Departamento de Ciência da Computação - IME - USP

MAC2166 Introdução à Computação - Grande área Elétrica

Escola Politécnica - Primeiro Semestre de 2015

Segundo Exercício Programa **Entrega: até 21 de abril de 2015 às 23h55m**

Operações Aritméticas com Números de Ponto Flutuante

"Nothing brings fear to my heart more than a floating point number."
Gerald Jay Sussman

---

1. Objetivos

O objetivo deste segundo exercício-programa é exercitar o uso de recursos de programação vistos nesta primeira parte da disciplina (ou seja, tudo o que está nos capítulos de 1 a 12 na **apostila do curso**). As **únicas construções** da linguagem C que você pode usar em seu programa são as dadas em aula. Acima de tudo, o objetivo de cada exercício, exercício-programa, aula e atividade de MAC2166 é desenvolver o **raciocínio aplicado na formulação e resolução de problemas computacionais** , como está descrito nos **objetivos** de MAC2166.

2. Introdução

2.1 Número de Ponto Flutuante

No **Exercício-Programa 1** vocês implementaram a visualização de uma representação simples de números reais baseada em números inteiros. Neste exercício-programa vamos implementar realmente o conceito de **números de ponto flutuante**.

Como vocês já devem ter visto em classe, em computação lidamos com números finitos. Isso se deve ao fato de que o espaço alocado para a representação de um número é limitado, em geral a algo entre 32 e 64 bits. Para representação de números reais neste contexto usamos a técnica de **ponto flutuante**, igual ao que vemos em calculadoras científicas. Nessa técnica representamos um número real utilizando dois valores: a **mantissa** e o **expoente**. Os dígitos do número real são guardados na mantissa e o expoente registra o valor de uma potência de dez que deve multiplicar a mantissa para que se obtenha o número pretendido. Exemplos:

-10,41576	=>	mantissa: -1041576, expoente: -5	=>	-1041576E-5
0,000123	=>	mantissa: 123, expoente: -6	=>	123E-6
1230	=>	mantissa: 123, expoente: 1	=>	123E1

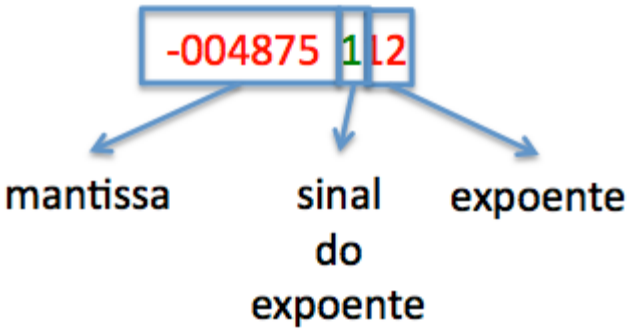
Podemos ver acima que dois números distintos podem ter a mesma mantissa. Neste caso, o que varia é o expoente, que determina o local onde irá a vírgula (ou o "ponto" na representação em inglês). Em outras palavras, meu ponto pode variar (ou "flutuar") dependendo do expoente.

Em uma representação de ponto flutuante, utilizamos um determinado número de dígitos para representar a mantissa e um determinado número de dígitos para a representação do expoente. Isso quer dizer que os números em ponto flutuante têm uma*precisão* finita, ou seja, conseguimos representar apenas alguns dígitos do número.

Em um computador moderno, os números são representados com *números binários*, ou seja, números onde os dígitos são apenas 0 e 1. Em números de ponto flutuante uma palavra de memória (32 ou 64 bits) é dividida em duas partes, em uma delas é armazenada a mantissa e em outra o expoente. No segundo exercício programa vocês deverão implementar a representação de números em ponto flutuante utilizando um número inteiro.

2.2 Representação para o EP2

Números inteiros do tipo `int` em C tem valor entre **-2.147.483.648** e **+2.147.483.647**. Como, apesar de termos até 10 dígitos, o décimo dígito pode ter apenas os valores 0, 1 ou 2, utilizaremos **apenas 9 dígitos** para representar um número de ponto flutuante neste EP. Nesse número, utilizaremos **os dois dígitos menos significativos para o valor do expoente**, **o terceiro dígito para o sinal do expoente** e **os 6 dígitos restantes para a mantissa**. Representaremos expoentes positivos com o dígito 0 e negativos com o dígito 1. Assim, se quisermos representar o número **-4875E-12** nessa codificação em 9 dígitos, ele ficaria da seguinte forma:



Vejamos como ficariam os números que vimos acima nessa representação em ponto flutuante com 9 dígitos (destacada em verde aparece a representação do sinal do expoente):

-10,41576	=>	mantissa: -104157, expoente: -4	=>	-104157E-4	=>	-104157104
0,000123	=>	mantissa: 123, expoente: -6	=>	123E-6	=>	-123106
1230	=>	mantissa: 123, expoente: 1	=>	123E1	=>	123001

Note que no primeiro número nós perdemos dígitos devido à precisão de apenas 6 dígitos da mantissa.

