Universidade Estadual de Santa Cruz – UESC

Projeto 4: Compilador

Docente César Alberto Bravo Pariente

Discente Matheus Miranda Brandão

Matrícula 201820065

Disciplina Compiladores.

Curso Ciência da Computação

Semestre 2022.2

Ilhéus – BA
2022

Índice

Índice **2**

1. Introdução **4**

1.1 Objetivo de um compilador 4

1.2 Gramática Livre de Contexto LL(1) 4

1.3 Linguagem de Programação 5

2. Módulos de um compilador **5**

2.1 Scanner (Analisador Léxico) 5

2.2 Parser (Analisador Sintático) 5

2.3 Ada2Asa (Árvore Sintática para Árvore Abstrata) 6

2.4 Asa2Np (Árvore Abstrata para Notação Polonesa) 7

2.5 Np2Gci 7

2.6 Gci2Gco 7

2.7 Compilação de todos os módulos 7

3. UML **8**

3.1 Diagrama de Sequência: 8

3.2 Casos de uso (Sucesso): 8

3.3 Casos de uso (Erro): 8

3.3.1 Token inesperado: 9

3.3.2 Cadeia de caracteres consumida e pilha não está vazia: 9

3.3.3 Cadeia de caracteres não consumida e pilha está vazia: 9

3.3.4 Função que não existe é atribuída a uma variável: 9

3.4 Cronograma: 9

4. Testes **10**

4.1 m(){h=g().;r(y);} 12

4.2 m(){f(x){f(y){.j=1}};z=5.;r(7);} 13

4.3 m(){f(x){f(y){j=1}};z=5.;r(7);} 14

4.4 g(){k=0.;r(8);}m(){f(x){w(y){f(1){h=g()}}};z=5.;r(7);} 17

4.5 m(){w(1){o(x;(1+1);1){f((1*(x+y))){k=2}}}.;r(7);} 20

4.6 n(){j=6.;r(4);}g(){z=x.;r(6);}m(){h=g();i=n();j=1.;r(y);} 24

Referências **29**

1. Introdução

Este projeto consiste na implementação em linguagem C de um compilador simples para uma gramática livre de contexto que ao fim o transforma em código objeto para execução em simulador de p-code machine. Os módulos desenvolvidos para seu funcionamento consistem em: Scanner (análise léxica) que percorre o arquivo fonte interagindo diretamente com o Analisador sintático; Parser (análise sintática) analisa a entrada para verificar se satisfaz a estrutura imposta pela gramática, ao fim gerando uma Árvore Sintática (ADA); Ada2Asa converte a Árvore Sintática em Abstrata (ASA), assim removendo itens sem elemento semântico; Asa2Np converte a ASA em notação polonesa (NP) para gerar o código intermediário. Ao fim do relatório, no tópico "Referências", encontra-se a implementação e casos de teste.

1.1 Objetivo de um compilador

Consiste na tradução de um código descrito em uma linguagem, normalmente de alto nível, que obedeça às regras da gramática livre de contexto descrita abaixo para um programa equivalente em código de máquina para arquitetura p-code machine.

1.2 Gramática Livre de Contexto LL(1)

P1: S → M | GM | NGM

P4: N → n(){ A; r(E); }

P5: G → g(){ A; r(E); }

P6: M → m() { A; r(E); }

P7: A → CB

P8: B → . | ;CB

P10: E → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | x | y | (EXE)

P23: X → + | - | * | /

P27: C → h=g() | i=n() | j=E | k=E | z=E | (EXE) | w(E){ CD | f(E){ CD | o(E;E;E){ CD

P36: D → } | ;CD

1.3 Linguagem de Programação

Em nossa linguagem deve-se declarar obrigatoriamente a main, além disso pode-se adicionar, facultativamente, até duas funções seguindo a regra dessa ordem: "m() { … }"; "g() { …} m() { … }"; "n() { … } g() { … } m() { … }".

O escopo dessas funções obedecem a mesma estrutura: {A; r(E); }, onde o "A" consiste em um conjunto de instruções que podem ser concatenadas por ";" e por fim um retorno de uma variável "r(E)".

Uma variável pode ser um valor numérico (0, 1 , … 8, 9), uma nova variável ("x", "y"), ou uma expressão aritmética fechada ("EXE"). O "X" representa os operadores que são do tipo binário ("+", "-", "*", "/").

O conjunto de instruções pode ser a atribuição de um valor uma variável que limita-se a duas categorias: atribuições de funções ("h=g()", "i=n()") e atribuições de valores ("j=E", "k=E", "z=E"). O conjunto de instruções também podem consistir em operações lógicas como o if ("f(E){ CD") e laços de repetição como o while ("w(E){ CD") e for("o(E;E;E){ CD").

Para finalizar o conjunto de instruções aninhados deve-se colocar uma flag de ponto final ".", ou caso estejam dentro de operações lógicas ou laços de repetição por um "}".

