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
Projeto de Graduação

Proposta de Tema

(Apêndice CC1)

1. Título

VL-D+: uma ferramenta de estudo para compiladores através de uma interface.

2. Tema

Este projeto de pesquisa delimita-se em propor e desenvolver uma ferramenta de estudo, em forma de aplicação *web*, que possa influenciar na compreensão de partes do processo de compilação, para os estudantes de compiladores.

3. Equipe

Aluno:	Sabrina Eloise Nawcki	Rubrica:	
E-mail:	snawcki@gmail.com	Telefone:	(41) 9 9511-6077

4. Orientador e Coorientador

Nome:	Professor Diógenes Furlan	Rubrica:	
E-mail:	mestredidi@yahoo.com.br	Telefone:	(41) 9 9626-2121
Nome:		Rubrica:	
E-mail:		Telefone:	

5. Reunião de Orientação

Dia da semana:	quinta-feira	Hora:	21h00
Local:	Comunicação remota via Telegram, Microsoft Teams e e-mail.		

6. Resumo

6.1. Introdução e Justificativa: Os compiladores são o meio intermediário entre a comunicação do ser humano com a máquina. De forma simplificada, o compilador é a ferramenta que lê um programa, escrito em uma linguagem fonte, e o traduz em um programa equivalente em outra linguagem, mantendo a semântica original. (Aho, 1995).

De acordo com Aho (1995), o compilador opera em fases, onde cada uma transforma o código fonte de uma representação para outra. A divisão das fases é dada como: análise léxica, análise sintática, análise semântica, geração de código intermediário, otimização de código e geração de código.

Na disciplina de compiladores, são estudadas técnicas e métodos para construir um compilador e por isso se apresenta como uma das disciplinas com maior dificuldade de

compreensão pelos discentes dos cursos de computação, tendo grande complexidade, difícil visualização e dependência de outras disciplinas.

Diante dessa premissa, devem-se criar meios que facilitem o aprendizado sem o auxílio direto de um docente, para que assim se obtenha um nível de absorção do conteúdo de forma ampla e prática.

A internet dispõe de formas comunicacionais, abrindo portas para o compartilhamento de informações, seja na utilização de imagens, vídeos, textos e outros. Assim no meio didático pode-se criar ferramentas que auxiliem a construção do conhecimento.

Sendo assim, esse trabalho de pesquisa visa desenvolver e disponibilizar na *web* uma ferramenta que possa cessar possíveis dificuldades, que os discentes têm na disciplina de compiladores.

6.2. Problema Proposto:

- Quais ferramentas de estudo que melhores se adequam a compreensão do processo de compilação para discentes?
- Como as ferramentas de estudo podem beneficiar o discente?
- Quais os métodos de aprendizagem que facilitam na compreensão de um conteúdo?
- Como esses métodos podem ser postos para o processo de compilação?
- É possível desenvolver um simulador de compilador em tempo real?

6.3. Objetivos: Desenvolver uma aplicação *web* como uma ferramenta de estudo para auxiliar no aprendizado do processo de compilação, tendo como benefício para os discentes a absorção rápida do conteúdo.

Objetivos específicos:

- Desenvolver interface dinâmica para a análise léxica.
- Desenvolver interface para árvore de análise sintática.
- Desenvolver interface para tabela de símbolos.

6.4. Trabalhos Relacionados

Trabalho 1: Auxílio no ensino em compiladores: software simulador como ferramenta de apoio na área de compiladores

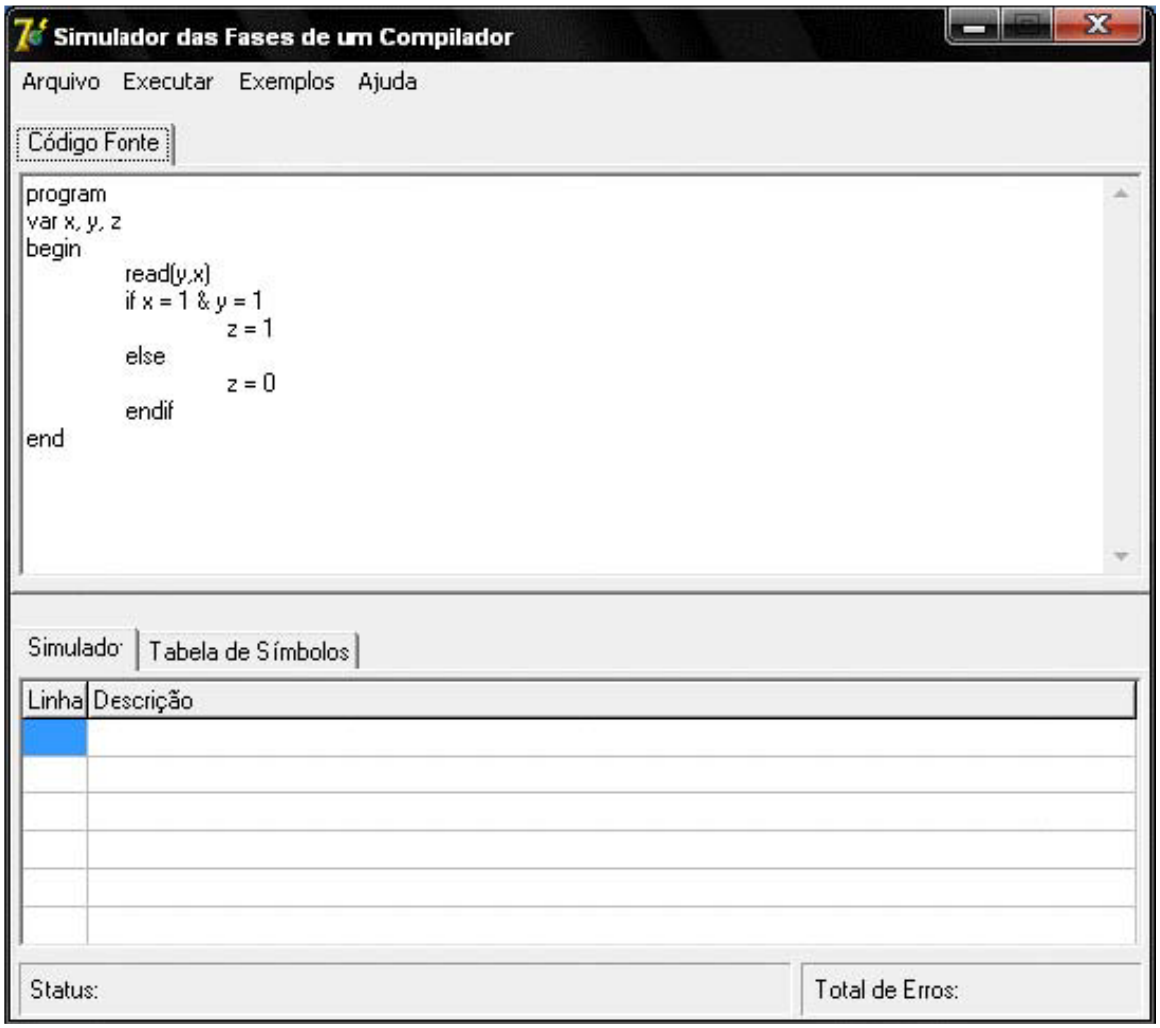
Autor: Kelton A. P. da Costa, Luis Alexandre da Silva, Talita Pagani Brito

Os pesquisadores Costa, Silva e Britto declararam em seu projeto a necessidade de elaborar uma técnica que pudesse facilitar a compreensão dos discentes, mais especificamente da disciplina de Compiladores, dos cursos que envolvem computação. Com essa premissa, eles projetaram e desenvolveram um software capaz de simular claramente o funcionamento interno das fases de um compilador.

