FUNDAÇÃO
UNIVERSIDADE FEDERAL
DE MATO GROSSO DO SUL
UFMS

Algoritmos e Programação I

Modularização

Prof. Fernando Maia da Mota
mota.fernandomaia@gmail.com
CPCX/UFMS

Modularização

- Os algoritmos que temos construído até então são muito simples, pois resolvem problemas simples e apresentam apenas os componentes mais elementares dos algoritmos:
  - constantes, variáveis, expressões condicionais e estruturas de controle.

Entretanto, a maioria dos algoritmos resolve problemas complicados, cuja solução pode ser vista como formada de várias subtarefas ou módulo, cada qual resolvendo uma parte específica do problema.

Modularização

- Um módulo nada mais é do que um grupo de comandos que constitui um trecho de algoritmo com uma função bem definida e o mais independente possível das demais partes do algoritmo.

- A execução de um algoritmo contendo vários módulos pode ser vista como um processo cooperativo.

Modularização

- A construção de algoritmos compostos por módulos, ou seja, a construção de algoritmos através de modularização possui uma série de vantagens:
  - Torna o algoritmo mais fácil de escrever;
  - Torna o algoritmo mais fácil de ler;
  - Eleva o nível de abstração;
  - Economia de tempo, espaço e esforço;
  - Estende a linguagem.
    - Ex. operadores leia e escreva.

Modularização

- A maneira mais intuitiva de proceder à modularização de problemas é realizada através da definição de um módulo principal, que organiza e coordena o trabalho dos demais módulos, e de módulos específicos para cada uma das subtarefas do algoritmo.

- Um módulo é formado pelo seu:
  - Corpo -> grupos de comandos
  - Interface -> descrição dos dados de entrada e de saída do módulo.

Modularização

- A maneira mais intuitiva de proceder à modularização de problemas é realizada através da definição de um módulo principal, que organiza e coordena o trabalho dos demais módulos, e de módulos específicos para cada uma das subtarefas do algoritmo.

- Um módulo é formado pelo seu:
  - Corpo -> grupos de comandos
  - Interface -> descrição dos dados de entrada e de saída do módulo.

Modularização

- O conhecimento da interface de um módulo é tudo o que é necessário para a utilização correta do módulo em um algoritmo.
- A interface de um módulo é definida em termos de **parâmetros**.
- Existem três tipos de parâmetros:
  - Parâmetros de entrada
  - Parâmetros de saída
  - Parâmetros de entrada e saída

Modularização

- Há dois tipos de módulos:
  - **Função**: uma função é um módulo que produz um único valor de saída. Ela pode ser vista como uma expressão que é avaliada para um único valor, sua saída, assim como uma função em Matemática.
  - **Procedimento**: um procedimento é um tipo de módulo usado para várias tarefas, não produzindo valores de saída.

Modularização

- Por exemplo, se um módulo para determinar o menor de dois números é necessário, ele deve ser implementado como uma função, pois ele vai produzir um valor de saída. Por outro lado, se um módulo para determinar o maior e o menor valor de uma sequência de números é requerido, ele deve ser implementado como um procedimento, pois ele vai produzir dois valores de saída.

Modularização

- A interface de uma **função** tem a seguinte forma geral.

```
função <tipo de retorno> <nome> (<lista de parâmetros formais>)
    var
    inicio
fimfunção
```

onde

- **nome** é um identificador único que representa o nome da função;
- **lista de parâmetros formais** é uma lista dos parâmetros da função;
- **tipo de retorno** é o tipo do valor de retorno da função.

Modularização

- Exemplo:

```
//função para calcular o quadrado de um número
função real quadrado (r : real )
    var
        // declaração de variáveis
        X : real

    inicio
    // calcula o quadrado
    x ← r * r

    // retorna o quadrado calculado
    retorna x

fimfunção
```

Modularização

- A interface de um **procedimento** tem a seguinte forma geral.

```
procedimento <nome> (<lista de parâmetros formais>)
    var
    inicio
fimfunção
```

onde

- **nome** é um identificador único que representa o nome do procedimento;
- **lista de parâmetros formais** é uma lista dos parâmetros da função.

Modularização

- Exemplo:

```
//procedimento para ler um número
procedimento ler_número (val : ref real)
    inicio
    // solicita a entrada de um número
    escreva "Entre com um número:"
    leia val

fimprocedimento
```

- A palavra-chave *ref*, que antecede o nome da variável na lista de parâmetros formais do procedimento, é utilizada para definir *val* como parâmetro de saída ou entrada e saída.

Modularização

- Funções e procedimentos não são diferentes apenas na forma como são implementados, mas também na forma como a solicitação da execução deles, ou simplesmente **chamada**, deve ser realizada.

- A chamada de uma função é usada como um valor constante que deve ser atribuído a uma variável ou como parte de uma expressão, enquanto a chamada de um procedimento é realizada como um comando a parte.

Modularização

- Como exemplo, considere um algoritmo para ler um número e exibir o seu quadrado.

- Este algoritmo deve utilizar o procedimento *ler_número* e a função *quadrado*, vistos antes, para ler o número e obter o seu quadrado, respectivamente.

