DD1361
Programmeringsparadigm
Föreläsning 1: Intro

Per Austrin

KTH

2016-08-30

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 LaTeX
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 F2 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
$\approx$
att programmera

Programmeringsparadigm! (?)

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

En programmeringsparadigm är en uppsättning programmeringskoncept.

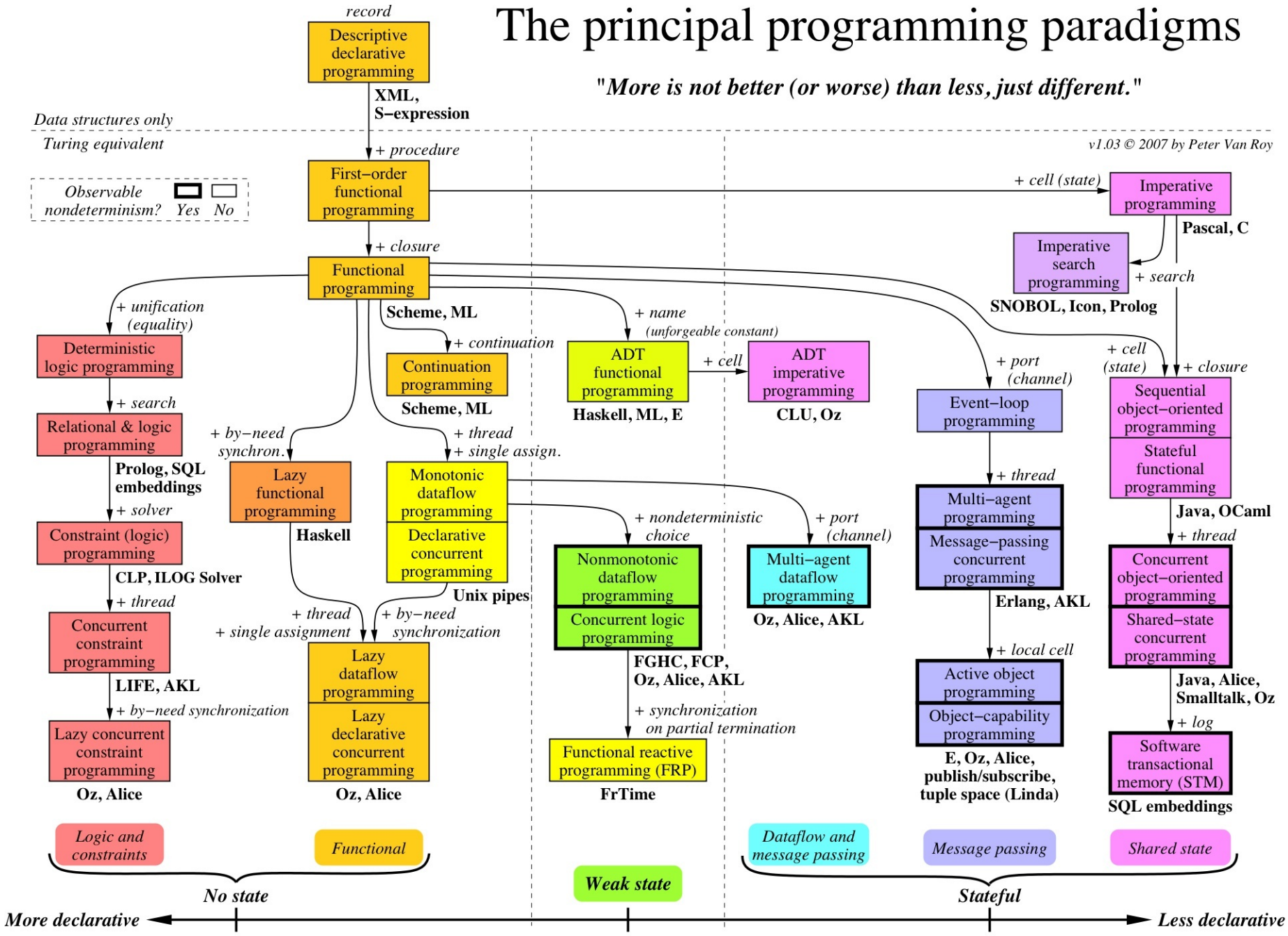
Exempel på koncept:

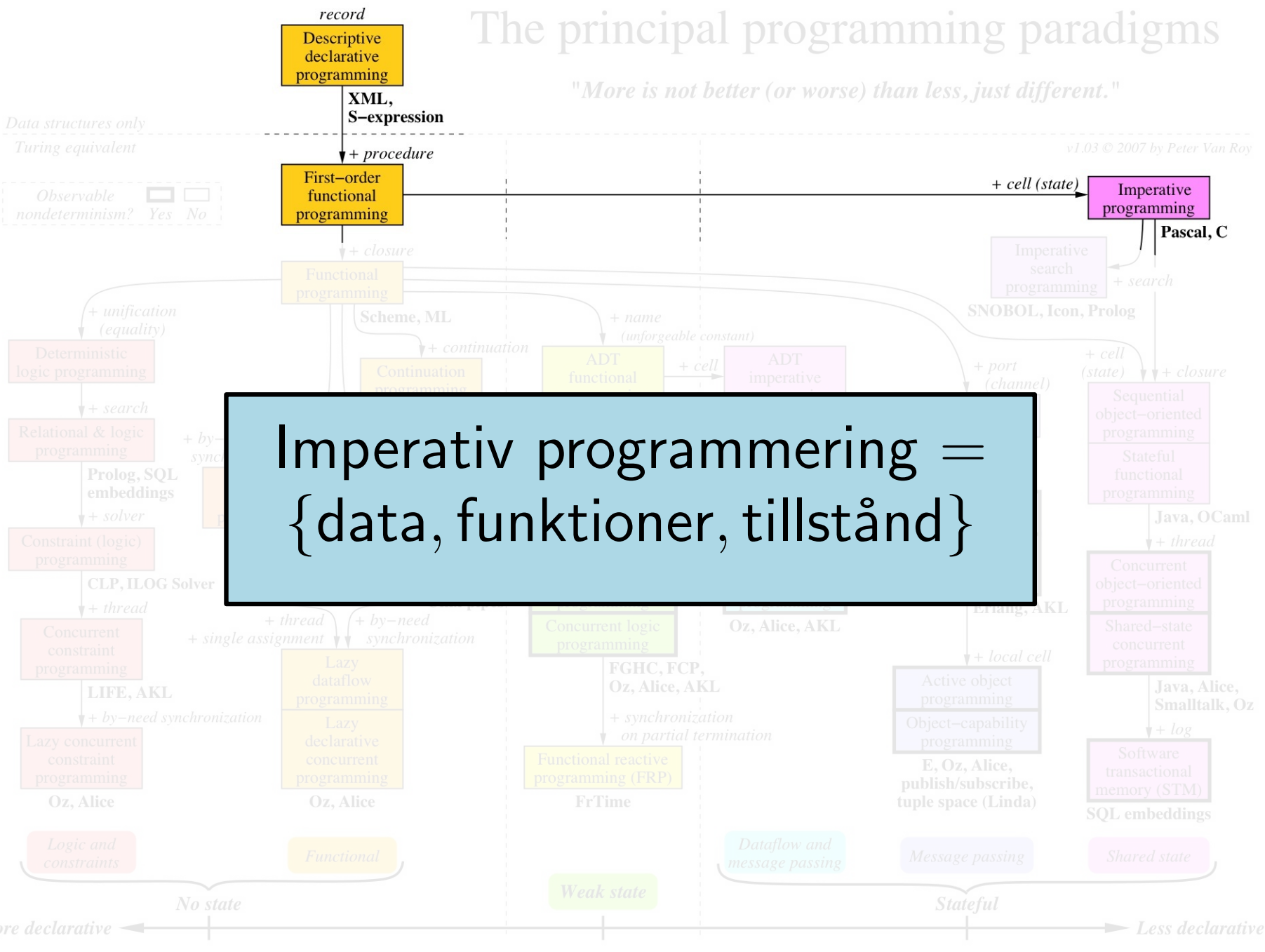
- trådar / parallelism
- objekt (á la objektorientering)

Ett programmeringsspråk som har alla paradigmens koncept tillhör paradigmen

The principal programming paradigms

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





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 *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*.