2. Módulos de um compilador

Apresentamos separadamente cada módulo que compõe nosso compilador, em nosso projeto todos possuem autonomia e podem ser executados individualmente, ao fim pode-se compilar todos de uma só vez. Obs:. Os módulos de Np2Gci e Gci2Gco não foram concluídos.

2.1 Scanner (Analisador Léxico)

Consiste num processo de varredura caractere por caractere do código fonte fornecido pelo usuário, traduzindo símbolos léxicos em tokens. Como o compilador desenvolvido foi uma versão simplificada, optou-se pela utilização de leitura fgetc() em um arquivo.

2.2 Parser (Analisador Sintático)

Este módulo consiste na análise do código fonte verificando se a cadeia de caracteres fornecida respeita as regras da gramática formal. Podemos separar este módulo em dois tópicos: Tabela de Parsing, Árvore Sintática.

- Tabela de Parsing: Utilizando o Scanner inicia-se um autômato de pilha que ao identificar um token referente a uma produção empilha todos os elementos contendo essa determinada produção, então vai consumindo os tokens do topo da pilha desempilhando-os. Caso identifique um erro sintático encerrará a execução e informará o elemento não esperado, caso o arquivo fonte esteja correto, o mesmo foi percorrido por completo e o topo da pilha estará vazio, ao fim retornando a lista de produções geradas.

  Esta etapa dialoga diretamente com o módulo anterior, para executá-lo necessita passar como argumento o arquivo de input, segue o exemplo:

  cd 01\ -\ Parser⁄Scanner/

  gcc parse_generator.c -o parse_generator

  ./parse_generator ../Examples/example

- Árvore Sintática: Ao terminar a etapa anterior percorre-se a lista de produções geradas os transformando em uma árvore n-ária onde tamanho corresponde ao pior caso dessa gramática (12 filhos). Para compactação essa árvore é montada em formato de vetor com um "hash" correspondente a sua real posição na árvore, ao fim retornando essa ADA.

  Esta etapa utiliza como entrada o output da anterior, para executá-lo necessita passar como argumento o arquivo de input, segue o exemplo:

  cd /02\ -\ Ada/

  gcc ada.c -o ada

  ./ada ../01\ -\ Parser⁄Scanner/productions

2.3 Ada2Asa (Árvore Sintática para Árvore Abstrata)

Este módulo é responsável pela transformação da árvore sintática em árvore abstrata, removendo qualquer elemento que não possua real valor semântico (ex: parênteses, vírgulas, chaves, quebras de linha, etc). A partir deste módulo o único erro que pode ser encontrado é a atribuição de uma função inexistente a uma variável, por ser um problema lógico. Percorre-se a Ada utilizando a estratégia de Backtracking para converter uma árvore de 12 filhos para a binária. Ao fim retornando a Asa convertida.

Este módulo utiliza como entrada o output do módulo anterior, para executá-lo necessita passar como argumento o arquivo de input, segue o exemplo:

cd 03\ -\ Ada2Asa/

gcc ada2asa.c -o ada2asa

./ada2asa ../02\ -\ Ada/ada_tree

2.4 Asa2Np (Árvore Abstrata para Notação Polonesa)

Este módulo tem como função converter a árvore abstrata (Asa) para a notação polonesa e notação polonesa reversa. Para a conversão a árvore é percorrida utilizando o recurso de pré-ordem (notação polonesa) e pós-ordem (notação polonesa reversa).

Este módulo utiliza como entrada o output do módulo anterior, para executá-lo necessita passar como argumento o arquivo de input, segue o exemplo:

cd 04\ -\ Asa2Np/

gcc asa2np.c -o asa2np

./asa2np ../03\ -\ Ada2Asa/asa_tree

2.5 Np2Gci

Este módulo tem como função converter a notação polonesa e notação polonesa reversa para código intermediário. O código intermediário corresponde em uma visão simplificada do código objeto que pode ser utilizado posteriormente para otimizações e reuso.

2.6 Gci2Gco

Este módulo tem como função converter o código intermediário para código objeto (máquina) para p-code machine.

2.7 Compilação de todos os módulos

Este módulo tem como função compilar todos os módulos em apenas um, para sua execução necessita-se apenas chamar o arquivo de input.

cd 05\ -\ Compiler/

gcc main.c parse_generator.c ada.c ada2asa.c asa2np.c -o main

./main ../Examples/example

3. UML

Serão apresentados detalhadamente diagramas de sequência e casos de uso, além de um cronograma de desenvolvimento do projeto.

