

1 Modularização

À medida que vamos resolvendo problemas mais complexos, o tamanho dos nossos programas vai crescendo, assim, fica difícil acompanhar as funcionalidades dos trechos de programas.

No Pascal e em outras linguagens estruturadas existe a possibilidade de agrupar trechos de programas em procedimentos e funções. Esta técnica de decomposição em unidades funcionais, proveniente da programação estruturada, é conhecida como **modularização**.

Estes trechos devem ser logicamente coerentes, isto é, cada um deve realizar uma função definida. De modo geral a estrutura do módulo (procedimento ou função) é a mesma do programa principal, isto é, tem o cabeçalho e o corpo.

Os procedimentos têm a seguinte estrutura:

```
procedure nome (lista de parametros)
    Área para declarações
begin
    Comandos
end;
```

Vamos estudar um exemplo para entender como são e como funcionam os procedimentos. Este programa foi elaborado para o Turbo Pascal para Windows (TPW).

```
program modulos;

uses windos, wincrt;

var a,b,c: integer;

procedure media(x,y,z: integer);
var
    m: real;
begin
    m := (x + y + z) / 3;
    writeln('Media = ',m:5:2);
end;

procedure troca(var x,y: integer);
var
    temp: integer;
begin
    temp := x; x := y; y := temp;
end;

begin { Programa Principal }
    write('Digite 1º número: '); readln(a);
```

```

write('Digite 2º número: '); readln(b);
write('Digite 3º número: '); readln(c);
media(a,b,c);
troca(a,b);
writeln('Valores de x e y = ',a:5,b:5)
end.

```

Função é um tipo especial de procedimento que permite retornar um valor diretamente. O formato de uma função é mostrado a seguir:

```

function nome (lista de parâmetros):tipo de dado a ser retornado;
    Área para declarações;
begin
    comandos
end;

```

Segue abaixo o programa anterior acrescido de funções.

```

program modulos;

uses windos, winCRT;

var { variáveis globais }
    a,b,c: integer;
    m1: integer;

procedure media(x,y,z: integer);
var
    m: real; { variável local }
begin
    m := (x + y + z) / 3;
    writeln('Media = ',m:5:2);
end;

procedure troca(var x,y: integer);
var
    temp: integer;
begin
    temp := x; x := y; y := temp;
end;

function calculaMaior(x,y: integer): integer;
begin
    if x > y then calculaMaior := x
    else calculaMaior := y;
end;

begin { Programa Principal }
    write('Digite 1º número: '); readln(a);
    write('Digite 2º número: '); readln(b);
    write('Digite 3º número: '); readln(c);
    media(a,b,c);
    troca(a,b);
    writeln('Valores de x e y trocados = ',a:5,b:5);
    m1 := calculaMaior(a,b);

```

```
writeln('Maior nota = ',calculaMaior(m1,c));
end.
```

Exemplo/Exercício 1

Observe a função abaixo:

```
function Palindrome (var s: String): boolean;
var
  i,m,n: integer;
  p: boolean;
begin
  p := true;
  n := length(s);
  while p and (i <= m) do
  begin
    p := (s[i] = s[n-i+1]);
    i := i + 1;
  end;
  palindrome := p;
end;
```

Com relação a esta função vejamos alguns itens:

- Qual o nome da função?
- O que ela faz?
- O que retorna?
- Escreva um programa completo que a utilize.
- Teste seu programa.

Exemplo/Exercício 2

Outro exemplo interessante onde podemos utilizar funções é o casamento de padrões: dadas duas seqüências de caracteres, uma chamada texto, e outra chamada padrão, verificar se existe uma ocorrência do padrão no texto. Caso ocorra uma ocorrência, a função retorna a posição do texto onde ocorre o padrão, caso contrário, retorna 0 (zero).

Uma idéia simples de implementação é percorrer todas as posições possíveis do texto, de se começar o padrão. Para cada uma destas posições, há uma outra estrutura de repetição que verifica se o padrão está começando naquela posição. A função pára assim que encontrar o primeiro padrão, ou até que todas as possibilidades tenham sido testadas. Segue abaixo o código desta função.

