

Programmering, grundkurs pgk

Föreläsningsanteckningar pgk (EDAA45)

Björn Regnell

Datavetenskap, LTH

Lp1-2, HT 2016

1 Introduktion

2 Kodstruktur

1 Introduktion

- Om kursen
- Om programmering
- Meddelande från [Code@LTH](#)

2 Kodstruktur

Om kursen

Veckoöversikt

<i>W</i>	<i>Modul</i>	<i>Övn</i>	<i>Lab</i>
W01	Introduktion	expressions	kojo
W02	Kodstrukturer	programs	–
W03	Funktioner, Objekt	functions	bugs
W04	Datastrukturer	data	pirates
W05	Sekvensalgoritmer	sequences	cards
W06	Klasser	classes	turtlegraphics
W07	Arv	traits	turtlerace-team
KS	KONTROLLSKRIVN.	–	–
W08	Mönster, Undantag	matching	chords-team
W09	Matriser, Typparametrar	matrices	maze
W10	Sökning, Sortering	sorting	surveydata-team
W11	Scala och Java	scalajava	lthopoly-team
W12	Trådar	threads	life
W13	Design	Uppsamling	Projekt
W14	Tentaträning	Extenta	–
T	TENTAMEN	–	–

Vad lär du dig?

- Grundläggande principer för programmering:
Sekvens, Alternativ, Repetition, Abstraktion (SARA)
⇒ Inga förkunskaper i programmering krävs!
- Konstruktion av algoritmer
- Tänka i abstraktioner
- Förståelse för flera olika angreppssätt:
 - **imperativ programmering**
 - **objektorientering**
 - **funktionsprogrammering**
- Programspråken **Scala** och **Java**
- Utvecklingsverktyg (editor, kompilator, utvecklingsmiljö)
- Implementera, testa, felsöka

Hur lär du dig?

- Genom praktiskt **eget arbete**: **Lära genom att göra!**
 - Övningar: applicera koncept på olika sätt
 - Laborationer: kombinera flera koncept till en helhet
- Genom studier av kursens teori: **Skapa förståelse!**
- Genom samarbete med dina kurskamrater: **Gå djupare!**

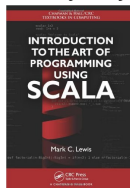
Kurslitteratur



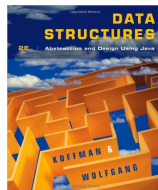
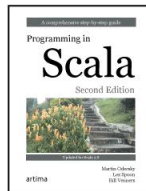
- **Kompendium** med föreläsningsanteckningar, övningar & laborationer
- Säljs på KFS <http://www.kfsab.se/>

Rekommenderade böcker

För nybörjare:



För de som redan kodat en del:



Personal

Kursansvarig:

Björn Regnell, bjorn.regnell@cs.lth.se

Kurssekreterare:

Lena Ohlsson

Exp.tid 09.30 – 11.30 samt 12.45 – 13.30

Handledare:

Maj Stenmark, Tekn. Lic., Doktorand

Gustav Cedersjö, Doktorand

Anton Klarén, D09

Maria Priisalu , D11

Anders Buhl, D13

Erik Bjäreholt, D13

Fatima Abou Alpha, D13

Cecilia Lindskog, D14

Emma Asklund, D14

Kursmoment — varför?

- **Föreläsningar**: skapa översikt, ge struktur, förklara teori, svara på frågor, motivera varför
- **Övningar**: bearbeta teorin med avgränsade problem, grundövningar för alla, extraövningar om du behöver öva mer, fördjupningsövningar om du vill gå vidare, **förberedelse** inför laborationerna
- **Laborationer**: lösa programmeringsproblem praktiskt, **obligatoriska** uppgifter; lösningar redovisas för handledare
- **Resurstider**: få hjälp med övningar och laborationsförberedelser av handledare, fråga vad du vill
- **Samarbetsgrupper**: grupplärande genom samarbete, hjälpa varandra
- **Kontrollskrivning**: **obligatorisk**, diagnostisk, kamraträttad; kan ge samarbetsbonuspoäng till tentan
- **Individuell projektuppgift**: **obligatorisk**, du visar att du kan skapa ett större program självständigt; redovisas för handledare
- **Tenta**: Skriftlig tentamen utan hjälpmedel, förutom **snabbpreferens**.

