

Programmering, grundkurs

Kompendium

EDAA45, Lp1-2, HT 2016

Datavetenskap, LTH

Lunds Universitet

<http://cs.lth.se/pgk>

Editor: Björn Regnell

Contributors: ...

Home: <https://cs.lth.se/pgk>

Repo: <https://github.com/lunduniversity/introprog>

This manuscript is on-going work. Contributions are welcome!

Contact: bjorn.regnell@cs.lth.se

LICENCE: CC BY-SA 4.0

<http://creativecommons.org/licenses/by-sa/4.0/>

Copyright © Computer Science, LTH, Lund University. 2016. Lund. Sweden.

Framstegsprotokoll

Genomförda övningar

Till varje laboration hör en övning med uppgifter som utgör förberedelse inför labben. Du behöver minst behärska de grundövningarna för att klara labben inom rimlig tid. Om du känner att du behöver öva mer på grunderna, gör då även extrauppgifterna. Om du vill fördjupa dig, gör fördjupningsuppgifterna som är på mer avancerad nivå. Genom att du kryssar för nedan vilka övningar du har gjort, blir det lättare för handledaren att förstå vilka förkunskaper du har inför labben.

Övning	Grund	Extra	Fördjupning
expressions	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
programs	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
functions	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
data	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
vectors	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
classes	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
traits	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
matching	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
matrices	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
sorting	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
scalajava	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
threads	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Godkända obligatoriska moment

För att bli godkänd på laborationsuppgifterna och inlämningsuppgiften måste du lösa deluppgifterna och diskutera dina lösningar med en handledare. Denna diskussion är din möjlighet att få feedback på dina lösningar. Ta vara på den! Se till att handledaren noterar när du blivit godkänd på detta blad, som är ditt kvitto. Spara detta blad tills du fått slutbetyg i kursen.

Namn:

Namnteckning:

Lab	Datum gk	Handledares namnteckning
kojoturtle
simplewindow
textfiles
cardgame
shapes
turtlerace-team
newlab-team
maze
surveydata-team
scalajava-team
life
Inl.Uppg.

Inlämningsuppgift (välj en)

- () bank
- () mandelbrot
- () draw
- () egendefinerad

Om egen, ge kort beskrivning:

Förord

Programmering är inte bara ett sätt att ta makten över de människoskapade system som är förutsättningen för vårt moderna samhälle. Programmering är också ett kraftfullt verktyg för tanken. Med kunskap i programmeringens grunder kan de som vill påbörja den livslånga läranderesan som det innebär att vara systemutvecklare och abstraktionskonstnär. Programmeringsspråk och utvecklingsverktyg kommer och går, men de grundläggande koncepten bakom *all* mjukvara består: sekvens, alternativ, repetition och abstraktion.

Detta kompendium utgör kursmaterial för en grundkurs i programmering, som syftar till att ge en solid bas för ingenjörstudenter och andra som vill utveckla system med mjukvara. Materialet omfattar en termins studier på kvartsfart och förutsätter kunskaper motsvarande gymnasienivå i svenska, matematik och engelska.

Kompendiet är framtaget för och av studenter och lärare, och distribueras som öppen källkod. Det får användas fritt så länge erkännande ges och eventuella ändringar publiceras under samma licens som ursprungsmaterialet. På kursens hemsida cs.lth.se/pgk och repo github.com/lunduniversity/introprog finns instruktioner om hur du kan bidra till kursmaterialet.

Läromaterialet fokuserar på lärande genom praktiskt programmeringsarbete och innehåller övningar och laborationer som är organiserade i moduler. Varje modul har ett tema och en teoridel i form av föreläsningsbilder med tillhörande anteckningar.

I kursen använder vi språken Scala och Java för att illustrera grunderna i imperativ och objektorienterad programmering, tillsammans med elementär funktionsprogrammering. Mer avancerad objektorientering och funktionsprogrammering lämnas till efterföljande fördjupningskurser.

Den kanske viktigaste framgångsfaktorn vid studier i programmering är att bejaka din egen upptäckarglädje och experimentlusta. Det fantastiska med programmering är att dina egna intellektuella konstruktioner faktiskt *gör* något som just *du* har bestämt! Ta vara på det och prova dig fram genom att koda egna idéer – det är kul när det funkar men minst lika lärorikt är felsökning, bugggrättande och alla misslyckade försök som efter hårt arbete vänds till lyckade lösningar och/eller bestående lärdomar.

