

DD1361
Programmeringsparadigm
Föreläsning 1: Intro

Per Austrin

KTH
2015-08-31

<https://www.kth.se/social/course/DD1361/>

Dagens föreläsning

Översikt om:

1. Ämnet
2. Lärarna
3. Kursformalia (betyg etc)

Dagens föreläsning

Översikt om:

1. Ämnet

2. Lärarna

3. Kursformalia (betyg etc)

Färgfråga

Vilket eller vilka av följande är inte ett programmeringsspråk?

Röd Java

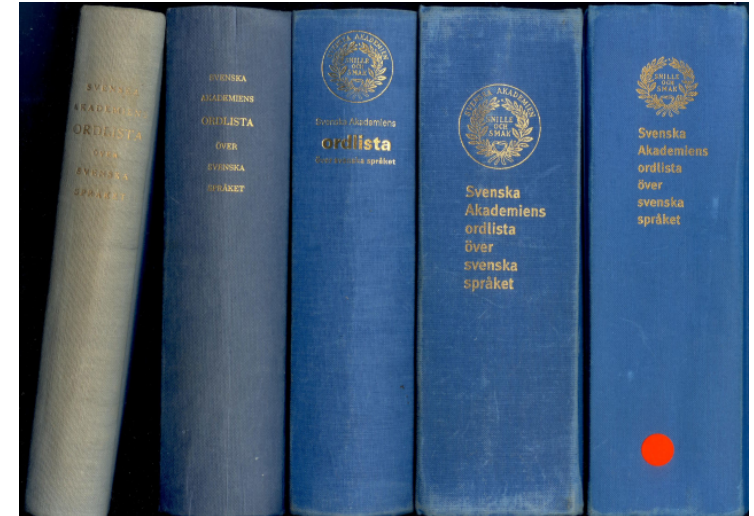
Gul Pseudokod

Blå L^AT_EX

Turkos: Python

Vit: HTML

Programmeringsparadigm?



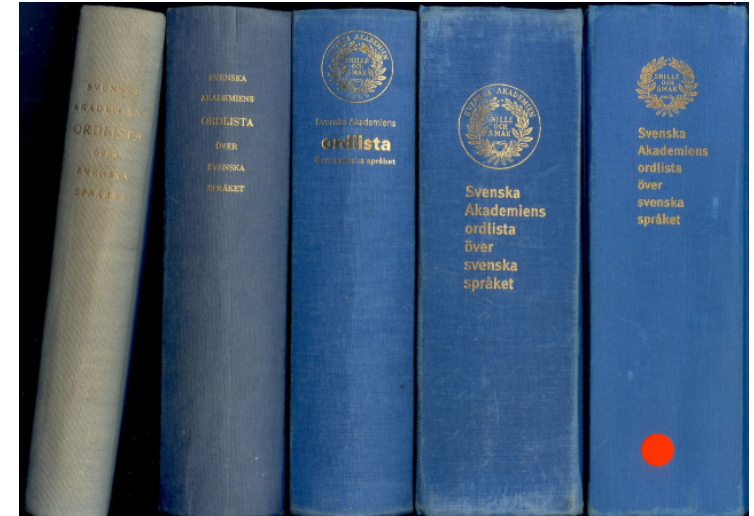
Programmeringsparadigm?

- Programmering: ... ✓



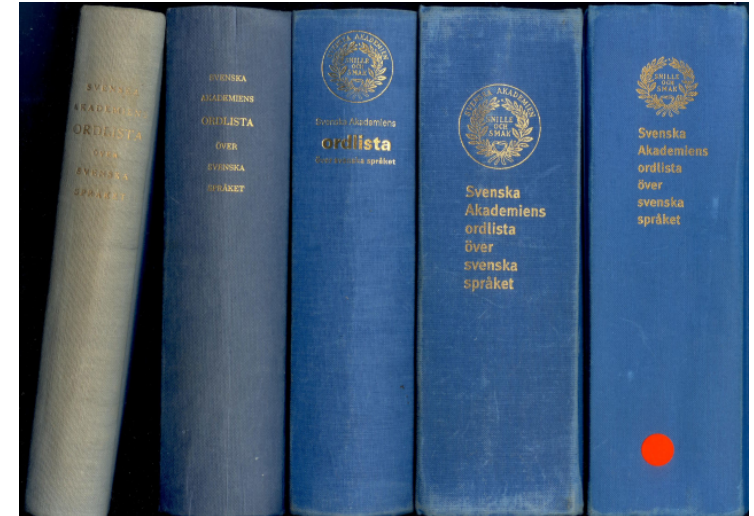
Programmeringsparadigm?

- Programmering: ... ✓
- Paradigm:
 - mönster (lingvistik)
 - världsbild (vetenskapsteori)



Programmeringsparadigm?

- Programmering: ... ✓
- Paradigm:
 - mönster (lingvistik)
 - världsbild (vetenskapsteori)



Programmeringsparadigm:
**konceptuellt sätt att se på hur man
beskriver ett program**

Exempel 1: Imperativ programmering

Gör först A, sedan B, sedan C, etc...

Exempel 1: Imperativ programmering

Gör först A, sedan B, sedan C, etc...

T.ex.

```
A: read integer n
```

```
B: set m = n*n
```

```
C: print m
```

```
D: ...
```

Ex. 2: Objektorienterad programmering

Klasser A, B, C, etc som har data och metoder

Ex. 2: Objektorienterad programmering

Klasser A, B, C, etc som har data och metoder

```
class A {  
    function read() { ... }  
}
```

```
class B {  
    function square() { ... }  
}
```

```
class C {  
    function print() { ... }  
}
```


Exempel 3: Deskriptiv programmering

Beskriv vad programmet ska göra

Exempel 3: Deskriptiv programmering

Beskriv vad programmet ska göra

```
skriv ut kortaste vägen  
från D1 till kårhuset,  
spela sedan en trudelutt
```

Färgfråga

Jag har lurats lite. Om vad?

Röd Deskriptiv programmering är inte en paradigm

Gul Objektorienterad programmering är också
imperativ

Blå Deskriptiv programmering finns inte

Turkos: Alla ovanstående

Vit: Kuggfråga! Per har inte alls lurats

Programmeringsparadigm! (?)

Programmeringsparadigm:

**konceptuellt sätt att se på hur man
beskriver ett program**

Programmeringsparadigm! (?)

Programmeringsparadigm:

konceptuellt sätt att se på hur man

beskriver ett program

→ att beskriva ett program

≈

att programmera

Programmeringsparadigm! (?)

Programmeringsparadigm:
**konceptuellt sätt att se på hur man
beskriver ett program**

En programmeringsparadigm är en uppsättning programmeringskoncept.

Exempel på koncept:

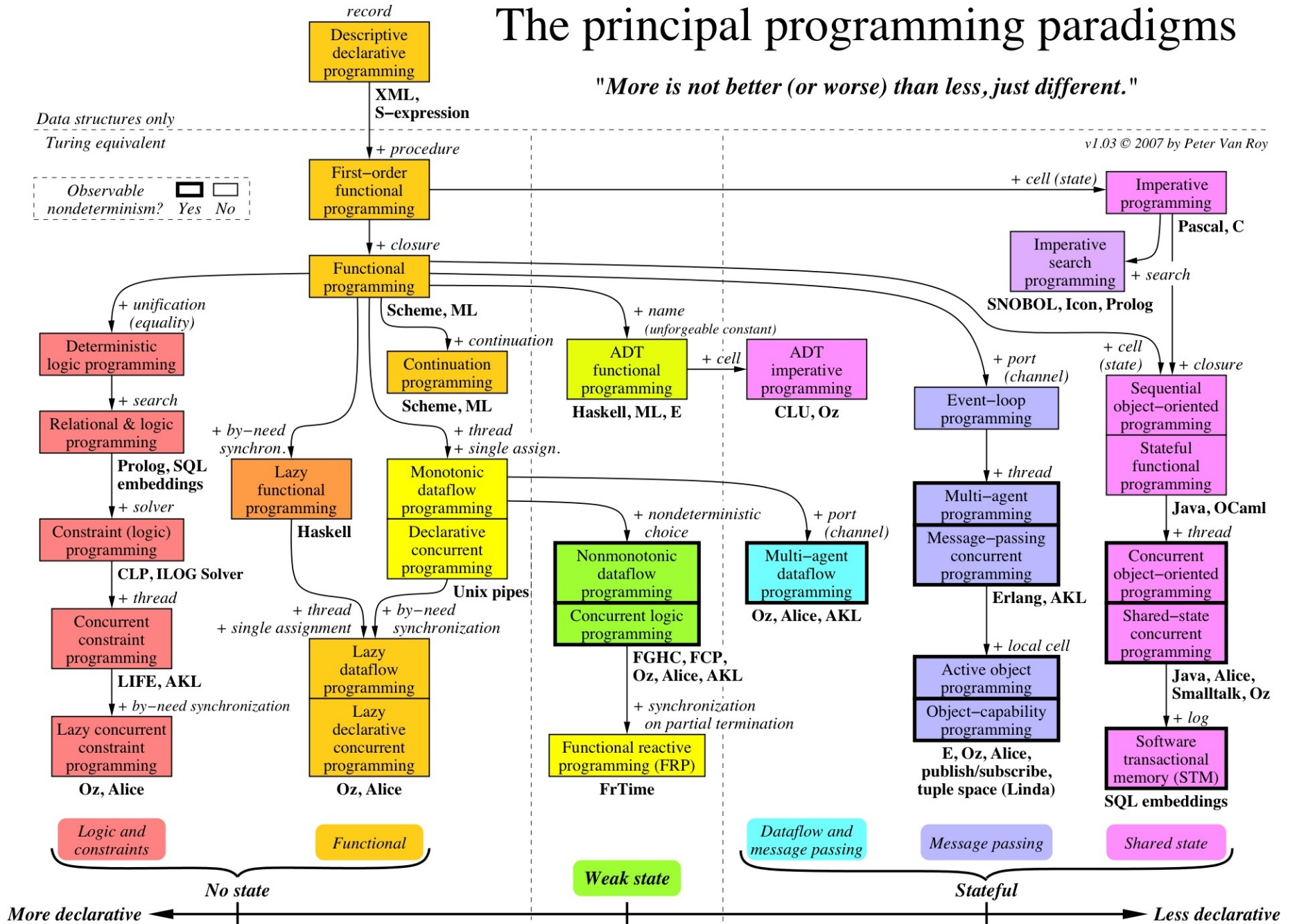
- trådar / parallelism
- objekt (á la objektorientering)

Ett programmeringsspråk som har alla paradigmens koncept tillhör paradigmen

The principal programming paradigms

"More is not better (or worse) than less, just different."

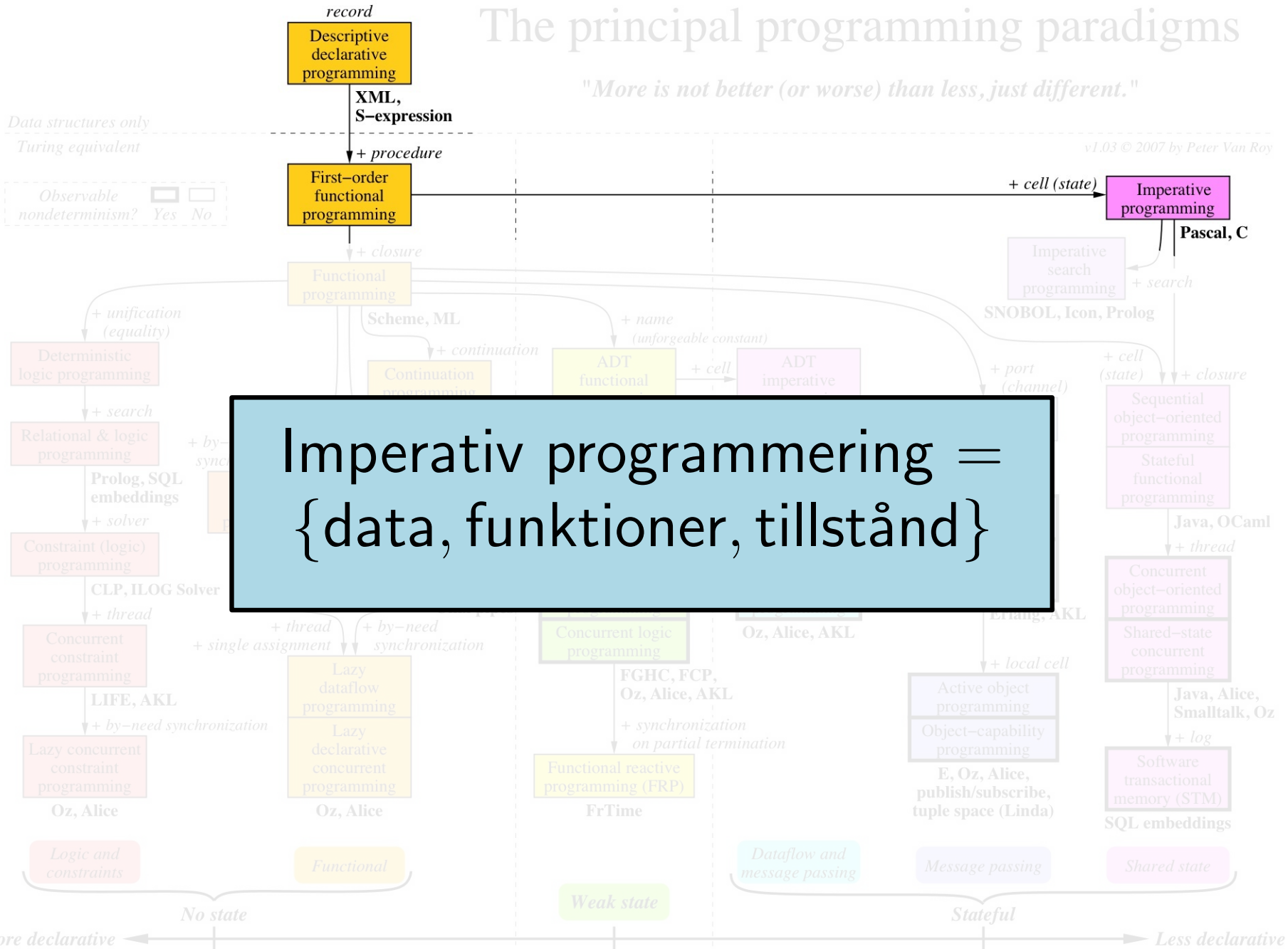
v1.03 © 2007 by Peter Van Roy



The principal programming paradigms

"More is not better (or worse) than less, just different."

v1.03 © 2007 by Peter Van Roy



Imperativ programmering = {data, funktioner, tillstånd}

Viktigt koncept: tillstånd

Imperativ programmering:

programmet har ett *tillstånd* (*state*), och programmet har instruktioner för hur tillståndet ska manipuleras steg för steg för att nå slutresultatet

Viktigt koncept: tillstånd

Imperativ programmering:

programmet har ett *tillstånd* (*state*), och programmet har instruktioner för hur tillståndet ska manipuleras steg för steg för att nå slutresultatet

Alla språk ni hittills stött på i kurser här på KTH har varit imperativa (Java, Go, assembler(?))

Viktigt koncept: tillstånd

Imperativ programmering:

programmet har ett *tillstånd (state)*, och programmet har instruktioner för hur tillståndet ska manipuleras steg för steg för att nå slutresultatet

Alla språk ni hittills stött på i kurser här på KTH har varit imperativa (Java, Go, assembler(?))

Motsats:

Deklarativ programmering:

koden definierar vad slutresultatet ska vara, inte hur det ska uppnås

Viktigt koncept: tillstånd

Imperativ programmering:

programmet har ett *tillstånd (state)*, och programmet har instruktioner för hur tillståndet ska manipuleras steg för steg för att nå slutresultatet

Alla språk ni hittills stött på i kurser här på KTH har varit imperativa (Java, Go, assembler(?))