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

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

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

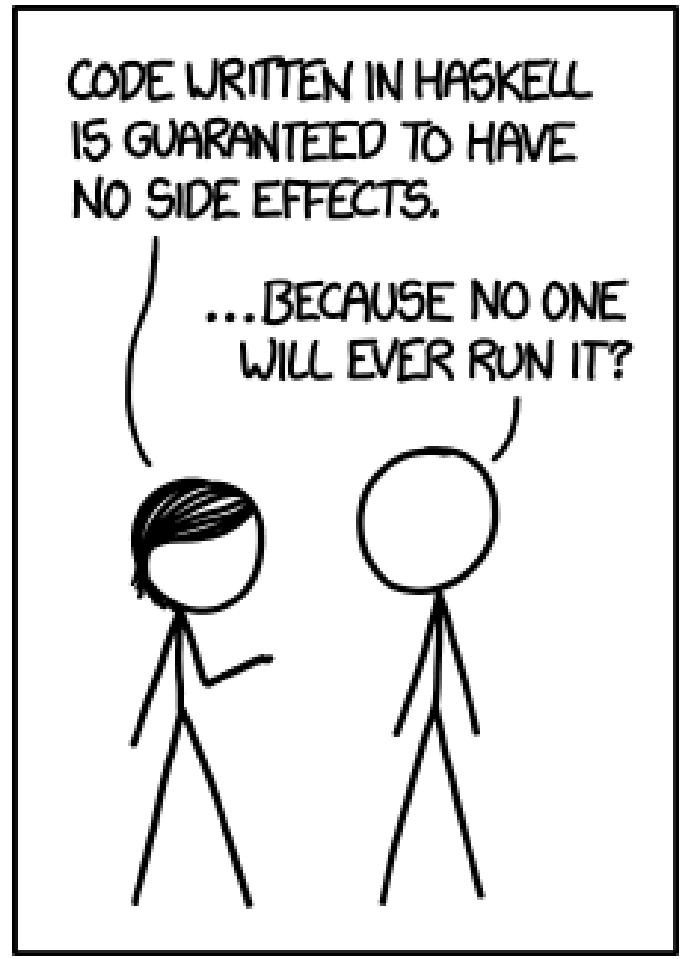
$$\begin{cases} L_1 & \text{if } \mathsf{length}(L_2) = 0 \\ L_2 & \text{if } \mathsf{length}(L_1) = 0 \\ \mathsf{head}(L_1) | \mathsf{merge}(\mathsf{tail}(L_1), L_2) & \text{if } \mathsf{head}(L_1) \leq \mathsf{head}(L_2) \\ \mathsf{head}(L_2) | \mathsf{merge}(L_1, \mathsf{tail}(L_2)) & \text{if } \mathsf{head}(L_1)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

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

Del 1 ($\approx$ September): Funktionell programmering

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

(xkcd #1270)



(xkcd #1312)

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 sort(L, S) som är sant om S är sorteringen av L.

sort$(L, S) =$ sorted(S) **and** 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 sort(L, S) som är sant om S är sorteringen av L.

sort(L, S) = sorted(S) **and** permutation(L, S)

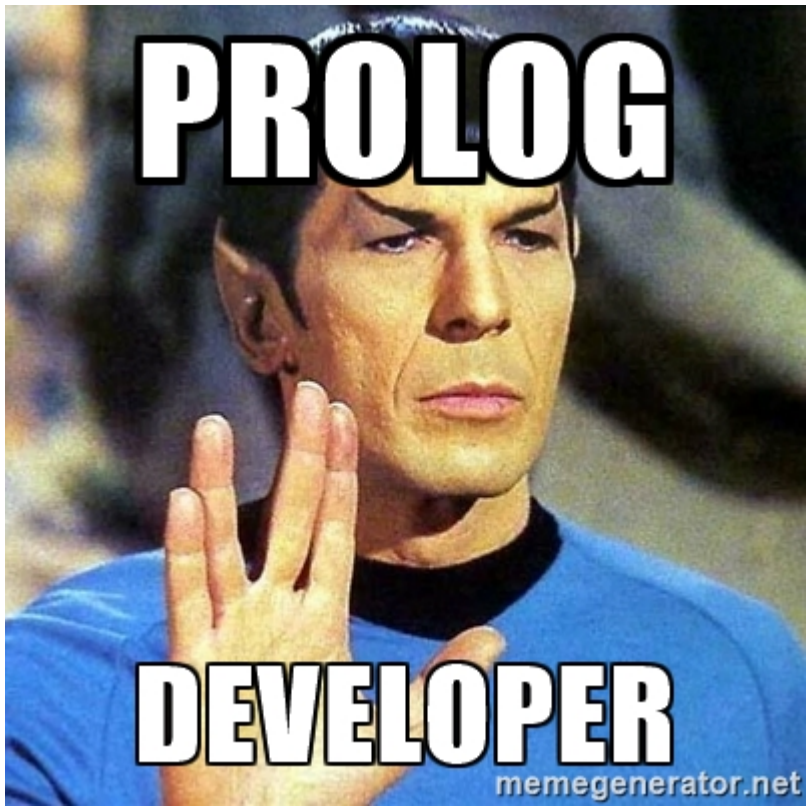
sort$([5, 12, 43, 1], S)$ = **true**

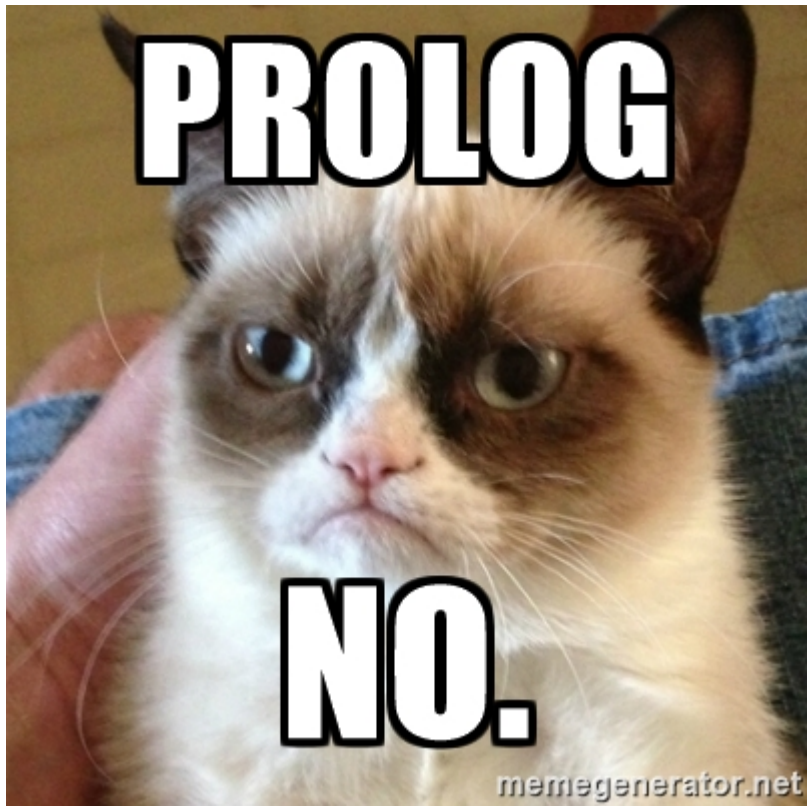
Medför att S måste vara $[1, 5, 12, 43]$

Del 2 (≈ Oktober): Logikprogrammering

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

Vanligast: första ordningens predikatlogik





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 paradigm

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...

