

# C-minus com operações matriciais

Leonardo Maffei da Silva\*

2019

## Resumo

Neste documento encontra-se a especificação da gramática de uma linguagem que estende uma versão mínima do C (conhecida por *C-minus*). A nova linguagem proposta apresenta como primitivo o tipo matriz, bem como a realização das operações aritméticas básicas e outras operações mais complexas tais como potenciação e escalonamento, sendo a última realizada por função da linguagem. Grande parte da gramática foi diretamente tirada de (ANÔNIMO, ).

**Palavras-chave:** C, linguagem, matriz, primitiva.

## 1 Introdução

Implementar-se-á, até a versão final deste artigo, um compilador para a linguagem proposta. Para sua realização, serão utilizados os conhecimentos adquiridos na disciplina *Tradutores*, ministrada pela professora Cláudia Nalon.

## 2 Usuário característico

Destina-se ao estudante de álgebra linear, o qual pode usar a linguagem para, por exemplo, confirmar se sua resolução de um sistema linear encontra-se correta, tudo isso de maneira rápida, eficiente e *offline*.

## 3 Motivação

Durante a realização do curso de Cálculo Numérico, o grupo do autor notou a ausência dessa *feature* na linguagem C. Desse modo, foi necessária a simulação desse tipo de dados, à época implementada por meio de inúmeras funções. Se houvesse um tipo nativo para matriz bem como operações elementares sobre seus elementos, teria sido de grande auxílio à codificação dos diversos métodos numéricos requeridos pela disciplina.

---

\*leoitu22hotmail.com@gmail.com. <<https://www.linkedin.com/in/leonardo-maffei-ti/>>

## 4 Gramática

A seguir, encontra-se a gramática da linguagem proposta:

$\langle \text{program} \rangle ::= \langle \text{declaration-list} \rangle$

$\langle \text{declaration-list} \rangle ::= \langle \text{declaration-list} \rangle \langle \text{declaration} \rangle \mid \langle \text{declaration} \rangle \mid \langle \text{comment} \rangle$

$\langle \text{comment} \rangle :: // \langle \text{ASCII} \rangle^* \text{EOL}$

$\langle \text{declaration} \rangle ::= \langle \text{var-declaration} \rangle \mid \langle \text{fun-declaration} \rangle$

$\langle \text{var-declaration} \rangle ::= \langle \text{base-type-specifier} \rangle \text{ID} ; \mid \langle \text{base-type-specifier} \rangle \text{ID} [\langle \text{expression} \rangle];$   
 $\mid \langle \text{mat-specifier} \rangle \text{ID} [\langle \text{expression} \rangle][\langle \text{expression} \rangle] ;$

$\langle \text{type-specifier} \rangle ::= \langle \text{base-type-specifier} \rangle \mid \langle \text{mat-specifier} \rangle$

$\langle \text{base-type-specifier} \rangle ::= \text{int} \mid \text{float}$

$\langle \text{mat-specifier} \rangle ::= \text{mat} < \langle \text{base-type-specifier} \rangle >$

$\langle \text{fun-declaration} \rangle ::= \langle \text{type-specifier} \rangle \langle \text{ID} \rangle ( \langle \text{params} \rangle ) \langle \text{compound-stmt} \rangle \mid \langle \text{mat-specifier} \rangle$   
 $\langle \text{ID} \rangle ( \langle \text{params} \rangle ) \langle \text{compound-stmt} \rangle$

$\langle \text{params} \rangle ::= \langle \text{param-list} \rangle \mid \epsilon$

$\langle \text{param-list} \rangle ::= \langle \text{param-list} \rangle , \langle \text{param} \rangle \mid \langle \text{param} \rangle$

$\langle \text{param} \rangle ::= \langle \text{type-specifier} \rangle \langle \text{ID} \rangle \mid \langle \text{type-specifier} \rangle \langle \text{ID} \rangle [ ] \mid \text{mat} < \langle \text{type-specifier-base} \rangle >$   
 $\langle \text{ID} \rangle$

$\langle \text{compound-stmt} \rangle ::= \langle \text{local-declarations} \rangle \langle \text{statement-list} \rangle$

$\langle \text{local-declarations} \rangle ::= \langle \text{local-declarations} \rangle \langle \text{var-declaration} \rangle \mid \epsilon$