3.1 Diagrama de Sequência:

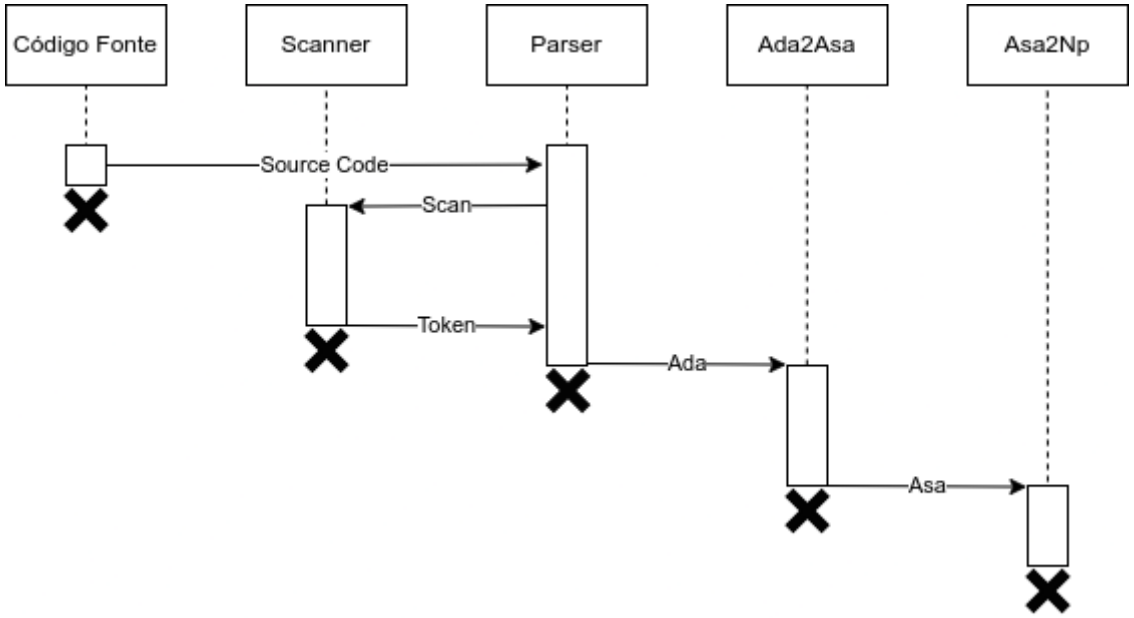


Figura 1. Diagrama de Sequência.

3.2 Casos de uso (Sucesso):

1. Usuário entra com o código fonte;
2. O compilador inicia o processo de parser;
3. O parser em conjunto com o scanner analisa caractere por caractere;
4. Parser consumiu todos os símbolos da pilha;
5. Scanner identifica que o arquivo encerrou;
6. Lista de produções é gerada;
7. A partir das produções é gerada a árvore sintática;
8. A árvore sintática é convertida em árvore abstrata;
9. Árvore abstrata é percorrida em pré-ordem gerando a notação polonesa e em pós-ordem gerando a notação polonesa reversa.
10. …

3.3 Casos de uso (Erro):

3.3.1 Token inesperado:

1. Usuário entra com o código fonte;
2. O compilador inicia o processo de parser;
3. O parser em conjunto com o scanner analisa caractere por caractere;
4. Encontra um token diferente do esperado pelo topo da pilha;
5. Erro é gerado.

3.3.2 Cadeia de caracteres consumida e pilha não está vazia:

1. Usuário entra com o código fonte;
2. O compilador inicia o processo de parser;
3. O parser em conjunto com o scanner analisa caractere por caractere;
4. Scanner identifica que o arquivo encerrou;
5. A pilha do parser ainda possui símbolos a serem consumidos;
6. Erro é gerado.

3.3.3 Cadeia de caracteres não consumida e pilha está vazia:

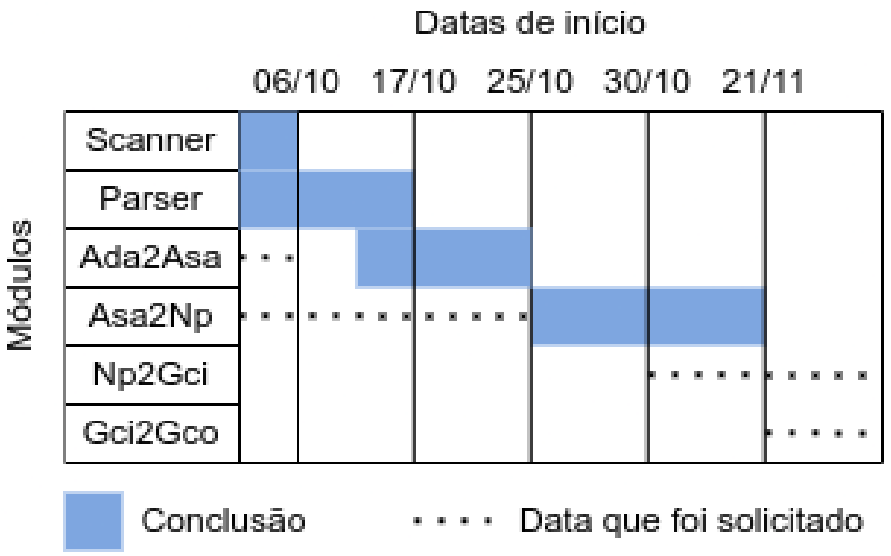
1. Usuário entra com o código fonte;
2. O compilador inicia o processo de parser;
3. O parser em conjunto com o scanner analisa caractere por caractere;
4. Parser consumiu todos os símbolos da pilha;
5. Scanner identifica que o arquivo não encerrou;
6. Erro é gerado.

