Técnicas e Desenvolvimento de Algoritmos

Educação a Distância
Cruzeiro do Sul Educacional
Campus Virtual

Material Teórico

Métodos (Procedimento e Funções)

Responsável pelo Conteúdo:
Prof. Esp. Alexander Gobbato Albuquerque

Revisão Textual:
Profa. Ms. Rosemary Toffoli

UNIDADE Métodos (Procedimento e Funções)

- Mecanismo de Funcionamento
- Funções
- Procedimentos

Objetivo de APRENDIZADO

Nesta unidade, estudaremos os conceitos básicos para a criação de subalgoritmos também chamados de subprogramas. Esses subprogramas podem ser chamados em qualquer parte do código e quantas vezes forem necessárias. Esses subprogramas podem se chamar procedimentos e funções.

Hoje veremos alguns assuntos introdutórios na nossa disciplina e aproveito para apresentar-lhes alguns conceitos que utilizaremos na estrutura de todas as nossas unidades.

Para obter um bom aproveitamento nesta unidade, vamos conferir sua estrutura:

Conteúdo Teórico: neste link, você encontrará o material principal de estudos na forma de texto escrito.

Atividade de Sistematização: os exercícios disponibilizados são de autocorreção e visam a que você pratique o que aprendeu na disciplina e que identifique os pontos em que precisa prestar mais atenção, ou pedir esclarecimentos a seu tutor. Além disso, as notas atribuídas aos exercícios serão parte de sua média final na disciplina.

Atividade de Aprofundamento: é uma atividade dissertativa ou de pesquisa.

Material Complementar e Referências Bibliográficas: nestes links, você poderá ampliar seus conhecimentos.

Vídeo aula: nestes links serão apresentadas algumas ferramentas na prática e também a resolução de alguns exercícios de forma prática.

Atenção

Lembramos importância de realizar todas as atividades propostas dentro do prazo estabelecido para cada Unidade, dessa forma, você evitará que o conteúdo se acumule e que você tenha problemas ao final do semestre.

Uma última recomendação, caso tenha problemas para acessar algum item da disciplina, ou dúvidas com relação ao conteúdo, não deixe de entrar em contato com seu professor tutor através do botão mensagens ou fóruns.

Contextualização

Para se desenvolver um programa, é extremamente importante a compreensão do problema. Esse entendimento faz parte da construção do algoritimo e quanto mais complexo for o problema, maior será o programa a ser desenvolvido.

Nas unidades anteriores, vimos como criar variáveis, os tipos, aprendemos recursos de programação que facilitam na hora de executar o código, mas tudo isso pode gerar um problema. Um programa complexo pode ter milhares de linhas de programa e o que podemos fazer para ajudar ou facilitar o desenvolvimento e a manutenção desse código.

Na unidade de hoje aprenderemos como criar funções e procedimentos para quando tivermos que desenvolver algo complexo, poderemos modularizar o problema e facilitar a solução e/ou manutenção.

Introdução

A complexidade dos algoritmos está intimamente ligada à da aplicação a que se destinam. Em geral, problemas complicados exigem algoritmos extensos para sua solução.

Sempre é possível dividir problemas grandes e complicados em problemas menores e de solução mais simples. Assim, pode-se solucionar cada um destes pequenos problemas separadamente, criando algoritmos para tal (subalgoritmos).

Um subalgoritmo é um nome dado a um trecho de um algoritmo mais complexo e que, em geral, encerra em si próprio um pedaço da solução de um problema maior a qual o algoritmo que a ele está subordinado.

Em resumo, os subalgoritmos são importantes na:

- Subdivisão de algoritmos complexos, facilitando o seu entendimento;
- Estruturação de algoritmos, facilitando principalmente a detecção de erros e a documentação de sistemas; e
- Modularização de sistemas, que facilita a manutenção de softwares e a reutilização de subalgoritmos já implementados.

A ideia da reutilização de software tem sido adotada por muitos grupos de desenvolvimento de sistemas de computador, devido à economia de tempo e trabalho que proporcionam. Seu principal objetivo é solucionar uma série de tarefas e possibilitar o acréscimo de novos algoritmos. A este conjunto dá-se o nome de biblioteca. No desenvolvimento de novos sistemas, procura-se ao máximo basear sua concepção em subalgoritmos já existentes na biblioteca, de modo que a quantidade de software realmente novo que deve ser desenvolvido é minimizada.

