UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
COLEGIADO DE CIÊNCIA DA COMPUTAÇÃO

# ESPECIFICAÇÃO DA LINGUAGEM C MAIS OU MENOS E ANALISADOR LÉXICO

PEDRO XAVIER, VALQUÍRIA BELUSSO

CASCAVEL - 2022

# SUMÁRIO

**Especificação da Linguagem: C Mais ou Menos**

# 1. Tipos de dados suportados

- Inteiros:
    - Inteiro com sinal (complemento de dois)
        - i8, i16, i32, i64.
    - Inteiro sem sinal
        - ui8, ui16, ui32, ui64.
- Ponto Flutuante:
    - f32, f64.
- Booleanos:
    - bool: byte único, false se zero, true caso contrário.

# 2. Operadores suportados:

## 2.1 Atribuição

É realizada com o símbolo =, podendo ser feita das seguintes maneiras:

- <tipo> <id> = <valor>, <id> = <valor>;
- <tipo> <id> = <valor>, <id>;
- <tipo> <id>, <id> = <valor>;
- <tipo> <id>, <id>;

## 2.2 Atribuição aritmética

É possível atualizar o valor de uma variável usando a seguinte expressão:

- <variável> <operador> <valor>

| Atribuição | Operador |
|---|---|
| soma | += |
| subtração | -= |
| multiplicação | *= |
| divisão | /= |
| módulo | %= |

## 2.3 Operadores aritméticos

| Operação | Operador | Sintaxe |
|---|---|---|
| soma | + | i + j |
| subtração | - | i - j |
| multiplicação | * | i * j |
| divisão | / | i / j |

| | | |
|---|---|---|
| módulo | % | i % j |

### 2.4 Operadores Lógicos

As operações lógicas resultam em um valor booleano. Para operandos não booleanos, zero significa falso, caso contrário, verdadeiro.

| Operação | Operador | Sintaxe |
|---|---|---|
| E | && | i && j |
| Ou | \| \| | i \| \| j |
| Não | ! | ! i |

### 2.5 Operadores Relacionais

Expressões de comparação resultam em um valor booleano.

| Operação | Operador | Sintaxe |
|---|---|---|
| Igual | == | i == j |
| Diferente | != | i != j |
| Maior ou igual | >= | i >= j |
| Maior | > | i > j |
| Menor ou igual | <= | i <= j |
| Menor | < | i < j |

## 3. Incremento e Decremento

| Operação | Operador | Sintaxe |
|---|---|---|
| Incremento | ++ | i ++ |
| Decremento | -- | i -- |

## 4. Literais

Há quatro formas de literais utilizados:

- String Literal: Qualquer texto entre "" é um literal de string. Não pode ser atribuído a uma variável, mas pode ser usado com a instrução write. Exemplo: "Olá!".
- Integer Literal: Qualquer sequência de números, uso de outras bases. Exemplo: 42, -13, 0xdeadbeef, 0o777, 0b1010 ...

- Float Literal: Qualquer sequência de números começando com um ponto ou com um ponto na sequência. Exemplo: 4.2, .5, -2.0. -.5 …
- Booleano Literal: true, false.

## 5. Comentários

Existe uma forma de escrever algo no programa de forma que este ignore o que foi escrito, essa forma é através dos comentários utilizando o símbolo #, sendo o compatível com expressões shebang em arquivos de script.

## 6. Estruturas suportadas pela linguagem

### 6.1 Estruturas de decisão

A linguagem utiliza as seguintes estruturas de decisão:

- if: verifica uma declaração (entre parênteses) e se for verdadeira, executa o seguinte bloco de código (entre chaves);
- else: deve estar após uma instrução if e suas instruções/bloco de código, executa um bloco de código a seguir se a condição na instrução if for falsa;

Exemplo:

```
# condicionais
i32 num = 25;

if ( num > 0 ) {
  write "number is positive", "\n";
} else {
  write "number is not positive", "\n";
}
```

### 6.2 Estruturas de repetição

Duas estruturas de repetição são utilizadas na linguagem, sendo eles for e while. Exemplo:

```
# loops
i32 i = 0;

while ( i < 2 ) {
  write "Inside the while", "\n";
  i += 1;
}

for (i32 j = 0; j < 3; j += 1) {
  write "Inside the for", "\n";
}
```

# 7. Identificadores e palavras-chave

## 7.1 Identificadores

As variáveis devem ser identificadas com nomes exclusivos, esses nomes exclusivos são chamados de identificadores. Um identificador deve começar com uma letra ou sublinhado e pode seguir com qualquer sequência de caracteres alfanuméricos (mais sublinhado).

Exemplos: i32 minutes = 60;
i32 _something = -1;
i32 MyVariableIs = 0;
i32 _123 = 123;

## 7.2 Declarações de ES

- read: lê de stdin e define o valor na variável. Sintaxe: read <variável>;
- write: escreve em stdout, pode escrever variáveis e literais. Sintaxe write <variáveis>;

# 8. Palavras reservadas

São palavras reservadas:

- tipos: i8, ui8, i16, ui16, i32, ui32, i64, ui64, f32, f64, bool.
- booleanos literais: true e false.
- estruturas de fluxo: if, else, for, while.
- comandos de ES: read e write.