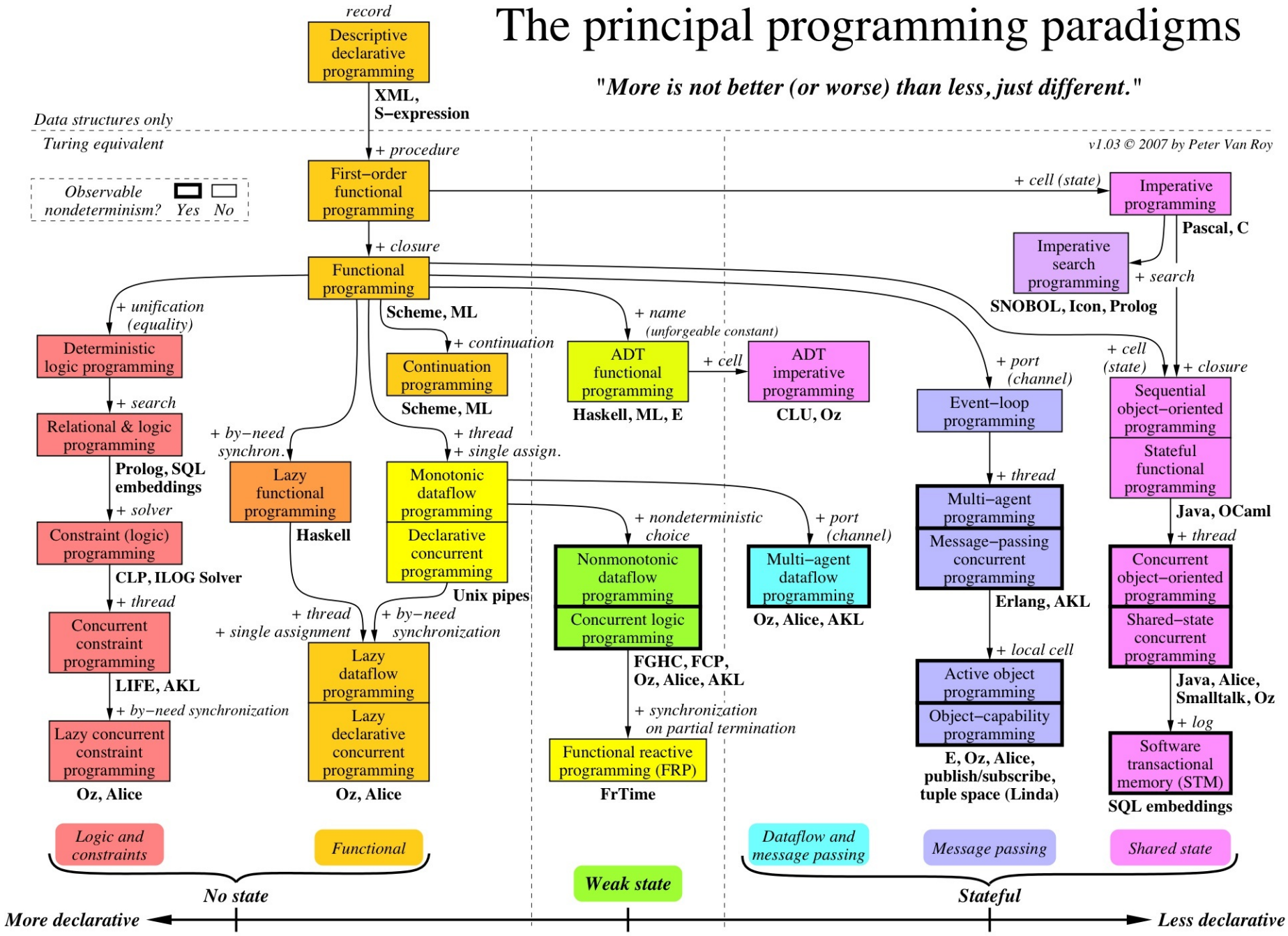
...men det gör dem också svårare att översätta till maskinkod

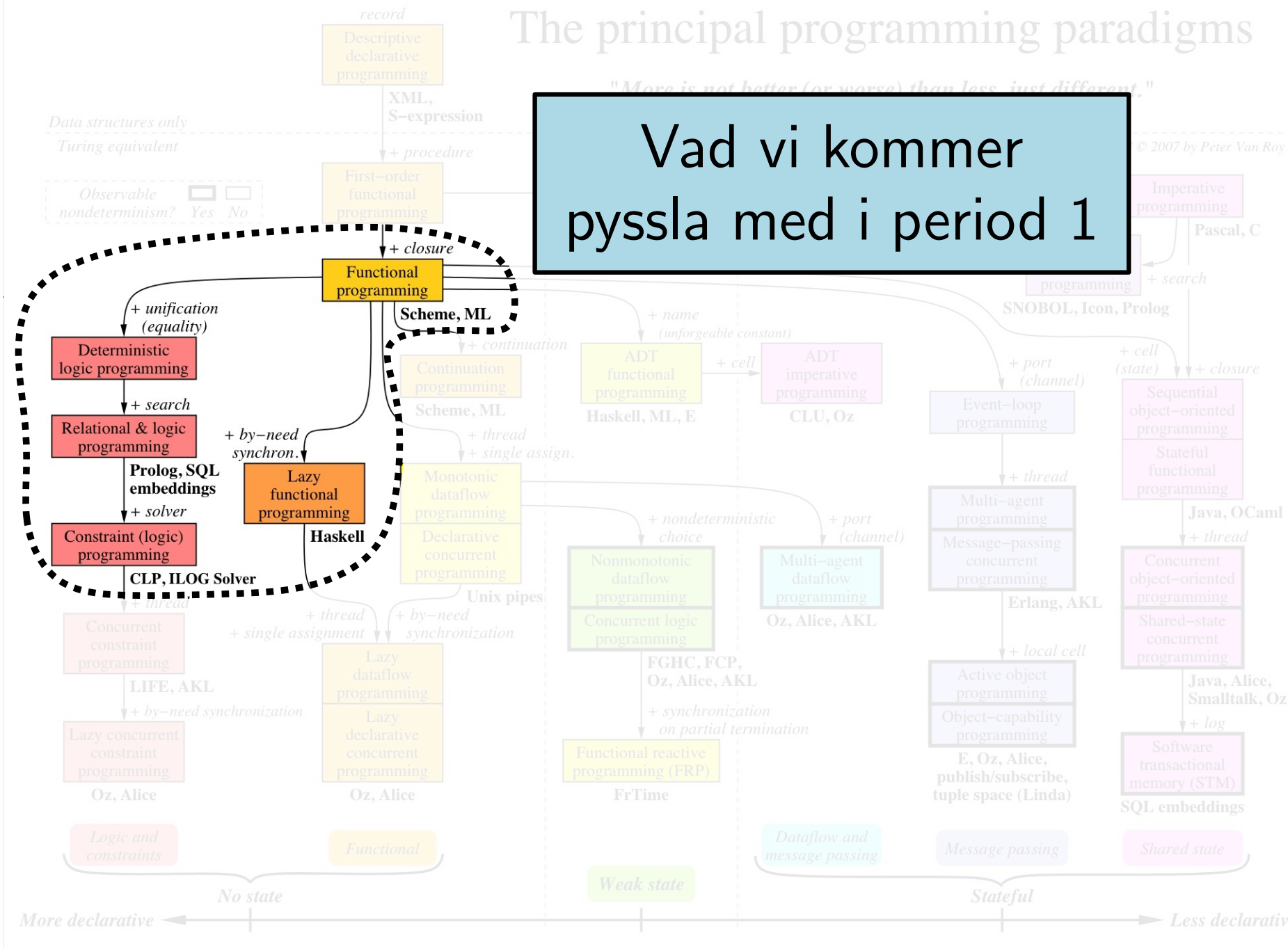
Imperativa språk tenderar att ge snabbare program...

