Linköpings Tekniska Högskola
Instutitionen för Datavetenskap (IDA)
Torbjörn Jonsson, Erik Nilsson 2015-05-26

Lab 0: Mycket grundläggande in- och utmatning

I denna inledande laboration kommer du att öva på de mest grundläggande operationerna för in- och utmatning.

Mål

Du ska efter denna laboration ha god insikt i:

- skillnaden mellan Get och Put samt New_Line och Skip_Line.
- Skillnader och likheter mellan de olika varianterna av Get och Put.

Uppgift

Skriv ett program som hämtar data från tangentbordet enligt exemplen som följer. Det som syns på skärmen när man kör programmet skall stämma till fullo överens med det som står i exemplen. Detta gäller även antal blanktecken mellan ord och tal.

Observera att denna laboration till stor del är till för att introducera dig till hur man hanterar vår datormiljö vad det gäller skrivande, kompilerande och körande av program. Om du inte hinner med allt på laborationstiden är det ingen katastrof. Fortsätt på egen tid med resterande delar så att målen med uppgiften är uppfyllda.

Koncentrera dig på hur Get/Put/Put_Line/New_Line och Skip_Line fungerar. Ett tips kan vara att du gör en liten del i taget och ser till att den fungerar för båda de givna körexemplen (se följande sidor). På detta sätt undviker du att få alla ”fel” på en gång.

OBS! Ditt program skall klara av både körexempel 1 och 2 (samma program). Dock behöver det inte klara av alla möjliga varianter av indata vilket i princip är omöjligt i detta läge.

Linköpings Tekniska Högskola
Instutitionen för Datavetenskap (IDA)
Torbjörn Jonsson, Erik Nilsson 2015-05-26

Programkörningar

Programmet ska ge resultat enligt följande exempel vid olika körningar (användarens indata markeras med ***kursiverad*** stil). Observera att eventuella inmatningar som är extra skall plockas bort innan nästa inmatning görs. Observera att du i ditt program ***endast får ha en variabel av respektive datatyp***. Det blir alltså bara fyra variabler i programmet.

Körexempel 1:

```
Skriv in ett heltal: 10
Du skrev in talet: 10

Skriv in fem heltal: 12 30 27 13 11
Du skrev in talen: 12 30 27 13 11

Skriv in ett flyttal: 15.20
Du skrev in flyttalet: 15.200

Skriv in ett heltal och ett flyttal: 67 3.141592
Du skrev in heltalet:        67
Du skrev in flyttalet:   3.1416

Skriv in ett tecken: T
Du skrev in tecknet: T

Skriv in en sträng med 5 tecken: Kalle
Du skrev in strängen: Kalle

Skriv in en sträng med 3 tecken: Kul
Du skrev in strängen: Kul

Skriv in ett heltal och en sträng med 5 tecken: -5 Nisse
Du skrev in talet |-5| och strängen |Nisse|.

Skriv in en sträng med 3 tecken och ett flyttal: Ria 1.2
Du skrev in ” 1.200” och ”Ria”.
```

Körexempel 2:

```
Skriv in ett heltal: 14.72
Du skrev in talet: 14

Skriv in fem heltal: 10000 3860 27 -18      0   0
Du skrev in talen: 10000 3860 27 -18 0

Skriv in ett flyttal: 15 Nisse
Du skrev in flyttalet: 15.000

Skriv in ett heltal och ett flyttal: 7


    202.718      Dum text
Du skrev in heltalet:         7
Du skrev in flyttalet: 202.7180

Skriv in ett tecken: !K
Du skrev in tecknet: !

Skriv in en sträng med 5 tecken: #4?".!
Du skrev in strängen: #4?".

Skriv in en sträng med 3 tecken: "HA12
Du skrev in strängen: "HA

Skriv in ett heltal och en sträng med 5 tecken: 1 Ploja!
Du skrev in talet |1| och strängen |Ploja|.

Skriv in en sträng med 3 tecken och ett flyttal: Led
012.3456.23
Du skrev in "12.346" och "Led".
```

Linköpings Tekniska Högskola
Instutitionen för Datavetenskap (IDA)
Torbjörn Jonsson, Erik Nilsson 2015-05-26

Lab 1: Enkel in- och utmatning

I denna inledande laboration kommer du att öva på de grundläggande styrstrukturerna för repetition och val samt enkel in- och utmatning.

Mål

Du ska efter denna laboration känna till och kunna använda

- hur man skriver ett litet Ada-program
- vad tecknet ”;” betyder och hur det används
- de tre formerna av repetitionssatsen loop: loop, for och while
- valsatsen if
- läsa in tal med Get (dock utan att ta hand om de fel som kan uppstå om användaren skriver in ett heltal då programmet läser ett reellt tal eller tecken)
- Put för att skriva ut både heltal, reella tal och strängar både med och utan formatdirektiv.

Uppgift

Skriv och testa momstabellprogrammet. De övriga tre deluppgifterna är frivilliga, men ju fler av dessa du gör desto bättre grund har du att stå på inför resterande laborationskurs.

För samtliga uppgifter gäller att snygga utskrifter ska åstadkommas med hjälp av formatdirektiv till Put, samt att du ska kunna motivera valet av iterationssats.

I programmet ska *felhantering* av användarens inmatning göras. Felhanteringen skall vara av typen rimlighetskontroll vilket innebär att användaren alltid kommer att mata in data av rätt datatyp. Detta ska göras så fort som felet *kan* upptäckas. Programmet ska ej avbrytas utan ny fråga ska ställas om användaren matar in ett felaktigt data! Tänk på att användaren kan mata in felaktiga data många gånger ...

