Gabarito da AD1 de Programação I

Rio de Janeiro, 18 de março de 2011

1. Funcionamento geral

O sistema proposto na AD1 deve executar a conversão entre números decimais e romanos, de 1 a 2.000.000. Para isso, um formulário deve ser criado:

Figura 1: Formulário principal

O formulário deve conter 2 campos (preenchidos pelo usuário), que receberão os números nos sistemas arábico (campo acima do TLabel 'Decimal') e romano (campo acima do TLabel 'Romano').

Quando o botão *Romano* for pressionado, o número em formato decimal preenchido pelo usuário no campo acima dele será convertido para romano e colocado no campo ao lado, referente a número romano. Comportamento semelhante será observado ao se clicar no botão *Decimal*, preenchendo o campo de número decimal com a conversão do número romano preenchido.

Para a representação de múltiplos de 1000, **foi escolhido colocar o algarismo entre parênteses**. Por exemplo:

Decimal	Romano (padrão)	Romano (parênteses)
30010	$\overline{XXXX}X$	*(XXX)X*

Tabela 1: Exemplo de Conversão

A seguir, há algumas imagens exemplificando o funcionamento desses botões.

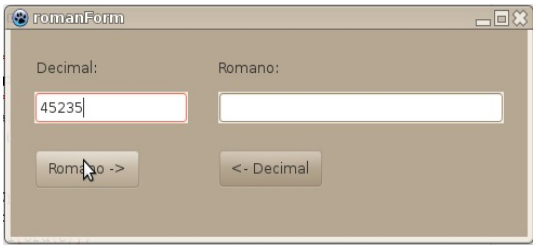

Figura 2: Conversão para Romano (antes)

Figura 3: Conversão para Romano (depois)

Figura 4: Conversão para Decimal (antes)

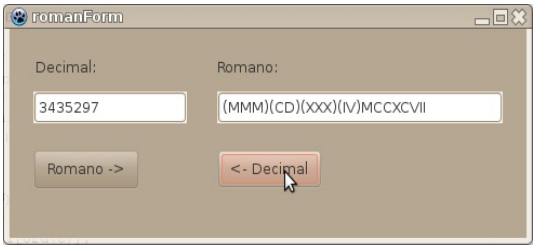

Figura 5: Conversão para Decimal (depois)

2. Observações

Observação 1: Para que o código-fonte seja pequeno, uma tabela de conversão foi utilizada, e foi declarado um *Array* na seção `const` do formulário, contendo todos os algarismos romanos existentes até 2.000.000. Por exemplo:

```
const
  Msymbols: array [0..6] of array [0..9] of string = (
  ('', 'I', 'II', 'III', 'IV', 'V',
       'VI', 'VII', 'VIII', 'IX'),       // units
  ('', 'X', 'XX', 'XXX', 'XL', 'L',
       'LX', 'LXX', 'LXXX', 'XC'),       // tens
  ('', 'C', 'CC', 'CCC', 'CD', 'D',
       'DC', 'DCC', 'DCCC', 'CM'),       // hundreds
  ('', 'M', 'MM', 'MMM', '(IV)', '(V)',
       '(VI)', '(VII)', '(VIII)', '(IX)'),   // thousands
  ('', '(X)', '(XX)', '(XXX)', '(XL)', '(L)',
       '(LX)', '(LXX)', '(LXXX)', '(XC)'),   // ten thousands
  ('', '(C)', '(CC)', '(CCC)', '(CD)', '(D)',
       '(DC)', '(DCC)', '(DCCC)', '(CM)'),   // hundred thousands
  ('', '(M)', '(MM)', '(MMM)', '', '',
       '', '', '', '')                   // millions (...)
```

Observação 2: A linguagem ObjectPascal oferece um conjunto de *functions* de verificação de conversão de tipos, muito úteis para a validação de campos de formulário:

```
Boolean tryStrToInt(String texto, Integer numero);
Boolean tryStrToCurr(String texto, Currency monetario);
Boolean tryStrToFloat(String texto, Float numeroDecimal);
(...)
```

Em particular, a função *tryStrToInt* pode ser usada para validar os números decimais digitados pelo usuário.

Observação 3: Em caso de digitação errônea por parte do usuário, o sistema deve dispor de meios para avisar ao usuário do ocorrido. Isso pode ser feito de diversas formas, dentre elas:

- Usando a *procedure* `ShowMessage`;
- Usando algum elemento do formulário (TLabel ou TEdit).

O código abaixo valida o campo 'Decimal', informando o erro encontrado no TEdit de romanos:

```
Try
   // decimalEdit é o campo decimal da tela
   // romanEdit é o campo romano da tela

   num := StrToInt (decimalEdit.text);
Except
   on E: EConvertError
   do begin
      romanEdit.text := 'Não é número';
      end;
end;

if ( (scnum > 7) or (num >= 2000001) ) then
     romanEdit.text := 'Número muito grande'

(...)
```

Esse tratamento encontra-se ilustrado nas figuras abaixo.

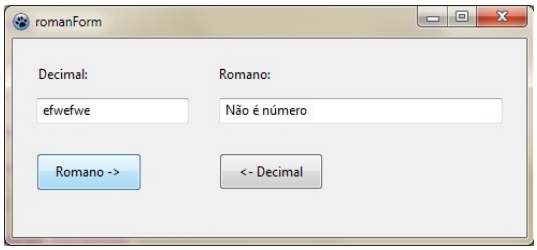

Figura 6: Erro de número inválido

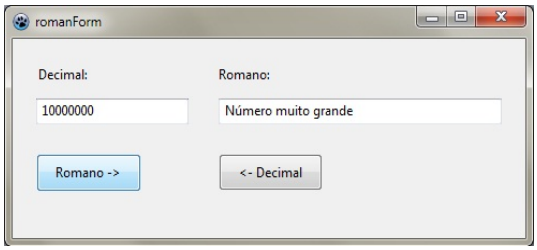

Figura 7: Erro de número muito grande

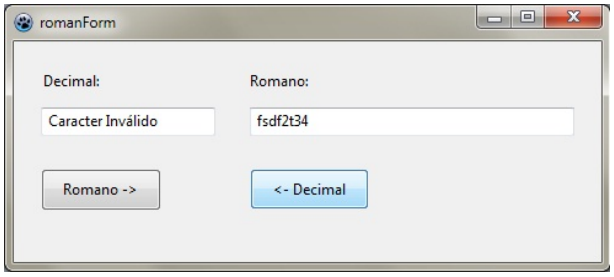

Figura 8: Erro de número romano inválido