...men kan kräva 10x mer kod

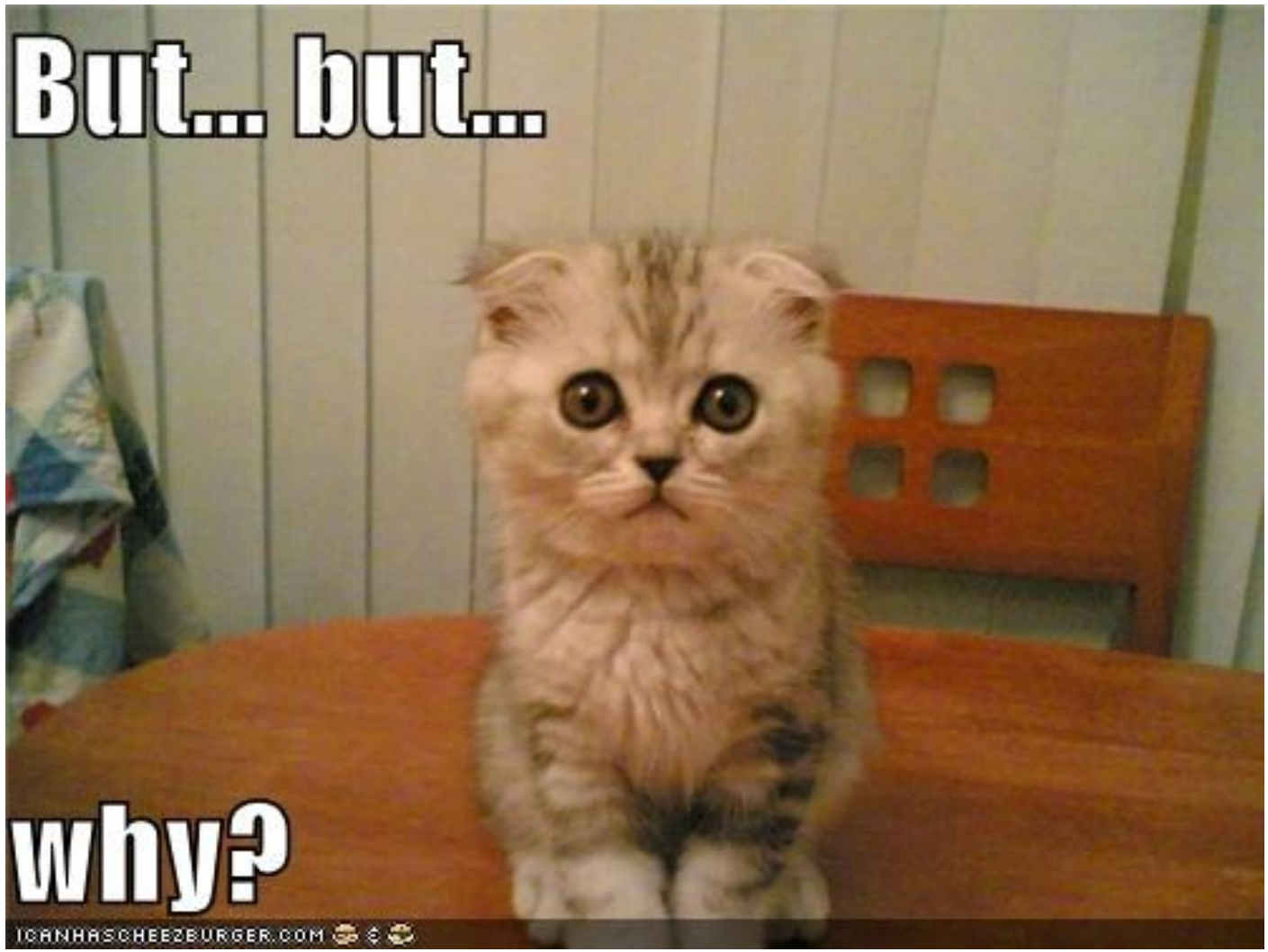
The principal programming paradigms

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





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 paradigm är bra på olika saker

Vad ska nu detta vara bra för?

Rätt verktyg för rätt projekt – olika språk och paradigm 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 paradigm 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

Marcus Dicander:
funktionell programmering i Haskell

Lärarna

Marcus Dicander:
funktionell programmering i Haskell

Dilian Gurov:
logik-programmering i Prolog

Lärarna

Marcus Dicander:
funktionell programmering i Haskell

Dilian Gurov:
logik-programmering i Prolog

Alexander Baltatzis:
internet-programmering

Lärarna

Marcus Dicander:
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/progp16`
- Logga in i systemet, klicka att ni går kursen

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

Kursrepresentanter?

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:

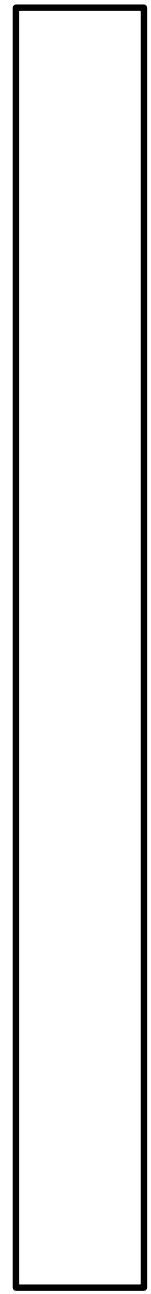
- "Personliga frågor"
  (t.ex. om era labbresultat inte rapporterats in)

För dessa: använd kursmailen `progp-16@csc.kth.se`

Kursöversikt

- 7.5 hp (dvs 200 timmar av ditt liv)

Två Ladok-moment:

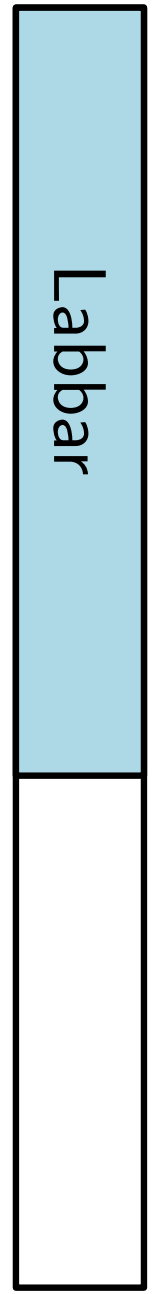


Kursöversikt

- 7.5 hp (dvs 200 timmar av ditt liv)

Två Ladok-moment:

- Laborationer: 4.5 hp



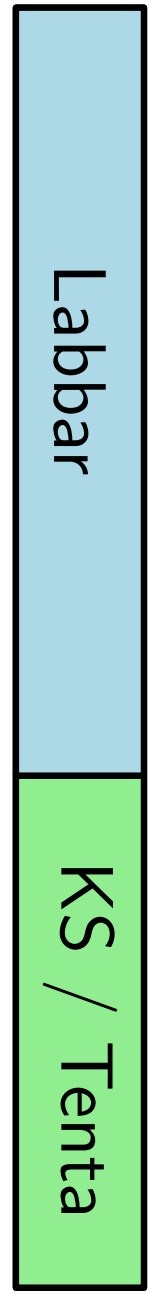
Kursöversikt

- 7.5 hp (dvs 200 timmar av ditt liv)

Två Ladok-moment:

- Laborationer: 4.5 hp

- Tenta: 3 hp



Kursdelar

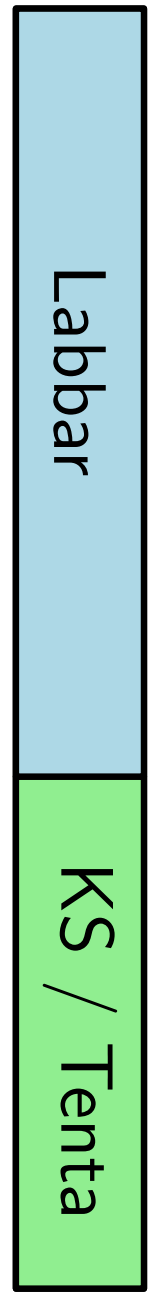
Funktionell programmering i Haskell

Logik-programmering i Prolog

Syntaxanalys

Internet-programmering

Extralabbar



Kursdelar

Funktionell programmering i Haskell

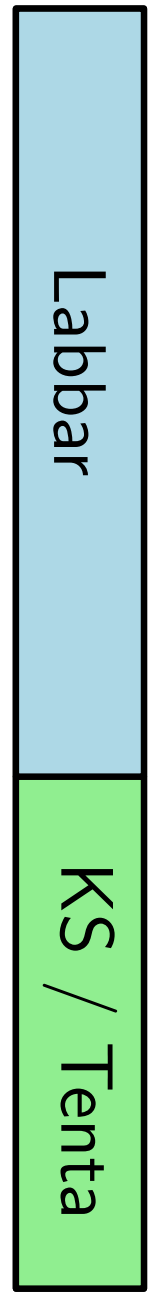
- 2 obligatoriska labbar
- 1 betygshöjande labb
- Kontrollskrivning / Tenta

Logik-programmering i Prolog

Syntaxanalys

Internet-programmering

Extralabbar



Kursdelar

Funktionell programmering i Haskell

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

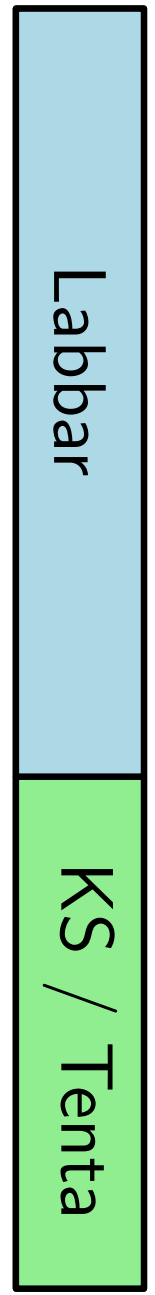
Logik-programmering i Prolog

- 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

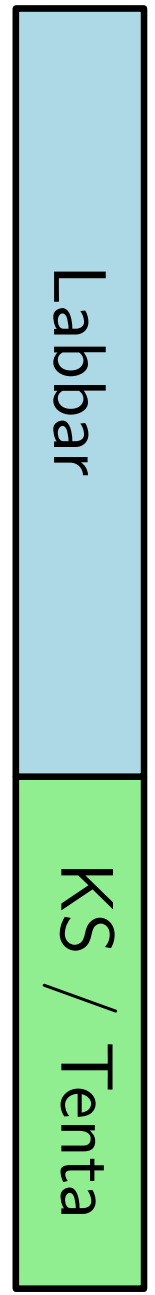
- 2 obligatoriska labbar
- 1 betygshöjande labb
- Kontrollskrivning / Tenta

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

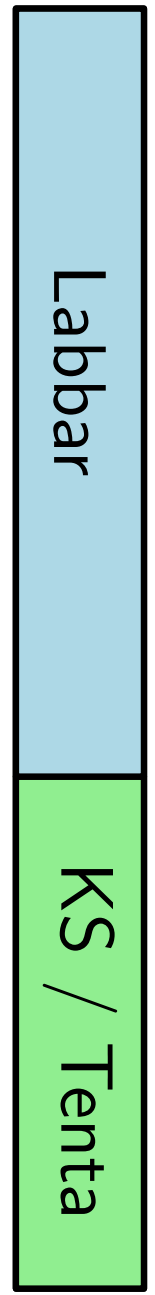
Syntaxanalys

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

Internet-programmering

- 1 obligatorisk labb

Extralabbar



Kursdelar

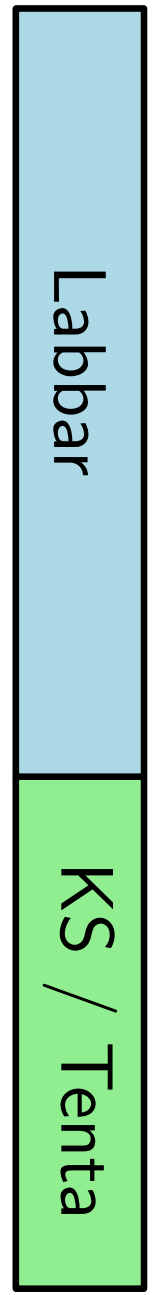
Funktionell programmering i Haskell
- 2 obligatoriska labbar
- 1 betygshöjande labb
- Kontrollskrivning / Tenta

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

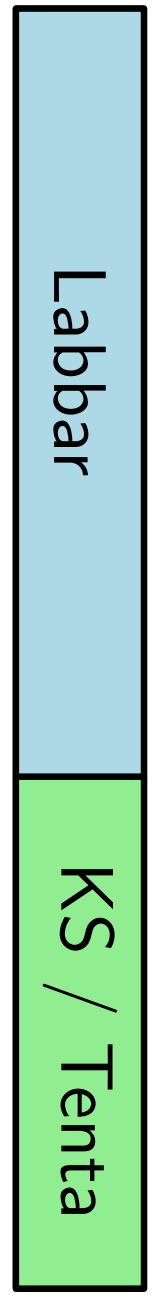
- 1 obligatorisk labb

Extralabbar

- 2 betygshöjande labbar

Totalt:

- 7 obligatoriska labbar
- 6 betygshöjande labbar
- 3 kontrollskrivningar



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

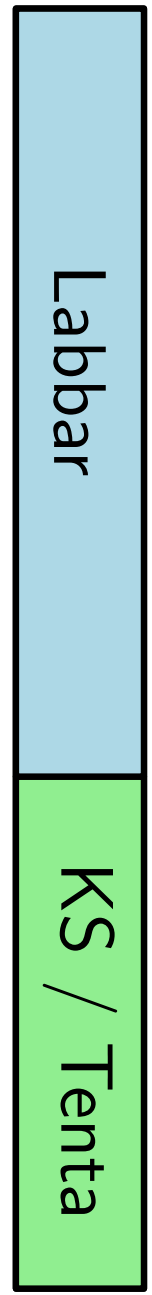
- 2 obligatoriska labbar
- 2 betygshöjande labbar
- Kontrollskrivning / Tenta