Neste EP, diremos que um número na representação em ponto flutuante com 9 dígitos está **normalizado** se ele possui o maior valor possível no expoente sem perder precisão. Ou seja, um número só está normalizado se **a sua mantissa não possui zeros à direita**OU **se o seu expoente é 99**. Exemplos:

- O número 123048 (= 123E48) é a versão normalizada dos números não normalizados 123000045 (= 123000E45), 12300046 (= 12300E46) e 1230047 (= 1230E47).
  De modo análogo, -123048 é a versão normalizada dos números não normalizados -123000045, -12300046 e -1230047.
- O número 123142 (= 123E-42) é a versão normalizada dos números não normalizados 123000145 (= 123000E-45), 12300144 (= 12300E-44) e 1230143 (= 1230E-43).
  De modo análogo, -123142 é a versão normalizada dos números não normalizados -123000145, -12300144 e -1230143.
- Os números 1230099 (= 1230E99), 12300099 (= 12300E99) e 123000099 (= 123000E99) já estão normalizados.
  De modo análogo, os números -1230099, -12300099 e -123000099 também já estão normalizados.

3. Tarefa

Neste exercício-programa, a sua tarefa será escrever um programa na linguagem C que deve funcionar como uma calculadora para números de ponto-flutuante. O seu programa deve:

1. Pedir, como operando 1, um número de ponto flutuante na representação de 9 dígitos descrita na **Seção 2.2**.
2. Perguntar qual a operação desejada, representada por um número (1: '+', 2: '-', 3: '*', 4: '/', 5: limpar, 6: parar).
3. Executar a operação
    a. Caso seja pedida uma operação aritmética, o programa deve pedir um novo número de ponto flutuante (na representação de 9 dígitos) como operando 2 e executar a operação sobre os operandos 1 e 2, imprimindo o número de ponto flutuante resultante na representação de 9 dígitos **no formato normalizado**. Depois, voltar ao passo 2, desta vez usando o resultado como operando 1.
    b. Caso seja pedida a operação de 'limpeza', o programa deve voltar ao passo 1.
    c. Caso seja pedida a operação de parada, o programa deve terminar.

As mensagens emitidas pelo seu programa devem ser *idênticas* às mensagens mostradas nos exemplos da **Seção 5**.

Importante: Não devem ser feitas verificações nem outras supostas *melhorias* no programa.

Nos passos 1 e 3.a, é esperado que o usuário digite números inteiros de até 9 dígitos e não entre com valores indevidos (como um texto ou um número real). No passo 2, é esperado que o usuário digite um número inteiro entre 1 e 6.

É importante ressaltar que, embora os números resultantes das operações realizadas pelo seu programa devam sempre ser exibidos no formato normalizado, os operandos digitados pelo usuário para as operações podem não estar normalizados. Por exemplo, o usuário pode digitar como operando o número 123000045; mas caso esse seja o valor do resultado de uma operação realizada pelo programa, ele deve ser exibido pelo programa como 123048.

Para corrigir o seu EP, vamos supor que o seu programa obedece *exatamente* o que está especificado neste enunciado. Tudo que fugir da especificação prejudicará a avaliação de seu trabalho.

As únicas construções --comandos, funções, etc-- da linguagem C que você **poderá usar** em seu programa são as ensinadas em aula.

3.1 Tratamento de Exceções

- Caso uma operação solicitada pelo usuário resulte em um número de ponto flutuante que quando normalizado tem expoente menor que -99, você deve transformar o número em zero (0E0), exibir uma mensagem ao usuário avisando sobre a ocorrência de um *transbordamento negativo* (ou ***underflow***) no expoente do número e continuar a execução do programa normalmente.
- Caso uma operação solicitada pelo usuário resulte em um número de ponto flutuante maior que 999999E99, você deve transformar o número em 999999E99, exibir uma mensagem ao usuário avisando sobre a ocorrência de um *transbordamento positivo* (ou ***overflow***) no expoente do número e continuar a execução do programa normalmente.
- Caso uma operação solicitada pelo usuário resulte em um número de ponto flutuante menor que -999999E99, você deve transformar o número em -999999E99, exibir uma mensagem ao usuário avisando sobre a ocorrência de um *transbordamento positivo* (ou ***overflow***) no expoente do número e continuar a execução do programa normalmente.
- Caso o usuário solicite uma divisão por zero, exiba ao usuário uma mensagem de erro e volte ao passo 2 (para pedir uma nova operação), usando o mesmo operando 1 que seria usado na operação anterior (a divisão que resultou em erro).

