

# T-106.3101 Ohjelmoinnin jatkokurssi T2 (6 op)

## Luento 7: Projektityön käytännön järjestelyistä. Yksittäisten bittien käsittely.

Sami Liedes

21. helmikuuta 2008  
(päivitetty 21. helmikuuta 2008)

## Projektityön käytännön järjestelyistä 1/3

- ▶ Projekti 3-4 hengen ryhmissä
- ▶ Alle 3 hengen ryhmiä ei hyväksytä
- ▶ Pakollinen osasuoritus, painoarvo arvosanassa 62,5 %
- ▶ *su 16.3.* Ryhmien ilmoittautuminen ja projektisuunnitelman palautus kurssisivuille tulevalla [www-lomakkeella](#)
- ▶ *su 6.4.* Projektityön 1. iteraation palautus
- ▶ *su 4.5.* Projektityön 2. iteraation palautus

## Projektityön käytännön järjestelyistä 2/3

- ▶ Jokainen ryhmän jäsen toteuttaa yhden helpon muutoksen ja yhden vaativamman muutoksen, aiheet julkaistaan pian
- ▶ Esimerkki helposta muutoksesta: Lisää komentoriviparametri, jolla voi valita muun portin numeron kuin 6666
- ▶ Esimerkkejä vaativammasta muutoksesta
  - ▶ Toteuta high score -lista
  - ▶ Toteuta undo
- ▶ 1. kierroksella lisäksi yksi yhteinen testaustehtävä, ohjeet julkaistaan pian

## Projektityön käytännön järjestelyistä 3/3

- ▶ Projektia varten ryhmälle annetaan ilmoittautumisen jälkeen tunnukset projektia varten valmiiksi perustettuun *svn*-versionhallintajärjestelmän repositoryyn
- ▶ Palautus tapahtuu yksinkertaisesti siten, että koodi on deadlineen aikaan repositoryssa
- ▶ Versionhallintajärjestelmistä tarkemmin to 28.2. luennolla
- ▶ Tarkemmat ohjeet myöhemmin...

## Projektisuunnitelma

- ▶ Yleisluontoinen suunnitelma, koodiin ei tarvitse vielä suunnitteluvaiheessa paneutua
  - ▶ Kuka toteuttaa minkäkin lisäyksen
  - ▶ Paljonko näihin arvioidaan menevän aikaa
  - ▶ yms.
- ▶ Arvostelu: hyväksytty/hylätty
- ▶ Suunnitelmasta saa myös poiketa myöhemmin kurssilla
- ▶ Tarkempi ohjeistus vaatimuksista julkaistaan pian, mutta vaatimustaso suunnitelman suhteen ei tosiaan ole kovin korkealla

## Binääriluvuista 1/2

- ▶ Toivottavasti tuttua asiaa matematiikankursseilta
- ▶ Luonnollinen tietokoneille – sähkövirta joko kulkee tai ei kulje, kaksi vaihtoehtoa
- ▶ 10-järjestelmä → 2-järjestelmä (binääri)

$$\begin{aligned}180 &= 128 + 32 + 16 + 4 \\&= 1 \times 128 + 0 \times 64 + 1 \times 32 + 1 \times 16 + 0 \times 8 + \\&\quad 1 \times 4 + 0 \times 2 + 0 \times 1 \\&= 1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + \\&\quad 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 \\&= 10110100_2\end{aligned}$$

## Binääriluvuista 2/2

- ▶ 2-järjestelmä → 10-järjestelmä

$$\begin{aligned}10110_2 &= 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \\ &= 1 \times 16 + 0 \times 8 + 1 \times 4 + 1 \times 2 + 0 \times 1 \\ &= 22\end{aligned}$$

Termejä:

- ▶ Ylin bitti = merkitsevin bitti
- ▶ Alin bitti = vähiten merkitsevä bitti = 1. bitti
- ▶ n:s bitti = n:neksi vähiten merkitsevä bitti

## Heksadesimaali 1/2

- ▶ 16-kantainen, muunnokset vastaavasti mutta kantana 16
- ▶ Muunnos binääriin ja takaisin helppo

| <i>Heksa</i> | <i>Binääri</i> | <i>Heksa</i> | <i>Binääri</i> |
|--------------|----------------|--------------|----------------|
| 0            | 0000           | 8            | 1000           |
| 1            | 0001           | 9            | 1001           |
| 2            | 0010           | A            | 1010           |
| 3            | 0011           | B            | 1011           |
| 4            | 0100           | C            | 1100           |
| 5            | 0101           | D            | 1101           |
| 6            | 0110           | E            | 1110           |
| 7            | 0111           | F            | 1111           |

## Heksadesimaali 2/2

- ▶ Muunnos binääriluvusta:
  - ▶ Lisää binäärilukuun etunollia kunnes bittejä on neljällä jaollinen määrä
  - ▶ Ryhmittele neljän bitin osiin
  - ▶ Muunna jokainen osa erikseen heksanumeroksi
  - ▶ Esim.  $110101_2 \rightarrow 00110101_2 \rightarrow 0011\ 0101 \rightarrow 35_{16}$
- ▶ Muunnos binääriluvuksi:
  - ▶ Muunna jokainen heksanumero suoraan neljäksi bitiksi
  - ▶ Esim.  $2A_{16} \rightarrow 0010\ 1010$
- ▶ C:ssä heksaluvut etuliitteellä **0x**, esim. **0x2A** (tai **0x2a**)
- ▶ Binäärilukujen suoraan ilmaisemiseen ei ole standardia tapaa...

