Tekniska Högskolan i Linköping
Institutionen för Datavetenskap (IDA)
Torbjörn Jonsson 2012-01-24

Ganska vettiga underprogram

I denna laboration kommer du att bekanta dig med underprogram i Ada. I Ada finns två olika typer av underprogram, procedurer och funktioner.

Mål

Du skall efter denna laboration kunna skriva underprogram i Ada. Dessutom skall du kunna välja vilken typ av underprogram som är lämplig till vad. För övrigt skall du få förståelse för parametrar och hur dessa används (både in-, ut- och in/ut-parametrar). Du skall dessutom förstå att man inte behöver ha all kod i samma fil.

Uppgift 1

Du skall skriva ett program som kommunicerar med användaren via en meny. Användaren skall kunna välja mellan olika funktioner, t.ex. konvertera mellan grader och radianer och vice versa eller räkna ut längden av en vektor givet två koordinater i rummet. Du skall därför skriva ett antal små procedurer och funktioner som utför enkla beräkningar eller andra uppgifter. Det är meningen att du skall använda dig av parametrar för att skicka data till och från underprogrammen. När det gäller funktionerna skall du inte skicka ut något via parametrarna. Dessa är enbart till för att funktionen skall få data att arbeta med.

TIPS: Det är inte säkert att du behöver börja med problem nummer 1. Det kan vara bättre att börja med problem 5. Läs igenom alla delproblemen innan du börjar programmera. Själva menyn är huvudprogrammet i denna uppgift.

Problem 1, Funktionerna To_Degree och To_Radian

Du skall skriva två funktioner. Den ena skall konvertera (omvandla) radianer till grader och den andra skall konvertera grader till radianer. In till funktionerna (som parameter/argument) skall ett reellt tal (flyttal) skickas som motsvarar den vinkel som skall konverteras.

Funktioner skall returnera sina resultat (funktioner gör detta generellt sett). INTE skriva på skärmen.

Problem 2, Funktionen Vector_Length

Du skall skriva en funktion, Vector_Length, som tar in tre reella tal via parameterlistan. De tre parametrarna skall motsvara en koordinat i rummet, d.v.s. x, y och z. Funktionen skall beräkna längden av den vektor som går från origo till denna koordinat. Längden blir också ett reellt tal.

Problem 3, Funktionen To_Integer

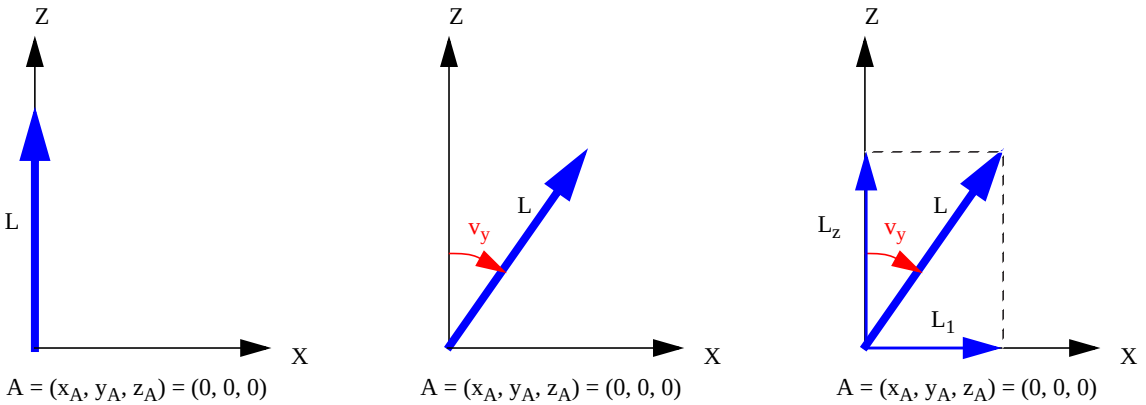
Du skall skriva en funktion, To_Integer, som tar in ett flyttal och returnerar det heltal som ligger närmast detta flyttal (d.v.s. det avrundade värdet av flyttalet).

Fundering: Problemen ovan är funktioner. Vad har dessa gemensamt? Varför var de inte procedurer?