Välkommen i programmeringens fascinerande värld och hjärtligt lycka till med dina studier!

LTH, Lund 2016

Innehåll

Framstegsprotokoll	3
Förord	5
I Om kursen	5
Kursens arkitektur	7
Anvisningar	11
Samarbetsgrupper	11
Föreläsningar	11
Övningar	11
Laborationer	11
Resurstider	11
Kontrollskrivning	11
Tentamen	11
Hur bidra till kursmaterialet?	13
II Moduler	15
1 Introduktion	17
1.1 Vad är programmering?	18
1.2 Vad är en kompilator?	18
1.3 Vad består ett program av?	19
1.4 Exempel på programspråk	19
1.5 Övning: expressions	20
1.5.1 Grunduppgifter	20
1.5.2 Extrauppgifter: öva mer på grunderna	27
1.5.3 Fördjupningsuppgifter: avancerad nivå	28
1.6 Laboration: kojoturtle	29
1.6.1 Obligatoriska uppgifter	29
1.6.2 Frivilliga extrauppgifter	29

2	Kodstrukturer	31
2.1	Övning: programs	32
2.1.1	Grunduppgifter	32
2.1.2	Extrauppgifter: öva mer på grunderna	32
2.1.3	Fördjupningsuppgifter: avancerad nivå	32
3	Funktioner, Objekt	33
3.1	Övning: functions	34
3.1.1	Grunduppgifter	34
3.1.2	Extrauppgifter: öva mer på grunderna	34
3.1.3	Fördjupningsuppgifter: avancerad nivå	34
3.2	Laboration: simplewindow	35
3.2.1	Obligatoriska uppgifter	35
3.2.2	Frivilliga extrauppgifter	35
III	Appendix	37
A	Terminalfönster och kommandoskal	39
A.1	Vad är ett terminalfönster?	39
A.2	Några viktiga terminalkommando	39
B	Editera	41
B.1	Vad är en editor?	41
B.2	Välj editor	41
C	Kompilera och exekvera	43
C.1	Vad är en kompilator?	43
C.2	Java JDK	43
C.2.1	Installera Java JDK	43
C.3	Scala	43
C.3.1	Installera Scala-kompilatorn	43
C.4	Read-Evaluate-Print-Loop (REPL)	43
C.4.1	Scala REPL	43
D	Dokumentation	45
D.1	Vad gör ett dokumentationsverktyg?	45
D.2	scaladoc	45
D.3	javadoc	45
E	Integrerad utvecklingsmiljö	47
E.1	Vad är en IDE?	47
E.2	Kojo	47
E.2.1	Installera Kojo	47
E.2.2	Använda Kojo	47
E.3	Eclipse och ScalaIDE	47
E.3.1	Installera Eclipse och ScalaIDE	47
E.3.2	Använda Eclipse och ScalaIDE	47

F Byggverktyg	49
F.1 Vad gör ett byggverktyg?	49
F.2 Byggverktyget sbt	49
F.2.1 Installera sbt	49
F.2.2 Använda sbt	49
G Versionshantering och kodlagringsplatser	51
G.1 Vad är versionshantering?	51
G.2 Versionshanteringsverktyget git	51
G.2.1 Installera git	51
G.2.2 Använda git	51
G.3 Vad är nyttan med en kodlagringsplats?	51
G.4 Kodlagringsplatsen GitHub	51
G.4.1 Installera klienten för GitHub	51
G.4.2 Använda GitHub	51
G.5 Kodlagringsplatsen Atlassian BitBucket	51
G.5.1 Installera SourceTree	51
G.5.2 Använda SourceTree	51
H Lösningsförslag till övningar	53
H.1 Lösningar till övning: expressions	54
H.1.1 Grunduppgifter	54
H.1.2 Extrauppgifter: öva mer på grunderna	54
H.1.3 Fördjupningsuppgifter: avancerad nivå	54
H.2 Lösningar till övning: programs	55
H.2.1 Grunduppgifter	55
H.2.2 Extrauppgifter: öva mer på grunderna	55
H.2.3 Fördjupningsuppgifter: avancerad nivå	55
H.3 Lösningar till övning: functions	56
H.3.1 Grunduppgifter	56
H.3.2 Extrauppgifter: öva mer på grunderna	56
H.3.3 Fördjupningsuppgifter: avancerad nivå	56
I Ordlista	57

