auxilia o desenvolvedor mostrando os erros no código de forma visual.

3. Equipe

Aluno: Gabriel Pinto Ribeiro da Fonseca_____ Rubrica: _____
E-mail: gabriel-prdf@hotmail.com_____ Telefone: (41)99607-5380__

4. Orientador

Nome: Diógenes Cogo Furlan_____ Rubrica: _____
E-mail: mestredidi@yahoo.com.br_____ Telefone: (41)99626-2121__

5. Reunião de Orientação

Dia da semana: Quinta-feira _____ Hora: 20:40_____
Local: Universidade Tuiuti do Paraná – Laboratório 06_____

6. Resumo

6.1. Introdução e Justificativa:

O aprendizado na área de compiladores é difícil, por conta da disciplina ter grande volume de conteúdo teórico, pouca pratica e apenas 6 meses de ensino, para o aluno compreender seus conceitos e analisar o passo a passo do compilador. Um interpretador desenvolvido da linguagem D+, auxilia os alunos na absorção do conteúdo, ajudando-os a fixar melhor a matéria da disciplina de compiladores, pois seria a implementação, no desenvolvimento de um compilador. Após inserir um código na interface, o interpretador compilará linha por linha ilustrando o código objeto. Caso exista um erro semântico ou alguma palavra especial esteja incorreta, de forma léxica ou de forma sintática, terá uma destaque com uma cor.

O objetivo deste trabalho é desenvolver uma ferramenta que vai aplicar conteúdos visto em sala de aula na prática, proporcionando ao aluno entender como eles são elaborados

dentro dos processos de analise léxica e sintática, ilustrando a saída do código em cada etapa das análises.

Outra justificativa se dá na área de documentação, os materiais disponíveis sobre o funcionamento, desenvolvimento e aplicação de interpretadores são escassos. Desta forma, o estudo proposto documentará o desenvolvimento de um interpretador com a gramática da linguagem D+, relatando as dificuldades encontradas e soluções obtidas, para ajudar no entendimento do tema.

6.2. Problema Proposto:

Com o desenvolvimento do interpretador da linguagem D+, pode-se questionar:

- Como é possível criar um interpretador?
- É possível tem uma melhor compreensão do processo de compilação, mostrando a entrada do código e saída de cada análise?
- Como é possível mostrar de forma clara os erros de compilação do programa, de forma a auxiliar o programador a solucionar seus erros?
- É possível criar uma interface funcional para um interpretador?
- É possível que a criação de um interpretador para a linguagem D+ auxilie o aluno a entendê-la melhor?

6.3. Objetivos:

O objetivo principal deste projeto é a criação de um interpretador da linguagem D+, que executará junto a uma interface desenvolvida para facilitar o aprendizado do estudante.
Objetivos específicos:

- Compilar um código desenvolvido em tempo real;
- Ilustrar a saída dos estágios de análise léxica e análise sintática.
- Caso o código tenha algum erro de análise léxica ou análise sintática, mostrar em tempo real ao desenvolvedor o erro;
- Facilitar o entendimento para alunos que estejam estudando compiladores;
- Criar a documentação do funcionamento do interpretador

6.4. Trabalhos Relacionados:

Trabalho 1: AMBIENTE DE PROGRAMAÇÃO VISUAL BASEADO EM COMPONENTES
Autora: Juliana Macedo Burigo
Ano: 2015

