Tallsystemer

Heltall oppgis vanligvis i det desimale tallsystemet, også kalt 10-tallssystemet.

Eksempel

Gitt tallet 3794. Dette kan skrives slik:

$3 \cdot 1000 + 7 \cdot 100 + 9 \cdot 10 + 4 = 3 \cdot 10^3 + 7 \cdot 10^2 + 9 \cdot 10^1 + 4 \cdot 10^0$

Tallet 10 kalles for *grunntallet* i det desimale tallsystem.

Alle hele tall større enn 1 kan være grunntall i et tallsystem.

Ethvert tall kan konverteres til et annet tallsystem med et annet grunntall, f.eks. grunntall = 12.

Da finnes det tall x, y, z og u som alle er fra 0 til 11 slik at

$3794_{10} = x \cdot 12^3 + y \cdot 12^2 + z \cdot 12^1 + u \cdot 12^0 = xyzu_{12}$

Tallene x, y, z og u er gitt ved x = 2, y = 2, z = 4 og u = 2.

Dermed blir $3794_{10} = 2242_{12}$.

Hvordan vi finner x, y, z og u kommer vi straks til.

I databehandlingen brukes stort sett grunntallene 2, 8, 10 og 16.

Grunntall	Tallsystem	Sifre	Eksempel
2	binært	0, 1	10010_2
8	oktalt	1, 2, 3, ... 6, 7	176_8
10	desimalt	0, 1, 2, ... 8, 9	1289_{10}
16	heksadesimalt	0, 1, 2, ... 8, 9, A, B, C, D, E, F	$A2F3_{16}$

Brukes flere tallsystemer i samme tekst er det vanlig å oppgi tallets grunntall som indeks.

I Java kan en oppgi tall på binær form ved å sette 0b foran tallet, oktal form med 0 foran tallet og heksadesimal form med 0x foran tallet:

```
public class Talltest
{
  public static void main(String ... args)
  {
    int a = 1234;           // 1234
    int b = 0b1001001001;   // 585
    int c = 0137;           // 95
    int d = 0xAB7D;         // 43901

    System.out.println( a +"\n" + b + "\n" + c + "\n" + d);
  }
}
```

Hvis grunntallet vi velger er større enn 10, brukes bokstavene i alfabetet til å representere tallene over 10:

A = 10, B = 11, C = 12, D = 13 osv.

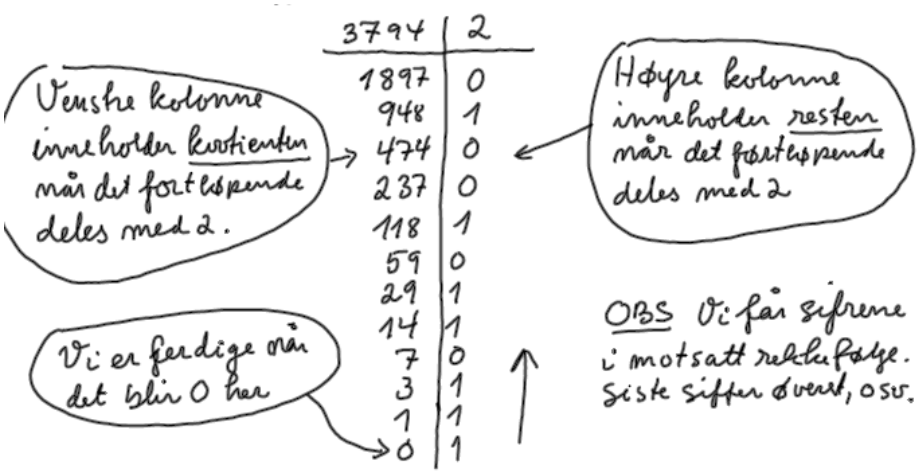
Kjør programmet selv og test ut med forskjellige grunntall!

Konvertering mellom tallsystemer

Fra desimal til binær, oktalt eller hexadesimalt.

Vi viser teknikken ved hjelp av et eksempel. Metoden blir tilsvarende for de andre tallsystemene.

La a = 3794_{10} Vi setter opp følgende skjema:



Svar: $3794_{10} = 111011010010_2$

Generell teknikk.

