INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
DO MATO GROSSO

NÍCOLAS TIMOTEU CUERBAS

RELATÓRIO FINAL DE ATIVIDADE ACADÊMICA – IMPLEMENTAÇÃO DE COMPILADOR

Cuiabá – MT

2018

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO MATO GROSSO
Campus Cuiabá
Departamento de Área de Informática

NÍCOLAS TIMOTEU CUERBAS

RELATÓRIO FINAL DE ATIVIDADE ACADÊMICA – IMPLEMENTAÇÃO DE COMPILADOR

Relatório Final de Atividade Acadêmica, referente a implementação de um compilador, como parte dos requisitos necessários para a conclusão da disciplina Compiladores do curso de Engenharia de Computação do Campus Cuiabá do Instituto Federal do Mato Grosso.

Professor da Disciplina: Dr. Ed' Wilson Tavares Ferreira

Cuiabá – MT
2018

RESUMO

Neste documento é apresentado o relato da experiência de implementação de um compilador, desenvolvido na disciplina de Compiladores. Foi proposto uma gramática baseada em LL(1), que possui recursos básicos de uma linguagem de programação. Todas as fases de um compilador foram desenvolvidas, porém optou-se em gerar o código final para fasm, em função de ser uma liguagem fácil para iniciantes em assembly e possuir uma vasta documentação feita pelos desenvoldores em seu site. Optou-se por desenvolver a analise sintatica, semântico, lexica e a geração de código em python3, em função da linguagem possuir muitas ferramentas para tratamento de texto e identação obrigatoria que facilita a leitura do código.

Palavras-chave: compilador, LL1, python, fasm, flat assembler.

ABSTRACT

In this document the experience of implementing a compiler, developed in Compiler discipline, is presented. We have proposed a grammar based on LL(1), which has basic features of a programming language. All phases of a compiler were developed, but it was decided to generate the final code for fasm, because it is an easy language for beginners in assembly and has extensive documentation made by the developers on its website. It was decided to develop the syntactic, semantic, lexical analysis and the generation of code in python3, because the language has many tools for word processing and obligatory indentation that facilitates the reading of the code.

Keywords: compiler, LL1, python, fasm, flat assembler

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

fasm-flat assembler

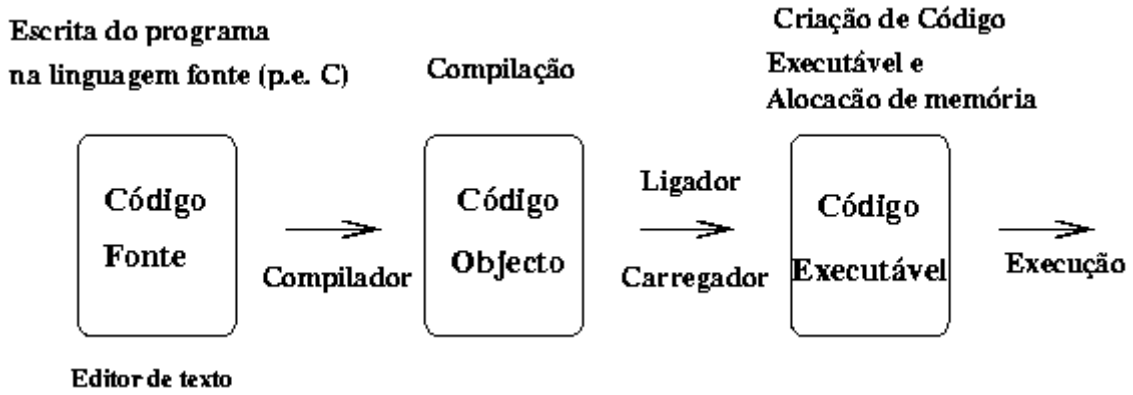
SUMÁRIO

INTRODUÇÃO

Compiladores são vistos como artefatos magicos, distantes para meros mortais. E os livros de compiladores falam em sua maioria em linguagem que apenas magos conseguem entendes(Ghuloum,2006, tradução nossa).

