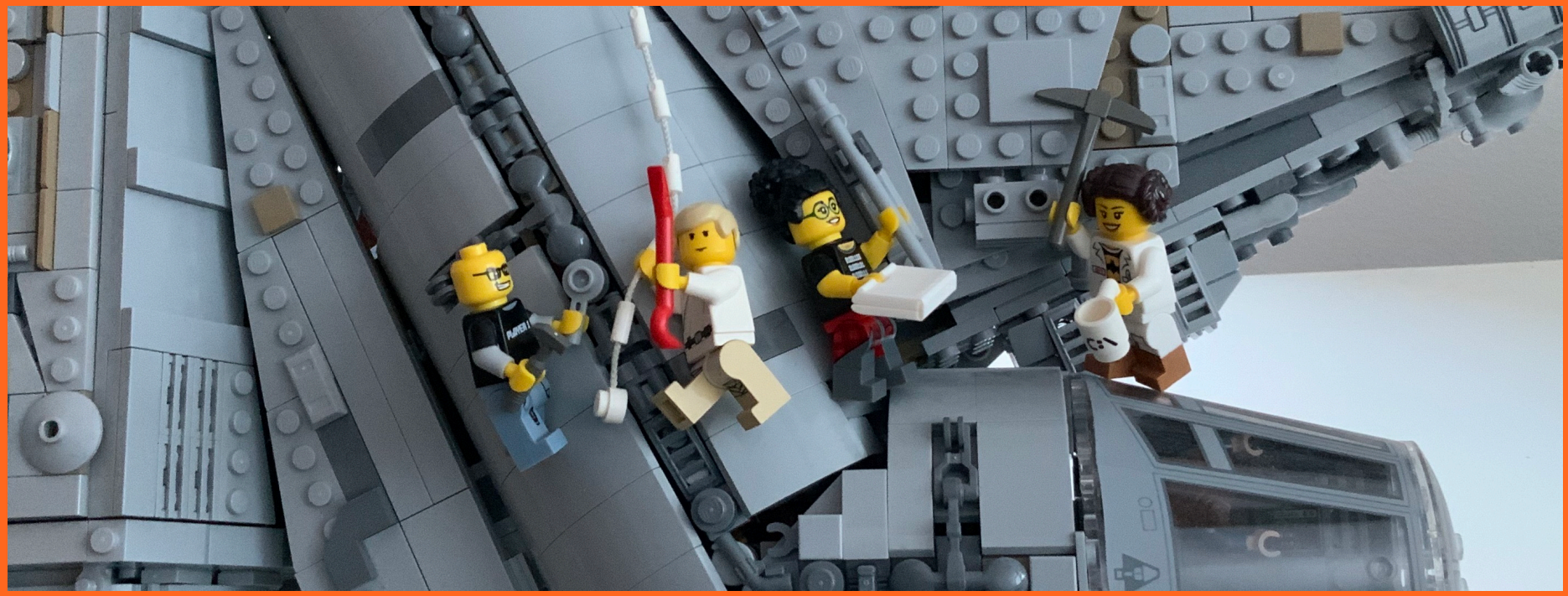
CS-A1120 Ohjelmointi 2



Aalto-yliopisto
Perustieteiden
korkeakoulu

Luento 1
kevät 2026

Sanna Suoranta (suomeksi) ja
Lukas Ahrenberg (in English, T1 hall)

23.2.2026

https://presemo.aalto.fi/o2fi2026

Tavoitteet

https://presemo.aalto.fi/o2fi2026

Kun olet pilkkonut monimutkaisen idean pieniksi vaiheiksi,

jotka tyhmäkin kone pystyy käsittelemään,

olet varmasti oppinut siitä jotain itsekin.

Douglas Adams, Dirk Gentlyn holistinen etsivätoimisto

(vapaasti suomennettuna englanninkielisestä versiosta)

Aalto-yliopisto
Perustieteiden
korkeakoulu

Kierroksen 1 oppimistavoitteet

- **Tämän viikon jälkeen,**
  - Olet kerrannut Scala-ohjelmointikieltä
  - Pystyt ajamaan testejä valitsemallasi IDE:llä
  - Olet ratkaissut muutaman perustehtävän ohjelmoimalla
  - Olet tutustunut CS-A1120 Ohjelmointi 2 –kurssin tietoihin

  - Ensimmäisen kierroksen harjoituksen DL tämän viikon perjantaina klo 18:00

Ohjelmointi 2

- **Porteista (gates) funktionaaliseen (functional) ja rinnakkaisohjelmointiin (parallel programming) – eli yksityiskohdista kohti kokonaisuutta**

- **Laskenta (computing) – yksi vähiten ymmärrettyjä luonnollisia ilmiöistä**
- **Ohjelmointi (programming) – laskennan valjastamista tiettyä tarkoitusta varten**

Kurssin oppimistavoitteet (korkealla tasolla)

1. **Pystyt kuvaamaan ohjelmoitavan tietokoneen rakenteen ja miten se voidaan rakentaa logiikan avulla**
2. **Pystyt rakentamaan piirirakenteita peruslaskentaa varten yksinkertaisista logiikkaporteista ja väylistä**
3. **Osaat määritellä laskennallisia resursseja (aika, tila, rinnakkaisuus jne) ja analysoida perusohjelmien suorituskykyä**
4. **Osaat selittää rekursion ja kehittää ohjelmia, joissa sitä käytetään funktionaalisella tyylillä**
5. **Osaat kuvata rajapinnan roolia ja toteuttaa perusalgoritmeja yleisiin tehtäviin (haku, lajittelu jne)**

Ja osaat vaikka mitä muutakin!

