Fundação CECIERJ - Vice Presidência de Educação Superior a Distância
Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação I
AD1 1^o semestre de 2014.

GABARITO

---

1. Questão Única

Escreva um programa para converter números decimais reais para o formato binário em ponto flutuante utilizado pelas CPUs modernas - IEEE 754 (32 bits).
(http://pt.wikipedia.org/wiki/IEEE_754).
Algumas sugestões e requerimentos da implementação:

- A sua interface deve permitir-lhe testar a sua implementação, bem como converter em ambos os sentidos. Você pode se basear no seguinte conversor em JavaScript e algoritmos fornecidos abaixo:
  - http://www.h-schmidt.net/FloatConverter/
  - http://orion.lcg.ufrj.br/python/_01%20-%20Programando%20em%20Python%20-%20Sistemas%20de%20Numeracao.pdf

  Basicamente, é necessário implementar as seguintes funções:

  - //retorna a representação equivalente numa base arbitrária (neste trabalho, base 2), de um dado um número real
    function ftoa (num: Real; base: Integer): String;
  - //retorna a representação equivalente no padrão IEEE 754 de um dado número real em ponto flutuante
    function ftoIEEE754 (num: Real): String;
  - //retorna a representação real em ponto flutuante de um número representado numa base arbitrária
    function atof (num: String; base: integer): Real;
  - //retorna a representação real em ponto flutuante de um número representado no padrão IEEE 754.
    function IEEE754tof (num: String): real;

1. Solução

A Figura 1 apresenta uma interface, que suporta as necessidades do problema proposto. Existem dois TEdits editáveis, que são responsáveis pela entrada dos valores que serão convertidos. O primeiro TEdit permite fornecer um número decimal real, ou exibir o resultado da conversão do terceito TEdit (IEEE754tof). O segundo TEdit contém o valor convertido em binário, e o terceiro TEdit serve para fornecer um número na representação IEEE 754, ou exibir a conversão do primeiro TEdit (ftoIEEE754).

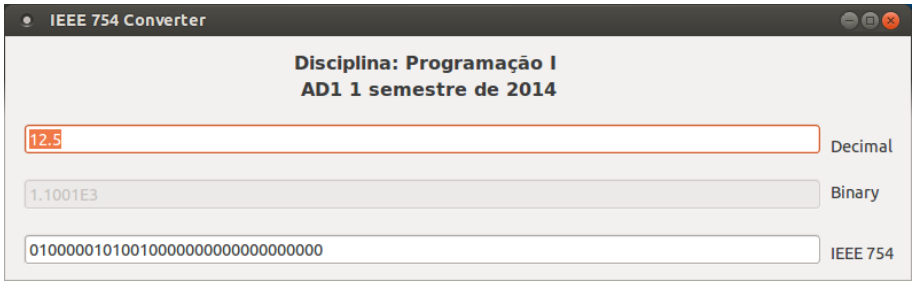

Figura 1: Interface da aplicação convertendo o número 12.5.

Basicamente, é necessário implementar as seguintes funções:

- **function ftoa** (num: Real; base: Integer): String;

  Essa função converte um número decimal real para uma string, que representa o mesmo número, mas utiliza uma base arbitrária. Observando-se a Figura 1, pode-se ver que essa função é responsável por converter o valor do TEdit Decimal em binário e colocar o resultado no segundo TEdit.

  Por exemplo, transformando-se o valor 12.5 para binário (base = 2):

```
12 mod 2 = 0
 6 mod 2 = 0
 3 mod 2 = 1
 1 mod 2 = 1
```

  Assim, a parte inteira = 1100

```
0.5 * 2 = 1.0 -> 1
0.0 * 2 = 0   -> 0
```

  e a mantissa = 10

  Deslocando-se o ponto, para normalizar o resultado: 1100.10 = 1.10010E3, obtém-se o valor convertido em string, que deve ser adicionado ao segundo TEdit.

- **function ftoIEEE754** (num: Real): String;

  Essa função retorna a representação equivalente no padrão IEEE 754 de um dado número real. Observando-se a Figura 1, essa função é responsável por converter o valor do TEdit Decimal em binário e colocar o resultado no terceiro TEdit.

  Essa função recebe o valor decimal do primeiro TEdit, e encontra o equivalente em binário utilizando a função ftoa. Portanto, considerando-se 1.10010E3, o expoente vale ”3”e a mantissa vale ”10010”. Adicionando-se 127 ao expoente:

$$3 + 127 = 130 = 10000010\ (base2).$$

  Logo, o resultado é (sem os espaços): ”0 10000010 10010000000000000000000”

  Para executar o processo inverso, ao realizado pelas duas funções anteriores, é necessário criar o par de funções a seguir.

- **function atof** (num: String; base: integer): Real;

  Retorna a representação real em ponto flutuante de um número representado numa base arbitrária.

  Considerando-se a base = 2, num = ”1.10010E3”, e deslocando-se o ponto, tem-se:

$$1100.10 = 2^3 + 2^2 + 2^{-1} = 12.5\ (base10).$$

- **function IEEE754tof** (num: String): real;

  Retorna um número real, correspondente ao número em ponto flutuante fornecido no padrão IEEE 754.

  Considerando-se a String num = ”0 10000010 10010000000000000000000”,

  sabe-se que o sinal é positivo (bit 32 vale ”0”). O expoente (bits 24-31) vale

$$10000010 = 2^7 + 2^1 = 130\ (base10)$$

  e subtraindo-se 127 (130-127), conclui-se que o expoente é igual a ”3”.

  A mantissa corresponde aos bits 1-23 ”10010000000000000000000”. Adicionando-se o bit invisível 1(.), pois o número está normalizado, obtém-se: ”1.10010000000000000000000”. Logo, a mantissa vale 1.5625 em decimal e, portanto, o valor retornado é:

$$sinal * 2^{expoente} * mantissa = 1 * 2^3 * 1.5625 = 12.5\ (base10).$$