3.3.4 Função que não existe é atribuída a uma variável:

1. Usuário entra com o código fonte;
2. O compilador inicia o processo de parser;
3. O parser em conjunto com o scanner analisa caractere por caractere;
4. Parser consumiu todos os símbolos da pilha;
5. Scanner identifica que o arquivo encerrou;
6. Lista de produções é gerada;
7. A partir das produções é gerada a árvore sintática;
8. No processo de conversão para árvore abstrata é identificada a atribuição;
9. Verifica-se a inexistência dessa função;
10. Erro é gerado.

3.4 Cronograma:

Cada módulo do compilador foi gerado ao longo do semestre como atividades individuais, ao fim sendo unificado em apenas um projeto.



4. Testes

Caso a entrada dada tenha algum erro sintático o programa irá imprimir a tabela parsing até o momento, e então informará o token inesperado. Em caso de erro lógico foram tratados um caso específico, onde uma variável recebe uma função inexistente, onde encerrará o programa e informará o erro. Sempre será imprimida a saída de cada módulo.

Para criação de palavras compatíveis com a linguagem gerada pela GLC foi utilizado o website "CFG Developer".

- A primeira entrada será igual para todos os casos:

  P1: S -> M

  P2: S -> GM

  P3: S -> NGM

  P4: N -> n(){A;r(E);}

  P5: G -> g(){A;r(E);}

  P6: M -> m(){A;r(E);}

  P7: A -> CB

  P8: B -> .

  P9: B -> ;CB

P10: E -> 0

P11: E -> 1

P12: E -> 2

P13: E -> 3

P14: E -> 4

P15: E -> 5

P16: E -> 6

P17: E -> 7

P18: E -> 8

P19: E -> 9

P20: E -> x

P21: E -> y

P22: E -> (EXE)

P23: X -> +

P24: X -> -

P25: X -> *

P26: X -> /

P27: C -> h=g()

P28: C -> i=n()

P29: C -> j=E

P30: C -> k=E

P31: C -> z=E

P32: C -> (EXE)

P33: C -> w(E){CD

P34: C -> f(E){CD

P35: C -> o(E;E;E){CD

P36: D -> }

P37: D -> ;CD

Arvore tera 12 filhos

4.1 m(){h=g().;r(y);}

m(){h=g().;r(y);}

i	Qi	Token	Stack	Pi
0	Q0	m		-
1	Q1	m	S	P1
2	Q1	m	M	P6
2	Q1	(	(	-
2	Q1	)	)	-
2	Q1	{	{	-
2	Q1	h	A	-
3	Q1	h	A	P7
4	Q1	h	C	P27
4	Q1	=	=	-
4	Q1	g	g	-
4	Q1	(	(	-
4	Q1	)	)	-
4	Q1	.	B	-
5	Q1	.	B	P8
5	Q1	r	r	-
5	Q1	(	(	-
5	Q1	y	E	-
6	Q1	y	E	P21
6	Q1	;	;	-
6	Q1	}	}	-

6 Q1 -

Palavra aceita.

Producoes: P1 P6 P7 P27 P8 P21

****

[0|0|S], [1|1|M], [2|13|m], [3|14|(], [4|15|)], [5|16|{], [6|17|A], [7|18|;], [8|19|r], [9|20|(], [10|21|E], [11|22|)], [12|23|;], [13|24|}], [14|205|C], [15|206|B], [16|2461|h], [17|2462|=], [18|2463|g], [19|2464|(], [20|2465|)], [21|2473|.], [22|253|y]

****

Erro função nao declarada !

4.2 m(){f(x){f(y){.j=1}};z=5.;r(7);}

m(){f(x){f(y){.j=1}};z=5.;r(7);}

i Qi Token Stack Pi

0 Q0 m -

1 Q1 m S P1

2 Q1 m M P6

2 Q1 ((-

2 Q1)) -

2 Q1 { { -

2 Q1 f A -

3 Q1 f A P7

4 Q1 f C P34

4 Q1 ((-

4 Q1 x E -

5 Q1 x E P20

5 Q1 { { -

5 Q1 f C -

6 Q1 f C P34

6 Q1 ((-

6 Q1 y E -

7 Q1 y E P21

7 Q1 { { -

7 Q1 . C -

Error! Token = '.' nao esperado.