Momstabell

Konstruera ett program som skriver ut en momstabell. Programmet ska på terminalen fråga efter och ta som inmatning följande värden (där alla värden ska rimlighetskontrolleras!):

- Nedre samt övre gräns för prisintervallet
- Steglängd i tabellen
- Momsprocenten (uttryckt som decimaltal i intervallet 0 till 100 %)

För att få lite överblick kan det vara vettigt att börja med antagandet att användaren kommer att mata in rimliga värden och skriva ett program utan felkontroller. Programmet ska ge resultat enligt nedan vid körning (användarens indata ***kursiverad***):

Körexempel 1:

```
Första pris: 10.00
Sista pris: 15.00
Steg: 0.5
Momsprocent: 10.00

=== Momstabell ===
Pris utan moms Moms Pris med moms
    10.00      1.00     11.00
    10.50      1.05     11.55
    11.00      1.10     12.10
    .          .        .
    .          .        .
    15.00      1.50     16.50
```

Körexempel 2:

```
Första pris: 10.00
Sista pris: 12.00
Steg: 0.3
Momsprocent: 20.00

=== Momstabell ===
Pris utan moms Moms Pris med moms
    10.00      2.00     12.00
    10.30      2.06     12.36
    10.60      2.12     12.72
    .          .        .
    .          .        .
    11.80      2.36     14.16
```

OBS! Sista “Pris utan moms”-värdet i andra exemplet är 11.80 och inte 12.00 (inte heller 12.10)!

Lämpliga testdata kan vara (kombinera lite olika för att se om ditt program verkar fungera):

Första pris:	-1	0	10	100				
Sista pris:	-1	0	1	11	12	15	101	1000000
Steg:	-1	0	0.1	0.25	0.3	0.5	1	10
Momsprocent:	-1	0	1	20	50	100	101	

Linköpings Tekniska Högskola
Instutitionen för Datavetenskap (IDA)
Torbjörn Jonsson, Erik Nilsson 2015-05-26

Befolkningsproblemet (frivillig)

Befolkningen i två länder antas växa exponentiellt, dvs ökar varje år med givna bråkdelar av befolkningen vid årets början. Skriv ett program som först läser in ett årtal, folkmängden detta år för två länder A och B samt folkökningen i procent för dessa. Programmet skriver därefter ut en tabell med årtal, hur stor folkökningen varit detta år (för varje land), och folkmängd i slutet av året. Detta fortsätter tills det land med från början mindre folkmängd har erhållit en folkmängd som är större än det andra landets. Detta förutsätter att landet med den mindre folkmängden har en större tillväxtprocent än det andra (kontrollera detta när indata matats in).

Programmet ska också besvara frågan "När går det (från början) befolkningsmässigt mindre landet om det större?" och det ska se ut enligt nedan när programmet körs (användarens indata ***kursiverad***).
Körexempel:

```
Ange startår: 1976
Ange land A:s folkmängd (i milj.): 200
Ange land A:s befolkningsökning (i %): 10.0
Ange land B:s folkmängd (i milj.): 100
Ange land B:s befolkningsökning (i %): 50.0
Årtal Ökning A Folkmängd A Ökning B Folkmängd B
1976    ---    200.00         ---   100.00
1977  20.00    220.00       50.00   150.00
1978  22.00    242.00       75.00   225.00
1979  24.20    266.20      112.50   337.50
År 1979 har land B gått om land A i befolkning.
```

Födelsedagsproblemet (frivillig)

Beräkna sannolikheten p(n) för att bland n personer åtminstone två har samma födelsedag, samt bestäm det n för vilket p(n) är större än 0.5. Vi bortser från skottår och att nativiteten inte är jämnt fördelad över året.

p(n)=1 - 365/365 * 364/365 * 363/365 ... (366-n)/365

Programmet ska läsa in ett maxvärde på n (exempelvis 100) och sedan skriva ut en tabell över n och p(n) för n = 1, 2, ... maxvärdet och till sist skriva ut det lägsta n-värde för vilket p(n) är större än 0.5

Kalendern (frivillig)

Skriv ett program som skriver ut en kalender för en månad. Utskriften ska se ut så här för en viss månad:

```
Må Ti On To Fr Lö Sö
           1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
```

Indata till programmet är veckodagsnumret (1 - 7) för första dagen i månaden och antalet dagar som månaden består av. I exemplet ovan är veckodagsnumret 4.

Linköpings Tekniska Högskola
Instutitionen för Datavetenskap (IDA)
Torbjörn Jonsson, Erik Nilsson 2015-05-26

Lab 2: Underprogram

Följande laboration introducerar underprogram; procedurer, funktioner och operatorer.

Mål

I denna laboration kommer du att lära dig:

- Hur man skriver underprogram och hur dessa anropas.
- Hur man lämpligtvis delar upp ett problem för att göra det lättare att lösa.
- Vad parametrar är, och hur de används.
- Vad kommandoradsargument är.

Uppgift

I denna laboration är del A och del B obligatoriska. Därefter får du välja fritt vilka av de övriga uppgifterna du vill göra, men **du måste göra minst 2 av uppgifterna C till H**.

Leta fram din kod från laboration 1. Kopiera den och lägg den i en ny mapp för lab 2. Du kommer att behöva koden i första deluppgiften.

Det finns ett par givna filer på hemsidan som handlar om slumpning av tal, hur man konverterar bokstäver till tal och vice versa och hur man använder kommandoradsargument. Vissa av uppgifterna involverar dessa element.