Gitt et grunntall g > 1 og et positivt heltall a. Da kan a skrives entydig på formen:

$a = s_n g^n + s_{n-1} g^{n-1} + s_{n-2} g^{n-2} + \ldots\ldots + s_2 g^2 + s_1 g^1 + s_0 g^0$

der $0 \leq s_i < g$ for $i = 0, 1, 2, 3, \ldots, n$

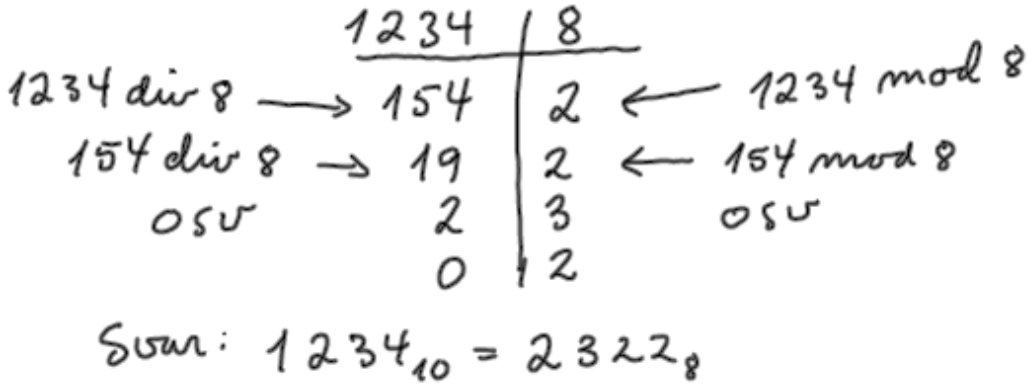
Eksempel

$3794_{10} = 7 \cdot 8^3 + 3 \cdot 8^2 + 2 \cdot 8^1 + 2 \cdot 8^0 = 7322_8$

Vi finner siste siffer ved s_0 = a mod g. Så setter vi a = a div g.
Dermed neste siffer s1 = a mod g. osv.

Eksempel

La $a = 1234_{10}$ og g = 8:



Algoritmen er slik:

La g være grunntallet.

a mod g gir siste binære siffer.

Så setter vi a lik a div g. Dermed vil a mod g neste gang gi nest siste siffer.

Fortsetter på samme måte til vi har fått alle sifrene.

Følgende javakode bruker denne teknikken for å konvertere et tall i 10-tallssystemet til et tall i det tallsystemet vi selv måtte ønske ut fra grunntallet vi velger:

```
import javax.swing.JOptionPane;

public class Tallkonvertering
{
  public static void main(String ... args)
  {
    String tallA = JOptionPane.showInputDialog("Skriv inn et tall" ) ;
    int a = Integer.parseInt( tallA);

    int g = Integer.parseInt( JOptionPane.showInputDialog("Skriv inn et grunntall" ));

    String tallet = "";

    while( a > 0)
    {
      int siffer = a % g;

      if( siffer >= 10 )
         tallet = (char)( 'A' + siffer - 10 ) + tallet;
      else
         tallet = siffer + tallet;

      a = a / g;
    }

    String utskrift = tallA + " = " + tallet + "\ni " + g + "-tallsystemet";
    JOptionPane.showMessageDialog( null, utskrift );
  }
}
```

Fra binært tall til oktale tall

Gitt a = 1101101110010110_2

Hvis vi skal finne a som oktalt tall grupperer vi tre og tre siffer fra høyre mot venstre:

$a = 1\,|\,101\,|\,101\,|\,110\,|\,010\,|\,110_2$

$1 \quad 5 \quad 5 \quad 6 \quad 2 \quad 6$

Hver gruppe på tre konverteres til et oktalt siffer etter følgende regel:

binært	000	001	010	011	100	101	110	111
oktalt	0	1	2	3	4	5	6	7

Dermed blir $1101101110010110_2 = 155626_8$.

OBS! Hvis det er færre enn 3 siffer i gruppen lengst til venstre, kan vi legge til en eller to 0-er slik at det blir tre siffer.

Eksempel

$a = 10101110_2 = 10|101|110 = 010|101|110 = 256_8$

Fra binært tall til hexadesimale tall

La a = 101101110010110_2

Grupper fire og fire binære siffer fra høyre mot venstre:

a = 101|1011|1001|0110

Gruppen lengst til venstre har kun tre siffer. Da kan vi legge på en ekstra 0 forrest slik at vi får 0101. Sifrene i hver gruppe konvergeres til et heksadesimalt siffer etter følgende tabell:

binært	0000	0001	0010	0011	0100	0101	0110	0111
heksadesimalt	0	1	2	3	4	5	6	7

1000	1001	1010	1011	1100	1101	1110	1111
8	9	A	B	C	D	E	F

Dette gir tallet

a = 101|1011|1001|0110

5 B 9 6

$101101110010110_2 = 5B96_{16}$

Fra binært tall til desimal tall

La a = 101101110010110_2. Hvert siffer representerer en potens av 2. Det kan settes opp slik:

a =	1	0	1	1	0	1	1	1	0	0	1	0	1	1	0
	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