Internet-programmering

- 1 obligatorisk labb

Extralabbar

- 2 betygshöjande labbar



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

PERIOD 2

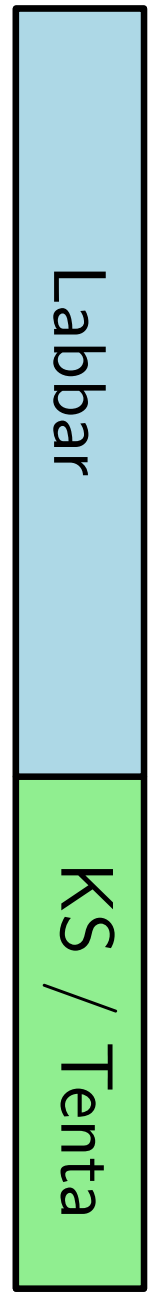
- 2 obligatoriska labbar
- 2 betygshöjande labbar
- Kontrollskrivning / Tenta

Internet-programmering

- 1 obligatorisk labb

Extralabbar

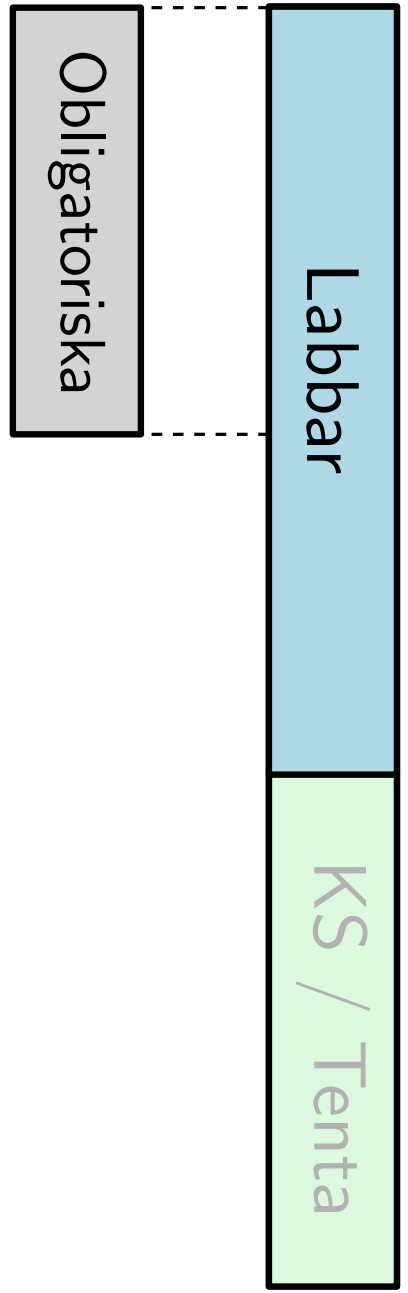
- 2 betygshöjande labbar



Labbarna

7 obligatoriska labbar

- varje obligatorisk labb har ett "bonusdatum": lämnar man in labben innan bonusdatum får man ett bonuspoäng till KS / Tenta



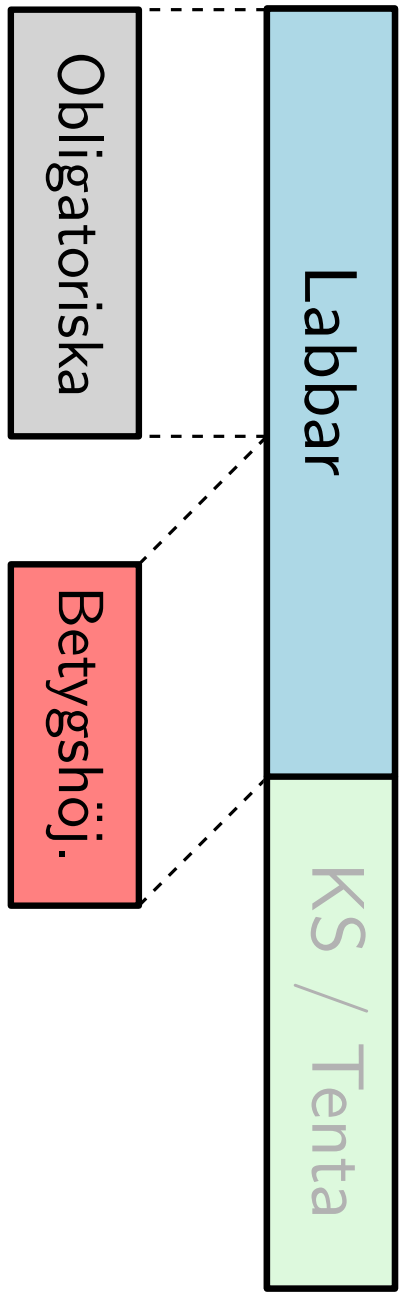
Labbarna

7 obligatoriska labbar

- varje obligatorisk labb har ett "bonusdatum": lämnar man in labben innan bonusdatum får man ett bonuspoäng till KS / Tenta

6 betygshöjande labbar

- antal betygshöjande labbar man gör avgör betyget



Labbarna

7 obligatoriska labbar

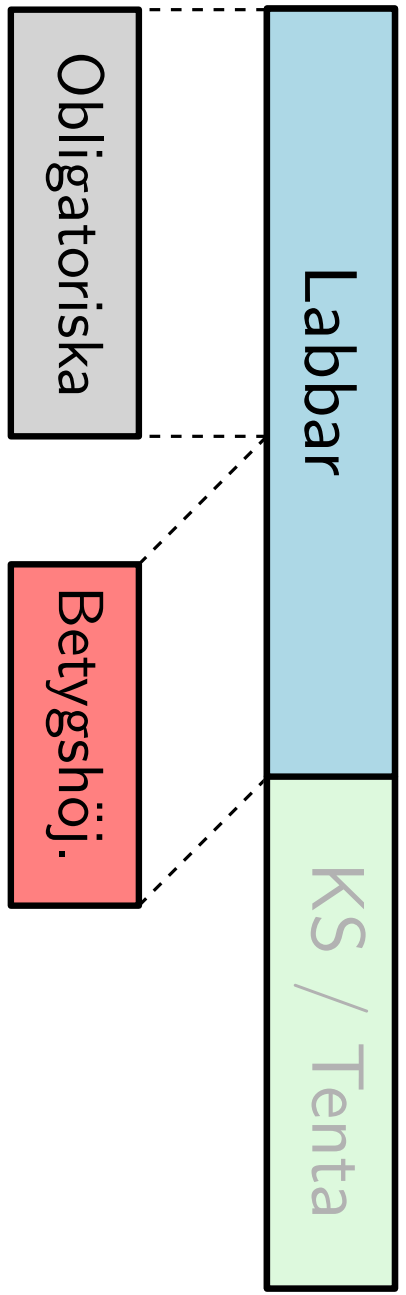
- varje obligatorisk labb har ett "bonusdatum": lämnar man in labben innan bonusdatum får man ett bonuspoäng till KS / Tenta

6 betygshöjande labbar

- antal betygshöjande labbar man gör avgör betyget

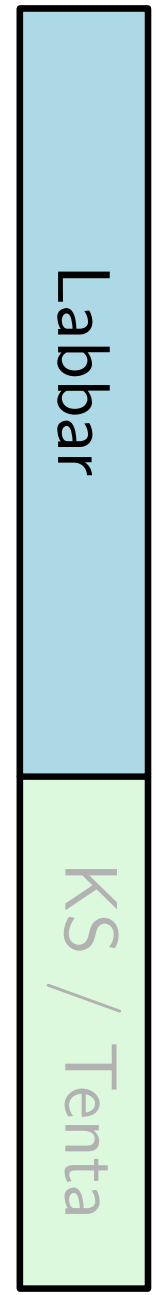
Arbeta helst två och två
(arbeta själv också tillåtet)

Skriv ut labbkvitto från hemsidan



Labbarna: Git

Vi arbetar med labbarna i versionshanteringssystemet **Git**



Labbarna: Git

Vi arbetar med labbarna i versionshanteringssystemet **Git**

Om ni inte använt Git förut kommer det antagligen kännas överväldigande i början.