Muitas vezes os subalgoritmos podem ser úteis para encerrar em si certa sequencia de comandos que é repetida várias vezes num algoritmo. Nestes casos, os subalgoritmos proporcionam uma diminuição do tamanho de algoritmos maiores.

Mecanismo de Funcionamento

Um algoritmo completo é dividido num algoritmo principal e diverso subalgoritmos (tantos quantos forem necessários). O algoritmo principal é aquele por onde a execução do programa sempre se inicia. Este pode eventualmente invocar os demais subalgoritmos ou subprogramas.

As definições dos subalgoritmos estão sempre colocadas no trecho após a definição das variáveis globais e antes do corpo do algoritmo principal, veja o modelo abaixo:

```
Algoritmo  <nome do algoritmo>
    Var  <definição das variáveis globais>
    <definições dos subalgoritmos>
Inicio
    <corpo do algoritmo principal>
Fim.
```

Durante a execução do algoritmo principal, quando se encontra um comando de invocação de um subalgoritmo, a execução do mesmo é interrompida. A seguir, passa-se à execução dos comandos do corpo do subalgoritmo. Ao seu término, retoma-se a execução do algoritmo que o chamou (no caso, o algoritmo principal) no ponto onde foi interrompida (comando de chamada do subalgoritmo) e prossegue-se pela instrução imediatamente seguinte.

Note, também, que é possível que um subalgoritmo chame outro através do mesmo mecanismo.

Definição de Subalgoritmos

A definição de um subalgoritmo consta de:

- Um cabeçalho, onde estão definidos o nome e o tipo do subalgoritmo, bem como os seus parâmetros e variáveis locais;
- Um corpo, onde se encontram as instruções (comandos) do subalgoritmo.
- O nome de um subalgoritmo é o nome simbólico pelo qual ele é chamado por outro algoritmo.
- O corpo do subalgoritmo contém as instruções que são executadas cada vez que ele é invocado.
- Variáveis locais são aquelas definidas dentro do próprio subalgoritmo e só podem ser utilizadas pelo mesmo.

Parâmetros são canais por onde os dados são transferidos pelo algoritmo chamador a um subalgoritmo, e vice-versa. Para que possa iniciar a execução das instruções em seu corpo, um subalgoritmo às vezes precisa receber dados do algoritmo que o chamou e, ao terminar sua tarefa, o subalgoritmo deve fornecer ao algoritmo chamador os resultados da mesma. Esta comunicação bidirecional pode ser feita de dois modos que serão estudados mais à frente: por meio de variáveis globais ou por meio da passagem de parâmetros.

O tipo de um subalgoritmo é definido em função do número de valores que o subalgoritmo retorna ao algoritmo que o chamou. Segundo esta classificação, os algoritmos podem ser de dois tipos:

- funções, que retornam um, e somente um, valor ao algoritmo chamador;
- procedimentos, que retornam zero (nenhum) ou mais valores ao algoritmo chamador.

Na realidade, a tarefa desempenhada por um subalgoritmo do tipo função pode perfeitamente ser feita por outro do tipo procedimento (o primeiro é um caso particular deste). Esta diferenciação é feita por razões históricas, ou, então, pelo grande número de subalgoritmos que se encaixam na categoria de funções.

Funções

O conceito de Função é originário da idéia de função matemática (por exemplo, raiz quadrada, seno, cosseno, tangente, logaritmo, entre outras), onde um valor é calculado a partir de outro(s) fornecido(s) à função.

A sintaxe da definição de uma função é dada a seguir:

```
Função <nome> ( <parâmetros> ) <tipo_de_dado>
    Var <variáveis locais>
Início
    <comando composto>
Fim
```

Temos que:

- **<nome>** é o nome simbólico pelo qual a função é invocada por outros algoritmos;
- **<parâmetros>** são os parâmetros da função;
- **<tipo de dado>** é o tipo de dado da informação retornado pela função ao algoritmo chamador;
- **<variáveis locais>** consiste na definição das variáveis locais à função. Sua forma é análoga à da definição de variáveis num algoritmo;
- **<comando composto>** é o conjunto de instruções do corpo da função.

