TDDE54/TDDD87/TDIU08/725G92/9AMA73: Tips för Ada.O11.1 och lite på O11.2 ...

Torbjörn Jonsson <torbjorn.jonsson@liu.se>

Sat 25/09/2021 16:35

To: **TDDD87_2021HT_FP <tddd87_2021ht_fp@student.liu.se>; TDDE54_2021HT_AU <tdde54_2021ht_au@student.liu.se>; TDIU08_2021HT_C9 <tdiu08_2021ht_c9@student.liu.se>; 725G92_2021HT_Z3 <725g92_2021ht_z3@student.liu.se>**

Hejsan.

Jag har fått ett mail som verkar som att det finns en känsla av att det är ok att få lite tips även på Ada.O11.1 (och kanske något mer på Ada.O11.2). Därav detta mail.

Om ni tycker att dessa mail inte är något att ha så får ni helt enkelt "kasta dem". Jag skriver dem bara för att fylla upp det som vi tyvärr inte har via de normala riktiga LIVE-FÖ som vi brukar ha, men som i år inte har kunnat ges av olika anledningar.

Om ni tycker att det är bra med dessa får ni gärna tala om det för assistenterna (även åt andra hållet förstås om ni inte gillar dem). Det är bra om vi får reda på så att vi inte lägger energi på saker som ingen vill ha.

Nåja. Nu kör vi med lite extra tips för Ada.O11.1 och kanske något om Ada.O11.2 också även om det redan finns ett mail om den uppgiften.

Vad är det vi är ute efter i denna uppgift?

Till att börja med kan man säga att det inte finns mycket roligt man kan göra om man inte har matematik i grunden. Det är massor med små saker som bygger på att man kan matte och detta även i denna kurs. Vi baserar oss däremot på saker som ligger lite längre bak i tiden, d.v.s. saker som tas upp i grundskolan och gymnasiet i första hand. Detta för att det är det enda som faktiskt ingår som "förkunskaper" i denna kurs gällande matematiken. Alla kanske inte har full koll på alla delar i detta ändå, men det är bara att fråga om det är saker som är besvärliga. INGA FRÅGOR ÄR DUMMA, men det kan vara "dumt" att inte fråga sa alltid min far. OBS! Fråga assistenterna alltså. Vi hinner inte ta allt som vanligt.

Vad är det nu som ingår i uppgiften? Till att börja med är det viktigaste i denna uppgift (eller båda dessa uppgifter) att komma i kontakt med standardpaketen "Ada.Numerics", "Ada.Numerics.Elementary_Functions" (för Ada.O11.1) och "Ada.Numerics.Discrete_Random" (för Ada.O11.2). I dessa paket finns en del bra saker som gör att man kan göra lite mer saker än att bara läsa in data och skapa triviala uppgifter.

Vi har dessutom dessa två uppgifter som inkörsport till underprogramdelen (Ada.O2) så att denna skall bli lättare att komma igenom då det varit en stor tröskel tidigare år. I år verkar det som att Ada.O2 går mycket smidigare så det är ju super.

I Ada.O11.1 är det meningen att ni skall se delar av vad som finns i de två första paketen (finns även i boken i bilagorna så att man kan se ännu mer bra saker som finns där). Vi räknar med att vi på P-uppgifterna och tentorna kan använda hela paketens innehåll så kolla lite extra på vad som ingår och prata om hur detta kan läsas för att kunna anropa de funktioner och procedurer

som finns där. I några av kurserna kommer ni också att skapa egna paket, men inte i alla så jag tar inte mer om detta här.

Hur kommer jag igång med uppgiften då?

För att komma igång behöver man titta på FÖ-filmen om underprogram. Det är också bra att diskutera med assistenterna på HA/LA om dessa. Samla ihop ett gäng och ta en genomgång på tavlan i salen under ett LA-pass så lossnar säkert massor.

I uppgifterna är det väldigt hårt styrt exakt vad som skall vara "parametrar" och vilka datatyper dessa skall ha och dessutom om de skall vara "in", "out" eller "in out" (d.v.s. vilken riktning man skickar data från/till huvudprogrammet (detta är ett stort område så det behöver ritas och diskuteras). Inget konstigt med det.

Vi är i rättningen inte så "petiga" gällande om man har "rätt mod ("in", "out", ...) på parametrarna, utan istället ger vi tips om vad det borde vara, men däremot är vi lite mer kinkiga med det som skall vara i huvudprogram respektive vad som skall vara i respektive underprogram.

Tänk att det som står att underprogrammet skall göra är exakt det som skall göras och det är en hjälp till huvudprogrammet som gör vissa delar själv, men behöver dela upp problematiken för att det inte skall bli så mycket att överskåda just där.

Viktigaste delarna är att man räknat med de saker som finns i paketen. Idetta fall "roten ur" (Sqrt-funktionen) lite aritmetiska saker som har med vinklar att göra (funktionerna Sin, Cos, Tan och kanske man skall kolla lite på Arctan och de andra som är "inverserna" till de innan).