Del A: Momstabellen med underprogram

Du skall nu förbättra din lösning från förra laborationen. Arbeta om din lösning från lab 1 så att den klarar följande krav:

- Det skall finnas **ett** underprogram som sköter inläsningen av varje data (första pris, sista pris, steg och momsprocent). Underprogrammet skall även sköta felhanteringen. D.v.s. låta användaren mata in igen om datat är för stort eller för litet. Huvudprogrammet skall alltså anropa detta underprogram fyra gånger (ett för varje data).
- Det skall finnas ett underprogram som skriver ut tabellen, som skall anropas från huvudprogrammet. Detta underprogram skall använda sig av ytterligare ett underprogram som skriver ut de enskilda tabellraderna.
- Huvudprogrammet behöver alltså inte innehålla mer än fem satser!

Linköpings Tekniska Högskola
Instutitionen för Datavetenskap (IDA)
Torbjörn Jonsson, Erik Nilsson 2015-05-26

Del B: Miniräknaren

I denna uppgift skall du göra en liten miniräknare. Miniräknaren skall ha följande funktioner:

- Beräkna N-fakultet.
- Addera två heltal.
- Addera ett heltal med ett flyttal och vice versa.
- Addera två flyttal.

Det finns ett givet huvudprogram på hemsidan där själva in- och utmatningen är klar. Det du behöver göra är att skriva de underprogram som löser problemen ovan. I denna uppgift skall du **endast** använda dig av funktioner. Du skall alltså inte göra några underprogram som procedurer.

Som du kanske vet så kan man inte använda operatorn "+" om operanderna har olika datatyp. T.ex. följande skulle ge kompileringsfel:

```
X := 3.0 + 2;
```

Lyckligtvis skapar du i denna uppgift två funktioner som löser detta. Nu skall du skriva om dessa till operatorer så att satsen ovan (och även den variant där heltalet står till vänster och flyttalet till höger) fungerar. Modifiera även anropet i huvudprogrammet så att du verkligen ser att detta fungerar som det skall.

KRAV: När du har fått programmet att fungera skall du flytta ut N-fakultetfunktionen till en separat fil (du måste du skapa både en .ads- och en .adb-fil för detta).

Linköpings Tekniska Högskola
Instutitionen för Datavetenskap (IDA)
Torbjörn Jonsson, Erik Nilsson 2015-05-26

Del C: Den lånade bilen

Vi har lånat en bil av en kompis, och vi har lovat att lämna tillbaka bilen med lika mycket bensin i tanken som när vi lånade den. Tyvärr vet vi inte hur mycket som fanns i tanken från början, hur mycket bilen drar eller ens hur många liter tanken rymmer. Vi har dock kommit fram till följande plan.

1. Vi kör först en sträcka S1 mil, som vi vet exakt hur lång den är. I slutet av sträckan tankar vi T1 liter bensin, så att tanken blir full.
2. Vi kör sedan en till sträcka S2 mil, som vi också vet hur lång den är. I slutet av sträckan tankar vi T2 liter bensin så att tanken återigen blir full.
3. Det gick åt T2 liter att köra sträckan S2, alltså drar bilen K = T2 / S2 liter per mil.
4. Under sträckan S1 gick det alltså åt S1 * K liter. Men vi tankade T1 liter, d.v.s ur tanken saknades det från början T0 = T1 - S1 * K liter.
5. Vi kör sedan ytterligare exakt S3 = T0 / K mil och sedan lämnar vi bilen där. Det blir kanske lite jobbigt för kompisen att hämta upp bilen, men vi håller i alla fall vad vi har lovat.

Skriv ett program där användaren får mata in S1, T1, S2 och T2 och som sedan matar ut K, T0 och, S3.

Körexempel 1:

```
Först kör vi sträckan S1 (i mil): 2
Vi tankar nu full tank, T1 (i liter): 15
Nu kör vi sträckan S2 (i mil): 5
Vi tankar nu full tank igen, T2 (i liter): 5

Bilen drar 1.00 liter / mil.
Ur tanken saknades det från början 13.00 liter.
Vi kör sedan exakt 13.00 mil och lämnar bilen där!
```

Körexempel 2:

```
Först kör vi sträckan S1 (i mil): 3
Vi tankar nu full tank, T1 (i liter): 8
Nu kör vi sträckan S2 (i mil): 10
Vi tankar nu full tank igen, T2 (i liter): 15

Bilen drar 1.50 liter / mil.
Ur tanken saknades det från början 3.50 liter.
Vi kör sedan exakt 2.33 mil och lämnar bilen där!
```

KRAV: Du skall ha enskilda underprogram för beräkningen av K, T0 och S3. Skall dessa vara procedurer eller funktioner? Motivera.

Linköpings Tekniska Högskola
Instutitionen för Datavetenskap (IDA)
Torbjörn Jonsson, Erik Nilsson 2015-05-26

Del D: Den allseende pyramiden

Den allseende pyramiden är en symbol som visar ett allseende öga som finns ovanför en pyramid (används bl.a. av Frimurarordern). Helhelten blir ofta lite likt en triangel. Det allseende ögat brukar ofta tolkas som en symbol för Guds öga som vakar över mänskligheten. I denna uppgift skall du få rita en helt ny version av denna allseende pyramid som vi i kursledningen har skapat (inte så snygg som andra varianter, men vi tar inte åt oss någon ära heller). Vi tänker oss att någon fotograferat detta monument och valt att se olika mycket av själva pyramiden. Detta anges av användaren. Då ögat är allseende tänker vi oss att det ser alla kombinationer av skrift som skrivits och vi fyller därför pyramiden med slumpmässiga versaler i intervallet ’A’ - ’Z’. Detta skall ditt program också göra därför kan det bli lite olika resultat vid olika programkörningar.