# 9. Exemplo de código

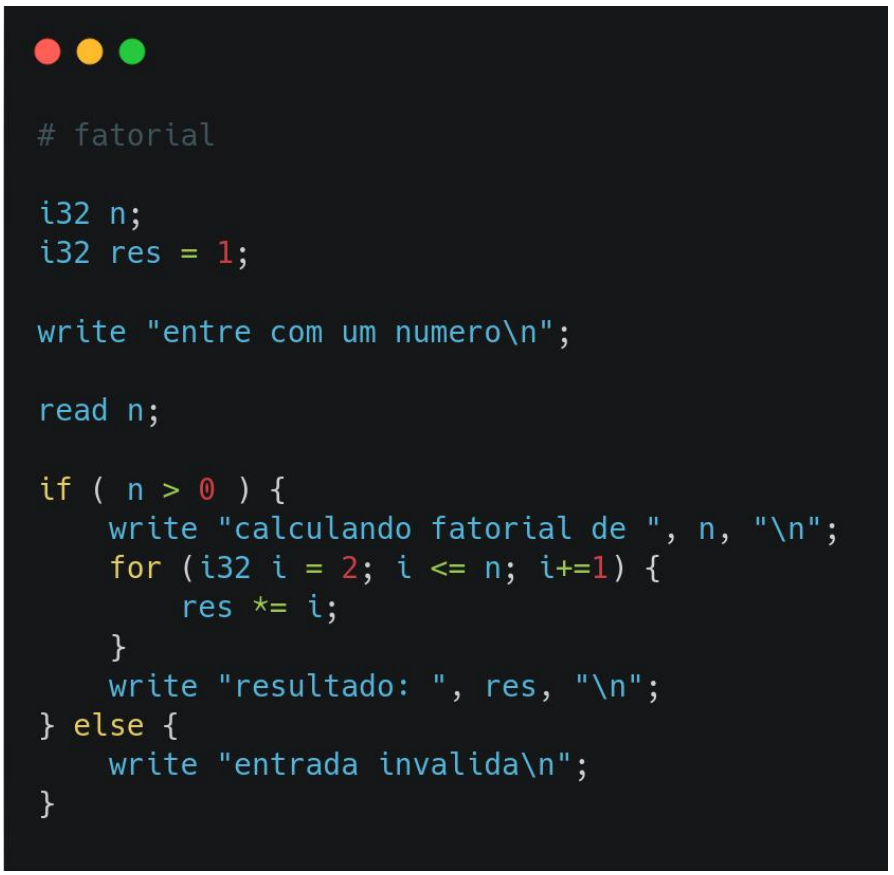
```
# fatorial

i32 n = 5;
i32 i = 1;

write "calculando fatorial de ", n, "\n";

while ( n > 1 ) {
    i *= n;
    n -= 1;
}

write "resultado: ", i, "\n";
```



```
# fatorial

i32 n;
i32 res = 1;

write "entre com um numero\n";

read n;

if ( n > 0 ) {
    write "calculando fatorial de ", n, "\n";
    for (i32 i = 2; i <= n; i+=1) {
        res *= i;
    }
    write "resultado: ", res, "\n";
} else {
    write "entrada invalida\n";
}
```

## 10. BNF

**<program>::=** <stmts>
**<stmts>::=**
    <stmt> <stmts>
  | ε
**<stmt>::=**
    <decl_stmt> ;
  | <attrib_stmt> ;
  | <command>
  | <flow>
**<decl_stmt>::=**
    <type> <id> <decl_end>
**<decl_end>::=**
    '=' <expr>
  | ε
**<attrib_stmt>::=**
    <id> <attrib_end>
**<attrib_end>::=**
    '=' <expr>
  | <attrib_op> <expr>
**<expr>::=**
    <bool_expr>
**<bool_expr>::=**

‘!’ <expr>
| <rel_expr> <bool_expr_c>
**<bool_expr_c>::=**
<bool_op> <rel_expr>
| ε
**<rel_expr>::=**
<arith_expr> <rel_expr_c>
**<rel_expr_c>::=**
<rel_op> <arith_expr>
| ε
**<arith_expr>::=**
<term> <arith_expr_c>
**<arith_expr_c>::=**
<arith_op_sum> <term>
| ε
**<term>::=**
<factor> <term_c>
**<term_c>::=**
<arith_op_mul> <factor>
| ε
**<factor>::=**
‘(‘ <expr> ‘)’
| <value>
**<flow>::=**
<if_stmt>
| <while_stmt>
| <for_stmt>
**<if_stmt>::=**
if ( <bool_expr> ) { <stmts> } <else_stmt>
**<else_stmt>::=**
else { <stmts> }
| ε
**<while_stmt>::=**
while ( <bool_expr> ) { <stmts> }
**<for_stmt>::=**
for ( <decl_stmt> ; <bool_expr> ; <attrib_stmt> ) { <stmts> }
**<command>::=**
read <id>
| write <value> <wchain>
**<wchain>::=**
‘,’ <value> <wchain>
| ε
**<value>::=**
<id>
| <literal>
**<literal>::=**
<literal_int>
| <literal_float>
| <literal_bool>
| <literal_str>

# 11. Aspectos semânticos

Há a declaração prévia das variáveis e estas são armazenadas na pilha de escopo;
O escopo de identificadores deverá ser global;
Cada identificador declarado em um escopo é único;
Se a variável for declarada e nunca utilizada, é considerado erro;
Não é realizado operações em variáveis de diferentes tipos;