Producoes: P1 P6 P7 P34 P20 P34 P21

4.3 m(){f(x){f(y){j=1}};z=5.;r(7);}

m(){f(x){f(y){j=1}};z=5.;r(7);}

i Qi Token Stack Pi

0 Q0 m -

1 Q1 m S P1

2 Q1 m M P6

2 Q1 ((-

2 Q1)) -

2 Q1 { { -

2 Q1 f A -

3 Q1 f A P7

4 Q1 f C P34

4 Q1 ((-

4 Q1 x E -

5	Q1	x	E	P20
5	Q1	{	{	-
5	Q1	f	C	-
6	Q1	f	C	P34
6	Q1	(	(	-
6	Q1	y	E	-
7	Q1	y	E	P21
7	Q1	{	{	-
7	Q1	j	C	-
8	Q1	j	C	P29
8	Q1	=	=	-
8	Q1	1	E	-
9	Q1	1	E	P11
10	Q1	}	D	P36
11	Q1	}	D	P36
12	Q1	;	B	P9
12	Q1	z	C	-
13	Q1	z	C	P31
13	Q1	=	=	-
13	Q1	5	E	-
14	Q1	5	E	P15
15	Q1	.	B	P8
15	Q1	r	r	-
15	Q1	(	(	-
15	Q1	7	E	-
16	Q1	7	E	P17
16	Q1	;	;	-

16 Q1 } } -

16 Q1 -

Palavra aceita.

Producoes: P1 P6 P7 P34 P20 P34 P21 P29 P11 P36 P36 P9 P31 P15 P8 P17

****

[0|0|S], [1|1|M], [2|13|m], [3|14|(], [4|15|)], [5|16|{], [6|17|A], [7|18|;], [8|19|r], [9|20|(], [10|21|E], [11|22|)], [12|23|;], [13|24|}], [14|205|C], [15|206|B], [16|2461|f], [17|2462|(], [18|2463|E], [19|2464|)], [20|2465|{], [21|2466|C], [22|2467|D], [23|29557|x], [24|29593|f], [25|29594|(], [26|29595|E], [27|29596|)], [28|29597|{], [29|29598|C], [30|29599|D], [31|355141|y], [32|355177|j], [33|355178|=], [34|355179|E], [35|4262149|1], [36|355189|}], [37|29605|}], [38|2473|;], [39|2474|C], [40|2475|B], [41|29689|z], [42|29690|=], [43|29691|E], [44|356293|5], [45|29701|.], [46|253|7]

****

[0|0|m], [1|1||], [2|2|r], [3|3|f], [4|4||], [5|7|x], [6|8||], [7|17|f], [8|18|}], [9|35|y], [10|36||], [11|73|=], [12|74|}], [13|147|j], [14|148|1], [15|9|=], [16|10|.], [17|19|z], [18|20|5], [19|5|7]

****

Notacao ponolesa: [m], [f], [x], [f], [y], [=], [j], [1], [}], [}], [=], [z], [5], [.], [r], [7], [f], [x], [f], [y], [=], [j], [1], [}], [}], [=], [z], [5], [.]

Notacao ponolesa reversa: [x], [y], [j], [1], [=], [}], [f], [}], [f], [z], [5], [=], [.], [7], [r], [x], [y], [j], [1], [=], [}], [f], [}], [f], [z], [5], [=], [.], [m]

****
****

4.4 g(){k=0.;r(8);}m(){f(x){w(y){f(1){h=g()}}};z=5.;r(7);}

g(){k=0.;r(8);}m(){f(x){w(y){f(1){h=g()}}};z=5.;r(7);}

i	Qi	Token	Stack	Pi
0	Q0	g		-
1	Q1	g	S	P2
2	Q1	g	G	P5
2	Q1	(	(	-
2	Q1	)	)	-
2	Q1	{	{	-
2	Q1	k	A	-
3	Q1	k	A	P7
4	Q1	k	C	P30
4	Q1	=	=	-
4	Q1	0	E	-
5	Q1	0	E	P10
6	Q1	.	B	P8
6	Q1	r	r	-
6	Q1	(	(	-
6	Q1	8	E	-
7	Q1	8	E	P18
7	Q1	;	;	-
7	Q1	}	}	-
7	Q1	m	M	-
8	Q1	m	M	P6
8	Q1	(	(	-

8	Q1	)	)	-
8	Q1	{	{	-
8	Q1	f	A	-
9	Q1	f	A	P7
10	Q1	f	C	P34
10	Q1	(	(	-
10	Q1	x	E	-
11	Q1	x	E	P20
11	Q1	{	{	-
11	Q1	w	C	-
12	Q1	w	C	P33
12	Q1	(	(	-
12	Q1	y	E	-
13	Q1	y	E	P21
13	Q1	{	{	-
13	Q1	f	C	-
14	Q1	f	C	P34
14	Q1	(	(	-
14	Q1	1	E	-
15	Q1	1	E	P11
15	Q1	{	{	-
15	Q1	h	C	-
16	Q1	h	C	P27
16	Q1	=	=	-
16	Q1	g	g	-
16	Q1	(	(	-
16	Q1	)	)	-