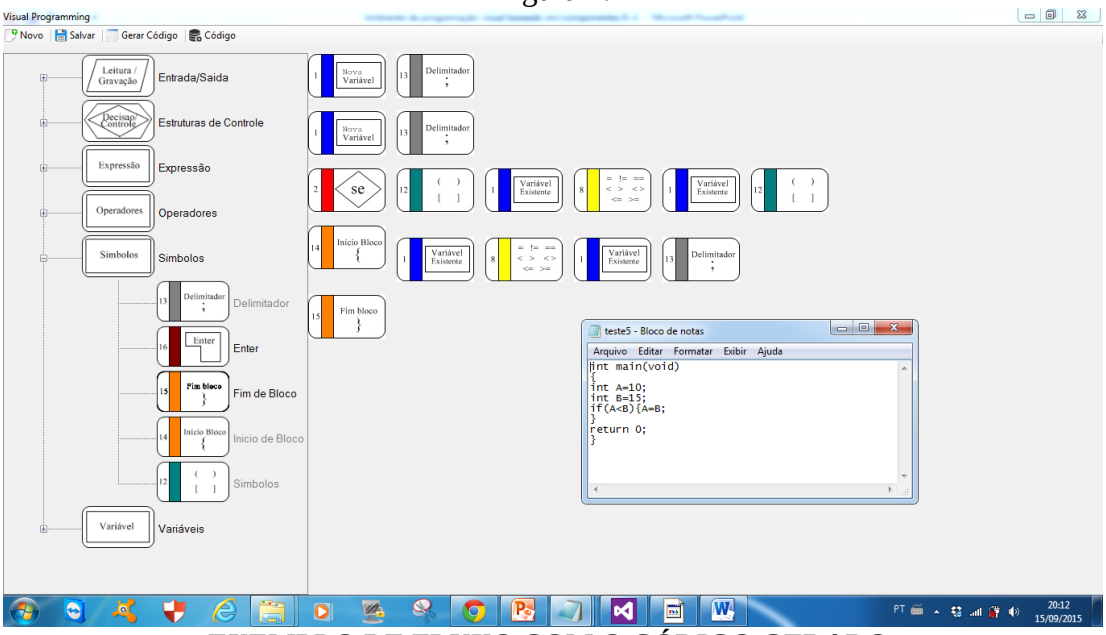
Macedo, J(2015) apresenta um trabalho sobre a criação de um ambiente de desenvolvimento utilizando componentes vindos de fluxogramas, e transformando-os em código da linguagem C, para que auxilie pessoas que estão iniciando na área de programação, e também pessoas que já tem familiaridade na área, mas que não possuem conhecimento da linguagem ao qual o fluxo é convertido.

Neste trabalho foi utilizada a linguagem C#, fluxogramas e o banco de dados SQL Server. A linguagem C#, foi escolhida devido a sua facilidade em criar um ambiente gráfico e de sua administração com imagens como fluxograma.

A utilização de componentes baseados em fluxogramas auxilia no desenvolvimento de um ambiente de programação visual, possibilitando o usuário criar código através destes componentes.

A figura 1 ilustra a interface criada pelo trabalho, tendo o menu esquerdo as funções possíveis de utilização, e a aba da direita sendo o resultado dos conjuntos escolhidos.

Figura 1.



EXEMPLO DE FLUXO COM O CÓDIGO GERADO.
Fonte: AMBIENTE DE PROGRAMAÇÃO VISUAL BASEADO EM COMPONENTES

SQL Server é um gerenciador de dados. Nele foram armazenados os fluxogramas para facilitar o processo de alocação, controle e manipulação, armazenando no final o código de saída do fluxograma montado. (MACED'O. J, 2015).

Foram realizados experimentos com 50 alunos, porém apenas 20% responderam o questionário proposto, e suas respostas informavam que o software chama a atenção para seu uso, porém não era mais simples o seu entendimento, obrigando o usuário a ter um conhecimento de lógica de programação maior do que o desejado, assim não atingindo seu objetivo.