A ferramenta *CompilerSim* foi criada para executar um processo de análise de código baseado na linguagem de programação Pascal, e, posteriormente, para converter o código para a linguagem de baixo nível Assembly x86. O desenvolvimento da ferramenta se baseou na série de artigos "*Let's Build a Compiler*", escritos pelo cientista Ph.D. Jack W. Crenshaw na década de 80. A partir dos artigos foi extraído o código original e o mesmo passou por modificações e adaptações com o intuito de que as etapas do processo de compilação fossem demonstradas através de uma interface gráfica.

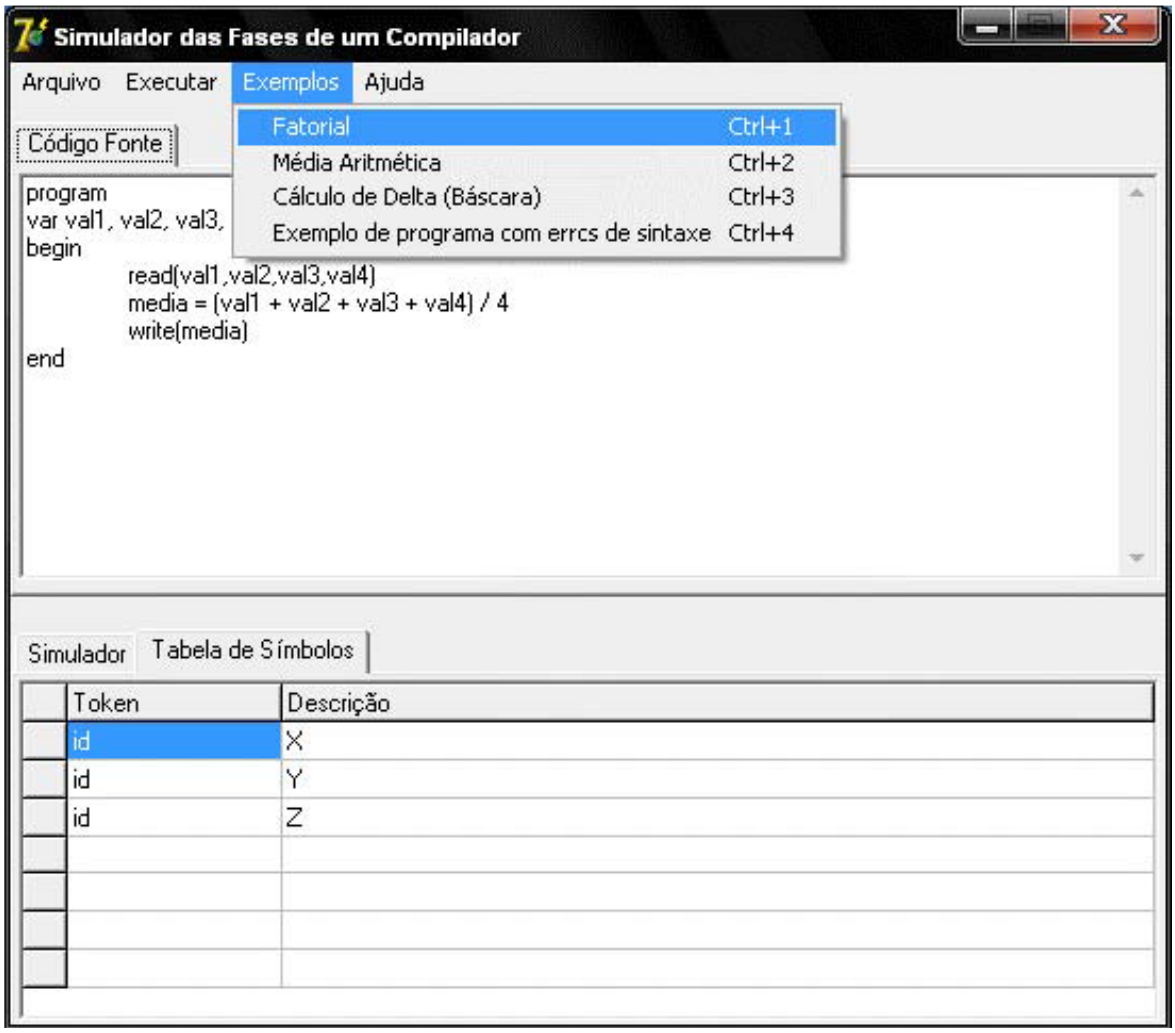
A interface gráfica do software apresenta algumas funcionalidades para o usuário. A primeira etapa da aplicação consiste em inserir o código fonte no programa, podendo ser feito manualmente pelo usuário (figura 1) ou carregado a partir da aba "Exemplos" (figura 2).

FIGURA 1 - Interface padrão do simulador *CompilerSim*



FONTE: COSTA, SILVA, BRITTO (2008)

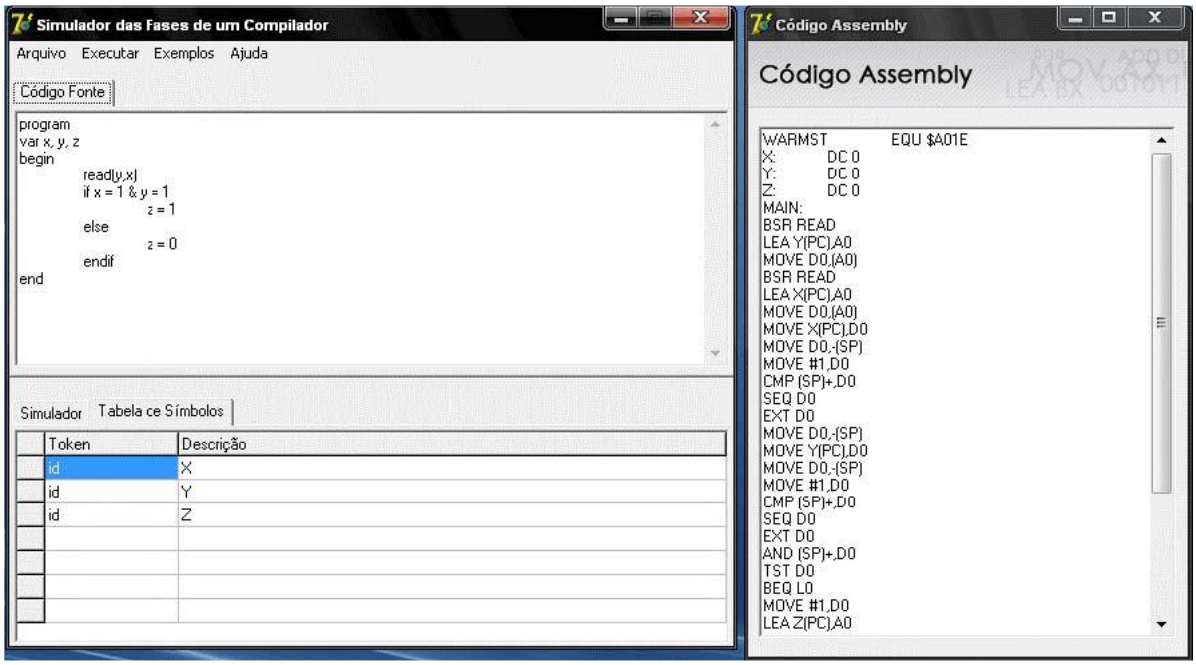
FIGURA 2 - Exemplos prontos para execução



