Funktionell programmering

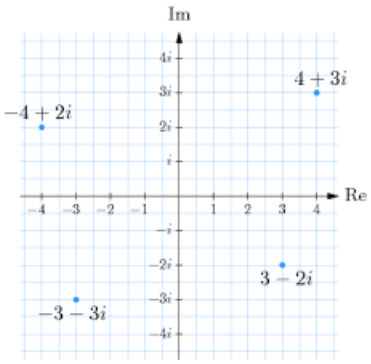
Laboration #1: ***Komplexa tal***

Deadline 1: 210920 klockan 13:00

Syftet med den här laborationen är att ge er träning i att stegvis utveckla ett program bestående av en mängd funktioner. Då detta är den första laborationen, kommer ni få hjälp med vilka funktioner som skall tillverkas och i vilken ordning. I ett skarpt fall är så klart uppdelningen i funktioner, och hur de bygger på varandra, en mycket viktig del av utvecklingen. På samma sätt bör vi vara medvetna om att vissa delar av laborationen skulle kunna lösas på ett bättre sätt genom att utnyttja koncept som vi kommer lära oss senare i kursen, exempelvis egna typer, rekursion och högre ordningens funktioner.

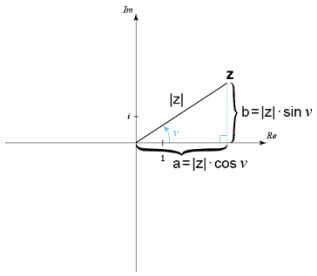
Bakgrund – komplexa tal

Laborationen behandlar komplexa tal. Ett komplext tal z utgör en punkt i det komplexa talplanet, se Figur 1 nedan.



Figur 1: Det komplexa talplanet

Ett komplext tal z består av en realdel längs x-axeln och en imaginärdel längs y-axeln. x-axeln kallas reella axeln och y-axeln kallas imaginära axeln. Man inför då en imaginär enhet i som har egenskapen $i^2 = -1$. Ett komplext tal kan därmed skrivas på formen a + bi. I Fig. 1 ovan ses fyra exempel på komplexa tal. Formen z = a + bi kallas för **rektangulär form**. Eftersom ett komplext tal är en punkt i det komplexa talplanet, kan z dock även skrivas på ett annat sätt, nämligen i **polär form** se Figur 2 nedan.



Figur 2: Polär form

I den polära formen består z i stället av r=|z|, vilket är avståndet från origo till z=a+bi och vinkeln v mellan x-axeln och linjen från origo till z=a+bi. Formen blir sedan: $z = r(\cos v + i \sin v)$. Ofta används en förkortad skrivform $z = r\lfloor v$. Exempelvis har det komplexa talet z=1+i (i rektangulär form) r=$\sqrt{2}$ och v=$\pi/4$ radianer vilket skrivs som $z = \sqrt{2}(\cos\frac{\pi}{4} + i \sin\frac{\pi}{4})$ eller $z = \sqrt{2}\left\lfloor\frac{\pi}{4}\right.$.

Sambanden mellan de båda representationerna ges nedan:

$$a = \mathrm{r}\cos v$$

$$b = r \sin v$$

$$r = \sqrt{a^2 + b^2}$$

$$v = \arctan\frac{b}{a}\ om\ a > 0, b > 0$$

$$v = \arctan\frac{b}{a} + 2\pi\ om\ a > 0, b < 0$$

$$v = \arctan\frac{b}{a} + \pi\ om\ a < 0$$

$$v = \frac{\pi}{2}\ om\ a = 0\ och\ b \geq 0$$

$$v = \frac{3\pi}{2}\ om\ a = 0\ och\ b < 0$$

När vi nu skall tillverka ett Haskell-program som hanterar komplexa tal behöver vi införa en Haskell-representation för komplexa tal. I den här laborationen väljer vi en representation som en trippel bestående av en sträng och två flyttal. Strängen är antingen ”R” eller ”P” och talar om vilken form (rektangulär eller polär) som talet är givet på, medan de två talen är antingen a och b i den rektangulära formen, eller r och v i den polära. Så för z ovan skulle detta skrivas (”R”, 1.0, 1.0) respektive (”P”, 1.4142, 0.7854) där så klart flyttalen skulle ha fler decimaler.

