Universidade Federal de Alagoas
Instituto de Computação
Curso de Ciência da Computação

Jvlal
Especificação da Linguagem

João Victor Ribeiro Ferro
Lucas Albuquerque Lisboa

Maceió
2021

Introdução

A linguagem de programação Jvlal foi criada com inspiração nas linguagens C e Python, tendo como meta de utilização o ensino instrutivo da programação.

Jvlal não é orientada a objeto e é case-sensitive (diferencia letras maiúsculas e minúsculas). Já na legibilidade, Jvlal não faz coerção (não admite conversões implícitas de tipo) e possui palavras reservadas, às quais estão em inglês e foram escolhidas para que fique o mais claro possível para mostrar o que o código está fazendo.

Estrutura geral do programa

Ponto de início de execução

O ponto inicial de execução do programa será identificado por uma construção específica da linguagem, sempre será a função main() com o tipo de retorno obrigatório int.

Ex.:

```
fun int main(){
    .
    .
    .
    return 0;
}
```

Definições de procedimentos / funções

Estas poderão existir se forem declaradas fora do corpo de uma outra função ou procedimento. Estas podem ser acessadas se tiverem sua implementação ou assinatura declaradas antes da função chamada. Jvlal não aceita passagem de subprogramas como parâmetro.

A declaração de de função é sempre iniciado pela palavra reservada fun, seguido pelo pelo seu tipo de retorno obrigatório **(int, float, bool, string, char)**, seu identificador único e uma lista de parâmetros (os parâmetros acompanharão de seu tipo e seu identificador) delimitada por abre e fecha parênteses **( )**. A abertura e fechamento do bloco é feita por abre e fecha chaves **{ }.**

A declaração de um procedimento seguirá o mesmo padrão, com exceção de que terá o tipo de retorno e será sempre iniciada, não terá o tipo de retorno e será sempre iniciada com a palavra reservada proc.

A passagem de parâmetro para os procedimentos e funções se dá por meio do modo de entrada e saída, para todos os tipos de estrutura.

Ex.:

```
fun tipoDoRetorno id ( listaDeParametros){
    …
    return x;
}

proc id (listaDeParametros) {
    …
}
```

A declaração de assinatura seguirá o padrão de função ou procedimento, com exceção de que finaliza com um ponto e vírgula ao invés de abre e fecha chaves.

Ex.:

```
fun tipoDoRetorno id (listaDeParametros);
proc id (listaDeParametros);
```

Definições de instruções

As instruções podem ser declaradas somente dentro do bloco de funções ou procedimentos, não podem ser escritas instruções do corpo dos parâmetros de uma função, tal como estruturas de iteração e estruturas de seleção.

Conjunto de tipo de dados e nomes

Identificador

Possuirão sensibilidade à caixa, limite de tamanho de 31 caracteres e seu formato será definido a partir da seguinte expressão regular:

(‘letter’ | ‘_`) (‘digit’ | ‘letter’ | ‘_’)*

Exemplos de nomes aceitos pela linguagem: _var1, var2, var ,_, _a;
Exemplos de nomes não aceitos pela linguagem: .abc, ab@c, 1ac.

Palavras reservadas

As palavras reservadas são sempre escritas em inglês, buscando objetividade e simplicidade são listadas a seguir:

const, char, int, float, string, bool, proc, read, print, if, ceif (check else if), else, false, true, for, while, return, lenght.

Comentário

Os comentários são indicados por “//”. Desta forma, o que se escrever na linha após será descartado. Os comentários podem ser feitos apenas por linha.

Definição de variáveis

Jvlal possui escopo global, logo as variáveis podem ser definidas fora do escopo e acessadas globalmente ou dentro do escopo como variáveis locais. São declaradas iniciando com o **tipoDaVariável (int, float, boll, string, char)** e seguidas pelo seu **identificador único.** Caso haja múltiplas variáveis em uma declaração de tipo único, as variáveis serão separadas por **vírgula**. A declaração termina com um **ponto e vírgula.**

Ex.:

```
int b = 2;
float c = 3.4, d;
char e;
bool g,h = true;
string i = “hello”;
```

Definição de constantes

Constantes possuem o mesmo padrão de declaração e definição de variáveis, com exceção que iniciam sempre com a palavra reservada const e que a atribuição de valor é obrigatório na declaração .