FONTE: COSTA, SILVA, BRITTO (2008)

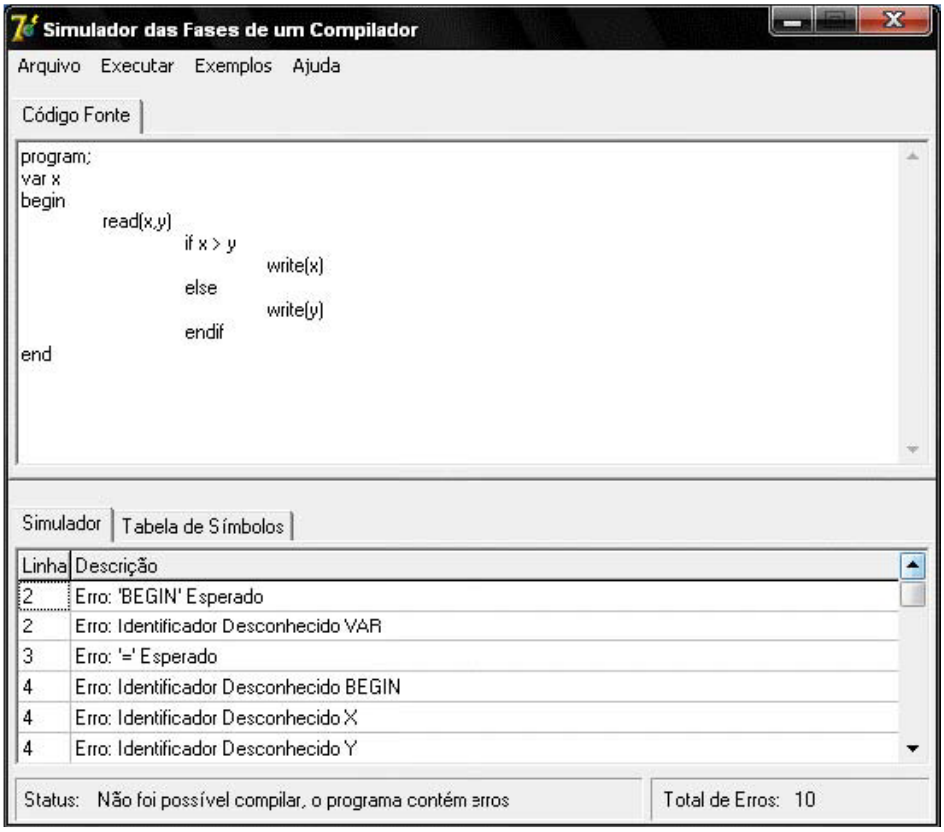
Após a execução do código, a ferramenta apresenta os seguintes resultados: os *tokens* gerados a partir da análise léxica do código, a interpretação e geração de código intermediário na linguagem Assembly, e, quando cabível, os erros gerados na compilação. A visualização desses itens pode ser observada nas figuras 3 e 4 respectivamente.

FIGURA 3 - Análise da linguagem e resultados abordados



FONTE: COSTA, SILVA, BRITTO (2008)

FIGURA 4 - Execução de um código e a indicação de erros



FONTE: COSTA, SILVA, BRITTO (2008)

Os autores do projeto *CompilerSim* concluíram através da utilização do software simulador em salas de aula que a ferramenta pode influenciar positivamente e significativamente o processo de ensino e aprendizagem na disciplina de compiladores.

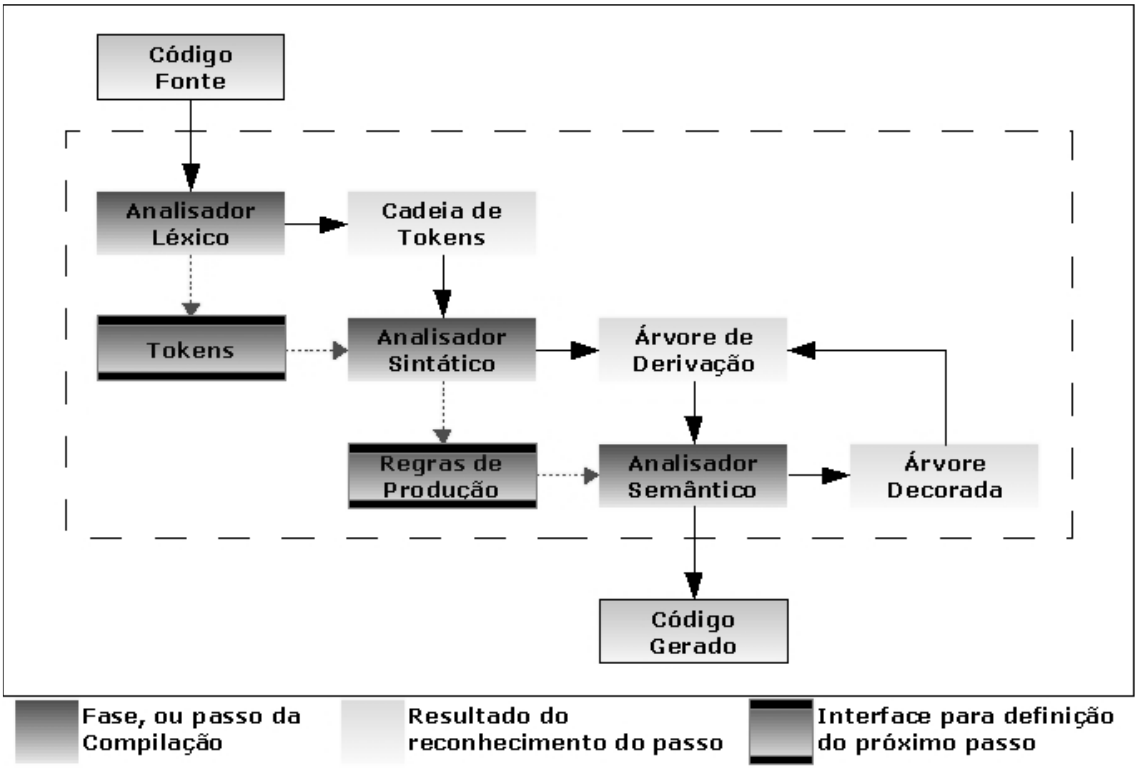
Trabalho 2: C-gen – Ambiente Educacional para Geração de Compiladores
Autor: Jerônimo Backes, Alessandra Dahmer

O *C-gen* é uma ferramenta, criada pelos pesquisadores Backes e Dahmer, com o intuito de sanar a carência de ferramentas com interface gráfica possíveis de serem utilizadas para orientar os discentes da disciplina de Compiladores, exibindo o funcionamento de todo o processo de compilação.