Detta är bara början...

Exempel på efterföljande kurser som bygger vidare på denna:

■ Årskurs 1

- Programmeringsteknik – fördjupningskurs
- Utvärdering av programvarusystem
- Diskreta strukturer

■ Årskurs 2

- Objektorienterad modellering och design
- Programvaruutveckling i grupp
- Algoritmer, datastrukturer och komplexitet
- Funktionsprogrammering

Registrering

- Fyll i listan som skickas runt.
- Kryssa i kolumnen **Ska gå** om du ska gå kursen¹²
- Kryssa i kolumnen **Kursombud** om du kan tänka dig att vara kursombud under kursens gång
 - Alla LTH-kurser ska utvärderas under kursens gång och efter kursens slut.
 - Till det behövs kursombud – ungefär 2 D-are och 2 W-are.
 - Ni kommer att bli kontaktade av studierådet.
SRD ordf: Amelia Andersson

¹D1:a som redan gått motsvarande högskolekurs? Uppsök studievägledningen

²D2:a eller äldre som vill bli omregistrerad? Prata med kursansvarig på rasten

Förkunskaper

- Förkunskaper \neq Förmåga
- Varken kompetens eller personliga egenskaper är statiska
- "Programmeringskompetens" är inte *en* enda enkel förmåga utan en komplex sammansättning av flera olika förmågor som utvecklas genom hela livet
- Ett innovativt utvecklarteam behöver många olika kompetenser för att vara framgångsrikt

Förkunskapsenkät

- Om du inte redan gjort det: fyll i denna enkät **snarast**:
<http://cs.lth.se/pgk/presurvey>
- Dina svar behandlas internt och all statistik anonymiseras.
- Enkäten ligger till grund för randomiserad gruppindelning i samarbetsgrupper, så att det blir en spridning av förkunskaper inom gruppen.
- Gruppindelning publiceras här:
<http://cs.lth.se/pgk/grupper/>

Samarbetgrupper

- Ni delas in i **samarbetsgrupper** om ca 5 personer baserat på förkunskapsenkäten, så att olika förkunskapsnivåer sammanförs
- Några av laborationerna är mer omfattande **grupplabbar** och kommer att göras i samarbetsgrupperna
- Kontrollskrivningen i halvtid kan ge **samarbetsbonus** (max 5p) som adderas till ordinarie tentans poäng (max 100p) med medelvärdet av gruppmedlemmarnas individuella kontrollskrivningspoäng

Bonus b för varje person i en grupp med n medlemmar med p_i poäng vardera på kontrollskrivningen:

$$b = \sum_{i=1}^n \frac{p_i}{n}$$

Varför studera i samarbetsgrupper?

Huvudsyfte: **Bra lärande!**

- Pedagogisk forskning stödjer tesen att lärandet blir mer djupinriktat om det sker i utbyte med andra
- Ett studiesammanhang med höga ambitioner och respektfull gemenskap gör att vi **når mycket längre**
- Varför ska du som redan kan mycket aktivt dela med dig av dina kunskaper?
 - Förstå bättre själv genom att förklara för andra
 - Träna din pedagogiska förmåga
 - Förbered dig för ditt kommande yrkesliv som mjukvaruutvecklare

Samarbetskontrakt

Gör ett skriftligt **samarbetskontrakt** med dessa och ev. andra punkter som ni också tycker bör ingå:

- 1 Återkommande mötestider per vecka
- 2 Kom i tid till gruppmöten
- 3 Var väl förberedd genom självstudier inför gruppmöten
- 4 Hjälp varandra att förstå, men ta inte över och lös allt
- 5 Ha ett respektfullt bemötande även om ni har olika åsikter
- 6 Inkludera alla i gemenskapen

Diskutera hur ni ska uppfylla dessa innan alla skriver på.
Ta med samarbetskontraktet och visa för handledare på labb 1.

