Conversor de base (continuação)

Paulo José da Silva e Silva

Entrega: 19 de setembro de 2012 (até 24hs)

1 Continuando o conversor de base

No último EP vocês escreveram um conversor simples de base que transforma um número dado em uma base b, $2 \leq b \leq 10$, e apresentavam o número na base 10. O seu trabalho agora sera generalizar esse resultado permitindo bases diferentes de 10 como "destino".

Para isso você deverá modificar a apresentação da resposta e apresentar o número lido não somente na base 10, mas em todas as bases de 2 a 10. Um exemplo de saída do seu programa deve ser:

```
Entre com a base: 7
Entre com um numero na base 7: 65352

O numero em base 2 e 11111110110001.
O numero em base 3 e 211100220.
O numero em base 4 e 3332301.
O numero em base 5 e 1010210.
O numero em base 6 e 203253.
O numero em base 7 e 65352.
O numero em base 8 e 37661.
O numero em base 9 e 24326.
O numero em base 10 e 16305.
```

Para que vocês possam fazer isso precisamos antes discutir dois tópicos: inteiros longos e como fazer a conversão de um número conhecido para uma base qualquer (e não somente para a base 10).

2 Inteiros longos

Como você pode ver no exemplo acima, o número em base 2 ficou longuíssimo 11111110110001. Se você tentarem usar o tipo inteiro que vimos em sala `int`, verão que o programa não vai dar certo, gerando uma outra resposta que parece lixo.

O problema nesse caso é que os inteiros que estão armazenados no computador ocupam espaço na memória que é finita e por isso apenas uma faixa finita de números pode ser representada. Nas máquinas modernas os inteiros usuais ocupam 4 bytes, ou seja, 32 bits. Assim um inteiro usual representável deve ficar dentro da faixa $-2.147.483.648$ até $2.147.483.647$. O número acima é maior que isso.[1] Isso explica qual o problema.

Para permitir números maiores, como o acima, é possível pedir para o C que ele use mais bytes para guardar os inteiros. Isso usa mais memória, fica um pouco mais lento, mas permite o uso de inteiros maiores.

Para isso, basta declarar as suas variáveis como do tipo `long` (que é uma abreviação para "inteiro longo"). Variáveis desse tipo armazenam inteiros que ocupam 8 bytes (64 bits) de memória. Nesse caso a faixa de valores é bem maior, indo de $-9,223,372,036,854,775,808$ até $9,223,372,036,854,775,807$. O resto funciona igual, ou seja você pode fazer com variáveis do tipo `long` o mesmo que pode fazer com variáveis do tipo `int`. Há apenas uma pegadinha. Agora no `scanf` e `printf` no lugar do `%d` você deve usar `%ld`.

3 Como converter para uma base qualquer

Por fim, devemos pensar um pouco como converter um número inteiro $n >= 0$ para uma base qualquer. Imagine que temos n na mão, como o convertemos para a base 10? Isso já sabemos fazer. Basta começar calculando o resto da divisão inteira por 10, o que nos dá o último algarismo e depois dividir por 10 para cortar o último algarismo fora. Continua-se esse processo até o número virar o 0.

E no caso de outra base? Como fazer? Fazemos o mesmo. Se queremos calcular um número na base b devemos iniciar calculando o resto da sua divisão por b. Assim obtemos o último algarismo do número nesse base. Depois dividimos o número por b, jogando fora esse último algarismo, e continuamos o processo.

Como exemplo consideremos o número (em base 10) 16305. Vamos convertê-lo para a base 7. O último algarismo sera `16305 % 7 == 2`. Depois dividindo o número por 7 para jogar fora esse último algarismo, agora conhecido, obtemos o novo número 2329. Continuamos então esse processo até chegar no zero. Veja abaixo o processo completo.

```
16305 % 7 == 2
16305 / 7 == 2329

2329 % 7 == 5
2329 / 7 == 332
```

[1] Apesar de ele representar um número em base 2, ele será guardando em um inteiro que será apresentado na base 10. Então devemos converter o número para a base 2 mas colocá-lo dentro de um inteiro de base 10 já que o computador sempre mostra o inteiro na base 10 quando usamos o `printf`.

```
332 % 7 == 3
332 / 7 == 47

47 % 7 == 5
47 / 7 == 6

6 % 7 == 6
6 / 7 == 0
```

O número na base 7 é o resultado dos restos de divisão, lidos de baixo para cima, ou seja 65352, exatamente o número na base 7 usando no primeiro exemplo, como tinha que ser.

4 Dicas

Algumas sugestões:

1. Pense nas aulas de descascar os números.
2. Você pode aproveitar a potência da base já calculada para calcular a próxima.
3. Considere que a entrada está correta. Ou seja a base original é menor ou igual a 10 e o número digitado nessa base é positivo com dígitos entre 0 e a base - 1.
4. Lembre-se que o `printf` apresenta números na base decimal quando o espaço para ele é reservado usando o `%ld`. Isso quer dizer que mesmo que você esteja agora calculando um número em uma outra base, digamos 7, você deve "colocá-lo" dentro de um número na base 10 para apresentar a resposta.