Omin sanoin, mitä tämä koodi tekee?

```
def test(m: Long) =
    var i = 1L
    var s = 0L
    while i <= m do
        s = s + i
        i = i + 1
    end while
    s
end test
```

- **Keskustele vieressä olevan kaverin kanssa (pari minuuttia)**
- **Ja laita vastauksesi presemoon (kun sen avaan kohta): https://presemo.aalto.fi/o2fi2026**

Kokeile while-summainta konsolissa

- **Laske yhteen ensimmäiset neljä miljardia positiivista kokonaislukua**

```
def test(m: Long) =
      var i = 1L
      var s = 0L
      while i <= m do
            s = s + i
            i = i + 1
      end while
      s
end test
```

```
val NANOS_PER_SEC = 1e9
val test_start_time = System.nanoTime
val x = test(4000000000L)
val test_end_time = System.nanoTime
val test_duration = test_end_time - test_start_time
println("the test took %.2f seconds".format(test_duration/NANOS_PER_SEC))
```

Kokeile while-summainta konsolissa

```
the test took 1.06 seconds
```

- **(Intel Core i5-1135G7 @ 2.40GHz × 8 processor and 16 GiB of main memory)**

Miljardeja laskutoimituksia sekunnissa

- **$4 * 10^9$ iteraatiota silmukassa**
- **1,06 sekunnissa**
- **Vertaa valon nopeuteen tyhjiössä: 299 792 458 m/s**
- **Eli karkeasti $(3 * 10^8) / (4 * 10^9) = 7.5 * 10^{-2}$ m/iteraatiokierros**
- **Eli ajassa, joka menee yhteen iteraatiokierrokseen (vähintään kaksi yhteenlaskua, vertailu ja hyppy), valo etenee tyhjiössä 7,5 cm**

Onko ”parempaa” tapaa implementoida `test`?

https://presemo.aalto.fi/o2fi2026

```
def test(m: Long) =
      var i = 1L
      var s = 0L
      while i <= m do
            s = s + i
            i = i + 1
      end while
      s
end test
```

Onko ”parempaa” tapaa implementoida `test`?

https://presemo.aalto.fi/o2fi2026

- **Yksi ehdotus**

```
def test2(m: Long) = m * ((m + 1)/2.0)
```

- **Toinen ehdotus**

```
def test3(m:Long) = (0L to m).sum
```

- **Kolmas rekursion avulla**

```
def test4(m :Long): Long = if m > 0 then m +test4(m -1) else 0
```

- **Ja neljäs (periaatteessa sama kuin toinen)**

```
def test5(m: Long): Long = (0L to m).reduce(_ + _)
```

Scalan kertausta: Range

- **Voimme käyttää joko `to` tai `until` Rangen luomiseen Scalassa**
- **`to` on inklusiivinen (viimeinen elementti mukana)**

```
scala> (0 to 10) val res:
scala.collection.immutable.Range.Inclusive = Range 0 to 10
scala> (0 to 10).last
val res: Int = 10
```

- **`until` on eklusiivinen (viimeinen elementti ei ole mukana)**

```
scala> (0 until 10)
val res: Range = Range 0 until 10
scala> (0 until 10).last
val res: Int = 9
```

Scalan kertausta: Range

- **Scalan `Range` perii `SeqOps`-piirteen, eli voimme käyttää metodeita `map, fold, zip` jne**
- **Esim. Neliöiden laskeminen**

```
scala> (1 to 3).map(x => x * x)
val res: IndexedSeq[Int] = Vector(1, 4, 9)

scala> Seq(1,2,3).map(x => x * x)
val res: Seq[Int] = List(1, 4, 9)
```

- **Samasta syystä voimme kirjoittaa summan laskemisen**

```
def test3(m: Long) = (0L to m).sum
```

Scalan kertausta: Rekursio

- **Rekursiossa funktio kutsuu itseään**

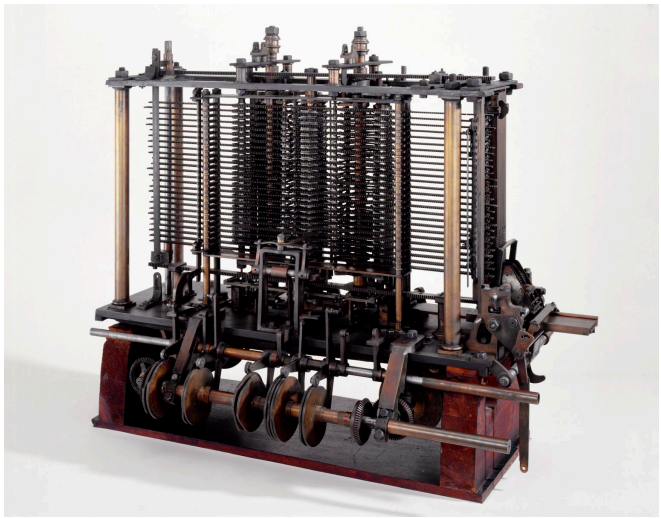
```
def test4(m :Long): Long =
  if m > 0 then
    m + test4(m - 1) // rekursioaskel
  else 0             // perustapaus (näitä voi olla monta)
```