Körexempel 1:

```
Ange pyramidens storlek: 1
    -_ | _-
_ _ //.\\ _ _
   _\\"//_
    - | -
  _________
 /UXNINNGGN\
```

Körexempel 2:

```
Ange pyramidens storlek: 3
     -_ | _-
  _ _ //.\\ _ _
     _\\"//_
      - | -
    _________
   /EUICDOPGR\
  /KNBWFDQSCPI\
 /IAYFMFIWUTUME\
```

Körexempel 3:

```
Ange pyramidens storlek: 4
       -_ | _-
    _ _ //.\\ _ _
       _\\"//_
        - | -
      _________
     /HSJGIOYMM\
    /IAHWCLXSYZX\
   /YQRISOXXODXID\
  /XCOCBIZNTYLUAJP\
```

TIPS: Ledtexten står alltid direkt till vänster i terminalen vid körning. Det lättaste sättet att finna var tecknen står är att jämföra med ledtextens positioner.

KRAV: Du skall dela upp ditt program i lämpliga underprogram. Inget huvud-/underprogram får ha mer än 10 satser mellan begin och end.

Linköpings Tekniska Högskola
Instutitionen för Datavetenskap (IDA)
Torbjörn Jonsson, Erik Nilsson 2015-05-26

Del E: "Speciella" Tal

Vissa tal är lite "speciella". Tag t.ex. talet 94311. Om man multiplicerar all siffrorna i talet så får man:

9 * 4 * 3 * 1 * 1 = 108.

Och om man summerar kvadraten av varje siffra i talet så får man:

81 + 16 + 9 + 1 + 1 = 108

Uppenbarligen är det så, att för vissa tal gäller det att produkten av siffrorna är lika med summan av kvadraterna på siffrorna - häftigt!

Skriv ett program som skriver ut alla sådana speciella tal som ligger mellan 1 och N där N är ett tal som matas in ***på kommandoraden*** då programmet startas.

KRAV: Skriv ett underprogram som sköter beräkningen av produkten, och ett som sköter beräkningen av summan av kvadraterna. Skall dessa underprogram vara procedurer eller funktioner?

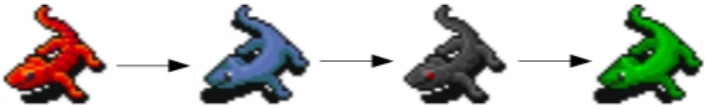
Linköpings Tekniska Högskola
Instutitionen för Datavetenskap (IDA)
Torbjörn Jonsson, Erik Nilsson 2015-05-26

Del F: Kamorferna

På planeten kamorfia finns det varelser som kallas kamorfer. Kamorfer finns i fyra färger, röda, blå, svarta och gröna. Kamorfpopulationen förändras varje år på följande vis: Varje år ökar antalet röda och gröna kamorfer med 1, **såvida inget** av följande gällar:

1. Vart 67:e år dör alla röda kamorfer. Alla blå kamorfer blir röda, alla svarta blir blå och alla gröna blir svarta. Förändringen sker samtidigt, och illustreras i bilden nedan

2. Vart 101:a år dör alla gröna kamorfer. Alla svarta kamorfer blir gröna, alla blå kamorfer blir svarta och alla röda kamorfer blir blå. Förändringen sker samtidigt, och illustreras i bilden nedan.



3. Vart 199:e år blir alla röda kamorfer blå, alla svarta kamorfer blir röda, alla gröna kamorfer blir svarta och alla blå kamorfer blir gröna. Förändringen sker samtidigt, och illustreras i bilden nedan.

Om fler av ovanstående tre regler är sanna samtidigt så utför man endast en av dem, och de prioriteras i ordningen ovan. Om man t.ex. betraktar år 6767 (= 67 * 101) så skulle endast regel 1 appliceras.

Vi antar nu att nuvarande år är år 1, och att det detta år finns 31 röda, 6 blå, 55 svarta, och 104 gröna kamorfer. Skriv ett program som tar in ett årtal ***på kommandoraden*** då programmet startar . Programmet skall sedan skriva ut hur många kamorfer det finns av varje färg för detta år.

Körexempel (startat från kommandoraden):

```
kamorf_programmet 100000
År 100000 finns det 9 röda, 2210 blå, 0 svarta och 76 gröna kamorfer.
```

KRAV: Du skall dela upp ditt program i lämpliga underprogram. Inget huvud-/underprogram får ha mer än 10 satser mellan begin och end.

Linköpings Tekniska Högskola
Instutitionen för Datavetenskap (IDA)
Torbjörn Jonsson, Erik Nilsson 2015-05-26

Del G: Kasta Tärning

Skriv ett program som simulerar ett eller flera tärningsslag. Resultatet på tärningarna skall slumpas. Med tärning avses en konventionell sexsidig tärning, även fast prickarna inte kanske riktigt ser ut som de brukar.

Programmet skall även mata ut summan av tärningsresultaten.

Körexempel 1:

```
Mata in antal slag: 3

+-----+
| *   |
|   * |
+-----+

+-----+
|* * *|
|* * *|
+-----+

+-----+
|* * *|
|* * *|
+-----+

Summan blev: 14
```

Körexempel 2:

```
Mata in antal slag: 5

+-----+
|     |
|  *  |
+-----+

+-----+
|*   *|
|  *  |
+-----+

+-----+
| *   |
|   * |
+-----+

+-----+
|* * *|
| * * |
+-----+

+-----+
| * * |
| * * |
+-----+

Summan blev: 15
```

Körexempel 3:

```
Mata in antal slag: 2

+-----+
|* * *|
| * * |
+-----+

+-----+
| * * |
| * * |
+-----+

Summan blev: 9
```

KRAV: Gör lämpliga underprogram. Inget huvud-/underprogram får ha mer än 10 satser mellan begin och end.