Dentro de um algoritmo, o comando de invocação de um subalgoritmo do tipo função sempre aparece dentro de uma expressão do mesmo tipo que o do valor retornado pela função.

A invocação de uma função é feita pelo simples aparecimento do nome da mesma, seguido pelos respectivos parâmetros entre parênteses, dentro de uma expressão. A função é executada e, ao seu término, o trecho do comando que a invocou é substituído pelo valor retornado pela mesma dentro da expressão em que se encontra, e a avaliação desta prossegue normalmente.

Dentro de uma função, e somente neste caso, o comando Retorne <expressão> é usado para retornar o valor calculado pela mesma. Ao encontrar este comando, a expressão entre parênteses é avaliada, a execução da função é terminada neste ponto e o valor da expressão é retornado ao algoritmo chamador. Vale lembrar que uma expressão pode ser uma simples constante, uma variável ou uma combinação das duas por meio de operadores. Esta expressão deve ser do mesmo tipo que o valor retornado pela função.

O algoritmo a seguir é um exemplo do emprego de função para calcular o valor de um número elevado ao quadrado.

```
Algoritmo Exemplo_de_função
Var X, Y : real

Função Quad(real w) real
Var
    real Z
Inicio
    Z = w * w
    Retorna Z
Fim

Início
    Escreva "Digite um número"
    Leia X
    Y = Quad(X)
    Escrever X, " elevado ao quadrado é = ", Y
Fim.
```

Do exemplo anterior é importante notar que:

- a função Quad toma W como parâmetro do tipo real, retorna um valor do tipo real e possui Z como uma variável local real;
- o comando de invocação da função Quad aparece no meio de uma expressão, no comando de atribuição dentro do algoritmo principal.

Procedimentos

Um procedimento é um subalgoritmo que retorna zero (nenhum) ou mais valores ao (sub) algoritmo chamador. Estes valores são sempre retornados por meio dos parâmetros ou de variáveis globais, mas nunca explicitamente, como no caso de funções. Portanto, a chamada de um procedimento nunca surge no meio de expressões, como no caso de funções. Pelo contrário, a chamada de procedimentos só é feita em comandos isolados dentro de um algoritmo, como as instruções de entrada (Leia) e saída (Escreva) de dados.

A sintaxe da definição de um procedimento é:

```
Procedimento  <nome>  ( <parâmetros> )
     Var  <variáveis locais>
Inicio
     <comando composto>
Fim.
```

Temos que:

- **<nome>** é o nome simbólico pelo qual o procedimento é invocado por outros algoritmos;
- **<parâmetros>** são os parâmetros do procedimento;
- **<variáveis locais>** são as definições das variáveis locais ao procedimento. Sua forma é análoga à da definição de variáveis num algoritmo;
- **<comando composto>** é o conjunto de instruções do corpo do procedimento, que é executado toda vez que o mesmo é invocado.

O exemplo a seguir é um exemplo simples, onde um procedimento é usado para escrever o valor das componentes de um vetor.

Algoritmo Exemplo_procedimento

```
Algoritmo Exemplo_procedimento

Var
     matriz[1..10] de real

Procedimento ESC_VETOR()
Var
     Inteiro i
Inicio
     para i de 1 até 10 faça
          Escreva vet[i]
     fim_para
Fim

Procedimento LER_VETOR()
Var
     inteiro i
Inicio
     para i de 1 até 10 faça
          ler vet[i]
     fim_para
Inicio
     LER_VETOR()
     ESC_VETOR()
Fim
```

No exemplo é conveniente observar:

- •a forma de definição dos procedimentos LER_VETOR( ) e ESC_VETOR( ), que não possuem nenhum parâmetro, e usam a variável local i para "varrer" os componentes do vetor, lendo e escrevendo um valor por vez; e
- a forma de invocação dos procedimentos, por meio do seu nome, seguido de seus eventuais parâmetros (no caso, nenhum), num comando isolado dentro do algoritmo principal.

Variáveis Globais e Locais

Variáveis globais são aquelas declaradas no início de um algoritmo. Estas variáveis são visíveis (isto é, podem ser usadas) no algoritmo principal e por todos os demais subalgoritmos.

