ALGORÍTMOS

Prof. Nilton

Aula de Hoje

- Exercícios
- Módulos / Funções / Métodos

Exercícios

3) Crie um programa que faça a rotação para a direita de um vetor, como o exemplo:

original	10	5	7	11	3	45
modificado	45	10	5	7	11	3

Módulos – Programação Modular

Conforme avançamos, os problemas se tornam mais complexos e longos fazendo a resolução desses problemas se tornem cada vez mais difícil.

Uma abordagem com eficiência comprovada que pode ajudar a resolver esses problemas com mais facilidade é a divisão de tarefas complexas e longas em tarefas mais simples e curtas.

Módulos – Programação Modular

Uma das vantagens da divisão das atividades é que muitas vezes a solução para um problema menor pode ser reaproveitada na resolução de diversos problemas maiores.

Módulos – Programação Modular

A modularização de um algoritmo/programa é uma forma de dividir as tarefas em subalgoritmos/subprogramas, e cada um desses módulos cuida de uma parte separada do problema. Cada módulo funciona semelhante a um programa, tendo em geral, entrada, processamento e saída.

Para que um módulo possa ser executado, precisa ser ativado ou chamado por um outro módulo.

O fluxo de execução de um algoritmo modularizado começa pelo módulo principal e é desviado para qualquer outro módulo que seja chamado pelo módulo principal.

Programação Modular

A Forma geral para definir um módulo é:

```
tipo_retorno nome_do_módulo ( lista de parâmetros )
{
        corpo do módulo
        return valor_tipo_retorno
}
```

Programação Modular - Fatorial

Para ilustrar a criação de funções, faremos um exemplo que calcula o fatorial de um numero inteiro:

```
// Função Principal
Public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n;
        System.out.println(“Digite um numero”);
        n = scan.nextInt();
        fat(n);
}
```

Programação Modular - Fatorial

```
// Função fatorial com passagem de parametro
static void fat(int n){
        int i;
        int f = 1;
        for ( i = 1; i <= n; i++)
                f *= i;
        System.out.println("Fatorial: " + f);
}
```

Neste exemplo a função fat não tem nenhum valor de retorno, portanto colocamos a palavra **void** antes do nome da função, indicando a ausência de um valor de retorno.

Programação Modular

Passagem de parâmetros

Parâmetros são os meios pelos quais nós passaremos dados para o módulo, e estes módulos irão trabalhar especificamente sobre essas informações que passamos.

Sempre que um módulo receber dados, estes serão recebidos através dos parâmetros.

Em Java, assim como em C ou C++, você precisa deixar bem claro quais e qual o tipo dos dados que você vai passar.

Programação Modular

Vejamos o exemplo anterior:

```
void fat(int n){

}
```

O parâmetro que será passado para a função **fat** deverá ser obrigatoriamente do tipo inteiro.

Programação Modular

Retorno de Valores

Os módulos podem ou não retornar valores. Esta característica pode ser extremamente útil e amplia bastante a possibilidade de utilização dos módulos. Caso um módulo não retorno nenhum valor, podemos dizer que se trata de um módulo com retorno neutro(nulo).

Em algumas linguagens os módulos que não retornam valores são chamados de **procedimentos** e os que retornam valores são chamados de **função**.

No Java os módulos são chamados, de forma geral, de **métodos**.

Programação Modular

Retorno de Valores

O retorno de valores é a saída do módulo. Por meio dele o módulo pode dar uma "resposta" para o módulo ativador.

Essa resposta pode ser um indicador de que todas as operações realizadas dentro do módulo foram realizadas com sucesso, o resultado de uma operação, entre outros.

As respostas devem ser valores do pertencentes aos tipos de dados da linguagem. (int, boolean, String, etc.).

Essa resposta de ser recebida e tratada pelo módulo ativador.

Programação Modular

Outro exemplo:

```
        static boolean par(int n){
            if (n%2 == 0)
                return true
            else
                return false;
        }
// método ativador
            if (par(n) == true)
                    System.out.println(“O Numero é Par”);
```

Exercícios

1) Altere o exercício 1 da aula 5 e transforme cada uma das opções do menu em módulos.

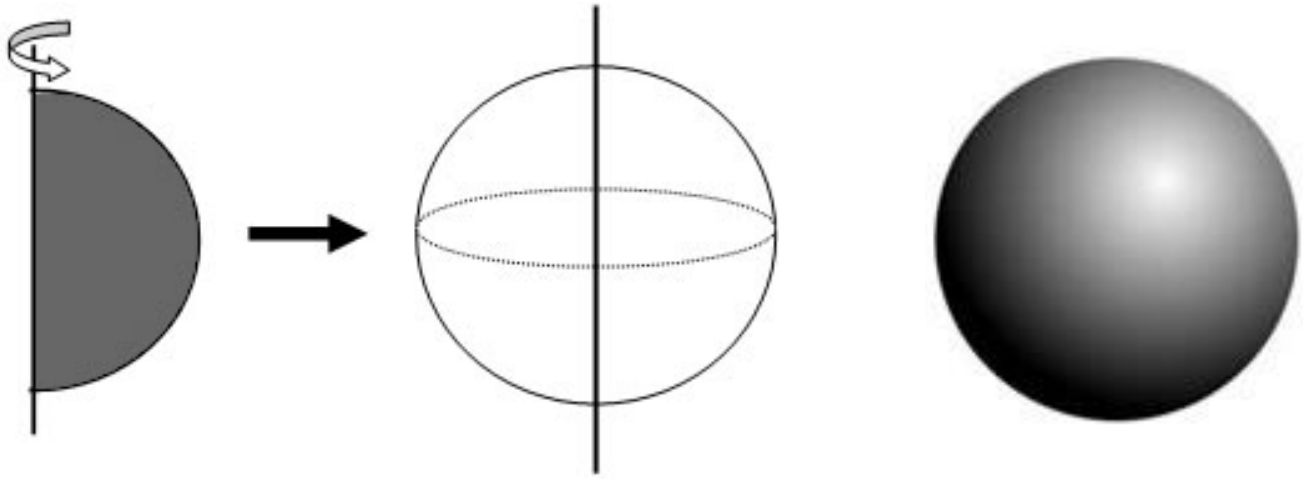
2) Faça um programa que contenha o modulo principal, onde serão informados 3 números que serão passados como parâmetros para um outro modulo que fará a rotação desses 3 números.

Por exemplo, se o módulo receber 5,8 e 9, nesta ordem, os valores de saída serão 8, 9 e 5.

Exercicios

3) Faça uma programa que possua uma módulo que recebe por parâmetro o raio de uma esfera e calcula o seu volume (v = (4 * PI * R^3) / 3).

A esfera surge da revolução de uma semicircunferência. Observe:



Escopo de Variáveis

Módulos são considerados subprogramas. A ideia é que eles sejam o mais independente possível entre si. Neste sentido é interessante que cada módulo possua suas próprias variáveis.

Antes de utilizarmos módulos, as variáveis definidas podias ser acessadas em qualquer parte do programa. Agora podemos ter duas situações: variáveis globais e variáveis locais.

Variáveis Globais

As variáveis globais são declaradas antes de todos os módulos e podem ser manipuladas em qualquer um dos módulos existentes.

Variáveis Locais

As variáveis locais são declaradas no início do módulo ao qual pertencem. Tais variáveis só podem ser acessadas dentro do módulo em que foram declaradas. Dizemos que o escopo é local.

OBRIGADO

nilton.cesar@etec.sp.gov.br