Om arbetet i samarbetsgruppen inte fungerar ska ni mejla kursansvarig och boka mötestid!

Bestraffa inte frågor!

- Det finns bättre och sämre frågor vad gäller hur mycket man kan lära sig av svaret, men **all undran är en chans** att i dialog utbyta erfarenheter och lärande
- Den som frågar **vill veta** och berättar genom frågan något om nuvarande kunskapsläge
- Den som svarar får chansen att **reflektera** över vad som kan vara svårt och olika vägar till djupare förståelse
- I en hälsosam lärandemiljö är det **helt tryggt** att visa att man ännu inte förstår, att man gjort "fel", att man har mer att lära, etc.
- Det är viktigt att våga försöka även om det blir "fel":
det är ju då man lär sig!

Plagiatregler

Läs dessa regler noga och diskutera i samarbetsgrupperna:

- <http://cs.lth.se/utbildning/samarbete-eller-fusk/>
- <http://cs.lth.se/utbildning/foereskrifter-angaaende-obligatoriska-moment/>

Ni ska lära er genom **eget arbete** och genom **bra samarbete**. Samarbete gör att man lär sig bättre, men man lär sig inte av att bara kopiera andras lösningar. Plagiering är förbjuden och kan medföra disciplinärende och avstängning.

En typisk kursvecka

- 1 Gå på **föreläsningar** på **måndag–tisdag**
- 2 Jobba med **individuellt** med teori, övningar, labbförberedelser på **måndag–torsdag**
- 3 Kom till **resurstiderna** och få hjälp och tips av handledare och kurskamrater på **onsdag–torsdag**
- 4 Genomför den obligatoriska **laborationen** på **fredag**
- 5 Träffas i **samarbetsgruppen** och hjälp varandra att förstå mer och fördjupa lärandet, förslagsvis på återkommande tider varje vecka då alla i gruppen kan

Se detaljerna och undantagen i schemat: cs.lth.se/pgk/schema

Laborationer

- **Programmering lär man sig bäst genom att programmera...**
- Labbarna är **individuella** (utom 2) och **obligatoriska**
- Gör övningarna och labbförberedelserna noga *innan* själva labben – detta är ofta helt nödvändigt för att du ska hinna klart. Dina labbförberedelserna kontrolleras av handledare under labben.
- Är du sjuk? Anmäl det *före* labben till bjorn.regnell@cs.lth.se, få hjälp på resurstid och redovisa på resurstid (eller labbtid, när handledaren har tid över)
- Hinner du inte med hela? Se till att handledaren noterar din närvaro, och fortsatt på resurstid och ev. uppsamlingstider.
- Läs noga anvisningarna i kompendiet
- Laborationstiderna är gruppindelade enligt [schemat](#). Du ska gå till den tid och den sal som motsvarar din [grupp](#).

Resurstider

- På resurstiderna får du hjälp med övningar och laborationsförberedelser
- Kom till minst en resurstid per vecka (se [schema](#))
- Handledare gör ibland **genomgångar** för alla under resurstiderna. Tipsa om handledare om vad du finner svårt.
- Resurstiderna är inte gruppindelade i schemat. Du får i mån av plats gå på flera resurstider per vecka. Om det blir fullt i ett rum prioriteras dessa grupper för att minimera schemakrockar:

Tid Lp1	Sal	Grupper med prio
Ons 10-12 v1-7	Hacke	09 & 10
Ons 13-15 v1-7	Hacke	07 & 08
Ons 15-17 v1-7	Panter	05 & 06
Ons 15-17 v1-7	Val	03 & 04
Tor 13-15 v1-7	Mars	01 & 02
Tor 15-17 v1-7	Mars	11 & 12

Om programmering

Vad är programmering?