Ex.:

```
const int i = 0, j = 1;
const bool debug = true;
```

Tipos de Dados

Todos os tipos e estruturas possuirão compatibilidade por nome.

Booleano

É a identificação da variável como sendo do tipo booleana, identificando pela palavra reservada bool. Possui dois valores possíveis: **true,false**

Ex.: bool b;

Inteiro

É a identificação da variável como sendo do tipo inteiro de 32 bits, identificado pela palavra reservada **int.** Seus literais são expressos como uma sequência de números decimais.

Ex.: int i;

Ponto Flutuante

É a identificação da variável como sendo do tipo ponto flutuante de 64 bits, identificado pela palavra reservada **float**. Seus literais são expressos como uma sequência de números decimais, seguido de um ponto e demais dígitos.

Ex.: float f;

Caractere

É a identificação da variável como sendo do tipo caractere 8 bits, identificado pela palavra reservada char. Guarda um código referente ao seu símbolo da tabela ASCII. A constante literal do caractere é delimitada por **apóstrofo**.

Ex.: char c;

Cadeia de Caracteres

É a identificação da variável como sendo do tipo cadeia de caracteres, identificado pela palavra reservada **string.** Seus literais são expressos como um conjunto, mínimo de 0 caracteres, mínimos de 0 caracteres e de tamanho máximo ilimitado. A constante literal é delimitada por aspas.

Ex: string s;

Arranjos Unidimensionais

Os arranjos unidimensionais são compostos pelos tipos determinados acima. Sua declaração iniciará como o tipo seguido do identificador único e, delimitado por abre e fecha parênteses **( )**, o tamanho do arranjo.

Há uma função que dirá o tamanho do arranjo, a função retorna um inteiro.

Ex.: int s = length(arr)

Ex.:

```
int arr(9);
float arr(1024);
bool arr(2);
```

Operações suportadas

Operador	Tipos que realizam a operação
‘!’	**bool**
‘^’ ‘*’ ‘/’ ‘+’ ‘-’	**int, float**
‘<’ ‘<=’ ‘>’ ‘>=’	**int, float, char**
‘==’ ‘!=’ ‘=’	**int, float, bool, char string**
‘&&’ ‘| |’	**bool**
‘::’	**string,char**

Coerção

Jvlal é estaticamente tipada, não aceitando coerção entre variáveis de tipo diferentes. Dessa forma, todas as verificações de compatibilidade de tipo serão feitas estaticamente.

Valores padrão

Tipo	Valor de inicialização
int	0
float	0.0
char	‘ ‘(caracter vazio)
string	“ ”(string vazia)
bool	false

Conjunto de Operadores

Aritméticos

- **Adição**: + ;
- **Subtração**: - ;
- **Multiplicação**: * ;
- **Divisão**: / ;
- **Unitário Positivo**: + ;

- **Unitário Negativo**: - ;

Relacionais

- **Igualdade**: == ;
- **Desigualdade**: != ;
- **Maior que:** > ;
- **Maior igual**: >= ;
- **Menor que**: < ;
- **Menor igual:** <= ;

Lógicos

- **Negação unária**: ! ;
- **Conjunção**: &&;
- **Disjunção**: '||' ;

Concatenação

- O operador de concatenação ::, pode ser aplicado a string e caracteres, resultando sempre em uma string;

Atribuição

- O operador de atribuição '=';

Precedência e Associatividade

Operadores	Associatividade
'!'(Not)	**Direita para esquerda**
'-'(unário negativo) ; '+' (unário positivo)	**Direita para esquerda**
'*' (multiplicação); '/'(divisão) ; '%'(resto)	**Esquerda para direita**
'+'(adição) ; '-' (subtração)	**Esquerda para direita**
'<' ;'>'; '<='; '>='	**Não associativo**
'==' ; '!='	**Não associativo**
'&&'	**Esquerda para direita**

‘||’	**Esquerda para direita**
‘::’ (concatenação)	**Esquerda para direita**
‘=’ (atribuição)	**Direita para esquerda**

Instruções

Uma linha de instrução acaba com o “;” no final. O bloco de instruções tem um escopo que inicia-se com abre chaves e termina com fecha chaves.

