Universidade Federal de Itajubá – UNIFEI
Curso de Engenharia de Computação
ECO 002 – Técnicas de Programação – Prof. Edmilson Marmo Moreira

Capítulo 2 – Fundamentos dos Algoritmos Estruturados

> *"Se a gente quisesse ser apenas feliz, isso não seria difícil. Mas a gente quer ficar mais feliz do que os outros, e isso é quase sempre difícil porque nós sempre achamos os outros mais felizes do que nós."*
>
> **Montesquieu**

2.1 Considerações Iniciais

As normas utilizadas neste curso não são únicas e seguem a estrutura definida na referência (FARRER, H. et. alli., 1999).

2.2 Constantes

Uma constante é um determinado valor fixo que **não** se modifica ao longo do tempo, durante a execução de um programa.

Uma constante pode ser um número, um valor lógico ou uma seqüência de caracteres.

Conforme o seu tipo, a constante é classificada como sendo **numérica**, **lógica** ou **literal**.

Constante Numérica

A representação de uma constante numérica nos algoritmos é feita no sistema decimal, podendo ser um número com ou sem parte fracionária.

Exemplos:

- `15;`
- `23,5;`
- `-10.`

Constante Lógica

É um valor lógico, isto é, que só pode ser **falso** ou **verdadeiro**, usado em proposições lógicas.

Só existem duas constantes deste tipo, sendo representadas pelas palavras `falso` e `verdadeiro`.

Constante Literal

Uma constante deste tipo pode ser qualquer seqüência de caracteres (letras, dígitos ou símbolos especiais) que forme um literal com algum significado para o problema em estudo.

As constantes literais devem aparecer no algoritmo entre aspas para que não sejam confundidas com outro item qualquer.

Exemplos:

- "Introdução a Computação"
- "12345"
- "XYZ"
- "08/03/2005"
- "FALSO"

2.3 Variáveis

Na matemática, uma variável é a representação simbólica dos elementos de um certo conjunto.

Nos algoritmos, uma variável corresponde a uma posição de memória, cujo conteúdo pode variar ao longo do tempo durante a execução de um programa.

Embora uma variável possa assumir diferentes valores, ela só pode armazenar um valor a cada instante.

Toda variável é identificada por um nome ou **identificador**. Assim, por exemplo, num algoritmo para o cálculo das raízes de uma equação do 2º grau ($ax^2 + bx + c = 0$), os identificadores `A`, `B` e `C` podem representar as posições de memória que armazenam os coeficientes da equação, fazendo, neste caso, o papel das variáveis na matemática.

Formação dos Identificadores

Um identificador é formado por um ou mais caracteres, sendo que o primeiro caractere deve, obrigatoriamente, ser uma letra e os caracteres seguintes, letras ou dígitos, não sendo permitido o uso de símbolos especiais.

Exemplos:

- `A`
- `Aluno`
- `XYZ123`
- `CADE`
- `OI235`

É recomendável que os nomes das variáveis sejam os mais significativos possíveis, isto é, que reflitam, da melhor maneira, a natureza dos valores que nelas estão sendo armazenados. Isto ajuda muito no entendimento do algoritmo.

A título de exemplo: se a variável vai armazenar o nome de um empregado, por que não escolher o identificador `NomeDoEmpregado` para representá-la, ao invés de `NE` ?

Declaração de variáveis

As variáveis só podem armazenar valores de um mesmo tipo, de maneira que também são classificadas como sendo **numéricas**, **lógicas** ou **literais**.

Para indicar o tipo de uma ou mais variáveis é usada a declaração de variáveis.

Uma vez declarada a variável, qualquer referência que se faça ao seu identificador implica a referência ao conteúdo do local da memória representado pelo mesmo.

Formato:

`declare lista-de-identificadores nome-do-tipo`

Onde:

`declare`	É uma palavra-chave do algoritmo
`lista-de-identificadores`	São os nomes escolhidos para as variáveis, que devem estar separados por vírgulas
`nome-do-tipo`	É uma das três palavras-chaves, numérico, lógico ou literal que indicam o tipo associado às variáveis.

Exemplos:

- `declare NomeDoFuncionario, CPF literal;`
- `declare Salario, HoraExtra numérico;`
- `declare PossuiDependentes lógico;`

2.4 Expressões Aritméticas

Denomina-se **expressão aritmética** aquela cujos operadores são aritméticos e cujos operandos são constantes e/ou variáveis do tipo numérico.