Variáveis locais são aquelas definidas dentro de um subalgoritmo e, portanto, somente visíveis (utilizáveis) dentro do mesmo. Outros subalgoritmos, ou mesmo o algoritmo principal, não podem utilizá-las.

No exemplo a seguir são aplicados estes conceitos.

```
Algoritmo Exemplo_variáveis_locais_e_globais
Var real X
Inteiro I

Função FUNC()real
Var
    X : matriz[1..5] de inteiro
    Y caracter[1...10] de inteiro
Inicio
...
Fim

Procedimento PROC
Var  lógico Y
Inicio
...
X = 4 * X
I = I + 1
...
Fim

Início
...
X = 3.5
...
Fim.
```

É importante notar no exemplo anterior que:

- as variáveis X e I são globais e visíveis a todos os subalgoritmos, à exceção da função FUNC, que redefine a variável X localmente;
- as variáveis X e Y locais ao procedimento FUNC não são visíveis ao algoritmo principal ou ao procedimento PROC. A redefinição local do nome simbólico X como uma matriz[5] de inteiro sobrepõe (somente dentro da função FUNC) a definição global de X como uma variável do tipo real;
- a variável Y dentro do procedimento PROC, que é diferente daquela definida dentro da função FUNC, é invisível fora deste procedimento;
- a instrução X = 8.5 no algoritmo principal, bem como as instruções X = 4 * X e I = I + 1 dentro do procedimento PROC, atuam sobre as variáveis globais X e I.

Parâmetros

Parâmetros são canais pelos quais se estabelece uma comunicação bidirecional entre um subalgoritmo e o algoritmo chamador (o algoritmo principal ou outro subalgoritmo). Dados são passados pelo algoritmo chamador ao subalgoritmo, ou retornados por este ao primeiro por meio de parâmetros.

Parâmetros formais são os nomes simbólicos introduzidos no cabeçalho de subalgoritmos, usados na definição dos parâmetros do mesmo. Dentro de um subalgoritmo trabalha-se com estes nomes da mesma forma como se trabalha com variáveis locais ou globais.

```
Função Média(real X, real Y) : real
Inicio
    Retorne (X + Y) / 2
Fim
```

No exemplo anterior, X e Y são parâmetros formais da função Média.

Parâmetros reais são aqueles que substituem os parâmetros formais quando da chamada de um subalgoritmo. Por exemplo, o trecho seguinte de um algoritmo invoca a função Média com os parâmetros reais 8 e 7 substituindo os parâmetros formais X e Y.

Z := Média(8, 7)

Assim, os parâmetros formais são úteis somente na definição (formalização) do subalgoritmo, ao passo que os parâmetros reais substituem-nos a cada invocação do subalgoritmo. Note que os parâmetros reais podem ser diferentes a cada invocação de um subalgoritmo.

Por fim, o uso do método é vantajoso devido ao seu estímulo à modularização de sistemas, que proporciona a criação de programas claros, fáceis de entender e, portanto, de manutenção mais barata.

Material Complementar

Para ampliar seus conhecimentos recomendo os livros abaixo:.

Explore

OLIVEIRA, J. F e MANZANO, J. A. N. G., **Algoritmos – Lógica para desenvolvimento de programação de computadores.** 22. ed. Editora: Erica.

LOPES, A. e GARCIA, G., **Introdução a programação – 500 algoritmos resolvidos.** 1. ed. Editora: Campus, 2002.

Referências

FARRER, H. **Algoritmos Estruturados.** 3. ed. Rio de Janeiro: Ltc-Livros Tecnicos e Cientifi, 1999.