- Programmering innebär att ge instruktioner till en maskin.
- Ett **programmeringsspråk** används av människor för att skriva **källkod** som kan översättas av en **kompilator** till **maskinspråk** som i sin tur **exekveras** av en dator.
- Ada Lovelace skrev det första programmet redan på 1800-talet ämnat för en kugghjulsdator.
- Ha picknick i Ada Lovelace-parken på Brunshög!
- sv.wikipedia.org/wiki/Programmering
- en.wikipedia.org/wiki/Computer_programming
- kartor.lund.se/wiki/lundanamn/index.php/Ada_Lovelace-parken

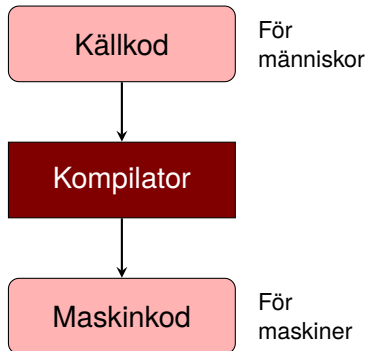


Vad är en kompilator?



Grace Hopper uppfann första kompilatorn 1952.

en.wikipedia.org/wiki/Grace_Hopper



Vad består ett program av?

- Text som följer entydiga språkregler (gramatik):
 - **Syntax**: textens konkreta utseende
 - **Semantik**: textens betydelse (vad maskinen gör/beräknar)
- **Nyckelord**: ord med speciell betydelse, t.ex. **if**, **else**
- **Deklaration**: definitioner av nya ord: **def** gurka = 42
- **Satser** är instruktioner som *gör* något: `print("hej")`
- **Uttryck** är instruktioner som beräknar ett *resultat*: `1 + 1`
- **Data** är information som behandlas: t.ex. heltalet 42
- Instruktioner ordnas i kodstrukturer: (SARA)
 - **Sekvens**: ordningen spelar roll för vad som händer
 - **Alternativ**: olika saker händer beroende på uttrycks värde
 - **Repetition**: satser upprepas många gånger
 - **Abstraktion**: nya byggblock skapas för att återanvändas

Exempel på programmeringsspråk

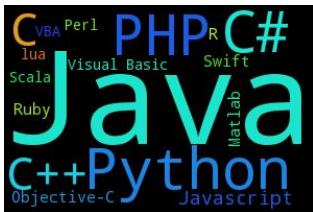
Det finns massor med olika språk och det kommer ständigt nya.

Exempel:

- Java
- C
- C++
- C#
- Python
- JavaScript
- Scala

Topplistor:

- TIOBE Index
- PYPL Index



Varför Scala + Java som förstaspråk?

■ Varför Scala?

- Enkel och enhetlig syntax => lätt att skriva
- Enkel och enhetlig semantik => lätt att fatta
- Kombinerar flera angreppssätt => lätt att visa olika lösningar
- Statisk typning + typhärledning => färre buggar + koncis kod
- Scala Read-Evaluate-Print-Loop => lätt att experimentera

■ Varför Java?

- Det mest spridda språket
- Massor av fritt tillgängliga kodbibliotek
- Kompatibilitet: fungerar på många plattformar
- Effektivitet: avancerad & mogen teknik ger snabba program

■ Java och Scala fungerar utmärkt tillsammans

- Illustrera likheter och skillnader mellan olika språk
=> Djupare lärande

Hello world

```
scala> println("Hello World!")  
Hello World!
```

```
// this is Scala
```

```
object Hello {  
  def main(args: Array[String]): Unit = {  
    println("Hejsan scala-appen!")  
  }  
}
```

```
// this is Java
```

```
public class Hi {  
  public static void main(String[] args) {  
    System.out.println("Hejsan Java-appen!");  
  }  
}
```

Utvecklingscykeln

editera; kompilera; hitta fel och förbättringar; editera; kompilera;
hitta fel och förbättringar; editera; kompilera; hitta fel och
förbättringar; editera; kompilera; hitta fel och förbättringar;
editera; kompilera; hitta fel och förbättringar; editera; kompilera;
hitta fel och förbättringar; ...

```
upprepa(1000){  
  editera  
  kompilera  
  testa  
}
```