Com o crescimento da internet, computadores e os software tem fornecido vastas ferramentas para o usuário que fornecem comunicação, noticias, entretenimento, e segurança. E os computadores embarcados estão mudando os nossos automoveis, aviões, telefones, televisores, e rádios. A Computação tem criado novas categorias de atividaes, de videogames até redes sociais. Supercomputadores fazem previsões do tempo diariamente e o curso de fortes tempestades. Computadores embarcados sincronizam o trafego de luz e enviam e-mais para o nosso bolso. Todas essa aplicações dependem de programas de computador que constroem ferramentas virtuais sobre as abstrações de baixo nível fornecidas pelo hardware. Quase todos destes programas são traduções de um outro programa de computador chamado de compilador. Compiladores são programas de computadores que traduzem um programa escrito em uma linguagem inicial e escrevem para uma outra linguagem. (Cooper, 2011, tradução nossa).

Imagem 1 – Tradução de um código fonte para um código executavel

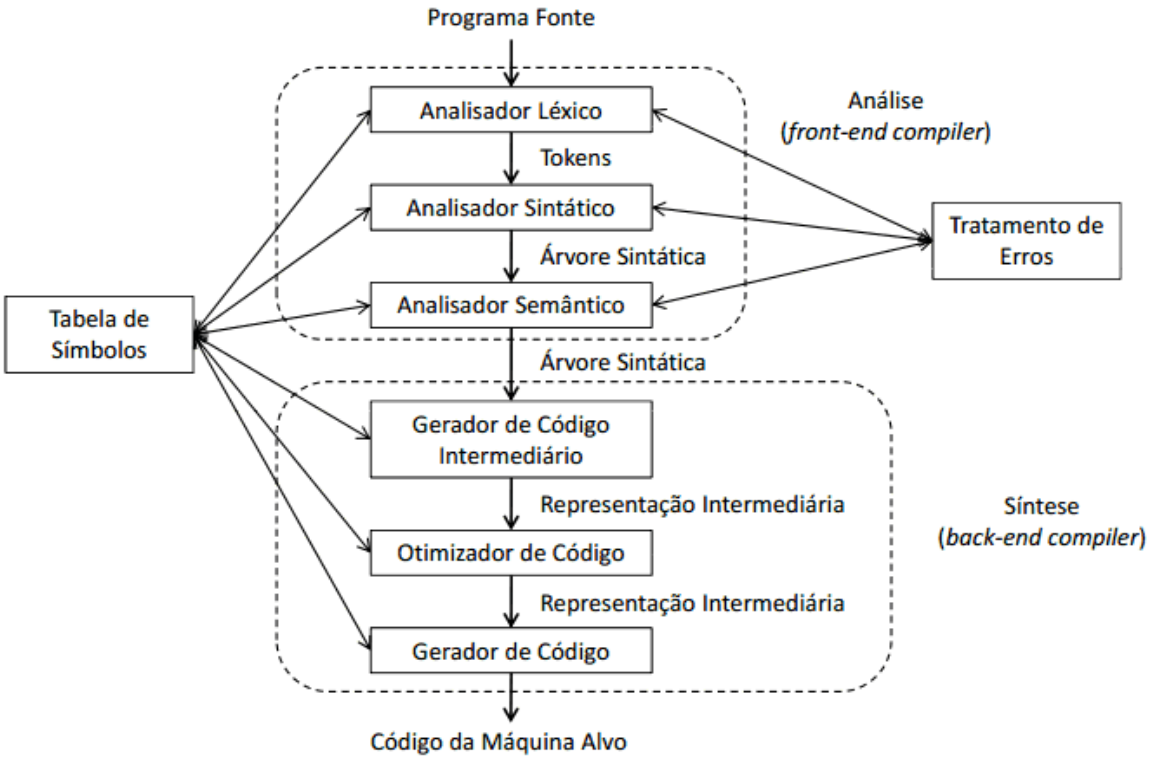


Fonte: (Camacam)

"O processo de compilação é muito complexo, existindo uma estrutura básica que divide esse processo em fases, essas fases estão representadas por duas tarefas conhecidas como **análise** e **síntese"**(Camacam).