Existem outros algoritmos para fazer busca de padrões que são computacionalmente mais eficientes, como o algoritmo de Knuth, Morris e Pratt e o algoritmo de Boyer e Moore.

```
type
  TipoString = String[255];
  TipoPadrao = String[50];

function BuscaPadrao (var texto: TipoString; var padrao: TipoPadrao): integer;
var
  i,j,TamTexto,TamPadrao: integer;
  achou, subsequenciaIgual: boolean;
```

```

begin
  TamTexto := length(texto);
  TamPadrao := length(padrao);
  i := 0;
  achou := false;
  while (not achou) and (i <= TamTexto - TamPadrao) do
  begin
    i := i + 1;
    j := 1;
    subsequenciaIgual := true;
    while (j <= TamPadrao) and (subsequenciaIgual) do
      if (padrao[j] = Texto[i+j-1]) then
        j := j + 1
      else
        subsequenciaIgual := false;
    achou := subsequenciaIgual;
  end;
  if (achou) then BuscaPadrao := i
  else BuscaPadrao := 0;
end;

```

Escreva um programa completo que leia do teclado um texto e o padrão a ser pesquisado, verifique se o padrão se encontra no texto e exiba mensagem informando o usuário do resultado do processamento.

O que mudaria no algoritmo para verificar a ocorrência de uma segunda, terceira, quarta ... ocorrências do padrão?

Exemplo/Exercício 3

Criptografia é a ciência e o estudo da escrita secreta. Um sistema criptográfico é um método secreto de escrita pelo qual um texto legível é transformado em um texto cifrado. O processo de transformação é conhecido como ciframento e a transformação inversa é conhecida como deciframento. Neste exercício iremos trabalhar mais com cadeias de caracteres e apresentar um método simples de criptografia por substituição, a cifra de César. Dados dois alfabetos A e A' , uma função de criptografia por substituição troca um caracter de um texto de A por outro de A' . Naturalmente deve haver uma função inversa para que o receptor da mensagem criptografada possa recuperar o texto original.

Um caso particular do método de substituição é o seguinte: consiste em alfabeto $A = (c_0, c_1, \dots, c_{n-1})$. Considere um inteiro k , $0 \leq k \leq n-1$ (chamado chave). Uma função de criptografia $f_k : A \rightarrow A$ e sua inversa f_k^{-1} (para descriptografar) são dadas a seguir:

$$f_k(c_i) = c_{(i+k) \bmod n} \quad f_k^{-1}(c_i) = c_{(i-k+n) \bmod n}$$

Assim, se o alfabeto é $A = (A, B, C, \dots, X, Y, Z)$ e $k = 3$ a função transforma A em D, B em E, C em F e assim por diante.

Desenvolva dois procedimentos (criptografar e descriptografar) que tenham como parâmetros a frase a cifrar (entrada), a frase a cifrada (retorno) e a chave (entrada). Em seguida crie o programa completo que contenha estes procedimentos e os utilize em frases fornecidas ao programa pelo teclado.

Criação de Units no Object Pascal

Podemos criar uma unidade de código tanto em Pascal como em Object Pascal. Na realidade podemos desenvolver bibliotecas de código (procedimentos / funções) que podem ser utilizadas posteriormente.

Observe o código abaixo:

```
unit UMinha;

interface { seção onde se declara os elementos públicos da unit }
  function fatorial(n:integer):integer;
  function mdc(x,y:integer):integer;

implementation {seção onde se declara os elementos privados da unit }

{ recebe como entrada um valor inteiro e retorna seu fatorial }
function fatorial(n:integer):integer;
var i,f: integer;
begin
  f := 1;
  for i := 1 to n do
    f := f * i;
  fatorial := f;
end;

{ recebe como entrada dois valores inteiros maiores que zero e retorna o mdc entre eles }
function mdc(x,y:integer):integer;
begin
  while (x <> y) do
    if x > y then
      x := x - y
    else
      y := y - x;
  mdc := x;
end;

end.
```