Trabalho 2: SCC: Um Compilador C como Ferramenta de Ensino de Compiladores
Autor(a): Juliano Henrique Foleiss, Guilherme Puglia Assunção, Eduardo Henrique Molina da Cruz, Ronaldo Augusto de Lara Gonçalves, Valéria Delisandra Feltrim
Ano: 2009

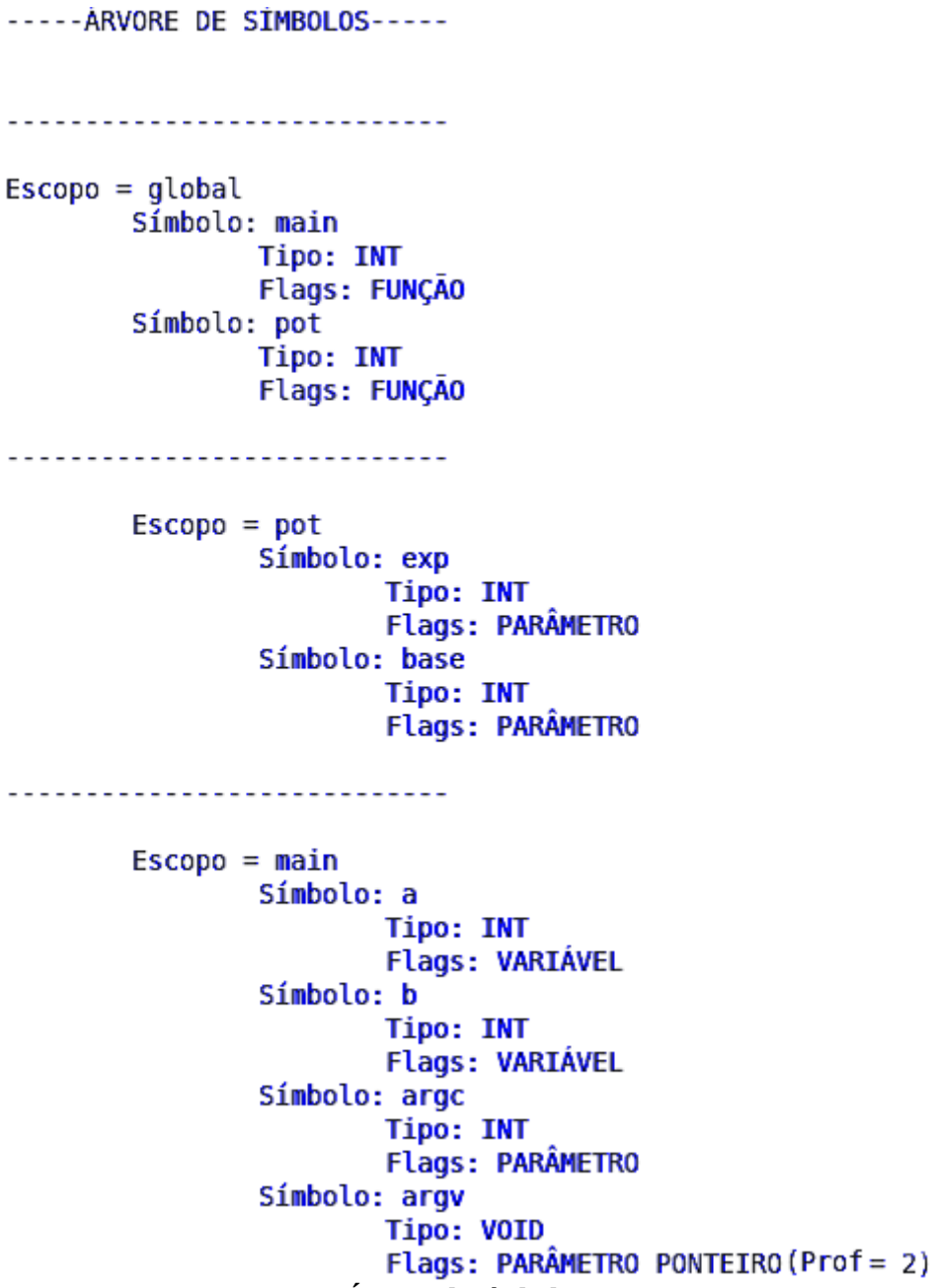
Foleisa, (2009) apresenta um trabalho do desenvolvimento de um compilador que permite criar programas na linguagem C e serem executados com supervisão em tempo real. Estas supervisões funcionam com uma execução detalhada, passo a passo. Com este recurso tem se um auxílio no aprendizado de compiladores, e suas etapas da geração do código.