4. Orientações para a Implementação

4.1 Sobre a Precisão da Mantissa

Na especificação do EP é requisitado que vocês utilizem a precisão de 6 dígitos na mantissa. Porém os números inteiros têm 9 dígitos de precisão completa. Utilize esses dígitos extras para lidar com resultados de operações que resultam em mais de 6 dígitos, reduzindo depois aos 6 mais significativos. Assim, nas iterações vocês devem verificar após cada resultado parcial se o número tem mais de 6 dígitos, dividindo a mantissa por 10 e ajustando o expoente no caso positivo (abaixo mostramos casos de ajuste).

CUIDADO: seus números podem ser positivos OU negativos.

4.2 Implementação da Soma

Para somar dois números em ponto flutuante, você deve primeiro se assegurar de que eles tenham grandezas compatíveis. Para isso, primeiro você pode "denormalizar" o número de maior expoente, decrementando o seu expoente até que ele fique com o mesmo valor do menor expoente ou até que a sua

mantissa fique com 6 dígitos (o que acontecer primeiro). Se a mantissa chegar a 6 dígitos antes do expoente ficar igual ao do menor, então será preciso cortar dígitos da mantissa do número de menor expoente, incrementando o seu expoente até que ele fique igual ao valor atual do maior expoente (e isso pode transformar a mantissa do número em zero se ele for muito pequeno em comparação ao outro número). Depois dessas transformações, você pode fazer a soma normalmente.

Assim, caso não haja precisão suficiente no sistema para a representação da soma, o programa deverá exibir como resposta o número de maior valor absoluto. Eventualmente, a perda de dígitos pode ser parcial, dependendo da diferença entre os expoentes e da quantidade de dígitos na mantissa de cada número.

Exemplo (supondo mantissa com precisão de 6 dígitos):

475E10 + 987E-10 = 475E10
325E8 + 456E3 = 325E8 + 4E3 = 325004E5

(Para entender melhor, expanda os números das mantissas com os zeros, faça a conta e veja quais são os dígitos mais significativos).

É importante verificar se a soma acarretou em uma mantissa de 7 dígitos. Neste caso, devemos eliminar o último dígito da mantissa e ajustar o expoente:

123456E0 + 9E5 = 1023456E0 = 102345E1

4.3 Implementação da Subtração

Basta usar uma transformação no segundo operando para utilizar o mesmo algoritmo que a soma.

4.4 Implementação da Multiplicação

Você deve fazer a operação dígito a dígito na mantissa do operando 2 e ir somando os resultados.

CUIDADO: isso não é trivial, você deve ajustar o expoente a cada passo e talvez precise ajustar a precisão, quando a soma do resultados parciais utilizar mais de 6 dígitos, ou seja, a cada passo devemos fazer uma soma de números de ponto flutuante.

Assim, se formos multiplicar 222E-8 por 432E-1, iremos calcular a soma aos poucos:

I. Soma = 0E0
II. Soma = Soma + (222E-8 * 2E-1) => Soma = 0E0 + 444E-9 => Soma = 444E-9
III. Soma = Soma + (222E-8 * 3E0) => Soma = 444E-9 + 666E-8 => Soma = 7104E-9
IV. Soma = Soma + (222E-8 * 4E1) => Soma = 7104E-9 + 888E-7 => Soma = 95904-9

A cada passo, após fazer o produto, verifique se a mantissa do produto possui mais de 6 dígitos. Se esse for o caso, você deve ajustar o expoente.

Como exemplo, vejamos o produto 222222E6 * 516E2:

I. Soma = 0E0
II. Soma = Soma + (222222E6 * 6E2) => Soma = 0E0 + 1333332E8 => Soma = 0E0 + **133333E9** => Soma = 133333E9
III. Soma = Soma + (222222E6 * 1E3) => Soma = 133333E9 + 222222E09 => Soma = 355555E9
IV. Soma = Soma + (222222E6 * 5E4) => Soma = 355555E09 + 1111110E10 => Soma = 355555E9 + **111111E11** => Soma = 114666E11

Neste caso, foi preciso fazer um ajuste após o produto (e antes de somar) para diminuir a precisão da mantissa para 6 dígitos nos passos I e III. O resto do problema é a reprodução da soma de dois números de ponto flutuante. No exemplo acima, os números ajustados após o produto estão destacados em vermelho.