Del I

Om kursen

Kursens arkitektur

Veckoöversikt

<i>W</i>	<i>Modul</i>	<i>Övn</i>	<i>Lab</i>
W01	Introduktion	expressions	kojoturtle
W02	Kodstrukturer	programs	–
W03	Funktioner, Objekt	functions	simplewindow
W04	Datastrukturer	data	textfiles
W05	Vektoralgoritmer	vectors	cardgame
W06	Klasser, Likhet	classes	shapes
W07	Arv, Gränssnitt	traits	turtlerace-team
KS	KONTROLLSKRIVN.	–	–
W08	Mönster, Undantag	matching	newlab-team
W09	Matriser	matrices	maze
W10	Sökning, Sortering	sorting	surveydata-team
W11	Scala och Java	scalajava	scalajava-team
W12	Trådar, Web, Android	threads	life
W13	Design	Uppsamling	Inl.Uppg.
W14	Tentaträning	Extenta	–
T	TENTAMEN	–	–

Kursen består av ett antal moduler med tillhörande teori, övningar och laborationer. Genom att göra övningarna bearbetar du teorin och föreber dig inför laborationerna. När du klarat av laborationen i varje modul är du redo att gå vidare till efterkommande modul.

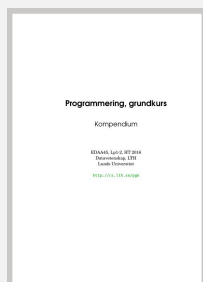
Vad lär du dig?

- 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!**

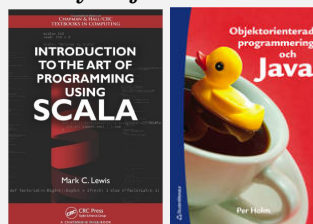
Kurslitteratur



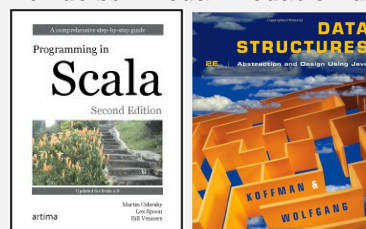
- **Kompendium** med föreläsningsanteckningar, övningar & laborationer
- Säljs på KFS
<http://www.kfsab.se/>

Rekommenderade böcker

För nybörjare:



För de som redan kodat en del:



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
- **Inlämningsuppgift**: **obligatorisk**, du visar att du kan skapa ett större program självständigt; redovisas för handledare
- **Tenta**: Skriftlig tentamen utan hjälpmedel, förutom **snabbreferens**.

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

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

Anvisningar

Samarbetsgrupper

Samarbetskontrakt

Föreläsningar

Övningar

Laborationer

Resurstider

Kontrollskrivning

Tentamen

Hur bidra till kursmaterialet?

Del II

Moduler

Kapitel 1

Introduktion

- sekvens
- alternativ
- repetition
- abstraktion
- programmeringsspråk
- programmeringsparadigmer
- editera-kompilera-exekvera
- datorns delar
- virtuell maskin
- värde
- uttryck
- variabel
- typ
- tilldelning
- namn
- val
- var
- def
- alternativ
- if
- else
- true
- false
- MinValue
- MaxValue
- aritmetik logiska uttryck
- de Morgans lagar
- while-sats
- for-sats

1.1 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!



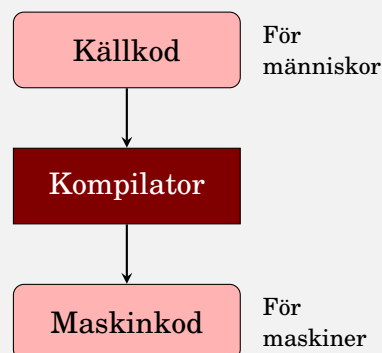
- sv.wikipedia.org/wiki/Programmering
- en.wikipedia.org/wiki/Computer_programming
- kartor.lund.se/wiki/lundanamn/index.php/Ada_Lovelace-parken

1.2 Vad är en kompilator?



Grace Hopper uppfann första kompilatorn 1952.

en.wikipedia.org/wiki/Grace_Hopper



1.3 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

1.4 Exempel på programspråk

Några programspråk:

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

Topplistor med de "vanligaste":

- [TIOBE Programming Community Index Augusti 2015](#)
- [Språktrender på GitHub 2008-2015](#)

1.5 Övning: expressions

Mål

- Lär dig detta
- Lär dig och detta

Förberedelser

- Läs kap. ??
- Säkerställ att du kan avända de grundläggande terminalkommandona `ls`, `cd`, `rm` och `mkdir` för att inspektera, navigera i, och manipulera filträdet, se kap. ??.
- Du behöver en dator med scala installerad. Om du inte har Scala installerad på din maskin, se installationsanvisningar i kap. ??
- Starta den editor du vill använda under övningarna, se kap. ??.

