POPUP Föreläsning 8

Flyttalsaritmetik och aritmetik på stora heltal

Torbjörn Granlund, tege@nada.kth.se, rum 1446

2008-11-03

DEL 1: Flyttal

Avrundning

```
double x = 0.29;
int y = (int) (100.0 * x);
printf("\%d\n", y);
```

Avrundning

```
double x = 0.29;
int y = (int) (100.0 * x);
printf("\%d\n", y);
```

Resultat: 28

Likhetsjämförelse

```
for (double x = 0.0;  x != 1.0;  x += 0.1)
   printf ("hello %.20lf\n", x);
```

Likhetsjämförelse

```
for (double x = 0.0;  x != 1.0;  x += 0.1)
   printf ("hello %.20lf\n", x);
```

Loopen terminierar inte!

Flyttalsrepresentation (1)

$s \times m \times 2^e, \quad s = \text{tecken}, \quad m = \text{mantissa}, \quad e = \text{exponent}$

Flyttalsrepresentation (1)

$s \times m \times 2^e$, s = tecken, m = mantissa, e = exponent

IEEE Enkel prec.:	1 teckenbit, 24 bitars mantissa, 8 bitars exponent
IEEE Dubbel prec.:	1 teckenbit, 53 bitars mantissa, 11 bitars exponent

Flyttalsrepresentation (1)

$s \times m \times 2^e$, s = tecken, m = mantissa, e = exponent

IEEE Enkel prec.:	1 teckenbit,	24 bitars mantissa,	8 bitars exponent
IEEE Dubbel prec.:	1 teckenbit,	53 bitars mantissa,	11 bitars exponent

Mantissan är *normaliserad*.

Mest signifikanta mantissabiten är *implicit*.

Flyttalsrepresentation (2)

s	1	mantissa	exp

Normalt ser vi mantissan som ett normaliserat *flyttal*:

$$x = (1 + 0.mantissa) * 2^{exp-1023}$$

Flyttalsrepresentation (2)

s	1	mantissa	exp

Normalt ser vi mantissan som ett normaliserat *flyttal*:

$$x = (1 + 0.mantissa) * 2^{exp-1023}$$

Men det kan vara praktiskt att se den som *heltal*:

$$x = (2^{52} + mantissa) * 2^{exp-1023-52}$$

Flyttalsrepresentation (3)

Vissa kvantiteter har speciell representation:

$+0$, -0
$+\infty, -\infty$
$+NaN$ Not-a-Number

Dessa skapas med reserverade värden på exp-fältet.

Avrundning

```
double x = 0.29;
int y = (int) (100.0 * x);
printf("%d\n", y);
```
Resultat: 28

Så här lagras 0.29 i IEEE double:

0.28999999999999998002
man = 1.15999999999999992006 exp=-2
man = $0x128f5c28f5c28f \cdot 2^{-54}$

Likhetsjämförelse

Vi återvänder till den icke-terminierande loopen:

```
for (double x = 0.0;  x != 1.0;  x += 0.1)
   printf ("hello %.20lf\n", x);
```

Vilket värde kommer x nå?

Hur loopar vi då...? (1)

Byta ”=” mot ”<=”?

```
for (double x = 0.0;  x <= 1.0;  x += 0.1)
   printf ("hello %.20lf\n", x);
```

12 varv, 0.0 till 1.09999999999999986677.

Hur loopar vi då...? (1)

Byta ”=” mot ”<=”?

```
for (double x = 0.0;  x <= 1.0;  x += 0.1)
   printf ("hello %.20lf\n", x);
```

12 varv, 0.0 till 1.09999999999999986677.

```
for (double x = 0.0;  x <= 2.0;  x += 0.1)
   printf ("hello %.20lf\n", x);
```

21 varv, 0.00000000000000044409.

```
 0:  0.00000000000000000000
 1:  0.10000000000000000555
 2:  0.20000000000000001110
 3:  0.30000000000000004441
 4:  0.40000000000000002220
 5:  0.50000000000000000000
 6:  0.59999999999999997780
 7:  0.69999999999999995559
 8:  0.79999999999999993339
 9:  0.89999999999999991118
10:  0.99999999999999988898
11:  1.09999999999999986677
12:  1.19999999999999995559
13:  1.30000000000000004441
14:  1.40000000000000013323
15:  1.50000000000000022204
16:  1.60000000000000031086
17:  1.70000000000000039968
18:  1.80000000000000048850
19:  1.90000000000000057732
20:  2.00000000000000044409
```

Hur loopar vi då...? (3)

Alternativ 1, avrundningstolerant:

```
for (double x = 0.0;  x < 1.0 + eps;  x += 0.1)
   printf ("hello %.20lf\n", x);
```

Hur loopar vi då...? (3)

Alternativ 1, avrundningstolerant:

```
for (double x = 0.0;  x < 1.0 + eps;  x += 0.1)
   printf ("hello %.20lf\n", x);
```

Alternativ 2, enumerera:

```
for (double x = 0.0;  x != 10.0;  x += 1.0)
   printf ("hello %.20lf\n", x * 0.1);
```

Beteeerndet hos flyttalsaritmetik kan vara svårförutsägbar.

Men aritmetiken är helt determininisiskt, och kan förstås i termer av heltalsaritmetik.

"Real programmers don't use floating point"

DEL 2: Heltal

Representation

Ett heltal X kan skrivas i någon bas B som:

$$X = x_{n-1}B^{n-1} + \cdots + x_2B^2 + x_1B + x_0$$

där man ofta normaliserar så att $0 \leq x_i < B$.

Talbasval

- Maskinens nativa, $B = 2^w$?

Talbasval

- Maskinens nativa, $B = 2^w$?
- Halva bredden, $B = 2^{w/2}$?

Talbasval

- Maskinens nativa, $B = 2^w$?
- Halva bredden, $B = 2^{w/2}$?
- 10-baserat, $B = 10^t < 2^w$?

Talbasval

- Maskinens nativa, $B = 2^w$?
- Halva bredden, $B = 2^{w/2}$?
- 10-baserat, $B = 10^t < 2^w$?
- 10-baserat, $B = (10^t)^2 < 2^w$?

Division med 10^9 resp 10^4

```
rax = word dividend
shr  9, rax                          rax = rax / 2^9
mov  19342813113834067, rdx          rdx = ceil(2^84 / 10^9)
mul  rdx                             rdx = floor(rax * rdx / 2^64)
shr  11, rdx                         rdx = rdx / 2^11

eax = word dividend
mov  3518437209, edx                 eax = ceil(2^45 / 10000)
mul  edx                             edx = floor(eax * edx / 2^32)
shr  13, edx                         edx = edx / 2^13
```

(Koden är fraan GCC.)

Exponentiering höger-till-vänster

```
pow(x, e)
1   z ← 1
2   while e ≠ 0
          do
3             if e mod 2 = 1
                 then z ← z · x
4   x ← x · x
5   e ← ⌊e/2⌋
6   return z
```

Exponentiering vänster-till-höger

$pow(x, e)$

1 **if** $e = 0$
  then return 1
2 $z \leftarrow pow(x, \lfloor e/2 \rfloor)$
3 $z \leftarrow z^2$
4 **if** $e \bmod 2 = 1$
  then $z \leftarrow z \cdot x$
5 **return** z