16 Q1 } D -

17 Q1 } D P36

18 Q1 } D P36

19 Q1 } D P36

20 Q1 ; B P9

20 Q1 z C -

21 Q1 z C P31

21 Q1 = = -

21 Q1 5 E -

22 Q1 5 E P15

23 Q1 . B P8

23 Q1 r r -

23 Q1 ((-

23 Q1 7 E -

24 Q1 7 E P17

24 Q1 ; ; -

24 Q1 } } -

24 Q1 -

Palavra aceita.

Producoes: P2 P5 P7 P30 P10 P8 P18 P6 P7 P34 P20 P33 P21 P34 P11 P27 P36 P36 P36 P9 P31 P15 P8 P17

****

[0|0|S], [1|1|G], [2|2|M], [3|13|g], [4|14|(], [5|15|)], [6|16|{], [7|17|A], [8|18|;], [9|19|r], [10|20|(], [11|21|E], [12|22|)], [13|23|;], [14|24|}], [15|205|C], [16|206|B], [17|2461|k], [18|2462|=], [19|2463|E], [20|29557|0], [21|2473|.], [22|253|8], [23|25|m], [24|26|(], [25|27|)], [26|28|{], [27|29|A], [28|30|;], [29|31|r], [30|32|(], [31|33|E], [32|34|)],

[33|35|;], [34|36|}], [35|349|C], [36|350|B], [37|4189|f], [38|4190|(], [39|4191|E], [40|4192|)], [41|4193|{], [42|4194|C], [43|4195|D], [44|50293|x], [45|50329|w], [46|50330|(], [47|50331|E], [48|50332|)], [49|50333|{], [50|50334|C], [51|50335|D], [52|603973|y], [53|604009|f], [54|604010|(], [55|604011|E], [56|604012|)], [57|604013|{], [58|604014|C], [59|604015|D], [60|7248133|1], [61|7248169|h], [62|7248170|=], [63|7248171|g], [64|7248172|(], [65|7248173|)], [66|7248181|}], [67|604021|}], [68|50341|}], [69|4201|;], [70|4202|C], [71|4203|B], [72|50425|z], [73|50426|=], [74|50427|E], [75|605125|5], [76|50437|.], [77|397|7]

****

[0|0|m], [1|1||], [2|2|r], [3|3|f], [4|4||], [5|7|x], [6|8||], [7|17|w], [8|18|}], [9|35|y], [10|36||], [11|73|f], [12|74|}], [13|147|1], [14|148||], [15|297|=], [16|298|}], [17|595|h], [18|596|g], [19|1193||], [20|1194|r], [21|2387|=], [22|2388|.], [23|4775|k], [24|4776|0], [25|2389|8], [26|9|=], [27|10|.], [28|19|z], [29|20|5], [30|5|7]

****

Notacao ponolesa: [8], [r], [g], [=], [}], [f], [}], [w], [}], [f], [z], [5], [=], [.], [h], [8], [}], [}], [}], [=], [z], [5], [.], [r], [7], [f], [x], [w], [y], [f], [1], [=], [h], [g], [=], [k], [0], [.], [r], [8], [}], [}], [}], [=], [z], [5], [.]

Notacao ponolesa reversa: [x], [y], [1], [h], [k], [0], [=], [.], [8], [r], [g], [=], [}], [f], [}], [w], [}], [f], [z], [5], [=], [.], [7], [r], [x], [y], [1], [h], [k], [0], [=], [.], [8], [r], [g], [=], [}], [f], [}], [w], [}], [f], [z], [5], [=], [.], [h]

****

4.5 m(){w(1){o(x;(1+1);1){f((1*(x+y))){k=2}}}.;r(7);}

m(){w(1){o(x;(1+1);1){f((1*(x+y))){k=2}}}.;r(7);}

i	Qi	Token	Stack	Pi
0	Q0	m		-

1	Q1	m	S	P1
2	Q1	m	M	P6
2	Q1	(	(	-
2	Q1	)	)	-
2	Q1	{	{	-
2	Q1	w	A	-
3	Q1	w	A	P7
4	Q1	w	C	P33
4	Q1	(	(	-
4	Q1	1	E	-
5	Q1	1	E	P11
5	Q1	{	{	-
5	Q1	o	C	-
6	Q1	o	C	P35
6	Q1	(	(	-
6	Q1	x	E	-
7	Q1	x	E	P20
7	Q1	(	E	-
8	Q1	(	E	P22
8	Q1	1	E	-
9	Q1	1	E	P11
10	Q1	+	X	P23
11	Q1	1	E	P11
11	Q1	;	;	-
11	Q1	1	E	-
12	Q1	1	E	P11
12	Q1	{	{	-