- **Se tarvitsee perustapauksen (base case)**
- **Ja laskettavan rekursioaskeleen (recursive step)**

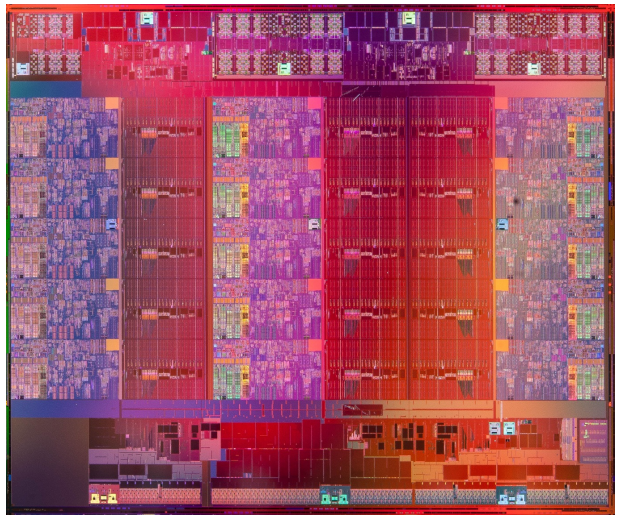
- **Kertaustehtävä rekursiosta nyt, mutta palaamme tähän kierroksella 8**

Kaikki ohjelmat vaativat fyysisen prosessin

Antikythera-laite
(kuva: wikipedia CC 2.5. Generic)



Charles Babbagen Analyyttinen kone
(kuva: wikipedia CC 2.5 SA)



Intel Xeon E7 v2 die
(kuva: Intel Newsroom Photo Gallery)

Laskenta on luonnollinen ilmiö

- **Ohjelmoinnilla voimme saada laskennan valjastettua ratkaisemaan käsillä olevaa tehtävää**
- **Fyysiset rajat rajoittavat ehkä sitä millaisia ongelmia voimme ratkaista, ainakin missä ajassa saamme niitä ratkaistua**

Ohjelmoinnin periaatteet

- **Riippumattomia ohjelmointikielestä ja laitteiston arkkitehtuurista**
- **Ohjelmointitaito ja laskennan ymmärtäminen ovat olennaisia tieteellisessä koulutuksessa**
- **Tietojenkäsittelytiede/tietotekniikka (computer science) tutkii laskennan ilmiöitä ja käyttää laskevia koneita**

Ohjelman ymmärtäminen vaatii tietoa

```
scala> def test(m: Long) = m * ((m + 1)/2.0)
def test(m: Long): Double
scala> test(30)
val res: Double = 465
scala>test(4000000000)
val res: Double = 8.000000002E18
```

- **Miksi Double? Entä jos…**

```
scala> def test(m: Long) = m * ((m + 1)/2)
def test(m: Long): Long

scala> test(30)
val res: Long = 450
```

- **kokeillaan isoa numeroa:**

```
scala> def test(m: Long) = (m * (m + 1))/2
def test(m: Long): Long

scala> test(4000000000)
val res: Long = -1223372034854775808
```

Ohjelman ymmärtäminen vaatii tietoa

```
scala> def test(m: Long) = m * ((m + 1)/2.0)
def test(m: Long): Double
scala> test(30)
val res: Double = 465
scala>test(4000000000)
val res: Double = 8.000000002E18
```

- **Miksi Double? Entä jos olisi…**

```
scala> def test(m: Long) = m * ((m + 1)/2)
def test(m: Long): Long

scala> test(30)
val res: Long = 450
```

- **kokeillaan isoa numeroa:**

```
scala> def test(m: Long) = (m * (m + 1))/2
def test(m: Long): Long

scala> test(4000000000)
val res: Long = -1223372034854775808
```