> O processo de compilação inicia com o **analisador léxico** que varre todo o programa fonte e transforma o texto em um fluxo de *tokens*, e nessa fase é criada a **tabela de símbolos**. Logo em seguida vem a **análise sintática** que lê o fluxo de *tokens* e valida a estrutura do programa criando a **árvore sintática**. A terceira fase é a **análise semântica** que é responsável por garantir as regras semânticas. Todas essas fases fazem parte da tarefa de análise(Camacam).

Imagem 2 - Estrutura de um compilador



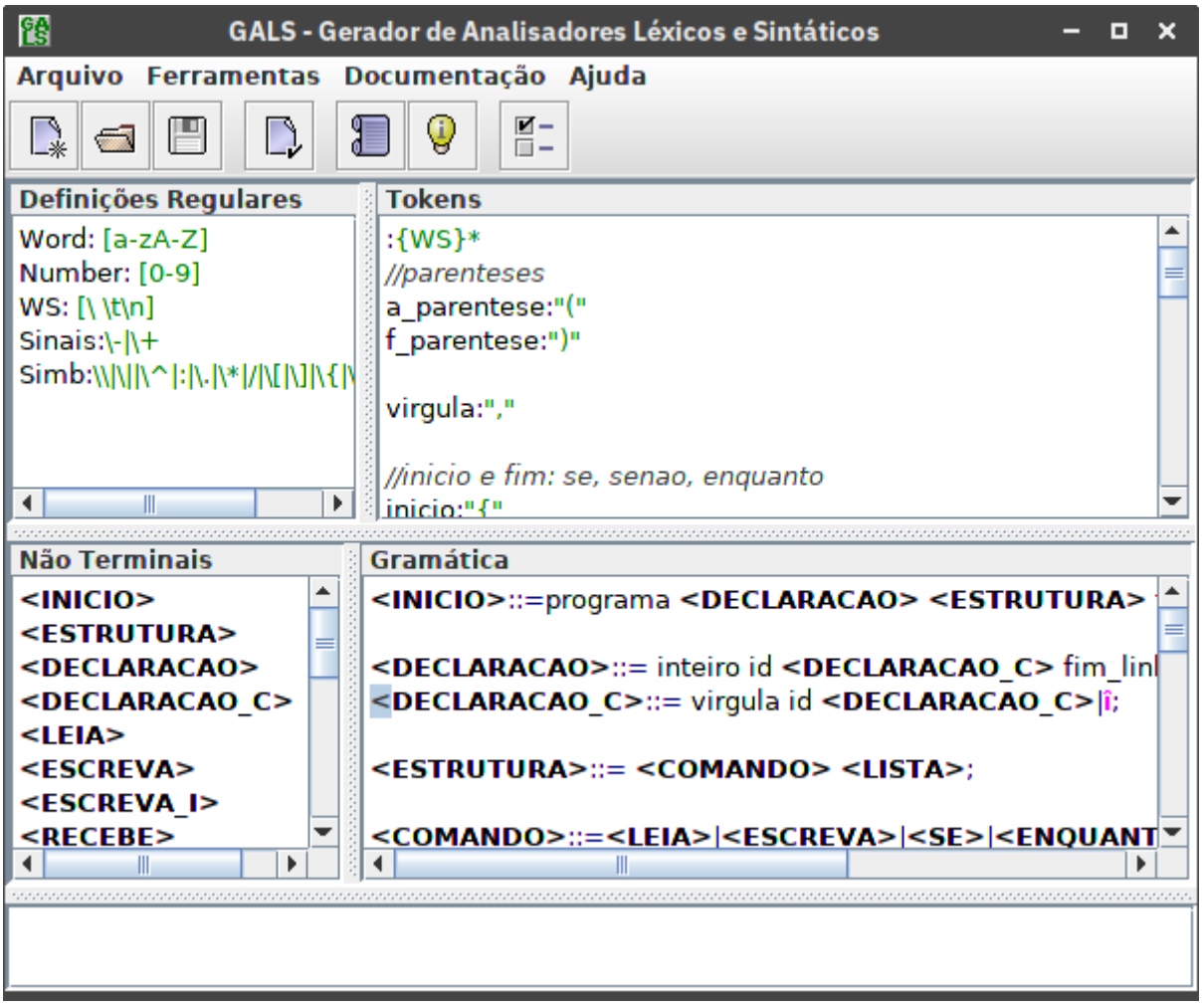
Fonte: (Camacam)