Linköpings Tekniska Högskola
Instutitionen för Datavetenskap (IDA)
Torbjörn Jonsson, Erik Nilsson 2015-05-26

Del H: Binära namn

Skriv ett program som låter användaren mata in sitt namn (kan vara godtyckligt långt, men utan å/ä/ö). Programmet skall sedan skriva ut namnet på ett 8-bitars binärt ascii-format. T.ex. motsvarar tecknet 'A' position 65 i teckentabellen. 65 binärt (med 8 bitar) blir 01000001. Du kan använda funktionen End_Of_Line (finns i Ada.Text_IO) för att kontrollera huruvida nästa tecken i tangentbordsbufferten är ett entertecken.

Körexempel 1:

```
Mata in ditt namn: Adam
Ditt namn binärt:
01000001
01100100
01100001
01101101
```

Körexempel 2:

```
Mata in ditt namn: May
Ditt namn binärt:
01001101
01100001
01111001
```

Körexempel 3:

```
Mata in ditt namn: Josebellalina
Ditt namn binärt:
01001010
01101111
01110011
01100101
01100010
01100101
01101100
01101100
01100001
01101100
01101001
01101110
01100001
```

KRAV: Gör lämpliga underprogram. Inget huvud-/underprogram får ha mer än 10 satser mellan begin och end.

Linköpings Tekniska Högskola
Instutitionen för Datavetenskap (IDA)
Torbjörn Jonsson, Erik Nilsson 2015-05-26

Lab 3: Datastrukturer

Följande laboration introducerar datastrukturer; fält och poster.

Mål

I denna laboration kommer du att lära dig:

- Hur man definierar egna datatyper.
- Hur man indexerar i fält.
- Hur man använder sig av poster.
- Hur man gör egna uppräkningsbara typer.

Del A: Fält

Skapa ett program vari du definierar en ny datatyp, Ten_Ints_Array_Type. Datatypen skall vara ett fält med plats för 10 heltal. Skapa sedan ett underprogram Get, som läser in till en parameter av denna datatyp och ett underprogram Put som skriver ut fältet på skärmen. Låt huvudprogrammet testa dina underprogram. Skall underprogrammen vara procedurer eller funktioner? Motivera!

Gör sedan ett underprogram Find_Maximum, som hittar det största heltalet i fältet och vilket index som värdet fanns på och returnerar dessa tal. Skall underprogrammet vara en procedur eller en funktion?

Gör även ett underprogram Reverse, som vänder ordningen på heltalen i fältet. Skall underprogrammet vara en procedur eller en funktion?

Del B: Poster

I denna uppgift skall du definiera datatypen String_Type, som skall klara av att lagra text. En String_Type skall innehålla två delvariabler, Char_Array ett fält med 256 tecken, och ett heltal Length som talar om hur mycket av fältet som används för tillfället. String_Type kommer endast kunna hantera textsnuttar som är mindre än eller precis 256 tecken. Observera att String_Type är något helt separat från *String* som redan finns definierat i Ada.

Du skall nu skapa ett underprogram Get som kan läsa in till din String_Type. Underprogrammet skall läsa in tecken för tecken och lagra dessa i parametern tills antingen 256 tecken har lästs in, eller man stöter på ett enter-tecken i tangentbordsbufferten. Man kan kolla om det ligger ett enter-tecken i bufferten med funtkionen End_Of_Line, som returnerar sant eller falskt.

Skapa sedan ett underprogram Put som skriver ut din String_Type på skärmen.

Till sist skall du även skapa en jämförelseoperator "=", så att man kan jämföra om två variabler av typen String_Type är lika. Tänk till här så att du bara jämför de delar av variablerna som är relevanta.

Självklart gör du även ett huvudprogram som testar dina underprogram.

Linköpings Tekniska Högskola
Instutitionen för Datavetenskap (IDA)
Torbjörn Jonsson, Erik Nilsson 2015-05-26

Del C: "Komplexa" datastrukturer

I denna uppgift skall du skapa en lite större datastruktur, Hero_Type, som skall användas för att representera hjältar och hjältinnor. Hero_Type skall innehålla följande delvariabler:

- Ett namn (använd din String_Type från del B)
- En ålder (ett heltal)
- Ett kön (välj själv en lämplig datatyp)
- En vikt (i kilogram, ett flyttal)
- En hårfärg (använd String_Type)
- En art (något av: Human, Elf, Orc, Halfling, Ogre, Lizardman)
- En ögonfärg (något av: Blue, Green, Brown, Gray, Yellow, Red, Black, Crazy)

För delarna som är "något av" skall du inte använda String_Type, du behöver skapa nya uppräkningsbara datatyper. Lämpligtvis kallar du dessa för Species_Type respektive Eye_Type.

Gör sedan ett underprogram Get och ett underprogram Put för din Hero_Type. Din Get behöver inte läsa in postens sista två delar om du inte vill. Om du ändå vill få till det, fråga din assistent om tips.

Om man vill hårdkoda en hjälte i programmet (d.v.s den skall inte matas in med Get), hur gör man detta?

Linköpings Tekniska Högskola
Instutitionen för Datavetenskap (IDA)
Torbjörn Jonsson, Erik Nilsson 2015-05-26

Lab 4: Hantering av poster och enkla paket

I denna laboration kommer du att skapa ett paket med rutiner som kan hantera ett datum. För lagring av datum används först datatypen post och sedan fält.