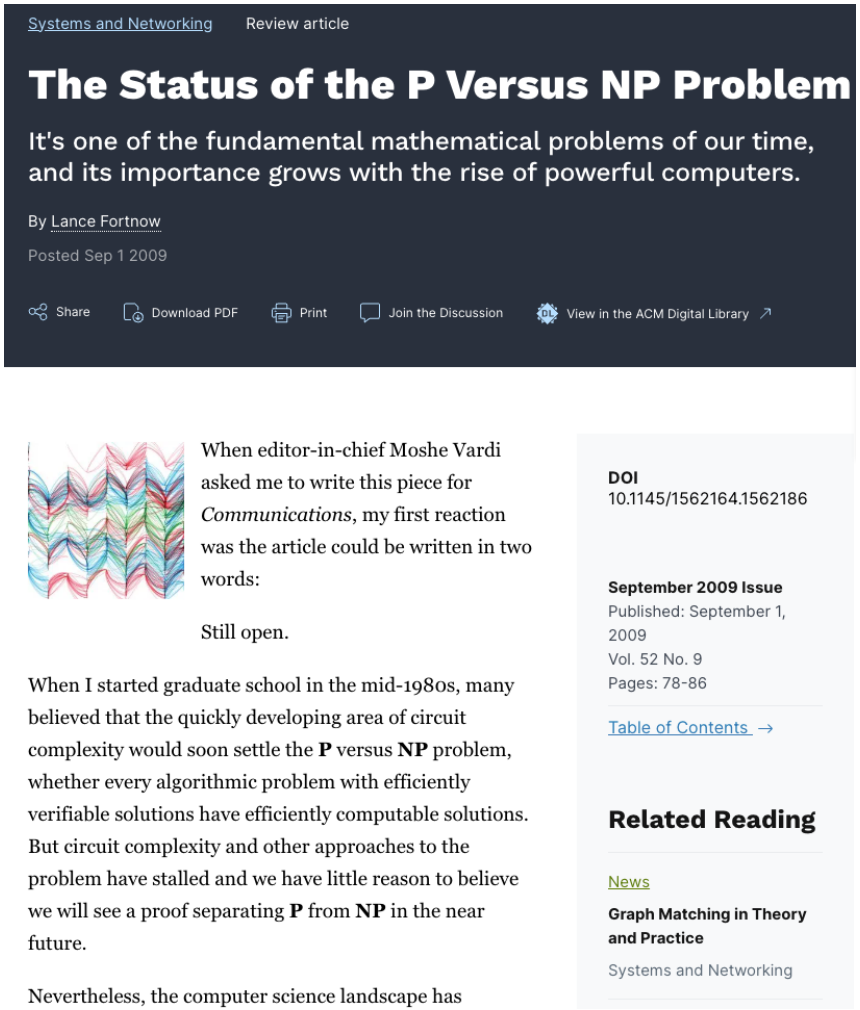
- **Esim. Miten eri tyypit käyttäytyy jakolaskun yhteydessä**
- **Miten iso luku mahtuu tiettyyn tyyppiin**

Tietokoneet ovat tehokkaita – entä muuta?

- **Mitä jos tietoa on liikaa yhdelle tietokoneelle?**
- **Miten ohjelmoit uuden arkkitehtuurin tietokonetta?**
- **Onko jotain mitä emme voi ratkaista ohjelmalla?**
- **Mitä resursseja tarvitaan jonkin laskennan suorittamiseen?**
- **…**

Tietojenkäsittelytiede on “nuori” ala

- **Tietojenkäsittelyn tutkimus ja teoria alkoi vasta 1900-luvulla**
- **P vs NP on mainio (ja kuuluisa) esimerkki**
  - P (polynomial) – ongelma voidaan ratkaista “nopeasti”
  - NP (non-polynomial) – ratkaisu voidaan tarkistaa ”nopeasti”
  - Onko nämä ongelmat samoja?
- **Meillä on puutteellinen tieto siitä, mitkä tehtävät voidaan ratkaista tehokkaasti**



Systems and Networking Review article

The Status of the P Versus NP Problem

It's one of the fundamental mathematical problems of our time, and its importance grows with the rise of powerful computers.

By Lance Fortnow

Posted Sep 1 2009

Share Download PDF Print Join the Discussion View in the ACM Digital Library

When editor-in-chief Moshe Vardi asked me to write this piece for *Communications*, my first reaction was the article could be written in two words:

Still open.

When I started graduate school in the mid-1980s, many believed that the quickly developing area of circuit complexity would soon settle the **P** versus **NP** problem, whether every algorithmic problem with efficiently verifiable solutions have efficiently computable solutions. But circuit complexity and other approaches to the problem have stalled and we have little reason to believe we will see a proof separating **P** from **NP** in the near future.

Nevertheless, the computer science landscape has

DOI
10.1145/1562164.1562186

September 2009 Issue
Published: September 1, 2009
Vol. 52 No. 9
Pages: 78-86

Table of Contents →

Related Reading

News
Graph Matching in Theory and Practice
Systems and Networking

Esimerkki P-tyyppisestä ongelmasta

```
89,117,153,265,395,477,516,664,670,795
```

P-tyyppinen ongelma

- **Onko numero 2026 yllä olevien kokonaislukujen listassa?**

- **Mitä pitää tehdä ongelman ratkaisemiseksi?**

Esimerkki NP-tyyppisestä ongelmasta

89,117,153,265,395,477,516,664,670,795

NP-tyyppinen ongelma

- **Voidaanko numero 2026 muodostaa summana yllä olevista kokonaisluvuista, jos jokaisen luvun voi käyttää vain yhden kerran?**

- **Mitä pitää tehdä ongelman ratkaisemiseksi?**

Miten lähtisit kirjoittamaan ohjelmaa, joka ratkaisisi tämän (yleisessä tapauksessa)?

- **On täysin mahdollista kirjoittaa ohjelmia, jotka etsivät vastausta, mutta**
- **Ei ole tunnettua tehokasta ratkaisumenetelmää tälle ongelmalle yleisessä tapauksessa**
  - Eli kun lukuja on mielivaltainen määrä ja ne on mielivaltaisen kokoisia

On kuitenkin hyvin helppo tarkistaa, onko ratkaisu pätevä vaiko ei

- **“Voiko luvun 2026 muodostaa annetuista luvuista yhteenlaskun avulla?”**
- **Oletetaan, että joku kertoo sinulle, että luvut ovat**

```
89, 117, 153, 395, 477 ja 795
```

- **Mitä pitää tehdä ratkaisun tarkastamiseksi?**

On kuitenkin hyvin helppo tarkistaa, onko ratkaisu pätevä vaiko ei

- **Tarkistetaan, että luvut löytyvät alkuperäisestä listasta**
- **Tarkistetaan, että jokainen luku on käytössä vain kerran**
- **Suoritetaan yhteenlasku**

