Material suplementar para ACH-2001

Bases numéricas

(salvo disposição em contrário, este material serve para qualquer base numérica. Os exemplos usam base binária pois é mais útil na disciplina ICC1).

Fábio Nakano, 03.04.2012

Um número inteiro é uma sequência de dígitos. Um número de quatro dígitos tem 4 posições que **devem** ser ocupadas por algarismos. Algarismos são símbolos "gráficos" e representam algum valor.

A quantidade de algarismos que pode ocupar uma posição é chamada **base (b)**. Se a base é dez, então há dez símbolos que podem ocupar uma posição.

Decomposição em dígitos: Um número (N) pode ser decomposto em dígitos (D_i) através da fórmula:

$N = D_{n-1} * b^{n-1} + D_{n-2} * b^{n-2} + D_{n-3} * b^{n-3} + ... + D_1 * b^1 + D_0 * b^0$. Numa notação mais compacta, $N = \sum_{i=0}^{n-1} D_i * b^i$

(note que há n dígitos - os índices vão de zero a n-1. Isso é feito de propósito para evitar confusões entre os índices de D e os expoentes de b)
(note também que na fórmula não fixamos a base, consequentemente, N pode estar em uma base e o somatório pode estar em outra, logo, esta fórmula também serve para converter entre bases).

Em princípio, não há garantia de que "enfileirando" dígitos, somos capazes de representar **todos** os números inteiros dentro de um determinado intervalo. Se você pode acreditar que isso é verdade, então não precisa se ocupar com a demonstração que está na Nota 1.

A questão agora é como usar a fórmula $N = \sum_{i=0}^{n-1} D_i * b^i$ para converter entre bases numéricas.

Devido a maneira como matemática nos é ensinada, somos muito mais eficientes se N for a representação decimal do número. (Da maneira como um computador digital é construído, ele é muito mais eficiente se N for a representação binária do número).

Para converter de decimal para outra base, precisamos calcular os D_i .

Pela a fórmula $N = D_{n-1} * b^{n-1} + D_{n-2} * b^{n-2} + D_{n-3} * b^{n-3} + ... + D_1 * b^1 + D_0 * b^0$, notamos que todas as parcelas exceto a última são divisíveis por b, logo, se dividirmos N por b, o resto da divisão será D_0 . O quociente, por sua vez, será $Q = D_{n-1} * b^{n-2} + D_{n-2} * b^{n-3} + D_{n-3} * b^{n-4} + ... + D_1 * b^0$. Se dividirmos Q por b, o resto será D_1 . Se continuarmos dividindo sucessivamente os quocientes resultantes por b, os restos serão os D_i . Note que o último resto corresponde a D_{n-1} . Este é o método das divisões sucessivas. Código em Java na Nota 2, exemplo abaixo:

Converter 1432 para a base 2 por divisões sucessivas:

N	quociente	resto
1432	716	0
716	358	0

358	179	0
179	89	1
89	44	1
44	22	0
22	11	0
11	5	1
5	2	1
2	1	0
1	0	1

resultado: $(1432)_{10}=(10110011000)_2$

há outros métodos: Tome $N=D_{n-1}*b^{n-1}+D_{n-2}*b^{n-2}+D_{n-3}*b^{n-3}+....+D_1*b^1+D_0*b^0$, note que todas as parcelas tem que ser **não negativas**. Se descobrirmos (chutar até acertar) qual o maior valor de D_{n-1} , digamos $E\,tq.\,E*b^{n-1}\leq N$, então D_{n-1} **tem que ser** *E* pois se escolhermos $D_{n-1}<E$, então não conseguiremos converter o restante do número (pois esse restante do número terá n dígitos e não n-1, como necessário). Se escolhermos $D_{n-1}>E$ então alguma parcela tem que ser negativa, o que não é permitido.

O problema é chutar o valor de *E*. Por outro lado se a base for binária, D_i tem que valer zero ou um, então fica simples: se conseguirmos subtrair b^{n-1} , então D_i tem que ser um, senão D_i tem que ser zero. Este é o "método das subtrações sucessivas", que requer uma tabela de potências da base. Código em Java na Nota 3, exemplo abaixo.