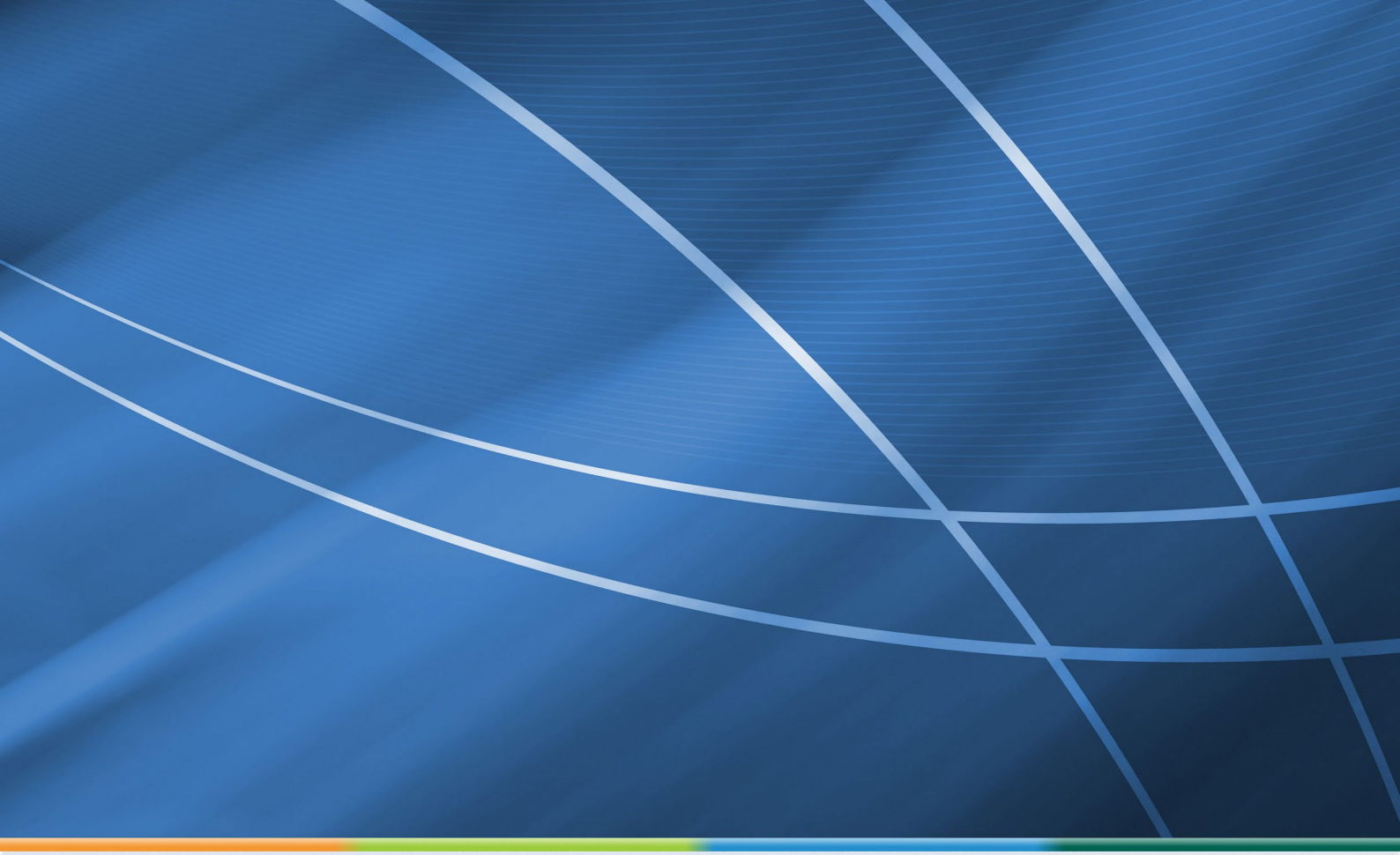
FORBELLONE, A. L. V.; EBERSPACHER, H. F. **Lógica de Programação: A Construção de Algoritmos e Estrutura de Dados.** 3. ed. São Paulo: Pearson Prentice Hall, 2008.

WIRTH, N. **Algoritmos e Estruturas de Dados.** Rio de Janeiro: Ltc-Livros Técnicos e Científicos, 1999.

Anotações


Educação a Distância
Cruzeiro do Sul Educacional
Campus Virtual

www.cruzeirodosulvirtual.com.br
Campus Liberdade
Rua Galvão Bueno, 868
CEP 01506-000
São Paulo SP Brasil
Tel: (55 11) 3385-3000



Universidade Cruzeiro do Sul

UNICID Universidade Cidade de S. Paulo

UNIFRAN Universidade de Franca

UDF Centro Universitário

Módulo Centro Universitário