Algoritmos e Programação Estruturada

Escopo e passagem de parâmetros

Você sabia que seu material didático é interativo e multimídia? Isso significa que você pode interagir com o conteúdo de diversas formas, a qualquer hora e lugar. Na versão impressa, porém, alguns conteúdos interativos ficam desabilitados. Por essa razão, fique atento: sempre que possível, opte pela versão digital. Bons estudos!

Nesta webaula vamos ver sobre escopo de variáveis, bem como a passagem de parâmetros em uma função.

Escopo de variáveis

As variáveis são usadas para armazenar dados temporariamente na memória, porém o local onde esse recurso é definido no código de um programa determina seu escopo e sua visibilidade.

Veja a seguir um exemplo de variáveis em funções.

No exemplo, há duas variáveis chamadas “x”, mas isso não acarreta erro, pois mesmo as variáveis tendo o mesmo nome elas são definidas em lugares diferentes, uma está dentro da função `main()` e outra dentro da `testar()` e cada função terá seu espaço na memória de forma independente. Na memória, as variáveis são localizadas pelo seu endereço, portanto mesmo sendo declaradas com o mesmo nome, seus endereços são distintos.

O escopo é dividido em duas categorias: **local** ou **global**.

Variáveis locais	Variáveis globais
Elas existem e são “enxergadas” somente dentro do corpo da função onde foram definidas.	Elas existem e são “enxergadas” por todas as funções do programa.

No exemplo anterior, ambas as variáveis são locais.

No exemplo a seguir, veja a declaração e uso da variável global.

Sua inclusão é feita após a biblioteca de entrada e saída padrão. Na função principal não foi definida nenhuma variável com o nome de “x” e mesmo assim pode ser impresso seu valor na linha 10, pois é acessado o valor da variável global. Já na linha 12 é impresso o valor da variável global modificado pela função `testar()`, que retorna o dobro do valor.

Vamos ver um exemplo de utilização do escopo global de uma variável, um programa que calcula a média entre duas temperaturas distintas. Na linha 2 foram declaradas duas variáveis. O programa inicia a execução na linha 8. Na 9 é solicitado ao usuário digitar duas temperaturas, as quais são armazenadas dentro das variáveis globais criadas. Na linha 11 a função `calcularMedia()`é invocada para fazer o cálculo da média usando os valores das variáveis globais.

Variável global e local com mesmo nome

Na linguagem C, para conseguirmos acessar o valor de uma variável global, dentro de uma função que possui uma variável local com mesmo nome, devemos usar a instrução **`extern`** (MANZANO, 2015). Veja no exemplo, que foi necessário criar uma nova variável chamada “`b`”, com um bloco de instruções (linhas 8 – 11), que atribui a nova

variável o valor “externo” de x.

Passagem de parâmetros para funções

Ao definir uma função, pode-se também estabelecer que ela receberá informações “de quem” a invocou, ou seja, “quem chamar” a função deve informar os valores, sobre os quais o cálculo será efetuado. Ao criar uma função que recebe parâmetros é preciso especificar qual o tipo de valor que será recebido. Uma função pode receber parâmetros na forma de **valor** ou de **referência** (SOFFNER, 2013).

Veja a seguir, a sintaxe utilizada:

```
<tipo de retorno> <nome> (<parâmetros>) {
  <Comandos da função>
  <Retorno> (não obrigatório)
}
```

Passagem por valor

Na passagem parâmetros por valores, a função cria variáveis locais automaticamente para armazenar esses valores e após a execução da função essas variáveis são liberadas. Veja a seguir, um exemplo de definição e chamada de função com passagem de valores.

Explicação

> Ao utilizar variáveis como argumentos, na passagem de parâmetros por valores, essas variáveis não são alteradas, pois é fornecido uma cópia dos valores armazenados para a função (SOFFNER, 2013).

Passagem por referência

A utilização funções com de passagem de parâmetros por referência está diretamente ligada aos conceitos de ponteiro e endereço de memória. A ideia da técnica é análoga a passagem por valores, ou seja, a função será definida de modo a receber certos parâmetros de “quem” faz a chamada do método deve informar esses argumentos. Entretanto, o comportamento e o resultado são diferentes.

> Na passagem por referência não será criada uma cópia dos argumentos passados, na verdade, será passado o endereço da variável e função irá trabalhar diretamente com os valores ali armazenados (SOFFNER, 2013).

Como a função utiliza o endereço (ponteiros), na sintaxe serão usados os operadores ***** e **&** (MANZANO, 2015).

Na definição da função os parâmetros a serem recebidos devem ser declarados com *****, por exemplo:

```
int testar(int* parametro1, int* parametro2)
```

Na chamada da função os parâmetros devem ser passados com o **&**, por exemplo:

```
resultado = testar(&n1,&n2)
```

Explicação

Passagem de vetor

Esse recurso pode ser utilizado para criar funções que preenchem e imprimem o conteúdo armazenado em um vetor, evitando a repetição de trechos de código. Na linguagem C, a passagem de um vetor é feita implicitamente por referência. Isso significa que mesmo não utilizando os operadores “*” e “&”, quando uma função que recebe um vetor é invocada, o que é realmente passado é o endereço da primeira posição do vetor.

Na definição da função os parâmetros a serem recebidos devem ser declarados com colchetes sem especificar o tamanho, por exemplo:

```
int testar(int v1[], int v2[])
```

Na chamada da função os parâmetros devem ser passados como se fossem variáveis simples, por exemplo:

```
 resultado = testar(n1,n2)
```

No exemplo a seguir o programa por meio de uma função preenche um vetor de três posições e em outra função percorre o vetor imprimindo o dobro de cada valor do vetor.

Explicação

Nesta webaula vimos sobre escopo de variáveis, bem como a passagem de parâmetros em uma função, como passagem de parâmetros por valor ou por referência. As técnicas apresentadas, fazem parte do cotidiano de um desenvolvedor em qualquer linguagem de programação.