Documentação Linguagem L−−

Lucas Montenegro e Letícia Medeiros

20/03/2021

Universidade Federal de Alagoas - Instituto de Computação
Compiladores
Professor Alcino Dall lgna Junior

Conteúdo

1 Especificações da Linguagem **3**

1.1 Estrutura Geral do Programa . . . 3
1.2 Estrutura da nomeação de variáveis e funções . . . 3
1.3 Símbolos especiais . . . 3
1.4 Declaração de variáveis e Especificação de tipos . . . 3
1.4.1 Inteiro . . . 4
1.4.2 Ponto flutuante . . . 4
1.4.3 Char . . . 4
1.4.4 Booleano . . . 4
1.4.5 String . . . 4
1.4.6 Arranjos unidimensionais . . . 4
1.5 Constantes Literais . . . 5
1.6 Operações . . . 5
1.6.1 Operadores Aritméticos . . . 5
1.6.2 Operadores Relacionais . . . 6
1.6.3 Operadores Lógicos . . . 7
1.7 Ordem de precedência e associatividade . . . 7
1.8 Instruções . . . 8
1.8.1 Atribuição . . . 8
1.8.2 Estrutura Condicional . . . 8
1.8.3 Loop . . . 8
1.8.4 Entrada . . . 9
1.8.5 Saída . . . 9
1.8.6 Comentários . . . 9
1.9 Funções . . . 10
1.10 Códigos de exemplo . . . 10
1.10.1 Hello World . . . 10
1.10.2 Shell sort . . . 10
1.10.3 Série de Fibbonaci . . . 10

1 Especificações da Linguagem

1.1 Estrutura Geral do Programa

Cada programa da linguagem irá conter funções e instruções. As funções podem ser declaradas em qualquer lugar do código, entretanto não podem ser declaradas dentro de outras funções. Além disso, o escopo de cada função é delimitado por chaves '{' '}'.

As variáveis e instruções só podem ser declaradas dentro de funções. Cada instrução deve ser finalizada com o ';'.

O ponto inicial do programa é a função main que é obrigatória e possui algumas particularidades como:

- A função main deve ser do tipo int
- Não pode ter parâmetros
- Só pode existir uma função main
- Ela deve sempre retornar 1

1.2 Estrutura da nomeação de variáveis e funções

Os nomes das funções/variáveis devem ser iniciados por letras e depois podem conter letras, números e o símbolo especial '_'. Vale ressaltar que a linguagem é case sensitive.

Ex: int minha_variavel, minhaVariavel1

1.3 Símbolos especiais

Os símbolos especiais aceitos na linguagem são (no formato flex):

- ' ' | '\;' | '\,' | '.' | ':' | '?' | '\!' | '\+' | '\-' |'*' | '\\' | '\/' | '_' | '\%' | '@' | '\&' | '\#' | '\$' | '\<' |'\>' | '\=' | '\(' | '\)' | '\[' | '\]' | '\{' | '\}' | '\|' | '\' ' | '\"' | '\^';

1.4 Declaração de variáveis e Especificação de tipos

Na declaração de uma variável é possível definir mais de uma na mesma instrução utilizando o símbolo especial ',' desde que seja do mesmpo tipo. Ademais, não é possível definir uma variável e atribuí-la na mesma instrução. Caso uma variável não seja atribuída, ela possui o valor 'null' por default.

A especificação de tipos é estática e possui o seguinte conjunto de tipos disponíveis:

1.4.1 Inteiro

O tipo inteiro é responsável pelos números inteiros e é definido pela palavra reservada 'int'. Possui a seguinte declaração e atribuição:

int nome_variavel;
nome_variavel = 1;

1.4.2 Ponto flutuante

O tipo ponto flutuante é responsável pelos números decimais e é definido pela palavra reservada 'float'. Possui a seguinte declaração e atribuição:

float nome_variavel;
nome_variavel = 1.0;

A variável deverá receber um número decimal com o símbolo especial '.' .

1.4.3 Char

O tipo caractere é responsável pelos caracteres e é definido pela palavra reservada 'char'. Possui a seguinte declaração e atribuição:

char nome_variavel;
nome_variavel = 'l';

A variável deverá receber um caractere entre aspas simples.

1.4.4 Booleano

O tipo booleano é responsável pelos valores booleanos true/false e é definido pela palavra reservada 'bool'. Possui a seguinte declaração e atribuição:

bool nome_variavel, nome_variavel2;
nome_variavel = true;
nome_variavel2 = false;

1.4.5 String

O tipo string é responsável por uma cadeia de caracteres e é definido pela palavra reservada 'string'. O seu tamanho varia de acordo com a última cadeia de caracteres atribuída. Possui a seguinte declaração e atribuição:

string nome_variavel;
nome_variavel = "Frase";

A variável deverá receber uma cadeia de caracteres entre aspas duplas.

