Lógica de Programação

Aula 14 - Funções, Procedimentos, Parâmetros e conceitos de bibliotecas

Apresentação da Aula

Nesta aula, você conhecerá as **funções** e os **procedimentos**, recursos existentes na maioria das linguagens de programação. Conhecerá, ainda, o **escopo de variáveis** que definem como uma variável pode ser acessada a partir de uma função ou de um algoritmo principal.

Objetivos

Explorar o conceito de **subalgoritmos**;

Conhecer os tipos de **subalgoritmo** existentes e os elementos que os constituem;

Conhecer a sintaxe das **funções** em algoritmos;

Criar funções.

Particionamento de Algoritmos em Subalgoritmos

Conforme um programa cresce em quantidade de comandos, é comum que sua complexidade também seja ampliada. Essa complexidade tende a crescer, geralmente, à medida que são necessárias mais linhas de código para definir as diversas operações que os algoritmos necessitam executar no programa.

As operações que compõem um grande algoritmo poderão estar presentes diversas rotinas dos mais diferentes tipos. Em outras palavras, quanto maior for o problema a ser resolvido por um programa, maior será a chance de que sejam necessárias diversas operações para solucioná-lo. Assim, o programa torna-se maior (em tamanho) e mais complexo.

Um dos recursos que pode ser utilizado para simplificar a construção de grandes softwares é o uso de **subalgoritmos**.

> Um **subalgoritmo** é um trecho de programa que possui um nome e uma estrutura com início e fim definidos. Em sua estrutura, é realizada uma operação mediante um conjunto de passos. Esse **subalgoritmo** poderá ser utilizado em qualquer região do programa principal, desde que sejam fornecidas as informações (os parâmetros) especificadas na estrutura dos **subalgoritmos**.

Por meio do uso de **subalgoritmo**, o programador poderá particionar o programa em diversos “microalgoritmos”. Cada um desses “pedaços” deve possuir uma função específica (solucionar um pequeno problema). Dessa forma, ao definir-se esses pequenos trechos, podemos tornar a leitura do programa mais simples.

Outra vantagem é que, ao construir algoritmos com esses particionamentos, geralmente, facilita-se as correções de erros (os populares *bugs*). Isso é possível pois, devido à divisão de um “grande problema” em vários problemas menores, o

programador é capaz de focar em apenas uma pequena parte do programa, isto é, no **"subalgoritmo"** que compõe todo o conjunto da solução.

Entretanto, antes de utilizar um **subalgoritmo**, algumas informações são necessárias para construí-lo. São elas:

- **Nome** – É o identificador pelo qual o **subalgoritmo** será chamado no algoritmo principal.
- **Parâmetros** – São as informações (os dados) que serão utilizadas pelos comandos internos do **subalgoritmo**. Esses dados serão processados pelas instruções do **subalgoritmo** que, ao final, poderá devolver o resultado do processamento.
- **Variáveis** – Nos **subalgoritmos**, é, normalmente, necessário declarar variáveis em seu interior para que sejam utilizadas nos comandos interno. A definição da quantidade e dos tipos de variáveis dependerá das operações que serão realizadas pelo **subalgoritmo**.
- **Tipo** – **Subalgoritmos** podem ser de dois tipos: **funções** ou **procedimentos**.

Classificação de um subalgoritmo

A maioria das linguagens possui dois tipos de **subalgoritmos**: as **funções** e os **procedimentos**.

As **funções** são caracterizadas por, comumente, possuírem **parâmetros de entrada** (dados de entrada) e um **valor de saída**. Esse valor de saída existente nas funções é denominado de **retorno**. Algumas linguagens de programação permitem definir **funções** que não possuem o valor de retorno. Além disso, ao se definir uma **função**, dependendo da linguagem de programação, pode ser necessário definir o tipo de dado resultante que será **retornado** (devolvido ao programa principal) quando é concluído o uso dessa **função** (o retorno será do tipo inteiro, real, lógico ou texto).

Já os **subalgoritmos** do tipo **procedimento** possuem apenas os **parâmetros de entrada** e não possuem um **valor de saída**. Nesse caso, o **procedimento** realizará as operações diretamente nas variáveis que foram informadas por meio dos **parâmetros de entrada**, alterando os valores das variáveis do algoritmo principal.

Um **subalgoritmo** poderá ser utilizado, também, internamente a outro **subalgoritmo**. Seu uso é permitido em qualquer trecho do programa, desde que fornecidos os parâmetros de entrada necessários.

Sintaxe de Funções

A partir de agora, você aprenderá como definir **funções** para que sejam utilizadas em seus algoritmos. Uma **função** precisa possuir um **nome** e os **parâmetros** necessários para que seja possível utilizá-la.

Na **função**, será necessário inserir, internamente, os comandos que ela executará quando for utilizada. Esses comandos devem ser “autossuficientes” para, a partir dos valores dos **parâmetros de entrada** informados, gerar um resultado. Em outras palavras, uma **função** é um pequeno algoritmo que, a partir dos **parâmetros de entrada**, é capaz de gerar um resultado quando acionada.