89+117+153+395+477+795=2026

Tämän kurssin suorittaminen

Aalto-yliopisto
Perustieteiden
korkeakoulu

Ohjelmointi 2 – kurssin sisältö

- **Johdanto**
  - 01– warm-up
- **Moduuli I – Tietokoneen salaisuudet**
  - 02 – bitit ja data
  - 03 – kombinaatiologiikka
  - 04 – sekvenssilogiikka
  - 05 – ohjelmoitava tietokone
- **Moduuli II – ohjelmoinnin abstraktiot ja analyysi**
  - 06 – kokoelmat ja funktiot
  - 07 – tehokkuus
  - 08 – rekursio
- **Moduuli III – raja-alueita**
  - 09 – samanaikaisuus ja rinnakkaisuus
  - 10 – oppivat koneet

Kurssin materiaalit ja työmäärä

- **Luennot suomeksi Sanna 23.2. – 30.3. AS2, 20.4-25.5. T1 ja englanniksi Lukas T1-sali, klo 10-12**
  - Luennot diat + sanasto: https://a1120.cs.aalto.fi/2026/slides/
  - Vanhat videot englanniksi, tekstitys suomeksi https://panopto.aalto.fi/
- **Materiaalit on tarjolla englanniksi**
  - Oppimateriaali: https://a1120.cs.aalto.fi/2026/notes/
  - Harjoitustehtävät: https://plus.cs.aalto.fi/cs-a1120/2026/
- **Työmäärä on 5 op, jaettuna 10 kierrokseen, viikottain**
  - Luento 2h
  - Itsenäinen kurssimateriaalin opiskelu 2h
  - Harjoitustehtävien itsenäinen työstäminen 9h

ja lisäksi

  - Tentti 3h ja siihen valmistautuminen 5h

Harjoitukset – ohjelmointia ja monivalintoja

- **Määräajat täysillä pisteillä viikottain perjantaisin klo 18:00**
  - Tarkat ajat A+:ssa, ei palautusta tenttiviikolla tai lomilla
- **Myöhäiset palautukset mahdollisia 31. toukokuuta asti vähemmillä pisteillä (70%), tämän jälkeen pisteitä ei enää saa mitenkään**
  - Huono puoli: malliratkaisuja ei julkaista
- **Kolmentyyppisiä harjoituksia: A & B ja haasteharjoituksia**
  - A- ja B-tason tehtäviä, joista B-tason tehtäviä pitää saada tietty määrä, on tarjolla n. 3800 pisteen edestä
  - Assistentit auttavat tavallisten (A&B-tason) harjoitusten kanssa
  - Haateharjoitukset (C-taso) ovat hankalampia, vaativat enemmän aikaa ja työtä, mutta voivat palkitsevia oivallusten muodossa
  - Arvosanan 5 voi saada myös ilman haasteharjoituksia

Harjoitusten ratkaisemisesta

- **Ohjeita harjoituksiin löytyy kurssimateriaalin liitteestä**
- **Pohjakoodit (zip-tiedostona sbt-projekti) on tarjolla A+ssa**
- **Lähetä vastauksesi A+:aan**
- **Käytämme pääosin Scala 3.3:sta, yksi kierros(omaa) assembleria**
- **Harjoitukset sbt-projekteina:**
  - Visual Studio Code, Scala Metals –laajennuksella
  - IntelliJ Scala-plugarilla
- **Vinkkejä**
  - Lue kurssimateriaali ennen aloittamista!
  - Lue A+-ohjeet, ohjelmakoodi ja kommentit huolella
  - Useimmat harjoitukset sisältävät testejä – käytä niitä
  - Huomaa, että sinulla on vain rajoitettu määrä palautuskertoja A+ssa
- **Aloita ajoissa*** **ja**
- **Osallistu harjoitusryhmiin!**

*Keinoja jahkailun vähentämiseen: https://osf.io/mwtrb

Harjoitusryhmiä joka arkipäivä

	ma	ti	ke	to	pe
8-10					T6
10-12	Luento (en)			T6	
12-14		T8	T8		
14-16	Luento (fi)				
16-18	T8				
18-20					

Harjoitusryhmät alkavat aina akateemisen vartin jälkeen parillisin tasatunnein

(harjoitusryhmien määrää saatetaan vähentää kurssin aikana)

Ei harjoituksia:

Tenttiviikolla 13-17.4,

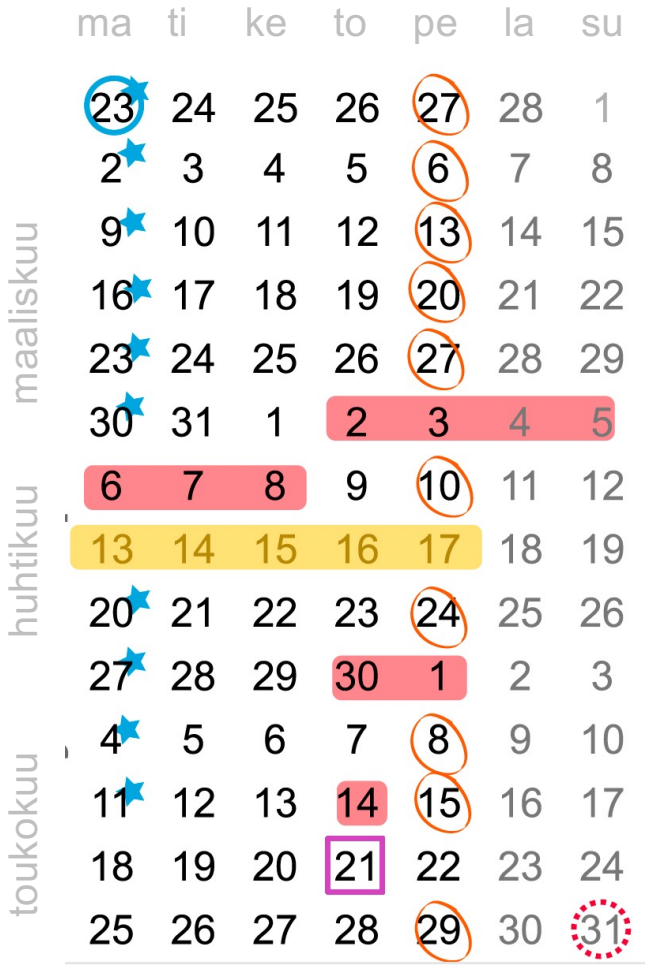
Kevättauolla 2.-8.4.

Eikä 30.4, 14.5, 29.5

Apua suomeksi ja englanniksi, ruotsiksi to 14-16 ja pe 12-14 (tarkista MyCosta muutokset)

- **Kampuksella T-talolla, ei etukäteisilmoittautumista**
- **Tilaisuudessa ilmoittaudutaan neuvontajonoon (LabQueue) A+:n kautta**
- **Pääasiassa keskitytään käsillä olevaan kierrokseen, mutta kaikki tervetulleita!**
- **Tarvitset pääsytunnisteen!**
- **Harjoituksia on vähemmän toukokuussa**

Kurssin kalenteri



Kurssi alkaa

Täysien pisteiden määräaika klo 18:00

Loma – ei opetusta

Tenttiviikko – ei opetusta

Jatkettu määräaika harjoituksille (osa pisteistä vielä saatavilla)

Tentti – muista ilmoittautua viim. viikkoa ennen Sisussa!

Luento klo 14:15-16:00 – vanhoja luentovideoita saatavilla panoptossa. Niissä on tekstitykset suomeksi ja englanniksi

Aalto-yliopisto
Perustieteiden korkeakoulu

23.2.2026

Tentti torstaina 21.5.2026 – ILMOITTAUDU!

- **Kurssilla on pakollinen tentti, johon on erillinen ilmoittautuminen**
  - Osasta I pitää saada vähintään 5p / 10p, eli 10 “pientä” tehtävää, joista puolet pitää osata (loppuarvosana 0-2)
  - Osaa II ei arvostella, jollei osasta I ole saatu riittävästi pisteitä
  - Osan II pisteet määrittävät tentin arvosanan 3-5
  - Tenttiarkiston vanhat tentit auttavata osaan II valmistautumiseen
- **Tenttiin pitää ilmoittautua erikseen**
  - Sisussa tentti-ilmoittautuminen on auki 22.3.-14.5.2026
- **“Kynä ja paperi” –tentti**
- **Yksi uusintamahdollisuus järjestetään kesän jälkeen elokuun loppupuolella**
  - aika ja ilmoittautuminen ilmaantuu Sisuun jossain vaiheessa

Kurssin arvostelu

		tentti					
		Osa I < 5p	Osa I ≥ 5p & Osa II < 50%	Osa I ≥ 5p & Osa II < 50%	Osa I ≥ 5p & Osa II ≥ 50%	Osa I ≥ 5p & Osa II ≥ 65%	Osa I ≥ 5p & Osa II ≥ 80%
Harjoitus-tehtävä-pisteet (yhteensä JA B-tason pisteitä vähintään)	0 -1199 tai < 200 B-tason pistettä	0	0	0	0	0	0
	1200 – 1799, joista ≥ 200 B-tasoa	0	1	1	1	1	1
	1800 – 2399, joista ≥ 600 B-tasoa	0	2	2	2	2	2
	2400 - 2999, joista ≥ 800 B-tasoa	0	2	2	3	3	3
	3000 - 3499, joista ≥ 1000 B-tasoa	0	2	2	3	4	4
	≥ 3500, joista ≥ 1300 B-tasoa	0	2	2	3	4	5

- **Harjoitukset (yhteispisteet ja B-tason pisteet) JA tentti määrittää arvosanan**
  - Tentin osasta I pitää saada vähintään 5p/10p
  - Osa II tarkistetaan jos osa I ok

Aalto-yliopisto
Perustieteiden korkeakoulu

Harjoituksista on saatavissa maksimissaan noin 6000 pistettä, (1550 A, 2450 B, 2000 C)

Mahdolliset lisäpisteet

- **Jokaisesta A+-kierroksesta, jossa olet antanut palautetta JA saanut pisteitä vähintään yhdestä (tason A, B tai C) tehtävästä, saat 5 lisäpistettä palautteestasi**
- **Jos vastaat erilliseen kurssipalautteeseen MyCoursesissa kurssin lopussa, saat harjoituspisteisiisi 50 lisäpistettä, jotka lisätään erikseen loppuarvioinnin yhteydessä**
  - Nämä pisteet eivät näy A+:ssa ja ne lisätään A-pisteisiin erikseen

Yhteistyö- ja AI-työkalujen politiikka

Voit antaa arvosteltavaksi vain itse tekemiäsi ratkaisuja. Et saa laittaa mitään kurssimateriaalista mihinkään tekoälytyökaluun tai muutoin jakaa materiaalia ulkopuolisille tahoille.