1.4.6 Arranjos unidimensionais

O tipo arranjo unidimensional é responsável por um array de um tipo (o array só pode ser do tipo int ou float) e é definido pela palavra reservada 'array'. O seu tamanho deve ser definido durante a declaração da variável. É atribuído um valor

ou acessado através do índice entre os '[' ']'. O índice de cada array começa com 0 e último índice acessível é tamanho_do_array - 1.

Declaração:

array tipo nome_variavel [tamanho];

Exemplo de declaração, atribuição e acesso na linguagem:

```
array int lista[3];
lista[0] = 1;
lista[1] = 2;
lista[2] = 3;
write("#i", lista[0]);
```

1.5 Constantes Literais

Segue abaixo as expressões regulares, utilizando o padrão flex, para cada constante literal:

Constante Inteira: {Número}+
Constante Float: ({Número})+\.({Número})+
Constante Boolean: ('true' | 'false')
Constante Char: (' \' ') ({Letra} | {Símbolo} | {Número}) (' \ ' ')
Constante String: = \" (({Letra} | {Símbolo} | {Número})∗) \"

Ver na especificação dos tokens da linguagem para obter informações de letra, símbolo e número.

1.6 Operações

As operações não realizam coerção. Segue a tabela com todas as operações suportadas pela linguagem:

Operador	Tipos que realizam a operação
' ∧ '	int
' ∗ ' ' / ' ' - ' ' + ' ' % '	int, float
' < ' ' <= ' ' > ' ' >= '	int, float, char
'==' '!=' '='	int, float, bool, char, string
'&&' '||' '!'	bool

1.6.1 Operadores Aritméticos

Realiza operações aritméticas entre variáveis ou constantes.

Adição

Realiza a adição entre dois números e retorna o seu resultado.
Exemplo: var1 + var2

Subtração

Realiza a subtração entre dois números e retorna o seu resultado.
Exemplo: var1 - var2
Pode ser também um operador unário, funcionando da seguinte forma:
Exemplo: - var1

Multiplicação

Realiza a multiplicação entre dois números e retorna o seu resultado.
Exemplo: var1 $*$ var2

Exponenciação

Realiza o cálculo da exponenciação entre dois números e retorna o seu resultado.
Exemplo: base $\wedge$ expoente

Módulo

Realiza o cálculo do resto entre dois números e retorna o seu resultado.
Exemplo: var1 % var2

1.6.2 Operadores Relacionais

Realiza uma operação relacional entre duas expressões. Uma expressão pode ser definida como uma variável ou um conjunto de variáveis do mesmo tipo com operações aritméticas.

Por fim, é realizado uma operação normal de relação entre expressões numéricas (int e float) e booleanos. Já com caracteres e strings, é feita uma análise lexicográfica verificando a posição de cada caractere na tabela ASCII.

Menor que

Verifica se a primeira expressão é menor que a segunda e retorna o resultado em booleano.
Exemplo: expressao1 < expressao2

Maior que

Verifica se a primeira expressão é maior que a segunda e retorna o resultado em booleano.
Exemplo: expressao1 > expressao2

Menor ou igual

Verifica se a primeira expressão é menor ou igual a segunda e retorna o resultado em booleano.
Exemplo: expressao1 <= expressao2

Maior ou igual

Verifica se a primeira expressão é maior ou igual a segunda e retorna o resultado em booleano.
Exemplo: expressao1 >= expressao2

Igual

Verifica se a primeira expressão igual a segunda e retorna o resultado em booleano.

Exemplo: expressao1 == expressao2

Diferente

Verifica se a primeira expressão é diferente da segunda e retorna o resultado em booleano.

Exemplo: expressao1 != expressao2

1.6.3 Operadores Lógicos

Realiza uma operação relacional entre duas expressões. Uma expressão pode ser definida como uma variável ou um conjunto de variáveis do mesmo tipo com operações aritméticas ou relacionais.

Além disso, qualquer coisa diferente de '0', 'null' e 'false' é considerado como 'true'.

AND

Realiza uma operação **AND** entre a primeira expressão e a segunda expressão, retornando o resultado em booleano.

Exemplo: expressao1 && expressao2

OR

Realiza uma operação **OR** entre a primeira expressão e a segunda expressão, retornando o resultado em booleano.

Exemplo: expressao1 || expressao2

NOT

Nega a expressão, retornando o resultado em booleano.

Exemplo: !expressao1

1.7 Ordem de precedência e associatividade

A ordem de precedência vai de cima para baixo e tem a seguinte tabela:

Ordem de precedência	Associatividade
' ! ' (Negação)	Direita para esquerda
' - ' (Operador unário negativo)	Direita para esquerda
' ∧ ' (Exponenciação)	Direita para esquerda
' ∗ ' , ' / ' e ' % ' (Operadores aritméticos)	Esquerda para direita
' + ' e ' - ' (Operadores aritméticos)	Esquerda para direita
' < ' , ' <= ' , ' > ' e ' >= ' (Operadores relacionais)	Não Associativo
'==' , '!=' e '=' (Operadores relacionais)	Não Associativo
' && ' (Conjunção)	Esquerda para direita
' || ' (Disjunção)	Esquerda para direita

1.8 Instruções

1.8.1 Atribuição

A instrução de atribuição faz com que a variável da esquerda receba o valor da expressão que está na direita. Ambos devem possuir o mesmo tipo.