A ferramenta desenvolvida permite que o discente a utilize enquanto aprende, possibilitando que a teoria de compiladores seja visualizada, assim facilitando a compreensão do conteúdo. Para isso, a ferramenta atende alguns requisitos, dentre eles: possibilitar a definição e disponibilizar uma interface adequada para os analisadores léxicos, sintáticos e semânticos; permitir o acompanhamento do processo de reconhecimento das fases da compilação, passo a passo.

A arquitetura do sistema é organizada para que cada passo do processo de compilação se comunique com o passo posterior através de uma representação intermediária, permitindo ser executado independentemente, conforme apresentado na figura 5.

FIGURA 5 - Arquitetura do protótipo



FONTE: BACKES, DAHMER (2006)

As fases da compilação são implementadas com *plug-ins* que reconhecem as suas respectivas entradas e produzem as saídas correspondentes.

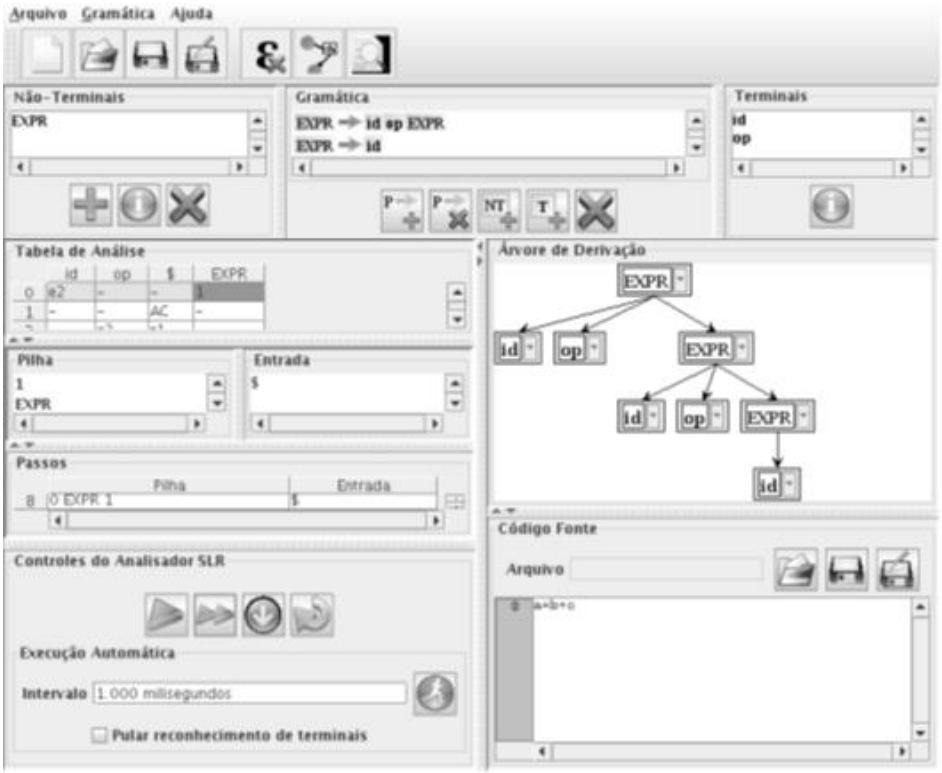
O *plug-in* responsável pelo analisador léxico foi implementado de forma que seja produzido um analisador através da definição de autômatos. A definição é informada pelo usuário, que é responsável pela criação de estados e transições do autômato e os caracteres que o mesmo deve reconhecer em cada transição. Após a edição do autômato, quando iniciado o processo de reconhecimento, a ferramenta converte os dados em um autômato finito determinístico e cria a tabela de transições, conforme ilustra a figura 6.

FIGURA 6 - Interface de edição de um autômato e reconhecimento

FONTE: BACKES, DAHMER (2006)

A análise sintática é feita através da especificação da gramática, considerando como terminais os *tokens* gerados pelo analisador léxico. A gramática é gerada pelo usuário no editor, que deve criar os símbolos não-terminais. O propósito do usuário gerar sua própria gramática é justamente para que ele não precise aprender uma nova notação para utilizar a ferramenta. A interface para essa confecção e de resultado da análise é ilustrada pela figura 7.

FIGURA 1 - Interface de edição de gramáticas e reconhecimento



FONTE: BACKES, DAHMER (2006)

Em suma, além do programa auxiliar os discentes no processo de aprendizagem da disciplina de Compiladores, a estrutura do *C-gen* permite expansões de funcionalidades via *plug-ins* ou através da contribuição voluntária de desenvolvedores, visto que o programa é um software livre.

Trabalho 3: Compilador Educativo VERTO: ambiente para aprendizagem de compiladores
Autor: Carlos Sérgio Schneider, Liliana Maria Passerino, Ricardo Ferreira de Oliveira

No Centro Universitário Feevale foi notada a necessidade de desenvolver uma ferramenta de apoio pedagógico, para a disciplina de Compiladores após a mesma enfrentar desmotivação e dificuldades de compreensão dos discentes.

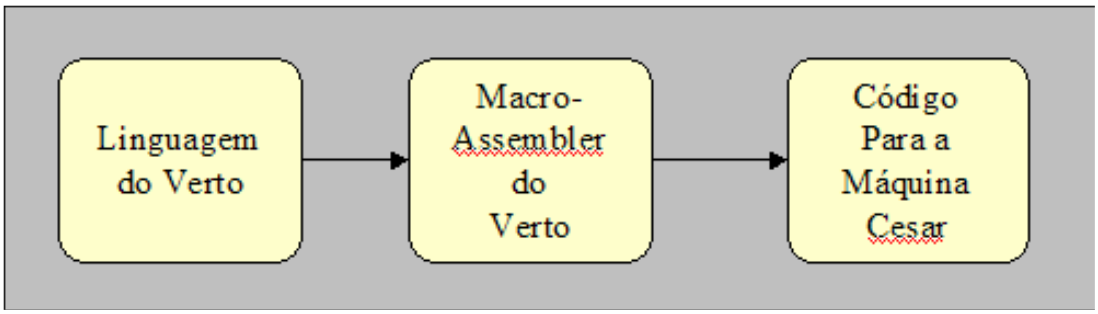
Os compiladores são, de forma geral, tradutores de alguma linguagem para um programa. De acordo com os pesquisadores Schneider, Passerino e Oliveira, a aprendizagem de compiladores consiste em absorver um processo composto por etapas que exigem estratégias e métodos específicos para aprender o processo de compilação, sendo ele: análise léxica, sintática e semântica, tabela de símbolos e geração do código objeto.

A pesquisa ressalta também que, a aprendizagem é a construção de uma representação pessoal de um conteúdo que é objeto de aprendizagem (Coll, 1998 *apud* SCHNEIDER, 2005). Para atingir esse nível de compreensão os estudantes devem ser capacitados a compreender as fases do compilador, sobretudo as fases finais de geração de código intermediário e objeto.