Motsats:

Deklarativ programmering:

koden definierar vad slutresultatet ska vara, inte hur det ska uppnås

T.ex. deskriptiv programmering (om det hade funnits)

Viktigt koncept: tillstånd

I **imperativ programmering** finns det variabler som utgör programmets tillstånd (tillsammans med implicita variabler som instruktionspekare och dylikt)

Viktigt koncept: tillstånd

I **imperativ programmering** finns det variabler som utgör programmets tillstånd (tillsammans med implicita variabler som instruktionspekare och dylikt)

Programmets instruktioner ändrar variablerna och påverkar därmed programmets tillstånd.

Instruktionerna har *side-effekter*

Viktigt koncept: tillstånd

I **imperativ programmering** finns det variabler som utgör programmets tillstånd (tillsammans med implicita variabler som instruktionspekare och dylikt)

Programmets instruktioner ändrar variablerna och påverkar därmed programmets tillstånd.

Instruktionerna har *sido-effekter*

I **deklarativ programmering** definierar man vad saker är och de kan inte ändra värde.

Instruktionerna har *inga sido-effekter*.

Viktigt koncept: tillstånd

I **imperativ programmering** finns det variabler som utgör programmets tillstånd (tillsammans med implicita variabler som instruktionspekare och dylikt)

Programmets instruktioner ändrar variablerna och påverkar därmed programmets tillstånd.

Instruktionerna har *sido-effekter*

I **deklarativ programmering** definierar man vad saker är och de kan inte ändra värde.

Instruktionerna har *inga sido-effekter*.

Fundamentalt annorlunda sätt att tänka på beräkningar än imperativ programmering!

Del 1 (\approx September): Funktionell programmering

En typ av deklarativ programmering där resultaten av beräkningar beskrivs som värdet av (matematiska) funktioner

Del 1 (\approx September): Funktionell programmering

En typ av deklarativ programmering där resultaten av beräkningar beskrivs som värdet av (matematiska) funktioner

$$\text{sort}(L) = \begin{cases} L & \text{if } \text{len}(L) \leq 1 \\ \text{merge}(\text{sort}(\text{firsthalf}(L)), \\ \quad \text{sort}(\text{secondhalf}(L))) & \text{otherwise} \end{cases}$$

$$\text{merge}(L_1, L_2) =$$

$$\begin{cases} L_1 & \text{if } \text{length}(L_2) = 0 \\ L_2 & \text{if } \text{length}(L_1) = 0 \\ \text{head}(L_1) | \text{merge}(\text{tail}(L_1), L_2) & \text{if } \text{head}(L_1) \leq \text{head}(L_2) \\ \text{head}(L_2) | \text{merge}(L_1, \text{tail}(L_2)) & \text{if } \text{head}(L_1) > \text{head}(L_2) \end{cases}$$

$\text{sort}([5, 12, 43, 1])$ returnerar $[1, 5, 12, 43]$

Del 2 (\approx Oktober): Logikprogrammering


En typ av deklarativ programmering där resultateten av beräkningar specificeras i någon formell logik

Vanligast: första ordningens predikatlogik

Del 2 (\approx Oktober): Logikprogrammering

En typ av deklarativ programmering där resultateten av beräkningar specificeras i någon formell logik

Vanligast: första ordningens predikatlogik

 Logik-kursen!

Del 2 (\approx Oktober): Logikprogrammering

En typ av deklarativ programmering där resultateten av beräkningar specificeras i någon formell logik

Vanligast: första ordningens predikatlogik

Definiera ett predikat $\text{sort}(L, S)$ som är sant om S är sorteringen av L .

$\text{sort}(L, S) = \text{sorted}(S) \mathbf{and} \text{permutation}(L, S)$

Del 2 (\approx Oktober): Logikprogrammering

En typ av deklarativ programmering där resultateten av beräkningar specificeras i någon formell logik

Vanligast: första ordningens predikatlogik

Definiera ett predikat $\text{sort}(L, S)$ som är sant om S är sorteringen av L .

$\text{sort}(L, S) = \text{sorted}(S) \textbf{ and } \text{permutation}(L, S)$

$\text{sort}([5, 12, 43, 1], S) = \textbf{true}$

Medför att S måste vara $[1, 5, 12, 43]$

Del 3 (\approx November): Formella språk

En del av kursen kommer vi också ägna åt formella språk och syntaxanalys

Teori och metoder för att tolka ("parsa") t.ex. kod i något programmeringsspråk

Första steget i att bygga en kompilator: hur skriver man ett program som avgör om en textfil är ett giltigt Java-program?

Andra viktiga paradigmer

Objektorienterad programmering

- Objekt, klasser, arv

Parallellprogrammering

- Tråder, synkronisering, etc

Distribuerade beräkningar

- Beräkningar på olika system
(t.ex. cloud computing)

Färgfråga

Finns det någon koppling mellan vilket/vilka programmeringsparadigm ett språk tillhör, och huruvida språket är ett högnivå-språk eller lågnivå-språk?

Röd Nej, ingen koppling

Gul Ja, språk som tillhör *många paradigm* tenderar att vara mer *högnivåspråk*

Blå Ja, språk som tillhör *många paradigm* tenderar att vara mer *lågnivåspråk*

Turkos: Ja, språk som är *mer deklarativa* tenderar att vara mer *högnivåspråk*

Vit: Ja, språk som är *mer deklarativa* tenderar att vara mer *lågnivåspråk*

Olika paradigmer = Olika styrkor

Tumregel: ju mer deklarativt ett språk är, desto längre från hårdvaran

Olika paradigmer = Olika styrkor

Tumregel: ju mer deklarativt ett språk är, desto längre från hårdvaran

I slutändan ska programmet man skriver köras på samma hårdvara, oavsett vilket språk man skriver i.

Olika paradigmer = Olika styrkor

Tumregel: ju mer deklarativt ett språk är, desto längre från hårdvaran

I slutändan ska programmet man skriver köras på samma hårdvara, oavsett vilket språk man skriver i.

Deklarativa språk gör det lättare att uttrycka komplexa beräkningar...

Olika paradigmer = Olika styrkor

Tumregel: ju mer deklarativt ett språk är, desto längre från hårdvaran

I slutändan ska programmet man skriver köras på samma hårdvara, oavsett vilket språk man skriver i.

Deklarativa språk gör det lättare att uttrycka komplexa beräkningar...

...men det gör dem också svårare att översätta till maskinkod

Olika paradigmer = Olika styrkor

Tumregel: ju mer deklarativt ett språk är, desto längre från hårdvaran

I slutändan ska programmet man skriver köras på samma hårdvara, oavsett vilket språk man skriver i.

Deklarativa språk gör det lättare att uttrycka komplexa beräkningar...

...men det gör dem också svårare att översätta till maskinkod

Imperativa språk tenderar att ge snabbare program...

Olika paradigmer = Olika styrkor

Tumregel: ju mer deklarativt ett språk är, desto längre från hårdvaran

I slutändan ska programmet man skriver köras på samma hårdvara, oavsett vilket språk man skriver i.

Deklarativa språk gör det lättare att uttrycka komplexa beräkningar...