4.5 Implementação da Divisão

De forma semelhante ao produto, você vai precisar "descascar" a divisão e utilizar o algoritmo da soma. Mas agora o procedimento é diferente. Lembre-se, temos apenas a divisão inteira. Assim vamos dividir o número e os restos sucessivos das divisões até termos zero de resto ou utilizarmos toda a precisão. Para isso, a cada passo multiplicamos a mantissa do resto por 10 e decrementamos o expoente de 1.

Como exemplo, vejamos a divisão 1E0 / 4E0:

I. Soma = 0E0
II. Soma = Soma + ((1/4)E0) => Soma = 0E0 + 0E0 => Soma = 0E0 (com resto = 1E0)
III. Soma = Soma + ((10/4)E-1) => Soma = 0E0 + 2E-1 => Soma = 2E-1 (com resto = 2E0)

IV. Soma = Soma + ((20/4)E-2) => Soma = 2E-1 + 5E-2 => Soma = 25E-2 (com resto = 0)

No caso acima, a divisão dá exata (25E-2) ainda na precisão de nossa aritmética. Mas se isso não acontecer, **precisamos parar na precisão máxima**, ou seja, quando tivermos 6 dígitos significativos na mantissa.

Mais um exemplo, a divisão 1E0 / 6E1:

I. Soma = 0E0
II. Soma = Soma + ((1/6)E-1) => Soma = 0E0 + 0E-1 => Soma = 0E-1 (com resto = 6E0)
III. Soma = Soma + ((10/6)E-2 => Soma = 0E-1 + 1E-2 => Soma = 1E-2 (com resto = 4E0)
IV. Soma = Soma +((40/6)E-3) => Soma = 1E-2 + 6E-3 => Soma = 16E-3 (com resto = 4E0)
V. Soma = Soma +((40/6)E-4) => Soma = 16E-3 + 6E-4 => Soma = 166E-4 (com resto = 4E0)
VI. Soma = Soma +((40/6)E-5) => Soma = 166E-4 + 6E-5 => Soma = 1666E-5 (com resto = 4E0)
VII. Soma = Soma +((40/6)E-6) => Soma = 1666E-5 + 6E-6 => Soma = 16666E-6 (com resto = 4E0)
VIII. Soma = Soma +((40/6)E-7) => Soma = 16666E-6 + 6E-7 => Soma = 166666E-7 (com resto = 4E0)

No exemplo acima, temos uma divisão cujo o resultado é uma dízima. O resultado dessa divisão com a precisão máxima de 6 dígitos para a mantissa é 166666E-7.

4.6 Casos que Requerem Cuidados Especiais

- **Valores zero no expoente e na mantissa:** mantissas zero com expoentes diferentes de zero podem acarretar problemas se vocês não estiverem atentos.
- **Números negativos:** as operações de divisão inteira ('/') e resto de divisão inteira ('%') envolvendo um número inteiro negativo fornecem como resultado um número inteiro negativo também. Isso pode causar erros na extração do expoente e do sinal se vocês não tiverem cuidado.

5. Exemplos de Execução

Nos exemplos, considere que tudo que aparece em **vermelho** foi digitado pelo usuário.

Exemplo 1

Uma sequência envolvendo todos os tipos de operações:

```
MAC2166 - EP2 - Calculadora para Numeros de Ponto
Flutuante



Operando 1...................: 325001



Operacoes disponiveis: 1 (+), 2 (-), 3 (*), 4 (/), 5
(limpar), 6 (parar)

Numero da operacao desejada..: 1



Operando 2...................: 2638104



Resultado (e novo operando 1): 325026102



Operacoes disponiveis: 1 (+), 2 (-), 3 (*), 4 (/), 5
(limpar), 6 (parar)

Numero da operacao desejada..: 3



Operando 2...................: 852103



Resultado (e novo operando 1): 276921102



Operacoes disponiveis: 1 (+), 2 (-), 3 (*), 4 (/), 5
(limpar), 6 (parar)

Numero da operacao desejada..: 2
```

```
Operando 2...................: 456908101



Resultado (e novo operando 1): -429216101



Operacoes disponiveis: 1 (+), 2 (-), 3 (*), 4 (/), 5
(limpar), 6 (parar)

Numero da operacao desejada..: 4



Operando 2...................: 72004



Resultado (e novo operando 1): -596133107



Operacoes disponiveis: 1 (+), 2 (-), 3 (*), 4 (/), 5
(limpar), 6 (parar)

Numero da operacao desejada..: 5



Operando 1...................: 222222006



Operacoes disponiveis: 1 (+), 2 (-), 3 (*), 4 (/), 5
(limpar), 6 (parar)

Numero da operacao desejada..: 3



Operando 2...................: -516002



Resultado (e novo operando 1): -114666011
```

```
Operacoes disponiveis: 1 (+), 2 (-), 3 (*), 4 (/), 5
(limpar), 6 (parar)

Numero da operacao desejada..: 6



Tchau!
```

