



Universidade Federal de Alagoas - UFAL

Instituto de computação - IC

Disciplina: Compiladores

Prof: Drº Alcino Dall Igna Junior

# Especificação da Linguagem ENL

## Compiladores 2019.1

Thiago Emmanuel G. Rodrigues

Erivaldo Lourenço Mariano

Maceió, 10 julho de 2019

<b>Introdução</b>	<b>3</b>
<b>Estrutura geral do programa</b>	<b>3</b>
Tipos de dados	<b>3</b>
Inteiro	3
Ponto flutuante	4
Cadeia de caractere	5
Booleano	5
Arranjo unidimensional	6
<b>Comentários</b>	<b>7</b>
<b>Funções</b>	<b>7</b>
<b>Inicialização</b>	<b>7</b>
Operadores	<b>8</b>
Operadores aritméticos	8
Operadores de comparação	9
Operadores lógico	9
<b>Coerção</b>	<b>10</b>
<b>Concatenação</b>	<b>10</b>
Atribuição	<b>10</b>
Instruções	<b>10</b>
<b>Impressão</b>	<b>11</b>
<b>Leitura</b>	<b>11</b>
<b>Programas:</b>	<b>12</b>
Alo mundo:	12
Fibonacci	12
Shell Sort	13

# Introdução

A linguagem ENL foi baseada na linguagem C, a linguagem é de tipagem estática, case sensitive, desenvolvida com a ideia de ser prática, fácil e rápida.

## Estrutura geral do programa

Na linguagem ENL a função *begin()* será o ponto de entrada do programa. Se a existir um programa na linguagem ENL ela tem que ter a função void *begin()*. Ex de um programa em ENL:

```
void begin()  
{  
}
```

Todo comando em ENL tem por obrigação terminar com ‘;’.

As variáveis podem ser declaradas dentro da função *begin*, de outras funções, dentro das estruturas de seleção e de interação.

A linguagem ENL só admite escopo local.

## Tipos de dados

A Linguagem ENL é estaticamente tipada e é necessário a declaração explícita do tipo das variáveis em sua declaração. As palavras que definem seu tipo são reservadas.

### Inteiro

Palavra **int** define um tipo inteiro. A declaração de uma variável do tipo inteiro é feita da seguinte forma:

```
int variavel;
```

Operações suportadas:

Tipo	Operadores aritméticos	Operadores de comparação	Operadores lógicos	Concatenação
int	Soma Subtração Multiplicação Divisão Módulo unário negativo	Igual Diferente Menor igual Maior igual Menor que Maior que		Se concatenados a um caractere ou cadeia de caracteres (formam um cchar)

### Ponto flutuante

Palavra **float** define um tipo ponto flutuante. A declaração de uma variável de ponto flutuante é da seguinte forma:

**float** variavel;

Tipo	Operadores aritméticos	Operadores de comparação	Operadores lógicos	Concatenação
float	Soma Subtração Multiplicação Divisão Módulo unário negativo	Igual Diferente Menor igual Maior igual Menor que Maior que		Se concatenados a um caractere ou cadeia de caracteres (Formam um cchar)

### Caractere

A palavra **char** define um tipo caractere que está representado no padrão ASCII puro. A declaração é da seguinte forma:

**char** caractere;

Tipo	Operadores aritméticos	Operadores de comparação	Operadores lógicos	Concatenação
char		Igual Diferente Menor igual Maior igual Menor que Maior que		Podem ser concatenados aos tipos int, float, char, cchar, bool(Formam um cchar)

### Cadeia de caractere

A palavra **cchar** define uma cadeia de caracteres. Uma cadeia de caractere é uma sequência de caracteres. A declaração é da seguinte forma:

**cchar** cadeia[tamanho];

Todos os elementos do cchar são do tipo char. As operações que podem ser feitas com variáveis do tipo cchar são:

Tipo	Operadores aritméticos	Operadores de comparação a outro cchar	Operadores lógicos	Concatenação
cchar		Igual Diferente Menor igual Maior igual Menor que Maior que		Podem ser concatenados aos tipos int, float, char, cchar, bool(Formam um cchar)

### Booleano

A palavra **boolean** define uma variável que tem como valores *true* ou *false*.

Sendo a declaração da seguinte forma:

**bool** variavel;

Tipo	Operadores aritméticos	Operadores de comparação	Operadores lógicos	Concatenação
bool		Igual Diferente	AND OR NOT	Se concatenados a um caractere ou cadeia de caracteres

### Arranjo unidimensional

**vector** - Define um arranjo de uma dimensão. Esse arranjo é uma coleção de objetos que contém obrigatoriamente, os tipos primitivos, e são declarados da seguinte forma:

O tamanho e acesso ao vetor é feito usando exclusivamente inteiros.

**vector** tipo variavel[tamanho\_int];

**vector int** exemplo[10];

Tipo	Operadores aritméticos	Operadores de comparação	Operadores lógicos	Concatenação
vector	Os elementos do vetores vão suportar as operações suportadas pelo tipo do vetor.	Os elementos do vetores vão suportar as operações suportadas pelo tipo do veto	Os elementos do vetores vão suportar as operações suportadas pelo tipo do veto	Os elementos do vetores vão suportar as operações suportadas pelo tipo do veto

## Comentários

Pode se fazer comentários usando o operador **#** no início e final do comentário.