1. ETAPAS DO PROJETO

1.1. Metodologia

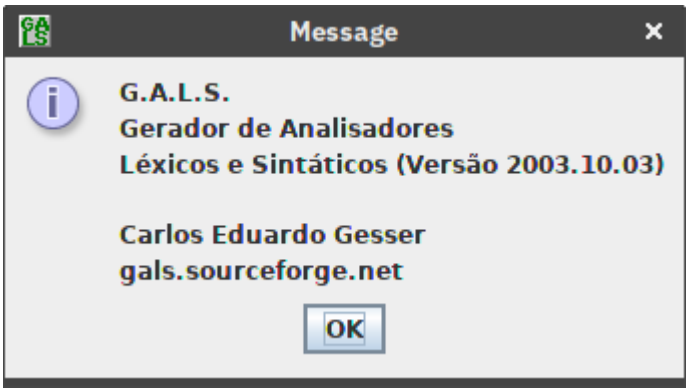
Para ajudar a construção da gramatica LL1, foi utilizado o software gals.

Imagem 2 - interface do gals



Fonte: fonte própria

Imagem 3 – Sobre do software gals



Fonte: própria

O software Gals roda em uma maquina virtual java, openjdk 10.0.2 2018-07-17. Para o analisador lexico, sintatico, semantico e geração de codigo final foi utilizado a linguagem Python na versão 3.6.7. E foi importado a biblioteca bs4 para filtragem da tabela sintatica gerada pelo gals.

Imagem 4 – Tabela sintático gerada pelo gals

	$	a_parentese	f_parentese	virgula	inicio	fim	fim_linha	id	numero	frase	programa	fimprograma	inteiro	leia	escreva	se	senao	enquanto	menor	maior	recebe	soma	subtracao	multiplicacao	divisao
<INICIO>	-	-	-	-	-	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<ESTRUTURA>	-	-	-	-	-	-	-	5	-	-	-	-	-	5	5	5	-	5	-	-	-	-	-	-	-
<DECLARACAO>	-	-	-	-	-	-	-	2	-	-	-	-	1	2	2	2	-	2	-	-	-	-	-	-	-
<DECLARACAO_C>	-	-	-	3	-	-	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<LEIA>	-	-	-	-	-	-	-	-	-	-	-	-	-	13	-	-	-	-	-	-	-	-	-	-	-
<ESCREVA>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	14	-	-	-	-	-	-	-	-	-	-
<ESCREVA_I>	-	-	-	-	-	-	-	16	16	15	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<RECEBE>	-	-	-	-	-	-	-	17	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<COMANDO>	-	-	-	-	-	-	-	10	-	-	-	-	-	6	7	8	-	9	-	-	-	-	-	-	-
<LISTA>	-	-	-	-	-	12	-	11	-	-	-	12	-	11	11	11	-	11	-	-	-	-	-	-	-
<SE>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	18	-	-	-	-	-	-	-	-	-
<SENAO>	-	-	-	-	-	20	-	20	-	-	-	20	-	20	20	20	19	20	-	-	-	-	-	-	-
<ENQUANTO>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	21	-	-	-	-	-	-	-
<V>	-	-	-	-	-	-	-	23	22	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<EXP>	-	24	-	-	-	-	-	24	24	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<EXP_I>	-	-	26	-	-	-	26	-	-	-	-	-	-	-	-	-	-	-	-	-	-	25	25	-	-
<TERM>	-	27	-	-	-	-	-	27	27	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<TERM_I>	-	-	29	-	-	-	29	-	-	-	-	-	-	-	-	-	-	-	-	-	-	29	29	28	28
<FACT>	-	30	-	-	-	-	-	31	31	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<OPERADORES_L>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	32	33	-	-
<OPERADORES_R>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	34	35
<COMPARA>	-	-	-	-	-	-	-	36	36	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<OPERADOR_LOGICO>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	37	38	-	-	-	-	-