O conjunto de operações básicas adotado é o que se conhece da Matemática, a saber:

- Adição
- Subtração
- Multiplicação
- Divisão
- Potenciação
- Radiciação

Exemplos:

- `X + Y`
- `X – 10`
- `2 × Nota`
- `TOTAL / N`
- $SOMA^2$

A notação utilizada para expressões aritméticas nos algoritmos é, basicamente, a mesma da Matemática, a menos das seguintes restrições:

1. Não é permitido omitir o operador de multiplicação, o que é comum nas expressões matemáticas. Isto evita confusão quanto aos nomes de variáveis, pois numa expressão da forma AB+C, como saber se AB é o nome de uma variável ou a multiplicação entre os conteúdos de duas variáveis, cujos termos são A e B? Pelo mesmo motivo, o operador da multiplicação deve sempre ser escrito com letra minúscula (x);

2. Nas expressões aritméticas, as operações guardam entre si uma relação de prioridade, tal como na matemática:

Prioridade	Operação
1ª.	Potenciação, radiciação
2ª.	Multiplicação, divisão
3ª.	Adição, subtração

Para se obter uma seqüência de cálculo diferente, vários níveis de parênteses podem ser usados para quebrar as prioridades definidas.

Não é permitido o uso de colchetes e chaves, uma vez que estes símbolos são utilizados nos algoritmos para outras finalidades.

Funções

Além das operações básicas, algumas funções muito comuns na Matemática podem ser usadas nas expressões aritméticas.

A tabela a seguir apresenta algumas das principais funções e o resultado fornecido por cada uma delas.

Nome	Operação
`LOG (x)`	Logaritmo na base 10 de x
`LN (x)`	Logaritmo neperiano de x
`EXP (x)`	O número *e* (base dos algoritmos neperianos) elevado a x
`ABS (x)`	Valor absoluto de x
`TRUNCA (x)`	A parte inteira de um número fracionário
`ARREDONDA (x)`	Transforma, por arredondamento, um número fracionário em inteiro
`SINAL (x)`	Fornece o valor –1, +1 ou zero conforme o valor de x seja negativo, positivo ou igual a zero, respectivamente.
`QUOCIENTE (x,y)`	Quociente inteiro da divisão de x por y.
`RESTO (x,y)`	Resto inteiro da divisão de x por y

2.5 Expressões Lógicas

É comum nos algoritmos surgirem situações em que a execução de uma ação está sujeita a uma certa condição.

Esta condição é representada no texto do algoritmo por meio de uma expressão lógica.

Denomina-se **expressão lógica** a expressão cujos operadores são lógicos e cujos operandos são relações, constantes e/ou variáveis do tipo lógico.

Relações

Uma expressão relacional, ou simplesmente relação, é uma comparação realizada entre dois valores do mesmo tipo básico.

Estes valores são representados na relação através de constantes, variáveis ou expressões aritméticas (para o caso de valores numéricos).

Os operadores relacionais, que indicam a comparação a ser realizada entre os termos da relação, são conhecidos da Matemática, a saber:

- = igual a
- ≠ diferente de
- > maior que
- < menor que
- ≥ maior ou igual a
- ≤ menor ou igual a

O resultado obtido de uma relação é sempre um **valor lógico**.

Operadores Lógicos

A Álgebra das Proposições define três conectivos usados na formação de novas proposições a partir de outras já conhecidas. Estes conectivos são os operadores nas expressões lógicas, a saber:

- **e** - para conjunção
- **ou** - para disjunção
- **não** - para negação

Neste contexto considera-se uma proposição como sendo uma variável lógica, uma relação ou uma expressão lógica composta.

Conjunção, $p \wedge q$

Duas proposições quaisquer podem ser combinadas pela palavra "e" para formar uma proposição composta, chamada de *conjunção* das proposições originais.

A conjunção de duas proposições p e q é representada simbolicamente por

$$p \wedge q$$

p e q são chamados **fatores** da expressão.

Exemplo:

- Seja p "Está chovendo" e seja q "O Sol está brilhando". Assim, $p \wedge q$ corresponde à proposição "Está chovendo e o Sol está brilhando".

O verdadeiro valor da proposição $p \wedge q$ satisfaz a seguinte propriedade:

V_1: Se p é verdadeiro e se q é verdadeiro, então $p \wedge q$ será verdadeiro; caso contrário $p \wedge q$ será falso.

Um meio conveniente de estabelecer **V_1** consiste em organizar a seguinte **tabela-verdade**:

p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F

Disjunção, $p \vee q$

Duas proposições quaisquer podem ser combinadas pela palavra "ou" para formar uma proposição composta, chamada de *disjunção* das proposições originais.