Generella principer

Att skriva program handlar inte bara om att få ett program som fungerar. Tvärtom skall man alltid sträva efter en hög kodkvalitet. I den här laborationen behöver ni tänka på bland annat följande saker:

- Sträva efter att ha små och tydliga funktioner.
- Använd existerande funktioner när ni tillverkar nya.
- Alla funktionsdeklarationer skall föregås av en typsignatur.

- Bryt bara isär de komplexa talen med hjälp av mönstermatchning när så behövs.
- Använd anonyma variabler när det är möjligt.
- Vi har ännu inte gått igenom felhantering, men ni bör ändå lägga till en minimal sådan i vissa funktioner. Mer konkret kan kommandot Error, vilket gör att programmet avbryts och en förklarande text skrivs ut, användas. Se körningsexemplet sist i laboration.
- En annan sak som vi inte gått igenom annat än kort på övning 2 är möjligheten att via kommandot type införa ett alias för en annan typ. Det här är en frivillig del av Lab 1, men ni rekommenderas att införa en typ Cplex som motsvarar den valda representationen av komplexa tal, samt då förstås använda typen Cplex i alla signaturer.

Uppgifter

Vi är nu äntligen beredda att börja koda!

Uppgift 1

- Skriv en funktion makeRec som tar två reella tal motsvarande a och b och returnerar det komplexa talet z = a + bi på rektangulär form.
- Skriv en funktion makePol som på motsvarande sätt tar två reella tal r och v och returnerar ett komplext tal på polär form. OBS: funktionen skall bara hantera vinklar mellan 0 och 2*pi. För andra argument skall ett fel ges.

Uppgift 2

- Skriv en funktion getRe som tar ett komplext tal på godtycklig form (alltså rektangulär eller polär) och returnerar realdelen.
- Skriv en funktion getIm som tar ett komplext tal på godtycklig form (alltså rektangulär eller polär) och returnerar imaginärdelen.
- Skriv en funktion getDist som tar ett komplext tal på godtycklig form (alltså rektangulär eller polär) och returnerar avståndet från origo till punkten som motsvarar det komplexa talet.
- Skriv en funktion getAngle som tar ett komplext tal på godtycklig form (alltså rektangulär eller polär) och returnerar vinkeln mellan x-axeln och punkten som motsvarar det komplexa talet. Notera här att vi alltid vill ha en positiv vinkel, medan de båda inbyggda funktionerna i Haskell atan och atan2 kan returnera negativa värden.

Uppgift 3

- Skriv en funktion toRec som tar ett komplext tal på godtycklig form (alltså rektangulär eller polär) och returnerar talet på rektangulär form.
- Skriv en funktion toPol som tar ett komplext tal på godtycklig form (alltså rektangulär eller polär) och returnerar talet på polär form.

Uppgift 4

Vi skall nu tillverka funktioner för de fyra räknesätten. Samtliga dessa funktioner skall fungera för komplexa tal oavsett representation. De får returnera resultatet i godtycklig form. Vad det gäller komplexa tal så görs addition och subtraktion lämpligast i rektangulär form medan multiplikation och division bör hanteras i polär form. Mer konkret:

$$(a + bi) + (c + di) = (a + c) + (b + d)i$$

$$(a + bi) - (c + di) = (a - c) + (b - d)i$$

$$r\lfloor u * s\lfloor v = r * s\lfloor (u + v)$$

$$\frac{r\lfloor u}{s\lfloor v} = \frac{r}{s}\lfloor (u - v)$$

- Skriv de fyra funktionerna compAdd, compSub, compMult och compDiv.