$\langle \text{statement-list} \rangle ::= \langle \text{statement-list} \rangle \langle \text{statement} \rangle \mid \epsilon$

$\langle \text{statement} \rangle ::= \langle \text{expression-stmt} \rangle \mid \langle \text{compound-stmt} \rangle \mid \langle \text{selection-stmt} \rangle$   
 $\mid \langle \text{iteration-stmt} \rangle \mid \langle \text{return-stmt} \rangle$

$\langle \text{expression-stmt} \rangle ::= \langle \text{expression} \rangle ; \mid ;$

$\langle \text{selection-stmt} \rangle ::= \text{if} ( \langle \text{expression} \rangle ) \langle \text{statement} \rangle \mid \text{if} ( \langle \text{expression} \rangle ) \langle \text{statement} \rangle \text{else}$   
 $\langle \text{statement} \rangle$

$\langle \text{iteration-stmt} \rangle ::= \text{while} ( \langle \text{expression} \rangle ) \langle \text{statement} \rangle$

$\langle \text{return-stmt} \rangle ::= \text{return} ; \mid \text{return} \langle \text{expression} \rangle ;$

$\langle \text{expression} \rangle ::= \langle \text{var} \rangle = \langle \text{expression} \rangle \mid \langle \text{simple-expression} \rangle \mid \langle \text{int-seq} \rangle \mid \langle \text{float-seq} \rangle \mid \langle \text{int-nested-seq} \rangle$   
 $\mid \langle \text{float-nested-seq} \rangle$

$\langle \text{int-seq} \rangle ::= \langle \text{simple-expression} \rangle \mid \langle \text{int-seq} \rangle , \langle \text{simple-expression} \rangle$

$\langle \text{float-seq} \rangle ::= \langle \text{simple-expression} \rangle \mid \langle \text{float-seq} \rangle, \langle \text{expression} \rangle$

$\langle \text{int-nested-seq} \rangle ::= \langle \text{int-seq} \rangle \mid \langle \text{int-nested-seq} \rangle, \langle \text{int-seq} \rangle \mid \epsilon$

$\langle \text{float-nested-seq} \rangle ::= \langle \text{float-seq} \rangle \mid \langle \text{float-nested-seq} \rangle, \langle \text{float-seq} \rangle \mid \epsilon$

$\langle \text{var} \rangle ::= \langle \text{ID} \rangle \mid \langle \text{ID} \rangle [ \langle \text{expression} \rangle ]$

$\langle \text{simple-expression} \rangle ::= \langle \text{additive-expression} \rangle \langle \text{relop} \rangle \langle \text{additive-expression} \rangle \mid \langle \text{additive-expression} \rangle$

$\langle \text{relop} \rangle ::= <= \mid < \mid > \mid >= \mid == \mid !=$

$\langle \text{additive-expression} \rangle ::= \langle \text{additive-expression} \rangle \langle \text{addop} \rangle \langle \text{term} \rangle \mid \langle \text{term} \rangle$

$\langle \text{addop} \rangle ::= + \mid -$

$\langle \text{term} \rangle ::= \langle \text{term} \rangle \langle \text{mulop} \rangle \langle \text{bin} \rangle \mid \langle \text{bin} \rangle$

$\langle \text{mulop} \rangle ::= * \mid /$

$\langle \text{bin} \rangle ::= \langle \text{bin} \rangle \langle \text{bin-logi} \rangle \langle \text{unary} \rangle \mid \langle \text{unary} \rangle$

$\langle \text{bin-logi} \rangle ::= \&\&$

$\begin{array}{c} \mid \quad \parallel \\ \mid \quad ^ \end{array}$

$\langle \text{unary} \rangle ::= \langle \text{unary-op} \rangle \langle \text{factor} \rangle \mid \langle \text{factor} \rangle$

$\langle \text{unary-op} \rangle ::= ! \mid \&$

$\langle \text{factor} \rangle ::= ( \langle \text{expression} \rangle ) \mid \langle \text{var} \rangle \mid \langle \text{call} \rangle \mid \langle \text{NUM} \rangle$

$\langle \text{call} \rangle ::= \langle \text{ID} \rangle ( \langle \text{args} \rangle )$

