Lógica & Algoritmos

Modularização

Prof. Dr. Joaquim Assunção

CENTRO DE TECNOLOGIA
UFSM
2023

Módulo

- Um módulo é uma unidade de algo (software).
- Em software, é um conjunto de comandos com uma funcionalidade ou um objetivo bem definida.
- Tais componentes devem ser o mais independente possível das outras partes do algoritmo; pois, eles realizam tarefas específicas dentro de alguma solução para o problema.

Modularização

- Um problema complexo pode ser simplificado quando dividido em vários problemas

- Exemplo: Calcular os atrasos e horas trabalhadas e imprimir o total de atrasos, somatório das horas trabalhadas e a média diária
  1. Ler o horário de chegada e saída
  2. Calcular os atrasos
  3. Calcular o total de horas trabalhadas
  4. Calcular a média
  5. Imprimir os resultados

Modularização

```
função nome_da_função_A() {
    // declaração de variáveis
    // sequência de ações
}

função nome_da_função_B() {
    // declaração de variáveis
    // sequência de ações
}
```

Modularização

```
inicio
    //declaração de variáveis
    nome_da_função_A();
    nome_da_função_B();
fim.
```

Componentes do módulo

- Um módulo tem:
- **Interface**: declaração das funcionalidades ofertas, descrição dos dados de entradas e de saída.
- **corpo**: implementação das funcionalidades, comandos que compões o trecho de algoritmo do módulo.

Vantagens

- Manutenção mais simples sem efeitos colaterais no resto do código.
- Independência na elaboração dos módulos.
- Testes e correções dos módulos podem ser feitos separadamente.
- Uniformidade no processo de desenvolvimento de software.
- Reutilização do código (trabalho).
- Legibilidade do código.

Modularização

- Calculadora
  - 4 funções: soma, subtração, multiplicação, divisão

```
função soma() {
    leia(a);
    leia(b);
    escreva(a+b);
}

função subtração() {
    leia(a);
    leia(b);
    escreva(a-b);
}
```

Modularização

```
inicio
      inteiro: opção;
      opção <- 0;
      enquanto(opção <> 3) {
            escreva(”[1] - Adição”);
            escreva(”[2] - Subtração”);
            escreva(“Digite 3 para sair”);
            leia(opção)
            escolha opção {
                  caso 1: soma();
                  caso 2: subtração();
            }
      }
fim.
```

Modularização

- Passagem de parâmetros
  - É possível fazer com que uma função receba parâmetros quando for chamada

```
função nome_da_função(tipo: nome_do_atributo) {
    // ações
}
```

Modularização

```
função soma(inteiro: a, b) {
    escreva(a+b);
}

função subtração(inteiro: a,b) {
    escreva(a-b);
}
```

Modularização

```
inicio
      inteiro: opção;
      opção <- 0;
      enquanto(opção <> 3) {
            escreva(”[1] - Adição”);
            escreva(”[2] - Subtração”);
            escreva(“Digite 3 para sair”);
            leia(opção)
            escreva(“Escreva os valores”);
            leia(a,b)
            escolha opção {
                  caso 1: soma(a,b);
                  caso 2: subtração(a,b);
            }
      }
fim.
```

Modularização

- As funções também podem retornar algum resultado

```
função soma(inteiro: a, b) {
    retorne a+b;
}

função subtração(inteiro: a,b) {
    retorne a-b;
}
```

Modularização

```
inicio
     inteiro: opção, res;
     opção <- 0;
     enquanto(opção <> 3) {
          escreva(”[1] - Adição”);
          escreva(”[2] - Subtração”);
          escreva(“Digite 3 para sair”);
          leia(opção)
          escreva(“Escreva os valores”);
          leia(a,b)
          escolha opção {
               caso 1: res <- soma(a,b);
               caso 2: res <- subtração(a,b);
          }
          escreva(“O resultado é: ”, res);
     }
fim.
```

Modularização

- Exemplos de funções:
  - Verificar se um número é positivo ou negativo
  - Verificar se um número é par ou ímpar
  - Calcular o inverso de um número
  - Retornar o reverso de um número inteiro
  - Calcular o número de dígitos
  - Potenciação