Mål

Du skall efter denna laboration kunna skapa egna paket med rutiner som senare kan användas i ett större program. Du skall kunna hantera undantag, och veta hur man fångar/kastar dem. Du skall även ha en förståelse för varför man vill skydda/gömma implementation i ett paket.

Uppgift

Att lagra ett datum går att göra på många olika sätt. T.ex. kan man lagra datumet som en textsträng ”1997-01-04” eller som tre heltal. I denna uppgift skall du lagra datumet i en post i vilken år, månad och dag lagras som tre heltal. Uppgiften är uppdelad för att du ska undvika alltför många saker på en gång. Man bör alltid försöka dela upp stora problem i mindre. Skriv ett litet testprogram som använder dina rutiner allt eftersom du programmerar.

Del A:

Skapa den datatyp som skall representera ett datum enligt beskrivningen ovan. Skriv dessutom de två procedurerna Get och Put som läser in ett datum från tangentbordet respektive skriver ut ett datum på skärmen. Ett exempel på kod som skall läsa in ett datum:

```
begin
Ada.Text_IO.Put(”Ange ett datum: ”);
Get(Date); -- Detta är anropet till din ”Get”.
end ...;
```

Inläsningen skall då se ut enligt följande:

```
Ange ett datum: 1997-01-04
```

Utskriften av ett datum skall följa svensk standard, d.v.s. på formen ÅÅÅÅ-MM-DD. Ett exempel på kod som skall skriva ut ett datum:

```
begin
Ada.Text_IO.Put(”Ett datum: ”);
Put(Date); -- Detta är anropet till din ”Put”.
end ...;
```

Utskriften skall då bli:

```
Ett datum: 1997-01-04
```

Observera att du skall skriva procedurerna Get och Put och att dessa får innehålla anrop till de redan färdigdefinerade Get och Put som finns i Ada.Text_IO.

Linköpings Tekniska Högskola
Instutitionen för Datavetenskap (IDA)
Torbjörn Jonsson, Erik Nilsson 2015-05-26

Del B:

Skriv funktionerna Next_Date och Previous_Date som får ett datum (t.ex. 1997-01-04) som indata och returnerar nästa (1997-01-05) respektive föregående (1997-01-03) datum. Tänk på att det är olika antal dagar i olika månader. Du behöver inte ta hänsyn till skottår, men om du vill det så är det skottår var fjärde år. Årtalet skall vara jämnt delbart med 4 om det är skottår. Dock finns det några undantag. Det är inte skottår då årtalet är jämnt delbart med 100. Dock är det skottår om årtalet är jämnt delbart med 400.

Del C:

De olika procedurerna som du nu skapat skulle kunna användas i ett antal olika program och för att man på ett enkelt sätt skall kunna återanvända dessa skapar vi ett paket av alltsammans. När du skapar ditt paket skall du inte ta med huvudprogrammet utan bara de procedurer och funktioner som skall återanvändas senare i andra program. I denna laboration är det alltså Get, Put, Next_Date och Previous_Date som skall finnas med i paketet samt hjälpprocedurer och hjälpfunktioner.
Ditt testprogram skall nu inkludera ditt paket och använda sig av de rutiner (procedurer och funktioner) som finns definierade där. Programmet skall givetvis fungera som tidigare.

Del D:

Skriv, i ditt paket, operatorerna ”<”, ”=” och ”>” för datum (t.ex. 1997-04-04 och 1998-01-05) returnerar sant värde om första datumet är mindre, lika med respektive större än det andra datumet. De två datumen skall vara parametrar till funktionerna. Du skapar nu operatoröverlagringar som kan användas på samma sätt som motsvarande operatorer för heltal eller flyttal, men i detta fall gäller de för värden av typen datum.

Del E:

Du skall nu lägga till lite kontroller i din rutin Get. Om användaren skulle mata in en ogiltig dag, månad eller ett ogiltigt år skall paketet kasta undantagen Day_Error, Month_Error, respektive Year_Error. Kontrollera även så att du kan fånga dessa undantag i ditt testprogram.

Linköpings Tekniska Högskola
Instutitionen för Datavetenskap (IDA)
Torbjörn Jonsson, Erik Nilsson 2015-05-26

Del F:

Skriv ett litet separat program som använder ditt paket (utöver ditt testprogram som du har sedan tidigare). Ditt program skall till att börja med läsa in ett datum som är nuvarande datum. Därefter skall programmet läsa in ett annat datum som antingen ligger före eller efter i tiden. Programmet skall sedan ange hur många dagar före eller efter det andra datumet var jämfört med det första.

Körexempel 1:

```
Mata in dagens datum: 2015-06-02
Mata in ett annat datum: 2015-06-10

Det andra datumet är 8 dagar efter det första.
```

Körexempel 2:

```
Mata in dagens datum: 2015-07-05
Mata in ett annat datum: 2015-06-29

Det andra datumet är 7 dagar innan det första.
```

TIPS: Använd de underprogram som du har gjort i ditt paket.

Del G:

Du skall nu ändra ditt paket så att din datatyp använder sig av ett fält istället för en post. Efter att du har ändrat skall ditt program från del E fortfarande gå att kompilera, utan ändringar.

Varför är detta sista krav viktigt?

Linköpings Tekniska Högskola
Instutitionen för Datavetenskap (IDA)
Torbjörn Jonsson, Erik Nilsson 2015-05-26

Lab 5: Rekursion

I denna laboration kommer vi att titta lite närmare på hur rekursion fungerar.