## Funções

As funções são definidas antes da função `begin()`, com a palavra reservada *function*, e após o tipo do retorno, o identificador da variável.

Dentro de uma função devem ser indicados o tipo e o identificador do parâmetro e podem ser `int`, `float`, `char`, `cchar`, `boolean`, `vector` além que a função pode receber outra função como parâmetro.

Exemplo de declaração de função:

```
function tipo_de_retorno identificador( tipo
parametro1, tipo parametro2, ... , tipo parametroN ){

    return valor_ou_alguma_variável;
}
```

## Inicialização

As variáveis podem ser inicializadas no momento da criação por um valor ou expressão. Caso não sejam inicializadas, seus valores default são os que seguem:

Tipo	Valor default
int	0
float	0.0
bool	false
array	inicializado com o valor padrão de seu tipo definido
char, string	null

## Operadores

Dividimos os operadores em vários tipos, verificados na tabela abaixo:

Operadores aritméticos	Símbolo
Soma Subtração Multiplicação Divisão Módulo unário negativo	+ - * / % ~

A associatividade para operadores aritméticos será da direita para a esquerda. A ordem de precedência segue abaixo:

- Unário negativo, Multiplicação, Divisão, Módulo.
- Adição e Subtração.

Precedência e associatividade dos operadores aritméticos na ordem abaixo:

- Unário negativo                      Associatividade da direita para a esquerda
- Multiplicação, Divisão, Módulo      Associatividade da esquerda para a direita
- Adição e subtração                      Associatividade da esquerda para a direita



Operadores de comparação	Símbolo
Igual Diferente Menor igual Maior igual Menor que Maior que	== != <= >= < >

A associatividade para operadores de comparação será da esquerda para direita. A ordem de precedência segue abaixo:

- < , > , <= , >=
- == !=

Operadores lógico	Símbolo
AND OR NOT	&&    !

A associatividade para operadores lógicos será da esquerda para direita. A ordem de precedência segue abaixo:

- !
- && ||

# Concatenação

A linguagem ENL suporta concatenação de caracteres através do operador **++**.

# Atribuição

A atribuição é feita usando o operador '=', como mostra na tabela de operadores. Funciona de modo tal que o valor da esquerda do operador = , recebe o valor da direita.

EX:

```
int a = b+1;
```

# Instruções

Estrutura condicional de uma via:

```
if ( expressão lógica ){  
    instruções  
}
```

Estrutura condicional de duas vias:

```
if ( expressão lógica ){  
    instruções  
}  
else{  
}
```

Estrutura iterativa com controle lógico:

```
while ( expressão lógica ){  
    instruções  
}
```

Estrutura iterativa controlada por contador:

Se o ritmo for omitido, o incremento do contador será igual a um.

```
for ( inicio; fim; ritmo) {  
    instruções;  
}
```

Sendo a variável de controle um tipo int  
ex:

```
for(int i = 0; 10; 1 ){  
    instruções;
```

A palavra reservada 'break' pode ser usada para sair da estrutura iterativa.

## Impressão

A função put recebe argumentos e imprime algo na tela.

%d para inteiros

%f para float

%c para char

%s para cchar

%b para bool

para imprimir os elementos do vector é necessário usar uma estrutura de iteração ou de controle lógico especificando o tipo de seu elemento

put("tipos a imprimir separados por vírgula", variáveis ou expressões a imprimir separados por vírgula)

ex:

```
put("%d", 10);
```

A instrução acima imprime 10 na tela;

## Leitura

A função read recebe argumentos e lê algo digitado pelo usuário.

%d para inteiros

%f para float

%c para char

%s para cchar

%b para bool

para ler os elementos do vector é necessário usar uma estrutura de iteração ou de controle lógico especificando o tipo de seus elementos  
read(“tipos a serem lidos separados por vírgula”, variáveis que vão armazenar os valores lidos separados por vírgula)

ex:

```
void begin()
{
    int i;
    read(“%d”, i);
}
```

o programa lê um inteiro digitado pelo usuário e armazena na variável i.

## Programas:

Ola mundo:

```
void begin()
{
    put(‘%s’, ‘Ola mundo’);
}
```

### Fibonacci

```
function void fibonacci(int num)
{
    int i = 0;
    int j = 1;
    int k = 1;
    int t = 0;

    while(k < num){
        t = i + j;
        i = j;
        j = t;
        put(‘%d, ’, j);
        k = k + 1;
    }
}

void begin()
{
    int i;
```

```

    read('%d', i);
    fibonacci(i);
}

```

## Shell Sort

```

function vector shellSort(int vet, int size) {
    int value;
    int j;

    int gap = 1;
    while(gap < size) {
        gap = 3*gap+1;
    }
    while (gap > 0) {
        for(int i = gap;size; i++) {
            value = vet[i];
            j = i;
            while (j > gap-1 && value <= vet[j - gap]) {
                vet[j] = vet [j - gap];
                j = j - gap;
            }
            vet [j] = value;
        }
        gap = gap/3;
    }

    return vet;
}

```

```

void begin()
{

    vector<int> vetor[10], tam = 10;
    vector<int> newvetor[10];
    for(int i = 0; i < tam; i = i + 1){
        read('%d', vetor[i]);
    }

    newvetor = shellsort(vetor, tam);
}

```

```
    for(int i = 0; tam; 1){  
        put("%d", newvetor[i]);  
    }  
  
}
```