Utvecklingsverktyg

- Din verktygskunskap är mycket viktig för din produktivitet.
- Lär dig kortkommandon för vanliga handgrep.
- Verktyg vi använder i kursen:
 - Scala **REPL**: från övn 1
 - **Texteditor** för kod, t.ex `gedit`: från övn 2
 - Kompilera med **scalac** och **javac**: från övn 2
 - Integrerad utvecklingsmiljö (IDE)
 - **Kojo**: från lab 1
 - **Eclipse** med plugin **ScalaIDE**: från lab 3
 - **jar** för att packa ihop och distribuera klassfiler
 - **javadoc** och **scaladoc** för dokumentation av kodbibliotek
- Andra verktyg som är bra att lära sig:
 - git för versionshantering
 - GitHub för kodlagring – men **inte** av lösningar till labbar!

Att skapa koden som styr världen

I stort sett alla delar av samhället är beroende av programkod:

- kommunikation
- transport
- byggsektorn
- statsförvaltning
- finanssektorn
- media & underhållning
- sjukvård
- övervakning
- integritet
- upphovsrätt
- miljö & energi
- sociala relationer
- utbildning
- ...

Hur blir ditt framtida yrkesliv som systemutvecklare?

- Redan nu är det en skriande brist på utvecklare och bristen blir bara värre och värre...
[CS 2015-08-17](#)
- Störst brist är det på kvinnliga utvecklare:
[DN 2015-04-02](#)
- Global kompetensmarknad
[CS 2015-06-14](#)
[CS 2015-08-15](#)

Utveckling av mjukvara i praktiken

- **Inte bara kodning:** kravbeslut, releaseplanering, design, test, versionshantering, kontinuerlig integration, driftsättning, återkoppling från dagens användare, ekonomi & investering, gissa om morgondagens användare, ...
- **Teamwork:** Inte ensamma hjältar utan autonoma team i decentraliserade organisationer med innovationsuppdrag
- **Snabbhet:** Att koda innebär att hela tiden uppfinna nya "byggstenar" som ökar organisationens förmåga att snabbt skapa värde med hjälp av mjukvara. Öppen källkod. Skapa kraftfulla API:er.
- **Livslångt lärande:** Lär nytt och dela med dig hela tiden. Exempel på pedagogisk utmaning: hjälp andra förstå och använda ditt API \Rightarrow *Samarbetskultur*

Programming unplugged: Två frivilliga?



Editara och exekvera ett program



Meddelande från Code@LTH

1 Introduktion

2 Kodstruktur

- Att göra denna vecka
- Algoritmer

Resurstider och Labbar

- Laborationer är **obligatoriska**.
Ev. sjukdom måste anmälas **före** till kursansvarig!
- Resurstiderna hade närvaro på endast ca. 50%.
Varför?

Att göra i Vecka 1: Fatta kodstruktur

- 1 Läs följande kapitel i kursboken:
2.1-2.6, 4, 5.4, 7.2, 7.5-7.6, 7.8-7.9
Begrepp: algoritm, pseudokod, abstraktion, oändlig loop, while-sats, for-sats, paket, import, referensvariabel, objekt, referenstilldelning, referenslikhet
- 2 Gör övning 2: Paket, kodfiler, och dokumentation
- 3 OBS! Ingen lab denna vecka
- 4 Träffas i samlingsgrupper och hjälp varandra att förstå
- 5 Gör klart [samarbetskontrakt](#) och visa för handledare på resurstid
- 6 **Koda på resurstiderna** och få hjälp och tips!
Varför var de så få kom till resurstiderna vecka 1?

Vad är en algoritm?

En **algoritm** är en sekvens av instruktioner som beskriver hur man löser ett problem

Exempel:
matrecept

Vad är en algoritm?

En **algoritm** är en sekvens av instruktioner som beskriver hur man löser ett problem

Exempel:

matrecept

uppdatera highscore i ett spel

...

XX