1.5.1 Grunduppgifter

Uppgift 1. Starta Scala REPL (eng. *Read-Evaluate-Print-Loop*) och skriv satsen `println("hejsan REPL")` och tryck på *Enter*.

```
> scala
Welcome to Scala version 2.11.7 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8).
Type in expressions to have them evaluated.
Type :help for more information.

scala> println("hejsan REPL")
```

- Vad händer?
- Skriv samma sats igen men "glöm bort" att skriva högerparentesen innan du trycker på *Enter*. Vad händer?
- Evaluera uttrycket `"gurka" + "tomat"` i REPL. Vad har uttrycket för värde och typ? Vilken siffra står efter ordet `res` i variabeln som lagrar resultatet?

```
scala> "gurka" + "tomat"
```

- Evaluera uttrycket `res0 * 42` men byt ut `0`:an mot siffran efter `res` i utskriften från förra evalueringen. Vad har uttrycket för värde och typ?

```
scala> res2 * 42
```

Uppgift 2. Skapa med hjälp av en editor en fil med namn `hello-script.scala` som innehåller denna enda rad:

```
println("hej skript")
```

Spara filen och kör kommandot `scala hello-script.scala` i terminalen:

```
> scala hello-script.scala
```

- a) Vad händer?
- b) Ändra i filen så att högerparentesen saknas. Spara och kör skriptfilen igen. Vad händer?

Uppgift 3. Skapa med hjälp av en editor en fil med namn `hello-app.scala`.

```
> gedit hello-app.scala &
```

Skriv dessa rader i filen:

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

- a) Kompilera med `scalac hello-app.scala` och kör koden med `scala Hello`.

```
> scalac hello-app.scala  
> ls  
> scala Hello
```

Vad heter filerna som kompilatorn skapar?



- b) Vilket alternativ går snabbast att köra igång, ett skript eller en kompilerad applikation? Varför? Vilket alternativ kör snabbast när väl exekveringen är igång?
- c) Ändra i din kod så att kompilatorn ger följande felmeddelande:
Missing closing brace

Uppgift 4. Skapa med hjälp av en editor en fil med namn `Hi.java`.

```
> gedit Hi.java &
```

Skriv dessa rader i filen:

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

- a) Kompilera med `javac Hi.java` och kör koden med `java Hi`.

```
> javac Hi.java  
> ls  
> java Hi
```

Vad heter filen som kompilatorn skapat?

Uppgift 5. Vad är en *literal*?



Uppgift 6. Vilken typ har följande literaler?

- a) 42
- b) 42L
- c) '*'
- d) "*"
- e) 42.0
- f) 42D
- g) 42d
- h) 42F
- i) 42f
- j) true
- k) false

Uppgift 7. Vad gör dessa satser? Till vad används klammer och semikolon?



```
scala> def p = { print("hej"); print("san"); println(42); println("gurka") }  
scala> p;p;p;p
```

Uppgift 8. Satser versus uttryck.



- a) Vad är det för skillnad på en sats och ett uttryck?
- b) Ge exempel på satser som inte är uttryck?
- c) Förklara vad som händer för varje evaluerad rad:

```
1 scala> def värdeSaknas = ()  
2 scala> värdeSaknas  
3 scala> värdeSaknas.toString  
4 scala> println(värdeSaknas)  
5 scala> println(println("hej"))
```

- d) Vilken typ har literalen ()?
- e) Vilken returtyp har println?

Uppgift 9. Vilken typ och vilket värde har följande uttryck?