...men det gör dem också svårare att översätta till maskinkod

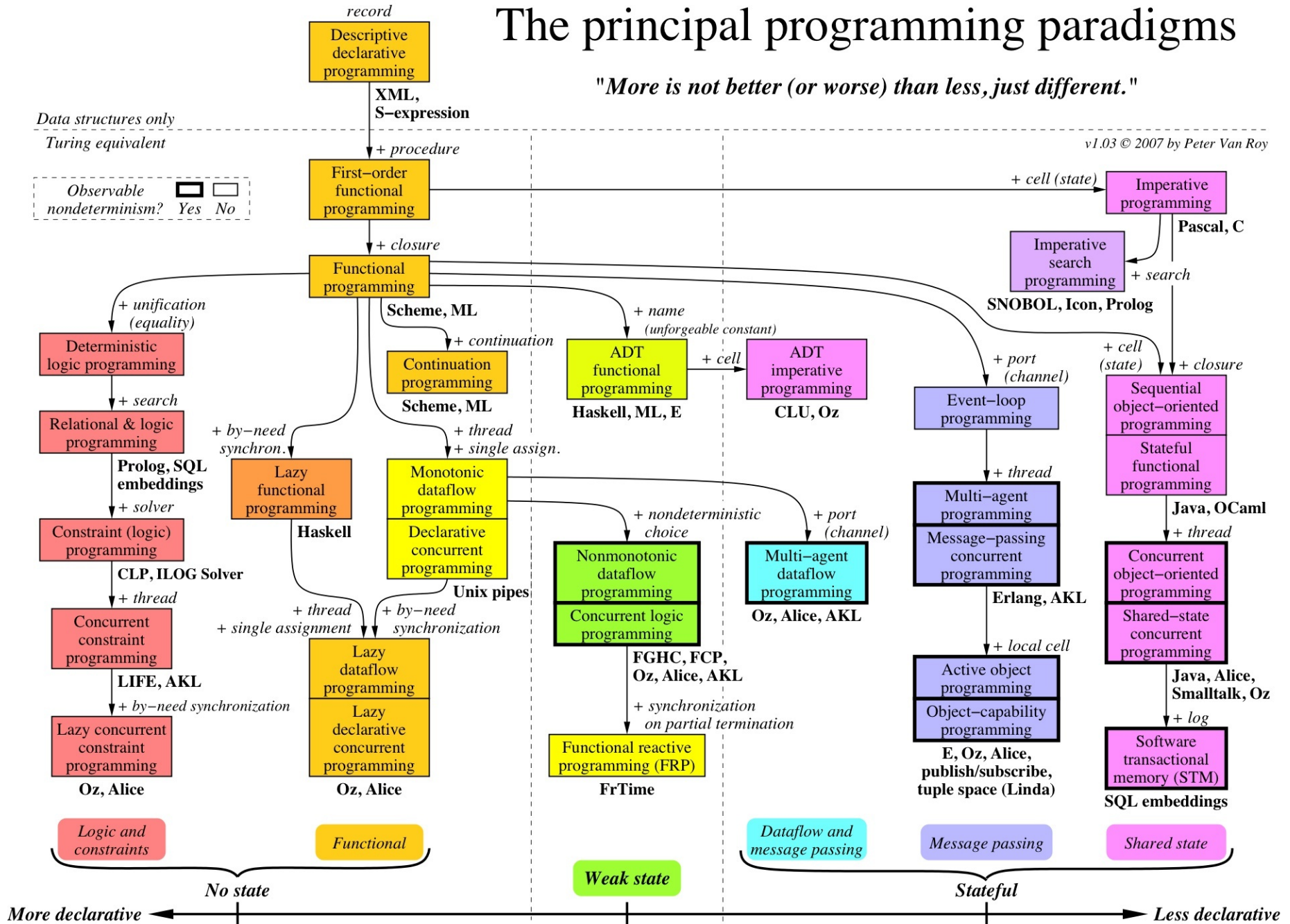
Imperativa språk tenderar att ge snabbare program...

...men kan kräva 10x mer kod

The principal programming paradigms

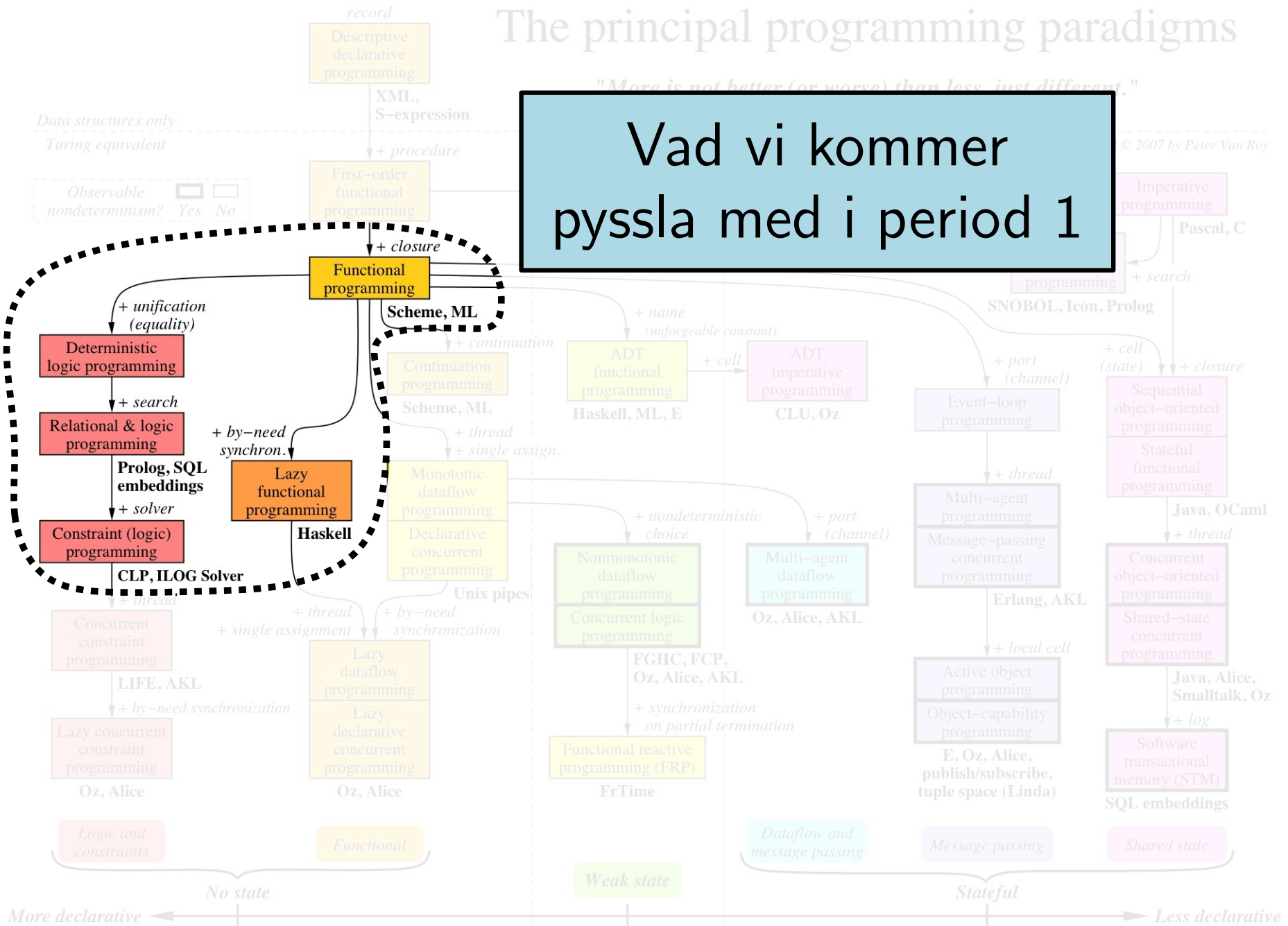
"More is not better (or worse) than less, just different."