N	subtraendo	resultado	Dígito
1432	1024	408	1
408	512	-	0
408	256	152	1
152	128	24	1
24	64	-	0
24	32	-	0
24	16	8	1
8	8	0	1
0	4	0	0
0	2	0	0
0	1	0	0

resultado: $(1432)_{10}=(10110011000)_2$

Conversão entre bases que são potências umas das outras.

Isto é especialmente útil em computação, quando tratamos bases 2, 8 (octal) e 16 (hexadecimal).

Lembrando - os algarismos na base binária são zero ou um. na base octal podemos usar

quaisquer 8 símbolos - em geral usam-se os números de zero a sete. Em hexadecimal, podemos usar quaisquer 16 símbolos e em geral usam-se os números de zero a nove e as letras de 'A' a 'F'. Essas letras correspondem (em decimal) a 10, 11, 12, 13, 14 e 15, que são os números que **tem que ser** representados por um único símbolo em hexadecimal, mas não tem tal representação em decimal. A consequencia disso é a seg. tabela de conversão:

decimal	binário	octal	hexadecimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

obs.: Se vc seguir as linhas de uma coluna, os números serão incrementados de 1 em 1. No decimal, chegando ao 9, se somar 1, resulta em zero e “vai um”. Isto se repete para as outras bases. Em binário, chegando ao 1, se somar 1 resulta em zero e vai um. Se chegar ao 11 e somar um, resulta em 0 e vai um, que resulta em 0 e vai um. No octal, chegando ao 7 (que é o último número de um dígito), se somar 1 resulta em zero e vai um.

IMPORTANTE!!

Na tabela pode-se notar que precisamos de três dígitos binários para representar um octal, e precisamos de 4 para representar um hexadecimal. Isso não é coincidência: se transformarmos cada grupo de três digitos binários em um único símbolo, resultam oito símbolos. A representação numérica com oito símbolos é a octal. Da mesma forma, 4 dígitos binários em um único símbolo resultam 16 símbolos e a representação correspondente é a hexadecimal.

Nota 1

Questão: “Enfileirando” dígitos somos capazes de representar todos os números inteiros dentro de um certo intervalo.

Demonstração por indução:

Hipótese: $N=\sum_{i=0}^{n-1} D_i * b^i$ representa **todos** os inteiros entre zero e b^n-1 .

Base: Um número de um dígito representa todos os inteiros de zero a b-1.

Prova da base: um dígito pode conter um de b símbolos. Um deles representa o zero, os

restantes, b-1 podem representar inteiros consecutivos até b-1. Portanto a base está provada.

Passo: Tome um número de n dígitos. Como a base é verdadeira, podemos dizer que ele representa todos os inteiros de zero a b^n-1 . Então precisamos demonstrar que com um número de n+1 dígitos podemos representar $b^{n+1}-1$ inteiros (não podemos aplicar a hipótese diretamente sobre o número de n+1 dígitos pois fazendo isso estaremos assumindo que o que queremos demonstrar é verdadeiro e terminaremos por não demonstrar nada).

Com o número de n dígitos representamos o zero e outros b^n-1 números, ou seja, representamos b^n números distintos. Acrescentando um dígito ao número, obtemos um número de n+1 dígitos.

Para cada símbolo no novo dígito, podemos representar b^n números, e todas essas representações são distintas, ou seja, podemos representar $b*b^n$ números. Separando um desses para representar o zero, obtemos $b*b^n-1=b^{n+1}-1$ números, cqd.

(note que não se diz nada sobre onde o dígito deve ser acrescentado, ou sobre a ordem dos números)

Nota 2

```
int b=5;
int N=1234; // o número que vc quer converter
do {
        System.out.println (N%b + " ");
        N=N/b;
} while (N>0);
```

(obs.: é muito mais cômodo codificar esse método usando do..while - tente fazer com while... e teste para N=0)

Nota 3

```
int potenciasDe2[]={1,2,4,8,16,32,64,128,256,512,1024};
int N=1234; // o número que vc quer converter
int digito=10;
do {
        if ((N-potenciasDe2[digito])>=0) {
                System.out.println ("Um");
                N-=potenciasDe2[digito];
        }
        else System.out.println ("Zero");
        digito--;
} while (digito>=0);
```