Visto a dificuldade dos discentes e a disciplina ser lecionada em apenas um semestre, foi desenvolvido o Compilador Educativo *Verto*, que faz parte de um ambiente de aprendizagem para compiladores. O compilador é composto por diversas ferramentas computacionais, um docente mediador, os discentes que o utilizam, as sequências didáticas que são planejadas no plano de ensino e aprendizagem apoiada pela metodologia de ensino embasada pelo docente.

O processo de compilação do *Verto* inicia-se com a geração do código intermediário em um formato *macro-assembler*. O formato dispõe de instruções simplificadas para facilitar a compreensão das estruturas compiladas. Após essa etapa, gera-se o arquivo final que contém as instruções no formato *César*, uma linguagem objeto criada com fins didáticos, permitindo que o estudante execute e analise o algoritmo. Na figura 8 pode-se observar o esquema de tradução do compilador *Verto*.

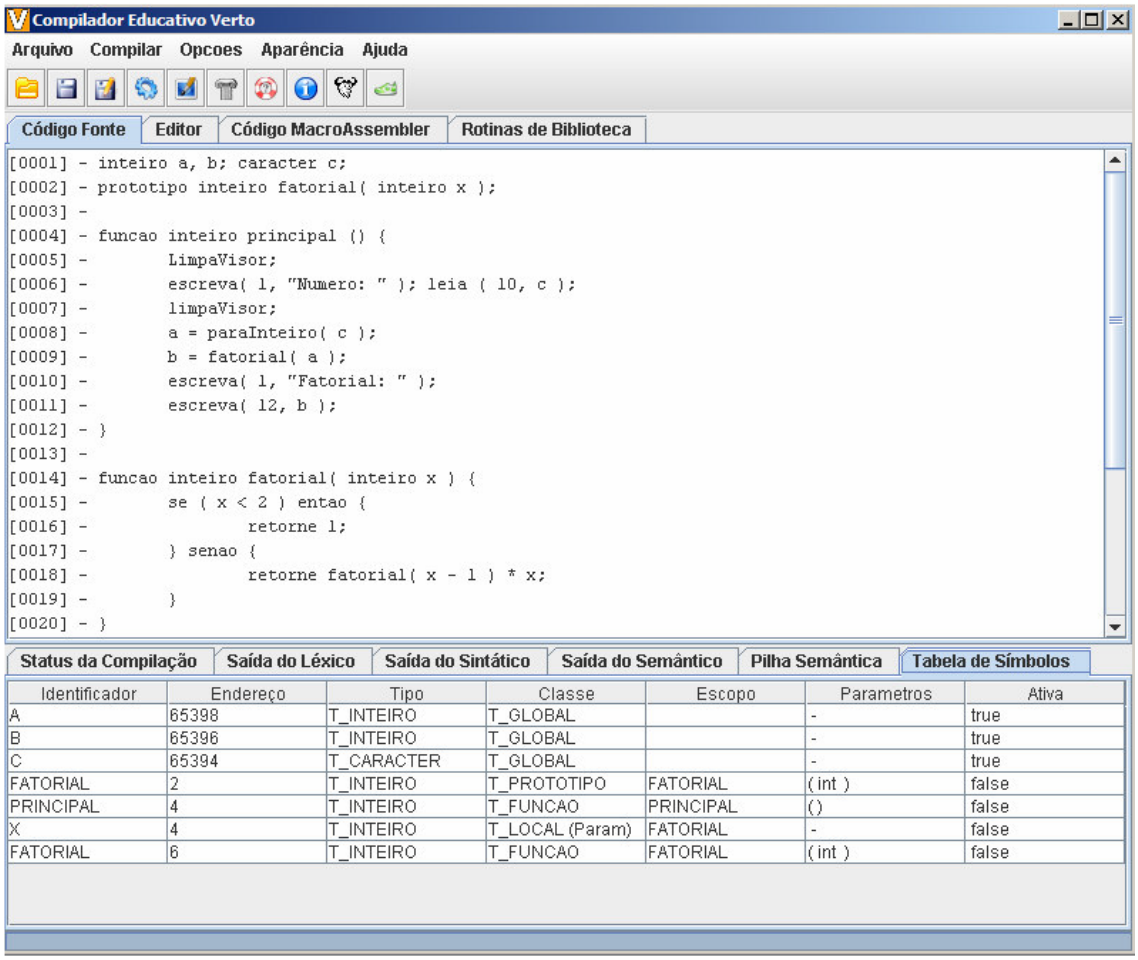
FIGURA 8 - Esquema de Tradução do Compilador *Verto*



FONTE: SCHNEIDER, PASSERINO, OLIVEIRA (2007)

Para o uso inicial da ferramenta, dispõe-se de uma tela para a edição de textos fonte escritos na linguagem *César*. Na figura 9 é possível observar a tela de edição e a tabela de símbolos gerada a partir do código inserido.

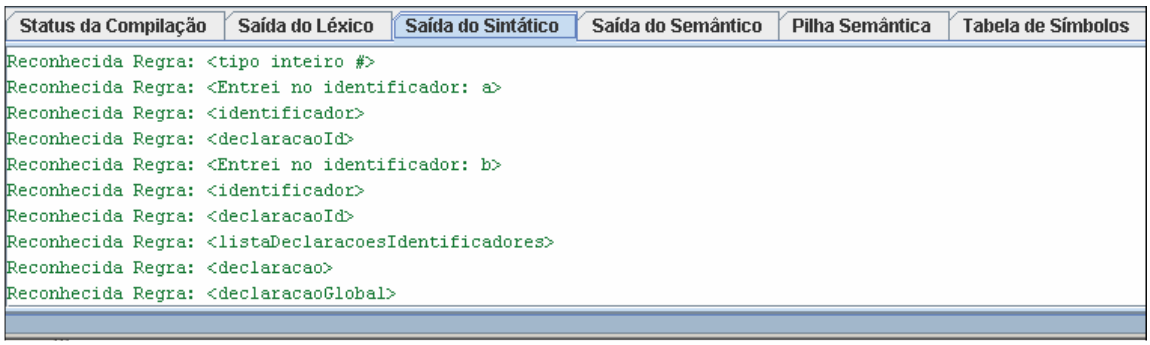
FIGURA 9 - Interface de Edição do Compilador *Verto*



FONTE: SCHNEIDER, PASSERINO, OLIVEIRA (2007)

A ferramenta focou as etapas finais da compilação, utilizando uma técnica de análise léxica simples e um método de análise sintática. A análise sintática, onde é feita a geração de erros, análise semântica e geração do código-objeto, é feita de forma recursiva, onde cada regra sintática reconhecida pode levar o compilador a disparar uma rotina de ação semântica ou de geração de código. A ampliação da saída do analisador sintático é ilustrada pela figura 10, onde é registrada a sequência de regras reconhecidas pelo compilador.

FIGURA 10 - Aba: Saída do Sintático



FONTE: SCHNEIDER, PASSERINO, OLIVEIRA (2007)

Os autores da ferramenta *Verto* a descrevem como uma ferramenta de auxílio de múltiplas disciplinas, sendo elas: compiladores, arquitetura de computadores e paradigmas de linguagens de programação.