$\langle \text{args} \rangle ::= \langle \text{arg-list} \rangle \parallel \langle \text{empty} \rangle$

$\langle \text{arg-list} \rangle ::= \langle \text{arg-list} \rangle, \langle \text{expression} \rangle \mid \langle \text{expression} \rangle$

$\langle \text{letter\_} \rangle ::= \text{a} \mid \dots \mid \text{z} \mid \text{A} \mid \dots \mid \text{Z} \mid \_$

$\langle \text{digit} \rangle ::= 0 \mid \dots \mid 9$

$\langle \text{ID} \rangle ::= \langle \text{letter\_} \rangle ( \langle \text{letter\_} \rangle \mid \langle \text{digit} \rangle )^*$

$\langle \text{NUM} \rangle ::= \langle \text{INT} \rangle \mid \langle \text{FLOAT} \rangle$

$\langle \text{INT} \rangle ::= \langle \text{digit} \rangle^+$

$\langle \text{FLOAT} \rangle ::= \langle \text{digit} \rangle^+. \langle \text{digit} \rangle^*$

Os *tokens* da linguagem são: `'', '.', '>=', '>', '<', '<=', '==', '!=', 'int', 'float', 'while', 'return', 'if', 'else', 'mat', 'EOL', '+', '-', '*', '/', '&&', '^', '||', '!', '&', ';', '(', ')', '[, ]', ',', ' ' e '//',` sendo este último um caso especial por delimitar comentários de uma linha. *EOL* simboliza o final de uma linha qualquer, e as seguintes palavras são reservadas e portanto não podem ser utilizadas como identificadores: **else**, **float**, **if**, **int**, **mat** e **return**. A regra de produção *ASCII* produz qualquer símbolo contido na tabela *ascii*. Por fim, não constam na gramática, mas as funções *scan* e *print* fazem parte da linguagem, sendo responsáveis pelas operações de leitura e impressão na saída padrão do ambiente de execução do programa (usualmente, o console).

## 5 Semântica

A semântica da linguagem é quase idêntica à da linguagem C: declarações de variável (a menos das do tipo **mat**), funções e expressões têm semântica similar. Sendo esta linguagem uma extensão de um subconjunto da linguagem C, a principal diferença está no tipo de dados **mat** (abreviação de *matrix*, "matriz" em inglês). Esse tipo de dados é similar aos *arrays* em C, porém limitado do ponto de vista da composição pois não é possível a criação de matrizes aninhadas. Entretanto, é possível a realização das quatro operações aritméticas básicas diretamente com matrizes, bem como a realização de potenciação de matrizes de forma *rápida* e algumas operações sobre elas, como resolução de sistemas lineares e escalonamento. Tais operações mais complexas estarão presentes por meio de funções padrão da linguagem.

## 6 Exemplo de programa na linguagem

A seguir, trechos de código pertencente à nova linguagem.

```
1 int foo(void);
2 mat<int> matriz[3][4]; // Criacao de um sistema
3 matrix = {
4     {1, 2, 3, 4},
5     {4, 3, 2, 1},
6     {-1, -2, -3, -4},
7     {45, 87, 9, 18}
8 };
9
10 int main(){
11     int x;
12     mat<float> casted = matriz + matriz;
13     scan(x);
14     print(x);
15 }
```

## 7 Agradecimentos

Agradecimentos ao autor anônimo responsável por ([ANÔNIMO](#), ), sem o qual este trabalho teria sido muito mais custoso, bem como ao usuário [CroCro](#), o qual possibilitou conforme ([CROCRO](#), ) rápida customização do código-exemplo da nova linguagem. Por fim, o pacote utilizado para redação da gramática foi sugerido pelo usuário [AlanMunn](#), conforme ([MUNN](#), ).

## Referências

ANÔNIMO. *C-syntax*: Bnf grammar for c-minus. Disponível em: <<http://www.csci-snc.com/ExamplesX/C-Syntax.pdf>>. Citado 2 vezes nas páginas 1 e 4.

CROCRO. Disponível em: <<https://tex.stackexchange.com/questions/348651/c-code-to-add-in-the-document>>. Citado na página 4.

MUNN, A. Disponível em: <<https://tex.stackexchange.com/questions/24886/which-package-can-be-used-to-write-bnf-grammars/39751>>. Citado na página 4.