Mål

Du ska efter denna laboration kunna:

- förstå hur rekursion går till
- använda rekursion
- förstå när man skall undvika rekursion
- veta vad begreppet ändrekursion betyder
- använda iteration i kombination med rekursion

Tonvikt läggs på:

- struktureringen av problemet
- läsbarhet av programkod

Del A (frivillig):

Du skall skriva den rekursiva implementationen av funktionen som har deklaration enligt följande:

```
function Factorial(N : in Natural) return Positive;
```

Funktionen skall beräkna N! utifrån det N som kommer in som parameter. För att visa att funktionen fungerar krävs förstås ett huvudprogram som anropar funktionen.

Del B (frivillig):

Du skall skriva de rekursiva implementationen av funktionen som har deklaration enligt följande:

```
function Power(X : in Float;
               N : in Integer) return Float;
```

Funktionen skall beräkna X^N.

Del C:

Du skall skriva en funktion som beräknar det N:te talet i Fibonacci-serien. Indata till funktionen skall vara N och funktionen skall givetvis vara rekursiv. Funktionen skall heta Fib.

Definition av Fibonacci-serien:

```
Fib(1) = 1
Fib(2) = 1
Fib(N) = Fib(N - 1) + Fib(N - 2)
```

Serien börjar alltså som följer:

```
1 1 2 3 5 8 13 21 34 ...
```

Testa med ett stort tal på N och se vad som händer. Varför blir det så?

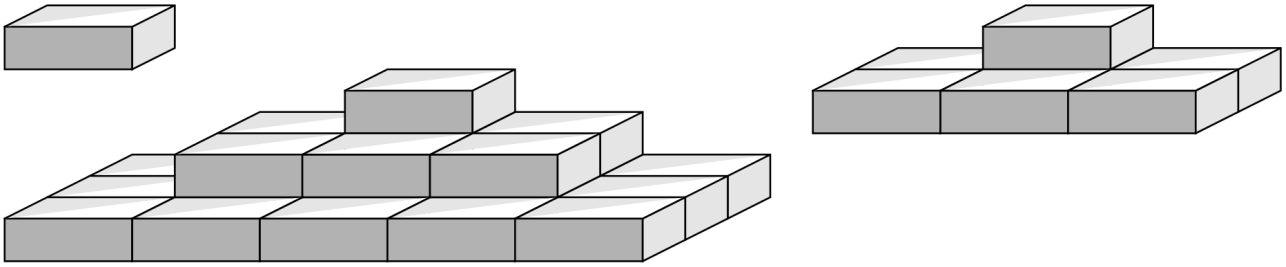
Linköpings Tekniska Högskola
Instutitionen för Datavetenskap (IDA)
Torbjörn Jonsson, Erik Nilsson 2015-05-26

Del D:

Gör en kopia av din kod där du skapade datatypen String_Type (Lab 3, del B). Du skall i denna uppgift göra om detta underprogram så att det är rekursivt. Underprogrammet skall funktionellt vara precis som tidigare.

Del E:

När man bygger sitt hus och kommer till den punkt där man skall bygga taket får man lite problem. Man behöver någon form av ställning. Antag att man bygger den av LECA-stenar på så sätt att man kan se det som en trapp oavsett från vilket håll man kommer (lite grann som en kapad pyramid). Se i figuren för att se hur trappen skulle se ut för hushöjderna 2, 3 och 4 (självklart behövs ingen trappa om man bara bygger ett lager stenar i huset).



Din uppgift är att skriva det program som räknar ut hur många stenar det behövs för att bygga ”trappan” givet en viss höjd på huset.

KRAV: Du skall ha en rekursiv funktion som beräknar antalet stenar i ditt program.

Körexempel 1:

```
Mata in husets höjd (HH = antal stenar i höjdled): 10
Det behövs 525 stenar för att bygga trappan.
```

Körexempel 2:

```
Mata in husets höjd (HH = antal stenar i höjdled): 100
Det behövs 651750 stenar för att bygga trappan
```

Linköpings Tekniska Högskola
Instutitionen för Datavetenskap (IDA)
Torbjörn Jonsson, Erik Nilsson 2015-05-26

Uppgift 6 (frivillig)

Skriv ett program som läser in ett antal rader med tre tecken långa strängar. När inmatningen ”---” kommer skall alla tidigare inmatningar skrivas ut i omvänd ordning. Det kan vara godtyckligt många rader att mata in (vilket gör att du inte kan lagra dem i ett fördimensionerat fält).

Körexempel:

```
Mata in ett antal rader (3 tecken per rad, avsluta med ”---”):
aaa
bbb
ccc
---
Du matade in följande rader:
ccc
bbb
aaa
```

Linköpings Tekniska Högskola
Instutitionen för Datavetenskap (IDA)
Torbjörn Jonsson, Erik Nilsson 2015-05-26

Lab 6: Enkel listhantering

I denna laboration kommer vi att implementera en dynamisk datastruktur, som till skillnad från fält (array) och poster (record) kan växa vid behov, och som till skillnad från filer kan växa i flera dimensioner.

Mål

Du ska efter denna laboration kunna:

- använda pekarstrukturer
- förstå hur rekursion går till
- förstå hur en privat datatyp fungerar och hur en sådan implementeras
- förstå hur ett generiskt paket fungerar och hur ett sådant instansieras

Tonvikt läggs på:

- struktureringen av problemet
- val av underprogram och parametrar samt namn till dessa
- att göra små elementära operationer istället för komplexa

Uppgift

Du ska skriva ett paket "sorted_hero_list" som hanterar lagringen av en enkel form av dynamiska listor. Datat som skall lagras i varje element i listan skall vara en post, mer specifikt din datatyp från lab 3, Hero_Type. De data som ska lagras i listan ska lagras sorterade i stigande ordning med avseende på ålder, d.v.s. den yngsta hjälten först. Alla procedurer och funktioner som kräver genomgång av listor ska göras rekursiva.