Exemplo 2

Uma sequência de operações que resulta em *underflow* de expoente:

```
MAC2166 - EP2 - Calculadora para Numeros de Ponto
Flutuante



Operando 1...................: 43210196



Operacoes disponiveis: 1 (+), 2 (-), 3 (*), 4 (/), 5
(limpar), 6 (parar)

Numero da operacao desejada..: 4



Operando 2...................: 1004



Resultado (e novo operando 1): 4321199



Operacoes disponiveis: 1 (+), 2 (-), 3 (*), 4 (/), 5
(limpar), 6 (parar)

Numero da operacao desejada..: 4



Operando 2...................: 10000
```

```
*** AVISO: ocorreu um underflow no expoente. ***

Resultado (e novo operando 1): 0



Operacoes disponiveis: 1 (+), 2 (-), 3 (*), 4 (/), 5
(limpar), 6 (parar)

Numero da operacao desejada..: 2



Operando 2...................: -34517156



Resultado (e novo operando 1): 34517156



Operacoes disponiveis: 1 (+), 2 (-), 3 (*), 4 (/), 5
(limpar), 6 (parar)

Numero da operacao desejada..: 6



Tchau!
```

Note que o Exemplo 2 também ilustra a possibilidade do usuário fornecer operandos em formatos não normalizados: 43210196 (43210E-96) e 10000 (10E0).

Exemplo 3

Uma sequência de operações que resulta em *overflows* de expoente:

```
MAC2166 - EP2 - Calculadora para Numeros de Ponto
Flutuante



Operando 1...................: 999478099
```

```
Operacoes disponiveis: 1 (+), 2 (-), 3 (*), 4 (/), 5
(limpar), 6 (parar)

Numero da operacao desejada..: 1



Operando 2...................: 80000097



*** AVISO: ocorreu um overflow no expoente. ***

Resultado (e novo operando 1): 999999099



Operacoes disponiveis: 1 (+), 2 (-), 3 (*), 4 (/), 5
(limpar), 6 (parar)

Numero da operacao desejada..: 3



Operando 2...................: -1000



Resultado (e novo operando 1): -999999099



Operacoes disponiveis: 1 (+), 2 (-), 3 (*), 4 (/), 5
(limpar), 6 (parar)

Numero da operacao desejada..: 4



Operando 2...................: 2000



Resultado (e novo operando 1): -499999099
```

```
Operacoes disponiveis: 1 (+), 2 (-), 3 (*), 4 (/), 5
(limpar), 6 (parar)

Numero da operacao desejada..: 1



Operando 2...................: -600000099



*** AVISO: ocorreu um overflow no expoente. ***

Resultado (e novo operando 1): -999999099



Operacoes disponiveis: 1 (+), 2 (-), 3 (*), 4 (/), 5
(limpar), 6 (parar)

Numero da operacao desejada..: 6



Tchau!
```

Exemplo 4

Uma sequência de operações envolvendo uma divisão por zero:

```
MAC2166 - EP2 - Calculadora para Numeros de Ponto
Flutuante



Operando 1...................: 1234102



Operacoes disponiveis: 1 (+), 2 (-), 3 (*), 4 (/), 5
(limpar), 6 (parar)

Numero da operacao desejada..: 4



Operando 2...................: 0
```

```
*** ERRO: nao e' possivel dividir por zero. ***



Operacoes disponiveis: 1 (+), 2 (-), 3 (*), 4 (/), 5
(limpar), 6 (parar)

Numero da operacao desejada..: 4



Operando 2...................: 2105



Resultado (e novo operando 1): 617003



Operacoes disponiveis: 1 (+), 2 (-), 3 (*), 4 (/), 5
(limpar), 6 (parar)

Numero da operacao desejada..: 6



Tchau!
```

6. Observações

- Leia as **INFORMAÇÕES SOBRE ENTREGA DE EPs** antes de entregar o seu EP.
- Certifique-se de que o seu programa foi realmente depositado no site verificando se você recebeu um e-mail com a confirmação da entrega.
- Executáveis deste exercício-programa estão disponíveis para donwload **aqui**. Caso você tenha dúvidas sobre qual deve ser o comportamento do seu programa em alguma situação, veja como se comporta o executável.