v1.03 © 2007 by Peter Van Roy

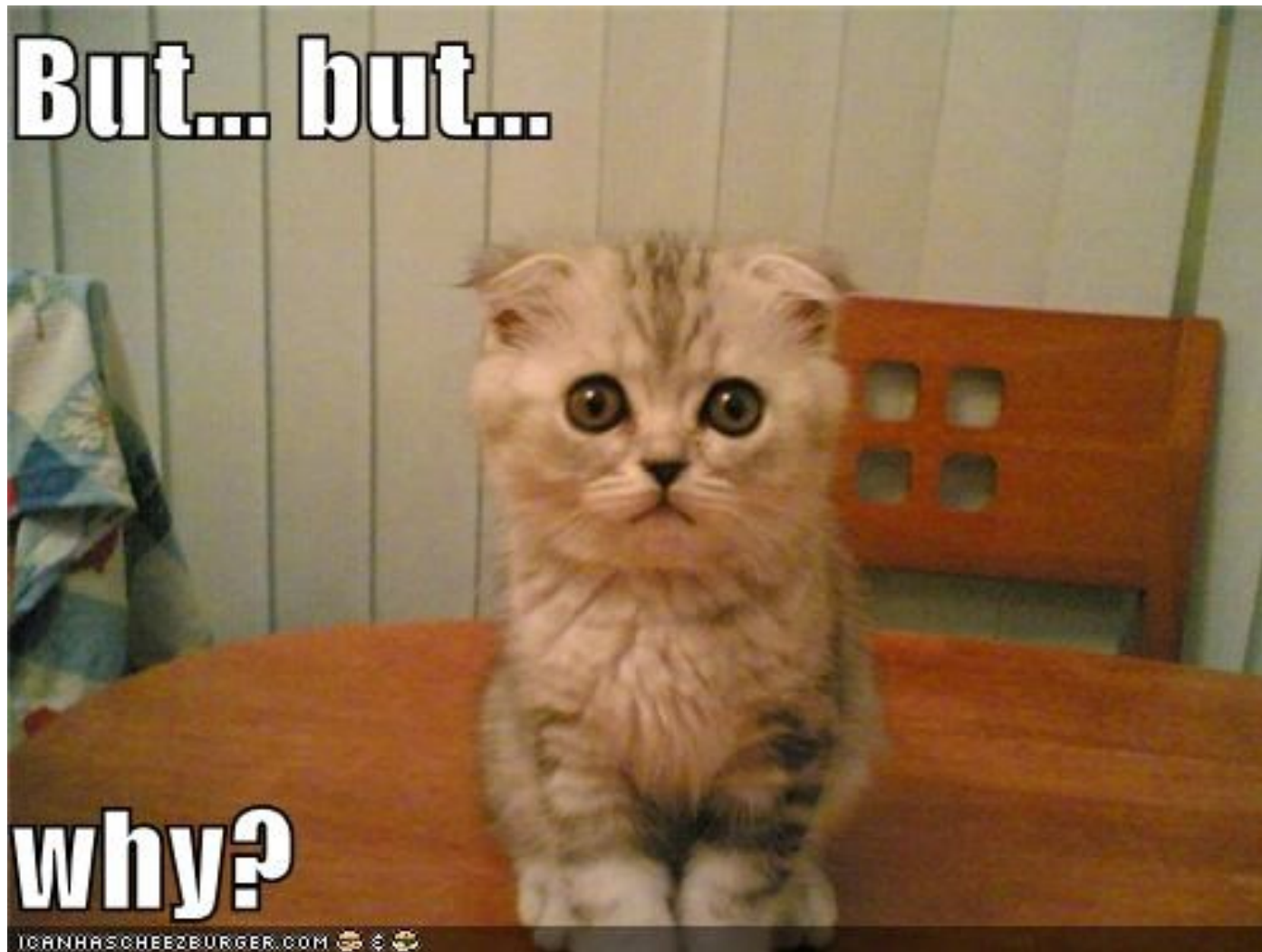


The principal programming paradigms

Vad vi kommer
pysla med i period 1



Vad ska nu detta vara bra för?



Vad ska nu detta vara bra för?

“It is essential for anyone who wants to be considered a professional in the areas of software to know several languages and several programming paradigms.”

— Bjarne Stroustrup (Skaparen av C++)



Vad ska nu detta vara bra för?

Rätt verktyg för rätt projekt – olika språk och paradigmer är bra på olika saker

Vad ska nu detta vara bra för?

Rätt verktyg för rätt projekt – olika språk och paradigmer är bra på olika saker

Bättre förståelse för programmering och algoritmik genom att få flera perspektiv på hur man kan tänka på programmering

Vad ska nu detta vara bra för?

Rätt verktyg för rätt projekt – olika språk och paradigmer är bra på olika saker

Bättre förståelse för programmering och algoritmik genom att få flera perspektiv på hur man kan tänka på programmering

Ni är här (= på datateknik på KTH) för att bli dataloger, inte för att lära er skriva enkla program i <populärast språk just nu>

Dagens föreläsning

Översikt om:

1. Ämnet

2. Lärarna

3. Kursformalia (betyg etc)

Lärarna

Carina Edlund:
funktionell programmering i Haskell



Lärarna

Carina Edlund:
funktionell programmering i Haskell



Dilian Gurov:
logik-programmering i Prolog



Lärarna

Carina Edlund:
funktionell programmering i Haskell



Dilian Gurov:
logik-programmering i Prolog



Alexander Baltatzis:
internet-programmering



Lärarna

Carina Edlund:
funktionell programmering i Haskell



Dilian Gurov:
logik-programmering i Prolog



Alexander Baltatzis:
internet-programmering



Per Austrin
syntaxanalys + kursledare



Dagens föreläsning

Översikt om:

1. Ämnet

2. Lärarna

3. Kursformalia (betyg etc)

Registrering

Registrera er på kursen!

- Registrering i "personliga menyn" i KTH Social (för KTH-studenter)

Registrering

Registrera er på kursen!

- Registrering i "personliga menyn" i KTH Social (för KTH-studenter)

Registrera er i Kattis-systemet:

- <https://kth.kattis.com/courses/DD1361/progp15>
- Logga in i systemet, klicka att ni går kursen

(Detta står under "Före kursstart" på kurshemsidan.)

Kursrepresentanter?

Labbgupper

För labbarna är ni indelade i tre grupper:

- Grupp A: alla med efternamn från A till och med H
- Grupp B: alla med efternamn från I till och med Q
- Grupp C: alla med efternamn från R till och med Ö

Labbgrupper

För labbarna är ni indelade i tre grupper:

- Grupp A: alla med efternamn från A till och med H
- Grupp B: alla med efternamn från I till och med Q
- Grupp C: alla med efternamn från R till och med Ö

Vecka 38 2015

Visa i Mitt
schema

● Mån 14 sep 10:00-12:00

Laboration



LABORATION

Bonusdatum. Redovisningar av labb F1 för grupp A prioriteras.

Plats: 5O1Spe, 5O2Spo

● Mån 14 sep 13:00-15:00

Laboration



LABORATION

Bonusdatum. Redovisningar av labb F1 för grupp B prioriteras.

Plats: 5O1Spe, 5O2Spo

● Tis 15 sep 13:00-15:00

Laboration



LABORATION

Bonusdatum. Redovisningar av labb F1 för grupp C prioriteras.

Plats: 5O1Spe, 5O2Spo

Labbgrupper

För labbarna är ni indelade i tre grupper:

- Grupp A: alla med efternamn från A till och med H
- Grupp B: alla med efternamn från I till och med Q
- Grupp C: alla med efternamn från R till och med Ö

Vecka 38 2015

Visa i Mitt
schema

För grupp B

● Mån 14 sep 10:00-12:00

Laboration



LABORATION

Bonusdatum. Redovisningar av labb F1 för grupp A prioriteras.

Plats: 5O1Spe, 5O2Spo

● Mån 14 sep 13:00-15:00

Laboration



LABORATION

Bonusdatum. Redovisningar av labb F1 för grupp B prioriteras.

Plats: 5O1Spe, 5O2Spo

● Tis 15 sep 13:00-15:00

Laboration



LABORATION

Bonusdatum. Redovisningar av labb F1 för grupp C prioriteras.

Plats: 5O1Spe, 5O2Spo

Labbgrupper

För labbarna är ni indelade i tre grupper:

- Grupp A: alla med efternamn från A till och med H
- Grupp B: alla med efternamn från I till och med Q
- Grupp C: alla med efternamn från R till och med Ö

Gå på vilka ni vill

Tors 17 sep 15:00-17:00

Laboration



LABORATION

Plats: 4V4Gul, 4V6Bru

Fre 18 sep 10:00-12:00

Laboration



LABORATION

Plats: 4V4Gul, 4V6Bru

Kurshemsida och KTH Social

Kurshemsida på KTH Social:

<https://www.kth.se/social/course/DD1361>



Kurshemsida och KTH Social

Kurshemsida på KTH Social:

<https://www.kth.se/social/course/DD1361>



För (nästan) alla typer av frågor, använd KTH Social

- Frågor om labbarna
- Jakt på labbpartner
- Etc...