A disjunção de duas proposições p e q é representada simbolicamente por

$$p \vee q$$

Exemplo:

- Seja p "Pedro estudou inglês na universidade" e seja q "Pedro morou nos Estados Unidos". Assim, $p \vee q$ corresponde à proposição "Pedro estudou inglês na universidade ou morou nos Estados Unidos".

O valor verdade da proposição $p \vee q$ satisfaz a seguinte propriedade:

V_2: Se p é verdadeiro ou se q é verdadeiro, ou ambos são verdadeiros, então $p \vee q$ será verdadeiro; caso contrário $p \vee q$ será falso.

V_2 também pode ser escrito na forma de uma tabela-verdade:

P	q	$p \vee q$
V	V	V
V	F	V
F	V	V
F	F	F

Negação, $\sim p$

Dada uma proposição p qualquer, uma outra proposição, chamada *negação* de p, pode ser formada escrevendo-se "É falso que …" antes de p ou, se possível, inserindo em p a palavra "não". Simbolicamente, a negação de p é designada por

$$\sim p$$

Exemplo:

- Seja p "Paris está na França" então $\sim p$ pode ser escrita como: "É falso que Paris esteja na França" ou "Paris não está na França".

O valor verdade da proposição $\sim p$ satisfaz a seguinte propriedade:

V_5: Se p é verdadeiro, então $\sim p$ é falso, caso contrário $\sim p$ é verdadeiro.

V_5 também pode ser escrito na forma de uma tabela-verdade:

p	$\sim p$
V	F
F	V

Se p for a sentença “Pedro é alto e magro”, então a sentença $\sim p$ será “É falso que Pedro seja alto e magro”, que pode ser reformulada como “Pedro não é alto ou não é magro”. Isso não é a mesma proposição que “Pedro é baixo e gordo”.

Prioridade

Como é possível ter mais de um operador lógico na mesma expressão uma ordem é preciso ser imposta.

A relação de prioridade guarda a seguinte ordem:

Prioridade	Operador
1ª.	Aritmético
2ª.	Relacional
3ª.	Não
4ª.	E
5ª.	Ou

2.6 Expressões Literais

Uma **expressão literal** é aquela formada por operadores literais e operandos que são constantes e/ou variáveis do tipo literal.

Supondo que `A` e `B` sejam duas variáveis literais e que o símbolo “|” é um operador de concatenação de literais, a expressão `A | B` fornece como resultado um único literal formado pelo conteúdo de A seguido do conteúdo de B.

2.7 Comando de Atribuição

Comando é a descrição de uma ação a ser executada em um dado momento.

O comando de atribuição permite que se forneça um valor a uma certa variável, onde a natureza deste valor tem de ser compatível com o tipo da variável na qual está sendo armazenado.

Formato:

```
Identificador ← expressão
```

Onde:

`Identificador`	É o nome da variável à qual está sendo atribuído o valor;
←	É o símbolo de atribuição
`expressão`	Pode ser uma expressão aritmética, expressão lógica ou expressão literal de cuja avaliação é obtido o valor a ser atribuído à variável.

Exemplos:

- `K ← 1`
- `COR ← "Azul"`
- `Media ← (N1 + N2) / 2`
- `Sim ← (X = 0) e (Y ≠ 2)`

Nos comandos em que o valor é representado por uma expressão aritmética ou lógica, estas devem ser avaliadas em primeiro lugar para que, então, o resultado obtido seja armazenado na variável.

2.8 Comandos de Entrada e Saída

As unidades de entrada e saída são dispositivos que possibilitam a comunicação entre o usuário e o computador.

Como determinar o momento da entrada dos dados para o programa e a saída dos resultados obtidos para o usuário?

Os **comandos de entrada e saída** são as ferramentas para esta finalidade.

Formato de um comando de entrada:

```
leia lista-de-identificadores
```

Onde:

`leia`	É uma palavra-chave;

`lista-de-identificadores`	São os nomes das variáveis, separados por vírgula, nas quais serão armazenados os valores provenientes do meio de entrada.

Exemplo:

Supondo que `NOTA` e `NUM` sejam variáveis do tipo numérico, o comando:

```
leia NOTA, NUM
```

Indica que dois valores numéricos serão lidos de uma unidade de entrada, quando este comando for executado.

Analogamente, um comando de saída tem a forma geral:

Formato de um comando de saída:

`escreva lista-de-identificadores` e/ou `constantes`

Onde:

`escreva`	É uma palavra-chave;
`lista-de-identificadores`	São os nomes das variáveis mostrados ao usuário através de um meio de saída.