- a) 1 + 41
- b) 1.0 + 41
- c) 42.toDouble
- d) (41 + 1).toDouble
- e) "gurk" + 'a'
- f) 'A'
- g) 'A'.toInt

- h) `'0'.toInt`
- i) `'1'.toInt`
- j) `'9'.toInt`
- k) `('A' + '0').toChar`
- l) `"*!%#".charAt(0)`

Uppgift 10. *De fyra räknesätten.* Vilket värde och vilken typ har följande uttryck?

- a) `42 * 2`
- b) `42.0 / 2`
- c) `42 - 0.2`
- d) `42L + 2d`

Uppgift 11. *Precedensregler.* Evalueringsordningen kan styras med parenteser. Vilket värde och vilken typ har följande uttryck?

- a) `42 + 2 * 2`
- b) `(42 + 2) * 2`
- c) `((-(2 - 42)) / (1 + 1 + 1)).toDouble`
- d) `((-(2 - 42)) / (1 + 1 + 1)).toDouble).toInt`

Uppgift 12. *Heltalsdivision.* Vilket värde och vilken typ har följande uttryck?

- a) `42 / 2`
- b) `42 / 4`
- c) `42.0 / 4`
- d) `1 / 4`
- e) `1 % 4`
- f) `2 % 42`
- g) `42 % 2`

Uppgift 13. *Heltalsomfång.* För var och en av heltalstyperna i deluppgifterna nedan: undersök i REPL med operationen `MaxValue` resp. `MinValue`, till exempel `Int.MaxValue` vad som är största och minsta värde.

- a) `Byte`
- b) `Short`
- c) `Int`
- d) `Long`

Uppgift 14. Klassen `java.lang.Math` och paketobjektet `scala.math`.

- a) Undersök genom att trycka på Tab-tangenten efter att du skriver nedan, vilka funktioner som finns i `Math` och `math`. Vad heter konstanten π i `java.lang.Math` respektive `scala.math`?

```
scala> java.lang.Math.    //tryck TAB
scala> scala.math.        //tryck TAB
```

- b) Undersök dokumentationen för klassen `java.lang.Math` här:
<https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html>
Vad gör `java.lang.Math.hypot`?
- c) Undersök dokumentationen för pakobjektet `scala.math` här:
<http://www.scala-lang.org/api/current/#scala.math.package>
Ge exempel på någon funktion i `java.lang.Math` som inte finns i `scala.math`.

Uppgift 15. Vad händer här? Notera undantag (eng. *exceptions*) och nogrannhetsproblem.

- a) `Int.MaxValue + 1`
- b) `1 / 0`
- c) `1E8 + 1E-8`
- d) `1E9 + 1E-9`
- e) `math.pow(math.hypot(3,6), 2)`
- f) `1.0 / 0`
- g) `(1.0 / 0).toInt`
- h) `math.sqrt(-1)`
- i) `math.sqrt(Double.NaN)`
- j) `throw new Exception("PANG!!!")`

Uppgift 16. *Booelska uttryck.* Vilket värde och vilken typ har följande uttryck?

- a) `true && true`
- b) `false && true`
- c) `true && false`
- d) `false && false`
- e) `true || true`
- f) `false || true`
- g) `true || false`
- h) `false || false`
- i) `42 == 42`
- j) `42 != 42`
- k) `42.0001 == 42`
- l) `42.000000000000000001 == 42`
- m) `42.0001 > 42`
- n) `42.000000000000000001 > 42`
- o) `42.0001 >= 42`
- p) `42.000000000000000001 <= 42`

- q) `true == true`
- r) `true != true`
- s) `true > false`
- t) `true < false`



Uppgift 17. *Variabler och tilldelning.* Rita en ny bild av datorns minne efter varje evaluerad rad nedan. Bilderna ska visa variablers namn, typ och värde.

```
1 scala> var a = 42
2 scala> var b = a + 1
3 scala> var c = (a + b) + 1.0
4 scala> b = 0
5 scala> a = 0
6 scala> c = c + 1
```

Efter första raden ser minnessituationen ut så här:

a: Int



Uppgift 18. *Deklarationer: var, val, def.* Evaluera varje rad nedan i tur och ordning i Scala REPL. Förklarar för varje rad vad som händer. Vilka rader ger kompileringsfel och i så fall vilket och varför?

```
1 scala> var x = 42
2 scala> x + 1
3 scala> x
4 scala> x = x + 1
5 scala> x
6 scala> x == x + 1
7 scala> val y = 42
8 scala> y = y + 1
9 scala> var z = {println("gurka"); 42}
10 scala> def w = {println("gurka"); 42}
11 scala> z
12 scala> z
13 scala> z = z + 1
14 scala> w
15 scala> w
16 scala> w = w + 1
```