Uppgift 5

I laborationens sista steg skall vi tillverka några funktioner som använder listbyggare.

- Tillverka en funktion genCompList som tar in två par av heltal, dock givna som flyttal, (x, y) (z, w) och genererar en lista bestående av alla polära tal i rektangulär form där realdelen är från x till y och imaginärdelen från z till w.
  Exempel: `genCompList (-1,1)(2,3)`
  `[("R",-1.0,2.0),("R",-1.0,3.0),("R",0.0,2.0),`
  `("R",0.0,3.0),("R",1.0,2.0),("R",1.0,3.0)]`
- Tillverka en funktion listToPol som tar in en lista av komplexa tal och konverterar samtliga till polär form.
- Tillverka en funktion filterLengths som tar in ett tal k och en lista xs med komplexa tal och returnerar en lista bestående av de komplexa tal i xs som inte ligger längre från origo än k. Funktionen skall fungera för godtycklig representation och talen skall behålla den form de har.
- Tillverka en funktion filterQuadrant som tar in ett tal m (mellan 1 och 4) och en lista xs med komplexa tal och returnerar en lista bestående av de komplexa tal i xs som ligger i kvadrant m. Funktionen skall fungera för godtycklig representation och talen skall behålla den form de har.

Organisation

Laborationen skall lösas i par. Skapa en grupp bestående av två personer i Canvas och lämna in laborationen från den gruppen.

Inlämningen skall bestå av två filer:

1. Källkoden.
2. Väl valda körningsexempel.

Ni har tillgång till två handledningar per grupp. Dessa genomförs online och bokas via Canvas. På Canvas finns även ett diskussionsforum kopplat till laborationen där ni kan ställa generella frågor så att alla får samma information. Notera att ingen extra handledning kommer ges, exempelvis via e-mail. Slutligen, kom ihåg att samarbeten mellan grupper är förbjudet då laborationen utgör del av examinationen.

Körningsexempel

Observera att detta bara är några exempel – ert program behöver så klart kunna hantera andra anrop och argument.

```
> r1 = makeRec 1 2
> r2 = makeRec 1 (-1)
> p1 = makePol 2 (pi/4)
> r1
("R",1.0,2.0)
> p1
("P",2.0,0.7853982)
> makePol 2 (-pi/2)
*** Exception: Angle needs to be between 0 and 2*pi
CallStack (from HasCallStack):error, called at lab1.hs:6:59 in main:Main
> getRe p1
1.4142135
> getRe ("S", 2.0, 1.0)
*** Exception: Not complex
> getIm p1
1.4142135
> getAngle r1
1.1071488
> toRec p1
("R",1.4142135,1.4142135)
> toPol r2
("P",1.4142135,5.4977875)
> compAdd r1 r2
("R",2.0,1.0)
> compSub r1 p1
("R",-0.41421354,0.58578646)
> compMult r1 p1
("P",4.472136,1.8925469)
> xs = genCompList (-1,1)(-2,3)
> ys = listToPol (genCompList (-2,-1)(1,2))
> zs = xs ++ ys
> filterLengths 2 zs
[("R",-1.0,-1.0),("R",-1.0,0.0),("R",-1.0,1.0),("R",0.0,-2.0),("R",0.0,-
1.0),("R",0.0,0.0),("R",0.0,1.0),("R",0.0,2.0),("R",1.0,-1.0),
("R",1.0,0.0),("R",1.0,1.0),("P",1.4142135,2.3561945)]
> filterQuadrant 1 zs
[("R",1.0,1.0),("R",1.0,2.0),("R",1.0,3.0)]
> filterQuadrant 4 zs
[("R",1.0,-2.0),("R",1.0,-1.0)]
> filterQuadrant 2 zs
[("R",-1.0,1.0),("R",-1.0,2.0),("R",-1.0,3.0),
("P",2.236068,2.6779451),("P",2.828427,2.3561945),("P",1.4142135,2.35619
45),("P",2.236068,2.0344439)]
```