Manual – Linguagem DB

- **Informações importantes**

  Arquivos de linguagem DB devem ter extensão “.DB” e são compilados/executados com o comando “java DB filename.DB” ou “java DB filelocal/filename.DB”.

  A linguagem DB é case sensitive, portanto, todos os comandos devem ser escritos no código fonte EXATAMENTE da forma como são descritos nesse manual.

  O separador de ponto flutuante na atribuição e na impressão é o caracter '.' (ponto) e na leitura o caracter ',' (vírgula).

  Cada linha de um código fonte escrito em DB, deve conter um e somente um, comando. Ou seja, códigos da forma:

  t = 1; i = 2; r = t + i;

  não são permitidos, devendo ser escritos na forma:

  t = 1;

  i = 2;

  r = t + i;

  É importante ressaltar, que apenas os comandos descritos nesse manual são permitidos, qualquer outro comando, pode ocasionar bugs na linguagem ou resultar em erro de sintaxe.

- **Tipos de dados**

  Os tipos de dados suportados pela linguagem DB são: inteiro, ponto flutuante e string.

- **Terminador de comandos**

  Todos os comandos, exceto os controladores de fluxo e os laços, devem possuir um terminador, sendo este o símbolo '$' (cifrão).

- **Declaração de variáveis**

  Todo comando de declaração de váriavel, DEVE ser precedido pelo símbolo '@' (arroba). Os tipos de dados são representados pelos caracteres 'i' (inteiro), 'f' (ponto flutuante) e 's' (string) e devem estar declarados logo após o '@', separados por um ou mais espaços. Um comando pode declarar mais de uma variável, sendo necessário separar cada uma pelo caracter ',' (vírgula). O nome de uma variável deve conter apenas números e letras e não deve começar com um número. É possível atribuir um valor a uma variável na sua declaração, através do caracter '=' (igual) e das aspas em caso de string. Toda declaração de

variável deve possuir um terminador.

Exemplos:

@ i num1, num2, np1 = 2$

@ f media$

@ s palavra = “teste”$

- **Atribuição de valor à variável**

  O comando de atribuição a variável é feito da seguinte forma: VARIAVEL = VALOR, onde valor, pode representar um valor numérico, uma string, uma variável ou uma operação entre números/variáveis. Quando uma variável inteira recebe um valor de ponto flutuante é feito arredondamento. Todo comando de atribuição deve possuir um terminador.

  Exemplos:

  num1 = 5$

  palavra = “teste2”$

  media = num1/2$

- **Operações**

  As operações devem ser realizadas com apenas dois operandos. Estão disponíveis as operações de adição ('+'), subtração ('-'), multiplicação ('*'), divisão ('/') e módulo ('%'). A operação de módulo está disponível apenas para valores inteiros.

  Exemplos:

  media = num1 + num2$

  media = media/2$

  media = media * 0.4$

  c = a%b$

- **Controlador de fluxo**

  A linguagem possui como controlador de fluxo o comando “if” que é dado da seguinte forma:

  if (expressão) {

  comandos$

  ...

}

de maneira que a sintaxe deve ser exatamente esta, excetuando quantidade de espaços/tabulações. A expressão deve tratar apenas de variáveis, valores numéricos ou comparações entre variáveis, valores numéricos ou strings, de forma que operações não são permitidas. Os operadores booleanos disponíveis são:

- “==” ─ Igual a
- “<” ─ Menor que
- “>” ─ Maior que
- “<==” ─ No máximo
- “>==” ─ No mínimo
- “#” ─ Diferente de

Strings permitem apenas comparação pelo operador “==”. Não é permitido o uso de um controlador de fluxo dentro de outro, já o uso de laços é liberado. A abertura e o fechamento do escopo do if são dados pelos caracteres '{' (abre chaves) e '}' (fecha chaves).

Exemplos Válidos:

```
if (num1>num2) {
    num1 = num2$
}
if (5<==7) {
    i = 2$
}
if (3#media) {
    np1 = 0$
}
if (1) {
    np2 = 1$
}
```

Exemplos Inválidos:

```
if (num1+ 1>2) {
    num = 2$
}
if (num==2)
```

```
{ num = 3$ }
if (num2) {
    if (np1>==0) {
        np1 = num2$
    }
}
```

- **Laço**

  O comando “repeatif” repete o código em seu escopo até que a expressão seja verdadeira e é dado da seguinte forma:

```
repeatif(expressão) [
    comandos$
    ...
]
```

  de maneira que a sintaxe deve ser exatamente esta, excetuando quantidade de espaços/tabulações. A expressão segue o modelo descrito no controlador de fluxo. Não é permitido o uso de um laço dentro de outro, já o uso de controladores de fluxo é liberado. A abertura e o fechamento do escopo do if são dados pelos caracteres '[' (abre conchetes) e ']' (fecha conchetes).

  Exemplos Válidos:

```
repeatif (num1<num2) [
    num1 = num1 + 1$
]
repeatif (k<==media) [
    k = k + 2$
}
```

  Exemplos Inválidos:

```
repeatif (num<==2)
[ num = num + 3$ ]
repeatif (num2>0) {
    repeatif (np1>==0) {
        np1 = np1 - 1$
    }
```

```
    num2 = np1 - 2$
}
```

- **Impressão na tela**

  Todo comando de impressão, DEVE ser precedido pelo caracter '!' (exclamação). Existem três tipos de impressão, impressão de variável, denotada pela letra 'v', impressão de texto, denotada pela letra 't' e impressão de quebra de linha denotada pela letra 'l'. Na impressão de variável e de texto, é passado entre parenteses a variável/texto a ser impresso e na impressão de linha apenas o l é escrito. Não é permitida impressão dos caracteres “(“ e “)” pelo comando de impressão de texto. Na impressão de valores de ponto flutuante, todas as casas decimais do número são impressas. Todo comando de impressão deve conter um terminador.

  Exemplo:
  Seja a variável num1 igual a 1, temos que o código:

```
    ! t (Variavel num1 = )$
    ! v (num1)$
    ! l$
```

  imprime na tela a mensagem:
  “Variavel num1 = 1(quebradelinha)”

- **Leitura**

  O comando “get” faz a leitura de um valor da entrada padrão e o atribui para a variável, da seguinte forma:

```
    get(variavel)$
```

  sendo a variável de qualquer um dos tipos. Apenas um valor pode ser lido por cada comando. Todo comando de leitura deve conter um terminador.