Uppgift 19. *if-sats.* Vad händer nedan?

```
scala> if (true) println("sant") else println("falskt")
scala> if (false) println("sant") else println("falskt")
scala> if (!true) println("sant") else println("falskt")
scala> if (!false) println("sant") else println("falskt")
scala> def kasta = if (math.random > 0.5) println("krona") else println("klave")
scala> kasta; kasta; kasta
```

Uppgift 20. *if-uttryck.* Följande variabler är deklarerade med nedan initialvärden:

```
scala> var grönsak = "gurka"
scala> var frukt = "banan"
```

Vad har följande uttryck för värden och typ?

- `if (grönsak == "tomat") "gott" else "inte gott"`
- `if (frukt == "banan") "gott" else "inte gott"`
- `if (frukt.size == grönsak.size) "lika stora" else "olika stora"`

Uppgift 21. *for-sats*.

- Vad ger nedan *for*-satser för utskrift?

```
scala> for (i <- 1 to 10) print(i + ", ")
scala> for (i <- 1 until 10) print(i + ", ")
scala> for (i <- 1 to 5) print((i * 2) + ", ")
scala> for (i <- 1 to 92 by 10) print(i + ", ")
scala> for (i <- 10 to 1 by -1) print(i + ", ")
```

- Skriv en *for*-sats som ger följande utskrift:

```
A1, A4, A7, A10, A13, A16, A19, A22, A25, A28, A31, A34, A37, A40, A43,
```

Uppgift 22. *while-sats*.

- Vad ger nedan satser för utskrift?




```
scala> var i = 0
scala> while (i < 10) { println(i); i = i + 1 }
scala> var j = 0; while (j <= 10) { println(j); j = j + 2 }; println(j)
```

- Skriv en *while*-sats som ger följande utskrift med hjälp av en variabel *k* som initialiseras till 1:

```
A1, A4, A7, A10, A13, A16, A19, A22, A25, A28, A31, A34, A37, A40, A43,
```

Uppgift 23. *Slumptal*. Undersök vad dokumentationen säger om funktionen `scala.math.random`:

<http://www.scala-lang.org/api/current/#scala.math.package>

- Vilken typ har värdet som returneras av funktionen `random`? 
- Vilket är det minsta respektive största värde som kan returneras? 
- Är `random` en *äkta* funktion (eng. *pure function*) i matematisk mening? 
- Anropa funktionen `math.random` upprepade gånger och notera vad som händer. Använd *pil-upp*-tangenten.

```
scala> math.random
```

- Vad händer? Använd *pil-upp* och kör nedan *for*-sats flera gånger. Förklara vad som sker.


```
scala> for (i <- 1 to 10) println(math.random)
```

f) Skriv en for-sats som skriver ut 100 slumpmässiga heltal från 0 till och med 9 på var sin rad.

```
scala> for (i <- 1 to 100) println(???)
```

g) Skriv en for-sats som skriver ut 100 slumpmässiga heltal från 1 till och med 6 på samma rad.

```
scala> for (i <- 1 to 100) print(???)
```

h) Använd *pil-upp* och kör nedan while-sats flera gånger. Förklara vad som sker.

```
scala> while (math.random > 0.2) { println("gurka") }
```

i) Ändra i while-satsen ovan så att sannolikheten ökar att riktigt många strängar ska skrivas ut.

j) Förklara vad som händer nedan.

```
scala> var slumptal = math.random
scala> while (slumpthal > 0.2) { println(slumpthal); slumptal = math.random }
```



Uppgift 24. *Logik och De Morgans Lagar.* Förenkla följande uttryck. Antag att poäng och highscore är heltalsvariabler medan klar är av typen Boolean.