OBS! Den största missen som vi sett hittills (som ger komplettering) är i princip att man gör felhantering av inmatningen av vilket val man gör i menyn ute i huvudprogrammet istället för inne i underprogrammet. Det näst största felet tror jag är att man inte låter huvudprogrammet välja vad som skall anropas i nästa steg (detta står i uppgiften nämligen).

Lite mer detaljer kring "paket" och just Ada.O11.1

Viktiga grejor med paket är att det i princip bara står i dessa (paketens specifikationsdel som ni har i bokens bilagor) vilka konstanter, typer och underprogram samt något som heter "exception" (undantag) som finns. Inte hur man använder dem.

Det som förhoppningsvis går att få fram är vad funktioner och procedurer gör via deras namn. Står det t.ex. 'procedure Get(...);' så betyder det att man kan skriva "Get" i huvudprogrammet och då utförs det som står i proceduren "Get" i paketet.

Samma sak om det står 'function Sin(...) return ...;' (vilket alltså ger sinusvärdet av en vinkel). I detta fall måste man anropa funktionen i ett uttryck vilket skiljer sig från procedurerna som anropas som separata satser i ditt huvudprogram (eller ifrån ett underprogram om man vill så).

Ibland står det 'function "+"(...) return ...;' eller liknande och då är det alltså en operator som finns där och denna anropas då precis som man skulle gjort i vilken beräkning som helst (t.ex. A + B).

När du skall skapa dina egna underprogram kommer det att se väldigt likt det som står i paketen, men där det avslutas med ";" efter procedurer, funktioner och operatorer i paketet skall du istället skriva in definitionen av vad ditt underprogram skall göra. Jag hänvisar här vidare till FÖ-filmen om underprogram för att ta detaljerna där och hoppas att ni lyckas bra med detta. Assistenterna tar gärna diskussioner om detta på LA/HA förstås. :-) Gärna i större grupper så att fler kan vara med och på det viset fler kan få hjälp på samma tid. Effektiviteten i detta är mer än bara för assistenterna kan jag tipsa om.

Undantag är något som ni råkar ut för ibland som gör att programmet "kraschar" (eller "dör" som vissa säger). T.ex. kan ni ha råkat ut för "End_Error", "Constraint_Error", "Data_Error", ... Exakt vad dessa gör kommer i en senare uppgift (om den ingår i era kur), men det är bra att veta att dessa alltså finns i paket i Ada och är väl definierade i vad de är till för. Vi räknar inte med att ni skall hantera dessa just nu.

Lite mer om Ada.O11,2 också

Som sagt finns det lite om detta i det specifika mailet om denna uppgift (se där), men något kan sägas kanske.

Vi kan ta detta som är lite udda. Det finns flera olika slumphanteringsmodeller att tillgå. Del en (generell) som är "grunden" till allt skulle man kunna säga. Den som har med att slumpa flyttal att göra. Denna baserar sig i princip alltid på att man slumpar ett tal i intervallet [0.0, 1.0[(d.v.s. 0.0 kan vara med och 1.0 kan inte vara med, men väldigt nära ...).

Av slumptalshanteringen med flyttal kan man sen skapa precis vad man vill gällande slump, men det blir väldigt ofta jobbigt och man måste tänka rätt varje gång så det kommer alltid till en punkt där det blir mer praktiskt att skapa några varianter av detta som hanterar olika fall. Vi bara berör detta på ytan som ni förstår.

Vi fortsätter lite till med flyttalen. Hur kan man slumpa tal som ligger i intervallet [10.0, 11.0[om man har tillgång till den som slumpar tal i intervallet [0.0, 1.0[? Detta tror jag att ni fixar eller hur.

En annan fråga då: Hur fixar man att få fram slumptal i intervallet [0.0, 10.0[om man har den urspungliga? Ni fixar detta också.

Alltså fixar ni det som har med det som står i uppgiften att göra. Eller hur?!

OBS! Vi vill inte ha fler slumptalsgeneratorer än nädvändigt och vi vill ha dem så GLOBALT som möjligt då vi ENDAST vill göra "Reset" EN gång i programmet för EN specifik slumptalsgenerator. Finns problem som uppstår annars.

För att slumpa heltal behöver man skapa en "subtype" som motsvarar den mängd tal man skall slumpa inom. Detta finns det exempel på hur man gör på hemsidan. Kolla där och "kopiera" koden som finns och diskutera sen med assistenten vad som egentligen händer i detta. Vi pratar om att vi skapar en "instans" av en slumptalsgenerator (eller egentligen paketet). Detta är det meningen att ni skall förstås och det ingår absolut att man skall kunna använda "generiska paket" (likt det som har med heltalsslumpningen att göra). Dock inte att kunna skapa dem själv i dessa kurser.

I uppgiften är det återigen viktigt att försöka skapa slumptalsgeneratorn så GLOBALT som

möjligt och därför kommer ni att behöva skapa 2 instanser av heltalsslumptalsgeneratorer då den ena inte går att skapa i huvudprogrammet. Vi vill dock ha en i huvudprogrammet också för att kunna se skillnaden.