Atribuição

É definido pelo símbolo ‘=’, sendo o lado esquerdo o identificador único e o lado direito o valor ou expressão. Todos os operandos devem possuir o mesmo tipo de variável alvo.

Estrutura condicional

Na estrutura condicional do “IF” está relacionada a uma expressão lógica ou a variável booleana dentro de parênteses e como é um bloco de instruções seu escopo será definido por chaves.

O algoritmo irá executar o código contido no “IF”, se somente se, a sua condição lógica for verdadeira, sem caso negativo poderá entrar no “ceif” caso também não aceite a expressão lógica será entrará no “else”.

Ex.:

```
if (expressão Lógica) {
        .
        .
        .
} ceif (expressaoLogica) {
        .
        .
        .
} else {
        .
        .
        .
}
```

Estrutura iterativa com controle lógico

O loop ocorrerá quando a condição for verdadeira, finalizando quando não satisfazer mais a condicional (expressão lógica) e como é um bloco de instrução definido por chaves.

Ex.:

```
while ( expressaoLogica) {
        .
        .
        .
}
```

Estrutura iterativa controlada por contador

No loop os três valores (a,b,c) representam expressões aritméticas. O valor de a será o valor inicial do contador, b será o valor final e c será o valor de incremento.

Ex.:

```
for (int i : (a,b,c)) {
        .
        .
        .
}
```

Constantes literais e suas expressões

As expressões regulares das constantes literais são denotadas da seguintes maneira:

1. Constante de inteiro: ((‘digit’)+)
2. Constante de float: ((‘digit’)+) (‘\.’) ((‘digit’)+)
3. Constante de char: (‘\ ‘’) (‘letter’ | ‘symbol’ | ‘digit’) (‘\’’)
4. Constante de bool (‘true’ | ‘false’)
5. Constantes de String (‘\”’) ((‘letter’ | ‘symbol’ | ‘digit’)*) (‘|” ’)

Desvios Incondicionais

Jvlal não aceita nenhum tipo de desvio incondicional.

Entrada e Saída

- Entrada é realizada por meio da função read()
    - A função read (identificador) atribuíra o valor lido da variável do identificador. Como o identificador já foi declarado, sabemos qual será o seu tipo.
- Saída: É realizada por meio da função print()
    - A função print (identificador): irá imprimir o valor na tela. O identificador nesta função também pode ser visto como uma string não instanciado, ou seja, apenas com abertura e fechamento de aspas pode-se escrever algo (entre as aspas) e isto será impresso na tela, não necessariamente tendo que ser uma variável.
    - A saída poderá sera formatada, os identificadores dos tipos de formatação são: %d inteiro, %c caractere, %f float, %s string, %b bool. Caso a saide encontra formatada, no código deverá ser especificado quais são as variáveis específicas a cada um dos identificadores de formatação;
      Ex .:
      print (“%s%d%c%b%f”, str, integer, ch, boolean, flo);

Programas Exemplos

- Hello World:

```
fun int main ( ) {
        print(“Hello world”);
        return 0;
}
```

- Série de Fibonacci

```
proc fib (int n) {
        int n1=0, n2 = 1, n3;
        if (n == 0){
                print(“%d\n”, n);
        } ceif (n ==1) {
                print (“0,%d\n”,n);
        } else{
                string separator = “”;
                print (“0,1,”);
                while (true){
                        print(“%s%d”,separator,n3);
                        if (n3>= n){
```

```
                print("\n");
                return ;
            }
            separator = " ,";
            n1 = n2;
            n2 = n3;
        }
    }
}
```

- Shell Sort

```
int n = 1000;

proc shellSort(int arr(n)){
    int gap = n/2;
    while(int i : (gap,n,1)) {
        for (int i : (gap,n,1)){
            int temp = arr[i];
            int j = i;

            while (j>= gap && (arr[j-gap] > temp)){
                arr[j] = arr[j-gap];
                j = j -gap;
            }
            arr[j] = temp;
        }
        gap = gap/2;
    }
}

fun int main ( ) {
    int arr(n);
    read (arr);

    for (int i : (0,n,1)) {
        print ("%d" , arr[i]);
    }

    shellSorte(arr);
    for (int i : (0,n,1)) {
        print ("%d", arr[i]);
    }
```

```
        print("\n");
        return 0;
}
```