I uppgifterna står inget om några huvudprogram. Givetvis måste sådana göras för att kunna testa paketen. Dessa huvudprogram behöver i sig inte göra något speciellt utan är till för dig som programmerare för att kunna på något sätt ”verifiera” eller åtminstone göra det troligt att paketen fungerar som de skall.

Var inte rädd bara för att det blir många paket. Paketen innehåller sina egna delar och det blir på detta sätt lättare att identifiera var olika fel uppkommit om detta händer.

Nedan refererar "söknyckel" till hjältens ålder, medans "data" refererar till själva hjälten. Detta medför att alla hjältar i listan har unika åldrar.

Del A:

Börja med att göra ett paket (hero_handling.ads/.adb) där du lägger allt som har med Hero_Type att göra. Lägg datatypen String_Type i ett separat paket som du kallar för string_handling. Givetvis skall alla datatyper vara privata i sina respektive paket.

Linköpings Tekniska Högskola
Instutitionen för Datavetenskap (IDA)
Torbjörn Jonsson, Erik Nilsson 2015-05-26

Del B:

Skapa först den privata datatypen List_Type som motsvarar en lista och en del operationer för att hantera en sådan. Det paket som du ska skapa ska ligga i filerna ”sorted_hero_list.ads” och ”sorted_hero_list.adb”. Tänk efter innan du börjar koda så slipper du en massa besvär. För att skapa sig förståelse för hur pekarstrukturen fungerar är det viktigt att **rita figurer!** Den privata datatypen ska implementeras som en pekare till en post, där posten ska innehålla dels själva datat (hjälten) och dels en pekare till nästa post i listan. Hela paketet ska ha förutsättningen att de data som lagras i listan ska vara kopior av det som stoppas in.

De uppgifter som här följer är de underprogram (metoder) som utgör gränssnittet mellan paketet och ett annat program (som använder paketet). I vissa fall kan det vara bra att införa hjälpoperationer för att underlätta hanteringen av de problem som uppstår. Dessa hjälpoperationer göms då lämpligen undan i filen ”sorted_hero_list.adb”. Du bör läsa igenom alla uppgifter innan du börjar programmera och lösa problemen. Det kan hända att du i tidigare uppgifter har nytta av rutiner som finns specificerade senare.

Funktionen Empty
Skriv en funktion som kontrollerar om listan är tom eller ej. Returnera ett sanningsvärde.

Proceduren Insert
Du ska skriva en procedur som stoppar in en hjälte sorterat i stigande ordning (m.a.p. ålder) i en redan sorterad lista. De parametrar som behövs är en lista och en hjälte. Proceduren ska inte stoppa in datat om det redan existerar ett data med samma söknyckel i listan.

Proceduren Put
Skriv en procedur som skriver ut hela listan på skärmen. Du får själv välja hur utskriften ska se ut. Som indata fås en lista. Observera att listan ska vara opåverkad efter denna rutin. Ett bra test är att skriva ut listan två gånger (från testprogrammet).

Funktionen Member
Du ska skriva en funktion som går igenom den sorterade listan och letar efter ett data med en viss söknyckel och returnera ett sanningsvärde som talar om ifall datat existerade i listan eller ej. Indata till funktionen är listan och en söknyckel.

Proceduren Remove
Nu ska du skriva en procedur som plockar bort ett element ur en sorterad lista (d.v.s. listan ska bli ett element “kortare”). De parametrar som ska finnas är en lista och den söknyckel som anger vilket data som ska tas bort. Du får förutsätta att alla söknycklar är unika i listan. Om det inte finns något element med denna söknyckel ska ett undantag (”exception”) resas. Dock behöver inte undantag tas om hand (fångas) i ditt program utan det får helt enkelt vara så att programkörningen avbryts. Observera att man måste återlämna minnesutrymmet för de poster man länkar ur listan. I annat fall får man något som kallas minnesläckor och detta gör i värsta fall att datorns minne tar slut.

Linköpings Tekniska Högskola
Instutitionen för Datavetenskap (IDA)
Torbjörn Jonsson, Erik Nilsson 2015-05-26

Proceduren Delete
Du ska nu skriva en procedur som tar bort en hel lista (återlämnar minnesutrymmet för alla element i listan). Som parameter till proceduren fås en lista. Givetvis ska pekaren till listan vara NULL efter avklarat verk.

Funktionen Find
Du ska skriva en funktion som letar reda på ett element i en lista (ej ta bort elementet ur listan). Som indata fås en lista och en söknyckel. Funktionen ska returnera det data som matchar söknyckeln. Om det inte finns något element med denna söknyckel ska ett undantag (”exception”) resas.

Proceduren Find
Nu ska du skriva en procedur som motsvarar funktionen ovan. Samma funktionella krav som för funktionen ovan. Tips: Fundera på vad som behöver ändras och hur du på ett enkelt sätt slipper rätta i både funktionen och proceduren om man upptäcker att det är något fel.

Att man har samma namn på två underprogram kallas att man överlagrar underprogrammen. Detta innebär dock inte att det ena eller andra underprogrammet försvinner. Det innebär att båda finns tillgängliga samtidigt.

Funktionen Length
Skriv en funktion som beräknar längden av en lista. Längden definieras som antalet element i listan.

OBS: Eftersom Hero_Type är en privat datatyp kommer du att behöva ett eller fler extra underprogram i ditt hjältepaket för att vissa av ovanstående underprogram skall fungera, vilket/vilka då?