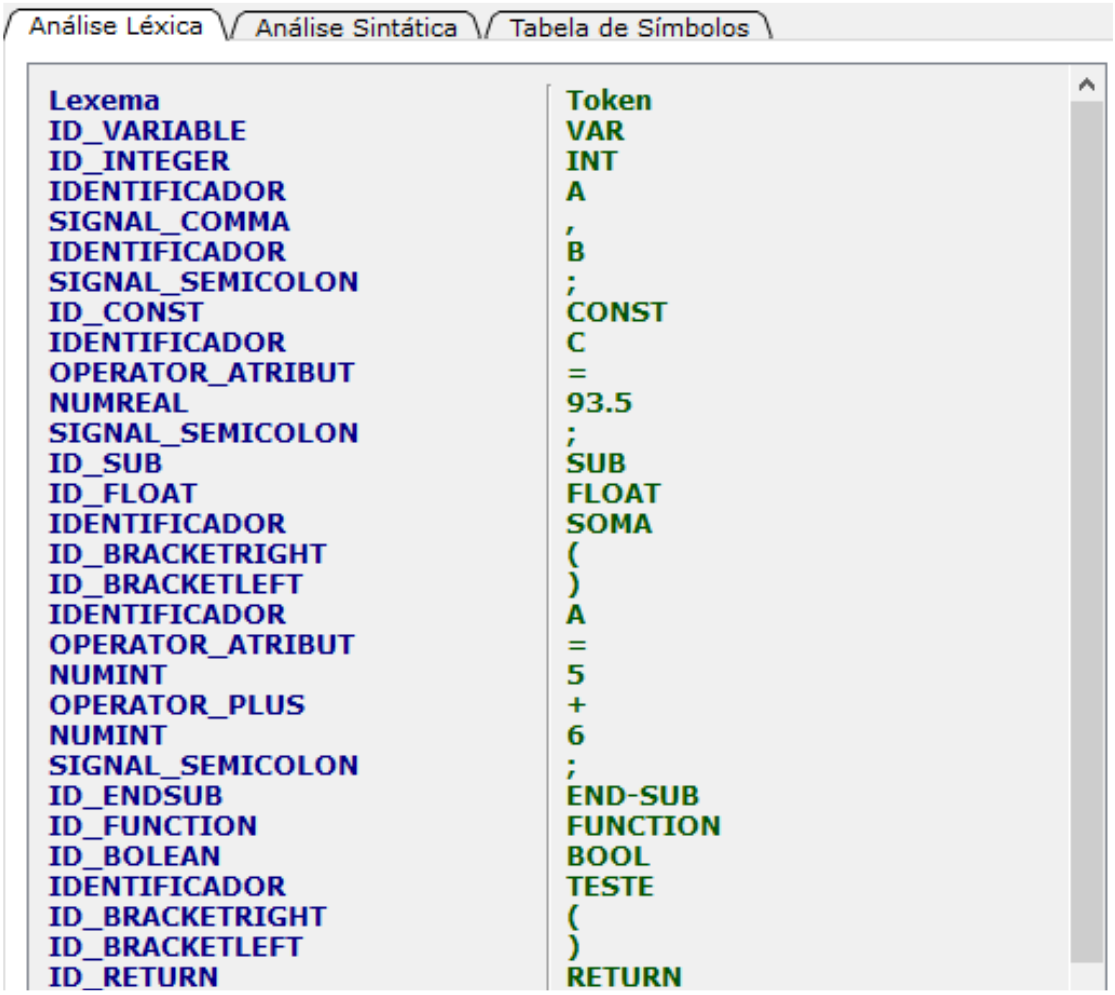
Trabalho 4: Interpretador da Linguagem D+
Autor: Gabriel Pinto Ribeiro da Fonseca

O interpretador da linguagem D+ é um software proposto e desenvolvido por Gabriel Ribeiro (2019). Ribeiro enfatiza em seu projeto a dificuldade de aprendizagem dos discentes da disciplina de Compiladores no Bacharelado em Ciência da Computação, ele descreve a disciplina como muito abrangente e profunda. Em sua pesquisa, o autor indica que mais da metade dos estudantes de compiladores tem grande dificuldade de aprender a disciplina.

Visto o problema da aprendizagem dos estudantes, Ribeiro (2019) desenvolveu um software integrado a uma interface com o intuito de amenizar a dificuldade dos discentes no aprendizado e auxiliá-los na absorção do conteúdo de compiladores. O software consiste em uma interface de fácil utilização, onde o usuário insere o código em linguagem D+, pressiona o botão para executar a compilação, e o resultado já é gerado pelo programa contemplando: análise léxica, análise sintática e tabela de símbolos.

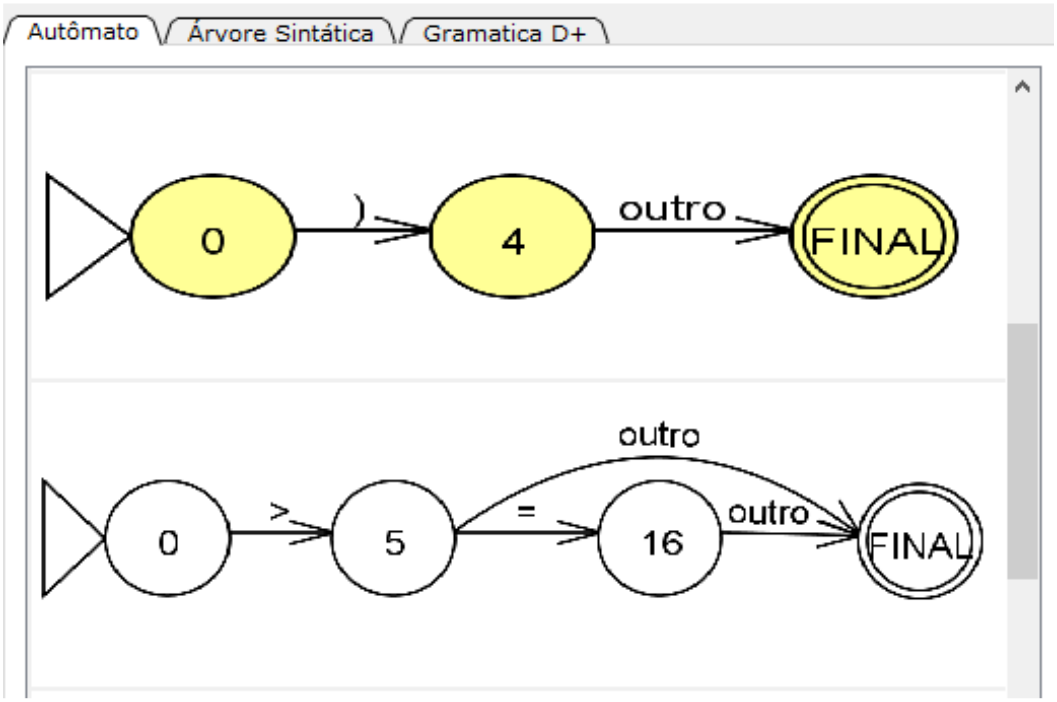
A análise léxica é demonstrada tanto por uma tabela de *tokens* e lexemas, quanto por um autômato finito que contém todos os autômatos possíveis para a linguagem D+. Após a compilação do código inserido pelo usuário, os autômatos utilizados são preenchidos com fundo amarelo. A figura 11 ilustra a tabela de saída da análise léxica e a figura 12 ilustra um autômato colorido quando é acessado, e outro em preto e branco quando não é acessado.