Modularização

- Exemplo:

```
//algoritmo para calcular e exibir o quadrado de um número
// fornecido como entrada
algoritmo "calcula_quadrado"
    var
        // declaração de variáveis
        num, x : real
    inicio
    // lê um número
    ler_número(num)

    // calcula o quadrado do número lido
    x ← quadrado(num)

    // escreve o quadrado
    escreva "O quadrado do número é:", x
fimalgoritmo
```

Modularização

- Os valores dos parâmetros reais de entrada são passados por um mecanismo denominado **cópia**, enquanto os valores dos parâmetros reais de saída e entrada e saída são passados por um mecanismo denominado **referência**.

Modularização

- O escopo de um módulo (ou variável) de um algoritmo é a parte ou partes do algoritmo em que o módulo (ou variável) pode ser referenciado.

- Os módulos de um algoritmo são organizados por níveis. No primeiro nível, temos apenas o algoritmo principal. Aqueles módulos que devem ser acessados pelo algoritmo principal devem ser escritos dentro dele e, nesta condição, são ditos pertencerem ao segundo nível. Os módulos escritos dentro de módulos de segundo nível são ditos módulos de terceiro nível. E assim sucessivamente.

Modularização

- Exemplo:

```
algoritmo "calcula quadrado"
    // módulo para cálculo do quadrado de um número
    função real quadrado (real r)
        var
            // declaração de variáveis
            X : real
        inicio
            // calcula o quadrado
            x ← r * r

            // passa para o algoritmo chamador o valor obtido
            retorna x

    fimfunção
```

Modularização

```
// módulo para ler um número
procedimento ler_número (val : ref real)
    inicio
    // solicita a entrada de um número
    escreva "Entre com um número:"
    leia val

fimprocedimento

// declaração de constantes e variáveis
var
    num, x : real
```

Modularização

```
Inicio

    // lê um número
    ler_número(num)

    // calcula o quadrado
    x ← quadrado(num)

    // escreve o quadrado
    escreva "O quadrado do número é: ", x

fimalgoritmo
```

Modularização

Problemas e soluções

Modularização

- Escreva um algoritmo para ler dois números e exibir o menor dos dois. A verificação de qual deles é o menor deve ser realizada por uma função.

Modularização

```
// algoritmo para o encontrar e exibir o menor de dois números
algoritmo "encontra menor de dois"

    // módulo para encontrar o menor de dois números
    função inteiro menor_de_dois (a, b : inteiro)
        var
            // declaração de variáveis
            menor : inteiro
        inicio
            menor ← a
            se a > b então
                    menor ← b
            fimse

            retorna menor
    fimfunção
```

Modularização

```
// declaração de constantes e variáveis
var
        x, y, z : inteiro

inicio

    // lê dois números
    escreva "Entre com dois números: "
    leia x, y

    // obtém o menor dos dois
    z ← menor_de_dois(x, y)

    // escreve o menor dos dois
    escreva "O menor dos dois é: ", z
fimalgoritmo
```

Modularização

- Escreva um algoritmo para ler três números e exibir o maior e o menor dos três. A obtenção do maior e do menor número deve ser realizada por um procedimento.

Modularização

```
// algoritmo para o encontrar e exibir o menor e o maior de três
// números
algoritmo "encontra min max"

    // módulo para encontrar o menor e o maior de três números
    procedimento min_max(a, b, c : inteiro)
        Var
            min, max : inteiro
        inicio
        // encontra o maior dos três
        se a ≥ b E a ≥ c então
            max ← a
        senão
            se b ≥ c então
                    max ← b
            senão
                    max ← c
            fimse
        fimse
```

Modularização

```
        // encontra o menor dos três
        se a ≤ b E a ≤ c então
            min ← a
        senão
            se b ≤ c então
                    min ← b
            senão
                    min ← c
            fimse
        Fimse
        Escreva min,max
Fimprocedimento

Var
// declaração de constantes e variáveis
x, y, z : inteiro
```

Modularização

```
inicio
    // lê três números
    escreva "Entre com três números: "
    leia x, y, z

    // obtém o menor e o maior dos três
    min_max(x, y, z)

fimalgoritmo
```

Modularização

- Escreva um algoritmo para ler três números e escrevê-los em ordem crescente. Utilize, obrigatoriamente, um procedimento para trocar o valor de duas variáveis.

Modularização

```
// algoritmo para ler três números e escrevê-los em ordem
// não decrescente
algoritmo "ordena três"
    // módulo para trocar o valor de duas variáveis
    procedimento troca(a, b : ref inteiro)
            // declaração de variáveis
            aux : inteiro
            // troca os valores
            aux ← a
            a ← b
            b ← aux
    fimprocedimento

    var
            // declaração de constantes e variáveis
            l, m, n : inteiro
```

Modularização

```
inicio
        // lê os três números
        escreva "Entre com três números: "
        leia l, m, n

        // encontra o menor e põe em l
        se l > m OU l > n então
                se m ≤ n então
                        troca(l,m)
                senão
                        troca(l, n)
                fimse
        fimse
        se m > n então
                troca(m, n)
        fimse

        escreva "Os números em ordem: ", l, m, n
fimalgoritmo
```

Modularização

- Neste algoritmo, os parâmetros do procedimento troca são parâmetros de entrada e saída, pois para trocar os valores dos parâmetros reais dentro de um procedimento, estes valores devem ser copiados para os parâmetros formais e as mudanças nos parâmetros formais devem ser refletidas nos parâmetros reais, uma vez que as variáveis precisam retornar do procedimento com os valores trocados.

Referências

- **SIQUEIRA, Marcelo F. Algoritmos e Estrutura de Dados. Mato Grosso do Sul: CCET/CPCX - UFMS, 2007.**