12 Q1 f C -

13 Q1 f C P34

13 Q1 ((-

13 Q1 (E -

14 Q1 (E P22

14 Q1 1 E -

15 Q1 1 E P11

16 Q1 * X P25

17 Q1 (E P22

17 Q1 x E -

18 Q1 x E P20

19 Q1 + X P23

20 Q1 y E P21

20 Q1)) -

20 Q1)) -

20 Q1 { { -

20 Q1 k C -

21 Q1 k C P30

21 Q1 = = -

21 Q1 2 E -

22 Q1 2 E P12

23 Q1 } D P36

24 Q1 } D P36

25 Q1 } D P36

26 Q1 . B P8

26 Q1 r r -

26 Q1 ((-

26 Q1 7 E -

27 Q1 7 E P17

27 Q1 ; ; -

27 Q1 } } -

27 Q1 -

Palavra aceita.

Producoes: P1 P6 P7 P33 P11 P35 P20 P22 P11 P23 P11 P11 P34 P22 P11 P25 P22 P20 P23 P21 P30 P12 P36 P36 P36 P8 P17

****

[0|0|S], [1|1|M], [2|13|m], [3|14|(], [4|15|)], [5|16|{], [6|17|A], [7|18|;], [8|19|r], [9|20|(], [10|21|E], [11|22|)], [12|23|;], [13|24|}], [14|205|C], [15|206|B], [16|2461|w], [17|2462|(], [18|2463|E], [19|2464|)], [20|2465|{], [21|2466|C], [22|2467|D], [23|29557|1], [24|29593|o], [25|29594|(], [26|29595|E], [27|29596|;], [28|29597|E], [29|29598|;], [30|29599|E], [31|29600|)], [32|29601|{], [33|29602|C], [34|29603|D], [35|355141|x], [36|355165|(], [37|355166|E], [38|355167|X], [39|355168|E], [40|355169|)], [41|4261993|1], [42|4262005|+], [43|4262017|1], [44|355189|1], [45|355225|f], [46|355226|(], [47|355227|E], [48|355228|)], [49|355229|{], [50|355230|C], [51|355231|D], [52|4262725|(], [53|4262726|E], [54|4262727|X], [55|4262728|E], [56|4262729|)], [57|51152713|1], [58|51152725|*], [59|51152737|(], [60|51152738|E], [61|51152739|X], [62|51152740|E], [63|51152741|)], [64|613832857|x], [65|613832869|+], [66|613832881|y], [67|4262761|k], [68|4262762|=], [69|4262763|E], [70|51153157|2], [71|4262773|}], [72|355237|}], [73|29605|}], [74|2473|.], [75|253|7]

****

[0|0|m], [1|1||], [2|2|r], [3|3|w], [4|4|.], [5|7|1], [6|8||], [7|17|o], [8|18|}], [9|35|;], [10|71|X], [11|36|;], [12|73|x], [13|74|;], [14|149||], [15|150|1], [16|143|1], [17|144|1], [18|299|f], [19|300|}], [20|599|X], [21|600||], [22|1199|1], [23|1200|X], [24|2401|x], [25|2402|y], [26|1201|=], [27|1202|}], [28|2403|k], [29|2404|2], [30|5|7]

****
****

Notacao ponolesa: [m], [w], [1], [o], [;], [X], [1], [1], [}], [.], [r], [7], [w], [1], [o], [;], [X], [1], [1], [}], [.]

Notacao ponolesa reversa: [1], [1], [1], [X], [;], [o], [}], [w], [.], [7], [r], [1], [1], [1], [X], [;], [o], [}], [w], [.], [m]

****
****

4.6 n(){j=6.;r(4);}g(){z=x.;r(6);}m(){h=g();i=n();j=1.;r(y);}

n(){j=6.;r(4);}g(){z=x.;r(6);}m(){h=g();i=n();j=1.;r(y);}

i	Qi	Token	Stack	Pi
0	Q0	n		-
1	Q1	n	S	P3
2	Q1	n	N	P4
2	Q1	(	(	-
2	Q1	)	)	-
2	Q1	{	{	-
2	Q1	j	A	-
3	Q1	j	A	P7
4	Q1	j	C	P29
4	Q1	=	=	-
4	Q1	6	E	-
5	Q1	6	E	P16
6	Q1	.	B	P8
6	Q1	r	r	-

