Modularização

Exercício 32:

Construa uma função capaz de receber um número inteiro como parâmetro e retornar se este é ou não um número impar.

Resposta:

```
funcao eh_impar (a: inteiro): logico
inicio
  retorne (a%2<>0)
fimfuncao
```


UNIVASF

Exercício 32:

Resposta alternativa:

```
funcao eh_impar (a: inteiro): caractere
inicio
   se (a%2<>0) entao
      retorne ("O número é impar.")
   senao
      retorne ("O número é par.")
   fimse
fimfuncao
```

Modularização

No caso em que o problema anterior se apresentasse da seguinte forma: Construa uma função que receba um número inteiro e retorne se este é ou não um número impar. Algo mudaria com relação à interpretação?

Sim. Neste caso não estaria especificado se a função receberia o número através de um parâmetro ou se este seria lido através da entrada padrão. Logo, a seguinte solução também seria correta:

```
funcao eh_impar (): logico
var n:inteiro
inicio
   escreva ("Entre com um valor inteiro: ")
   leia (n)
   retorne (n%2<>0)
fimfuncao   //a chamada da função eh_impar deve ser
            //feita da seguinte forma: eh_impar()
```

Exercício 33

Utilizando-se do conceito de modularização construa um algoritmo que resolva o problema de obter as raízes reais de uma equação do segundo grau, caso existam raízes reais.

```
algoritmo "Calcular Raízes"
var
   a, b, c, d: real
   funcao calcular_delta(a:real; b:real; c:real):real
   inicio
      retorne (b^2-4*a*c)
   fimfuncao
inicio
   escreva("Algoritmo que calcula as raízes reais ")
   escreval ("de uma equação do tipo: ax^2+bx+c")
   repita
      escreva ("Entre com o valor de a: ")
      leia (a)
   ate (a<>0)
   escreva ("Entre com o valor de b: ")
   leia (b)
   escreva ("Entre com o valor de c: ")
   leia (c)
   d<-calcular_delta(a,b,c)
   se (d<0) entao
      escreva ("A equação não possui raízes reais.")
   senao
      escreval ("x1 =",(-b+d^0.5)/(2*a))
      escreval ("x2 =",(-b-d^0.5)/(2*a))
   fimse
fimalgoritmo
```

Exercício 34

Construa uma função que receba, como parâmetro, um número inteiro positivo, o qual representa a posição de um determinado termo na série de Fibonacci, a função deve retornar o valor do termo correspondente à posição recebida.

```
funcao fibonacci (p: inteiro):inteiro
var a, b: inteiro
inicio
   a<-0
   b<-1
   enquanto (p>1) faca
        b<-a+b
        a<-b-a
        p<-p-1
   fimenquanto
  retorne a
fimfuncao
```

Modularização

Exercício 35

Construa um algoritmo que seja capaz de efetuar uma multiplicação entre valores naturais quaisquer e também seja capaz de calcular o fatorial de um número natural qualquer. Tanto no cálculo da multiplicação quanto no cálculo do fatorial os únicos operadores aritméticos que podem ser utilizados são os de soma e subtração.

O algoritmo em questão deve possibilitar ao usuário fazer a seleção de qual operação será realizada. As entradas devem ser validadas e o conceito de modularização deve ser aplicado.

UNIVASF