Foram utilizadas a linguagem C e Assembly juntamente com a ferramenta SASM, que é um software que gera códigos objetos compatíveis com a arquitetura IA-32.

A maior parte do trabalho foi desenvolvido na linguagem C, porém algumas das rotinas básicas foram desenvolvidas em Assembly para melhorar o desempenho. Utilizando também o SASM, para testar a compatibilidade do compilador com esta ferramenta.

A figura 2 ilustra uma árvore de símbolos como saída de um código escrito para este compilador.

Figura 2.



Árvore de símbolos.
Fonte: SCC - Um Compilador C como Ferramenta de Ensino de Compiladores

Trabalho 3: COMPILER BASIC DESIGN AND CONSTRUCTION
Autores: Mahak Jain, Nidhi Sehrawat, Neha Munsi
Ano: 2014

Jain, M. (2014) apresenta um trabalho com sistema de compilação adaptativa, com o objetivo de fornecer uma documentação sobre o projeto e desenvolvimento do compilador, para auxiliar na compreensão do tema e criar técnicas eficazes para desenvolver.

Neste trabalho foi utilizada a linguagem Scheme para a implementação do compilador, e código de montagem (Assembly) como linguagem alvo. As técnicas implementadas foram análise léxica e análise sintática. A análise léxica tem como objetivo verificar a parte gramática de acordo com as regras da linguagem criada ou utilizada, validando o que está

correto ou não. A análise sintática, é responsável por verificar a ordem dos símbolos e sentido.

Trabalho 4: Interpretador/Compilador Python
Autores: Eduardo Bastos, Juliano Freitas
Ano: 2010

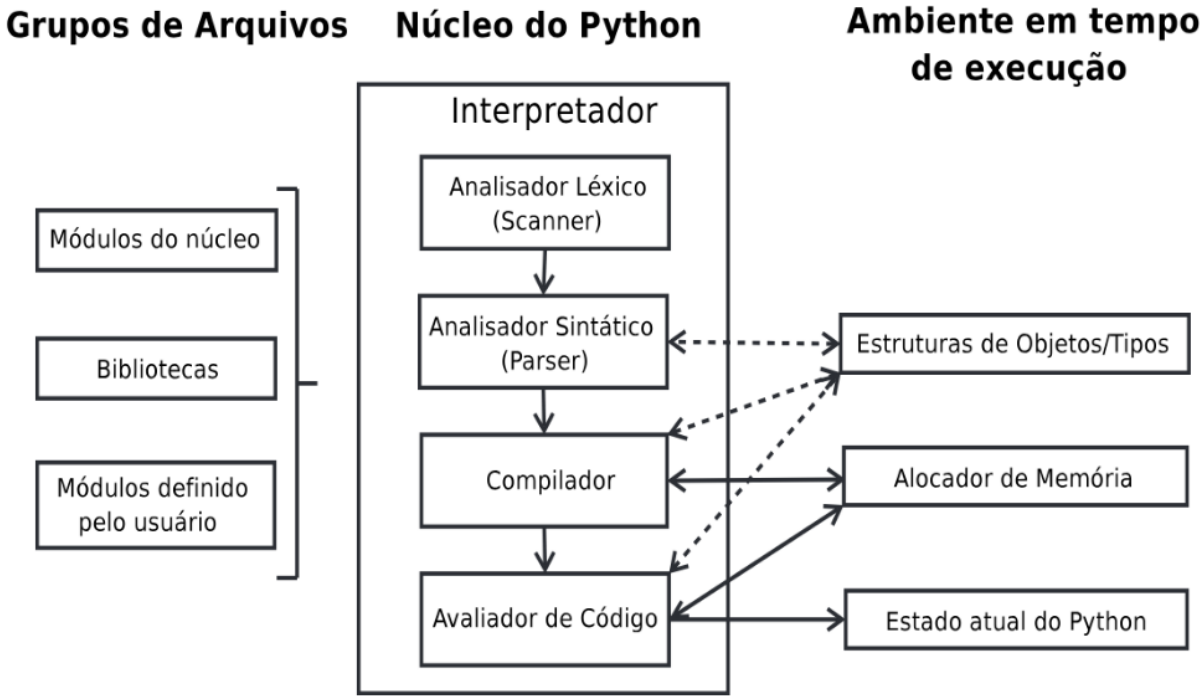
Bastos, E. (2010) apresenta um trabalho sobre o funcionamento da arquitetura do Python, analisando os processos de análise léxica, sintática e a geração de código. Este artigo realiza uma abordagem sobre a estrutura da arquitetura, do interpretador, a gramática da linguagem e as características.