- **Harjoitustehtävät ovat henkilökohtaisia ja yksin tehtäviä**
  - On kiellettyä esim. laatia ohjelmakoodia yhdessä, kopioida toistilta koodia, mahdollistaa muiden pääsy omaan ratkaisuunsa (joko antamalla se toiselle suoraan tai laittamalla se jaetulle tai julkiselle foorumille tarjolle jne)
- **On kiellettyä käyttää tekoälytyökaluja (AI tools) tehtävien ratkaisemiseen tällä kurssilla**
  - Toisin sanoin, et saa käyttää työkaluja, jotka generoivat ohjelmakoodia tai muita ratkaisuja tehtävissä
  - Se on huijaamista, samoin kuin toiselta ihmiseltä kopioiminen

https://www.aalto.fi/fi/hakemukset-ohjeet-ja-saannot/hyva-akateeminen-kaytanto-opinnoissa-aalto-yliopistossa-voimassa-112025-alkaen

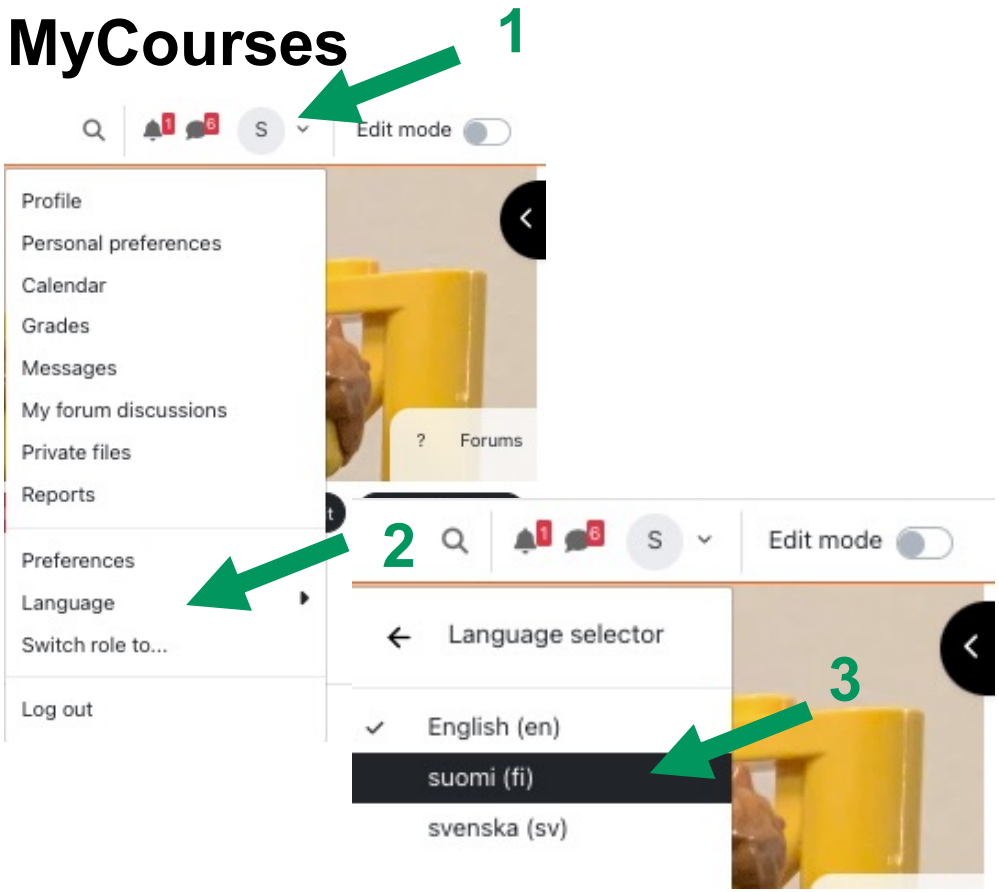
Edellä olevasta huolimatta –

Puhukaa toisillenne!