(xkcd #1597)

Labbarna: Git

Vi arbetar med labbarna i versionshanteringssystemet **Git**

Om ni inte använt Git förut kommer det antagligen kännas överväldigande i början. Det går över!

(xkcd #1597)

Labbarna: Git

Vi arbetar med labbarna i versionshanteringssystemet **Git**

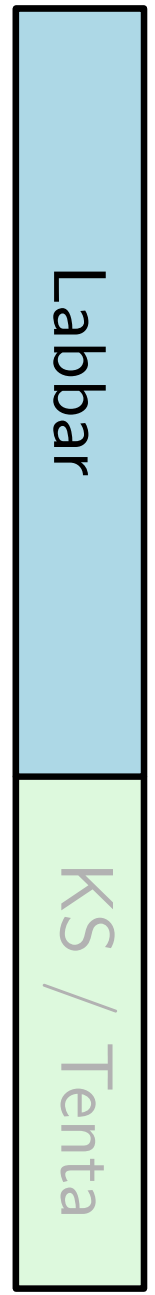
Om ni inte använt Git förut kommer det antagligen kännas överväldigande i början. Det går över!

- Skapa konto på KTH Git:
  https://gits-15.sys.kth.se
  (genom att logga in med ert KTH Id)
- Studera sidan "Inlämning av labbar via Git" på kurshemsidan.

Labbarna: Git

Vi arbetar med labbarna i versionshanteringssystemet **Git**

För er som använt Git i annan kurs (t.ex. INDA) eller i annat sammanhang.



Labbarna: Git

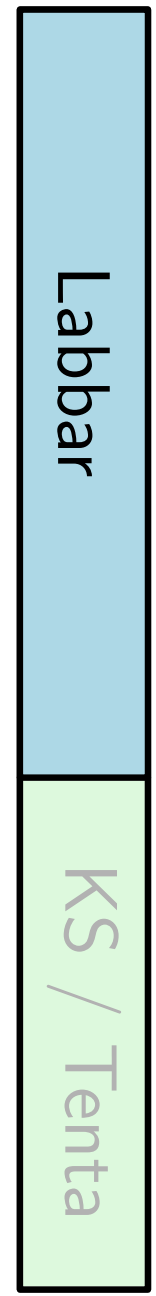
Vi arbetar med labbarna i versionshanteringssystemet **Git**

För er som använt Git i annan kurs (t.ex. INDA) eller i annat sammanhang.

Ni ska arbeta på ett specifikt sätt i den här kursen:

1. Ni får ett repo konstruerat åt er, men *ni ska inte arbeta direkt i detta*!
2. Istället ska ni skapa en egen fork av detta repo som ni arbetar i. *OBS! Ej i en branch!*
3. Labblösning lämnas sedan in som en "pull request" till det ursprungliga repot.

Studera sidan "Inlämning av labbar via Git" på kurshemsidan, även om du är van Git-användare,



Labbarna: Git

Vi arbetar med labbarna i versionshanteringssystemet **Git**

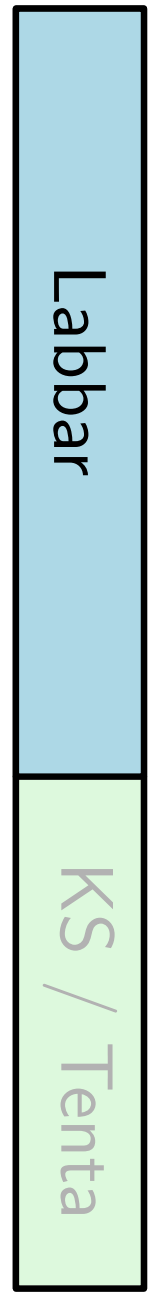
Det här är första året vi använder Git i den här kursen.

Vi har därför inte fått in rutinerna för det än, och speciellt i början av kursen kommer det säkert hända ibland att något inte riktigt funkar, eller kunde funka bättre.

Hoppas ni har överseende och tålamod med detta!

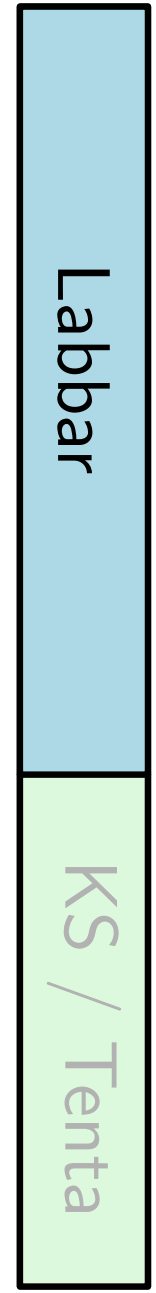
Synpunkter och förslag på förbättringar? Hör av er!

(Det gäller naturligtvis hela kursen, men extra mycket för Git...)



Labbarna: Kattis

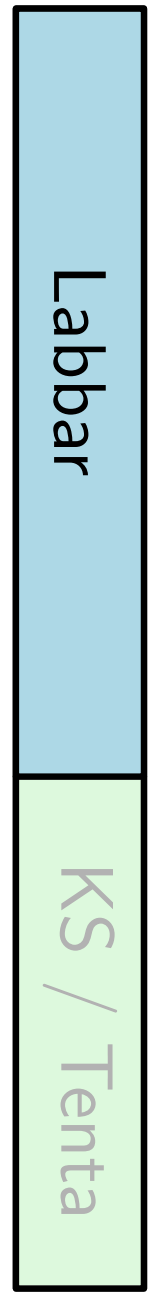
De flesta labbarna rättas i systemet Kattis:
(`https://kth.kattis.com/`)