Neste artigo foi utilizada a linguagem Python, uma linguagem de programação interpretada interativa e orientada a objetos, junto com o seu interpretador.

Após os estudos foi verificado que o interpretador obteve uma implementação diferenciada dos demais nos processos de análise léxica e análise sintática. Foi observado também que o Python utiliza máquina virtual para executar os códigos intermediários(bytecodes).

A figura 3 ilustra a arquitetura Python, tendo no centro ó interpretador, e nele os passos seguidos para a compilação do código.

Figura 3.



Interpretação da Arquitetura Python.
Fonte: Interpretador/Compilador Python

Trabalho 5: Compiler Construction
Autores: Aastha Singh, Sonam Sinha, Archana Priyadarshi
Ano: 2013

Singh(2013), apresenta um artigo informando técnicas e exemplos, que facilite a implementação de um compilador.

Neste artigo é apresentado exemplos das técnicas de analise léxica e analise semântica, também mostrando trechos de códigos para melhor compreensão.

Foi utilizado a linguagem Scheme para a construção do compilador, e a linguagem de montagem, código de máquina (Uma linguagem composta apenas de números na base binaria), como linguagem alvo. Schema é uma linguagem que suporta programação funcional e procedural, facilitando assim a elaboração do compilador. O compilador criado utilizou de um gerenciamento de armazenamento do tipo pilha, essa estrutura tem a característica o maneja mento como, o último elemento a entrar nela, é o primeiro a sair.

Quadro 1 – Comparação dos trabalhos relacionados.

Trabalho	**Métodos**	**Gera uma saída**	**Ferramentas utilizadas**	**Linguagens utilizadas**
Trabalho 1	Componentes de fluxogramas; Administração de Imagens.	Código na linguagem C.	SQL Server; Fluxogramas.	C#; SQL/
Trabalho 2	Análise sintática recursiva descendente	Árvore sintática; Árvore de Símbolos.	SASM.	C; Assembly/
Trabalho 3	Análise Léxica; Análise Sintática.	Código de Montagem (Assembly).	-	Scheme; Assembly.
Trabalho 4	Análise Léxica; Análise Sintática.	-	Interpretador Python.	Python.
Trabalho 5	Análise Léxica; Análise Sintática.	Código de Máquina.	-	Scheme.

6.5. Produtos a serem Gerados:

- Interpretador da Linguagem D+;
- Interface para desenvolvimento utilizando o interpretador;
- Extensão da linguagem ao interpretador.
- Relatório de testes.

6.6. Cronograma:

Fases do estudo (TCC)	Jul.	Ago.	Set.	Out.	Nov.	Dez.
Definição do Tema						
Elaboração da Proposta de Tema						
Pesquisa e Leitura de Referência						
Escrita de Fundamentação Teórica						

Elaboração da Metodologia de Desenvolvimento						
Implementação da Analise Léxica						
Implementação da Analise Sintática						
Implementação da Analise Semântica						
Desenvolvimento da Interface						
Efetuar testes						
Entrega da Primeira parte do TCC						
Entrega final do TCC						
Escrita do TCC						
Banca						

Curitiba, 26 de Agosto de 2019.