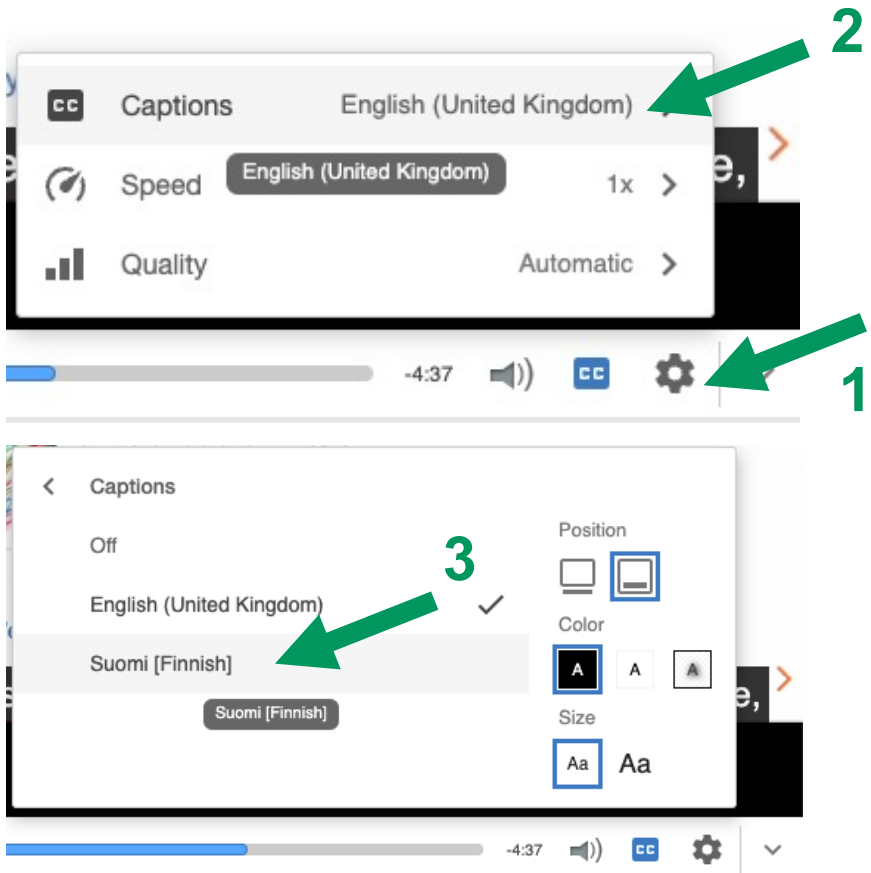
- **Voitte, ja teidän myös kannattaa, tietenkin keskustella tehtävistä kurssimateriaalista toistenne kanssa! Mutta, tehkää se ilman kurssin harjoituksiin liittyviä ohjelmakoodeja.**
- **Saadaksesi apua oman koodisi kanssa ja harjoitusten tekemiseen:**
  - Tulkaa harjoitusryhmiin!
  - Kysykää Zulipissa kurssin henkilökunnalta

Kielen vaihto – MyCourses ja Panopto

MyCourses



Panopto: tekstitykset



Lisää tietoa ja yhteydenotot

- **MyCourses – tilaa ilmoitukset sähköpostiisi**
  - https://mycourses.aalto.fi/course/view.php?id=46390
  - Kielen vaihto onnistuu omista asetuksista (suomi / englanti)
  - Pitkäaikaiset sairastumiset ja esteet: kts. mycourses
- **Kurssin oppimateriaali**
  - https://a1120.cs.aalto.fi/2026/notes/
  - Vuoden 2022-23 luentovideot Panoptossa https://panopto.aalto.fi/
- **Tehtävät löytyvät A+:sta: https://plus.cs.aalto.fi/cs-a1120/2026/**
- **Zulip-keskustelufoorumi kysymyksille**
  - https://cs-a1120-2026.zulip.aalto.fi
  - Käytä Koodisäilöä koodin lähettämiseksi, jos et ole tehnyt palautusta tehtävään (tai olet muuttanut koodiasi)

Yhteystiedot

- **Huom. Kurssin henkilökunta ei vastaa sähköpostitse tai suoriin Zulip-viesteihin harjoitustehtävien tekemistä koskevissa kysymyksissä. Tule harjoitusryhmiin (tai kysy Zulipissa)**
- **Kurssin vastuuopettajat**
  - Lukas Ahrenberg (en/sv) ja Sanna Suoranta (fi/en)
- **Henkilökohtainen tapaaminen**
  - Luentojen yhteydessä (SISUssa L01 enkuksi Lukas ja L02 suomeksi Sanna)
- **Sähköpostiosoitteet**
  - lukas.ahrenberg@aalto.fi, sanna.suoranta@aalto.fi

Ajantasaiset tiedot MyCourses-sivulla

Nämä diat sisältävät katsauksen MyCourses-sivuihin kurssin alussa (23. helmikuuta 2026)

Ajantasaiset tiedot löytyvät kurssin MyCourses-sivulta

Kurssin alku - tarkastuslista

- **[ ] Ilmoittaudu SISUssa**
- **[ ] Lue kurssin tiedot MyCourses-sivuilta**
- **[ ] Hanki pääsytunniste/kortti/sovellus tietokoneluokkia varten**
- **[ ] Lue ohjeet** https://a1120.cs.aalto.fi/2026/notes/instructions.html **ja tarkista, että:**
  - [ ] Pystyt käynnistämään IDE:n (tietokoneluokassa tai omalla koneella)
  - [ ] Osat työskennellä tehtävien kanssa: https://a1120.cs.aalto.fi/2026/notes/instructions--basics.html
  - [ ] Tiedät milloin/missä harjoituksia järjestetään ja miten saat apua https://a1120.cs.aalto.fi/2026/notes/instructions--help.html
- **[ ] Lue ja aloita kierroksen 1 tehtävien tekeminen tänään!**

Kiitokset

- **Tämä kurssi perustuu aikaisempaan materiaaliin, jota ovat kehittäneet**
  - Petteri Kaski
  - Tommi Junttila
  - Mikko Kivelä
  - Shaofeng Jiang
  - sekä lukuiset kurssin assistentit