Problem 4, Proceduren Create_Vector_Data

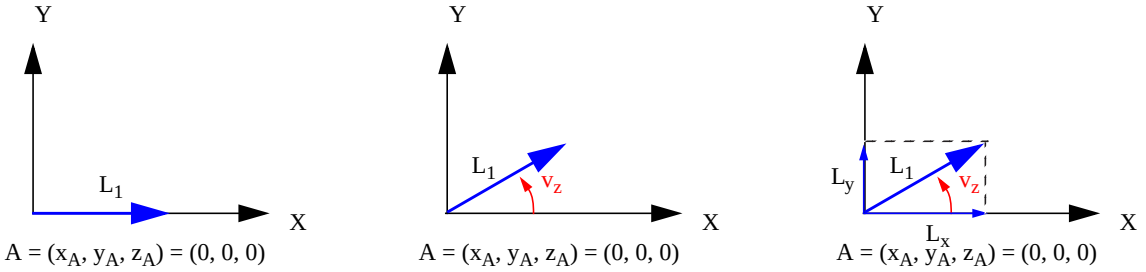
En liten bakgrund till denna deluppgift.

Vi börjar med en beskrivning som förklarar hur den slutgiltiga figuren är uppbyggd. Det kan annars bli lite abstrakt. Vi begränsar oss lite till en början och antar att vi har en vektor med längden L som utgår från punkten A (som ligger i origo) och följer Z-axeln.



Från ursprungsläget vrider vi ner vår vektor mot X-axeln med en given vinkel v_y. Denna vinkel får endast ligga i intervallet 0-180 grader. Nästa steg är att vi projicerar vektorn L på de två axlarna. Detta ger att vi får två nya ”vektorer” som vi kallar L_1 och L_z som ligger längs axlarna. Vi återkommer till varför vi kallar den ena för L_1 (vi har en anledning att inte kalla den L_x som man kanske i första stund skulle vilja kalla den).

Om vi nu tänker oss att vi ser detta ”uppifrån” (d.v.s. att man ser från pilen på Z-axeln och tittar ner mot punkten A) så ser vi X-Y-planet istället. Vi ritar inte ut L_z och L i bild, men de finns givetvis kvar i vårt minne.



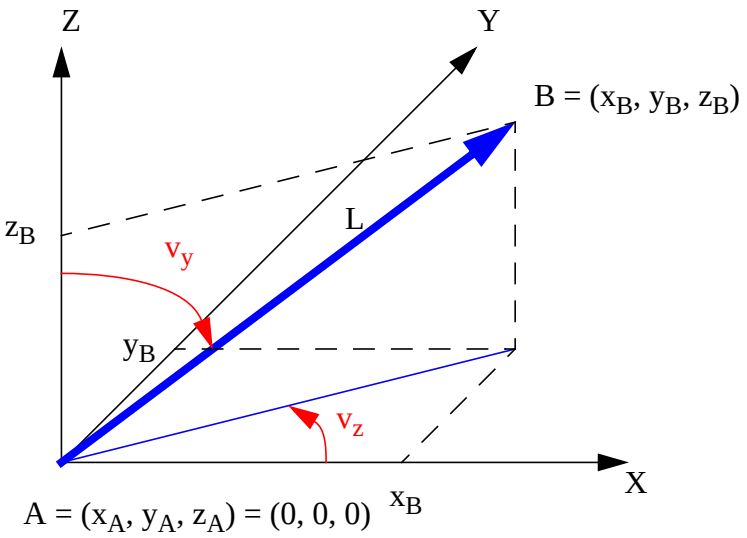
Det vi gör med L_1 nu är att vrida den motsols (d.v.s. mot Y-axeln) med en vinkel v_z som ligger mellan -180 och 180 grader. Detta innebär att vi vrider hela L på samma sätt i det 3-dimensionella rummet. Försök att tänka dig hur detta ser ut. I figuren på nästa sida ser du hur det ser ut. Projicerar man L_1 på X- och Y-axeln får vi fram vektorerna L_x och L_y.

Vektorn L sträcker sig nu från punkten A till en punkt i rummet som vi kallar B (se figuren på nästa sida). B:s koordinater (x_B, y_B, z_B) kan räknas fram utifrån punkten A genom att lägga till de tre vektorerna L_x, L_y och L_z.

Nu till din uppgift. Det är att gå bakvägen mot beskrivningen ovan. Du får alltså koordinaterna för punkterna A och B och skall därifrån räkna ut L, v_y och v_z. Du skall skriva en procedur, Create_Vector_Data, som räknar ut dessa data givet koordinarna för A och B (som kommer in som parametrar). Koordinaterna anges i *centimeter.*

Proceduren skall returnera de tre värdena; längden av vektorn mellan A och B (L, i enheten *meter*), den vinkel i X-Y-planet som vektorn har räknat från x-axeln (v_z) samt den vinkeln mellan z-axeln och själva vektorn (v_y). Vinkeln v_y ligger alltså i det plan som spänns upp av z-axeln och vektorn. Se i figuren nedan för mer visuell bild av vilka data som finns givna respektive vad som skall beräknas av din procedur.