Exemplo:

O comando:

```
escreva A, X, 35
```

Indica que a constante `35` e mais os conteúdos das posições de memória, representados pelos identificadores `A` e `X`, serão exibidos em uma unidade de saída.

2.9 Estrutura Seqüencial

Num algoritmo aparecem em primeiro lugar as declarações seguidas por comandos que, se não houver indicação em contrário, deverão ser

executados numa seqüência linear, seguindo-se o texto em que estão escritos, de cima para baixo.

Exemplo:

```
Algoritmo
    declare A, B, C numérico
    leia A, B
    C ← 0
    C ← (A + B) × C
    escreva A, B, C
fim Algoritmo
```

Neste exemplo, após serem definidos os tipos das variáveis `A, B, C,` os valores de `A` e `B` serão lidos, o valor de `C` calculado e os valores contidos em `A, B` e `C` serão escritos.

2.10 Estrutura Condicional

A estrutura condicional permite a escolha do grupo de ações e estruturas a ser executado quando determinadas condições são ou não satisfeitas.

A estrutura condicional pode ser apresentada através de uma estrutura **simples** ou de uma estrutura **composta**.

Formato de uma Estrutura Condicional Simples:

```
se condição
    então seqüência de comandos
fim se
```

Exemplo:

```
Algoritmo
    declare A, B, C numérico
    leia A, B, C
    se A + B < C
        então escreva "MENSAGEM"
    fim se
fim Algoritmo
```

Formato de uma Estrutura Condicional Composta:

```
se condição
    então seqüência 1 de comandos
    senão seqüência 2 de comandos
fim se
```

Neste caso, a seqüência 1 de comandos só será executada se a condição for verdadeira e a seqüência 2 de comandos só será executada se a condição for falsa.

Exemplo:

```
Algoritmo
    declare A, B, C, D numérico
    leia A, B
    se A = B
        então   C ← 1,5
                D ← 2,5
        senão   C ← -1,5
                D ← -2,5
    fim se
    escreva C, D
fim Algoritmo
```

2.10 Estrutura de Repetição

A estrutura de repetição permite que uma seqüência de comandos seja executada repetidamente até que uma determinada condição de interrupção seja satisfeita.

A estrutura é delimitada pelo comando `repita` e pela expressão `fim repita` e a interrupção é feita através do comando `interrompa`.

A condição de interrupção que deve ser satisfeita é representada por uma expressão lógica.

A estrutura de interrupção pode apresentar três formas de interrupção: no Início da repetição, no meio da repetição ou no final da repetição.

Formato para a interrupção de início da estrutura:

```
repita
    se condição
        então interrompa
    fim se
    seqüência de comandos
fim repita
```

Exemplo:

```
Algoritmo
    declare PAR, SOMA numérico
    SOMA ← 0
    PAR ← 100
    repita
        se PAR > 200
            então interrompa
        fim se
        SOMA ← SOMA + PAR
        PAR ← PAR + 2
    fim repita
    escreva SOMA
fim Algoritmo
```

Formato para a interrupção no meio da estrutura:

```
repita
    seqüência 1 de comandos
    se condição
        então interrompa
    fim se
    seqüência 2 de comandos
fim repita
```

Exemplo:

```
Algoritmo
    declare PAR, SOMA numérico
    SOMA ← 0
    PAR ← 100
    repita
        PAR ← PAR + 2

        se PAR > 200
            então interrompa
        fim se
        SOMA ← SOMA + PAR
    fim repita
    escreva SOMA
fim Algoritmo
```

Formato para a interrupção no final da estrutura:

```
repita
    seqüência de comandos
    se condição
        então interrompa
    fim se
fim repita
```

Exemplo:

```
Algoritmo
    declare PAR, SOMA numérico
    SOMA ← 0
    PAR ← 100
    repita
        SOMA ← SOMA + PAR
        PAR ← PAR + 2
        se PAR > 200
            então interrompa
        fim se
    fim repita
    escreva SOMA
fim Algoritmo
```

Bibliografia

FARRER, H. et. alli. *Algoritmos Estruturados.* 3ª. edição. Rio de Janeiro. LTC – Livros Técnicos e Científicos Editora S/A, 1999.

GERSTING, J. L. *Fundamentos Matemáticos para a Ciência da Computação.* 3ª. edição. Rio de Janeiro. LTC – Livros Técnicos e Científicos Editora S/A, 1993.