Exemplo: var1 = expressao1;

1.8.2 Estrutura Condicional

Cada estrutura condicional deverá possuir obrigatoriamente no início um **if** que é a condição inicial. Há também a estrutura **elsif** que deverá estar entre if e else, podendo haver mais de um elsif entre eles. Por fim, a ultima estrutura é o **else** que abrange todas as condições não abordadas pelas estruturas anteriores.

Todas as estruturas condicionais (if, elsif e else) devem possuir chaves para a abertura e fechamento do seu escopo.

Exemplo de uma estrutura condicional:

```
if (expressaoLogica1) {
    Bloco de código
} elsif (expressaoLogica2) {
    Bloco de código
} else {
    Bloco de código
}
```

1.8.3 Loop

Existem duas estruturas de loops, o primeiro é o while que irá iterar enquanto a sua expressão lógica for 'true'.

Exemplo do loop com while:

```
while (expressaoLogica1) {
    Bloco de código
}
```

A segunda estrutura de loop é o for. Essa é uma estrutura que utiliza um contador, sendo dividido em 3 partes: a primeira é a inicialização do contador, a segunda é a condição do for e a terceira é o passo. Além disso, cada parte é separada por um ';'.

Na primeira parte a linguagem aceita apenas variáveis inteiras. Na segunda parte deve ocorrer uma expressão que retorne um valor booleano e na terceira parte deverá apenas ter o valor inteiro do passo, caso seja negativo deverá utilizar o operador unário '-'. Por fim, o loop for encerra quando a condição for 'false'

Exemplo do loop com for:

```
for (i = 10; i <= 10; 1) {
    Bloco de código
```

}

Todas as estruturas de loop (while e for) devem possuir chaves para a abertura e fechamento do seu escopo.

1.8.4 Entrada

A instrução de entrada poderá receber um ou mais parâmetros e não é necessário especificar o seus tipos, apenas realizar uma separação entre cada variável pelo símbolo especial ',' .

Exemplo da instrução de entrada:

```
read (a, b, c);
```

1.8.5 Saída

A instrução de saída deverá estar entre aspas duplas e poderá receber um ou mais parâmetros. Ao inserir variáveis é necessário inserir '#' + o código responsável pela formatação de cada tipo, além disso cada variável deve ser separada por ',' , inclusive após o fechamento das aspas duplas.

Exemplo da instrução de saída:

```
a = "Lucas";
b = 'L';
write ("Ola #s tudo bem? Minha letra e #c", a, b);
```

Segue abaixo a tabela com os códigos para a utilização de variáveis na instrução de saída:

Formatação	
#i	Para inteiros
#f	Para números flutuantes com 1 casa decimal
#f.d	Para números flutuantes com d casas
#c	Para caracteres
#s	Para string
#b	Para boolean

Para utilizar algum símbolo especial que não é possível normalmente, deverá utilizar o caractere de escape \. Por exemplo: write("Esse e o \\ (backslash) e esse e o \# (jogo da velha)");

1.8.6 Comentários

O comentário inicia com $ e vai até o final da linha.

Exemplo de um comentário:

```
a = "Lucas";  $variavel a recebe lucas
```

1.9 Funções

As funções podem ser declaradas em qualquer lugar do código (menos dentro de outras funções) e seu escopo é delimitado pelas chaves '{' '}' que são obrigatórias.

Cada função deve ser iniciada pela palavra reservada 'function', seguido pelo seu tipo por parâmetros. Ela pode ter ou não parâmetros e caso tenha ela deve especificar entre '(' ')' cada parâmetro e seu tipo, além de separá-los por ',' .

Estrutura da declaração da função:

```
function tipo nome_funcao (tipo param1, tipo param2) {
    Bloco de código
    return x;
}
```

Exemplo de função:

```
function char retornar_letra (char letra) {
    return letra;
}
```

A função pode ter os seguintes tipos: int, bool, float, char e void. Ela deverá utilizar um 'return' (palavra reservada) que é obrigatório e deverá obedecer o tipo da função.

Para usar uma função, basta chamá-la em qualquer função e colocando os nomes das variáveis (ou os valores) dentro dos parâmetros (se existirem).

Exemplo:

```
function int main () {
    retornar_letra('u');
    return 1;
}
```

Por fim, para especificar um array em um parâmetro deverá ser utilizada a mesma formatação da declaração do array, com a palavra reservada 'array', o seu tipo, o nome da variável e o tamanho entre colchetes.

1.10 Códigos de exemplo

1.10.1 Hello World

```
function int main () {
    write ("Hello World!");
    return 1;
}
```

1.10.2 Shell sort

1.10.3 Série de Fibbonaci