Para usar esta Unit, no projeto fazemos da seguinte forma:

```
program PrjMinha;

{$APPTYPE CONSOLE}

uses
  SysUtils, UMinha in 'UMinha.pas';

var x,y,z: integer;

begin
  repeat
    write('Digite um número inteiro: ');
    readln(x);
    if x < 0 then
      writeln('número invalido, digite um numero maior ou igual a 0');
  until (x >= 0);
  writeln('fat(',x,') = ',fatorial(x));
  repeat
    write('Calcular o MDC entre: ');
    readln(x,y);
```

```
if (x < 0) or (y < 0) then
  writeln('digite números maiores que 0');
until (x > 0) and (y > 0);
writeln('MDC(',x,',',y,') = ',mdc(x,y));
readln { simplesmente dá uma pausa para vermos o resultado }
end.
```

Observe que estamos utilizando comandos do tipo read/write. As aplicações desenvolvidas com este tipo de comunicação com usuário (texto) são criadas utilizando a opção File / New / Console Application. Para criar uma Unit File / New / Unit.

Procure dar nomes sugestivos aos seus arquivos. Por exemplo, a Unit do exemplo acima é UMinha. O projeto que é o ponto de partida para a execução é PrjMinha.

Seja organizado(a). Crie pastas para armazenar cada projeto desenvolvido.

Por enquanto não utilizaremos componentes do Delphi, estes ficam para a parte 3, onde começaremos com os objetos.

Observações.:

- ✓ Para esta lista crie uma Unit para implementar e disponibilizar todas as funções da lista e coloque, no projeto (programa), a devida chamada para todas as funções elaboradas.
- ✓ Não se esqueça de fazer a devida checagem dos dados de entrada. Por exemplo, os termos da seqüência de Fibonacci são do 1º em diante, não faz sentido o termo 0 (zero) ou negativo. Caso o usuário digite uma posição inválida, o programa dever informar ao usuário o erro.

1. Implemente o cálculo de potência: dada uma base e o expoente, retornar a base elevada ao expoente (Exponencial). Por exemplo, $2^3 = 2 \times 2 \times 2 = 8$.

2. Seqüência de Fibonacci

Dá-se o nome de seqüência de Fibonacci a seqüência:

1	1	2	3	5	8	13	21	34	55	...
1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	...

Qual o seu padrão de comportamento? Escreva um subprograma (procedimento ou função?) que receba n (n-ésimo termo da seqüência) e o retorne. Por exemplo: fib(4) = 3, fib(5) = 5

3. Implemente uma função para multiplicar dois números utilizando somas sucessivas. Por exemplo:

- $3 \times 5 = 5 + 5 + 5 = 15$
- $4 \times 10 = 10 + 10 + 10 + 10 = 40$

4. É preciso determinar o maior elemento em um vetor. Desenvolva uma função para esta tarefa. Para esclarecer, analise o seguinte código que utiliza a função solicitada.

```
program Project2;
{$APPTYPE CONSOLE}
uses SysUtils, UModulos;
Const n = 10;

Type TVetor = array [1..n] of integer;

Var Vetor: TVetor;
    i: integer;

begin
  for i := 1 to n do
    begin
      write('Vetor [',i,'] = ');
      readln(vetor[i]);
```

```
end;  
writeln('Maior elemento = ',maior(vetor));  
readln;  
end.
```

5. E para o menor o que muda? Crie esta função também.
6. Crie uma outra função que retorne a posição do maior elemento do vetor. Crie também uma função para retornar o menor elemento do vetor.
7. Em um vetor tenho os nomes das crianças de uma turma e em outro o peso das mesmas crianças, uma amostra é mostrada abaixo:

nomes

Claudia	Denílson	Eduardo	Franciele	Vanessa
---------	----------	---------	-----------	---------

pesos

25.45	27.50	23.12	37.80	40.30
-------	-------	-------	-------	-------

Escreva um programa, que utilize funções já criadas, que exiba:

- o nome da criança mais pesada
- o nome da criança mais leve