Kurshemsida och KTH Social

Kurshemsida på KTH Social:

<https://www.kth.se/social/course/DD1361>



För (nästan) alla typer av frågor, använd KTH Social

- Frågor om labbarna
- Jakt på labbpartner
- Etc...

Undantag:

- “Personliga frågor”

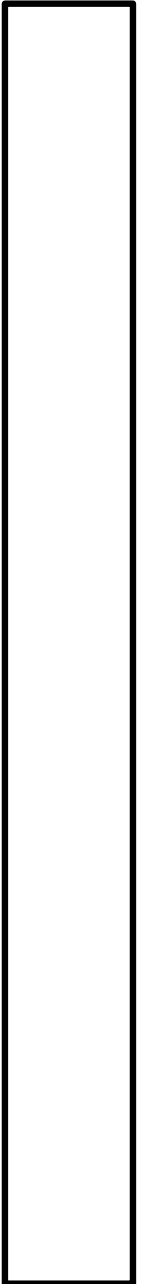
(t.ex. om era labbresultat inte rapporterats in)

För dessa: använd kursmailen progp-15@csc.kth.se

Kursöversikt

- 7.5 hp (dvs 200 timmar av ditt liv)

Två Ladok-moment:

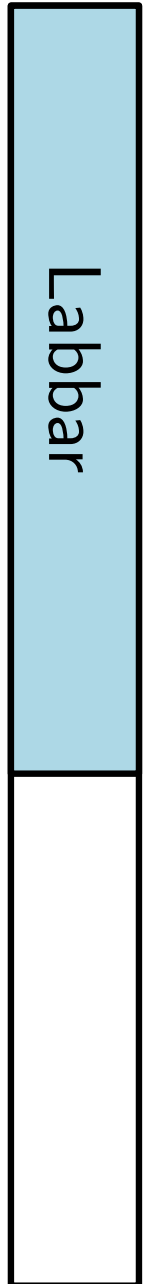


Kursöversikt

- 7.5 hp (dvs 200 timmar av ditt liv)

Two Ladok-moment:

- Laborationer: 4.5 hp



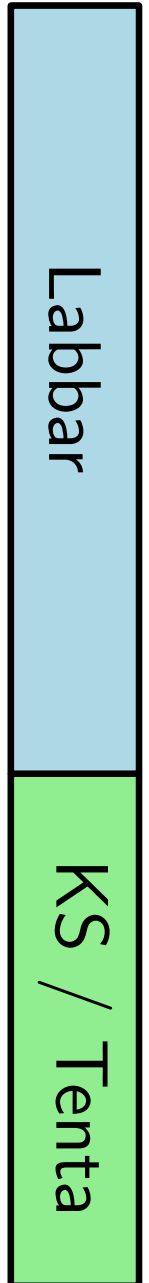
Kursöversikt

- 7.5 hp (dvs 200 timmar av ditt liv)

Två Ladok-moment:

- Laborationer: 4.5 hp

- Tenta: 3 hp



Kursdelar

Funktionell programmering i Haskell

Logik-programmering i Prolog

Syntaxanalys

Internet-programmering

Extralabbar

Labbar

KS / Tenta

Kursdelar

Funktionell programmering i Haskell

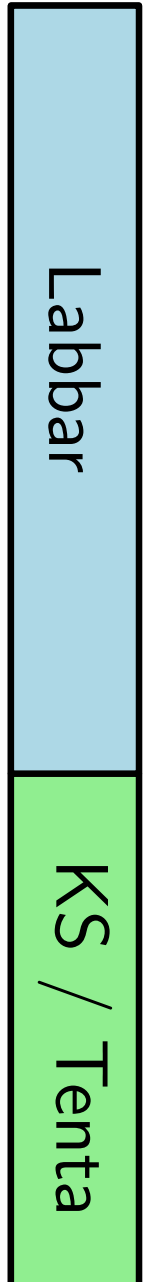
- 2 obligatoriska labbar
- 1 betygshöjande labb
- Kontrollskrivning / Tenta

Logik-programmering i Prolog

Syntaxanalys

Internet-programmering

Extralabbar



Kursdelar

Funktionell programmering i Haskell

- 2 obligatoriska labbar
- 1 betygshöjande labb
- Kontrollskrivning / Tenta

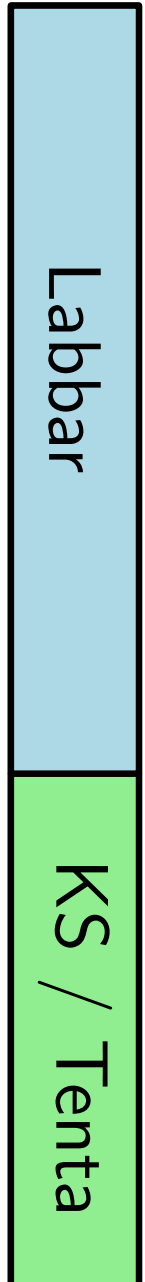
Logik-programmering i Prolog

- 2 obligatoriska labbar
- 1 betygshöjande labb
- Kontrollskrivning / Tenta

Syntaxanalys

Internet-programmering

Extralabbar



Kursdelar

Funktionell programmering i Haskell

- 2 obligatoriska labbar
- 1 betygshöjande labb
- Kontrollskrivning / Tenta

Logik-programmering i Prolog

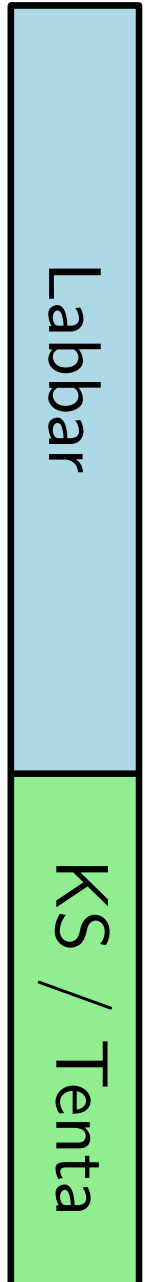
- 2 obligatoriska labbar
- 1 betygshöjande labb
- Kontrollskrivning / Tenta

Syntaxanalys

- 2 obligatoriska labbar
- 2 betygshöjande labbar
- Kontrollskrivning / Tenta

Internet-programmering

Extralabbar



Kursdelar

Funktionell programmering i Haskell

- 2 obligatoriska labbar
- 1 betygshöjande labb
- Kontrollskrivning / Tenta

Logik-programmering i Prolog

- 2 obligatoriska labbar
- 1 betygshöjande labb
- Kontrollskrivning / Tenta

Syntaxanalys

- 2 obligatoriska labbar
- 2 betygshöjande labbar
- Kontrollskrivning / Tenta

Internet-programmering

- 1 obligatorisk labb

Extralabbar

Labbar

KS / Tenta

Kursdelar

Funktionell programmering i Haskell

- 2 obligatoriska labbar
- 1 betygshöjande labb
- Kontrollskrivning / Tenta

Logik-programmering i Prolog

- 2 obligatoriska labbar
- 1 betygshöjande labb
- Kontrollskrivning / Tenta

Syntaxanalys

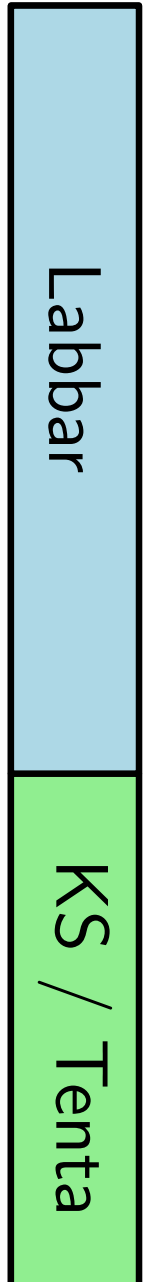
- 2 obligatoriska labbar
- 2 betygshöjande labbar
- Kontrollskrivning / Tenta