0 - <INICIO> ::= programa <DECLARACAO> <ESTRUTURA> fimprograma
1 - <DECLARACAO> ::= inteiro id <DECLARACAO_C> fim_linha
2 - <DECLARACAO> ::= î
3 - <DECLARACAO_C> ::= virgula id <DECLARACAO_C>
4 - <DECLARACAO_C> ::= î
5 - <ESTRUTURA> ::= <COMANDO> <LISTA>
6 - <COMANDO> ::= <LEIA>
7 - <COMANDO> ::= <ESCREVA>
8 - <COMANDO> ::= <SE>
9 - <COMANDO> ::= <ENQUANTO>
10 - <COMANDO> ::= <RECEBE>
11 - <LISTA> ::= <COMANDO> <LISTA>
12 - <LISTA> ::= î
13 - <LEIA> ::= leia a_parentese id f_parentese fim_linha
14 - <ESCREVA> ::= escreva a_parentese <ESCREVA_I> f_parentese fim_linha
15 - <ESCREVA_I> ::= frase
16 - <ESCREVA_I> ::= <V>
17 - <RECEBE> ::= id recebe <EXP> fim_linha
18 - <SE> ::= se a_parentese <COMPARA> f_parentese inicio <ESTRUTURA> fim <SENAO>
19 - <SENAO> ::= senao inicio <ESTRUTURA> fim
20 - <SENAO> ::= î
21 - <ENQUANTO> ::= enquanto a_parentese <COMPARA> f_parentese inicio <ESTRUTURA> fim
22 - <V> ::= numero
23 - <V> ::= id
24 - <EXP> ::= <TERM> <EXP_I>
25 - <EXP_I> ::= <OPERADORES_L> <TERM> <EXP_I>
26 - <EXP_I> ::= î
27 - <TERM> ::= <FACT> <TERM_I>
28 - <TERM_I> ::= <OPERADORES_R> <FACT> <TERM_I>
29 - <TERM_I> ::= î
30 - <FACT> ::= a_parentese <EXP> f_parentese
31 - <FACT> ::= <V>
32 - <OPERADORES_L> ::= soma
33 - <OPERADORES_L> ::= subtracao
34 - <OPERADORES_R> ::= multiplicacao
35 - <OPERADORES_R> ::= divisao
36 - <COMPARA> ::= <V> <OPERADOR_LOGICO> <V>
37 - <OPERADOR_LOGICO> ::= menor
38 - <OPERADOR_LOGICO> ::= maior

Fonte: própria

E foi utilizado a IDE ***PyCharm 2018.3.1 (Community Edition)***:

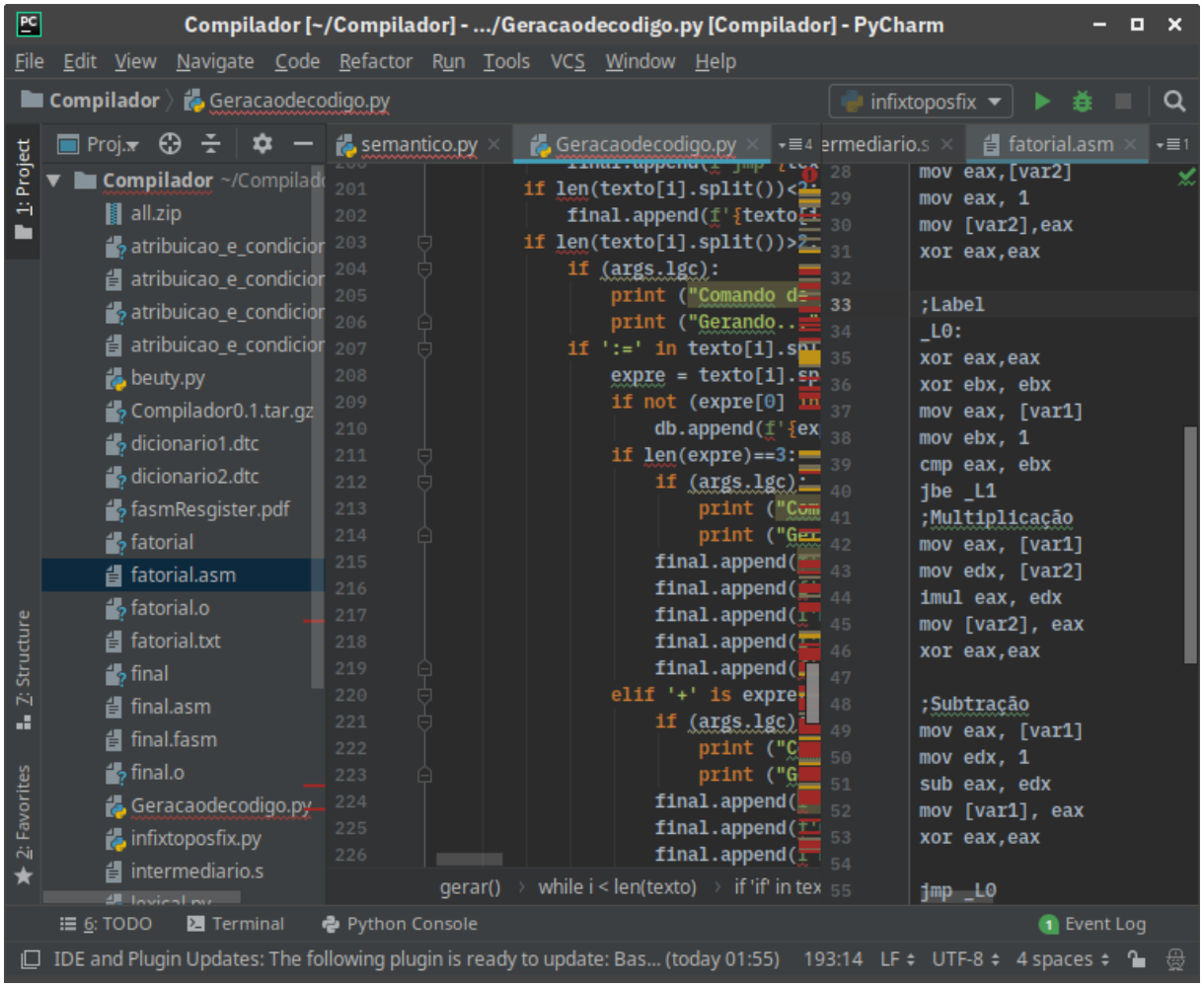


Imagem 4 – interface do PyCharm

E para geração da linguagem de maquina foi utilizado o montados FASM na versão 1.73.02

1.2. Objetivo Geral

Construir um compilador funcional com a gramática LL1, na linguagem python.