6 Q1 ((-

6 Q1 4 E -

7 Q1 4 E P14

7 Q1 ; ; -

7 Q1 } } -

7 Q1 g G -

8 Q1 g G P5

8 Q1 ((-

8 Q1)) -

8 Q1 { { -

8 Q1 z A -

9 Q1 z A P7

10 Q1 z C P31

10 Q1 = = -

10 Q1 x E -

11 Q1 x E P20

12 Q1 . B P8

12 Q1 r r -

12 Q1 ((-

12 Q1 6 E -

13 Q1 6 E P16

13 Q1 ; ; -

13 Q1 } } -

13 Q1 m M -

14 Q1 m M P6

14 Q1 ((-

14 Q1)) -

14	Q1	{	{	-
14	Q1	h	A	-
15	Q1	h	A	P7
16	Q1	h	C	P27
16	Q1	=	=	-
16	Q1	g	g	-
16	Q1	(	(	-
16	Q1	)	)	-
16	Q1	;	B	-
17	Q1	;	B	P9
17	Q1	i	C	-
18	Q1	i	C	P28
18	Q1	=	=	-
18	Q1	n	n	-
18	Q1	(	(	-
18	Q1	)	)	-
18	Q1	;	B	-
19	Q1	;	B	P9
19	Q1	j	C	-
20	Q1	j	C	P29
20	Q1	=	=	-
20	Q1	1	E	-
21	Q1	1	E	P11
22	Q1	.	B	P8
22	Q1	r	r	-
22	Q1	(	(	-
22	Q1	y	E	-

23 Q1 y E P21

23 Q1 ; ; -

23 Q1 } } -

23 Q1 -

Palavra aceita.

Producoes: P3 P4 P7 P29 P16 P8 P14 P5 P7 P31 P20 P8 P16 P6 P7 P27 P9 P28 P9 P29 P11 P8 P21

****

[0|0|S], [1|1|N], [2|2|G], [3|3|M], [4|13|n], [5|14|(], [6|15|)], [7|16|{], [8|17|A], [9|18|;], [10|19|r], [11|20|(], [12|21|E], [13|22|)], [14|23|;], [15|24|}], [16|205|C], [17|206|B], [18|2461|j], [19|2462|=], [20|2463|E], [21|29557|6], [22|2473|.], [23|253|4], [24|25|g], [25|26|(], [26|27|)], [27|28|{], [28|29|A], [29|30|;], [30|31|r], [31|32|(], [32|33|E], [33|34|)], [34|35|;], [35|36|}], [36|349|C], [37|350|B], [38|4189|z], [39|4190|=], [40|4191|E], [41|50293|x], [42|4201|.], [43|397|6], [44|37|m], [45|38|(], [46|39|)], [47|40|{], [48|41|A], [49|42|;], [50|43|r], [51|44|(], [52|45|E], [53|46|)], [54|47|;], [55|48|}], [56|493|C], [57|494|B], [58|5917|h], [59|5918|=], [60|5919|g], [61|5920|(], [62|5921|)], [63|5929|;], [64|5930|C], [65|5931|B], [66|71161|i], [67|71162|=], [68|71163|n], [69|71164|(], [70|71165|)], [71|71173|;], [72|71174|C], [73|71175|B], [74|854089|j], [75|854090|=], [76|854091|E], [77|10249093|1], [78|854101|.], [79|541|y]

****

[0|0|m], [1|1||], [2|2|r], [3|3|=], [4|4||], [5|7|h], [6|8|g], [7|17||], [8|18|r], [9|35|=], [10|36|.], [11|71|z], [12|72|x], [13|37|6], [14|9|=], [15|10||], [16|19|i], [17|20|n], [18|41||], [19|42|r], [20|83|=], [21|84|.], [22|167|j], [23|168|6], [24|85|4], [25|21|=], [26|22|.], [27|43|j], [28|44|1], [29|5|y]

****

Notacao ponolesa: [=], [i], [j], [6], [=], [.], [4], [r], [n], [=], [j], [1], [=], [.], [r], [6], [.], [r], [4], [=], [j], [1], [.], [r], [y], [=], [h], [g], [=], [z], [x], [.], [r], [6], [=], [i], [n], [=], [j], [6], [.], [r], [4], [=], [j], [1], [.]

Notacao ponolesa reversa: [h], [z], [x], [=], [.], [6], [r], [g], [=], [i], [j], [6], [=], [.], [4], [r], [n], [=], [j], [1], [=], [.], [y], [r], [h], [z], [x], [=], [.], [6], [r], [g], [=], [i], [j], [6], [=], [.], [4], [r], [n], [=], [j], [1], [=], [.], [r]

****
****

Referências

1. CFG Developer. <https://web.stanford.edu/class/archive/cs/cs103/cs103.1156/tools/cfg/> Acessado em: 27/10/2022.
2. A. V. Aho, R. Sethi, J. D. Ullman: Compiladores: Princípios, técnicas e ferramentas. LTC - Livros Técnicos e Científicos Editora, 1995.
3. Link para download: https://github.com/MatBrands/Compiladores/tree/master/Proj4/