Internet-programmering

- 1 obligatorisk labb

Extralabbar

- 2 betygshöjande labbar



Kursdelar

Funktionell programmering i Haskell

- 2 obligatoriska labbar
- 1 betygshöjande labb
- Kontrollskrivning / Tenta

Logik-programmering i Prolog

- 2 obligatoriska labbar
- 1 betygshöjande labb
- Kontrollskrivning / Tenta

Syntaxanalys

- 2 obligatoriska labbar
- 2 betygshöjande labbar
- Kontrollskrivning / Tenta

Internet-programmering

- 1 obligatorisk labb

Extralabbar

- 2 betygshöjande labbar

Totalt:

- 7 obligatoriska labbar
- 6 betygshöjande labbar
- 3 kontrollskrivningar

Labbar

KS / Tenta

Kursdelar

Funktionell programmering i Haskell

- 2 obligatoriska labbar
- 1 betygshöjande labb
- Kontrollskrivning / Tenta

Logik-programmering i Prolog

- 2 obligatoriska labbar
- 1 betygshöjande labb
- Kontrollskrivning / Tenta

PERIOD 1

Syntaxanalys

- 2 obligatoriska labbar
- 2 betygshöjande labbar
- Kontrollskrivning / Tenta

Internet-programmering

- 1 obligatorisk labb

Extralabbar

- 2 betygshöjande labbar

Labbar

KS / Tenta

Kursdelar

Funktionell programmering i Haskell

- 2 obligatoriska labbar
- 1 betygshöjande labb
- Kontrollskrivning / Tenta

Logik-programmering i Prolog

- 2 obligatoriska labbar
- 1 betygshöjande labb
- Kontrollskrivning / Tenta

Syntaxanalys

- 2 obligatoriska labbar
- 2 betygshöjande labbar
- Kontrollskrivning / Tenta

Internet-programmering

- 1 obligatorisk labb

Extralabbar

- 2 betygshöjande labbar

PERIOD 2

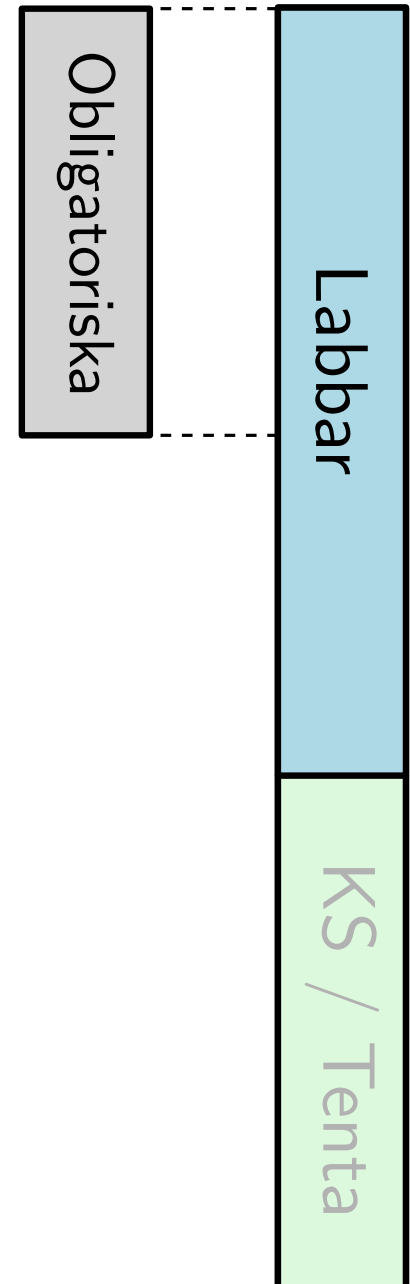
Labbar

KS / Tenta

Labbarna

7 obligatoriska labbar

- varje obligatorisk labb har ett “bonusdatum”: redovisar man labben innan bonusdatum får man ett bonuspoäng till KS / Tenta



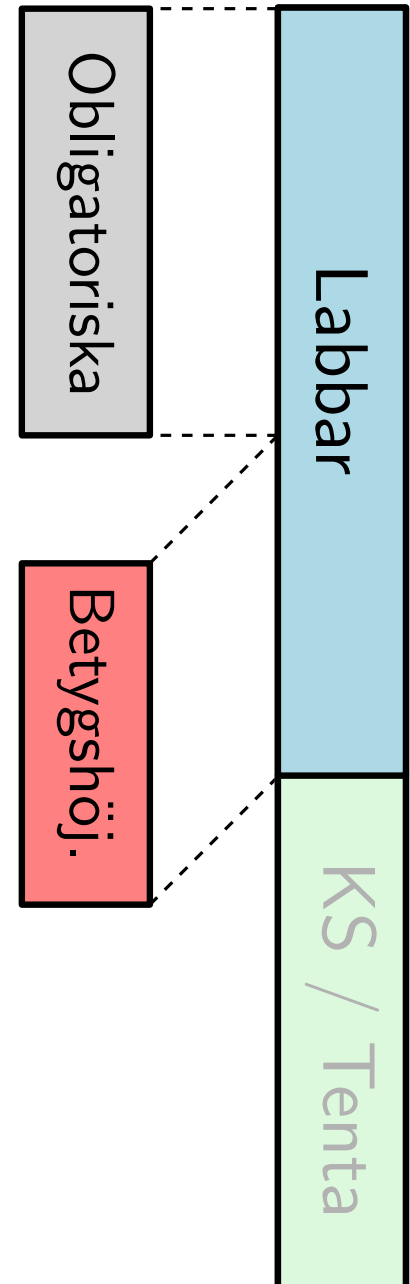
Labbarna

7 obligatoriska labbar

- varje obligatorisk labb har ett “bonusdatum”: redovisar man labben innan bonusdatum får man ett bonuspoäng till KS / Tenta

6 betygshöjande labbar

- antal betygshöjande labbar man gör avgör betyget



Labbarna

7 obligatoriska labbar

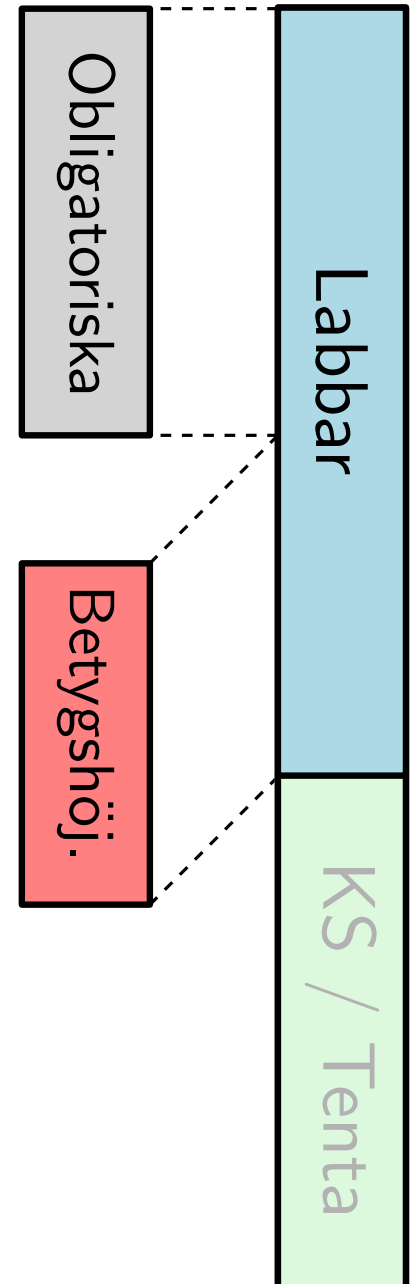
- varje obligatorisk labb har ett “bonusdatum”: redovisar man labben innan bonusdatum får man ett bonuspoäng till KS / Tenta

6 betygshöjande labbar

- antal betygshöjande labbar man gör avgör betyget

Arbeta helst två och två
(arbeta själv också tillåtet)

Skriv ut labbkvitto från hemsidan



Labbarna

Labbarna i funktionell programmering finns uppe på kurshemsidan.

De flesta labbarna rättas i systemet Kattis:
(<https://kth.kattis.com/>)

Labbar

KS / Tenta

Labbarna

Labbarna i funktionell programmering finns uppe på kurshemsidan.