1.3. Objetivos Específicos

Conhecer a metodologia da construção de um compilado

Aprender a utilizar a linguagem Python

Conseguir construir um arquivo compilado executavel para Gnu/Linux

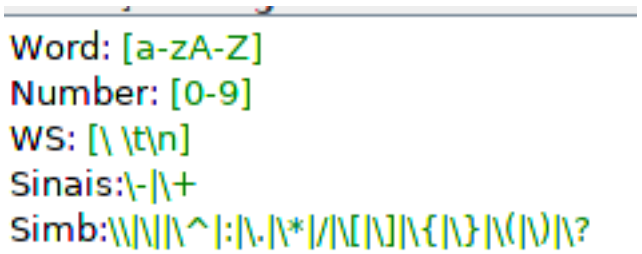
1.4. Dificuldades Encontradas

A ausencia de

1.5. Gramática

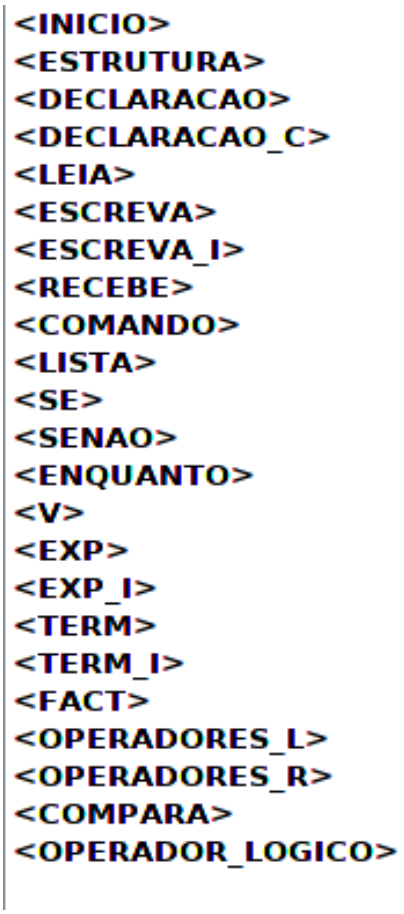
A gramática foi construida usando o programa Gals.

Imagem 4 – Definições regulares



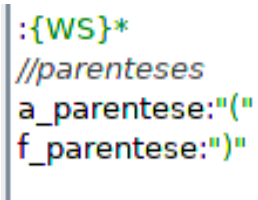
fonte: própria

Imagem 5 – Não terminais



fonte: própria

Imagem 6 – Tokens



Fonte: Própria

Imagem 7 – Gramatica

```
<INICIO>::=programa <DECLARACAO> <ESTRUTURA> fimprograma;

<DECLARACAO>::= inteiro id <DECLARACAO_C> fim_linha|î;
<DECLARACAO_C>::= virgula id <DECLARACAO_C>|î;

<ESTRUTURA>::= <COMANDO> <LISTA>;

<COMANDO>::=<LEIA>|<ESCREVA>|<SE>|<ENQUANTO>|<RECEBE>;
<LISTA>::= <COMANDO> <LISTA>|î;

//Comandos de entrada e saida de dados
<LEIA>::= leia a_parentese id f_parentese fim_linha;
<ESCREVA>::= escreva a_parentese <ESCREVA_I>  f_parentese fim_linha;
<ESCREVA_I>::=frase|<V>;
<RECEBE>::= id recebe <EXP> fim_linha;
//condicional
<SE>::= se a_parentese <COMPARA> f_parentese inicio <ESTRUTURA> fim <SENAO>;
<SENAO>::=senao inicio <ESTRUTURA> fim| î;
//repetição
<ENQUANTO>::=enquanto a_parentese <COMPARA> f_parentese inicio <ESTRUTURA>fim ;

//expressão Matematica
<V>::= numero|id;

<EXP>::= <TERM> <EXP_I>;
<EXP_I>::= <OPERADORES_L><TERM><EXP_I>|î;
<TERM>::=<FACT> <TERM_I>;
<TERM_I>::=<OPERADORES_R><FACT><TERM_I>|î;
<FACT>::=  a_parentese <EXP> f_parentese | <V>;

<OPERADORES_L>::= soma|subtracao;
<OPERADORES_R>::=multiplicacao|divisao;

<COMPARA>::=<V><OPERADOR_LOGICO><V>;
<OPERADOR_LOGICO>::=menor|maior;
```