- a) poäng > 100 && poäng > 1000
- b) poäng > 100 || poäng > 1000
- c) !(poäng > highscore)
- d) !(poäng > 0 && poäng < highscore)
- e) !(poäng < 0 || poäng > highscore)
- f) klar == true
- g) klar == false

1.5.2 Extrauppgifter: öva mer på grunderna

Uppgift 25. *Slumptal.*

a) Ersätt ??? nedan med literaler så att tärning returnerar ett slumpmässigt heltal mellan 1 och 6.

```
scala> def tärning = (math.random * ??? + ???).toInt
```

b) Ersätt ??? med literaler så att rnd blir ett decimaltal med max en decimal mellan 0.0 och 1.0.

```
scala> def rnd = math.round(math.random * ???) / ???
```

c) Vad blir det för skillnad om `math.round` ersätts med `math.floor` ovan? (Se dokumentationen av `java.lang.Math.round` och `java.lang.Math.floor`.)

1.5.3 Fördjupningsuppgifter: avancerad nivå

Uppgift 26. `Integer.toString`, `Integer.toHexString`

Uppgift 27. `0x2a`

Uppgift 28. `i += 1`; `i *= 1`; `i /= 2`

Uppgift 29. `BigInt`, `BigDecimal`

Uppgift 30. Vad händer här?

```
scala> Math.multiplyExact(2, 42)
scala> Math.multiplyExact(Int.MaxValue, Int.MaxValue)
```

Uppgift 31. Sök reda på dokumentationen för funktionen `multiplyExact` i javadoc för klassen `java.lang.Math` i JDK 8.

Uppgift 32. Sök i javadoc för `Math` efter förekomster av texten *"throwing an exception if the result overflows"*. Vilka fler funktioner finns i `java.lang.Math` som hjälper en att upptäcka om det blir overflow?

Uppgift 33. Använda Scala REPL för att undersöka konstanterna nedan. Vilket av dessa värden är negativt? Vad kan man ha för praktisk nytta av dessa värden i ett program som gör flyttalsberäkningar?

- a) `java.lang.Double.MIN_VALUE`
- b) `scala.Double.MinValue`
- c) `scala.Double.MinPositiveValue`

Uppgift 34. För typerna `Byte`, `Short`, `Char`, `Int`, `Long`, `Float`, `Double`: Undersök hur många bitar som behövs för att representera varje typs omfång?

Tips: Några användbara uttryck:

```
Integer.toString(Int.MaxValue + 1).size
```

```
Integer.toString((math.pow(2,16) - 1).toInt).size
```

`1 + math.log(Long.MaxValue)/math.log(2)` Se även språkspecifikationen för Scala, kapitlet om heltalsliteraler:

<http://www.scala-lang.org/files/archive/spec/2.11/01-lexical-syntax.html#integer-literals>

a) Undersök källkoden för paketobjektet `scala.math` här:

<https://github.com/scala/scala/blob/v2.11.7/src/library/scala/math/package.scala>

Hur många olika överlagrade varianter av funktionen `abs` finns det och för vilka parametertyper är den definierad?

1.6 Laboration: kojoturtle

Mål

-

Förberedelser

-

1.6.1 Obligatoriska uppgifter

Uppgift 1.

1.6.2 Frivilliga extrauppgifter

Uppgift 2.

Kapitel 2

Kodstrukturer

- samling: Range
- for-uttryck
- algoritm: swap
- algoritm: min/max
- algoritm: summering
- paket
- import
- filstruktur
- jar
- dokumentation
- programlayout
- JDK
- konstanter vs föränderlighet
- objektorientering
- klasser
- objekt
- referensvariabler
- referenstilldelning
- anropa metoder
- block
- namnsynlighet SimpleWindow

2.1 Övning: programs

Mål

-

Förberedelser

-

2.1.1 Grunduppgifter

Uppgift 1.

a)

2.1.2 Extrauppgifter: öva mer på grunderna

Uppgift 2.

2.1.3 Fördjupningsuppgifter: avancerad nivå

Uppgift 3.

Kapitel 3

Funktioner, Objekt

- parameter
- returtyp
- värdeandrop
- namnanrop
- namngivna parametrar
- aktiveringspost
- rekursion
- basfall
- anropsstacken
- objektheapen
- objekt
- modul
- lazy val
- aritmetik
- slumpstal

3.1 Övning: functions

Mål

-

Förberedelser

-

3.1.1 Grunduppgifter

Uppgift 1.

a)

3.1.2 Extrauppgifter: öva mer på grunderna

Uppgift 2.

3.1.3 Fördjupningsuppgifter: avancerad nivå

Uppgift 3.

3.2 Laboration: simplewindow

Mål

- Att lära sig.

Förberedelser

- Att göra.

3.2.1 Obligatoriska uppgifter

Uppgift 1. En labbuppgiftsbeskrivning.