## Kahden komplementti

- ▶ Muistiin tallennetuilla binääriluvuilla on pituus
  - ▶ Esim. 0000 0001, eikä 1
- ▶ Etumerkillisissä luvuissa ylin bitti kertoo yleensä, onko luku negatiivinen
- ▶ Yleisin on *kahden komplementti*: Vastaluku muodostetaan kääntämällä bitit ja lisäämällä 1

|     |    |
|-----|----|
| 011 | 3  |
| 010 | 2  |
| 001 | 1  |
| 000 | 0  |
| 111 | -1 |
| 110 | -2 |
| 101 | -3 |
| 100 | -4 |

## Luvut muistissa

- ▶ Tavu (8 bittiä) on yleensä pienin muistiyksikkö
  - ▶ Muiden muuttujien tilavaatimus on tavun monikerta
- ▶ Valitettavasti valinnanvaraa on myös siinä, missä järjestyksessä isomman luvut tavut tallennetaan (endianness)
- ▶ Eri järjestelmissä eri järjestykset
- ▶ Esim. arvo  $0x12345678$  voidaan tallettaa tavuina järjestyksessä 12 34 56 78 (ns. *big endian*) tai 78 56 34 12 (*little endian*)
- ▶ Bittitason endianness harvoin mielekäs käsite

## Liukuluvut

- ▶ Liukuluku on tietokoneissa yleisesti käytetty tapa esittää reaalityluvut
- ▶ Esittämiseen tarvitaan etumerkki ( $s$ ), mantissa ( $m$ ) ja eksponenttiosa ( $c$ )
- ▶ *float*, *double* ja *long double* eroavat mantissan ja eksponenttiosan pituudessa
- ▶ Kaava:  $(-1)^s \times m \times 2^c$
- ▶ Vrt. kymmenpotenssinotaatio ( $2.5 \times 10^3$ ) – erotuksena tässä kanta on 10:n sijaan tietokoneille kätevämpi 2

# Miksi?

- ▶ Miksi kukaan haluaisi käsitellä yksittäisiä bittejä?
  - ▶ Laiteajurit
  - ▶ Tietoliikenneprotokollat
  - ▶ Tiedon tiivistäminen
  - ▶ Salaukset ja tiivisteet toimivat usein bittitasolla

## Bittien käsittely C:ssä 1/2

- ▶ Bittitason operaatiot (and, or, xor, not) määritelty kokonaisluvuille
- ▶ Yleensä käytetään etumerkittämiä lukuja ja heksadesimaaliesitystä
- ▶ Huomaa että bittioperaatiot oikeastaan käsittelevät itse lukua, ei sen esitystä muistissa
  - ▶ Eli jos haluat käsitellä esim. kokonaisluvun yhdeksättä bittiä, ei tarvitse miettiä endianness-ongelmia
- ▶ Bittitason operaattorit C:ssä:
  - & and (&& on looginen operaattori)
  - | or (|| on looginen operaattori)
  - ^ xor (muista, että tämä ei ole potenssiinkorotus)
  - ~ not (! on looginen operaattori)

## Bittien käsittely C:ssä 2/2

- ▶ Näillä voidaan käsitellä yksittäisiä bittejä *bittimaskin* avulla
- ▶ Esim.  $a \& 0x10$  (kertoo, onko viides bitti) päällä
- ▶  $a | 0x10$  (asettaa kyseisen bitin)
- ▶ Bitin nollaus:  $a \& (\sim 0x10)$
- ▶ Bitin kääntäminen:  $a \wedge 0x10$

## Siirto-operaattorit (bit shift) 1/2

- ▶ << vierittää vasemmalle, >> oikealle
- ▶ Määritelty kokonaisluvuille, mutta yksikäsitteisesti vain etumerkittömille luvuille
- ▶ "Putoavat" bitit heitetään menemään, tyhjä tila täytetään nolllalla
- ▶ Esim. 123 >> 2 vierittää lukua 123 kaksi bittiä oikealle, eli bittijonosta poistetaan kaksi viimeistä numeroa

## Siirto-operaattorit (bit shift) 2/2

- ▶ Käytännössä  $123 \gg 2$  jakaa luvun 123 neljällä ( $2^2$ ), pyöristäen alaspäin
  - ▶ Miksi? Mieti, mitä tapahtuu, jos desimaaliluvusta poistetaan viimeisiä numeroita.
- ▶ Vastaavasti  $123 \ll 2$  kertoo neljällä
- ▶ Jos tarkoituksesi on jako- tai kertolasku, käytä mieluummin oikeaa operaattoria / tai \*
- ▶ Huomaa operaattoreiden  $\ll$  ja  $\gg$  sitovuus
  - ▶ Heikompi kuin yhteen- ja vähennyslaskulla
  - ▶  $123 \ll 4 + 65 \gg 2 = 123 \ll (4+65) \gg 2$ , mikä tuskin oli se mitä ohjelmoija halusi
  - ▶ Käytä sulkuja vaikka osaisitkin sitovuusjärjestyksen ulkoa, lukija ei välttämättä osaa

## Lopuksi

- ▶ Kiitoksia minun puolestani, menestystä loppukurssille
- ▶ Jäljellä vielä yksi luento, to 28.2.: Ryhmätyön Best Practices ja versionhallinta (Jari Vanhanen)
- ▶ Muuten tulossa enää neljäs (viimeinen) harjoitustyökierros ja projektityö
- ▶ Vapaaehtoisesta skriptauskurssista lisätietoa myöhemmin