Labbarna: Kattis

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 lämnar ni in via git

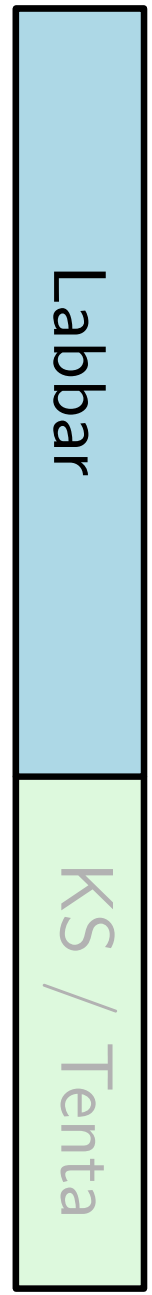


Labbarna: Kattis

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 lämnar ni in via git

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



Dokumentations-krav

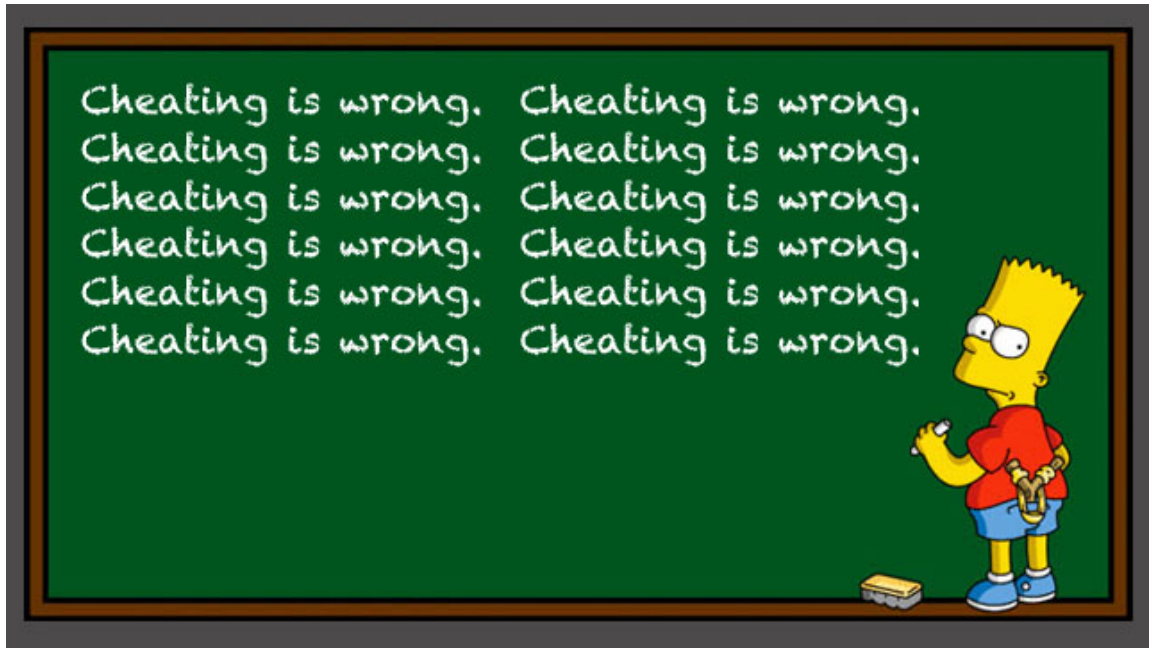
1. Det ska vara **tydligt dokumenterat** i kommentar högst upp i koden **vilka som har skrivit koden**. Detta gäller **alla inskickningar ni gör till Kattis**, och är inte något ni kan lägga till i slutet när ni väl fått er kod att bli godkänd av Kattis.

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

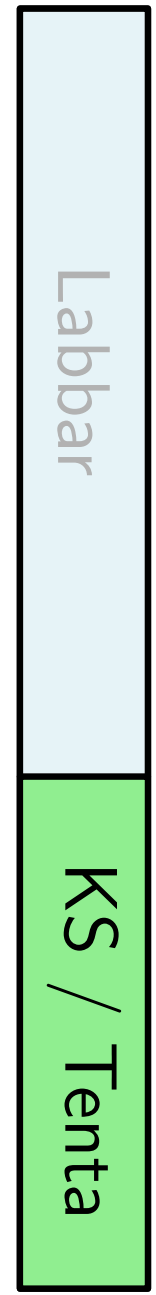
`http://www.kth.se/csc/student/hederskodex`



- Ge inte era lösningar till era medstudenter,
- Anger vilka källor ni tagit hjälp av,
- Gör inte era medstudenters jobb åt dem,
- etc...

KS / Tenta

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

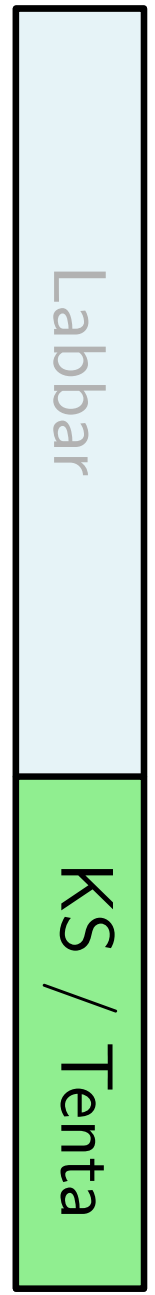


KS / Tenta

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

KS:

- 20 sep: Funktionell programmering
- 13 okt: Logikprogrammering
- 24 nov: Syntaxanalys



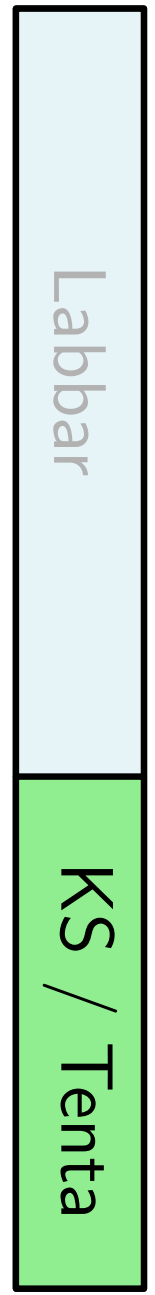
KS / Tenta

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

KS:

- 20 sep: Funktionell programmering
- 13 okt: Logikprogrammering
- 24 nov: Syntaxanalys

13 jan: Tenta!



KS / Tenta

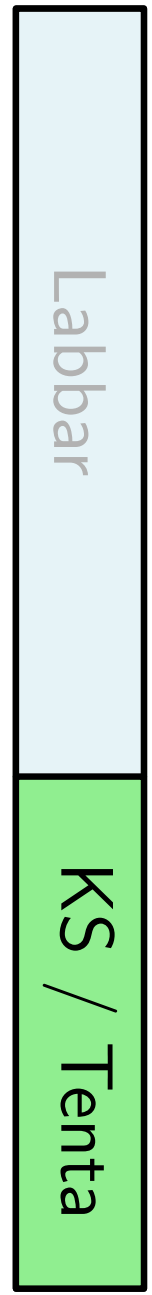
KS/Tenta på kursdelarna funktionell programmering, logikprogrammering, och syntaxanalys

KS:

- 20 sep: Funktionell programmering
- 13 okt: Logikprogrammering
- 24 nov: Syntaxanalys

13 jan: Tenta!

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



KS / Tenta

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

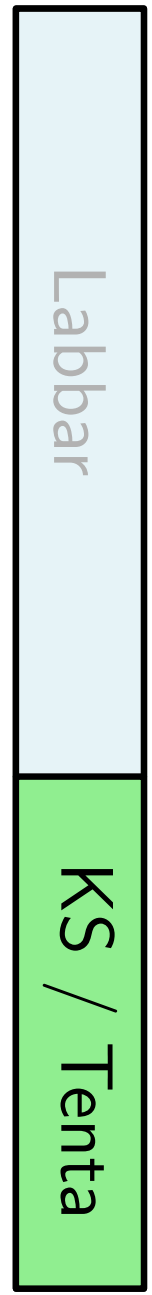
KS:

- 20 sep: Funktionell programmering
- 13 okt: Logikprogrammering
- 24 nov: Syntaxanalys

13 jan: Tenta!

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

Klarat alla tre KS $\Rightarrow$ behöver inte skriva tentan alls



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...

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

Försök komma igång med den här kursen (och era andra kurser) snarast! Ni kommer tacka er själva efteråt...

För progp:

1. Läs sidan ”Komma igång”
2. Testa inlämningssystemet via git (se sidan ”Inlämning av labbar via Git” under Laborationer på kurshemsidan)
3. Kom igång med första labben (F1): öppna labb-kompendiet, läs instruktionerna och uppgiften, testa att skicka in kod-skelettet till Kattis.

Frågor?