- En underuppgift.
- En underuppgift.

3.2.2 Frivilliga extrauppgifter

Uppgift 2. En labbuppgiftsbeskrivning.

- En underuppgift.
- En underuppgift.

Del III

Appendix

Appendix A

Terminalfönster och kommandoskal

A.1 Vad är ett terminalfönster?

I ett terminalfönster kan man skriva kommandon som till exempel kör program och hanterar filer på din dator. När man programmerar använder man ofta terminalkommando för att kompilera och exekvera sina program.

Terminal i Linux

PowerShell i Microsoft Windows

Microsoft Windows är inte Unix-baserat, men i kommandotolken PowerShell finns alias definierat för en del vanliga unix-kommandon. Du startar Powershell t.ex. genom att trycka på Windows-knappen och skriva powershell.

Terminal i Apple OS X

Apple OS X är ett Unix-baserat operativsystem. Många kommandon som fungerar under Linux fungerar också under Apple OS X.

A.2 Några viktiga terminalkommando

Tipsa om ss64.com

Appendix B

Editera

B.1 Vad är en editor?

B.2 Välj editor

Appendix C

Kompilera och exekvera

C.1 Vad är en kompilator?

C.2 Java JDK

C.2.1 Installera Java JDK

C.3 Scala

C.3.1 Installera Scala-kompilatorn

C.4 Read-Evaluate-Print-Loop (REPL)

För många språk, t.ex. Scala och Python, finns det en interaktiv tolk som gör det möjligt att exekvera enstaka programrader och direkt se effekte. En sådan tolk kallas Read-Evaluate-Print-Loop eftersom den läser en rad i taget och översätter till maskinkod som körs direkt.

C.4.1 Scala REPL

Kommandon i REPL

`:paste`

Kortkommandon: Ctrl+K etc.

Appendix D

Dokumentation

D.1 Vad gör ett dokumentationsverktyg?

D.2 scaladoc

D.3 javadoc

Appendix E

Integrerad utvecklingsmiljö

E.1 Vad är en IDE?

E.2 Kojo

E.2.1 Installera Kojo

E.2.2 Använda Kojo

E.3 Eclipse och ScalalIDE

E.3.1 Installera Eclipse och ScalalIDE

E.3.2 Använda Eclipse och ScalalIDE

Appendix F

Byggverktyg

F.1 Vad gör ett byggverktyg?

F.2 Byggverktyget sbt

F.2.1 Installera sbt

F.2.2 Använda sbt

Appendix G

Versionshantering och kodlagringsplatser

G.1 Vad är versionshantering?

G.2 Versionshanteringsverktyget git

G.2.1 Installera git

G.2.2 Använda git

G.3 Vad är nyttan med en kodlagringsplats?

G.4 Kodlagringsplatsen GitHub

G.4.1 Installera klienten för GitHub

G.4.2 Använda GitHub

G.5 Kodlagringsplatsen Atlassian BitBucket

G.5.1 Installera SourceTree

G.5.2 Använda SourceTree

Appendix H

Lösningsförslag till övningar

H.1 Lösningar till övning: expressions

H.1.1 Grunduppgifter

Uppgift 1.

- a) 42
- b) Lösningstext.

H.1.2 Extrauppgifter: öva mer på grunderna

Uppgift 2.

- a) 42
- b) Lösningstext.

H.1.3 Fördjupningsuppgifter: avancerad nivå

Uppgift 3.

- a) 42
- b) Lösningstext.

H.2 Lösningar till övning: programs

H.2.1 Grunduppgifter

Uppgift 1.

- a) 42
- b) Lösningstext.

H.2.2 Extrauppgifter: öva mer på grunderna

Uppgift 2.

- a) 42
- b) Lösningstext.

H.2.3 Fördjupningsuppgifter: avancerad nivå

Uppgift 3.

- a) 42
- b) Lösningstext.

H.3 Lösningar till övning: functions

H.3.1 Grunduppgifter

Uppgift 1.

- a) 42
- b) Lösningstext.

H.3.2 Extrauppgifter: öva mer på grunderna

Uppgift 2.

- a) 42
- b) Lösningstext.

H.3.3 Fördjupningsuppgifter: avancerad nivå

Uppgift 3.

- a) 42
- b) Lösningstext.

Appendix I

Ordlista