O resultado gerado ao final de sua execução, isto é, o dado — proveniente da resolução das operações realizadas internamente — deve ser "enviado" para o algoritmo principal. Essa ação é denominada de **retorno** da função.

O código abaixo apresenta uma função que realiza a **soma** de dois números inteiros. Observe com atenção: no código presente na linha 1, há a definição do **nome da função** e de **dois parâmetros de entrada** necessários para que a função seja utilizada.

```
1 soma(x: inteiro, y: inteiro)   # Declaração de função com corpo
2   var d := x + y               # Comandos executados na função
3   retorne d                    # A última linha da função possui o comando de retorno
4 fim                            # O comando fim, ao final, define o 'limite' da função declarada
```

O código a seguir mostra como utilizar a função **soma** definida em um algoritmo qualquer, no qual foi necessário utilizar uma soma.

```
1  soma(x: inteiro, y: inteiro)   # Declaração de função com corpo
2    var d := x + y               # Comandos executados na função
3    retorne d                    # A última linha da função possui o comando de retorno
4  fim                            # O comando fim, ao final, define o 'limite' da função declarada
5
6  var a := 5
7  var b := 9
8  var c := 0
9
10 c := soma(a,b)
11
12 escreva "A soma de {a} + {b} é {c}"
13
```

No algoritmo visto, foi declarada a função **soma** que possui 2 **parâmetros** de entrada (x e y) do tipo **inteiro**. Internamente à função, em sua primeira linha, é realizada a soma dos valores de x + y. Já na segunda linha, o comando '**retorne** d' devolverá o resultado do processamento da função para o programa que solicitou a função **soma**. Nesse algoritmo, a linha 10 corresponde ao uso da função **soma** e na qual foram informados os parâmetros de entrada (variáveis **a** e **b**).

Escopos de Variáveis

As linguagens de programação possuem as **variáveis** para armazenar alguma informação durante a execução dos algoritmos. O que você não viu até agora é que essas **variáveis** podem ser declaradas em qualquer espaço do seu programa. Por esse motivo, é importante ficar atento ao que chamamos de **escopo da variável**. Você deverá sempre observar a abrangência em que uma variável estará disponível. Caso contrário, poderá tentar acessar uma variável que, por não se encontrar disponível em algum trecho específico de seu programa, provocará um erro.

As variáveis que são declaradas com o **escopo global** estão disponíveis em qualquer região de seu programa, independentemente do tamanho que seu programa possua.

Já as variáveis de **escopo local** estão disponíveis, apenas, na região em que foram declaradas. Por exemplo, uma variável que foi definida dentro de uma função existe apenas dentro daquela função. Após a execução e o encerramento de uma função, essa variável não mais existirá e, se o seu valor (conteúdo) não for armazenado em uma variável **global**, ele será descartado.

Lembre-se de que as variáveis ocupam espaço na memória do computador, por esse motivo, declarar variáveis de **escopo global** não utilizadas fará com que seu programa utilize mais memória do computador do que é realmente necessário. Assim, declare no **escopo global** apenas as variáveis de seu programa que você necessitará acessar em qualquer região do algoritmo.

Geralmente, você necessitará declarar as variáveis de **escopo local** ao utilizar uma **função** ou **procedimento**. Elas serão imprescindíveis ao processamento interno dessas estruturas, pois os dados precisam ser armazenados temporariamente para que os passos sejam realizados até a conclusão de todo processamento.

Comumente, ao final, apenas o resultado dessas estruturas será necessário, permitindo o descarte dos conteúdos armazenados nas variáveis utilizadas internamente nessas estruturas sem prejuízo ao programa principal.

Estamos nos aproximando do final desta disciplina 😢 (cara chorando - face crying). No encontro de hoje, você conheceu o que são **funções** e **procedimentos** e, além disso, viu que as variáveis possuem **escopos**. Esses conteúdos são importantes para compreender como particionar o seu programa de maneira que ele se torne mais legível, permitindo que a complexidade também seja particionada.

Os recursos que você acabou de conhecer estão presentes em todos os programas de computadores existentes. Portanto, fique atento a cada detalhe que você viu e, se ainda restar alguma dúvida, o melhor caminho é exercitar para entender como tais recursos funcionam. Além disso, compartilhe e discuta com os seus colegas o que você compreendeu desse conteúdo. Converse também com o seu mediador, caso ainda reste alguma dúvida.

Até a nossa próxima aula! 😉 (cara piscando - winking face)

Resumo

Nesta aula, você exercitou o uso de **funções** em algoritmos, conheceu o **escopo de variáveis** e aprendeu como isso afeta a construção de programas que utilizam **funções** e **procedimentos**.

Na próxima aula, você terá a oportunidade de exercitar a construção de **funções** na resolução de problemas.

Referências

Linguagem Potigol: Programação para todos. Disponível em: <http://potigol.github.io/>. Acesso: 16 out. 2018.