Fonte: Própria

1.6. Resultados Alcançados

O compilador foi terminado com sucesso, está funcioal. Foi implementado o laço condicional se e senão, equivalente ao if e else na linguagem C, e o laço de repetição enquanto, equivalente ao while em C. E as funções leia e escreva, equivalente ao printf e scanf respectivamente. Conseguir entender melhor a linguagem Python e ter experiência na criação de programa funcional.

1.7. Exemplos de Uso

Na Imagem 8 na linha 2, falta a ‘”’ para fechar a frase, e na imagem 9 a mensagem de erro será apresentada no terminal:

Imagem 8 – Erro de léxico

```
1  programa
2  inteiro var1, var2;
3  escreva("Escreva um numero: );
4  leia(var1);
5  var2=1;
6  enquanto(var1 > 1){
7   var2 = var1 * var2 ;
8   var1 = var1 - 1;
9  }
10 escreva(var2);
11 escreva("\n");
12 fimprograma
```

Fonte: Própria

Imagem 9 – Mensagem apresentada no console.

```
##################################################################
Erro lexico
'"' não esperado na linha 3 e coluna 7
" de fechamento não encontrada
##################################################################
Erro lexico
':' não esperado na linha 3 e coluna 20
caracter desconhecido
##################################################################
Erro sintático
'um' inesperado linha 3, coluna 17
escreva("Escreva um numero: );
                ^
```

Fonte: Própria

Na imagem 10 apresenta um erro semantico, ausencia de ponto e virgula. Na imagem 11 mostra o erro que ira apresentar na tela.

Imagem 10: Erro sintático

```
 1 programa
 2 inteiro var1, var2;
 3 escreva("Escreva um numero: ");
 4 leia(var1)
 5 var2=1;
 6 enquanto(var1 > 1){
 7  var2 = var1 * var2 ;
 8  var1 = var1 - 1;
 9 }
10 escreva(var2);
11 escreva("\n");
12 fimprograma
```

Fonte: Própria

Imagem 11 – Mensagem de erro aprensentado no console

```
########################################
Erro sintático
')' inesperado linha 4, coluna 9
leia(var1)
        ^
Era esperado: ;
```

Fonte: Própria

erro semântico

funcionamento correto, pelo menos um programa (com a apresentação dos códigos gerados, intermediáro/final, sugestão: cálculo de fatorial, sequencia de números primos, cálculo de máximo divisor comum, cálculo de pi através da Fórmula de Leibniz https://pt.wikipedia.org/wiki/Fórmula_de_Leibniz_para_π).

CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Veja dicas em http://www.portaltccendo.com.br/conclusao-do-tcc/

REFERÊNCIAS

RICARTE, Ivan. **Introdução à compilação**. Elsevier Brasil, 2012.

Ghuloum, Abdulaziz. **An incremental approach to compiler construction.** In: *Proceedings of the 2006 Scheme and Functional Programming Workshop, Portland, OR. Citeseer*. 2006.

Cooper, Keith, and Linda Torczon. ***Engineering a compiler***. Elsevier, 2011.

Camacam, Savio. **Compiladores para Humanos.** Disponível em <https://johnidm.gitbooks.io/compiladores-para-humanos/content/part1/introduction-and-overview-about-compilers.html>. Acesso em: 16 dez. 2018.