De flesta labbarna rättas i systemet Kattis:
(<https://kth.kattis.com/>)

1. Ni skickar in er kod till Kattis
2. Kattis testkör er kod på en bunt (hemliga) testfall
3. Kattis talar om ifall er kod är godkänd, om den var för långsam, om den kraschade, om den gjorde fel...
4. När ni blivit godkända på labben i Kattis redovisar ni er lösning för en handledare i labbsal

Labbar

KS / Tenta

Labbarna

Labbarna i funktionell programmering finns uppe på kurshemsidan.

De flesta labbarna rättas i systemet Kattis:
(<https://kth.kattis.com/>)

1. Ni skickar in er kod till Kattis
2. Kattis testkör er kod på en bunt (hemliga) testfall
3. Kattis talar om ifall er kod är godkänd, om den var för långsam, om den kraschade, om den gjorde fel...
4. När ni blivit godkända på labben i Kattis redovisar ni er lösning för en handledare i labbsal

Kattis är väldigt snål med information om vad ni har för fel! Det är ert jobb att hitta detta!

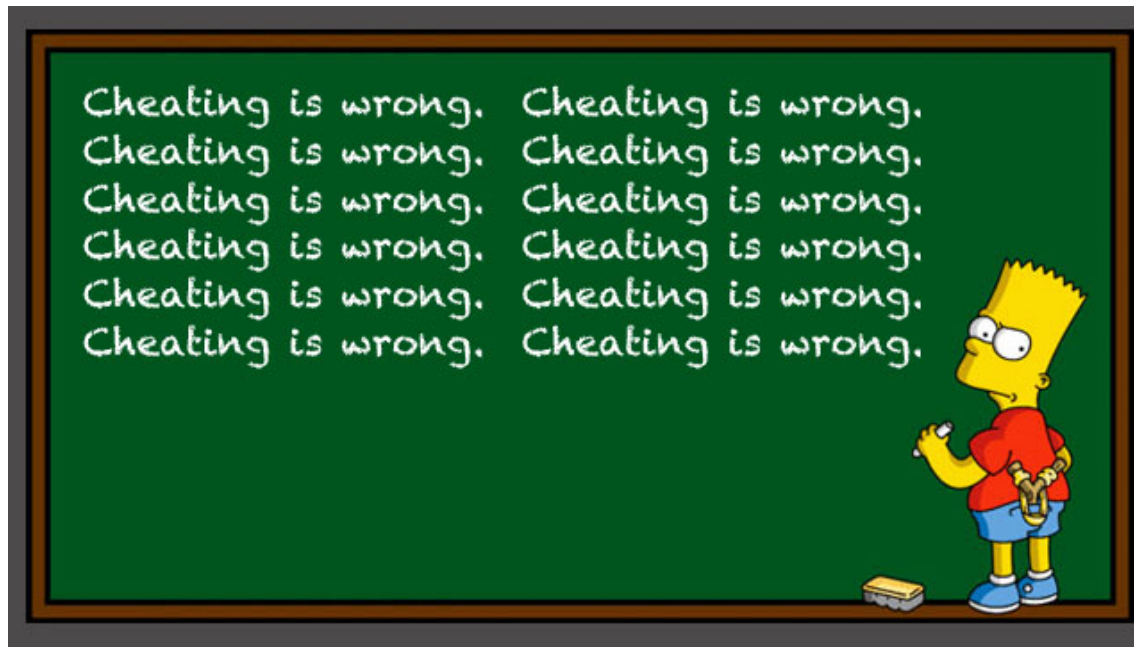
Dokumentations-krav

1. Det ska vara **tydligt dokumenterat** i kommentar högst upp i koden **vilka som har skrivit koden**. Detta gäller **alla inskickningar ni gör till Kattis**, och är inte något ni kan lägga till i slutet när ni väl fått er kod att bli godkänd av Kattis.
2. Själva koden ska vara ordentligt kommenterad. Syftet med olika funktioner/predikat som ni definierar ska förklaras. (Detta kan, till skillnad från punkt 1, däremot läggas till i efterhand när ni väl fått koden att fungera, som förberedelse för redovisning.)

Hederskodex

Alla kurser på CSC följer skolans hederskodex

<http://www.kth.se/csc/student/hederskodex>



Man anger vilka källor man tagit hjälp av,
man ger inte sina lösningar till sina medstudenter,
man gör inte sina medstudenters jobb åt dem,
etc...

KS / Tenta

KS/Tenta på kursdelarna **funktionell programmering**, **logikprogrammering**, och **syntaxanalys**

Labbar

KS / Tenta

KS / Tenta

KS/Tenta på kursdelarna **funktionell programmering**, **logikprogrammering**, och **syntaxanalys**

KS:

- 24 sep: Funktionell programmering
- 26 okt: Logikprogrammering
- 1 dec: Syntaxanalys

Labbar

KS / Tenta

KS / Tenta

KS/Tenta på kursdelarna **funktionell programmering**, **logikprogrammering**, och **syntaxanalys**

KS:

- 24 sep: **Funktionell programmering**
- 26 okt: **Logikprogrammering**
- 1 dec: **Syntaxanalys**

15 jan: Tenta!

Labbar

KS / Tenta

KS / Tenta

KS/Tenta på kursdelarna **funktionell programmering**, **logikprogrammering**, och **syntaxanalys**

KS:

- 24 sep: **Funktionell programmering**
- 26 okt: **Logikprogrammering**
- 1 dec: **Syntaxanalys**

15 jan: Tenta!

Klarat en KS \Rightarrow behöver inte göra den delen på tentan

Labbar

KS / Tenta

KS / Tenta

KS/Tenta på kursdelarna **funktionell programmering**, **logikprogrammering**, och **syntaxanalys**

KS:

- 24 sep: **Funktionell programmering**
- 26 okt: **Logikprogrammering**
- 1 dec: **Syntaxanalys**

15 jan: Tenta!

Klarat en KS \Rightarrow behöver inte göra den delen på tentan

Klarat alla tre KS \Rightarrow behöver inte skriva tentan alls

Labbar

KS / Tenta

Bonuspoäng

De bonuspoäng ni får (upp till 7 stycken om ni gör alla de obligatoriska labbarna innan deadline) används på KS / Tenta:

Bonuspoäng

De bonuspoäng ni får (upp till 7 stycken om ni gör alla de obligatoriska labbarna innan deadline) används på KS / Tenta:

- Bonuspoäng från **logikprogrammering**, **funktionell programmering**, och **syntaxanalys** ger bonuspoäng på respektive KS (*ej på tentan!*)

Bonuspoäng

De bonuspoäng ni får (upp till 7 stycken om ni gör alla de obligatoriska labbarna innan deadline) används på KS / Tenta:

- Bonuspoäng från **logikprogrammering**, **funktionell programmering**, och **syntaxanalys** ger bonuspoäng på respektive KS (*ej på tentan!*)
- Bonuspoäng från Inet-labben ger bonuspoäng på tentan (*ej på omtentan!*)

Betyg

Krav för godkänt:

- Alla 7 obligatoriska labbarna godkända
- Tenta godkänd

Betyg

Krav för godkänt:

- Alla 7 obligatoriska labbarna godkända
- Tenta godkänd

Betyg:

#Betygshöjande labbar	0	1	2	3	4	5	6
Betyg	E	E	D	C	B	A	A

(Om kravet för godkänt uppnått. För betyg A krävs dessutom att vissa specifika betygshöjande labbar gjorda)

Avslutningsvis...

Nästa föreläsning (den första “riktiga” föreläsningen) är inte förrän om 10 dagar.

Den här terminen för CDATE-programmet är **TUNG**.

Försök komma igång med den här kursen (och era andra kurser) snarast, även om ni inte har så mycket schemalagd undervisning! Ni kommer tacka er själva efteråt...

För progp: kom igång med första labben (labb F1) genom att:

1. Öppna labb-kompendiet, läsa instruktionerna och uppgiften
2. Läs sidan ”Komma igång” (under ”Laborationer”)
3. Kolla föreläsningssanteckningarna för nästa föreläsning (finns på kurshemsidan), sök mer info på Internet
4. Försök göra några av deluppgifterna i F1

Frågor?