I figuren antar vi för enkelhelts skull att A ligger i origo, men er procedur skall klara av att hantera att A ligger på en annan plats i rummet (vi kan se det som ett ”virtuellt origo”).



Att vi valt namnen v_y respektive v_z på vinklarna i figuren beror på att man kan se dessa vinklar som att de anger hur mycket man skall rotera en vektor kring y- och sedan z-axeln för att man skall komma fram till just denna vektor (beskrivet på föregående sida). Rotationerna görs isåfall motsols runt axeln givet att man står och tittar mot origo från spetsen på den aktuella axeln.

Det är givet att indata till proceduren är heltal och det är meningen att proceduren skall skicka tillbaka vektordata som är avrundade till heltal. Detta leder till att vinklarna skall vara i *grader.*

TIPS 1: Något man måste tänka på är att man skall få ett så bra värde som möjligt på vinklarna. Man skall i princip kunna få fram koordinat B utifrån koordinat A och dessa utdata. Det ger att man kanske inte kan räkna exakt hela tiden.

TIPS 2: Om resultatvektorn får längden 0 (noll) spelar det inte någon roll vilka vinklar proceduren returnerar. Svårt att säga vad som är rätt i detta fall.

Fundering: Varför kan detta inte vara en funktion? Man skickar ju in data och får ut data.

På nästa sida följer ett antal indata med tillhörande resultat (utdata) tillhörande proceduren Create_Vector_Data. Bra att testa dessa innan man redovisar sin laboration. Vissa värden blir fel om man inte ser till att använda det avrundade värdet på L för sina fortsatta beräkningar. Detta beroende på att man annars får värden som inte “hänger ihop” (d.v.s. att man inte kan komma tillbaka till ursprungsvärdena om man räknar “baklänges igen”).

x_A	y_A	z_A	x_B	y_B	z_B		L	v_y	v_z
0	0	0	1000	0	0	=>	10	90	0
0	0	0	0	1000	0	=>	10	90	90
0	0	0	0	0	1000	=>	10	0	0
0	0	0	-1000	0	0	=>	10	90	180
0	0	0	0	-1000	0	=>	10	90	-90
0	0	0	0	0	-1000	=>	10	180	0
0	0	0	1000	1000	1000	=>	17	54 - 55	45
0	0	0	-1000	-1000	1000	=>	17	54 - 55	225
0	0	0	-1000	-1000	-1000	=>	17	125 - 126	225
1000	1000	1000	-1000	-1000	-1000	=>	35	125	225
0	0	0	0	0	0	=>	0	* Egalt *	* Egalt *
500	0	500	0	500	0	=>	9	124 - 125	135
-500	500	-500	500	-500	500	=>	17	54 - 55	-45
0	0	0	1000	-1000	-1000	=>	17	125 - 126	-45
123	234	345	456	567	678	=>	6	55 - 56	45
500	500	500	1500	0	1200	=>	13	57 - 58	-27
127	345	789	-123	1234	-543	=>	16	145 - 146	106

Problem 5, Proceduren Meny

Du skall skriva en procedur som hanterar en meny (om du vill kan detta vara ett underprogram till ett annat huvudprogram och då uppstår funderingen: vilka parametrar skall man då skicka till eller från denna procedur?). Du får givetvis dela upp proceduren i flera delar om detta är lämpligt (det kan det vara i denna uppgift, t.ex. separata underprogram som hanterar inmatning och utskrifter för respektive alternativ i menyn). Du får också använda dig av de underprogram du skapat tidigare.

Denna procedur skall skriva ut vilka alternativ användaren har att välja bland, fråga efter vad han/hon vill göra och sedan utföra detta. Efter utförande skall menyn återigen skrivas ut. I menyn skall det finnas ett alternativ som gör att man kan avsluta.

Uppgift 2

Du skall nu dela upp ditt program i flera olika delar. De delar som skall separeras från de övriga är funktionen *Vector_Length* och proceduren *Create_Vector_Data*. Dessa skall inkluderas i ditt program, men alltså ligga på separata filer (liggande i samma mapp som huvudprogrammet). Om det är fler underprogram som behöver flyttas ut på separata filer får du göra detta också.