Till sist skall man också skapa en teckenslumpning. Denna går till på samma sätt som för heltal så jag låter denna ingå i ovanstående och hoppas att ni löser detta med assistenten om det är något ni undrar.

OBS! Har man programmerat mycket kommer det garanterat att finnas olika åsikter om vad som är bra och dåligt i ovanstående, men tänk på att vi arbetar med detta i en vy där vi vill visa på ett av sätten och det är inte ett dåligt sätt iallafall. Finns saker som till och med skulle kunna vara superbra med att göra saker som bryter mot det vi hittills sagt i kursen, men vi tar inte den dikussionen utan arbetar utifrån dessa premisser så blir det lättare att göra bra kod just nu.

Jag tar gärna en diskussion med dem som har bra sätt att tänka och ger min input på vad som är "bra", "mindre bra", "skapligt" och "dåligt" (och kanske "ännu värre" :-)). Jag skall försöka klämma in ett sådant tillfälle framöver där ni bjuds in. Lovar inget helt, men vill gärna detta.

Ett par heta tips för detta med underprogram och programmering i allmänhet

När man programmerar är det viktigt att titta över helheten så att man vet ungefär vad hela uppgiften är. Efter detta skall man dela upp problemen i mindre delar och lösa dem var för sig (detta blir väldigt ofta underprogram som sköter dessa delar).

När man löst en del. Stanna upp och testa att den fungerar! Gör inte hela programmet på en gång då det bara bli massor av kompileringsfel överallt som beror på varandra och en massa följdfel som man inte kunde förutspå.

Lös det du kan först och skjut på problem som du inte kan. Det kan ju vara saker som du kan göra lite här och där. Gör dessa. Man måste inte skriva programmet från start till slut. Man kan ju lösa ett delproblem först och testa detta.

Var inte rädd för att skriva ett extra litet program som testar om det verkar rätt som du tänkt. När det verkar fungera: Klipp in koden i ditt "riktiga" program.

Var inte rädd för att testa saker. Varje fel man gör lär man sig av. Gäller oavsett vad du håller på med.

Om du vill slippa testa saker som du redan "vet" är ok. Kommentera ut detta temporärt. Det finns "snabbsätt" att kommentera ut hela områden i Emacs. Snabbkommandon till detta är "C-SPACE", flytta markören, "C-c" + ";" (inte "C-;"). Man kan också använda menyerna i Emacs (under Ada finns en del saker). Avkommentering av ett område gör man med samma kommando, men "C-c" + ":" istället för "C-c" + ";". Det finns en lathund för Emacs på hemsidan med fler tips om sådant.

Om man inte vet hur man skall göra en sak (t.ex. hur man gör ett underprogram) kan man börja med att göra allt i huvudprogrammet. Får man det att fungera där kan man sen flytta sakerna till underprogram och göra anropet som behövs i efterhand. Detta är ett väldigt vanligt sätt att arbeta (speciellt som nybörjare).

När man senare kommer på hur super det är med underprogram gör man precis tvärsom och börjar med att bara göra anropen i huvudprogrammet och sen skapar man underprogrammen allteftersom. OBS! Man bör inte glömma bort att kanske kommentera bort anrop som inte det finns underprogram till ännu då det annars ger kompileringsfel som bara är tråkiga.

Avslutning

Dessa två uppgifter är alltså inte lika strikt rättning på och därför kommer ni snabbare att få godkänt på dessa med allra största sannolikhet. Det gör också att vi snabbare kommer framåt och ni får igen det som varit lite tungt i de två första uppgifterna där det varit många saker som varit "petiga". Hoppas att ni kommer att känna detta.

Vi kommer givetvis att påpeka saker som inte är bra, men då som tips för att ni skall komma rätt gällande senare delar och att det nog ingår i slutänden när det gäller P-uppgifterna och tentauppgifterna.

OBS! När ni är klara med Ada.O11.1 är det dags att böra skicka in Ada.O2 också. Den "blockerar" massor av uppgifter, men sen släpper det lös och ni kan "leka" med alla möjliga delar och kan göra saker under tiden ni väntar på rättningen från assistenterna. Ni kommer då inte att behöva känna att det tar så lång tid då ni kommer att ha att göra. Ni kommer dessutom att börja testa er på Ada.P2 så fort ni har Ada.O2 och Ada.O11.2 klara. Det närmar sig. Super!

Hoppas nu att ni har något att börja med i även Ada.O11.1 och Ada.O11.2 och att det går bra för er.

M.v.h.
/TJ
--

- -
_/_/_/_/_/_/_/_/_/ ***Torbjörn Jonsson***
_/ _/ ***013-28 24 67***
_/ _/ ***Torbjorn.Jonsson@LiU.SE***
_/ _/ _/ *IDA/SaS/**UPP***
_/ _/_/ ***Institutionen för Datavetenskap***
- - - - - - - - - - ***Linköpings universitet***