FIGURA 11 - Saída da análise sintática



FONTE: RIBEIRO (2019)

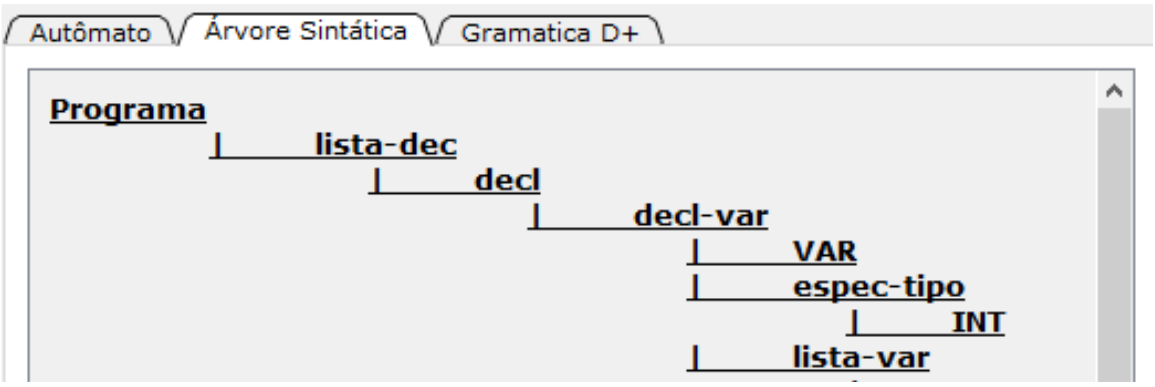
FIGURA 12 - Interface com os autômatos da análise sintática



FONTE: RIBEIRO (2019)

A análise sintática é representada por uma árvore sintática conforme ilustra figura 13.

FIGURA 13 - Árvore Sintática montada



FONTE: RIBEIRO (2019)

A tabela de símbolos ilustra os identificadores localizados pelo interpretador e traz informações que auxiliam na compreensão de cada palavra inserida no código, conforme apresentado na figura 14.

FIGURA 14 - Tabela de símbolos

#	Nome	Tipo	Categoria	Linha
0	A	INT	VARIAVEL	1
1	C	CONST	CONSTANTE	2
2	SOMA	FLOAT	PROCEDURE	4
3	TESTE	BOOL	FUNÇÃO	8
4	D	BOOL	VARIAVEL	13

FONTE: RIBEIRO (2019)

As tecnologias utilizadas para o desenvolvimento do interpretador de D+ foram a linguagem de programação C++ para a codificação do compilador, QT *Creator* para criação da interface e JFLAP para a criação dos autômatos finitos.

Para testes de eficiência, o interpretador de D+ foi distribuído junto a um questionário sobre o uso da ferramenta, para discentes da disciplina de Compiladores, que gerou resultados positivos, afirmando que a ferramenta ajuda na compreensão das etapas do compilador. O projeto também constou que para trabalhos futuros poderiam ser implementadas na interface a: análise semântica, geração do código de máquina, dinamicidade na apresentação da informação e o *log* da análise semântica.

Trabalho 5: SCC: Um Compilador C como Ferramenta de Ensino de Compiladores
Autor: Juliano Henrique Foleiss, Guilherme Puglia Assunção, Eduardo Henrique Molina da Cruz, Ronaldo Augusto de Lara Gonçalves, Valéria Delisandra Feltrim

O compilador SOIS C *Compiler* (SCC), de Foleiss, Assunção, Cruz, Gonçalvez e Feltrim (2009), se destaca entre os compiladores de C por contemplar todo o conjunto de instruções de um compilador, assim proporcionando um serviço de ajuda no ensino de disciplinas que abordam o processo de compilação.

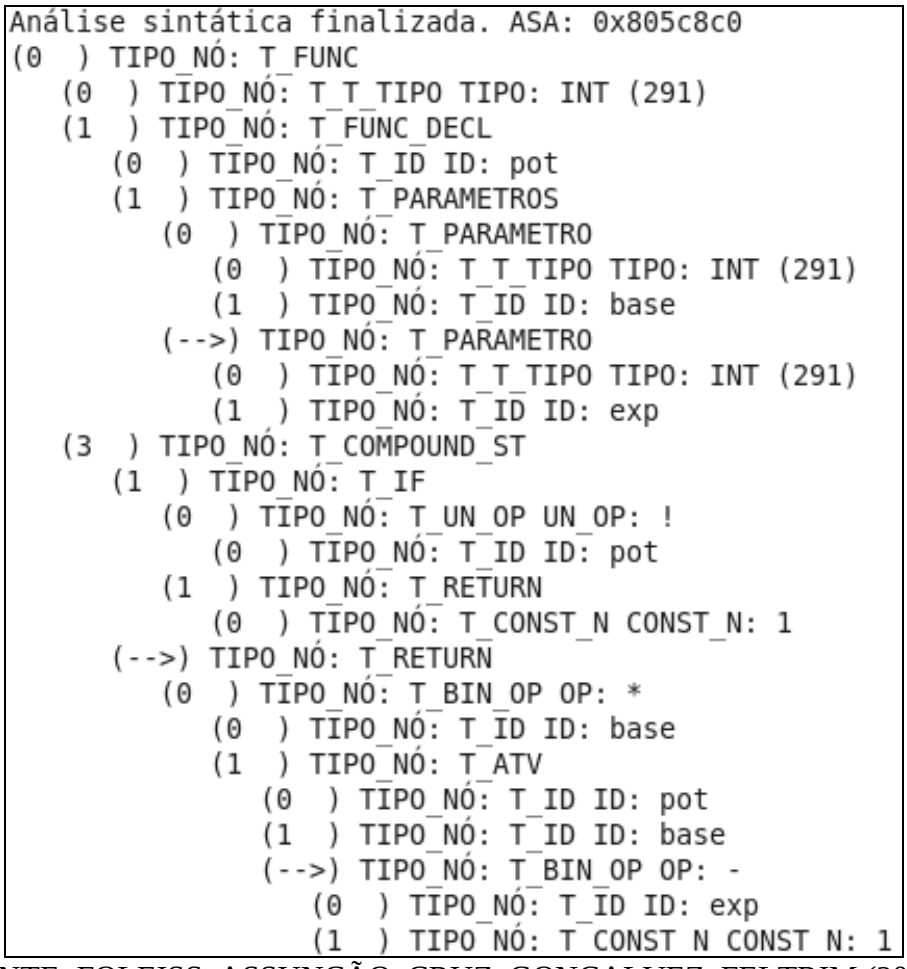
O Sistema Operacional Integrado Simulado (SOIS), foi escolhido justamente por ser um ambiente que permite a escrita, execução e depuração de programas em um ambiente simulado. Por sua vez, o montador SASM possui modos de depuração que podem demonstrar todas as fases do processo de compilação, a inter-relação dos seus artefatos e componentes e os resultados obtidos.

O projeto foi desenvolvido com os objetivos de gerar código que permita que o simulador do ambiente execute programas da linguagem C e de ser uma ferramenta de ensino, tendo funcionalidades que auxiliem a compreensão de áreas específicas do processo de um compilador real.