Her kan vi finne a ved å summere potensene som hører til 1-sifrene:

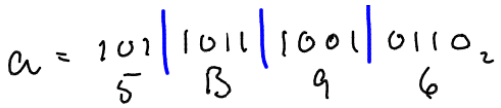
$a = 2^{14} + 2^{12} + 2^{11} + 2^9 + 2^8 + 2^7 + 2^4 + 2^2 + 2^1$

a = 16384 + 4096 + 2048 + 512 + 256 + 128 + 16 + 4 + 2

$a = 23446_{10}$

En enklere metode:

Vi kan først konvertere a til et heksadesimalt tall:



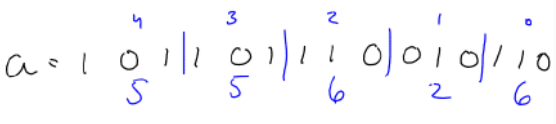
$a = 5B96_{16}$

$a = 5 \cdot 16^3 + B \cdot 16^2 + 9 \cdot 16 + 6 = 5 \cdot 16^3 + 11 \cdot 16^2 + 9 \cdot 16 + 6$

$= 16(5 \cdot 16^2 + 11 \cdot 16 + 9) + 6 = 16(16(5 \cdot 16 + 11) + 9) + 6$

$= 16(16 \cdot 91 + 9) + 6 = 16 \cdot 1465 + 6 = 23446_{10}$

NB! Vi kunne også gått via det tilsvarende oktale tallet:



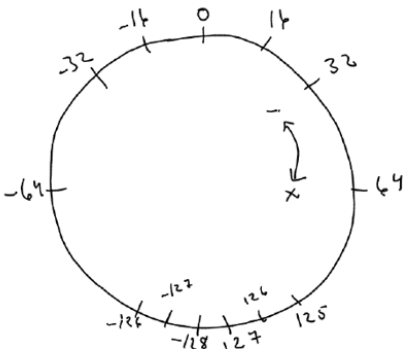
$a = 55626_8 = 5 \cdot 8^4 + 5 \cdot 8^3 + 6 \cdot 8^2 + 2 \cdot 8 + 6 = 23446_{10}$

2-komplement, se notatet:

http://www.iu.hio.no/~evav/DM/to-komplement18.pdf

Tallsirkelen

Vi bruker her 8 biter som fast bitformat:



Hva blir 124 + 5?

I vanlig regning blir det 130. Men finner vi 125 på sirkelen og beveger oss 5 enheter med klokka kommer vi til -126!

Dermed blir 25 + 5 = 126 når vi har 8 som fast bitformat.

Vi finner -127 – 2 ved å gå to enheter mot klokka. Da kommer vi til 127. Dermed blir -127 – 2 = 127.

Dette kan vi også finne ved binær addisjon:

$$-127_{10} = 10000001_2$$
$$-2_{10} = 11111110_2$$
$$+ \quad = \cancel{1}0\,1111111_2 = 127$$

<u>Shift-operatorer:</u>

Prøv shift-operatorene Lsh og Rsh og se hvordan bit'ene i tallets binære representasjon flytter seg til hhv. venstre og høyre. For hver plass bit'ene flyttes mot venstre dobles tallet og for hver plass bit'ene flyttes mot høyre halveres tallet.

Bruk Windows-kalkulatoren til å teste det ut:

Shift-operatorer i Java:

Følgende program demonstrerer shift-operatorene i Java:

```
import javax.swing.JOptionPane;
import javax.swing.JTextArea;

public class Shift
{
  public static  void main(String ... args)
  {
    int n = lesTall("Skriv inn et heltall, avslutt med -1");

    while( n != -1 )
    {
      int shift = lesTall("Skriv inn antall shift");

      int vShift = n << shift;
      int hShift = n >> shift;

      String utskrift =
      "Innlest tall:\t" + n + "\tBinært:   " + Integer.toBinaryString( n ) +
      "\nVenstre Shift:\t" +  vShift + "\tBinært:   " + toBinaryString( n ) +
      "    ->    " + toBinaryString( vShift) +
      "\nHøyre Shift:\t" + hShift  + "\tBinært:   " + toBinaryString( n ) +
      "    ->    " + toBinaryString( hShift);

      skrivUt( utskrift );
      n = lesTall("Skriv inn et heltall, avslutt med -1");
    }
  }
```

Hele programmet kan lastes ned fra.

http://www.iu.hio.no/~evav/DM/Programmer/Shift.java

2-komplement, se notatet:

http://www.iu.hio.no/~evav/DM/to-komplement18.pdf