A pesquisa escolheu o processo de compilação proposto por Louden que possui cinco etapas: análise léxica, análise sintática, análise semântica, geração de código intermediário e geração de código final. No SCC, são geradas: a árvore sintática, a árvore de símbolos, a árvore sintática abstrata anotada e um analisador semântico.

A análise sintática tem como principal artefato a árvore sintática abstrata, nela é mostrada a estrutura sintática do programa, sendo possível visualizar o mapeamento do programa-fonte. A figura 15 ilustra um trecho de uma árvore sintática gerada pelo SCC.

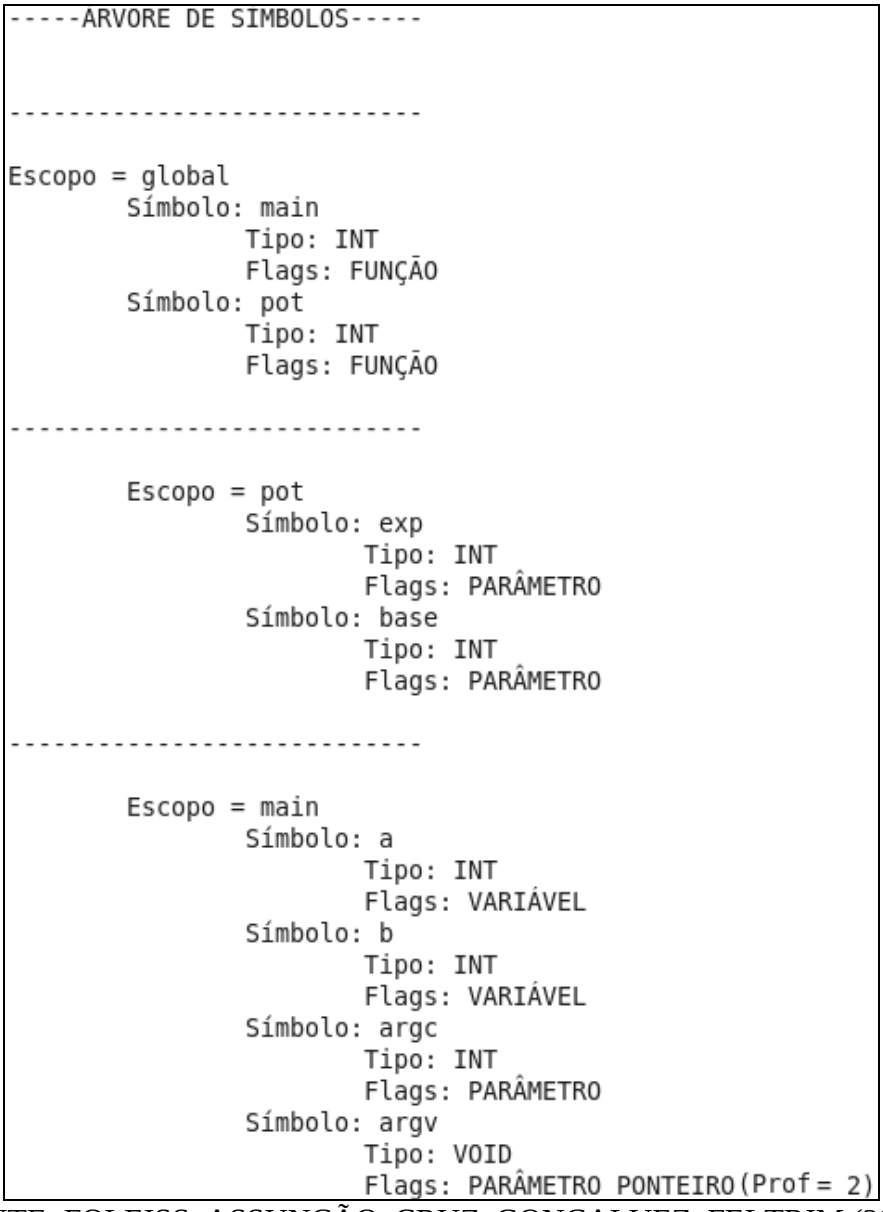
FIGURA 15 - Árvore Sintática



FONTE: FOLEISS, ASSUNÇÃO, CRUZ, GONÇALVEZ, FELTRIM (2009)

A árvore de símbolos é outro artefato que explora o entendimento do compilador, é ela a responsável por transcrever a tabela de símbolos gerada a partir do código fonte. A figura 16 ilustra a árvore de símbolos, que é dividida ente os escopos existentes no código.

FIGURA 16 - Árvore de símbolos



FONTE: FOLEISS, ASSUNÇÃO, CRUZ, GONÇALVEZ, FELTRIM (2009)

A proposta do SCC é apresentar, aos discentes da disciplina de Compiladores, um compilador que mostre os desafios presentes na implementação de um compilador, utilizando métodos empregados no desenvolvimento de compiladores profissionais.

Além dos modos de depuração o SCC provê ao usuário interfaces com algumas partes do processo de compilação. O projeto também citou a possibilidade de incluir um módulo de otimização no sistema e a visualização gráfica da árvore sintática como trabalhos futuros.

Título	Possui saída visual?	Ilustra a Análise Léxica?	Ilustra a Análise Sintática?	Ilustra a Tabela de Símbolos?	Ilustra código intermediário?	Ilustra os erros de compilação?	É *off-line* ou *on-line*?	Ilustração estática ou dinâmica?
Auxílio no ensino em compiladores: software simulador como ferramenta de apoio na área de compiladores	SIM	NÃO	NÃO	SIM	SIM	SIM	OFF	ESTÁTICA
C-gen – ambiente educacional para geração de compiladores	SIM	SIM	SIM	NÃO	NÃO	NÃO	OFF	ESTÁTICA
Compilador educativo verto: ambiente para aprendizagem de compiladores	SIM	SIM	SIM	SIM	SIM	NÃO	OFF	ESTÁTICA
Interpretador da linguagem D+	SIM	SIM	SIM	SIM	NÃO	SIM	OFF	ESTÁTICA
SCC: um compilador C como ferramenta de ensino de compiladores	SIM	NÃO	SIM	SIM	NÃO	NÃO	OFF	ESTÁTICA

6.5. Produtos a Serem Gerados: No desenvolvimento do presente trabalho deverão ser gerados:

- Protótipo do software
- Monografia
- Relatório de testes do software
- Resultados de satisfação com os usuários que utilizaram o software.

6.6. Cronograma

Fases do Estudo – TCC 1	Março	Abril	Maio	Junho
Proposta de Tema				
Trabalhos Relacionados				
Revisão Bibliográfica				
Metodologia				
Apresentação para Banca				

Fases do Estudo – TCC 2	Setembro	Outubro	Novembro	Dezembro
Proposta de Tema Atualizada				
Entrega do TCC 1 Corrigido				
Protótipo – Análise Léxica				
Protótipo – Análise Sintática				
Protótipo – Tabela de Símbolos				
Protótipo – Registro de Erros				
Protótipo – Testes				
Entrega do TCC 2				
Apresentação para Banca				
Entrega do *Pen Drive*				

Curitiba, 09 de setembro de 2020.